imav2019

11th International Micro Air Vehicle Competition and Conference

September 29-04- October 04, 2019

Madrid, Spain

www.imavs.org

vision & aerial robotics

CAR
CENTRO DE AUTOMÁTICA Y ROBÓTICA

CSIC
Consejo Superior de Investigaciones Científicas

Universidad Politécnica Madrid
POLITÉCNICA

Contents

Preface

It is our pleasure to present the proceedings of the 11^{th} International Micro Air Vehicle Competition and Conference, which was held in Madrid, Spain from September 29-04 till October 4, 2019. This event aims at stimulating and challenging the research in the area of Micro Air Vehicles (MAVs) for real world applications, by both competing in realistic scenarios (indoors and outdoors) and also by presenting the new proposed solutions in conference papers.

This year, the event has a special focus on the growing interest of using flying robots in the area of logistics. We are very grateful that Correos is our platinum sponsor and could bring real-world elements to the competitions. Together with gold sponsors Airbus, Parrot, AFOSR and Aura, and silver sponsor BitCraze, this enabled IMAV to contribute with innovative answers to realistic industrial problems by using small flying robots.

These proceedings contain 28 peer-reviewed scientific papers presented at the IMAV-2019. 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 of the field of Micro Air Vehicles.

Finally, we want to express the hope that the specific nature of the IMAV event (a combination of a scientific conference and a real-world competition) in combination with the open-access nature of the publications, will continue to advance the state-of-the-art in the area of small flying robots for real world challenges.

September 2019

IMAV-2019 Program Committee:

P. Campoy
Universidad Politécnica de Madrid

International IMAV Committee

Abdulghani, Mohamed ..... RMIT, Australia
Campoy, Pascual ..... U.P.M., Spain
Carranza, José M. ..... INAOE, Mexico
Carrio, Adrián ..... U.P.M., Spain
Chen, Ben ..... The Chinese University of Hong Kong
de Croon, Guido ..... TUDelft, The Netherlands
De Plinval, Henry ..... ONERA, France
De Wagter, Christophe ..... TUDelft, The Netherlands
Hattenberger, Gautier ..... ENAC, France
Holzapfel, Florian ..... TUM, Germany
Lacroix, Simon ..... LAAS-CNRS, France
Lai, Shupeng ..... NUS, Singapore
Moormann, Dieter ..... RWTH Aachen, Germany
Morin, Pascal ..... UPMC, France
Moschetta, Jean-Marc ..... ISAE-SUPAERO, France
Puente, Paloma ..... U.P.M., Spain
Reeder, Mark ..... Air Force Institute of Technology, USA
Remes, Bart ..... TUDelft, The Netherlands
Serokhvostov, Sergey ..... MIPT, Russia
Shkarayev, Sergey ..... University of Arizona, USA
Watkins, Simon ..... RMIT University, Australia

IMAV2019 National Organizing Committee

Pascual Campoy ..... General Chair
Beatriz Gatoo ..... General Manager
Paloma Espín ..... General Manager

Sergio Dominguez ..... Conference
Paloma de la Puente ..... Conference

Adrian Carrio ..... Competitions (General)
Hriday Bavle ..... Outdoor competition
Javier Rodriguez ..... Indoor competition
Carlos Sampedro ..... Competitions
Alejandro Rodriguez ..... Competitions
Ramón Suarez ..... Competitions
Lu Liang ..... Competitions

Adrian Alvarez ..... Assistants
Miguel Fernandez ..... Assistants
Paul Espinosa ..... Assistants
Joaquin Fernandez ..... Assistants
Marco Luna ..... Assistants
Rupal Kalra ..... Assistants
Alexander Yunda ..... Assistants
Sadeq Issac ..... Assistants
Pablo Salazar ..... Assistants
Alberto Díez ..... Assistants
Hector Gutierrez ..... Assistants
Marta Centeno ..... Assistants

Sponsors

Platinum Sponsor:

Gold Sponsors:

Parrot

AIRBUS

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
UNITED STATES AIR FORCE

aura

Silver Sponsors:

bitcraze

Organizers:

vision & aerial robotics

CAR
CENTRO DE AUTOMÁTICA Y ROBÓTICA

INDUSTRIALES
ETSII | UPM

POLITÉCNICA

Conference Oral Presentations

System Modelling and Identification

Autonomy of UAVs

Intelligent Navigation I

Design of UAVs

Perception for UAVs

Flight Control

Intelligent Navigation II

Guest Speakers

Urban Air Mobility, a vision by Airbus

Dr. Isabel Del Pozo de Poza Head of UTM – Urban Air Mobility AIRBUS

Isabel del Pozo de Poza has over 10 year experience in the ATM field. She started working at Boeing Research & Technology Europe (BR&TE) participating in common initiatives between NextGen and SESAR, supporting the trajectory based operation concept on both ATM modernization programmes. Isabel holds a PhD addressing the "Assessment of Fairness and Equity in Trajectory Based Air Traffic Management". She joined Airbus Helicopters in 2013 as an Expert and in the field of ATM and Civil Operations and Head of Department for Mission Management. She is leading within Airbus the traffic management initiative to support and enable new operations.

Autonomous Drone Racing

Prof. Davide Scaramuzza, Professor of Robotics and Perception at Dept of Informatics at University of Zurich Dept. of Neuroinformatics at Univ. of Zurich and ETH Zurich

Davide Scaramuzza does research at the intersection of robotics, computer vision, and neuroscience. Specifically he investigates the use of standard and neuromorphic cameras to enable autonomous, agile, navigation of micro drones in search-and-rescue scenarios. He did his PhD in robotics and computer vision at ETH Zurich (with Roland Siegwart) and a postdoc at the University of Pennsylvania (with Vijay Kumar and Kostas Daniilidis). From 2009 to 2012, he led the European project sFly, which introduced the PX4 autopilot and pioneered visual-SLAM-based autonomous navigation of micro drones.

For his research contributions in vision-based navigation with standard and neuromorphic cameras, he was awarded the IEEE Robotics and Automation Society Early Career Award, the SNSF-ERC Starting Grant, a Google Research Award, KUKA, Qualcomm, and Intel awards, the European Young Research Award, the Misha Mahowald Neuromorphic Engineering Award, and several conference paper awards. He coauthored the book "Introduction to Autonomous Mobile Robots" (published by MIT Press; 10,000 copies sold) and more than 100 papers on robotics and perception published in top-ranked journals (TRO, PAMI, IJCV, IJRR) and conferences (RSS, ICRA, CVPR, ICCV).

In 2015, he cofounded a venture, called Zurich-Eye, dedicated to the commercialization of visual-inertial navigation solutions for mobile robots, which later became Facebook-Oculus Switzerland and Oculus' European research hub. He was also the strategic advisor of Dacuda, an ETH spinoff dedicated to inside-out VR solutions, which later became Magic Leap Zurich. Many aspects of his research have been prominently featured in the popular press, such as Discovery Channel, BBC, IEEE Spectrum, MIT Technology Review Magazine.

R&T supporting UAS development. The role of Research Organizations in UAS/RPAS innovation chain

Francisco Muñoz Sanz Head of Department: Center of Aeronautical Innovation and Systems Engineering Subdirectorate of Aeronautical Systems INTA, National Institute of Aerospace Technology. Spain

Aeronautical engineer. Polytechnic University of Madrid. 1980. He joined INTA in 1983, where he has played the following roles:

- 1983-1990 Flight Test Engineer
  - Qualification and validation engineer of Aircraft Performance, Flying Qualities and Flight Control in C101, C212, CN235, T26 programmes.
  - Flying qualities and Flight Control engineer and delegate in the Group of Official Test Centers (OTC) of the EF 2000 Eurofighter Programme. 1987 1993.
- In 1993 he was appointed responsible for the development and project manager of the SIVA UAS. First RPAS development system at INTA and first national project in that size and complexity category.
- Following the above, he has been the head of the workgroup (later Department) in charge of all developments on unmanned systems that INTA has addressed:
  - Surveillance systems: SIVA (300kg), ALO (50kg), MILANO (1000kg).
  - Air target systems: ALBA (25kg), DIANA turbojet (160kg).
  - Other studies and developments of related technologies. Nowadays, he conducts the Aeronautical Innovation and Systems Engineering Center (Centro de Innovación Aeronáutica e Ingeniería de Sistemas) at Aeronautical Systems Subdirectorate, which is coordinating all activities related to RPAS/UAS under development at INTA by endorsing from Project Management up to the Final Integration and Flight Test duties..

The European Regulation Framework for UAS Operations

Juan José Sola Bañasco Head of Remotely Piloted Aircrafts Unit (RPAS/DDRONES) at Agencia Estatal de Seguridad Aérea

Juan José Sola heads the Dept. of Remote Control Piloted Aircraft (RPAS) at AESA State Air Security Agency. He is Aeronautical Engineer from Universidad Politécnica de Madrid, belonging to the State Aeronautical Engineers Corps, and Master in Leadership and Public Management from the Menéndez Pelayo International University. He has been Project Manager for the multinational company SENER and afterwards he developed various activities related to Air Navigation for the ISDEFE consultancy. He has extensive experience in aeronautical auditing with about a hundred inspections as Team Leader. He is currently a member of the ICAO (International Civil Aviation Organization) and participates in national and international working groups, such as the JARUS (Joint Authorities for Rulemaking on Unmanned Systems) initiative. Finally, he has given numerous courses and papers on regulations and supervision in the field of air traffic services and unmanned aircraft.

Papers

Authors

N

O

P

R

S

T

U

V

W

Revision

13853, 2019-10-01 20:38
Created by: C. De Wagter
c.dewagter@tudelft.nl

Forward flight tests of a quadcopter UAV with various spherical body diameters

B. Theys*, J. De Schutter
Robotics Research Group, KU Leuven & Core Lab ROB, Flanders Make

ABSTRACT

This paper presents experimental results on the relation between forward airspeed, pitch angle and power consumption of a quadcopter UAV. The quadcopter consists out of an interchangeable spherical body, four cylindrical arms and small propellers mounted at $1m$ diagonal distance to minimize interference between body and propellers. This simple geometry facilitates results reproduction and comparison with simulation. Two different takeoff masses and four diameters of spherical bodies are tested for their steady-state speed and power for pitch angles up to $-45°$. The steady-state horizontal flight is recorded with on-board sensors at the end of flying long straight lines at a constant pitch angle in wind-still conditions. The best effective lift-to-drag ratio increases for smaller bodies and occurs at higher speeds for increasing mass. Results show that the equivalent frontal surface stays constant for pitch angles further than $-5°$ up to the maximum recorded $-45°$ and increases linearly with the frontal surface of the body.

1 INTRODUCTION

Multicopter UAVs, or 'drones', have become a popular platform for applications such as aerial imaging, mapping and inspection. These applications usually do not require high speeds or ranges but benefit of the vertical takeoff and landing (VTOL) capabilities of the multicopters. Moreover, in many countries, flying beyond visual line of sight (BVLOS) is not permitted yet. Therefore, most of today multicopter designs are optimized for maximum flight time and payload capacity near the hover flight condition. New applications in which the drones will fly BVLOS such as drone deliveries or offshore inspections, require long flight times, high speeds and range. The majority of published research on multicopters focuses on dynamics and control of the vehicles such as Huang [1] and Hoffmann [2] who incorporate the aerodynamics of multicopters by using helicopter theory to improve control when deviating significantly from the hover regime. Sufficient experimental data is required to develop models that can accurately predict the performance and can be used in improved control, design software or trajectory planning. However, there is only little published data for the flight performance of multicopters in forward flight. Schiano [3] performed wind tunnel experiments on a quadcopter to create data for a complete aerodynamic model. However, the experiments were carried out without turning propellers. Neumann et al. [4, 5] determined the relation between attitude and the wind velocity by performing wind tunnel experiments in hover and forward flight conditions to determine the 2D wind direction and speed when tracking hazardous gases. Russell [6] performed wind tunnel tests on five commercially available multicopters with varying geometries to determine forces, moments and power as function of the wind speed, rpm and attitude. Marino [7] performed wind tunnel tests to map the relation between power and the wind velocity vector to later use the multicopter as a flying wind sensor. Prudden [8] performed multiple experiments in wind tunnel conditions to map forward flight behavior. The focus was on the influence of frame geometry variations and the mutual interference between the rotors. For wind tunnel tests, creating a free-floating steady-state regime requires tuning the individual motor rpm to create a zero net force and moment. Next to that, vibrations created by the propulsion disturb these force readings or damage the sensor and for larger of heavier drones the propeller induced flow could significantly influence the wind tunnel flow. Rather than simulating steady-state flight conditions in a wind tunnel, this paper focuses on the forward flight regime with a constant speed and altitude in real outdoor flight in wind-still conditions as presented in figure 1.

Fig. 1: Multicopter with $0.5m$ spherical body in hover in wind-still conditions 10 $minutes$ before sunset.

*Email address: bart.theys@kuleuven.be

2 EXPERIMENT

2.1 Components, configurations, conditions

The geometry of the quadcopter consists out of an interchangeable sphere as body, four cylindrical arms with a diameter $20mm$ and a diagonal distance between the propeller shafts of $1m$. The Graupner 9x5 propellers for which the geometry is thoroughly described in [9], have a diameter of $0.23m$ which is small with respect to the distance between them to minimize the mutual interference and the interference with the body. Figure 2 illustrates these dimensions.

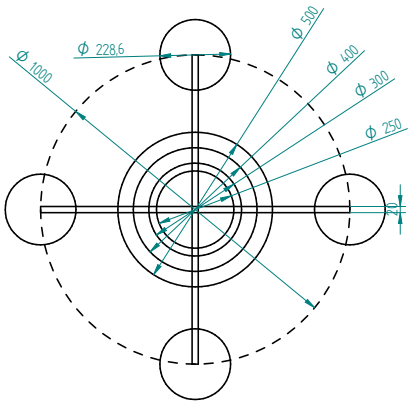


Fig. 2: quadcopter dimensions in mm with four different possible diameters of spherical body.

Figure 3 shows the multicopter fitted with a $40cm$ diameter spherical body during mass check right before takeoff.

Fig. 3: Multicopter fitted with a $40cm$ spherical *Styrofoam* body during mass check right before takeoff.

Table 1 lists an overview of the used components that complete the quadcopter test setup.

component	type	specifications
motor	T-motor 2216-11	900KV
ESC	FVT LittleBee30A	BLHeli Firmware
propeller	Graupner e-prop	9x5
voltage sensor	attopilot 45A	$15.70105\ V/V$
current sensor	attopilot 45A	$27.3224\ A/V$
flight controller	PixHawk	ArduCopter
battery	Zippy Li-Po 6s	$5000mah$, $30C$

Tab. 1: Used components for the test setup.

The four different diameters of hollow *Styrofoam* spheres can be fitted onto the body with cut-outs for the arms. Lead is added along the inner surface of the sphere to bring them all to the same mass. Table 2 presents an overview of the used spherical bodies.

diameter $[m]$	**foam mass** $[kg]$	**extra lead mass** $[kg]$
0.25	0.072	0.248
0.30	0.094	0.226
0.40	0.213	0.107
0.50	0.320	0

Tab. 2: *Styrofoam* spheres with different diameters used for the shape of the body, brought to a total mass of $0.320kg$.

With all bodies at the same mass, an additional lead mass can be added to achieve a total takeoff mass of $2.13kg$ and $2.50kg$ as presented in table 3.

component	**mass** $[kg]$
multicopter w/o bat. & body	1.00
battery	0.81
body	0.32
additional mass	0 - 0.37
total takeoff mass	2.13 - 2.50

Tab. 3: Mass distribution of the quadcopter for two different total masses.

Tests took place at two different days. Table 4 presents the atmospheric conditions at the time of the tests.

data set	**T** $[°C]$	**p** $[hPa]$	$V_{wind}[m/s]$	**humidity**
2.13 kg	23	1025	<1	65%
2.50 kg	28	1017	<1	45%

Tab. 4: Environmental conditions during the experiments. All experiments with the same mass are recorded consecutively without significant changes in conditions.

2.2 Test procedure and data processing

The goal of the test procedure is to record the power consumption and resulting speed in steady-state level flight of the quadcopter as a function of its pitch angle. Speed, endurance, range and payload capacity are fully determined if this relation is known for different total masses [7].

Flights are performed flying up and down one path at constant altitude. The pitch angle is gradually increased after each run up to a maximum pitch angle of $-45°$ and then decreased again for several consecutive runs up to $0°$. The results therefore contain measured cruise flight points in two flight directions so that any small wind speed would have minimum influence on overall accuracy of the result.

Figure 4 presents one run out of the recorded data during a flight. After turning the nose $180°$ with respect to the prior run, the multicopter is pitched $-45°$ and accelerates to a steady-state velocity of $20m/s$.

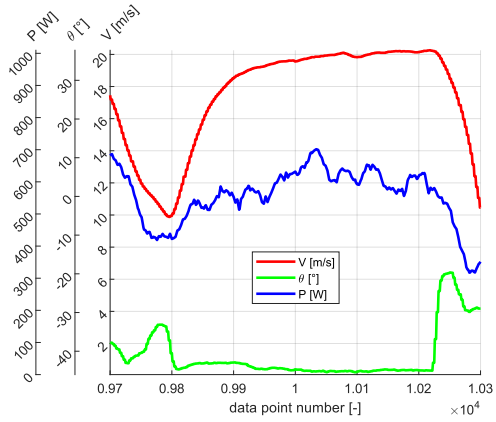


Fig. 4: Illustration of one run out of the recorded data during a flight. The multicopter accelerates to a steady-state velocity of $20m/s$ for a constant pitch angle of $-45°$.

Figure 5 presents the total power consumption and horizontal velocity as a function of the pitch angle for all recorded data during the test flight of the configuration with a total mass of $2.13kg$ and a $40cm$ body. The recorded data points during steady-state regimes at the end of each run are marked in orange. The top graph shows that even in steady-state the measured power fluctuates up to 25% around its mean value presented by the black dots. These variations are due to constant adjustments of the motor rpm for attitude and altitude control. The bottom graph shows there is also some variation of the resulting speed V for one run; there can be difference up to $2m/s$ between two runs with the same pitch angle. This can be explained by a wind speed of about $1m/s$ along the trajectory during the up and down run. Because both directions are flown, the accuracy of the overall experiment will not be influenced. For the further presentation of the results, the mean values for speed and power during the steady-state regimes are used.

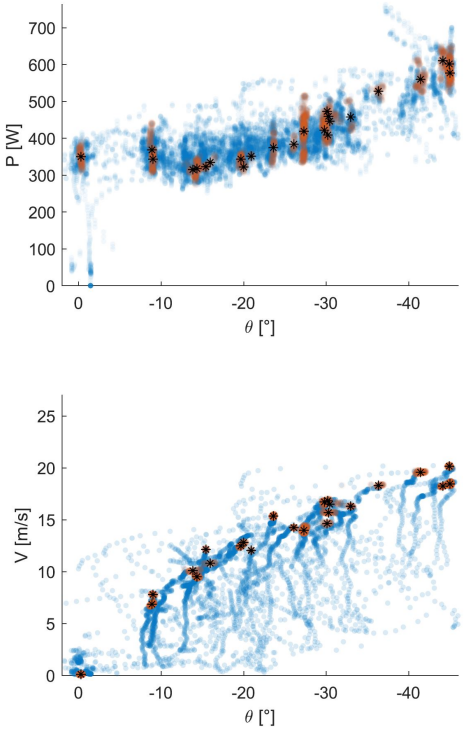


Fig. 5: Total power consumption and horizontal velocity as a function of the pitch angle for the configuration with $2.13kg$ and $40cm$. Blue: all recorded data during the test flight. Orange: recorded data during identified steady-state regime. Black: average values for each steady-state regime

This procedure is applied to all configurations resulting in eight data sets of steady-state horizontal flight power and speed as a function of the pitch angle that varies between $0°$ and $-45°$. Table 5 presents the number of steady-state horizontal flights that are identified per configuration and for which the results are presented in the next section.

	$25cm$	$30cm$	$40cm$	$50cm$
$2.13kg$	14	18	24	17
$2.50kg$	22	21	15	17
$2.93kg$	10	5	-	-

Tab. 5: Zones of steady-state horizontal flight recorded per configuration.

The initially planned total mass of $2.93kg$ was too heavy for the used propulsion for flying at high speeds, therefore only the hover data is used.

3 Results & Discussion

3.1 Resulting speed for increasing pitch angles

Figure 6 shows the cruise speed at pitch angles from hover to $-45°$ for the four different body diameters for the two total masses. Although the individual data points are not clearly separated, clear trends are visible. As expected, the resulting speed increases for decreasing body diameters and higher mass. For a higher total mass, the influence of the body diameter on the speed becomes larger and is more clear on the graph. For the $2.13kg$ experiments there is no noticeable difference in speed between the $25cm$ and $30cm$ sphere.

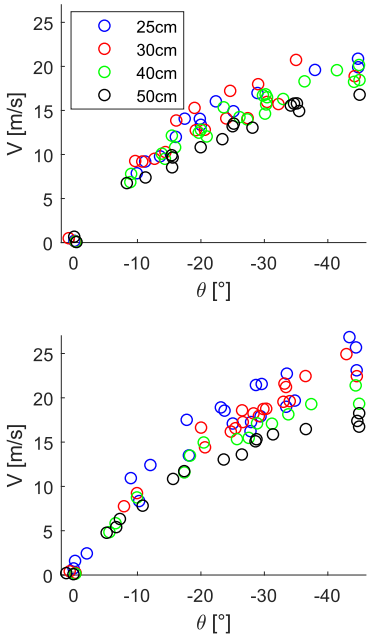


Fig. 6: Cruise speed at pitch angles from hover to $-45°$ for four different body diameters and two total masses. Top: 2.13kg. Bottom: 2.50kg.

To visualize the influence of body diameter and mass on the top speed, figure 7 shows the average top speed at $-45°$ pitch as a function of the body diameter for the two total masses. The top speed decreases for increasing body diameters and increases for a higher total mass. The difference in top speed for $2.13kg$ and $2.50kg$ of total mass is small for high body diameters and increases for decreasing body diameters. Although top speed for a given maximum pitch angle increases with mass, this is only possible if the propulsion system is capable of maintaining altitude at this high pitch angle and mass.

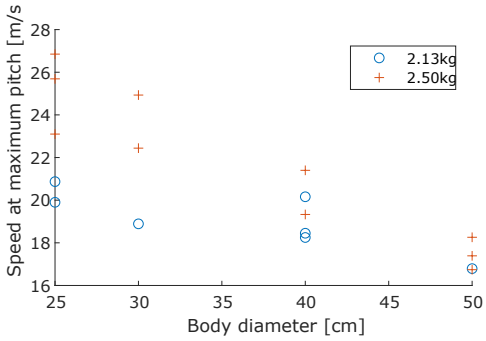


Fig. 7: Measured top speeds at $-45°$ as a function of the body diameter for two masses.

3.2 Power consumption in forward flight

Figure 8 shows the cruise power for pitch angles from hover to $-45°$ (left) or for speeds from hover to maximum speed (right) for four different body diameters and the two total masses. The required power slightly decreases at increasing speeds from hover which is in line with helicopter theory [10]. After reaching a minimum, the power increases significantly. The speeds at which minimum power is achieved lie higher for $2.50kg$ than for $2.13kg$ and the decrease of power with respect to hover is more pronounced for large body diameters.

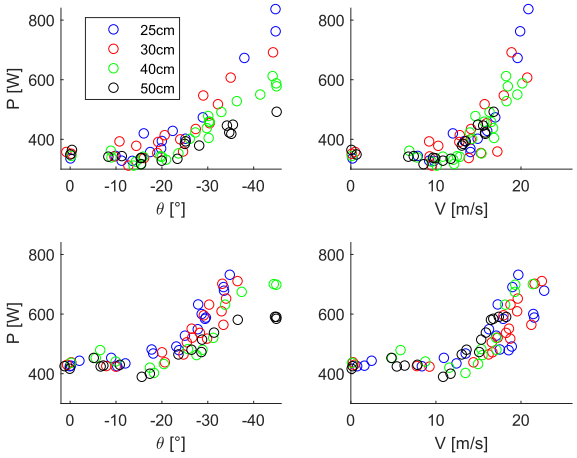


Fig. 8: Power as function of pitch θ and speed V for four different body diameters and two total masses. Top: $2.13kg$. Bottom: $2.50kg$.

3.3 Effective lift-to-drag ratio

The efficiency with which aircraft can move through the air can be expressed with the glide ratio, also known as the aerodynamic efficiency:

$$\frac{L}{D} = \frac{m_{tot}\, g}{D} = \frac{m_{tot}\, g\, V}{D\, V}[-], \tag{1}$$

in which $D\ V[W]$ presents the required power to fly. For a glider, this power can be directly calculated as the loss of potential energy. In other cases, an effective lift-to-drag ratio can be used, defined as:

$$\frac{eL}{D} = \eta_{prop}\ \frac{L}{D}\ = \frac{m_{tot}\ g\ V}{P}[-] \tag{2}$$

With P the required power from the energy source and η_{prop} the total efficiency of the propulsion system. For a multicopter this is the combined efficiency of the ESCs, motors and propellers.

Figure 9 shows the calculated effective lift-to-drag ratio between hover and the maximum speed at $-45°$ pitch for four different body diameters and two total masses. The maximum effective lift-to-drag ratios lie between 0.70 and 0.95. The speed at which the highest effective lift-to-drag ratio is achieved is higher for the experiments with $2.50kg$ compared to the experiments with $2.13kg$. The $2.50kg$ data set shows a trend of increasing maximum effective lift-to-drag ratios and the speeds at which they occur for decreasing body diameters. For the $2.13kg$ data set, this trend is less visible and between 12 to $13m/s$ the $40cm$ and $50cm$ bodies even seem to have a small advantage. This means the multicopter configurations with larger bodies consumed less power flying at this speed compared to the smaller diameters. This can be caused because less power is required from the propulsion system due to a decrease in drag, additional production of lift or an increase in efficiency of the propulsion system in this flight regime. Because spherical bodies of increasing diameter are used, the latter is the most likely explanation. As presented in figure 6 the pitch angle of the $25cm$ and $30cm$ body is approximately $-17°$ pitch for both and for the larger $40cm$ and $50cm$ body diameter approximately $-22°$ and $-25°$ pitch respectively which suggests that the propulsion system used in this paper is more efficient in these flight regimes.

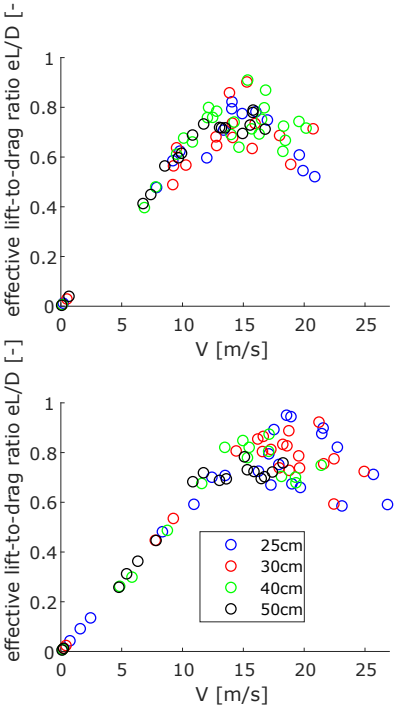


Fig. 9: Calculated effective lift-to-drag ratio at pitch angles from hover to $-45°$ for four different body diameters and two total masses. Top: $2.13kg$. Bottom: $2.50kg$.

3.4 Equivalent frontal surface

As it is common practice in helicopter performance identification [10], the drag of the body can be represented with an equivalent frontal surface area A_{eq} with $C_D = 1$:

$$D = \frac{1}{2}\ V^2\ \rho\ A_{front}\ C_D = \frac{1}{2}\ V^2\ \rho\ A_{eq}\ [N] \tag{3}$$

With V the speed and ρ the air density. For a multicopter without lifting surfaces and flying at a constant low speed, the forces that apply are the weight, drag and the force of the propellers. The latter one assumed to be along the shaft F_x. These forces are schematically presented in Figure 10.

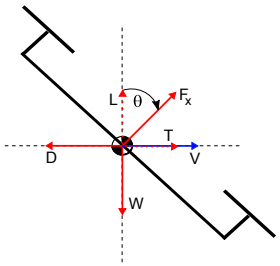


Fig. 10: Forces on a VTOL UAV without lifting surfaces at constant speed and altitude.

For this case, we can write the equivalent frontal surface as:

$$A_{eq} = \frac{2\ m_{tot}\ g\ \tan(-\theta)}{\rho\ V^2}\ [m^2] \tag{4}$$

Figure 11 shows the equivalent frontal surface at pitch angles from hover to $-45°$ for four different body diameters and two total masses. The equivalent frontal surfaces for data points between hover and $-5°$ pitch are not calculated because of the low value for speed in the denominator of equation 4. Although the data are scattered, different body diameters are clearly distinguishable in the data with a higher

body diameter resulting in a higher equivalent frontal surface. Between $-5°$ and $-15°$ the equivalent frontal surface shows a decreasing trend. Between $-15°$ and $-45°$ no clear trend is visible and the equivalent frontal surface could be approximated by a constant equal to the mean calculated value, presented by the horizontal lines.

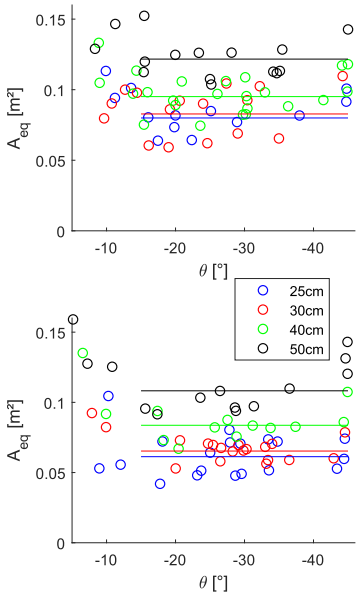


Fig. 11: Calculated equivalent frontal surface at pitch angles from hover to $-45°$ for four different body diameters and two total masses. The horizontal lines present the mean value for pitch angles from $-15°$ to $-45°$.

Figure 12 shows these mean equivalent frontal surface for angles between $-5°$ and $-45°$ as a function of the frontal surface of the spherical bodies for comparison.

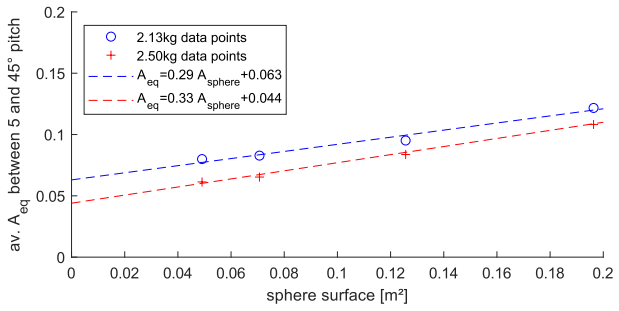


Fig. 12: Average calculated equivalent frontal surface for pitch angles between $-5°$ and $-45°$ for four different frontal surfaces of the spherical body and two total masses.

For the $2.13kg$ and $2.50kg$ data, a linear fit with slope 0.29 and 0.33 is found respectively. This lies within the expected drag coefficient of spherical bodies which, depending on the Reynolds number and surface quality lie between 0.1 and 0.5 [11]. Since the drag of the multicopter is not only due to the spherical body, there is an offset which takes into account the drag of the arms and motors with propellers. This offset is respectively $0.063m^2$ and $0.044m^2$ which would be the theoretical equivalent frontal surface of only the arms and propellers at -45 pitch.

3.5 Hover efficiency

As an example on how these data can be used for model validation, for the unique case of hover, the momentum theory as described by Rankine - Froude [12] and also known as the Actuator Disk Theory, can be used to predict the power as a function of the total mass m_{tot} and the total disk area covered by the propellers A_{disk}. The power required from the battery P_{batt} can be calculated with this basic model as:

$$P_{batt} = \frac{(m_{tot}\ g)^{1.5}}{\sqrt{2\ \rho\ A_{disk}}} \frac{1}{\eta_{esc}\ \eta_{mot}\ \eta_{plr}}\ [W] \qquad (5)$$

With ρ the density of the air, g the gravity constant and $\eta_{esc}\ \eta_{mot}\ \eta_{plr}$ the efficiency of the ESC, motor and propeller respectively. The efficiency of the propeller is also referred to as the Figure of Merit [10].

Figure 13 shows the average of the measured hover power for three different masses. Next to $2.13kg$ and $2.50kg$ for the forward flight tests, a higher total mass of $2.93kg$ is tested in hover and added to this graph. A combined total efficiency η_{tot} of 44.3% for $\eta_{esc}\ \eta_{mot}\ \eta_{plr}$ resulted in a good fit.

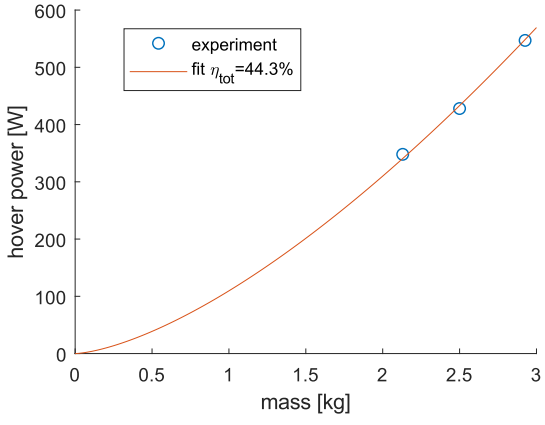


Fig. 13: Hover power and fit based on momentum theory with the assumption of a constant efficiency from battery power to kinetic energy of the accelerated air.

4 CONCLUSION AND FUTURE WORK

Steady-state horizontal velocity and power of a quadcopter with a basic geometry and four interchangeable bodies of different diameters was tested at two total masses and pitch angles from $0°$ to $-45°$. Although experiments took place in wind-still conditions, there is significant variation in the measured data. However, clears trends were visible and the influence of different masses and shapes could be clearly identified. Results show that for a constant pitch angle, increased mass or smaller body diameter resulted in an increased speed. A maximum speed of approximately $27m/s$ was achieved for a $2.50kg$ total mass and $25cm$ body diameter at $-45°$ pitch. For every configuration, the minimum power did not occur during hover. A decrease in power is observed from hover to forward flight. The speed for minimum power increased for a higher mass. The difference between hover power and minimum power was more pronounced for increasing body diameters. The maximum effective lift-to-drag ratios for all configurations ranged between 0.70 and 0.95. The latter occurred at the heaviest configuration with the smallest body diameter. The equivalent frontal surface showed to be rather constant for each configuration for pitch angles between $-15°$ and $-45°$. With momentum theory clearly matching the results for hover, a total hover efficiency of 44.3% was found for the quadcopter in these experiments.

The data set obtained during the tests for this paper can be used as a reference for modeling the flight behavior of multicopter UAVs in forward flight. Trends are observed in the data but for further validation, more test data of different multicopters is required. Additional sensors can be added to monitor the individual states of each propulsion unit and measure rpm, voltage, current to allow also model validation of the propulsion system.

ACKNOWLEDGEMENTS

This research was funded by VLAIO grant HBC.2017.0198

REFERENCES

[1] Haomiao Huang, Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. *International Conference on Robotics and Automation*, 2009.

[2] Gabriel M Hoffmann, Haomiao Huang, Steven L. Waslander, and Claire J. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, volume 2, 2007.

[3] Schiano F. et al. Towards estimation and correction of wind effects on a quadrotor uav. *IMAV*, 2014.

[4] P. Neumann and M. Bartholmai. Real-time wind estimation on a micro unmanned aerial vehicle using its inertial measurement unit. *Sens. Actuators, A 235, 300310*, 2015.

[5] Neumann P. et al. Micro-drone for the characterization and self-optimizing search of hazardous gaseous substance sources: a new approach to determine wind speed and direction. *ROSE 2010-2010 IEEE International Workshop on Robotic and Sensors Environments, Proceedings, pp. 16*, 2010.

[6] Russel C. et al. Wind tunnel and hover performance test results for multicopter uas vehicles. *AHS 72nd Annual Forum*, 2016.

[7] 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.

[8] Prudden S. et al. An investigation into the effects rotor wake interference on quadrotor uas forward flight performance. *18th Australian Aerospace Congress, Melbourne*, 2018.

[9] B. Theys, G. Dimitriadis, P. Hendrick, and J. De Schutter. Experimental and numerical study of mini-uav propeller performance in oblique flow. *AIAA Journal of Aircraft V.54-3 p1076-1084*, 2017.

[10] Raymond W. Prouty. *Helicopter performance, stability, and control*. 1995.

[11] S. F. Hoerner. *Fluid-Dynamic Drag*, page 455. 1965.

[12] W. Froude. On the elementary relation between pitch, slip, and propulsive efficiency. Technical Report 19930080719, National Advisory Committee for Aeronautics, Washington, DC, United States, 1920.

Optimal Flight Altitude for the Small Solar-Powered Airplane

Sergey V. Serokhvostov[1]*, and T.E. Churkina[2]
[1]Moscow Institute of Physics and Technology
9, Institutsky per., 141701, Dolgoprudny, Moscow Region, Russia
[2]Moscow Aviation Institute (National Research University)
4, Volokolamskoye shosse, 125993, Moscow, Russia

ABSTRACT

The problem of optimal flight altitude for the small airplane with the electrical powerplant and solar cells in the long-time flight is investigated for the task of maximization of accumulator energy at the end of the flight. Atmosphere density models and solar radiation models as functions of altitude and day time are formed and investigated. Optimal altitude for the fixed-altitude mission and altitude as function of time for non-fixed altitude mission are obtained. Sensitivity of the optimized function to the deviation of flight parameters from the optimal ones is investigated. The influence of airplane parameters on the results is analyzed.

1 INTRODUCTION

By this time a set of "heavy" solar-powered (SP) airplanes was designed and built (such as "Pathfinder", "Centurion", "Helios", "Solar Impuls", "SoLong", "Zephyr" and others). Also, information about a set of small-sized solar planes can be found (some of them [1]–[5] You can see in Figures 1–5).

Figure 1: Aircraft [1]

Figure 2: Aircraft [2]

Figure 3: Aircraft [3]

But for the present state of the art the design of the SP airplane for the long-endurance mission is still a serious problem because of the moderate value of the solar radiation intensity, rather low efficiency and rather high density of the solar cells, insufficient energy density of the onboard energy storage and some other factors.

Some questions concerning the optimal multi-day flight were investigated in [6, 7, 8]. First of all, the optimal altitude for the flight at constant altitude of the "heavy" airplane at high altitudes was found. But in these investigations rather simple model of solar radiation was used and the analysis was restricted by the altitudes higher than 11 km.

In the present research all the altitudes from 0 km to 20 km for the small airplane and more precise density and radiation models are investigated.

*Email address: serokhvostov@phystech.edu

Figure 4: Aircraft [4]

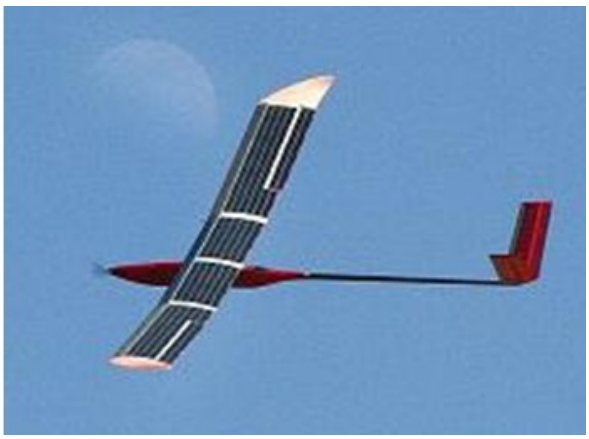


Figure 5: Aircraft [5]

2 OPTIMAL FLIGHT ALTITUDE. PROBLEM STATEMENT

Suppose that the airplane must fly at constant altitude (that can be chosen) with constant velocity V for all its long-time mission with the minimum value of the total energy E consumed from accumulator. For this case the equations of motion are

$$W\eta - (C_{D0} + AC_L^2)\rho\frac{V^3}{2}S = 0 \tag{1}$$

$$C_L\rho\frac{V^2}{2}S - mg = 0 \tag{2}$$

$$\dot{E} = W - W_{PV} \tag{3}$$

where m is an aircraft mass, W — power consumed by the engine, η — powerplant efficiency (assumed to be constant), C_{D0} — drag coefficient at zero lift, $A = 1/(\pi\lambda)$, λ — aspect ratio of the wing, C_L — lift coefficient, $\rho = \rho(h)$ — air density depending on flight altitude h, S — wing area, g — acceleration of gravity, W_{PV} — power of solar cells. For this problem all these characteristics except W_{PV} and E are assumed to be constant during the flight.

Power W_{PV} depends on the flight altitude, day time, aircraft orientation and the sun location. But for the first step of our investigations we can assume that W_{PV} is mean value and depends only on altitude:

$$W_{PV} = I(h)S\beta$$

where I is a part of solar radiation intensity normal to the solar panel, β is the ratio of the solar cells area to the wing area. So, from the mathematical point of view, we can find the minimum of $(W - I(h)S\beta)$ (3) at conditions (1)–(2).

The solution of this problem for the common case gives

$$\begin{cases} C_L = \sqrt{\frac{3C_{D0}}{A}} \\ V = -\sqrt{\frac{3}{AC_{D0}}}\frac{\eta\rho}{2mg\frac{\partial\rho}{\partial h}}\frac{\partial I}{\partial h}S\beta \\ C_{D0}V^3S\frac{\partial\rho}{\partial h} = -\frac{\partial I}{\partial h}\eta S\beta \end{cases} \tag{4}$$

One can see that for the solution one needs to know the atmosphere properties — the dependence of air density and solar radiation intensity as functions of altitude and day time.

3 ATMOSPHERE MODEL

For the atmosphere density it is worth using International Standard Atmosphere (ISA) model. Within this model, the atmosphere's temperature decreases linearly with altitude from 0 km to about 10 km altitude, then from about 11 km to 20 km the atmosphere is isothermal. So, according to these data, the expression for atmosphere density for 0–10 km can be derived and written as

$$\rho = \rho_0\left(1 - \frac{\alpha h}{T_0}\right)^{\mu g/(\alpha R)-1}$$

where ρ_0 — density at sea level (1.225 kg/m^3), $\alpha \approx 6.5 \cdot 10^{-3}$ K/m), T_0 — tempreture at sea level (288 km), μ — molar mass of the air (0.029 kg/mole), $R = 8.31$ Joule/(mole·K). For these data

$$\rho = \rho_0\left(1 - \frac{\alpha h}{T_0}\right)^{4.38}$$

For the isothermal part of the atmosphere the formula for air density can be written as

$$\rho_0 = \rho_{01}\exp(-(h - 11\,000)/h_0)$$

where $h_0 = 6\,374$ m, ρ_{01} is the air density at 11 000 m altitude.

Analysis [9] shows that the intensity of the solar radiation can be expressed as the multiplication of two functions one of them depends only on the altitude and another depends only on the day time.

As for the dependency of solar intensity on the altitude, let's make some assumptions. First of all, assume that the

composition of atmosphere (percentage of the gases) is the same at all altitudes. Second, assume that there is no pollutions that can absorb the solar light. So, one can use the Bouguer's law written in the form

$$\frac{dI}{dh} = B\rho(h)I \tag{5}$$

where B — coefficient of proportionality that depends on the atmosphere composition. Assume that the composition is constant at all the altitudes, so B is constant. The analysis of the data from [9] proves this assumption at least for the altitudes of 11–20 km. The value of B found by authors on the basis of [9] is $B = 5.7 \cdot 10^{-5}$ m^2/kg.

For low atmosphere, the above formula becomes

$$\frac{dI}{dh} = IB\rho_0 \left(1 - \frac{\alpha h}{T_0}\right)^{4.38}$$

which gives

$$I = I_0 \exp\left(\frac{B\rho_0 T_0}{5.38\alpha}\left[1 - \left(1 - \frac{\alpha h}{T_0}\right)^{5.38}\right]\right)$$

This formula can be used for the numerical investigation but rather difficult for the analytical ones.

To simplify the formula it is useful to analyze the experimental data for the dependency of solar radiation on the altitude. It is known that above the atmosphere the solar intensity is ~ 1.3 kW/m^2, and at the sea level it is about 1 kW/m^2. On the other hand, the air density at sea level and at the altitude of 10 km differs by nearly 3 times. So, the main effect of different absorption at different altitudes is due to the air density. The idea is to assume that the intensity does not differ significantly, and

$$\frac{dI}{dh} = B\rho_0 I_0 \left(1 - \frac{\alpha h}{T_0}\right)^{4.38}$$

So

$$I = I_0 \left(\frac{B\rho_0 T_0}{5.38\alpha}\left[1 - \left(1 - \frac{\alpha h}{T_0}\right)^{5.38}\right] + 1\right) \tag{6}$$

The ability to use all above mentioned assumptions must be proved by the comparison of precise solution with the approximate one.

For the isothermal part of the atmosphere

$$\frac{dI}{dh} = B\rho_{01} \exp\left(-\frac{h}{h_0}\right) I$$

and

$$I = I_{01} \exp\left[h_0 B\rho_{01}\left(1 - \exp\left(-\frac{h}{h_0}\right)\right)\right]$$

where I_{01} is the solar intensity at 11 km.

The simplification gives

$$\frac{dI}{dh} = B\rho(h)I_{01}$$

and

$$I = I_{01}\left(1 + h_0 B\rho_{01}(1 - \exp\left(-\frac{h}{h_0}\right))\right) \tag{7}$$

The dependency of solar intensity on the day time at fixed altitude for the first steps of analysis (assuming that day is 12 hours and the night is 12 hours) can be expressed [6] as constant zero function for the night time and the sine function with the zero moment at the start of sunrise with the period of 24 hours for the day time.

4 OPTIMAL ALTITUDE PROBLEM. SOLUTION AND ANALYSIS

System (4) under assumption (5) can be resolved for the variable h in the common case as

$$\frac{(\rho(h))^5 (I(h))^2}{\left(\frac{\partial\rho(h)}{\partial h}\right)^2} B^2 = \left(\frac{2mg}{S}\right)^3 \frac{1}{(\eta\beta)^2}\sqrt{\frac{A^3 C_{D0}}{27}} \tag{8}$$

For the low atmosphere it gives

$$\left(1 - \frac{\alpha h}{T_0}\right)\left(I_0 \exp\left(\frac{B\rho_0 T_0}{5.38\alpha}\left[1 - \left(1 - \frac{\alpha h}{T_0}\right)^{5.38}\right]\right)\right)^2 =$$
$$= \left(\frac{2mg}{S\rho_0}\right)^3 \frac{1}{(\eta\beta B)^2}\sqrt{\frac{A^3 C_{D0}}{27}}\left(4.38\frac{\alpha}{T_0}\right)^2 \tag{9}$$

For the isothermal part of atmosphere

$$\exp\left(-\frac{3h}{h_0}\right)\left(I_{01}\exp\left[h_0 B\rho_{01}\left(1 - \exp\left(-\frac{h}{h_0}\right)\right)\right]\right)^2 =$$
$$= \left(\frac{2mg}{S\rho_0}\right)^3 \frac{1}{(\eta\beta B h_0)^2}\sqrt{\frac{A^3 C_{D0}}{27}} \tag{10}$$

Equations (9), (10) can't be solved analytically.

To solve the problem numerically some data about aircraft are required. Let's use the data as for "Cirrus" glider upgraded with solar cells by authors (see Fig. 6). Assume that the maximum value of solar radiation at the zero altitude is 825 W/m^2.

The main data for the airplane are $m = 2$ kg, $C_{D0} = 0.02$, aspect ratio $\lambda = 15$, $S = 0.72$ m^2, area covered by cells is $S/2$ (i.e. $\beta = 0.5$), powplant efficiency — 50%, solar cell efficiency — 20%.

The flight altitude as function of solar intensity is given in the Fig. 7.

One can see that the graphs for the low atmosphere and isothermal models do not cross at the altitudes of 10–11 km as can be expected. This can be explained by the fact that at the altitudes between 10 km and 11 km the atmosphere changes the properties, the graph for low atmosphere is valid

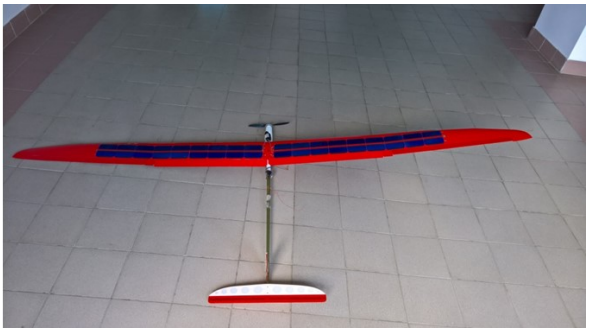


Figure 6: Glider "Cirrus" with solar panels.

for $h < 10$ km, graph for isothermal atmosphere is valid for $h > 11$ km, and from 10 to 11 km these graphs must be interpolated by continuous and smooth line from one graph to another.

To solve the problem of optimal altitude for 24h mission one must find the "mean" value of I for 24 hours. Using the sine dependency of intensity on the day time, one can find that

$$I_{\text{mean}} = I_{\max} \cdot 0.318$$

For these data the flight altitude calculated using the formula for low atmosphere gives the value of 9 812 m, the calculation with the formula for the isothermal atmosphere gives 8 907m. But it should be mentioned that these formulas are for the equator conditions. For the $45°$ latitude (for example, Torino conditions) the low atmosphere formula gives 7854 m, isothermal formula gives 7 107 m. The both cases give the result corresponding to the low atmosphere for the both models, the conclusion is that the low atmosphere model must be used for the precise results.

Now let's estimate the disadvantage for the flight at not-optimal altitude. Figure 8 shows the mean electrical power incoming to the accumulator as function of altitude for the case of $45°$ latitude conditions for the airplane analyzed.

One can see that the disadvantage at the altitude difference of 1 km with respect to the optimal one is about 0.2% of the accumulator incoming power and practically the same value of 0.2% with respect to the power consumed by the motor. This means that it is not necessary to maintain the flight altitude precisely for the best value of accumulator incoming power. Also, if necessary, the model of isothermal atmosphere can be used in low altitudes as the maximal difference between these two graphs at $h < 10$ km is less than 1 km.

Now let's investigate the accuracy of the simplified formula (6). Figure 9 shows the results of calculations.

The deviations does not exceed 1 km, so the more simple formula can be used.

As for the accuracy of formula (7), the calculations show that the results with use of formula (7) practically coincide with precise ones.

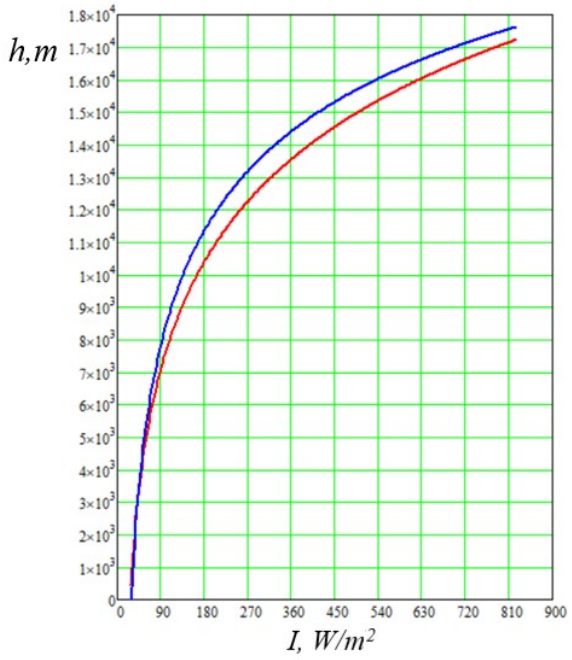


Figure 7: Altitude as function of solar intensity. Blue line — low atmosphere model (9), red line — isothermal model (10).

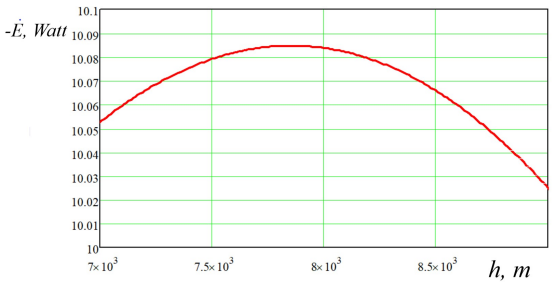


Figure 8: Accumulator incoming power as function of altitude near the maximum.

Some words should be told about the formula (8) itself. One can see that left part of this formula deals only with "atmospheric variables" and the right part deals only with aircraft parameters. For any shape of density and intensity functions it is possible to analyze the sensitivity of the result to the aircraft characteristics.

First of all, one can see that the main influence gives the wing loading (mg/S), as it is in the third power. Also the powerplant efficiency and cells area to wing area ratio can influence significantly. As for the drag coefficient at zero lift C_{D0}, its influence is small enough.

5 FLIGHT PATH OPTIMIZATION

For a set of flight tasks the flight altitude can be variable. In this case the aircraft can change the altitude during the day to gather the maximum of the energy available. The back-

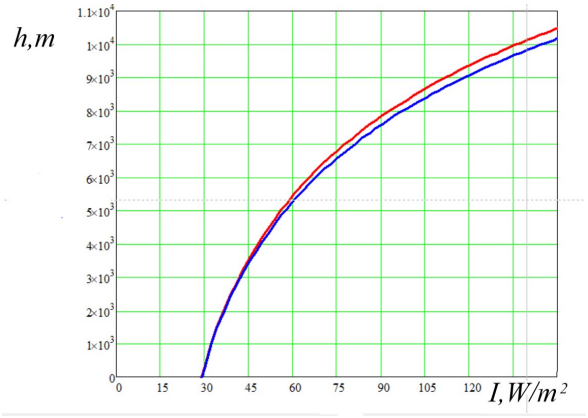


Figure 9: Results of calculation for various models. Red — precise, blue — model (6).

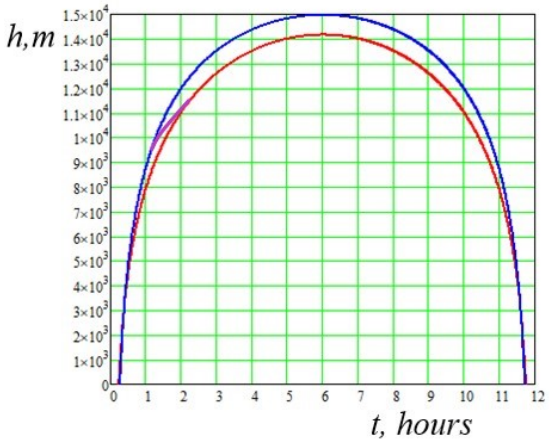


Figure 10: Altitude as function of time for the airplane and conditions investigated. Blue line — low atmosphere model (9), red line — isothermal model (10), purple — interpolation between the models.

ground of this effect is that at lower altitudes the power consumption by the powerplant is lower and at higher altitudes the solar radiation is higher. So, at the day time it is better to fly higher, at the night time it is better to fly lower. This task in general case was analyzed previously in [6]. So, it is better to shortly repeat the problem statement and the result in common case and implement them to the problem investigated here.

The problem solved is as follows: the airplane with solar panels can fly at non-fixed altitude. There are no restrictions on accumulator power (charging and discharging) and capacity, vertical velocity of the plane, maximal accelerations along all the axes. There is minimal altitude restriction (Earth's surface or some others). Only the motion in vertical plane is analyzed. The goal is to have maximal available energy in the accumulators at the end of the flight. For this problem the possible parts of the trajectory are available

1. Maximal power mode.
2. Minimal power mode.
3. Flight along the restriction(s).
4. Cruise mode.

The main part of flight is cruise mode. The solution shows that the altitude, velocity and lift coefficient are defined by the same formulas as for the task of optimal altitude, but for the solar intensity value, not the mean but the values of intensity at the each moment of time must be used. As an example, the altitude as the function of time for the above mentioned airplane and equator conditions for the day time is shown in Fig. 10.

Minimal and maximal power modes are used at the beginning (to reach cruise mode from initial conditions) and at the end of flight (to reach final conditions).

One can see that the altitude difference during the day time is high enough in both models and is about 14 km. The most intensive altitude change (and, consequently, vertical velocity) is at the time near sunrise and sunset. These parts require more thorough investigation (it is planned in the future work).

Also, as in the previous chapter, the sensitivity of the deviations is low enough, so the altitude error of 1 km from the optimal one does not give valuable disadvantage. So, for the preliminary analysis one can use only one model in all the range of altitudes.

At the end it is worth saying that all the results obtained can be implemented not only to the flight in the atmosphere of the Earth, but also for the flight in the atmosphere of any planet which atmospere model can be described by the model mentioned above.

6 Conclusion

1. The problem of optimal altitude for the small solar-powered airplane was stated and analyzed. It was shown that the mathematical models of the air density and solar radiation as function of time are necessary for this task.

2. Models of density and solar radiation were proposed. Some simplifications in the expressions were proposed and analyzed for the applicability and accuracy.

3. Formula for the defining the optimal flight altitude was obtained in the common case. Formulas for the cases of low atmosphere and isothermal atmosphere were derived.

4. The optimal fixed altitudes were obtained for some test airplane.

5. For the case of non-fixed altitude the optimal flight altitude as a function of time was found numerically for the test airplane.

6. The influence of airplane parameters on the optimal altitude was investigated. It was shown that the main influence gives the wing loading and the smallest influence gives drag coefficient at zero lift.

7. Sensitivity of optimized functions to the altitude deviations was investigated. It was shown that the altitude deviation from optimum up to 1 km gives practically no disadvantage.

REFERENCES

[1] https://arduinics.blogspot.com/2019/02/diy-solar-powered-rc-plane.html. Last visit 01.05.2019

[2] https://www.umt.edu.pk/News/Pakistans-first-Solar-Airplane-ready-to-fly-in-the-air.aspx. Last visit 01.05.2019

[3] https://www.flitetest.com/articles/simple-solar-rc-plane. Last visit 01.05.2019

[4] https://www.flitetest.com/articles/solar-wonder-electric-flyier-without-batteries. Last visit 01.05.2019

[5] http://www.sky-sailor.ethz.ch/. Last visit 01.05.2019

[6] S.V. Serokhvostov and T.E. Churkina. Optimal Control for the Sun-Powered Airplane in a Multi-Day Mission. *Scientific Proceedings of Riga Technical University, Series 6 "Transport and Engineering. Transport. Aviation Transport"* 27:212–220, 2008.

[7] S. Serokhvostov. Some Results of Electrical Aircraft Flight Path Optimization With the Help of Pontryagin Maximum Principle and Their Implementation to Design Optimization. In *1st International Workshop Extremal and Record-Breaking Flights of the UAVs and the Aircraft with Electrical Power Plant ERBA-2013*, 2013.

[8] S. Serokhvostov and T. Churkina. Optimization of motor for the maximization of solar-powered aircraft performance. In *31 Congress of ICAS*, 2018.

[9] G. Romeo, G. Frula and L. Fattore. HELIPLAT. A Solar-Powered HALE-UAV for Telecommunication Applications. Design and Parametric Results. Analysis, Manufacturing and Testing of Advanced Composite Structures. In *International Technical Conference on "UNINHABITED AERIAL VEHICLES UAV 2000"*, 2000.

Actuator Modeling for Attitude Control Using Incremental Nonlinear Dynamic Inversion

F. Binz*, D. Moormann
Institute of Flight System Dynamics, RWTH Aachen University, Wüllnerstr. 7, 52062 Aachen

ABSTRACT

Recently, the concept of Incremental Nonlinear Dynamic Inversion (INDI) has seen an increasing adoption as an attitude control method for a variety of aircraft configurations. The reasons for this are good stability and robustness properties, moderate computation requirements and low requirements on modeling fidelity. While previous work [1] investigated the robust stability properties of INDI, the actual closed-loop performance may degrade severely in the face of model uncertainty. We address this issue by first analyzing the effects of modelling errors on the closed-loop performance by observing the movement of the system poles. Based on this, we analyze the neccessary modeling fidelity and propose simple modeling methods for the usual actuators found on small-scale electric aircraft. Finally, we analyze the actuator models using (flight) test data where possible.

1 INTRODUCTION

Incremental Nonlinear Dynamic Inversion (INDI) has been applied to a variety of aicraft including quadrotors, hybrid aircraft (tailsitter, tiltwing) and conventional airplanes [2, 3, 4, 5]. The method was first introduced by NASA [6] and then further developed at TU Delft [2, 3]. At the core of INDI a simple control law given by

$$\delta \mathbf{u} = \mathbf{M}_u^{-1} \cdot J \cdot (\nu - \dot{\Omega}) \tag{1}$$

is used, where ν is the commanded angular acceleration, J is the aircrafts inertia and $\mathbf{M}_u$ describes the actuator effectivity. This paper concentrates on the last term $\mathbf{M}_u$ and the associated neccessary dynamic actuator models. We summarize modeling approaches which have been successfully applied in practice for quadrotors [3], tiltwing [7] or tailsitter [4, 8] aircraft. Since the problem of oscillations frequently arises when applying INDI, we try to gain some insight into this issue by observing the closed-loop system poles. A similar analysis was already done in previous work [1], but concentrates on the stability properties of the closed loop.

*Email address: binz@fsd.rwth-aachen.de

2 EFFECTS OF MODELING UNCERTAINTY

The goal of Nonlinear Dynamic Inversion (NDI) is to invert the plant dynamics, so that the resulting closed-loop dynamics are a series of integrators. In principle, the INDI formulation in [3] shares this goal. However, due to the way in which state-derivatives (i. e. angular accelerations) are calculated, the resulting closed-loop dynamics from the commanded anguluar accelerations ν to the actual angular accelerations $\dot{\Omega}$ are the actuator dynamics $A(z)$ in the nominal case. Thus, the design of outer controllers (e.g. angular rate and attitude controllers) is influenced by these actuator dyanmics $A(z)$. One major motivation in using NDI (or INDI), is to simplify the design of outer loop controllers. We thus want to develop an understanding of how well the actuator dynamics and effectivity needs to be known, to still achieve an appropriate outer loop performance.

To analyze this, we assume an INDI-based angular rate controller, as shown in figure 1. For simplicity, we only analyze the single-input single-output case, but expect the results to be transferable in principle to the multiple-input multiple-output case aswell. The parameters of the system consist of the plant parameters – namely the control effectivity M_η, actuator time constant T and actuator delay τ – and the corresponding controller parameters $\hat{M}_u, \hat{T}, \hat{\tau}$. The actuator dynamics are modelled as first-order lags with an optional delay:

$$A(s) = \frac{1}{1+Ts} e^{-\tau s} \tag{2}$$

In the nominal case $\hat{M}_u = M_u, \hat{T} = T$ and $\hat{\tau} = \tau$. To develop an understanding of how uncertainty in the parameters affects the closed loop system, we analyze movement of the system poles when parameters are changed.

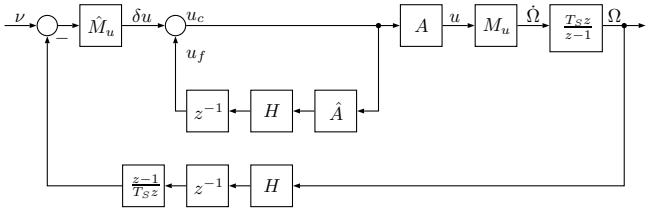


Figure 1: Controller structure

2.1 *Poles of the INDI controller*

In the nominal case, the closed loop transfer function from the commanded angular accelerations ν to the actual angular

accelerations $\dot{\Omega}$ is equal to the actuator dynamics A. In the non-nominal case additional dynamics appear, because the system poles and zeros don't cancel each other. Since the analytical expression for the closed-loop transfer function in the non-nominal case is somewhat convoluted, we graphically analyze the behaviour of the poles instead. While INDI is an inherently discrete-time control algorithm, we choose to display the poles (and zeros) in the continous-time domain because we are more familiar with this setting.

Figure 2 shows the movement of the poles of the INDI loop, when the control effectivity M_u is incorrect. In the

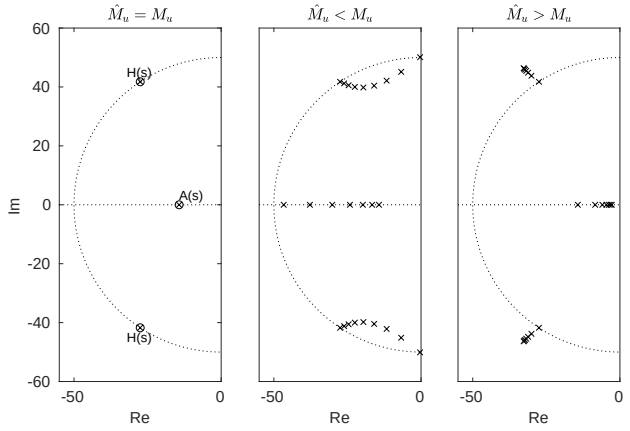


Figure 2: Effect of uncertainty in control effectivity on significant poles of INDI loop, $0.2 < \hat{M}_u/M_u < 4$

nominal case (left), the poles of the filter H are completely cancelled and thus don't influence the closed-loop dynamics. Only the poles of the actuator dynamics A remain. We selected the filter parameters of H as follows:

$$\begin{aligned} H(s) &= \frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \\ \omega_0 &= 50\,\mathrm{rad\,s^{-1}} \\ \zeta &= 0.55 \end{aligned} \tag{3}$$

For $\hat{M}_u \neq M_u$ the cancellation of the filter poles does not occur. For $\hat{M}_u < M_u$, the poles of the filter H become less damped and start to show as oscillations in the time-domain. The frequency of this oscillation roughly equals the natural frequency ω_0 of the filter H. For $\hat{M}_u > M_u$ the system dynamics basically slows down, because the poles of the assumed actuator dynamics $\hat{A}$ move to the right. For the chosen filter parameters (see (3)), the closed-loop becomes unstable for $\frac{\hat{M}_u}{M_u} < 0.2$, though clearly visible oscillations start to appear at around $\frac{\hat{M}_u}{M_u} < 0.5$. Note, that these margins change when chosing different filter parameters H. In general, a larger damping ratio ζ and a larger natural frequency ω_0 lead to more robustness w.r.t. uncertainty in M_u. At the same time, these filter parameters influence the amount of noise introduced when calculating $\dot{\Omega}$ as well as the disturbance rejection performance [3]. Thus, the filter parameters will be a trade-off between robustness w.r.t. to M_u, performance of disturbance rejection and noise.

Figure 3 shows a similar analysis as before, this time changing the assumed actuator dynamics $\hat{T}$. Here, for $\hat{T} <$

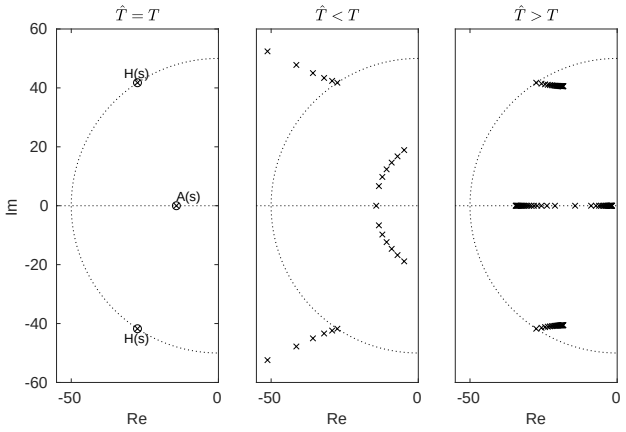


Figure 3: Effect of uncertainty in actuator dynamics on significant poles of INDI loop, $T = 0.07, 0.01 < \hat{T} < 0.14$

T the poles associated with the actuator dynamics become underdamped and move towards lower damping while the frequency stays roughly the same. Again, the more benign direction is an overastimation of the actuator time constant $\hat{T}$, since in this case the most significant pole merely moves towards lower frequencies while still being fully damped.

Finally, figure 4 shows a similar analysis for the effect of uncertainty in the time delay $\hat{\tau}$. As was already discussed in

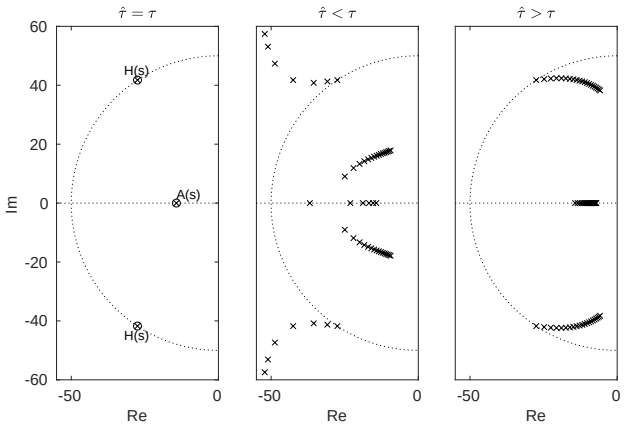


Figure 4: Effect of uncertainty in actuator delay on significant poles of INDI loop (sample time $T_s = 0.001$ sec), $\tau/T_s = 40, 0 < \hat{\tau}/T_s < 80$

the literature [1], a lack of assumed time delay $\hat{\tau}$ can lead to oscillations in the closed-loop. In case of underestimating the delay $\hat{\tau}$, the significant poles first tend towards faster dynamics, while still being fully damped. Once a critical error in the estimated delay $\hat{\tau}$ is reached, the dynamics become underdamped. The more benign case is again an overestimation of the delay $\hat{\tau}$, which leads to slower system dynamics. However, with an increasing overestimation of $\hat{\tau}$, the poles of the filter H become less damped and lead to visible oscillations in the time-domain response. Unfortunately, an error in the estimated delay leads to a behaviour, which is similar to the behaviour in case of an error in either the control effectivity M_u or the actuator time constant T. Determination of the delay τ can however be quite easily accomplished by either analyzing flight test data or performing dedicated testing of the actuators.

In summary, this analysis gives some insight into the behaviour of INDI in case of uncertainty. In our experience, a common problem when implementing INDI for a new aircraft are oscillations. Based on the discussion above, observing the frequency of the oscillations can give a hint as to what the source of the oscillations is. Appropriate mitigations can either be to adapt the assumed model $(\hat{M}_u, \hat{T}, \hat{\tau})$ accordingly or the parameters (ω_0, ζ) of the filter H. Additionally, overestimation of the control effectivity M_u and the actuator time constant T generally leads to slower dynamics and thus might serve as a good starting point for new controller designs. In section 4 we show how the performance of an INDI controller can easily be assesed and tuned in real-time, enabling rapid controller development.

3 ACTUATOR MODELS

Small electric aircraft typically feature two kinds of actuators: rudders and electric motors with propellers to produce thrust. Depending on the configuration, the rudders might additionally be positioned in the slip-stream of the propellers. This configuration is often used to create rudder effectivity even when there is no aerodynamic velocity (e. g. flying-wings, tiltwing aircraft).

To model the effectivity of these actuators, we propose a two-step approach: First, we calculate the thrust, slip-stream velocity and effectivity of the motors. Second, we calculate the effectivity of the rudders, taking into acount slip-stream velocities if neccessary.

In addition to these *static* acutator model properties, we also model the *dynamic* behaviour of the actuators. This is of course only strictly necessary, when the actuator positions are not measured. Still, for designing the outer rate and attitude controllers, an estimate of the actuator dynamics is beneficial.

For both, the static and dynamic properties we rely as much as possible on properties which are either easily measurable or specified by the manufacturers. In Section 4 we analyze how well these actuator models actually perform and how this compares to the requirements on modeling fidelity derived in Section 2.

3.1 Static Actuator Effectivity Models

Within the scope of attitude control, the actuator effectivity describes the change in moments due to changes in actuator position (i. e. rudder deflection or throttle). For many applications it is sufficient to look at the force induced by an actuator and use the corresponding lever to calculate the induced moment. We thus get expressions of the form

$$\mathbf{M}_u = \frac{\partial \mathbf{M}}{\partial u} = \mathbf{r} \times \frac{\partial \mathbf{F}}{\partial u} \tag{4}$$

where $\mathbf{M}_u$ describes the actuator effectivity of an actuator u in the body-fixed coordinate frame given by the cross-product of the actuator position $\mathbf{r}$ and the induced change in force $\mathbf{F}$. In the following section we will mostly focus on determining the term $\partial \mathbf{F} / \partial u$.

3.1.1 Motors

The most common type of electric motor used in electric aircraft is the synchronous AC motor. It needs to be driven by a specialized electronic component called an Electronic Speed Controller (ESC), see Figure 5. An ESC is controlled via a throttle value δ, which can typically be normalized to ranges from 0 to 1 (or -1 to 1, if the ESC supports driving the motor in reverse). The ESC generates the appropriate voltages to drive the motor, resulting in an angular velocity measured as Revolutions Per Minute (RPM) n. Depending on the propeller and inflow conditions, these angular velocities then result in a thrust F. This description makes the simplifying assumption

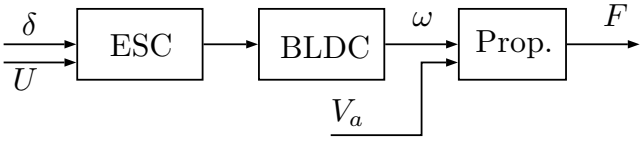


Figure 5: Motor model

that the angular velocity is independent of the inflow. It thus enables using a simpler model at the cost of modeling fidelity.

The motor model we propose consists of two parts: a mapping from the throttle δ and the supply voltage U of the ESC to the RPM n and a mapping from n combined with the inflow V_a to the thrust F.

ESC/BLDC model When the motor RPMs are not measured, we use the following model based on the supply voltage U, the motor speed constant K_V and the throttle setting δ

$$n = U \cdot K_V \cdot \delta \tag{5}$$

This model basically assumes that the motor is in a no-load condition, which is a very crude approximation. The advantage is however, that only the parameter K_V needs to be known, which is usually specified by the motor manufacturer.

Propeller Model For small electric aircraft a database of measured propeller performance exists [9]. Also some manufacturers provide additional performance preditictions [10] based on analytical methods. To model the propeller thrust we first calculate the static thrust produced at zero inflow speed and then correct this value using an estimate of the current inflow speed. We model the static thrust as

$$T_1 = K_1 n^2 \tag{6}$$

The value of K_1 can either be derived using one of the previously mentioned propeller databases, from simple test setups or from previously acquired flight data.

To correct for the inflow velocity, we add a correction term, resulting in

$$T = K_1 n^2 + K_2 V n \tag{7}$$

where V represents the axial inflow speed. The actuator effectivity according to (4) thus becomes

$$T_n = \frac{\partial T}{\partial n} = 2K_1 n + K_2 V \tag{8}$$

This choice of correction term is informed by the propeller data displayed in Figure 6. Figure 6 shows the thrust produced by a propeller[1] at different axial velocities and at different RPMs. For the relevant inflow speeds ($< 20\,\mathrm{m\,s^{-1}}$) and the

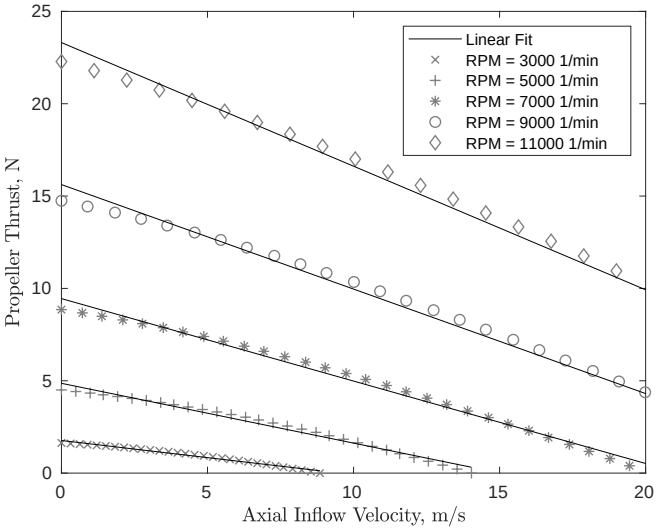


Figure 6: Thrust over axial velocity

region of relevant RPMs an affine function approximates the data well. Given these data, K_1 and K_2 can be found by fitting the propeller model (7) to the data. If propeller data are not available K_2 can be calculated using analytical approaches like blade element momentum theory [11].

As a first approximation, K_2 can also be interpolated from available propeller performance data. The performance database published in [10] was calculated using blade element momentum theory. While here the parameter K_1 is consistently overestimated, the parameter K_2 matches the measured data published in [9] well. The data suggest that K_2 can be approximated as a function of the propeller diameter D and the propeller pitch S:

$$K_2 = p_{11} D^2 + p_{10} D + p_{20} S + p_0 \tag{9}$$

Fitting this function over the available data results in the following model parameters (all units in inch, if applicable):

Database	p_{11}	p_{10}	p_{20}	p_0
UIUC [9]	-1.79e-06	1.70e-05	2.30e-06	-6.75e-05
APC [10]	-1.75e-06	1.70e-05	8.51e-06	-9.28e-05

Since the UIUC database [9] features a wide range of different propeller types and manufacturers, we expect that the corresponding model will extrapolate well to new propellers.

3.1.2 Rudders

We approximate rudders as thin plates, where the rudder effectivity is given by

$$F_\delta = \frac{\partial F}{\partial \delta} = 2\pi \cdot \frac{\rho}{2} V^2 S \cdot \frac{\Lambda}{\Lambda + 2} \tag{10}$$

with the air density ρ, inflow speed V, rudder aera S and aspect ratio Λ. If a rudder is partly in the slip stream of a propeller, the rudder is split accordingly into separate parts. In this case, the aspect ratio Λ still represents the aspect ratio of the whole rudder. The inflow speed V is either the free stream speed or the slip stream speed. In the latter case, we apply momentum theory and assume that the slip stream is fully developed. [5] shows a more detailed example of this approach. Using the propeller model (7) to calculate the thrust T produced by the corresponding propeller, this gives the slip stream velocity as

$$V = \sqrt{\frac{T}{S}\frac{2}{\rho} + V_A^2} \tag{11}$$

where V_A is the inflow speed of the propeller, usually the measured airspeed.

3.2 Dynamic Actuator Models

For dynamic actuator models we use first-order lags with time-delay and optional rate-limit, which is a common approach found in the literature [3, 4]. Figure 7 shows the corresponding block diagram. The actuator time constant T and the rate limit (denoted as $\dot{\theta}_{\max}$) can either be measured or approximated using the manufacturers specifications. Typically, we only model servo motors with a rate limit.

[1] APC 10x3.8 Slow Fly

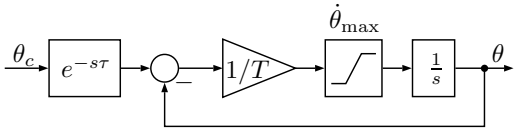


Figure 7: Actuator Model

4 ANALYSIS

In this section we analyze the accuracy of the previously described models. Where possible, we validate the models using measured data from flight tests or other test setups. We already presented the rudder model presented here in previous work [5]. Determining the fidelity of the model would require dedicated wind-tunnel testing which was beyond the scope of this work. We thus don't discuss the rudder effectivity model further in the following analysis.

4.1 Motor model

The motor model consists of two parts: the ESC/BLDC model and the propeller model. To validate the ESC/BLDC model, we analyzed flight test data of a tiltwing aircraft, where the RPM were measured. Figure 8 shows a comparison of the predicted RPM and the measured RPM. As mentioned previously, the ESC/BLDC model assumes a no-load condition, which naturally does not reflect the actual flight conditions. Thus, using the K_V value specified by the manuafacturer will always result in an overestimation of the RPM. The relative error between the expected RPM $\hat{n}$ and the measured RPM n is about $20\,\%$. We found similar accuracies when analyzing wind-tunnel measurement data. The model can be significantly improved by adjusting (i. e. lowering) the K_V value to account for the additional load-conditions. However, implementing such an adjustment requires measuring the actual RPM, in which case the ESC/BLDC model is not needed anyways. Thus, we recommend using the manufacturers K_V value, if RPM measurements are not available. Note, that this model fails to represent fast decelerations, because in this case the propellers are in a windmilling state and are accelerated by the inflow, which is not represented in the model.

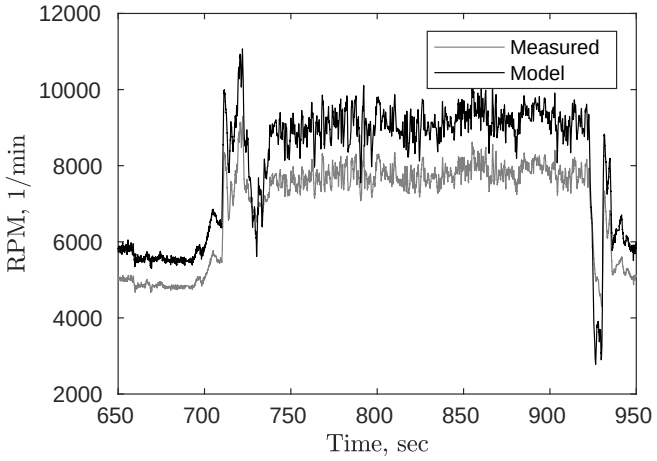


Figure 8: BLDC/ESC model

The propeller model (8) consists of two constants K_1 and K_2. We assume that K_1 can be accurately determined from the static motor model (6). K_2 however has to be either measured in wind-tunnel tests or determined using analytical methods. The approximation of K_2 as a function of the propeller diameter and propeller pitch given in (9) can be used as a first approximation if no other data are available. It is however not clear if over- or underestimation occurs. In our experience, K_2 only becomes significant at high airspeeds, at which point the rudder effectivity usually is high enough for the rudders to act as the primary control surface.

In summary, we expect the motor model to be sufficiently accurate with a tendency to overestimate the propeller effectivity. Thus, reffering to Figure 2, this should result in fully damped system dynamics.

4.2 Dynamic actuator model

The rate-limited first-order lag model used to model servo motors has three parameters: the rate limit $\dot{\theta}_{\max}$, the time constant T and the delay τ. Typically, servo manufacturers only specify a "servo speed" given as the time needed to travel a certain angular distance. Unfortunately it is not clear how exactly this speed specification relates to the servo parameters given above. Directly using the servo speed as the rate limit does certainly not result in an accurate model. To find the model parameters, dedicated tests have to be conducted, for example by probing the internal potentiometer output of the servo motor as suggested in [4]. As an alternative, we built a servo testbench, which also permits us to study the frequency-dependent behaviour of a servo motor. Figure 9 shows the commanded angle and the measured servo angle over time. As is clearly visible, the servo motor cannot reach the commanded amplitude at this frequency. The rate-limit thus leads to an attenuation of the input signal. There is also a phase delay between the commanded and the actual signal. The attenuation and the phase delay give rise to a Bode plot, where we define the phase shift as the value which provides the best-fit between input and output signal.

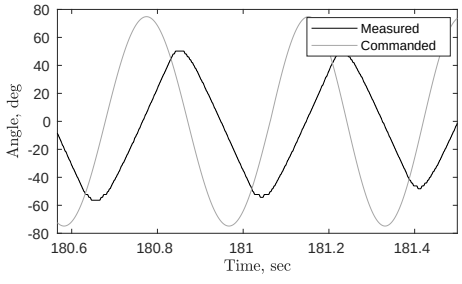


Figure 9: Servo dynamics at high frequencies

Figure 10 shows an example Bode plot obtained by running the test displayed in Figure 9 for many different frequencies. It is clear, that a low-order linear servo model cannot capture the magnitude and phase behaviour of the nonlinear servo model. The sharp edge in the magnitude plot is related to the nonlinear effects of the rate limit. The time constant T of the servo actually has little impact on the overall modeling accuracy. How and if this actuator model should influence the

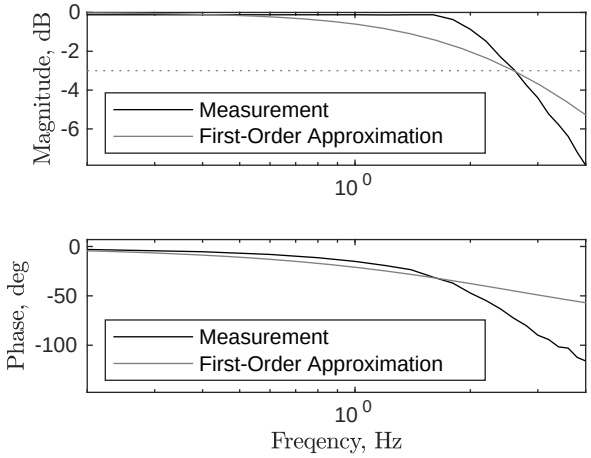


Figure 10: Bode plot of servo dynamics

design of outer loop controllers still needs to be investigated. As a comparison, Figure 10 also shows an approximation of the nonlinear servo model using a first-order lag, where the time constant is chosen such that it matches the edge frequency of the nonlinear servo model. In terms of designing outer loop controllers, such an approximation might serve as a useful abstraction of the nonlinear dynamic model to still allow the application of linear control methods.

In summary, the actuator models presented here are able to capture the dynamic behaviour of the real actuators well. In case of electric motors, simple linear models seem to sufficiently capture the relevant dynamics. In the case of servo motors, the model parameters are hard to derive based on the typical manufacturer specifications. To apply the analysis summarized in Figure 3 suitable alternative (linear) actuator models need to be derived.

5 CONCLUSION

This paper presented our approach to modeling actuators for use in the framework of INDI. First, by studying the effects of modeling uncertainty on the poles of the closed-loop system, the robustness properties of INDI controllers were analyzed. We confirmed the known stability properties of INDI, but found that the uncertainty bounds of acceptable closed-loop performance are (of course) much tighter. With that in mind, we then considered the typical actuator elements found in small electric aircraft, namely electric motors with propellers and rudders actuated by servo motors. We derived suitable models for these elements, trying to rely as much as possible and easily obtainable information.

In the following analysis we assessed the resulting model fidelity using real flight data or measurements where possible. Special consideration was given to typical servo models, which feature a nonlinear rate-limit element. We discussed some effects of this nonlinearity, though further works needs to investigate how and if these nonlinearities should be considered in the design of outer loop controllers.

REFERENCES

[1] R C van't Veld, E van Kampen, and Q P Chu. Stability and Robustness Analysis and Improvements for Incremental Nonlinear Dynamic Inversion Control. (January), 2018.

[2] 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.

[3] 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.

[4] Murat Bronz, Ewoud J Smeur, Hector Garciade de Marina, and Gautier Hattenberger. Development of A Fixed-Wing mini UAV with Transitioning Flight Capability. In *AIAA Applied Aerodynamics Conference, Aviation Forum, At Denver, Colorado*, 2017.

[5] Fabian Binz, Tobias Islam, and Dieter Moormann. Attitude Control of Tiltwing Aircraft Using a Wing-Fixed Coordinate System and Incremental Nonlinear Dynamic Inversion. In *10th International Micro Air Vehicle Competition and Conference*, pages 86 – 93, 2018.

[6] Barton J Bacon and Aaron J Ostroff. Reconfigurable Flight Control using Nonlinear Dynamic Inversion with a Special Accelerometer Implementation. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, number August, 2000.

[7] Fabian Binz, Philipp Hartmann, and Dieter Moormann. Nonlinear Model Predictive Flight Path Control for an Unmanned Powered Paraglider. In *Euro GNC 2017 - 4th CEAS Specialist Conference on Guidance, Navigation & Control*, 2017.

[8] 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(January):79–90, 2018.

[9] J.B. Brandt, R.W. Deters, G.K. Ananda, and M.S. Selig. UIUC Propeller Database, 2019.

[10] APC Propeller Performance Data.

[11] Matthew H McCrink and James W Gregory. Blade Element Momentum Modeling of Low-Re Small UAS Electric Propulsion Systems. *AIAA Aviation*, (June):1–23, 2015.

Experimental versus computational determination of the dynamical model of a glider

Ana Carolina DOS SANTOS PAULINO *,⋄, Antoine MURIE⋆,⋄, Thomas PAVOT⋄, Martin LEFEBVRE⋆,⋄, Renaud KIEFER⋆,⋄, Edouard LAROCHE⋆, Sylvain DURAND⋆,⋄
⋆ICube Laboratory (UMR CNRS 7357), Strasbourg university, Strasbourg, France.
⋄ Institut National des Sciences Appliquées (INSA), Strasbourg, France.

ABSTRACT

In this paper we present and compare two aircraft model identification techniques that are easy to implement and suitable for various airplane models, gliders comprised. One of them relies on flight data, while the second one uses a virtual model of the plane. To obtain the flight data, we propose a flight protocol that is simple to follow. Our analysis show that the methods find resembling results for similar airspeeds.

1 INTRODUCTION

Autonomous flight of aircraft is a subject that has drawn attention of both academia and companies since many years [1]. Several laboratories in this field, such as ETH Zurich's Autonomous Systems Lab, the Drone Lab in Hohschule Rhein-Wall, and the Uninhabited Aerial Vehicle Lab in the University of Minnesota, as well as companies, such as Parrot, DJI and Xiaomi, are investing in research and development of new models as this market is promising for the years to come. As a matter of fact, estimatives of commercial drone revenues indicate a growth from US$1 billion in 2018 to around US$12.6 billion in 2025[1]. The range of applications in logistics, surveillance, security and entertainment is vast and promising, stimulating the development of solutions both in hardware and in software.

In order to autonomously control an Unmanned Aerial Vehicle (UAV), it is useful to design a dynamic model that reproduces its input-output behavior. Such a model might enable the realization of a stability analysis and, later, the implementation of a model-based control strategy. The choice of a dynamical model takes into account the trade-off between the simplicity of the model and its precision over the entirety of the operation point envelope. We seek a model that represents the real aircraft as close as possible into the operation conditions with affordable complexity. Research of the numerical coefficients of the dynamical model that will lead to a reasonable representation of reality is named as *model identification*. It can be done with in-flight and off-flight data [2, 3], and may count on CAD-type softwares (XFLR5, CREO, StarCCM+).

To the best of the authors' knowledge, in the myriad of works about aircraft model identification available, surveys on different flight protocols for identification are not prolific, so that the beginners in the field have little information on which are the suitable signals to excite the dynamical system. Besides, in the universe of modeling and control of aircraft, few comparisons between different identification methods are performed for fixed wing. Among the found techniques, some require expensive setup or firmware modifications [4, 5]. Therefore, the contributions we aim to provide through this paper are the following: first, an overview on the different identification techniques found in the literature is provided; second, two different strategies for the identification of the dynamic model of an aircraft are proposed; and third, a simple flight protocol that provides relevant data for in-flight identification is established. The aforementioned identification strategies are convenient for various types of aircrafts: one of them uses in-flight data that is processed with well-known system identification numerical tools, and the other uses a numerical model for the aerodynamic coefficients' computation. We show that, for the identification of the relationship between aileron deflection and roll angle, both techniques lead to models that are close to each other when subject to the same airspeed.

The present article has the given structure: section 2 describes the setup used in our study, section 3 details the proposed identification techniques and conveys the numerical results found, and in section 4, we conclude this manuscript and evoke some perspectives on future work.

2 TEST SETUP

In this study, the choice was made to use a commercially available remote-controlled aircraft, i.e. an *Epsilon* glider[2], with a standard aircraft geometry. This airplane has many advantages: it is easy to handle, allows gliding and is inexpensive. It has two ailerons, two flaps, one elevator, one rudder and a thruster. Its dimensions make it easy to implement a flight controller in order to transform this aircraft into a drone and recover all flight data. The relative speed of the aircraft is a fundamental information for model characterization, being used in aicraft standard control laws. For these reasons, a Pitot probe was installed on the plane. This probe is the only

*Email address: ac.dossantos@unistra.fr

[1] https://www.tractica.com/newsroom/press-releases/commercial-drone-hardware-and-services-revenue-to-reach-12-6-billion-by-2025/

[2] https://www.absolu-modelisme.com/epsilon-competition-v3-pnp.html

addition to the original structure of the Epsilon glider. Some data from our glider in flight can be found in Table 1.

Wingspan	3.5 m	**Chord**	20.4 cm
Mass	3098 g	**Aspect ratio**	20.47
Length	1.5 m	**Speed**	10-25 m/s
Airfoil	MH32	**Battery**	LiPo 4S 1400 mAh

Table 1: Glider specifications.

To control our Epsilon glider, we use a Pixhawk board. It is an independent open-hardware project which supports multiple open source flight stacks such as PX4 and ArduPilot. The Pixhawk board has several advantages: its accessibility, its low cost and its very active community (scientific and industrial). In the case of this study we chose to work with PX4 because its code allows a great adaptability of the geometries. Through a *mixer file*, it is possible to develop a custom firmware that associates each actuator with a movement (roll, pitch and yaw)[3]. The Epsilon glider uses 7 of the 14 PWM outputs available on the Pixhawk board to operate the entire drone. A more detailed explanation of the positive and negative aspects of using PX4 and several other types of firmware has been done in [6].

3 AIRCRAFT MODEL IDENTIFICATION

In the first part of our study, we are interested in modeling the input-output relationship of our aircraft through a linear dynamic model. We take as control inputs the control surfaces: aileron, elevator and rudder; and as outputs, the plane attitude angles: roll, pitch and yaw. The relations between control surface inputs and plane angular displacements in each axis is modeled by transfer functions, and we assume that the dynamics for each axis are decoupled. This means that control surfaces aileron, elevator and rudder influence respectively roll, pitch and yaw angles. However, in practice, the use of one control surface has an influence over other axes so that, for example, the ailerons produce a yawing moment in addition to a rolling moment when they are deflected [7]. As a matter of fact, vertical and horizontal stabilizers on the tail of the plane and differential mixers on flaps tend to reduce this effect.

3.1 Model structure

The mathematical modeling of an aircraft has been detailed in several sources [1, 2, 3, 8, 9, 10] and can be obtained through Newton's law applied to translational and rotational movements. Here, we present the nonlinear model of a fixed wing plane that will be linearized around an operation point to obtain the aileron-roll transfer function. The mathematical notations summarized in Table 2 and represented in the aircraft body frame in Figure 1, are borrowed from [8].

[3] https://dev.px4.io/en/

p_n, p_e, p_d	positions north, east, down in inertial frame
u, v, w	velocities north, east, down in body frame
ϕ, θ, ψ	roll, pitch and yaw angles
p, q, r	roll, pitch and yaw rate angles in inertial frame
M	airplane mass
g	gravitational acceleration
S	wing area
b	wingspan
c	wing main chord
ρ	air density
f_x, f_y, f_z	forces north, east, down in body frame
l, m, n	roll, pitch and yaw moments in body frame
J_x, J_z	moments of inertia
J_{xz}	product of inertia
F_g^b	gravitational forces in body frame
V_a^b	airspeed in body frame
u_w, v_w, w_w	windspeeds in body frame
$C_L,\ C_D,\ C_Y,$ C_l, C_m, C_n	nondimensional aerodynamic coefficients
$C_{l_{\delta_a}},\ C_{n_{\delta_a}},$ C_{n_p}, C_{l_p}	aerodynamic coefficients
α, β	angle of attack and sideslip
$\delta_a, \delta_e, \delta_r$	aileron, elevator and rudder angles

Table 2: Nomenclature table.

3.1.1 Kinematics

The expressions of the state variables relative to the ground with respect to the ones relative to the body of the plane are expressed in equations (1) and (2), where c_θ and s_θ stand for $\cos\theta$ and $\sin\theta$ respectively.

$$\begin{pmatrix} \dot{p_n} \\ \dot{p_e} \\ \dot{p_d} \end{pmatrix} = \begin{pmatrix} c_\phi c_\theta & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ s_\phi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\psi s_\theta c_\phi - c_\phi s_\psi \\ s_\theta & -c_\theta s_\psi & -c_\theta c_\psi \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} \tag{1}$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \tag{2}$$

3.1.2 Dynamic motion

By applying the second Newton's law, the translational and rotational dynamic motions can be expressed as in (3) and (4)

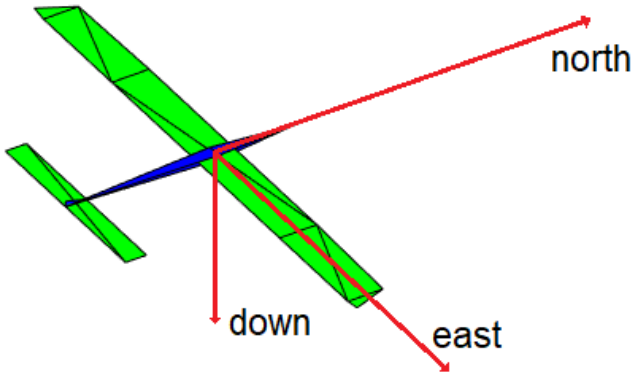


Figure 1: Aircraft with body axes north (roll axis), east (pitch axis) and down (yaw axis).

respectively:

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} rv - qw \\ pw - ru \\ qu - pv \end{pmatrix} + \frac{1}{M} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} \tag{3}$$

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \Gamma_1 pq - \Gamma_2 qr + \Gamma_3 l + \Gamma_4 n \\ \Gamma_5 pr - \Gamma_6 \left(p^2 - r^2\right) + \Gamma_7 m \\ \Gamma_8 pq - \Gamma_1 qr + \Gamma_4 l + \Gamma_9 n \end{pmatrix} \tag{4}$$

with

$$\begin{cases} l = \frac{1}{2}\rho V_a^2 S b C_l(\beta, p, r, \delta_a, \delta_r) \\ m = \frac{1}{2}\rho V_a^2 S c C_m(\alpha, q, \delta_e) \\ n = \frac{1}{2}\rho V_a^2 S b C_n(\beta, p, r, \delta_a, \delta_r) \end{cases} \tag{5}$$

and

$$\begin{cases} \Gamma_1 = \frac{J_{xz}\left(J_x - J_y + J_z\right)}{\Gamma} & \Gamma_6 = \frac{J_{xz}}{J_y} \\ \Gamma_2 = \frac{J_z\left(J_z - J_y\right) + J_{xz}^2}{\Gamma} & \Gamma_7 = \frac{1}{J_y} \\ \Gamma_3 = \frac{J_z}{\Gamma} & \Gamma_8 = \frac{\left(J_x - J_y\right) J_x + J_{xz}^2}{\Gamma} \\ \Gamma_4 = \frac{J_{xz}}{\Gamma} & \Gamma_9 = \frac{J_x}{\Gamma} \\ \Gamma_5 = \frac{J_z - J_x}{J_y} & \Gamma = J_x J_z - J_{xz}^2 \\ V_a^b = \begin{bmatrix} u - u_w \\ v - v_w, \\ w - w_w \end{bmatrix} & V_a = ||V_a^b|| \end{cases}$$

3.1.3 External forces and moments

The external forces can be divided in three main factors, namely gravitational, aerodynamic and propeller forces. The gravitational force is expressed in the body frame as in (6).

$$F_g^b = \begin{pmatrix} -Mg\sin\theta \\ Mg\cos\theta\sin\phi \\ Mg\cos\theta\cos\phi \end{pmatrix} \tag{6}$$

The aerodynamic forces are expressed as a function of the airspeed relative to the plane (V_a defined above) and several aerodynamic coefficients which depend on the shape of the foils as well as the attitude of the body with respect to the air flow. These forces act on the three directions of the aircraft frame: they oppose the forward movement towards north with F_{drag}, they hold the plane up on the sky with F_{lift} and they displace the plane laterally with F_y. Usually, the aerodynamic forces and moments are decomposed in two groups: the longitudinal one with pitch moment (m in (5)) and its mechanical efforts expressed in (7), and the lateral one with its roll and yaw moments (l and n in (5)) and its effort in (8).

$$\begin{aligned} F_{lift} &= \frac{1}{2}\rho V_a^2 S C_L(\alpha, q, \delta_e) \\ F_{drag} &= \frac{1}{2}\rho V_a^2 S C_D(\alpha, q, \delta_e) \end{aligned} \tag{7}$$

$$F_y = \frac{1}{2}\rho V_a^2 S C_Y(\beta, p, r, \delta_a, \delta_r) \tag{8}$$

Finally, the thrust force produced by the propeller depends on the motor used and the speed of the plane through the air. It is not detailed here as the aircraft is only used in glider mode throughout the experiments.

The linearized relationship between aileron displacement δ_a and roll angle ϕ is obtained in the following. Developing (2), we find that $\dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta$. Considering

$$C_l = C_{l_0} + C_{l_\beta}\beta + C_{l_p}\frac{b}{2V_a}p + C_{l_r}\frac{b}{2V_a}r + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r,$$

$$C_n = C_{n_0} + C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_a}p + C_{n_r}\frac{b}{2V_a}r + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r,$$

$\theta \approx 0$ and the effects of pitch and yaw rates (q, r) to be negligible over $\dot{\phi}$, we derive this equation with respect to time and substitute the expression of $\dot{p}$ given in (4). Because of the moments l and n, the aileron deflection δ_a appears. The coefficients C_{l_0} and C_{n_0} are null for symmetric aircrafts and the sideslip angle is taken as $\beta \approx 0$ to obtain the following transfer function:

$$\frac{\phi(s)}{\delta_a(s)} = \frac{a_{\phi_2}}{s(s + a_{\phi_1})} \tag{9}$$

where s is the Laplace operator, $a_{\phi_2} = 1/2\rho V_a^2 S b C_{p_{\delta_a}}$, $a_{\phi_1} = 1/4\rho V_a S b^2 C_{p_p}$, $C_{p_{\delta_a}} = \Gamma_3 C_{l_{\delta_a}} + \Gamma_4 C_{n_{\delta_a}}$ and $C_{p_p} = \Gamma_3 C_{l_p} + \Gamma_4 C_{n_p}$.

3.2 Model identification procedure

3.2.1 State of the art

Flying tests are a cheap way to evaluate the aerodynamics of a plane. The classical method used for the determination of aerodynamic coefficients, or aerodynamic derivatives, is the testing in wind tunnels [11, 4]. It has been used to design the airfoil of the plane and so, to determine aerodynamic coefficients with satisfactory results. However, this approach requires a wind tunnel big enough to at least a plane wing to fit in the best case scenario. Other solutions are to use some simulation tools to deduce the aerodynamic coefficients. Some of them are freely available, such as XFoil and XFLR5[4]. It can be a good start to create a model of the plane, evaluate stability properties and obtain aerodynamic coefficients.

Other works have dealt with the determination of the aerodynamic coefficients through flight tests, such as [12, 13, 14, 15]. Many articles do not have or do not detail a proper flight protocol to obtain the sought aerodynamic coefficients, but some of them do. We find, for instance, methods that extract information from the phugoid mode [4], as well as from the dutch-roll mode[5]. Some other works generate a specific input excitation signal to the control surfaces in order to identify the plane model. The works [5, 16] propose the use of frequency sweep with a rich enough range of frequencies to obtain good estimates of the aerodynamic derivatives.

Furthermore, in order to get a good evaluation of the aerodynamic coefficients as a function of the angle of attack, a proposed method is to let the plane descend with a constant speed and several different slopes to get the aerodynamic coefficients for different angles of attack, either gliding or with activated thrusters. Paper [17] proposes positive and negative slopes for the test. Likewise, [18] presents an identification technique with data obtained from a straight and level path at a constant throttle setting over a large distance. Particularly for our plane, it was not possible to measure the airspeed and activate the thruster at the same time due to the location of the Pitot probe in the wake of the propeller. Therefore, we performed experiments in glide configuration because the activation of the thruster creates a perturbation flow which distorts the sensor's measurements.

Finally, other methods for determining aerodynamic derivatives can be based on neural networks, like in [19]. However, a massive amount of training data is required for a precise identification and a re-initialization for every new type of aircraft model must be performed as well.

In the sequel, we propose two new (in-flight and out-flight) aircraft model identification techniques.

3.2.2 In-flight data identification

For the in-flight data identification procedure, data is collected during the flight and used for identifying a transfer function. This strategy has been chosen in several different papers, some of them employing circular and ascending or descending trajectories [20, 17] and others, producing a sinusoidal input of varying frequency [16, 5]. On the one hand, the use of circular and ascending or descending trajectories may have poor frequency content. On the other hand, the use of frequency-varying sinusoidal inputs requires either changing the radio controller firmware[6] or modifying the autopilot firmware. Concerning the autopilot firmware modification, a more user-friendly approach is to use the Matlab Embedded Coder Support Package for PX4 Autopilots[7]. Summarizing, the so-far presented options can be either financially or technically costly, depending on the user. We have chosen to explore an approach where the pilot maneuvers the plane in an arbitrary design that fairly excites the different modes of the airplane.

The analysis here introduced consists in the characterization of the dynamic relationship between aileron deflection and roll movement by means of a transfer function, but the technique can be transposed for the identification of other relations, eg. elevator deflection to pitch angle and rudder deflection to yaw angle.

Note that, because of the mathematical development that leads to (9), we seek a transfer function of degree 2. Furthermore, through the current method we are interested in identifying the unknown parameters $a_{\phi 1}$ and $a_{\phi 2}$. These parameters are dependent upon aerodynamic coefficients, airspeed, air density, coefficients of inertia and aircraft dimensions, therefore the knowledge of $a_{\phi 1}$ and $a_{\phi 2}$ can lead to the identification of the unknown aerodynamic coefficients. This latter identification was not performed in the scope of this work, but would certainly be a pertinent investigation.

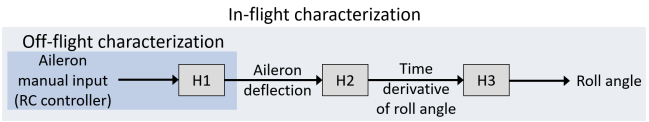


Figure 2: Identification of the transfer function (manual aileron inputs to roll angle). Image modified from [21].

Figure 2 presents how an aileron input affects the roll angle. The relationship between aileron deflection and aileron manual inputs (i.e. H1) is characterized off-flight, by measuring with an incidence meter the aileron displacements associated to various manual aileron inputs (*cf.* Figure 3). This procedure disregards any existing dynamics between manual aileron inputs and aileron displacements under the hypothesis that the aileron dynamics is much faster than the aircraft dy-

[4] https://sourceforge.net/projects/xflr5/

[5] http://www.xflr5.tech/docs/XFLR5_Mode_Measurements.pdf

[6] https://www.open-tx.org/lua-instructions.html

[7] https://fr.mathworks.com/matlabcentral/fileexchange/70016-embedded-coder-support-package-for-px4-autopilots

namics. From Figure 3, we find that the block H1 can be approximated by a constant gain of value 0.0466. Once H1 has been determined, the transfer function H2·H3 is yet to be obtained. This is done with flight tests, from which H1 ·H2·H3 can be identified. Furthermore, from the transfer function (9), we know that the block H3 consists of an integrator.

In the flight test, we bring the airplane up to a certain altitude and, then, we excite the ailerons arbitrarily in amplitude and frequency while the airplane is in glider mode, having its throttle input at zero. Such an aileron input signal is convenient for being easily produced and for allowing rich excitation due to the fast variations of the joystick. To illustrate it, two realizations of aileron manual input signals are shown in Figure 4.

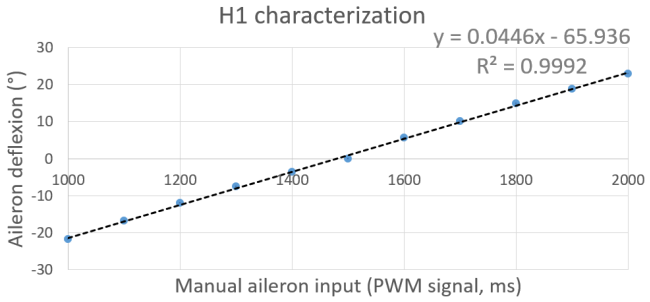


Figure 3: Characterization of the relation (manual aileron inputs to roll angle) through linear regression over the measurements. Image modified from [21].

Once the experiment is performed, a file of format *.ulg* containing several recorded measurements is produced. Two visualization tools available for interpreting these files are Flight Review[8] and pyulog[9]. Particularly with the latter one, it is possible to obtain *.csv* files that can be read by general-purpose applications, such as Matlab and Octave. There also exists a parser, named plotulog[10] that uses pyulog and Octave to display the measurements contained in a *.ulg* file. In a *.ulg* file, one can find information about the plane's altitude, attitude, airspeed, manual control inputs, actuator outputs, GPS position, battery voltage, and more. For our study, we use altitude, thruster, roll and aileron manual inputs to select the experiment windows and to identify transfer functions.

Experiments were performed in glider configuration, meaning that the throttle input was null over the test interval. This is particularly important for our case because of the Pitot probe placement in the wake of the propeller, *cf.* Figure 5. The test datasets were, therefore, taken from the time intervals in which the altitude had an overall decreasing slope and throttle was deactivated, as shown in Figure 6. In this particular dataset, the experiments done by exciting the aileron manual inputs are given by the second, third and fifth grey areas.

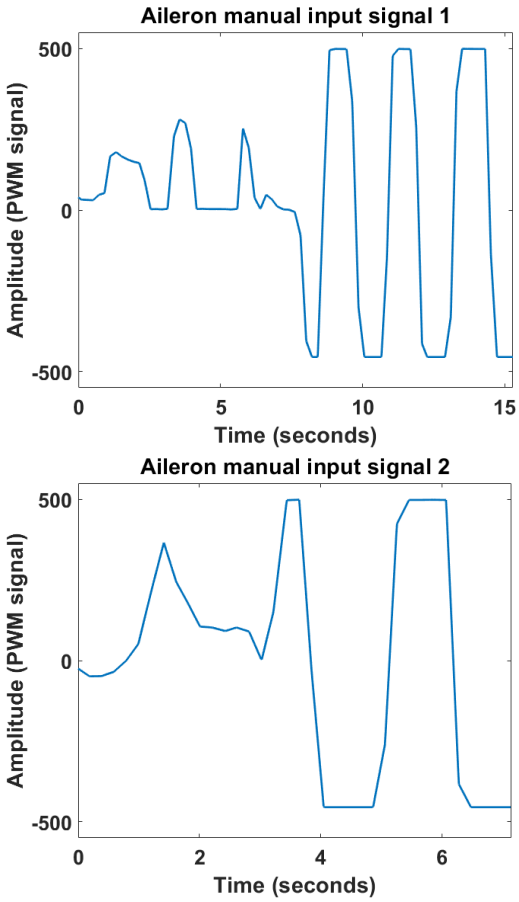


Figure 4: Realizations of arbitrary aileron manual input signals.

Manual inputs, as well as other data, may contain biases. For our experiments, we must observe whether the aileron manual input data contains a bias and, if so, remove it, so that the system identification algorithms can work properly. Furthermore, we know that the transfer function between aileron displacement and roll angle is of second order and that one of its poles is an integrator. Because the integrator is an unstable pole, in the first moment, we identify the transfer function between the aileron manual input and the time derivative of the roll angle. The roll angle is a discrete vector, and its discrete derivative was computed using backward differences divided by the sampling time.

Once the data intervals are selected, system identification can be performed. In our case, we used the Matlab system identification toolbox while calling the functions *data = iddata(OutputVector,InputVector,SamplingTime,'Tstart',0)* to define input-output data objects from excerpts of the flight data; *tf_data = tfest(data,1,0)*[11] to estimate a transfer function from *data* containing 1 pole and no zeros; and *fit = compare(data,tf_data)* to ascertain how good the estimated trans-

[8] https://docs.px4.io/en/log/flight_review.html
[9] https://github.com/PX4/pyulog
[10] https://github.com/kyuhyong/plotulog

[11] The function *tfest* initializes the sought parameters with the Instrument Variable method and obtains its estimation by the minimization the weighted prediction error norm.
Source: https://fr.mathworks.com/help/ident/ref/tfest.html

Figure 5: Plane with highlighted propeller and Pitot probe.

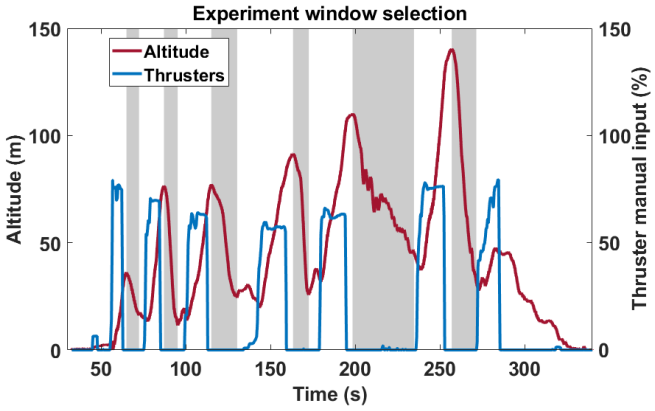


Figure 6: Selection of time intervals for the time experiments indicated by grey areas.

fer function is to reproduce the input-output behavior stored in the object *data*. The quantity *fit* is the "normalized root mean square (NRMSE) measure of the goodness of the fit between simulated response and measurement data"[12].

Besides the Matlab toolbox, there are several other open source alternatives to perform system identification, such as Mataveid[13], Octave system identification toolbox[14], SIPPY[15] or Contsid[16].

For approximate airspeeds, each experiment can be used as training dataset for a transfer function and can be validated using all the datasets. In this sense, for $\mathbf{n}$ datasets, $\mathbf{n}$ transfer functions can be produced, and we can calculate $\mathbf{n}$ values of fit for each transfer function. Among the $\mathbf{n}$ identified transfer functions, we must choose the most suitable to represent the system dynamics in the time derivative of the roll. For that, we perform a weighted average of the fit values produced by each transfer function. Given that the manual aileron inputs in the radio controller are arbitrary, we can imagine that some sequences have better quality than others, being able to produce a transfer function that better represents the overall behavior of the aircraft on the roll axis. These "good quality experiments" have better fit coefficients for most of the transfer functions, so we can consider that they might give more reliable information on the identification process. We consider an experiment to be of *acceptable quality* if the sum of its $\mathbf{n}$ fits is positive. Therefore, we can expect that $\mathbf{q}$ experiments are of acceptable quality, $\mathbf{q} \leq \mathbf{n}$. For example, in Table 3 $\mathbf{n} = \mathbf{q} = 3$, while in Table 7, $\mathbf{n} = 10$ and $\mathbf{q} = 9$.

tf \ ex	1	2	3
1	79.9296	74.7091	82.3494
2	78.7124	75.8576	82.0165
3	79.6587	75.3255	82.6885
sum	238.3008	225.8921	247.0544

Table 3: Values of fit for each transfer function and each experiment (aileron displacement to time derivative of roll angle).

Therefore, the calculation of a general fit for a transfer function obtained from a given dataset is given as:

$$F_{d,i,\mathbf{q}} = \frac{\sum_{j=1}^{\mathbf{q}} f_{d,i,j} \sum_{k=1}^{\mathbf{n}} f_{d,k,j}}{\sum_{l=1}^{\mathbf{n}} \sum_{m=1}^{\mathbf{q}} f_{d,l,m}} \tag{10}$$

where $F_{d,i,\mathbf{q}}$ stands for the general fit of the i-th transfer function from the dataset d and $f_{d,i,j}$ is the fit of i-th transfer function using j-th experiment as validation data. The general fit values obtained from the time derivative of the roll angle are indicated as $F_{\dot{\phi},i,\mathbf{q}}$, with $i = 1, \ldots, \mathbf{n}$, see Table 4.

tf	1	2	3
	$\frac{51.48}{s+17.05}$	$\frac{58.04}{s+18.35}$	$\frac{53.2}{s+16.32}$
$F_{\dot{\phi},i,3}$	79.1121	78.9534	79.3349

Table 4: General fit values obtained from the derivative of the roll angle.

Afterwards, we add an integrator to the found transfer functions and proceed with the fit calculation on roll angle data. We expect that, out of the $\mathbf{n}$ data intervals, $\mathbf{r}$ are of acceptable quality, $\mathbf{r} \leq \mathbf{n}$. Likewise, we can come up with a weighted average of the fit values produced by each transfer function and calculate a general fit as in (10), that we identify with $d = \phi$. These general fit values are, therefore, represented as $F_{\phi,i,\mathbf{r}}$, with $i = 1, \ldots, \mathbf{n}$. At this point, we have two general fit values associated to each transfer function: one related to the roll angle, and the other to its time derivative. To choose the best transfer function candidate, we perform a

[12] https://fr.mathworks.com/help/ident/ref/compare.html
[13] https://github.com/DanielMartensson/Mataveid
[14] https://octave.sourceforge.io/control/overview.html
[15] https://github.com/CPCLAB-UNIPI/SIPPY/blob/master/user_guide.pdf
[16] http://www.contsid.cran.univ-lorraine.fr/

weighted average given by:

$$\mathcal{F}_i = \frac{\left(\sum_{l=1}^{\mathbf{n}}\sum_{m=1}^{\mathbf{q}} f_{\dot{\phi},l,m}\right)\cdot F_{\dot{\phi},i,\mathbf{q}} + \left(\sum_{l=1}^{\mathbf{n}}\sum_{m=1}^{\mathbf{r}} f_{\phi,l,m}\right)\cdot F_{\phi,i,\mathbf{r}}}{\left(\sum_{l=1}^{\mathbf{n}}\sum_{m=1}^{\mathbf{q}} f_{\dot{\phi},l,m}\right) + \left(\sum_{l=1}^{\mathbf{n}}\sum_{m=1}^{\mathbf{r}} f_{\phi,l,m}\right)} \quad (11)$$

Finally, the function with highest value $\mathcal{F}_i$ is the best suitable transfer function to represent the relation between aileron deflection and roll angle. Recalling Figure 3, to obtain the transfer function from aileron displacement to roll angle, we must multiply each transfer function by $1/\text{H1}$. For this experiment, results are found in Table 5. A comparison between the expected input-output behavior and the outcome from the identified transfer functions can be found in Figure 7. Some of the reasons that contribute to the disparity between the expected validation curve and the transfer functions' outcomes are the numerous simplifications that convert the full airplane nonlinear system into a linear one, the disregard of wind inflow and inter-axes couplings, and the consideration that every experiment was performed under constant airspeed.

tf	1 $\frac{51.48}{s^2+17.05s}$	2 $\frac{58.04}{s^2+18.35s}$	3 $\frac{53.2}{s^2+16.32s}$
$\mathcal{F}_i$	77.62	79.53	78.73
airspeed (m/s)	22.22	26.67	31.17

Table 5: Aileron displacement to roll angle identified transfer functions.

To confirm the aforementioned findings, results for data of a second flight, similar to the previous one and containing 10 experiments, can be found in Appendix A.

3.2.3 Out-flight data identification

XFLR5 is an open-source program that performs foil analysis and 3D analysis for aircraft using a combination of inviscid vortex-lattice method and viscous analysis. With this application, we can import and modify foils, create a plane model, generate polar curves for different Reynolds numbers, evaluate efforts for different angles of attack, compute stability properties and aerodynamic derivatives, visualize the movement caused by airplane dynamic modes, and so on. In this section, we use the software XFLR5 to calculate the aerodynamic coefficients of the aircraft. We do so by building a model of the airplane in XFLR5 (*cf.* Figure 8) and equipping it with ailerons.

We ascertain whether the plane is pitching moment inherently stable by performing a *Plane analysis* and verifying that the pitching moment (Cm) is a negative-slope function of the angle of attack. Then, we perform a stability analysis with

[17] https://youtu.be/U7saOcozpi8

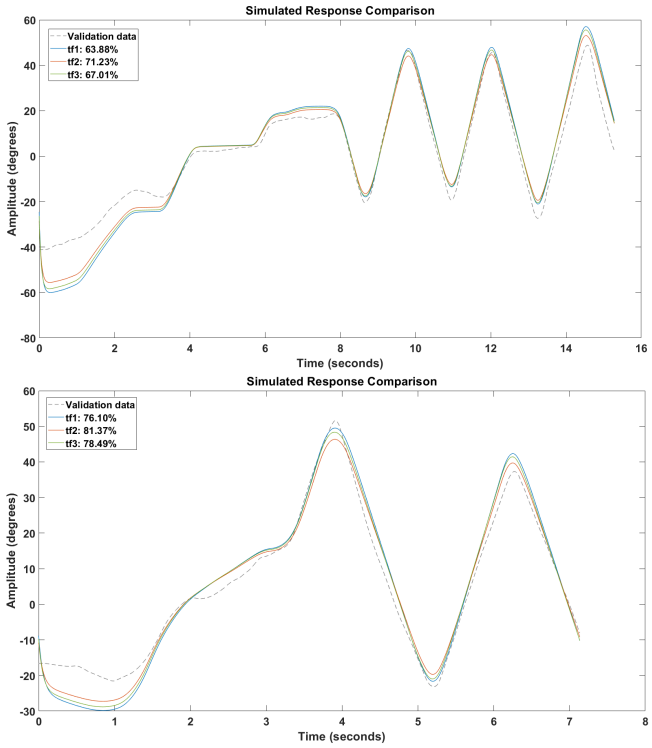


Figure 7: Comparison between the expected input-output behavior (Validation data) and the outcome from the identified transfer functions (tf1, tf2, tf3).

actuated ailerons. The analysis is recorded in a log file that provides various information about inertia coefficients, lateral and longitudinal modes, and aerodynamic coefficients. We can use this data to either complete a full nonlinear model of the plane, or to compose an already linearized dyamic model to compute a transfer function. In the first case, in [8] it is suggested to use Matlab and Simulink for building the nonlinear model, numerically "trimming" it to a specific trajectory and computing the associated transfer functions. In the second case, one can use open source tools, such as Octave[18] or Python language[19], to define a transfer function and evaluate its properties.

$\mathbf{V_a}$	11.46m/s	$\mathbf{S}$	0.637m^2
$\mathbf{b}$	3.18m	$\mathbf{J_x}$	0.869kg/m^2
$\mathbf{J_z}$	1.093kg/m^2	$\mathbf{J_{xz}}$	-0.003446kg/m^2
$\mathbf{C_{l_{\delta_a}}}$	0.3381	$\mathbf{C_{n_{\delta_a}}}$	0.00005847
$\mathbf{C_{l_p}}$	-0.6440	$\mathbf{C_{n_p}}$	-0.07775

Table 6: XFLR5 stability analysis coefficients for our plane model and $\rho = 1.225$kg/m^3.

The transfer function associated to the aileron deflection

[18] https://octave.sourceforge.io/control/function/tf.html

[19] https://python-control.readthedocs.io/en/latest/classes.html

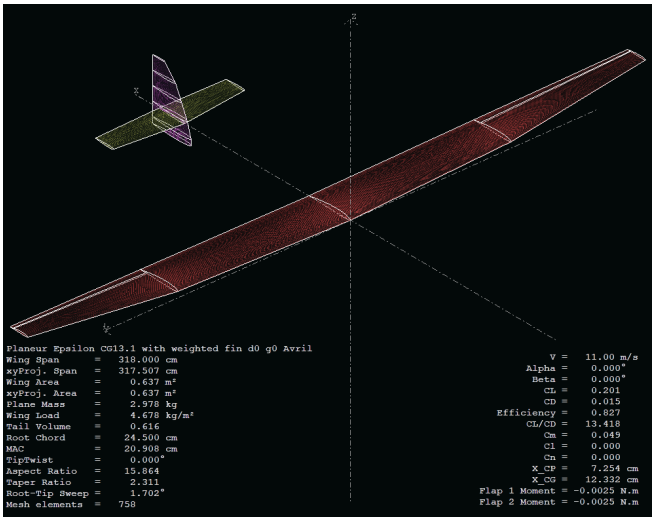


Figure 8: XFLR5 plane model. Note that the model does not contain the body of the plane. This is a recommended practice to avoid numerical issues[17].

and roll angle is given in (9). Among the terms that compose this transfer function, ρ is given by the user, and V_a, S, b, J_x, J_z, J_{xz}, $C_{l_{\delta_a}}$, $C_{n_{\delta_a}}$, C_{l_p} and C_{n_p} are calculated by the XFLR5 stability analysis. The airspeed V_a given by the analysis is the speed that balances the airplane weight. From the stability analysis ran in the airplane model, we obtain the coefficients in Table 6, leading to the resulting transfer function:

$$\frac{\phi(s)}{\delta_a(s)}_{XFLR5} = \frac{63.40}{s(s+16.75)}. \tag{12}$$

We notice that this transfer function presents a gain and poles that are close to the ones found for in-flight experiments with approximate airspeed (*cf.* Appendix A, Table 11, tf9). As a matter of fact, for a difference of airspeeds of $|\Delta V_a| = 0,41365$m/s, we observe a difference between the gains of 1.2801, or 2% with respect to tf9 gain, and a difference between the poles of 1.8482, or 10% with respect to the tf9 non-null pole. However, note that the same does not happen for different airspeeds. For instance, for an airspeed $V_a = 22.2235$m/s, we obtain $\frac{\phi(s)}{\delta_a(s)}_{XFLR5} = \frac{238.49}{s(s+32.49)}$, a very different result from what was obtained in Table 5.

3.2.4 Comparison of in-flight and out-flight identification methods

We stress the fact that both in-flight and off-flight identification techniques are subject to different types of approximations and have advantages and pitfalls. In the in-flight case, we can perform relatively simple experiments and use largely known identification techniques to compute a transfer function. These functions are obtained with the simplifying hypotheses that inter-axes couplings and influences of the wind are negligible, and that the airspeed is constant. On the other hand, the current out-flight identification technique does not even require flight data and can be done by directly calculating the coefficients of (9). However, this technique is also subject to simplifications associated to neglecting inter-axes couplings and the airplane body in the aerodynamic coefficients' calculations. Furthermore, the author of XFLR5 warns that "XFLR5 postulates that the viscous and inviscid contributions to aerodynamic forces are linearly independent" and that "the independence hypothesis is not supported by a theoretical model" [22].

4 CONCLUSIONS AND PERSPECTIVES

In this article we present two different methodologies for airplane model identification that rely on opensource solutions, we propose a flight protocol of simple execution and we bring together some of the state of the art techniques in airplane model identification. Experiments are performed in order to characterize the relationship between aileron deflection and roll angle. One of the air plane model identification methods requires in-flight data and uses the suggested flight protocol. The other utilizes aerodynamic coefficients obtained from a virtual plane model. For the same airspeeds, both techniques convey results in the same order of magnitude. This work aimed to determine a standard protocol for parameter identification through a set of procedures, preferably simple, that leads to an accurate modeling of aircraft input-output behavior. The use of two independent techniques, in-flight and off-flight, endorse the accuracy of the found transfer functions.

As future work, we will evaluate these identification techniques with other planes, one of them being a flying wing, and proceed with the identification with other control surfaces. Knowledge about the dynamical behavior of different aircraft will integrate a Matlab model along with the PX4 PI-FF control structure, so that we will be able to simulate and tune the controller gains for each plane.

ACKNOWLEDGEMENTS

The ELCOD[20] project is co-funded by the European Regional Development Fund (ERDF) and the co-financed project partners Region Grand Est and the countries of Baden-Württemberg and Rhineland-Palatinate in the framework of the INTERREG V Upper Rhine program.

REFERENCES

[1] Emmanuel Roussel. *Contribution à la modélisation, l'identification et la commande d'un hélicoptère miniature*. PhD thesis, 2017.

[2] Robert F. Stengel. *Flight Dynamics*. Princeton University Press, 2004.

[3] Robert C. Nelson. *Flight Stability and Automatic Control*. McGraw-Hill Education, oct 1997.

[20] www.elcod.eu

[4] M. Scherer and P. Mathé. Mesure des derivées aérodynamiques en soufflerie et en vol. Technical report, Organisation du Traité de l'Atlantique Nord, 1961.

[5] Said S Hamada. *Development of a Small Unmanned Aerial Vehicle Longitudinal Model for Future Flutter Testing*. Master thesis, Embry-Riddle Aeronautical University, 2018.

[6] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *Proceedings of IEEE ICRA*, pages 6235–6240, jun 2015.

[7] E H J Pallett, Amraes S Coyle, and Msetp Blackwell. *Automatic Flight Control*. Wiley-Blackwell - 4th edition, dec 1993.

[8] Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft*. Princeton University Press, apr 2015.

[9] Néstor Alonso Santos Ortiz. *Analyse de la robustesse de la loi de commande d'un quadrirotor embarquant une charge suspendue par câble*. Master iriv, INSA de Strasbourg, 2017.

[10] Bernard Etkin and Lloyd Duff Reid. *Dynamics of Flight: Stability and Control*. Wiley, oct 1995.

[11] *Modeling the Aircraft*, chapter 2, pages 63–141. John Wiley & Sons, Ltd, 2015.

[12] G. T. Chapman and D. B. Kirk. A method for extracting aerodynamic coefficients from free-flight data. *AIAA Journal*, 8(4):753–758, 1970.

[13] Lawrence E. Hale, Mayuresh Patil, and Christopher J Roy. Aerodynamic Parameter Identification and Uncertainty Quantification for Small Unmanned Aircraft. *Journal of Guidance, Control, and Dynamics*, 40(3):680–691, 2016.

[14] Jieliang Shen, Yan Su, Qing Liang, and Xinhua Zhu. Calculation and identification of the aerodynamic parameters for small-scaled fixed-wing UAVs. *Sensors (Switzerland)*, 18(1), jan 2018.

[15] Ruiyong Zhai, Zhaoying Zhou, Wendong Zhang, Shengbo Sang, and Pengwei Li. Control and navigation system for a fixed-wing unmanned aerial vehicle. *AIP Advances*, 4(3), mar 2014.

[16] Watcharapol Saengphet, Suradet Tantrairatn, Chalothorn Thumtae, and Jiraphon Srisertpol. Implementation of system identification and flight control system for UAV. In *Proccedings of 3rd IEEE ICCAR 2017*, pages 678–683, jun 2017.

[17] Brent Michalowski and Nathaniel Varano. UAV flight test characterization using minimal test equipment. *Proceedings of ICUAS 2017*, pages 1737–1741, 2017.

[18] Jon Ostler and W. Bowman. Flight Testing of Small, Electric Powered Unmanned Aerial Vehicles. *2005 U.S. Air Force T&E Days*, 1(3), 2005.

[19] Dennis J. Linse and Robert F. Stengel. Identification of aerodynamic coefficients using computational neural networks. *Journal of Guidance, Control, and Dynamics*, 16(6):1018–1025, 1993.

[20] Pedro L. Jimenez, Jorge A. Silva, and Juan S. Hernandez. Experimental validation of Unmanned Aerial Vehicles to tune PID controllers in open source autopilots. *Proceedings of the 7th EUCASS*, 2017.

[21] Anthony Coindevel and Eustáquio Fernandes. Commande de drone de type Avion dans le cadre du projet ELCOD - Rapport final - Projet de Recherche Technologique, INSA Strasbourg. Technical report, 2017.

[22] Model analysis with XFLR5. In *R/C Soaring Digest 2008*.

APPENDIX A DATA FROM A SECOND IN-FLIGHT EXPERIMENT

tf \ ex	1	2	3	4	5
1	66.77	25.51	12.88	-50.26	70.11
2	41.59	35.92	16.82	-41.26	57.58
3	45.08	35.34	16.96	-42.93	60.12
4	-9.62	-29.46	5.71	3.93	-9.54
5	63.36	24.29	13.64	-51.84	73.80
6	64.67	22.90	12.62	-52.16	72.69
7	63.91	27.96	14.85	-49.84	73.09
8	58.15	30.96	16.00	-47.91	69.13
9	63.77	24.43	13.62	-51.73	73.77
10	61.15	20.94	12.47	-53.47	73.33
sum	518.82	218.79	135.57	-437.46	614.08

tf \ ex	6	7	8	9	10
1	46.51	70.66	59.93	73.43	74.07
2	38.86	60.44	59.17	58.79	57.14
3	40.01	63.23	62.20	61.82	60.49
4	-0.97	-3.10	-6.44	-4.66	-2.18
5	46.59	72.28	62.32	75.70	77.31
6	46.96	71.17	59.79	75.15	76.94
7	46.22	73.05	64.61	74.99	75.42
8	44.14	70.96	66.49	71.27	71.07
9	46.68	72.31	62.21	75.71	77.25
10	46.53	70.60	59.67	75.15	77.81
sum	401.54	621.58	549.95	637.36	645.32

Table 7: Values of fit for each transfer function and each experiment (aileron displacement - time derivative of roll angle). The columns in black color contain experiments of acceptable quality.

tf	1 $\frac{23.68}{s+161.9}$	2 $\frac{1.598}{s+15.61}$	3 $\frac{1.269}{s+11.41}$	4 $\frac{-0.03452}{s+6.141e-05}$
$F_{\dot{\phi},i,9}$	63.3592	52.2972	54.8808	-6.1620
tf	5 $\frac{2.732}{s+17.55}$	6 $\frac{4.439}{s+28.35}$	7 $\frac{2.517}{s+17.25}$	8 $\frac{1.552}{s+11.37}$
$F_{\dot{\phi},i,9}$	64.7947	64.1104	64.9631	62.4581
tf	9 $\frac{2.885}{s+18.6}$	10 $\frac{2.93}{s+17.91}$		
$F_{\dot{\phi},i,9}$	64.8382	63.6704		

Table 8: General fit values obtained from the derivative of the roll angle.

ex / tf	1	2	3	4	5
1	95.54	-112.19	-167.79	-56.05	-127.79
2	64.80	-40.21	-103.34	-37.89	-60.41
3	68.47	-51.15	-115.93	-40.59	-73.29
4	65.90	-70.51	34.93	86.89	-344.35
5	93.52	-123.80	-181.93	-58.53	-142.94
6	95.20	-126.98	-183.58	-59.50	-144.37
7	89.73	-107.92	-167.18	-54.75	-127.24
8	81.93	-90.33	-152.85	-50.23	-112.27
9	93.69	-123.06	-181.04	-58.39	-141.94
10	93.49	-136.76	-193.92	-61.59	-155.73
sum	842.3	-982.9	-1412.6	-390.6	-1430.4
ex / tf	6	7	8	9	10
1	4.88	-34.98	61.23	-32.70	5.90
2	8.97	-11.55	63.55	2.01	20.14
3	8.52	-15.59	67.16	-4.13	17.92
4	29.45	-43.87	-3.58	-78.06	-39.60
5	3.34	-40.45	61.43	-41.43	1.38
6	3.18	-41.17	59.62	-42.21	1.03
7	4.86	-34.48	65.01	-32.49	5.96
8	6.13	-28.88	68.84	-24.30	9.63
9	3.44	-40.09	61.48	-40.84	1.71
10	1.97	-45.44	57.81	-48.84	-2.59
sum	74.7	-336.5	562.6	-343.0	21.5

Table 9: Values of fit for each transfer function and each experiment (aileron displacement - roll angle). The columns in black color contain experiments of acceptable quality.

tf	1 $\frac{23.68}{s(s+161.9)}$	2 $\frac{1.598}{s(s+15.61)}$	3 $\frac{1.269}{s(s+11.41)}$	4 $\frac{-0.03452}{s(s+6.141e-05)}$
$F_{\phi,i},4$	76.8858	60.9124	64.2674	36.5376
tf	5 $\frac{2.732}{s(s+17.55)}$	6 $\frac{4.439}{s(s+28.35)}$	7 $\frac{2.517}{s(s+17.25)}$	8 $\frac{1.552}{s(s+11.37)}$
$F_{\phi,i},4$	75.6831	75.9327	75.0408	72.2169
tf	9 $\frac{2.885}{s(s+18.6)}$	10 $\frac{2.93}{s(s+17.91)}$		
$F_{\phi,i},4$	75.8080	74.1839		

Table 10: General fit values obtained from the roll angle.

tf	1 $\frac{530.9}{s(s+161.9)}$	2 $\frac{35.83}{s(s+15.61)}$	3 $\frac{28.45}{s(s+11.41)}$	4 $\frac{-0.7739}{s(s+6.141e-05)}$
$\mathcal{F}_i$	66.8336	54.5100	57.2918	4.8055
V_a (m/s)	11.1412	8.7697	13.2818	13.7666
tf	5 $\frac{61.25}{s(s+17.55)}$	6 $\frac{99.52}{s(s+28.35)}$	7 $\frac{56.44}{s(s+17.25)}$	8 $\frac{34.79}{s(s+11.37)}$
$\mathcal{F}_i$	67.5914	67.1470	67.5516	64.9647
V_a (m/s)	11.1168	11.5762	10.7339	9.7307
tf	9 $\frac{64.68}{s(s+18.6)}$	10 $\frac{65.69}{s(s+17.91)}$		
$\mathcal{F}_i$	67.6558	66.3708		
V_a (m/s)	11.0446	12.0568		

Table 11: Aileron displacement - roll angle identified transfer functions.

Control and Guidance of an Autonomous Quadrotor Landing Phase on a Moving Platform

M. S. Ale. Isaac*, A. Naghash, and S. H. Mirtajedini
Amirkabir University of Technology, 424 Hafez Ave, Tehran, Iran

ABSTRACT

This summary describes an application of a vision-based implementation of three control algorithms to a rather light quadrotor to land on a moving platform with a random path and unknown velocity. Comparing sliding mode (SM), PID, and model reference adaptive (MRAC) controllers in both MATLAB SimScape synthetic space and real-world, we proved the superiority of the former one. The guidance method is an online tracking which uses a linear regression to estimate the landing point. We used a state-of-the-art visual odometry algorithm, SVO, augmented by IMU to correct the path angle. No prior information about the quadrotor or the landing platform is required.

1 INTRODUCTION

Quadrotor landing phase has been a salient research challenge in recent years. The challenge will arise when the landing platform is a moving object which constantly changes its course on a randomly generated path. By enabling a quadrotor to land on such a platform in a robust and smooth manner, this will be more prepared to be deployed by moving machines, such as next generation of autonomous cars, boats an even planes. Besides, This ability has various benefits, include, faster charging for more flight endurance, mapping, search, rescue, and assistance mobile objects [1, 2, 3]; explicitly, this has been used in robotic challenges like IMAV competitions. The bottlenecks contain, first, detecting the platform and find its position, as well as estimating a reliable spot to be considered as a goal for our robot to touch the moving platform; second, implementing a control algorithm which is able to track the platform and in the meantime, immune to disturbances that, in practice, are imposed to the plant during the landing.

1.1 Related Work

There are a few thrived projects in the same submission, Lee et al. ponder a line of sight (LOS) algorithm for the guidance section, compounding with a conventional controller (e.g. PID) will flourish [4]; however, following in this situation requires a fairly large camera with a wide field of view, in order not to miss the moving platform, and consequently, is not operational easily. Falanga et al. worked on a cascade controller (compounding of two PID controllers with the feedback of states, velocities, and accelerations), compiled on an onboard computer, equipping with state estimation and path planning [1], but lacking random target estimation.

1.2 Contribution

In this paper we propose a quadrotor system which detects, follows, and lands on a moving platform just using an onboard computer. The merit of our work is summarized in measuring the platform's random positions online and then, fitting a convenient curve on previous points. By calculating the polynomial coefficient on the curve, it will produce the new point, based on the time step and estimated velocity of the platform. No prior information about neither velocity nor position and not even the path of the moving platform or the quadrotor is needed. Every point under the quadrotor will be covered and mapped, then the localization process will start.

2 SYSTEM OVERVIEW

Here, we will describe following items:

- Control and guidance;
- Position estimation;
- Landing platform detection;
- Virtual platform;
- Experimental platform.

2.1 Control and Guidance

This subsection is devided into 4 segments:

- Sliding Mode algorithm;
- PID algorithm;
- MRAC algorithm;
- Navigation.

Comparing aforementioned controllers, we investigate which ones fulfill the following requirements: stability in both descending and landing phase, resistance to the probable noises, either internal or external, faster response to a sudden and random path deviation, and ability to thrust again right after landing. Consequently, we put our attention into considering

*The authors are with the Micro Air Vehicle Group, Amirkabir University of Technology—http://autmav.com Email address(es): sadegh_al@aut.ac.ir, naghash@aut.ac.ir, sehomi@aut.ac.ir

all nonlinear terms of the system, not just in controllers nor the main dynamic model of the system, except the PID controller, which has a linear base and so conducive to linearizing the system. Worthwhile, the main strategy for compounding each controller with the navigation algorithm is based on immediate reaction of the plant to fly over the moving platform and keep lock its sight on until landing. Besides, the controllers are divided into inner and outer loops [5]; the reference values are x_{ref}, y_{ref}, z_{ref}, and ψ_{ref} which are determined by the guidance law and computing the angle of plant trajectory, using tangent inverse. Moreover, state variables, x and y are controlled in the outer loop and in counterpart ϕ, θ, and ψ are controlled in the outer loop, and z is controlled separately, but not in this division. Thereby, the slow dynamic equations could be considered [3, 6] as:

$$\begin{cases} \ddot{x} = \frac{F_z}{m}(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\ \ddot{y} = \frac{F_z}{m}(\cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi) \end{cases} \tag{1}$$

Solving equations (1), we have:

$$\begin{cases} \phi_{des} = \frac{m}{F_z}(\ddot{x}\sin\psi - \ddot{y}\cos\psi) \\ \theta_{des} = \frac{m}{F_z}(\ddot{x}\cos\psi + \ddot{y}\sin\psi) \end{cases} \tag{2}$$

Equations (2) state that the outer controller loop could be nonlinear and the desired attitudes depend on the thrust force, longitudinal and lateral accelerations, and yaw angle; if so, using a PD controller instead all above will satisfy our criteria; however, using PID, MRAC or SM methods have latency because of their integral components and is so against our goal to be fast responding in transient phase. For fast dynamic [3] we have:

$$\begin{cases} \ddot{z} = \frac{U_1}{m}(\cos\phi\cos\theta) \\ \ddot{\phi} = \dot{\theta}\dot{\psi}\frac{Iy-Iz}{Ix} + \frac{J_r}{I_x}\dot{\theta}\Omega_r + \frac{l}{I_x}U_2 \\ \ddot{\theta} = \dot{\phi}\dot{\psi}\frac{Iz-Ix}{Iy} + \frac{J_r}{I_y}\dot{\phi}\Omega_r + \frac{l}{I_y}U_3 \\ \ddot{\psi} = \dot{\phi}\dot{\theta}\frac{Ix-Iy}{Ix} + \frac{1}{I_z}U_4 \end{cases} \tag{3}$$

U_1 is the total thrust force which equals F_z and other $U_2, U_3, and U_4$ are the roll moment, pitch moment, and yaw moment respectively. Meanwhile, J_r is the rotor gyroscopic inertia and Ω_r is the rotor angular velocity.

2.1.1 *Sliding Mode Algorithm*

To compute the switiching surface for any system with the degree of n, we have:

$$x^{(n)} + f(x) = u \rightarrow S(x,t) = \left(\frac{d}{dt} + \lambda\right)^{n-1} e \tag{4}$$

The quadrotor is a second order system, consequently, all the nonlinearities which refer to the $f(x)$ equal zero [7]. Computing the equal energy for reaching the switching surface u_{eq}, and determining total energy u we have:

$$\begin{cases} u_{eq} : \dot{S} = 0 \rightarrow x^{(n-1)} - {x^{(n-1)}}_d + \lambda\dot{e} = 0 \\ \rightarrow u_{eq} = {x^{(n-1)}}_d - \lambda\dot{e} \\ u = u_{eq} - Ktanh(S) \rightarrow u = {x^{(n-1)}}_d - \lambda\dot{e} - K\tanh(S) \end{cases} \tag{5}$$

K value refers to a discontinuous component against system noises, which is calculated from try and error, means if it more than a determined magnitude, the system will be stable. This is derived from the fact that how far negative is the Lyapunov function derivative, it will converge to a value more negative and so will be stable faster. In addition, instead of $sign$, we use $tanh$ function to make the chatterings of the switching surface more smooth, so nor requires integration of swithcing surface. Totally, the controller will be designed as hereunder:

$$\begin{cases} \ddot{x}_d = -\lambda\dot{e}_x - K\tanh(S_x) \\ \ddot{y}_d = -\lambda\dot{e}_y - K\tanh(S_y) \\ U_1 = m(\ddot{z}_d - \lambda\dot{e}_z) - K\tanh(S_z) \\ U_2 = \frac{I_x}{l}(\ddot{\phi}_d - \lambda\dot{e}_\phi) - K\tanh(S_\phi) \\ U_3 = \frac{I_y}{l}(\ddot{\theta}_d - \lambda\dot{e}_\theta) - K\tanh(S_\theta) \\ U_4 = I_z(\ddot{\psi}_d - \lambda\dot{e}_\psi) - K\tanh(S_\psi) \end{cases} \tag{6}$$

λ values are computed in the simulation and then corrected by implementation results.

2.1.2 *PID Algorithm*

Using a PD as outer loop and PID for inner one, we have:

$$\begin{cases} \ddot{x}_d = K_{d_x}\dot{e}_x + K_{p_x}e_x \\ \ddot{y}_d = K_{d_y}\dot{e}_y + K_{p_y}c_y \\ U_1 = K_{d_z}\dot{e}_z + K_{p_z}e_z + K_{i_z}\int e_z \\ U_2 = K_{d_\phi}\dot{e}_\phi + K_{p_\phi}e_\phi + K_{i_\phi}\int e_\phi \\ U_3 = K_{d_\theta}\dot{e}_\theta + K_{p_\theta}e_\theta + K_{i_\theta}\int e_\theta \\ U_4 = K_{d_\psi}\dot{e}_\psi + K_{p_\psi}e_\psi + K_{i_\psi}\int e_\psi \end{cases} \tag{7}$$

The two former equations are related to the outer controller loop which helps us computing the desired longitudinal and lateral accelerations.

2.1.3 *MRAC Algorithm*

The adaptation law is based on the trajectory following. We introduce a second order system which must adapt the model to the reference model [7, 6]. The chosen model, reference model, and the adaptation law are, respectively:

$$G(s) = \frac{1}{s(s+a)} \tag{8}$$

$$G_m(s) = \frac{w^2}{s^2+2\xi\omega s+\omega^2}$$
$$\rightarrow \ddot{x}_m + \underbrace{2\xi\omega}_{a_1} x_m + \underbrace{\omega^2}_{a_2} x_m = \underbrace{\omega^2}_{b} u_c \quad (9)$$
$$u = \theta_1 u_c - \theta_2 \dot{x} - \theta_3 x \quad (10)$$

a is the estimation parameter to adapt our model to the reference model. ξ and ω are damping ration and system frequency, and θ_is are values we compute to update the adaptation law. Differencing equations (5) and (6), adding $a_1\dot{x}+a_2x$ term to both sides of the equation, and simplifying, we have:

$$e = \frac{1}{s^2+2\xi\omega s+\omega^2}\left(\begin{bmatrix} \dot{x} & -x & u_c \end{bmatrix}\begin{bmatrix} \widetilde{\theta}_2 \\ \widetilde{\theta}_3 \\ \widetilde{\theta}_1 \end{bmatrix}\right) \quad (11)$$

The system is not strictly positive real (SPR); therefore, we cannot use Kalman Yakubovich Lemma [5] and use state space equations to solve the system. Hence:

$$A = \begin{bmatrix} O_{6\times6} & I_{6\times6} \\ -a_2 & -a_1 \end{bmatrix} = \begin{bmatrix} O_{6\times6} & I_{6\times6} \\ -w_i^2 & -2\xi_i\omega_i \end{bmatrix} \quad (12)$$

ξ_i and ω_i magnitudes are exploited by various tests. To calculate errors in state space form, we have:

$$\begin{bmatrix} \dot{e} \\ \ddot{e} \end{bmatrix} = A\begin{bmatrix} e \\ \dot{e} \end{bmatrix} + B\underbrace{\begin{bmatrix} -\dot{x} & -x & u_c \end{bmatrix}}_{\bar{\Phi}}\begin{bmatrix} \widetilde{\theta}_2 \\ \widetilde{\theta}_3 \\ \widetilde{\theta}_1 \end{bmatrix} \quad (13)$$

$$\begin{bmatrix} \dot{\widetilde{\theta}}_2 \\ \dot{\widetilde{\theta}}_3 \\ \dot{\widetilde{\theta}}_1 \end{bmatrix} = -\Gamma\bar{\Phi}^T B^T P\begin{bmatrix} \dot{e} \\ \ddot{e} \end{bmatrix} \quad (14)$$

Γ and P are symmetric positive-definite matrixes which defined to satisfy $A^TP + PA = -Q$, and Q is the same as P. The Lyapunov candidate function proof of stability comes in the Appendix.

2.1.4 *Navigation in cases of visible platform and temporarily lost platform*

We use a simple method for both tracking and landing on moving platform. Explicitly, because of unknown velocity or path pattern of the object, we consider a minuscule component of the time, in order to reduce computation, especially in implementation. For sake of estimating the touch down spot (the estimated landing point), a second order regression is implemented, which takes a buffer of last 15 positions of the platform trajectory into account, and as a result, the coordinates on the fitted curve at a certain number of time-steps (in our case, 5) after the current platform position is considered as the estimated landing point. In other words, it is expected that the quadrotor and the moving platform will meet, on this calculated position. Calculating such a point seems to be an obligation due to the fact that in vision-based, precise landing, scenarios such as the present work, the platform will get out of sight as soon as the camera gets closer than a threshold (in our case, it is 0.3 m). From this point onward, the robot needs to blindly reach the estimated landing point and the reliability of this estimation as well as the accuracy of both, robot estimated position and platform detected position comes into play, which will be explained in proceeding sections.

2.2 Position Estimation

A very necessary objective of our drone is to maintain its stability even in case of losing its landing target so position control is still active and the procedure of testing will be less hazardous. The prerequisite of a great position control is to have position feedback. Many common sensors used for having position feedback are GPS, IMUs, Infrared Markers, Radio Beacons and motion capture systems (MOCAP). For a quadrotor to be truly autonomous, all computations of position estimation must be onboard and since we are planning for the precise landing on a moving platform, the position feedback also needs to be both precise and enough accurate. Here we implement one of the most absolute methods of position and attitude estimation, close to ground or rather featureful environments, referred to as visual inertial odometry (VIO) methods. A comparison between state-of-the-art VIO approaches could be found in [1].
Here we chose the Semi-Direct Visual Odometry (SVO) [8] algorithm for position estimation of our quadrotor. This algorithm is compiled on an Odroid XU4 companion computer and a forward-looking camera. The reason for the forward-looking camera setup is to prevent any position estimation error when the drone is close to the landing platform because in down-looking setup most of the camera field of view will be filled with the platform itself.
We have tested the precision of the system with two configurations in a scenario close to our objective, to test the limits of the estimation algorithm for our specific setup. The scenario is moving the camera on a sine-like path close to the ground with changing the camera heading so that the camera view is being changed constantly.

- **First configuartion**: the algorithm runs in monocular configuration and the trajectory in the XY plane is shown in "Figure 3". The trajectory is compared to a ground truth waypoint path and the result is showing that there is a huge difference between two trajectories. The SVO trajectory is scaling down as the camera is moving further which could result in any unpredictable behavior of the control system. Similar issues have been reported for implementations of the algorithm with different camera models, especially for non-global shutter cameras. This could be explained by the blurriness occurring in images when the non-global shutter cameras are rotating and changing the

view and the algorithm is unable to estimate the camera rotations properly. Generally, the visual odometry (VO) has problems in "Pure Rotation" movements. A discussion about it can be found in [9, 10].

- **Second configuartion**: one way to work around the errors due to camera rotations and changes in camera view (especially in non-global shutter cameras) is the integration of an IMU system as the SVO has its own extended Kalman filter (EKF) [8] running at 200 Hz. So in this configuration, an IMU data at 150 Hz is provided to the filters and with the same dataset, the results are shown in Figures 4, 5. Note that the trajectory of SVO is more close to the ground truth waypoint path and its scale is not decreasing as the camera moves further. Only the trajectory is slightly getting away from the Ground Thruth but still, this position feedback is good enough for controlling the quadrotor positions in landing phase because the quadrotor target is to follow the landing platform and because of that, even small drifts in position will not make our precise landing fail.

2.3 Landing Platform Detection

For the purpose of detecting the Moving Platform, a down-looking camera is mounted on the quadrotor capturing images at a rate of 10 Hz. The target on the moving platform must be a standard and easy to detect marker, so we have used an Augmented Reality markers board (AR), which is shown in fig 4, and the ArUco module of OpenCV library[5] to detect both the position and orientation of the landing platform in the down-looking Camera frame $\{\mathbf{C_d}\}$ By performing a homogeneous transformation from down-looking camera frame $\{\mathbf{C_d}\}$ to the forward-looking camera $\{\mathbf{C_f}\}$ (the camera which is used for SVO) and then using another homogeneous transformation from forward-looking camera $\{\mathbf{C_f}\}$ to the world frame $\{\mathbf{W}\}$, obtained from position estimations of SVO, we will have the landing platform position in the world frame.

2.4 Virtual Platform

We built a complete process of our mission in the powerful MATLAB SimScape simulator. All the three controllers, moving platform, and cameras are simulated to compare their performance and achieve a precise implementation [11, 12]. Meanwhile, most of the controllers' gains are set in the simulator and then, corrected in real-world. Besides of the pros of MATLAB simulator, such can be easily done, friction and noise included, model-based dynamics and so forth, there are a few defects, like weak and difficult collision avoidance simulation, no detection probability. Considering all merits and demerits we suppose optimum detection of the downward camera and make the dynamic model of the quadcopter in the platform. No dynamic equation is needed to build the model, just by exporting from CATIA or SolidWorks, either a .xml or .STL file, to the MATLAB software. Based on our knowledge, the sliding mode then, PID, and finally the MRAC controllers keep the plant more stable, respectively. Specifically, when the stochastic noises grow or the object velocity increases, or even when the standard deviation of the random path (σ) moves upright, comparison result will be more observable that the sliding mode controller works really spectacular. Some of the best virtual results are shown in Table 1, and Figure 1 shows the the drones which are based on three controller methods in the simulator; besides, the results of Simulink with $2.5m/s$ are shown in Figures 6, 7:

A (m/s)	Controller	B (cm)	C (cm)
0.5	SM	8	2
	PID	10	3
	MRAC	15	13
1.5	SM	10	8
	PID	15	12
	MRAC	26	15
2.5	SM	11	13
	PID	14	17
	MRAC	66	82

Table 1: Results of the virtual test. Note that **A** refers to the moving platform velocity, **B** mentions the longitudinal deviation with the platform center, and **C** refers to the lateral deviation with the platform center.

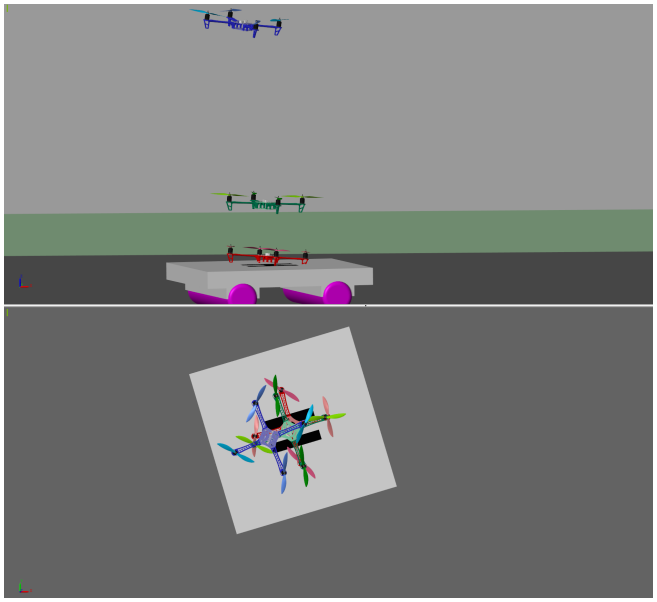


Figure 1: Comparison among virtual drones with three controllers in MATLAB Simscape dynamic space. The red drone refers to PID, the green one to SLD, and blue drone to MRAC controller.

2.5 Experimental Platform

In this submission, we put all our attempts into finding the best practical controller; regardless of any experimental result, the PID works perfect because of fewer coefficients we must tune and thanks to the proper sensors giving feedback of position, velocity, and accelerations. The most challenging section of implementation we faced, could be summarized in localization of the drone and if so very well, we will be able to do various missions. Therefore, we could not test very random cases because a rectangular or a circular random, supposedly, are difficult for our algorithms so we limited deviations. Moreover, when the trajectory of the object is random, its velocity impacts a lot on detection because it might conducive to losing the platform and causing miscalculation even in the landing process of the drone, so we did not test with more than $1.5m/s$ velocity. Some of the best results are shown in Table 2 and Figures 8, 9, 10, 11; also, the landing platform pattern is shown in Figure 2.

A (m/s)	Controller	B (cm)	C (cm)
0.5	PID	7	5
	SM	10	5
	MRAC	25	16
1.5	PID	12	10
	SM	21	12
	MRAC	30	42

Table 2: Results of the practical test in real-world. **A** refers to the moving platform velocity, **B** refers the longitudinal deviation with the platform center, and **C** refers to the lateral deviation with the platform center.

3 CONCLUSION

In this paper, we introduced a fully autonomous quadrotor landing on a moving platform even if it moves on a random path. During the work, three control algorithms are compared to find the best one. To the best of our knowledge, compounding of a PD controller (for inner loop) and SM (for outer one) does better in the simulation, but a little lax in the implementation because of its inordinate coefficients those cannot be tunned practically wholly. Notwithstanding, the fair performance of the SM and MRAC, PID is hardly deniable; this works perfectly in both virtual and real-world.
To continue, we compiled a fantastic visual odometry algorithm (SVO) on the plant for mapping, detecting the ARCode installed on the surface of the moving platform, and tracking. There is no need for any prior information about platform velocity, quadrotor location, and even the path line. No need to a special strategy, when missing the object; just using a simple 2D regeression based on last considerations and estimating a new point as landing target.

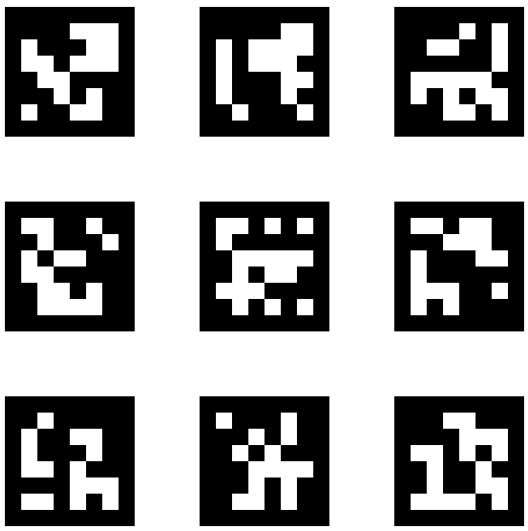


Figure 2: The ARCode icon sheet which is installed on the moving platform to be detected

ACKNOWLEDGEMENTS

The authors would like to thank N. Shahsavari for camparing the PTAM algorithm with our chosen counerpart, A. Yazdanshenas for creating a base CAD model, the Aerospace AUTMAV laboratory adminisrator for the use of a long time indoor testbed, and all of its members for their influential insights.

REFERENCES

[1] Davide Falanga, Alessio Zanchettin, Alessandro Simovic, Jeffrey Delmerico, and Davide Saramuzza. Vison-based autonomous quadrotor landing on a moving platform. In *International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2017.

[2] Matthias Schreier. Modeling and adaptive control of a quadrotor. In *IEEE International Conference on Mechatronics and Automation*, 2012.

[3] Zachary T. Dydek, Anuradha M. Annaswamy, and Eugene Lavretsky. Adaptive control of control uavs: A design trade study with flight evaluation. *IEEE Transactions on Control Systems Technology*, 42(13566251):1400–1406, 2012.

[4] Hanseob Lee, Seokwoo Jung, and David Hyunchul Shim. Vison-based uav landing on the moving platform. In *International Conference on Unmanned Aircraft System (ICUAS)*, 2016.

[5] Tammaso Bresciani. *Modeling, Identification, and Control of a Quadrotor Helicopter*. MSc Theses, Department of Automatic Control, Lund University, 2008.

[6] Andrew Zulu and Samuel John. A review of control algorithms for atonomous quadrotors. *Open Journal of Applied Sciences*, 28(12524328):77–89, 2011.

[7] Eugene Lavretsky and Kevin Wise. *Robust and Adaptive Control*. Springer, Springer Nature Switzerland, AG, 2013.

[8] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometery. In *IEEE International Conference onRobotics and Automation (ICRA)*, 2014.

[9] Georg Klein and David Muuray. Parallel tracking and mapping for small ar workspaces. In *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[10] Bruno Herisse, Tarel Hamel, Robert Mahony, and Francois-Xavier Russotto. Landin a vtol unmanned aerial vehicle on a moving platform using optical flow. *IEEE Transactions on Robotics*, 28:77–89, 2012.

[11] John Olaya, Nicolas Pintor, Dscar F. Aviles, and Juan Chaparro. Analysis of 3 rps robotic platform motion in simscape and matlab gui environment. *International Journal of Applied Engineering Research*, 12:1460–1468, 2017.

[12] Viliam Fedak, Frantisek Durovsky, and Robert Uveges. Analysis of robotic system motion in simmechanics and matlab gui environment. *MATLAB Applications for the Practical Engineer*, 3, 2014.

APPENDIX A: DATA

The proof of Lyapunov statbility comes hereupon, we prove that with the chosen $P \cdot D\cdot$ candiadate function, its derivative will be negetive. The Lyaponov candidtae is similar to one used in [3].

$$\dot{V} = \dot{e}^T P e + e^T P \dot{e} + tr\left(\dot{\widetilde{\theta}}^T \Gamma^{-1} \widetilde{\theta} + \widetilde{\theta}^T \Gamma^{-1} \dot{\widetilde{\theta}}\right) \tag{15}$$

tr in equation (14) refers to the trace function, means the summation of main diagonal terms of the matrix.

$$\begin{aligned} &\rightarrow \dot{V} = (\dot{x} - \dot{x}_m)^T P e + \\ &e^T P (\dot{x} - \dot{x}_m) + \frac{\Gamma^{-1}\tilde{\theta}^T d\left(\tilde{\theta}+\tilde{\theta}\right)}{dt} \end{aligned} \tag{16}$$

$$\begin{aligned} &\rightarrow \dot{V} = \left(\overbrace{A_m x - A_m x_m}^{A_m e} + \overbrace{\dots}^{\alpha\tilde{\theta}}\right)^T P e + \\ &e^T P \left(A_m e + \alpha\tilde{\theta}\right) + 2\Gamma^{-1}\tilde{\theta}^T \frac{d\tilde{\theta}}{dt} \end{aligned} \tag{17}$$

$$\begin{aligned} &\rightarrow \dot{V} = e^T A_m^T P e + \widetilde{\theta}^T \alpha^T P e + \\ &e^T P A_m e + e^T P \alpha\tilde{\theta} + 2\Gamma^{-1}\tilde{\theta}^T \frac{d\tilde{\theta}}{dt} \end{aligned} \tag{18}$$

$$\begin{aligned} &\rightarrow \dot{V} = e^T \overbrace{\left(A_m^T P e + P A_m e\right)}^{-Q} + \widetilde{\theta}^T \alpha^T P e + \\ &\overbrace{\left(\widetilde{\theta}^T \alpha^T P e\right)^T}^{e^T P \alpha\tilde{\theta}} + 2\Gamma^{-1}\widetilde{\theta}^T \frac{d\tilde{\theta}}{dt} \end{aligned} \tag{19}$$

$$\rightarrow \dot{V} = \underbrace{-e^T Q}_{N\cdot D\cdot} + \underbrace{2\widetilde{\theta}^T \left(\Gamma \alpha^T P e + \frac{d\widetilde{\theta}}{dt}\right)}_{=0} \tag{20}$$

To prove strictly negetiveness of the equation (19), $2\widetilde{\theta}^T \left(\Gamma \alpha^T P e + \frac{d\widetilde{\theta}}{dt}\right)$ must equals zero. The first term is $N \cdot D\cdot$ clearly because we have supposed the negetiveness of the Q matrix, before. Hence, either $\widetilde{\theta}^T = 0$ or $\dot{\widetilde{\theta}} = -\Gamma \alpha^T P e$ is true. The former phrase cannot be true so the latter is correct.

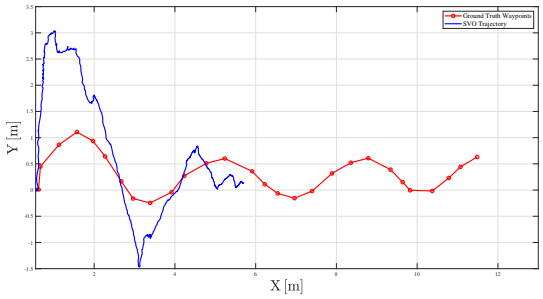


Figure 3: The trajectory of camera vs. the ground thruth path

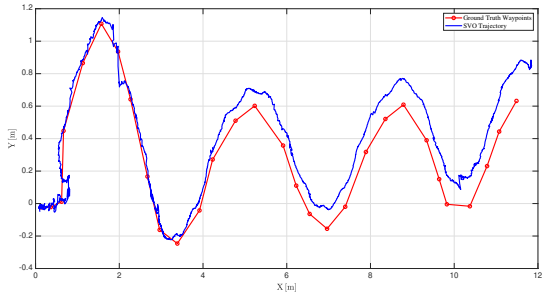


Figure 4: The trajectory of camera + IMU vs. the ground thruth path

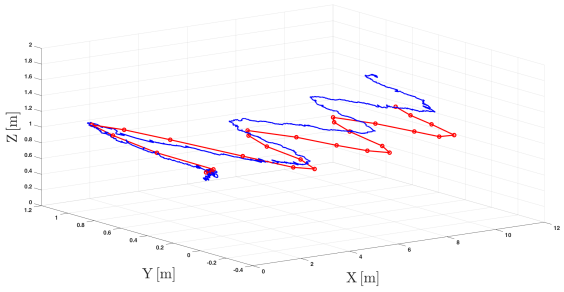


Figure 5: 3D trajectory of camera vs. the ground thruth path

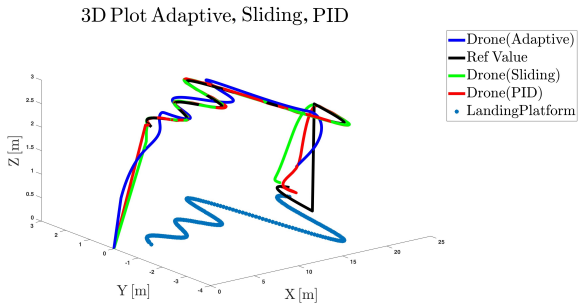


Figure 6: The 3D comparison of three controllers with $2.5m/s$ velocity in MATLAB Simscape

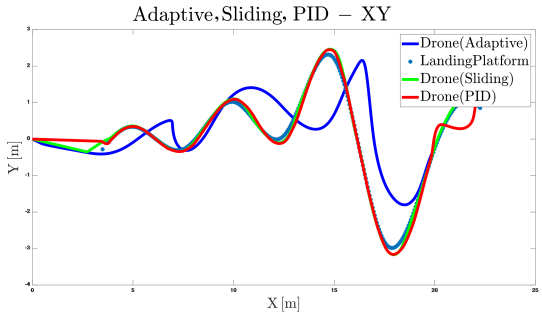


Figure 7: The XY comparison of three controllers to the landing platform with $2.5m/s$ velocity in MATLAB Simscape

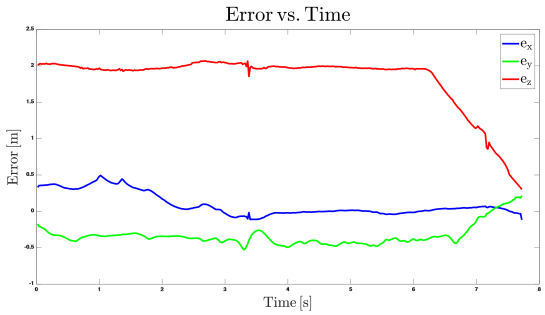


Figure 8: The XY errors of the quadrotor center with the moving platform center with $0.5m/s$ velocity with PID controller

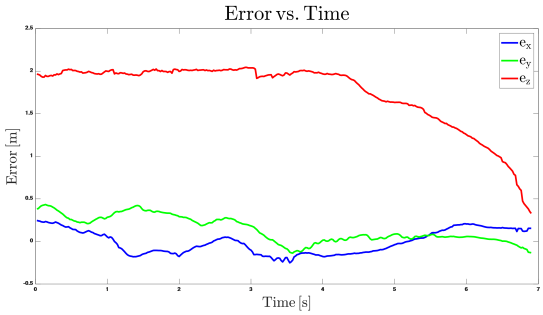


Figure 9: The XY errors of the quadrotor center with the moving platform center in with $0.5m/s$ velocity with sliding controller

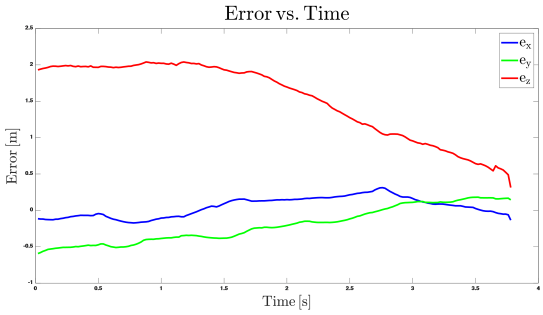


Figure 10: The XY errors of the quadrotor center with the moving platform center with $1.5m/s$ velocity with PID controller

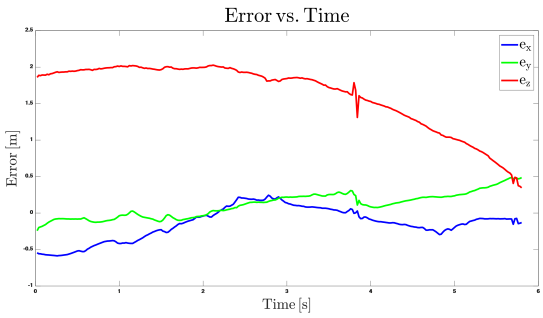


Figure 11: The XY errors of the quadrotor center with the moving platform center with $1.5m/s$ velocity with sliding controller

Design and Testing of a Vertical take-off and Landing UAV optimized for carrying a Hydrogen Fuel-cell with Pressure Tank.

Christophe De Wagter, Bart Remes, Rick Ruisink, Freek van Tienen and Erik van der Horst*
Micro Air Vehicle Lab, Delft University of Technology, Kluyverweg 1, 2629HS Delft, the Netherlands

Abstract

Flight endurance is still a bottleneck for many types of UAV applications. While battery technology improves over the years, for flights that last an entire day, batteries are still simply insufficient. Hydrogen powered fuel-cells offer an interesting alternative but pose stringent requirements on the platform. The required cruise power must be sufficiently low and flying with a pressurized tank poses new safety and shape constraints. This paper proposes a hybrid transitioning unmanned air vehicle that is optimized towards carrying a hydrogen tank and fuel cell. Hover is achieved using twelve redundant propellers connected to a dual CAN network and dual power supply. Forward flight is achieved using a tandem wing configuration. The tandem wing not only minimizes the required wing span to minimise perturbations from gusts during hover, but it also handles the very large pitch inertia of the inline pressure tank and fuel cell very well. During forward flight, eight of the twelve propellers are folded while the tip propellers counteract the tip vortexes. The propulsion is tested on a force balance and the selected fuel-cell is tested in the lab. Finally a testing prototype is built and tested in-flight. Stable hover, good transitioning properties and stable forward flight were demonstrated.

Figure 1: NederDrone2 with 12 propellers of which 8 are fold-able and stop during forward flight. The 4 tip propellers remain active during forward flight and have a higher pitch for efficient fast flight. The large fuselage can accommodate a 9 liter pressure tank and a fuel-cell.

1 Introduction

The advent of Unmanned Air Vehicles (UAV) offers many great new opportunities for surveying and inspection tasks. Many tasks however are requiring flight times of several hours, as well as vertical take-off and landing [1, 2, 3, 4]. To achieve very efficient forward flight, fixed wings have clearly shown to be the most efficient way of flying [5]. But the requirement for a runway or launch and recovery system limits their applicability [6].

Several hybrid concepts have been proposed to merge the advantages of hovering aircraft with efficient fixed wing aircraft [7, 8, 9]. The DelftaCopter [7] has proposed a conventional helicopter rotor combined with delta-wings. While good efficiency was obtained, the concept had a high center of gravity and many single points of failures, which is not ideal when more dangerous fuels are used. [10] has proposed to use coaxial rotors to simplify control and remove the need for tip propellers, but does not solve the issues of the previous concept. Several researchers have proposed tilt-wing UAV [9, 11]. These concepts are great but have difficult control properties and require a complex wing actuation mechanism.

Many tandem tailsitter concepts have been proposed for a long time already [12, 13]. [14] presents the design of a tandem tailsitter and its control. [15] also describes the design and control of tailsitter tandem wing UAV. While the tandem configuration offers good properties for the installation of all hydrogen systems, the fact that it sits upright and can fall over is seen as a problem for a fuel-cell VTOL long endurance

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

aircraft, especially when operating on moving ships.

The current paper presents the NederDrone concept. It consists of an angled tandem wing with 12 propellers for the hover, 8 of which are fold-able during forward flight. The concept was named NederDrone and is shown in Figure 1.

Section 2 explains the design choices behind the concept. Section 3 investigates the required propulsion. Given the design specifications, the selected fuel-cell will be tested in Section 4. Finally Section 5 presents flight test results of the concept using battery power. Conclusions are presented in Section 6.

2 Concept optimization

While the typical application requirements for marine operations are very long flight and vertical take-off and landing, the fuel-cell poses several extra design requirements. Safety is amongst the top requirements. The fuel-cell being fuelled by a 300 Bar carbon pressure tank, avoiding crashes is primordial. This leads to a requirement of redundancy in all flight controls. No single electronic point of failure was allowed is the design.

Hovering is achieved using 12 independent propellers. This allows the failure of at least 2 propellers without endangering the flight. If more propellers are to fail, then the concept can still fly in forward flight, given sufficient altitude at the time of failure. To overcome electrical failures in hover, every Brush-less Electronics Speed Controller (ESC) of every motor receives power from the 2 power busses and can fly with a single power bus. The command cables are also doubled. On top of that, monitoring of all ESC was required. This quickly amounted to an overwhelming amount of control cables. Therefore a dual Controller Area Network (CAN) control bus was designed through the airframe. To convert the commands to normal ESC pulses, special electronics was designed that accepts commands from any CAN bus and sends status information back for health monitoring. The PCB design is shown in Figure 2. The motor controllers are housed inside 3D printed motor mounts made from ABS plastic, which blend nicely into the wing and let the propellers fold nicely over the controller housing (See Figure 11).

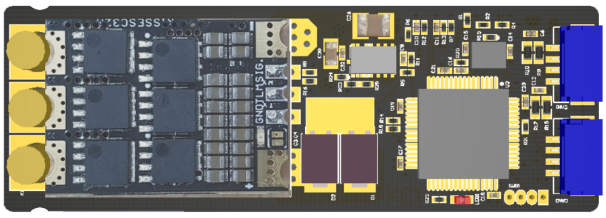


Figure 2: Dual power bus and dual CAN control network brushless electronic speed controller.

Also during forward flight, the heavy, bulky and long hydrogen pressure tank places a lot of constraints on the airframe. The fuel-cell itself also made the fuselage longer. The very large moment of inertia of the fuselage in the pitch direction that results from this spread of mass requires a very large horizontal stabilizer. In hover, the wings can catch turbulence and complicate the hover. To reduce this effect to the minimum, shorter wings are better and create smaller perturbing torques. Both previous constraints lead to the choice of a tandem wing configuration with equal wing span. This maximizes longitudinal stability, minimizes the grip gusts have on the airframe during hover and it yields optimal structural properties.

Finally, to allow a stable passive attitude after the landing, the tailsitter concept was discarded .The long pressure tank would make the risk of tipping over too high. Instead, after landing the fuselage sits stable and flat on the ground. To nevertheless allow autonomous take-off without the need for extra support, the wings were pitched up, hereby slightly pointing the propellers up while on the ground. This makes sure the propeller tips have sufficient clearance from the ground.

Table 1 shows the final design specifications of the NederDrone2. A schematic view is shown in the Appendix Figure 11.

Table 1: Specifications of the Nederdrone2.

	Precision	**Recall**
Wingspan	2.24	m
Length	1.32	m
Airspeed	17	m/s nominal
MTOM	8	kg
c.g.	32	cm from leading edge

3 Propulsion optimization

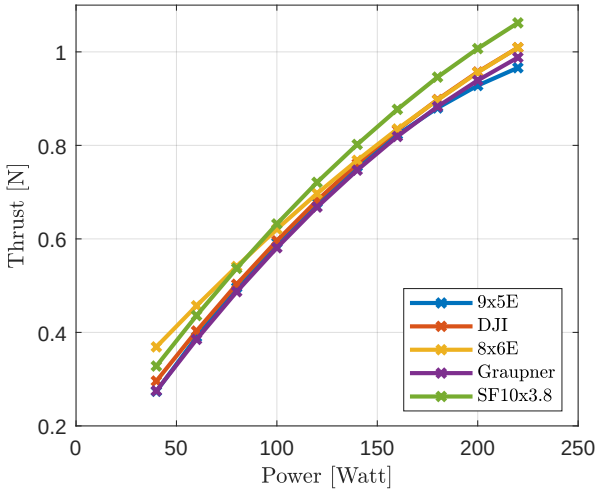


Figure 3: Thrust in function of power for a selected combination of propellers.

One important aspect of the forward flight is that 8 propellers are folded while the 4 tip propellers provide the required thrust. The tip propellers are placed such that they counteract the tip vortexes. But since the choice of foldable propellers is limited, an own folding mechanism was designed. To validate that the selected propeller and motor combination was sufficient for flight, static balance testing was performed.

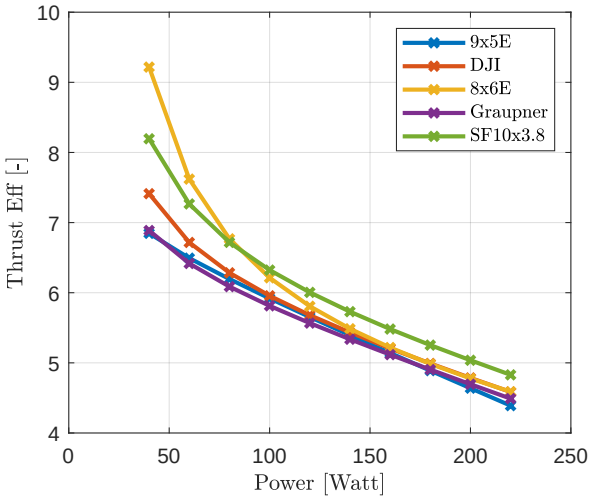


Figure 4: Thrust efficiency in function of power for a selected combination of propellers.

A Hacker A20-38L motor was designed to fit the selected propellers. Figure 3 presents the results of thrust measurements on a static test setup. Figure 4 shows the efficiency estimates associated with it. The selected propeller is the DJI propeller with a custom folding mechanism. The results show it performs almost as well as the best rigid propellers. The total available thrust with 12 motors was shown to be 12 kilograms. This leaves a factor of 50% given the design weight of 8 kg.

4 Fuel Cell Testing

With the airframe and propulsion design figures, a suitable fuel-cell was searched. The Intelligent Energy 800 Watt cell was selected for availability, price and specification reasons. To verify the data-sheet specifications, a laboratory test setup was created in which the power output could be evaluated. Figure 6 shows the test setup with the fuel cell. Specifications of the cell are given in Table 2.

The fuel-cell was found to deliver the 800 Watt reliably. However, when more power than 1100 Watt was used, the total fuel-cell system would shut down. It is therefore crucial to limit the current drawn from the system.

While fuel cells can provide power for a very long time, they provide only little power at a time. To provide sufficient power during the power hungry take-off, landing and hover

Table 2: Specifications of the Fuel-Cell.

	Precision	**Recall**
Max Cont Power	800	Watt
Max Peak Power	1400	Watt
Mass	880	gram
Output voltage	19.6 to 25.2	Volt
Size	196 x 100 x 140	mm

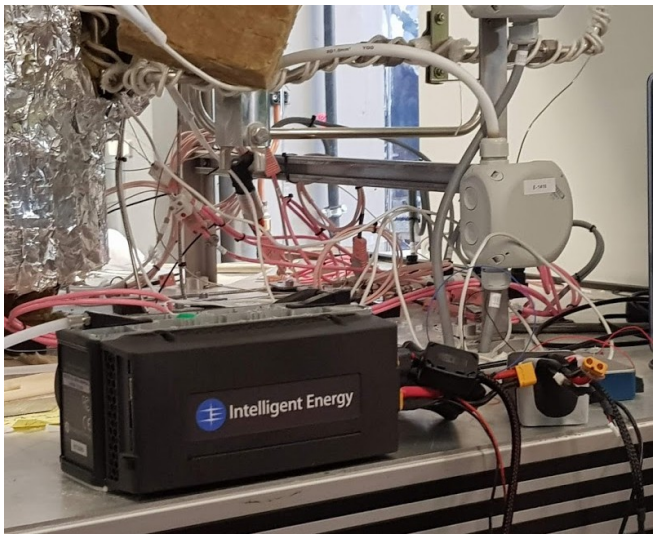


Figure 5: Testing of the fuel-cell in the lab and measuring the current and voltage output under different loads.

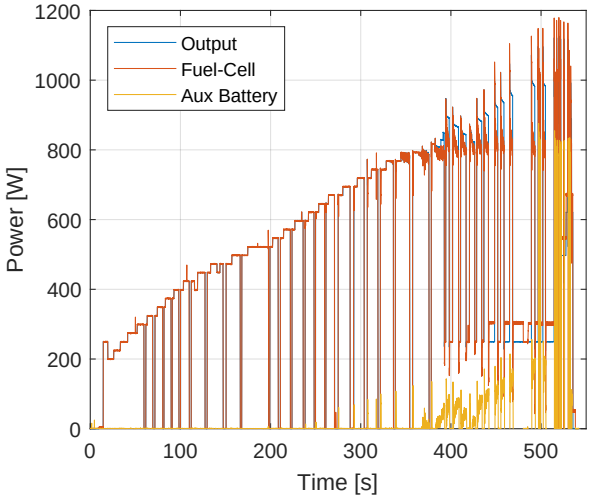


Figure 6: Testing of the fuel-cell in the lab and measuring the output power. When more than 800 Watt is required, the auxiliary battery starts (yellow) to deliver power as well. Every time the power required became larger than 1100 Watt, the system would shut down.

phases, an extra battery is added to the total system. This Lithium-Polymer battery is sized to allow 5 minutes of hovering and is recharged during low power cruise flight when the fuel cell has spare power.

Besides the selection of the fuel-cell, the selection of the tank is a crucial design component. A *CTS Composite Technical Systems* 6.8 Liter 300 bar tank was selected. The weight of the tank is 3.3 kg. At 300 bar it contains 140.7 gram of hydrogen. This results in a system with an efficiency 1415.5 Wh/kg and 4.25 wt%/h2. With a total energy content of 4671 Wh and an estimated 55% fuel-cell efficiency this results in 2569 Wh usable. At the 25V output this results in a 103Ah 6-cell LiPo equivalent.

5 Test flight

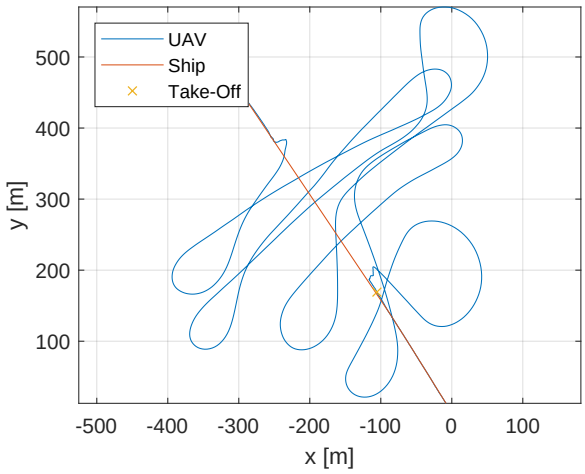


Figure 7: Ground track from a flight from a ship on the North Sea. Stable hover above the moving ship was possible and very stable and smooth forward flight was shown in figures of eight following the moving ship.

The UAV was equipped with a Pixhawk 4 autopilot running Paparazzi-UAV software [16, 17]. The motor controllers equipped with CAN drivers were programmed with an implementation of UAV-CAN with own messages. The datalink consists of a Herelink 2.4GHz + 433MHz (backup), capable of transmitting both video and telemetry. The radio control is a TBS Crossfire Diversity 868MHz.

Before more dangerous test flights are attempted with fuel-cells onboard, the NederDrone was equipped with Lithium-Ion batteries for testing. The hover controller was first tuned in an indoor flight test facility of the TUDelft. Once the hover loop was tuned, the NederDrone was tested outdoors. The hover gains were also good for slow forward

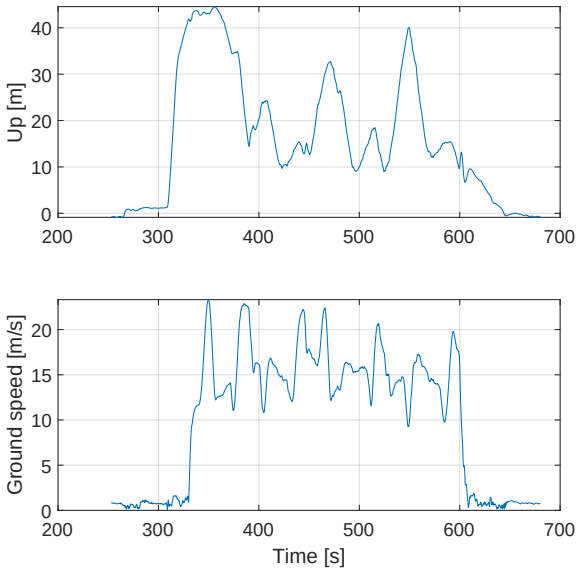


Figure 8: Height and ground speed of a test-flight from a ship.

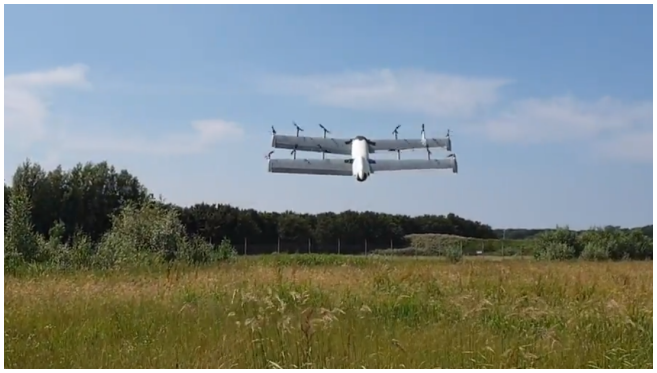


Figure 9: NederDrone2 in-flight.

flight, and for faster flight the forward gains were reduced until stable flight was achieved. Figure 9 shows the NederDrone2 in-flight.

6 Conclusions

A new transitioning tandem wing UAV concept was proposed which is in between a quad-plane and a tailsitter. The tandem wings give it excellent stability despite the huge moment of inertia in the pitch direction due the the long pressure tank and fuel-cell. The orientation of the wing allows very good passive stability when laying on the ground and eliminates the risks of tipping over that are associated with tailsitters. At the same time the NederDrone2 can take-off vertically. The 12 hover propellers give it excellent redundancy and the forward flight capability further increases the resilience to failures in flight. The same propellers can be used during forward flight, where 8 of the 12 propellers fold

back.

7 RECOMMENDATIONS

While the concept was shown to fly very successfully, it has not flown using hydrogen power yet. Many other aspects remain to be investigated in more detail. Test flights with a missing propeller were already performed but a detailed analysis is still needed how many props may fail. Recovering from hover to forward flight is also a maneuver that requires more investigation. Finally, working with hydrogen is a significant operational challenge requiring a lot of research and development.

ACKNOWLEDGMENTS

The authors would like to thank the Royal Netherlands Navy for making this research possible.

REFERENCES

[1] Wei Jin, Hong-Li Ge, Hua-Qiang Du, and Xiao-Jun Xu. A review on unmanned aerial vehicle remote sensing and its application [j]. *Remote Sensing Information*, 1:88–92, 2009.

[2] Telmo Adão, Jonáš Hruška, Luís Pádua, José Bessa, Emanuel Peres, Raul Morais, and Joaquim Sousa. Hyperspectral imaging: A review on uav-based sensors, data processing and applications for agriculture and forestry. *Remote Sensing*, 9(11):1110, 2017.

[3] Ninghao Yin, Ruian Liu, Beibei Zeng, and Nan Liu. A review: Uav-based remote sensing. In *IOP Conference Series: Materials Science and Engineering*, volume 490, page 062014. IOP Publishing, 2019.

[4] Huang Yao, Rongjun Qin, and Xiaoyu Chen. Unmanned aerial vehicle for remote sensing applications—a review. *Remote Sensing*, 11(12):1443, 2019.

[5] Philipp Oettershagen, Amir Melzer, Thomas Mantel, Konrad Rudin, Thomas Stastny, Bartosz Wawrzacz, Timo Hinzmann, Stefan Leutenegger, Kostas Alexis, and Roland Siegwart. Design of small hand-launched solar-powered uavs: From concept study to a multi-day world endurance record flight. *Journal of Field Robotics*, 34(7):1352–1377, 2017.

[6] A Klimkowska, I Lee, and K Choi. Possibilities of uas for maritime monitoring. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 41:885, 2016.

[7] Christophe De Wagter, Rick Ruijsink, Ewoud JJ Smeur, Kevin G van Hecke, Freek van Tienen, Erik van der Horst, and Bart DW Remes. Design, control, and visual navigation of the delftacopter vtol tail-sitter uav. *Journal of Field Robotics*, 35(6):937–960, 2018.

[8] Murat Bronz, Ewoud J Smeur, Hector Garcia de Marina, and Gautier Hattenberger. Development of a fixed-wing mini uav with transitioning flight capability. In *35th AIAA applied aerodynamics conference*, page 3739, 2017.

[9] Marten Schütt, Philipp Hartmann, and Dieter Moormann. Fullscale windtunnel investigation of actuator effectiveness during stationary flight within the entire flight envelope of a tiltwing mav. In G. de Croon, E.J. van Kampen, C. De Wagter, and C. de Visser, editors, *International Micro Air Vechicle Competition and Conference 2014*, pages 77–83, Delft, The Netherlands, aug 2014.

[10] Juan Escareno, Anand Sanchez, Octavio Garcia, and Rogelio Lozano. Modeling and global control of the longitudinal dynamics of a coaxial convertible mini-uav in hover mode. In *Unmanned Aircraft Systems*, pages 261–273. Springer, 2008.

[11] Koji Muraoka, Noriaki Okada, Daisuke Kubo, and Masayuki Sato. Transition flight of quad tilt wing vtol uav. In *28th Congress of the International Council of the Aeronautical Sciences*, pages 2012–11, 2012.

[12] H Stone and KC Wong. Preliminary design of a tandem-wing tail-sitter uav using multi-disciplinary design optimization. In *AUVSI-PROCEEDINGS-*, pages 163–178, 1996.

[13] R Hugh Stone. Modelling and control of a tandem-wing tail-sitter uav. In *Modelling and Control of Mini-Flying Machines*, pages 133–164. Springer, 2005.

[14] Sebastian Verling, Basil Weibel, Maximilian Boosfeld, Kostas Alexis, Michael Burri, and Roland Siegwart. Full attitude control of a vtol tailsitter uav. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3006–3012. IEEE, 2016.

[15] A Alonge, Filippo D'Ippolito, and Caterina Grillo. Takeoff and landing robust control system for a tandem canard uav. In *AIAA/CIRA 13th International Space Planes and Hypersonics Systems and Technologies Conference*, page 3447, 2005.

[16] 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 pp–xxxx, 2006.

[17] Balazs Gati. Open source autopilot for academic research-the paparazzi system. In *American Control Conference (ACC), 2013*, pages 1478–1481. IEEE, 2013.

APPENDIX A: SPECIFICATIONS

Figure 10: A composite photo from a NederDrone 1 prototype in hover and subsequently in forward flight, operated from a ship on the North Sea.

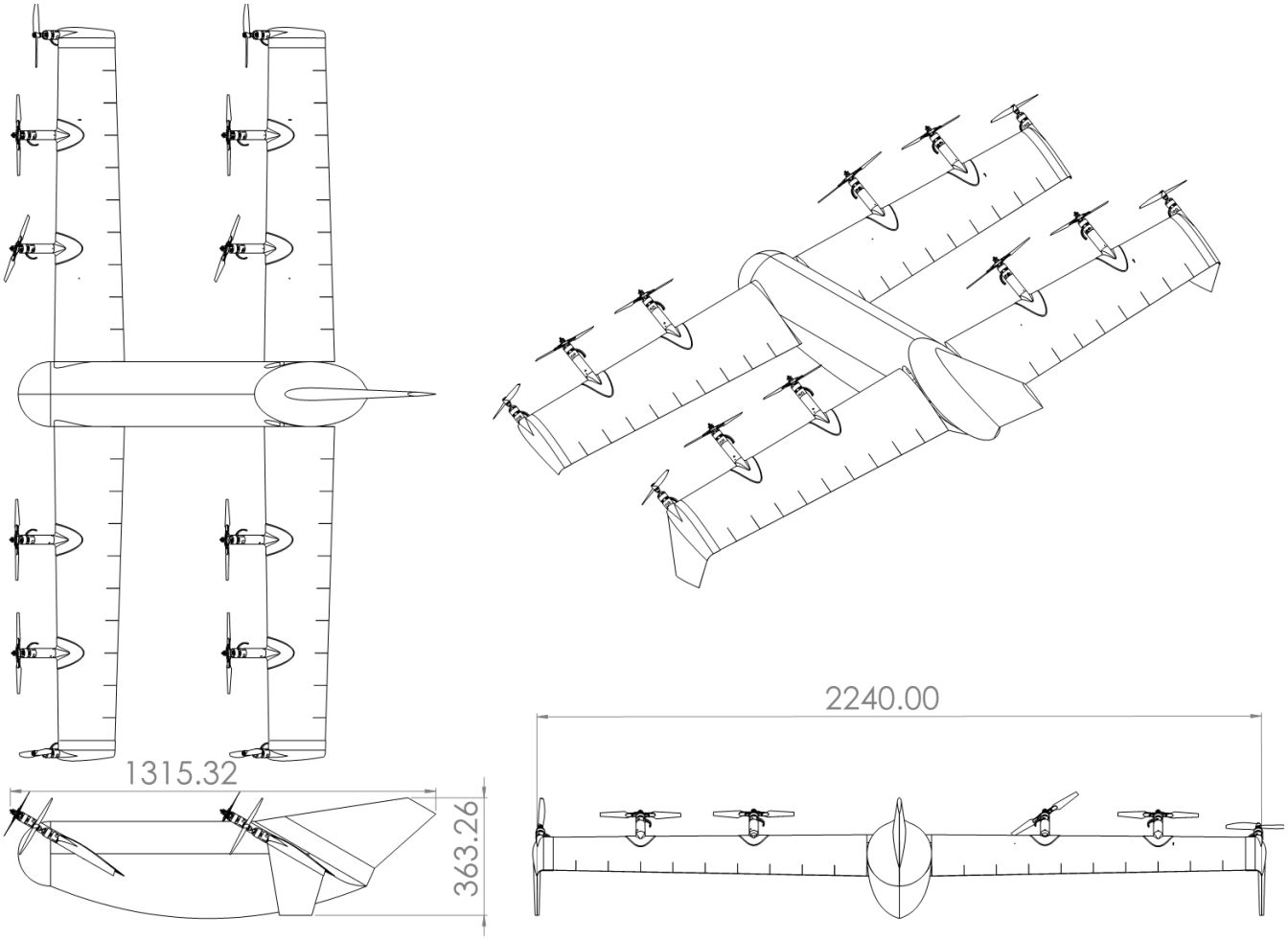


Figure 11: NederDrone2 top, side, back and isometric views. The hydrogen tank forms the main part of the fuselage while the tandem wings are placed at an angle to combine high passive stability on the ground with the possibility of automatic vertical take-off. The span is 2m24 while the length is 1m31.

Trajectory Following with a MAV Under Rotor Fault Conditions

Claudio D. Pose*[1], Francisco Presenza[1], Ignacio Mas[3,4], and Juan I. Giribet[1,2,3]

[1]Facultad de Ingeniería, Universidad de Buenos Aires, Argentina
[2]Instituto Argentino de Matemática "Alberto Calderón" (IAM)
[3]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina
[4]Instituto Tecnologico de Buenos Aires, Argentina

ABSTRACT

Lately, a novel multirotor aerial vehicle capable of handling single rotor failures was presented. When a rotor fails, physically reconfiguring one of the remaining rotors of an hexarotor allows to compensate for maneuverability limitations. In this work, experimental results show the performance of the vehicle in a trajectory-following task in both nominal and fault conditions.

1 INTRODUCTION

Multirotor aerial vehicles have become very popular in recent years, due to the fact that the electronic systems needed to fly them have increased their availability and usefulness, decreasing their cost and weight. Simplicity and cost-effectiveness have turned out to be very appealing and, as a consequence, an increasing number of applications have risen in many fields, such as agriculture, surveillance, and photography, among others. Fault tolerance has been addressed in the literature as a matter of high importance, in particular for multirotor vehicles, see for instance [1, 2, 3, 4, 5, 6] and references therein.

In particular, in [7] is studied the capability of compensating for a rotor failure without losing the ability to exert torques in all directions, and therefore keeping full attitude control in case of failure. For this, at least six rotors are needed, and have to be tilted with respect to the vertical axis of the vehicle. The proposed solution in [7], was tilting the rotor (or arms) of the hexarotor inwards. Experimental results for the proposed solution can be found in [8], where the vehicle takes off, performs different maneuvers and lands successfully with one motor in total failure, maintaining full attitude and altitude control. While the system proved to work correctly, there was a direction that, when exerted torque in, performed noticeably worse with respect to the rest.

To overcome this limitation, in [9] a slight modification was proposed for the vehicle, where, besides tilting the rotors inwards, servomotors were added in two of them to reconfigure their position in case of a failure. Simple experiments were performed with the vehicles in cases with and without failure, in a hovering state and with simple maneuvers, and it was concluded that the new fault tolerant design performed much better than the one proposed and evaluated in [7, 8].

This work presents a more extensive performance evaluation to compare the maneuverability of the vehicle proposed in [9] in cases with and without failure, by means of a trajectory following experiment in an indoor environment.

The manuscript is organized as follows. First, a short description of the proposed vehicle is presented. Then, the characteristics of the vehicle used as a platform for the experiments are described, as well as the setup of the indoor environment where the flights were carried out. Finally, the results obtained are shown and compared for the flights of the vehicle with and without a total failure in one rotor.

2 PROPOSED FAULT-TOLERANT HEXAROTOR

When dealing with total rotor failures in hexarotors, it has been proved that a standard hexarotor configuration (one with the rotors spaced evenly in a plane, pointing upwards, with alternated spinning direction, as in Fig. 1) is not fault tolerant in the event of a failure of this type, in the sense of maintaining control over its four degrees of freedom (rotation around its three axis, and vertical speed). One degree of freedom will be lost, being generally the yaw axis the one chosen to be lost control of, as it allows the possibility to land the vehicle safely. From this point on, fault tolerance will be meant in the sense that the system maintains complete altitude and attitude control.

Suppose a standard hexarotor configuration with $\gamma = 90^o$ (see Fig. 2), which, with the vehicle in hovering mode, suffers a total loss of rotor number 3 (M3), a counter-clockwise (CCW) rotating motor. Then, this rotor no longer generates thrust to produce torque on the x-axis, and neither does it generate torque on the z-axis due to the spinning propeller. Then, turning off the opposite rotor (M6), which generates exactly the opposite torque, is an adequate solution. In this case, the system is not fault tolerant, as there will exist a torque $\mathbf{q}_w = (M_x, M_y, M_z)$ (worst case direction torque) that will require a negative speed from M6 (see [7]), which cannot be achieved. The solution using the inward-tilted rotors with $\gamma > 90^o$, allows M6 to hold the hovering state with a

*Email address: cldpose@fi.uba.ar

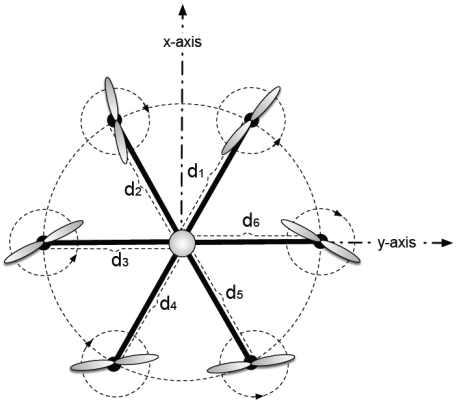


Figure 1: Top view of the proposed vehicle.

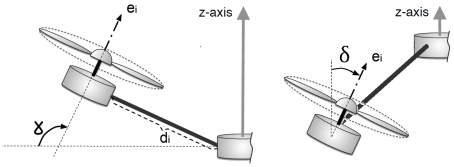


Figure 2: Side view of the proposed vehicle. γ denotes the inward/outward tilt and δ the side tilt.

small positive speed, which in turn allows the vehicle to exert torque in the direction $\mathbf{q}_w$. However, the maximum achievable magnitude of this torque is small, as the maneuver is limited by the saturation of M6. Rotation in the yaw axis is the most stressful maneuver, as they require higher speed variations from the motors with respect to similar maneuvers in pitch or roll.

The work done in [9] proposed to add servomotors in two of the vehicle's arms, in order to tilt the rotors at an angle $\delta \neq 0$ (see Fig. 2), in case of a failure of one of the rotors. By doing this, part of the vertical thrust produced by the rotor is used to generate torque in yaw, allowing to compensate the low maneuverability in that axis. Which rotor will be tilted, and the magnitude and direction of the tilt angle will depend in which of the six rotors is under failure.

3 EXPERIMENTAL SETUP

To provide a comparison of flight performance between the hexa-rotor in a nominal and a failure state, two identical experiments were carried out. An identical fixed trajectory to follow is given both for the case of the vehicle without failure, and for a case where rotor 3 is under total failure.

The vehicle used for the experiments is based on a commercial model. The frame is the DJI-F550, with a distance between rotors of 550mm. The actuators installed on this frame are T-Motor 2212-920KV motors, with 9545 plastic self-tightening propellers, driven by 20A electronic speed controllers (ESC). The battery used is a 4S 5000mAh 20C LiPo that allows approximately 15 minutes of hovering flight (without failures). The flight computer used is a custom-designed board [10] developed by the GPSIC Lab [11] to support experiments that are usually carried out on this kind of vehicles. It is based on the LPC-1769 microcontroller, an ARM Cortex M3 that runs at 120MHz, and several sensors such as the MPU-6000 IMU, the HMC5883L digital compass and the BMP180 barometer, sending flight information to MATLAB (for data analysis) through a 57600bps XBee wireless connection. The control loop runs at 200Hz, where the pitch, roll, and yaw angles are estimated and a PID control algorithm calculates the torque for vehicle stabilization. Then, the allocation algorithm gives the force of each motor in order to achieve the desired torque, and a simple function converts this value into the PWM signals commanded to the ESC. Two additional PID control loops are used for position control in the *XY* plane, where the input is the error in position, and the output actuates over the pitch and roll commands. One last PID control loop is used for height control, actuating directly on the vertical thrust command.

To switch between the different configuration of the rotors for the nominal and failure case, a servomotor is added in rotor 1, that tilts it over the arm's axis at $\delta_1 = 0^o$ for the vehicle without failures, and at $\delta_1 = 10^o$ in the case of a failure in rotor 3, as shown in Figure 3.

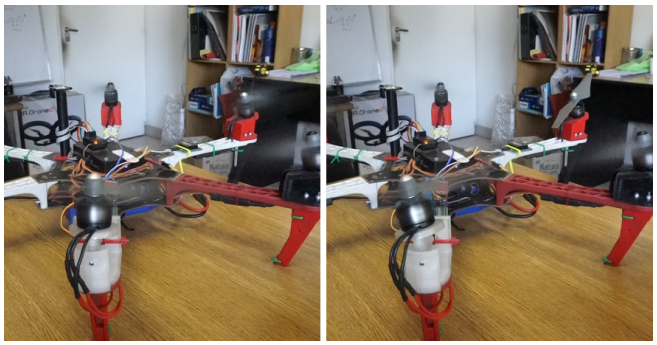


Figure 3: Servomotor in rotor 1 in the case without failure (left) and in the case of a failure in rotor 3 (right).

In order to provide position information, an ultrasonic-based indoor navigation system from Marvelmind was used. This system consists of a network of stationary ultrasonic beacons interconnected via radio interface, one mobile beacon installed on the vehicle to be tracked, and one central modem that calculates the position of the mobile beacon. For the experiments, four stationary beacons were placed in a square with a side length of 8m, 40cm above the floor, as shown in Fig. 4.

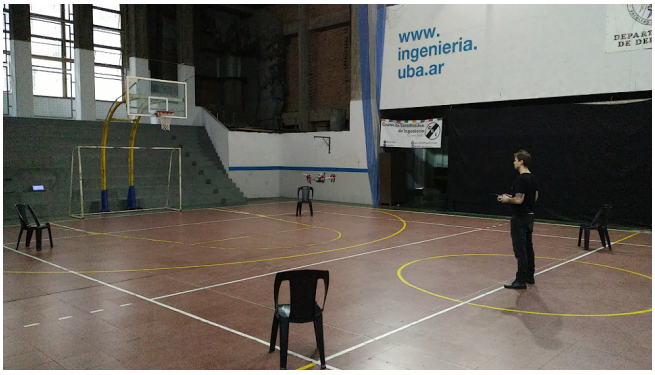


Figure 4: Environment setup for the experiments. The stationary beacons are placed on chairs at a height of 40cm, in a square of 8x8m.

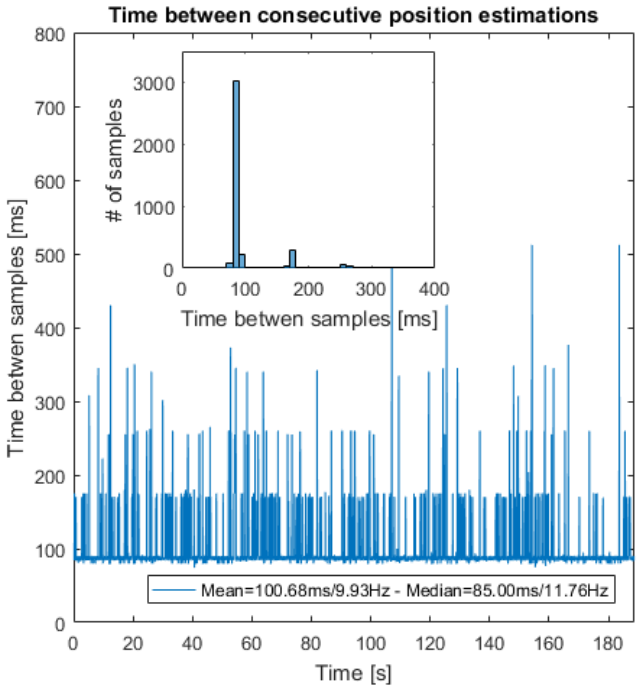


Figure 5: Time between consecutive position estimations from the indoor positioning system, during one of the flights. (Inset) Histogram of the time plot, using 10ms intervals.

The system was configured to provide position estimation at a 12Hz rate, but may not provide data (or provide data with low accuracy) in cases where the line of sight between the mobile beacon and the stationary beacons is obstructed. In Figure 5, the time between consecutive samples of position information (accurate or not) is shown, during one of the flights of the experiments. The data rate of the positioning system is mostly stable at 85ms (around 12Hz), but it can be observed that there are several occasions where this time is doubled, corresponding to a failure to obtain position information. An inset of axis shows the histogram of the same experiment, where around 90% of the samples correspond to a time interval of 85ms$\pm$10ms.

The chosen path for the experiments was the Gerono trajectory (or "infinity" trajectory) in the XY plane. The yaw direction was fixed at zero during the entirety of the experiments, so that a maneuver in pitch moves the vehicle along the X axis, and a maneuver in roll moves it in the Y axis. The vertical thrust remained manually controlled by the pilot for safety reasons. The vehicle takes off from the ground, is positioned around the center of the flight area, and the position control is activated. In the moment of activation, the current position is taken as the center of the Gerono trajectory.

4 RESULTS

The flight trajectory for the vehicle without failures is shown in Figure 6. The vehicle takes off at $t = 0s$, and the position control is activated at $t = 33s$, where the current position is taken as reference. The vehicle performs three and a half full Gerono trajectories, before the position control is deactivated at $t = 190s$, where it lands safely. It can be observed that there are several outliers in the measured position at $t = 72s$, $t = 109s$, $t = 126s$ and $t = 184s$, that correspond to errors in the position calculation of the Marvelmind tracking system, to which the vehicle reacts accordingly, but recovers quickly and remains on path.

In Figure 7, the PWM values commanded to the six rotors are shown. As expected for a nominal case of a hexarotor in a near-hovering situation, all the PWM values are almost equal, driving the rotors at around 50% of their maximum speed, which provides a wide margin for performing maneuvers without saturating any rotor.

The flight trajectory for the vehicle with a failure in rotor 3 is shown in Figure 8. The failure is activated before take-off, and is present during the full flight. The vehicle takes off at $t = 0s$, and the position control is activated at $t = 18s$, where the current position is, again, taken as reference. The vehicle again performs three and a half full Gerono trajectories, before the position control is deactivated at $t = 176s$, and is returned to the take-off point to land. In this test, there were no occurrences of glitches in the position estimation.

It can be noticed that in the Y-axis sinusoidal trajectory, there is some overshoot in the positive direction, while there is no significant overshoot in the negative direction. This is because the rotor in failure state is positioned over the Y axis of the vehicle, and the roll maneuver in one direction seems to be less responsive than in the other.

In Figure 9, the PWM values commanded to the six rotors are presented for the case with failure. While the PWM value for rotor 3 is zero during the flight, the commands to the rest of the rotors do not significantly differ for the previous case.

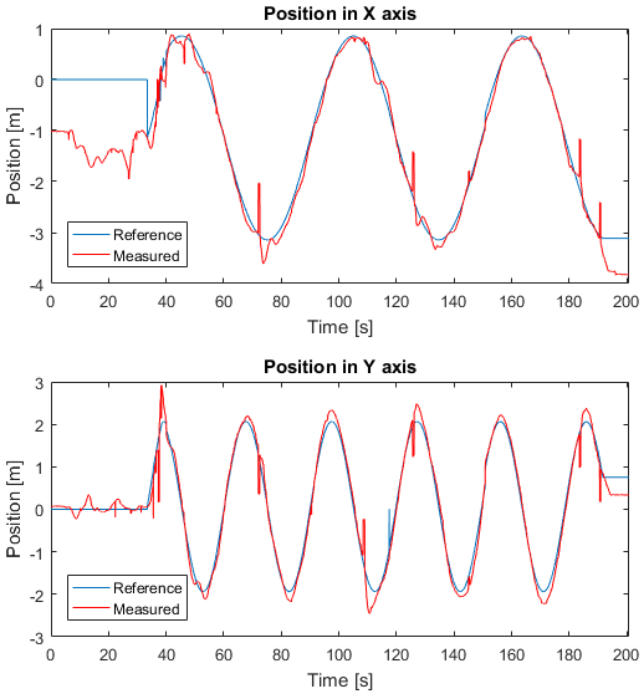


Figure 6: Gerono trajectory for an hexarotor without failure.

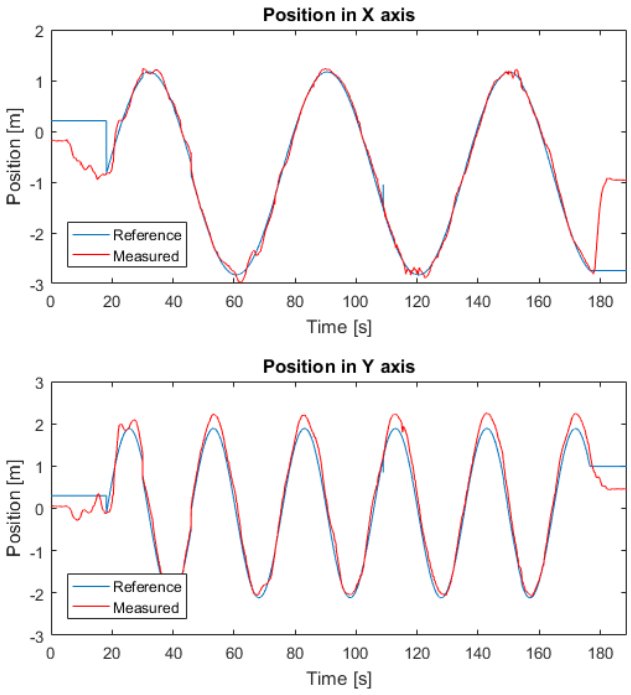


Figure 8: Gerono trajectory for an hexarotor with a failure in rotor 3.

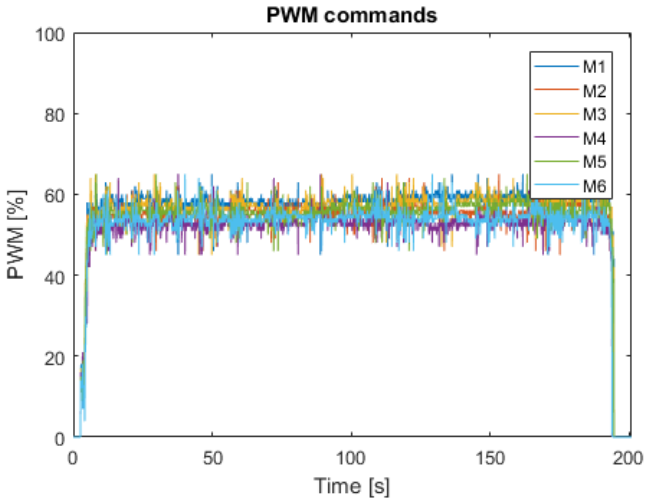


Figure 7: PWM values during the flight without failure

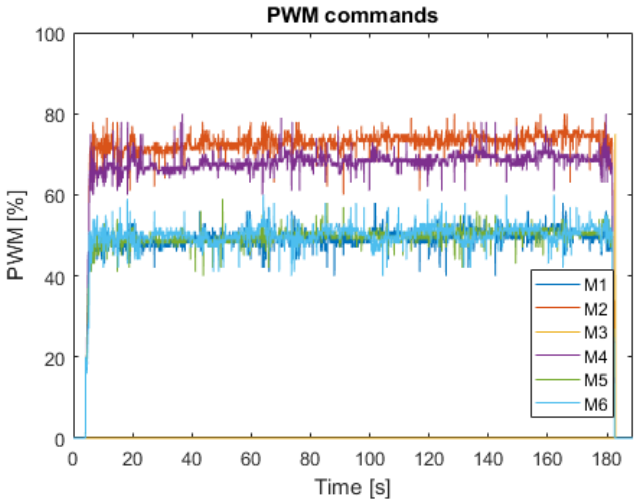


Figure 9: PWM values during the flight with a failure in rotor 3.

Rotors 2 and 4 increase its speed (and thus its thrust), to compensate for the lack of thrust of the motor located between them. All the maneuvers performed during the trajectory do not require a great variation of speed (the PWM values for all the rotors only vary around $\pm 5\%$). This suggests that the vehicle with failure also is able to perform aggressive maneuvers without saturating the rotors.

Both cases performed satisfactorily, as the trajectory was correctly followed. Moreover, during manual take off, landing, and the diverse maneuvers made to position the vehicle for the experiments, it was not noticeable any difference in maneuverability between both cases, even while performing very aggressive movements to test the system. A video of the

preliminar test of the fault tolerant vehicle can be sen in [12], and another of one of the experiments is attached to this work.

5 CONCLUSION

The proposed hexarotor vehicle was able to follow with good performance a given trajectory, both in a nominal case, and with a total failure in one rotor. Moreover, there is no appreciable difference in the behaviour between both cases, as all the rotors operate at a speed pretty far away from their saturation limits, giving plenty of margin for different maneuvers.

Still, the failure case shows slight asymmetries in the trajectory, indicating that there are some maneuvers that are performed better than others. This may be caused either by the rotors working at different average speeds, or by the maneuver requiring different speed variations from each rotor.

ACKNOWLEDGEMENTS

This work has been sponsored through the UBA-PDE-18-2019 grant from Universidad de Buenos Aires, Argentina, Instituto Tecnologico de Buenos Aires ITBACyT Interdiciplinario DT13/2018, and PICT 2016-2016 grant from Agencia Nacional de Promoción Científica y Tecnológica (Argentina). Claudio Pose would like to thank the Peruilh Foundation, whose grant made this research possible.

REFERENCES

[1] M. Saied, B. Lussier, I. Fantoni, C. Francis, H. Shraim, and G. Sanahuja. Fault diagnosis and fault-tolerant control strategy for rotor failure in an octorotor. *IEEE International Conference on Robotics and Automation*, pages 5266–5271, 2015.

[2] D. Vey and J. Lunze. Structural reconfigurability analysis of multirotor UAVs after actuator failures. *54th Conference on Decision and Control*, pages 5097–5104, 2015.

[3] Duc-Tien Nguyen, David Saussie, and Lahcen Saydy. Fault-tolerant control of a hexacopter uav based on self-scheduled control allocation. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 385–393, 2018.

[4] D. Vey and J. Lunze. Experimental evaluation of an active fault-tolerant control scheme for multirotor UAVs. *3rd International Conference on Control and Fault-Tolerant Systems*, pages 119–126, 2016.

[5] G. P. Falcon, V. A. Marvakov, and F. Holzapfel. Fault tolerant control for a hexarotor system using incremental backstepping. *IEEE Conference on Control Applications (CCA)*, pages 237–242, 2016.

[6] M. W. Mueller and R. D'Andrea. Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 45–52, May 2014.

[7] Juan I Giribet, Ricardo S Sanchez-Peña, and Alejandro S Ghersin. Analysis and design of a tilted rotor hexacopter for fault tolerance. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4):1555–1567, 2016.

[8] Juan I Giribet, Claudio D Pose, Alejandro S Ghersin, and Ignacio Mas. Experimental validation of a fault tolerant hexacopter with tilted rotors. *International Journal of Electrical and Electronic Engineering and Telecommunications*, 7(2):1203–1218, 2018.

[9] C. Pose and J. Giribet. Fault tolerance analysis of an hexarotor with reconfigurable tilted rotors (preprint). *58th Conference on Decision and Control*, 2019.

[10] L. Garberoglio, M. Meraviglia, C. D. Pose, J. I. Giribet, and I. Mas. Choriboard III: A Small and Powerful Flight Controller for Autonomous Vehicles. In *2018 Argentine Conference on Automatic Control (AADECA)*, pages 1–6, Nov 2018.

[11] Grupo de Procesamiento de Señales, Identificación y Control, Facultad de Ingenieria, Universidad de Buenos Aires. `http://psic.fi.uba.ar`.

[12] Fault tolerant control of a double-tilted hexacopter. `https://www.youtube.com/watch?v=der3QUWz4Vg`.

A Tailless Flapping Wing MAV Performing Monocular Visual Servoing Tasks

D.A. Olejnik *, B.P. Duisterhof †, M. Karásek, K.Y.W. Scheper, T. van Dijk and G.C.H.E. de Croon
MAVLab, Control and Operations, Faculty of Aerospace Engineering, TU Delft

ABSTRACT

In the field of robotics, a major challenge is achieving high levels of autonomy with small vehicles that have limited mass and power budgets. The main motivation for designing such small vehicles is that, compared to their larger counterparts, they have the potential to be safer, and hence be available and work together in large numbers. One of the key components in micro robotics is efficient software design to optimally utilize the computing power available. This paper describes the computer vision and control algorithms used to achieve autonomous flight with the ∼30-gram tailless flapping wing robot, used to participate in the IMAV 2018 indoor micro air vehicle competition. Several tasks are discussed: line following, and circular gate detection and fly-through. The emphasis throughout this paper is on augmenting traditional techniques with the goal to make these methods work with limited computing power while obtaining robust behaviour.

1 INTRODUCTION

Recently, there has been a growing interest in developing autonomous micro aerial vehicles (MAVs) due to their agility and inherent safety. However, limited on-board processing and sensory information still pose a challenge for the real-time robot operations in a complex environment.

A primary role in the attitude and position determination of MAVs is played by accelerometers, gyroscopes, inertial measurement units (IMU) and global positioning system (GPS). Unfortunately, the mentioned sensors tend to be noisy and drift with time. In indoor environments position information can be also obtained from motion tracking systems like Vicon or OptiTrack. Although these systems are highly accurate, they are not feasible for larger spectrum of applications. As an alternative, an on-board camera can be utilized to complement state estimation.

In many applications, stereo vision system is preferred due to depth measurements that do not suffer from scaling ambiguity. However, a single camera is lighter and requires less power than two cameras. Especially, when it comes to MAVs, lightweight and power efficient solutions are the most desirable.

Monocular visual servoing has a great potential that is currently exploited by many researchers [1, 2, 3, 4]. In [5], desired heading of a drone flying indoors is established based on longitudinal lines. In a similar fashion [6] makes use of line correspondence using images of a corridor and window to calculate the position and heading of a MAV. The paper [7] applies visual servoing to power lines inspection, where guidance of drone relies on extracted information from images with linear features. Alternatively, [2] discusses position control for MAVs using circular landmark. Here, camera pose estimation is based on a geometric approach and derived from an epileptic appearance of a circle in a perspective projection.

In this paper, we present a monocular vision based approach for visual servoing tasks such as line following and precise flight through a set of increasingly small hoops. We propose simple algorithms to perform these flight elements that reduce computational effort and enhance robustness. Our main contribution is an implementation of those strategies on a flapping wing MAV with limited on-board processing and sensory information. Finally, we demonstrate autonomous flight capabilities of the MAV in the indoor competition of the International Micro Air Vehicle Conference and Competition (IMAV2018).

2 SYSTEM OVERVIEW

2.1 The DelFly Nimble

The vehicle used in the competition was the DelFly Nimble [8], the latest flapping wing MAV developed within the DelFly project [9]. Compared to its predecessors, which were stabilized and controlled by a tail with conventional control surfaces [10, 11], the Nimble is a tail-less design. The flapping wings are thus not only used to generate sufficient lift and thrust. The vehicle is, similar to insects, controlled by adjustments of the motion of the individual wings.

The tailless concept has many advantages over tailed designs. It allows for fully controlled hovering flight as well as flight in any direction: up/down, left/right or forward/backward. This opens up new possibilities in autonomous flight of FWMAVs. Unlike the tailed autonomous DelFly versions [12, 13], the vehicle can fly sideways e.g. to align itself with the center of a window before flying through it. It can also stop and turn around, when reaching a dead end,

*D.A.Olejnik@tudelft.nl
†b.p.duisterhof@student.tudelft.nl

which was not possible before as the tailed vehicles needed to maintain a minimal forward velocity to stay airborne.

For the competition, we equipped the Delfly Nimble with the VL53L0X range sensor and a custom built mono-camera system, which uses the same hardware as the stereo vision system of the Delfly Explorer [11]. The VL53L0X is a Time-of-Flight (ToF) laser-ranging module providing absolute distance measurement up to 2m. The sensor weighs only 0.54 gram. The custom-made camera module with STM32F405 processor for onboard vision processing weighs 2g and reaches a clock-speed of 168 MHz with 192 kb of RAM [1]. Together both lightweight sensors allowed us to carry out flight tests with height estimation and shape recognition. Finally, an ESP8266 ESP09 WiFi module was installed to provide a bi-directional datalink. This was invaluable during the testing, as it provided live telemetry and allowed online tuning of the various control parameters.

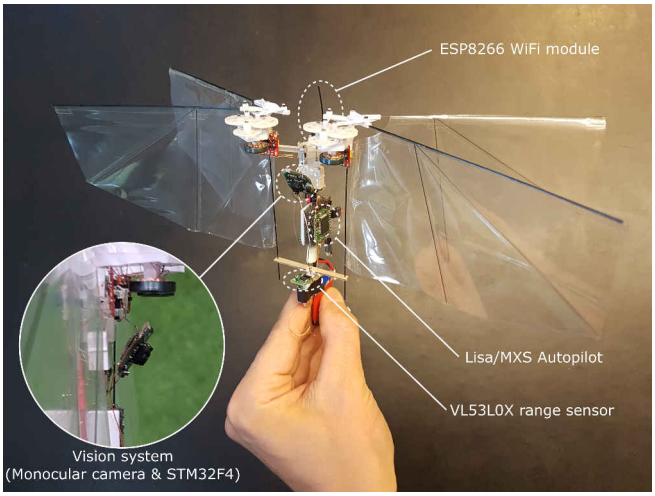


Figure 1: DelFly Nimble tailless flapping wing MAV configured for the IMAV 2019 competition.

The final vehicle configuration, with a total weight of 29.92 g, is in Figure 1. The camera system was mounted via a thin metal strip, which allowed to manually adjust the camera angle according to the task needs. For the line following task, the camera angle was chosen in a trade-off between more information about the future or close to the vehicle. If we point the camera more up, we can look further into the future, but the area close to the DelFly is not visible anymore. For the circle detection task, the camera is placed such that it is looking straight forward. This, in effect, is dependent on the speed at which we fly, as the pitch angle changes with velocity. Because the vehicle was most of the time operated at slow forward flight, the laser range sensor, placed at the bottom of the vehicle, was oriented to point down and slightly forward ($\sim$ 15 degrees) with respect to the vertical body axis.

[1]https://www.st.com/en/microcontrollers-microprocessors/stm32f4-series.html

2.2 Control of a flapping-wing MAV

2.2.1 Attitude stabilization

Tailless flapping wing MAVs are, like multicopters, inherently unstable and require active attitude stabilization. For this, the vehicle was equipped with an open-source STM32F4-based Lisa/MXS autopilot[2] running the open-source Paparazzi UAV autopilot system[3]. The autopilot board was mounted on a soft-mount consisting of PU foam blocks in order to prevent saturation of the on-board 6DOF IMU (MPU 6000) signals. The attitude was stabilized by a standard PD controller with additional low pass filtering, more details can be found in reference [8]. Although no magnetometer was present, the drift of the estimated heading was relatively slow and would typically be just a few degrees over the time needed to complete the competition task. Moreover, in the tasks like line following, the vehicle controls its heading relative to the line direction and an accurate absolute heading was thus not needed.

2.2.2 Height control

Although the autopilot board is equipped with a barometric pressure sensor, this sensor alone proved to be insufficient for precise height control. Even in ideal indoor conditions, the vehicle did oscillate more than +/- 0.5 m from the set point. Thus, for height estimation we have fused the pressure reading with the laser ranger reading using a complementary filter. Nominally, the absolute laser ranger reading was given a high weight. This weight was lowered when very high climb/descend rates were seen by the laser, typically when flying over an obstacle. In such situation, more trust was given to the pressure sensor. Finally, only low-pass filtered pressure based estimate was used when no valid laser measurements were available, e.g. when the floor reflection was insufficient, or when the vehicle got out of the laser sensor range.

The laser ranger worked reliably up to 1.5 m on the multiple floor surfaces where we were testing prior to leaving to the competition in Australia. Unfortunately, as we have noticed at the competition site, the sensor does not work in direct sunlight (high IR light content). And, it also does not work well on tarmac (an unexpected floor material of the indoor competition), which does not reflect enough light back. Thus, we were forced to use wind shades to limit the amount of sunlight and covered the floor with blankets that would reflect the IR light.

In order to maintain a leveled flight we are using a PI control to minimize the error between the desired and measured height. Furthermore, because the demand for throttle level increases as the battery discharges, the altitude controller included also a battery-level dependent feed-forward control

[2]https://wiki.paparazziuav.org/wiki/Lisa/MXS_v1.0
[3]http://wiki.paparazziuav.org/

based on measurements in Figure 2.

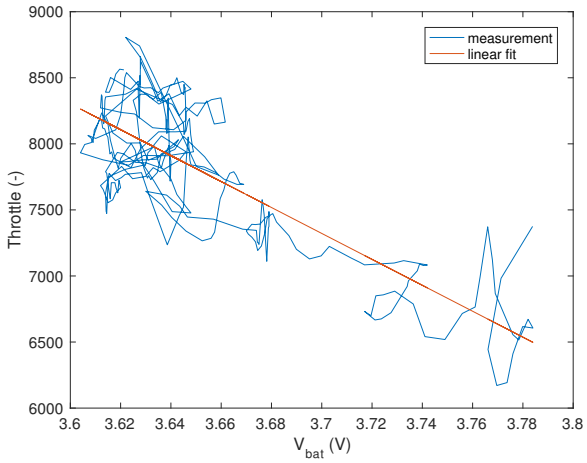


Figure 2: Throttle level at near hover against battery voltage; experiment (blue) and linear fit (red).

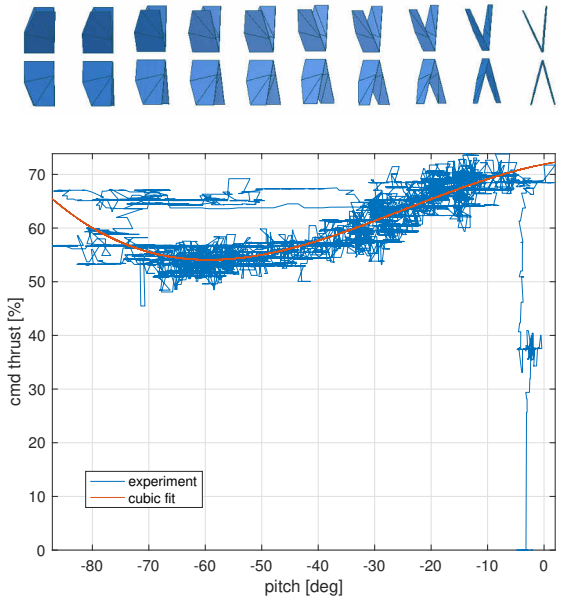


Figure 3: Throttle level against body pitch when flying in the wind tunnel with increasing wind speed and maintaining approximately levelled flight; experiment (blue) and cubic fit (red); the robot's body posture is shown from top view.

Near hover, tail-less flapping wing MAVs are comparable to quadrotors. The vehicle stays airborne thanks to the thrust generated due to flapping motion. Higher thrust, and thus climbing, can be achieved by higher flapping frequency. Flying forward and sideways is achieved by titling the entire MAV, and thus by titling the thrust vector, forward and sideways, respectively.

In forward flight, flapping wing MAVs generate additional lift due to a change in body posture and the oncoming airflow, where the wing surface acts as an airfoil. Thus, higher pitch angles require less throttle in order to achieve the same lift force. These flapping-wing-specific coupling effects were characterized while flying in a wind tunnel [14] (Figure 3) and were accounted for in the feed-forward controller in the form of a cubic fit.

2.3 Competition Tasks

The competitions were focused around topics like aircraft efficiency and innovative designs, light and small MAVs, autonomy and image processing. The indoor mission accommodated a flight through windows, hoops, and following a predetermined flight path. The detailed map of the indoor competition is shown in Figure 4.

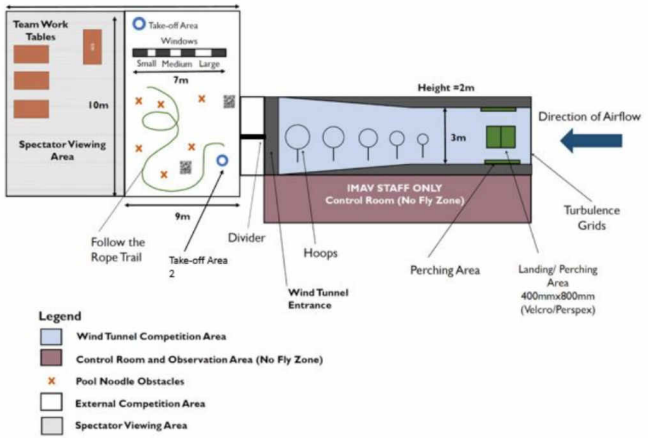


Figure 4: Indoor competition map.[4]

In this paper we will focus on two tasks: line following and precise flight through a set of increasingly small hoops. To complete the first task the MAV had to follow the rope all the way to the end and navigate around obstacles. The rope used during the challenge had high contrast against the floor. The obstacles were represented as green poles fixed in dark blue buckets. The second task was performed in the wind tunnel test section. Five hoops of different sizes, starting with largest, and getting smaller were placed in equal distances. Points were awarded for each hoop flown through. The mission could have been carried out in various wind conditions. Due to limited gust rejection capabilities of our platform the decision was made to fly without wind. The final setup of the indoor competition is shown in Figure 5.

3 LINE FOLLOWING

The goal of this task is to follow a line in an unknown environment. The line can have any shape or curvature which makes accurate line following essential. A test setup was created in TU Delft's Cyberzoo, depicted in Figure 6.

[4] imav2018.org

Figure 5: A screenshot from a video stream of the IMAV's Indoor Competition which shows the DelFly Nimble performing the task of precise flight through a set of increasingly small hoops.

Figure 6: Line Following test setup.

3.1 Perception

This was a best case scenario, as the 'line' is of uniform colour and much thicker than what is expected in the competition.

3.1.1 Segmentation

The fact that the colour of the line is known can be used to our advantage. The least computationally expensive method to determine line position is to do straightforward per-pixel segmentation. However, segmentation of the entire image would be too heavy for the on-board processor and would result in low frame rates [15]. Sub-sampling is used to reduce the computational effort, resulting in a set of pixels that lie on the line. This set of pixels is the starting point for the algorithms considered.

3.1.2 Centroid

As the DelFly Nimble had never been flown autonomously before, the team started off by implementing a straightforward strategy. One of the most straightforward ways to perform the task utilizes the centroid of the line. The coordinates of this centroid in the image frame can then be used as an input for the control system.

Various subsampling methods have been considered, being 1) selecting an upper or lower fraction of the image, 2) selecting evenly spaced or randomly selected rows along the image and 3) using randomly selected pixels over the entire image. All lead to similar results, but the third strategy seems to be the best approach, as every pixel gets the same chance to land in the subset. That is, in theory, the resulting set of points is most representative of the real world.

Even though this algorithm is relatively simple, it already led to promising results. The DelFly Nimble was able to follow the circle fully autonomously. The main shortcoming here was that even though the DelFly would follow the circle, tight corners would cause the system to stop tracking the line. The reason for this is that once a large portion of a turn enters the camera's field of view (FOV), the position of the centroid becomes a less ideal control input. Figure 7 shows a curve in the line as an example, where the vehicle is tempted to steer into the curve due to the c.g. position. This will result in this corner to be 'cut', that is, the vehicle will stop flying directly over the line. Considering the obstacles surrounding the line, accurately following of the line will be essential.

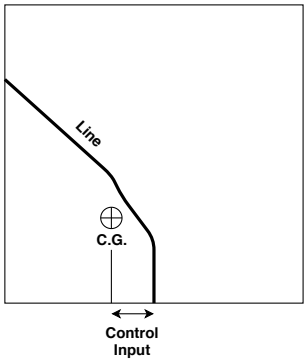
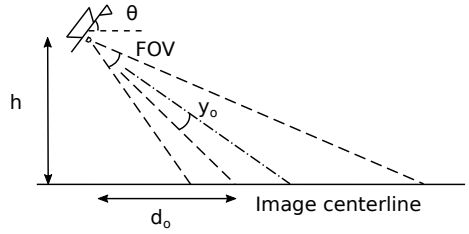


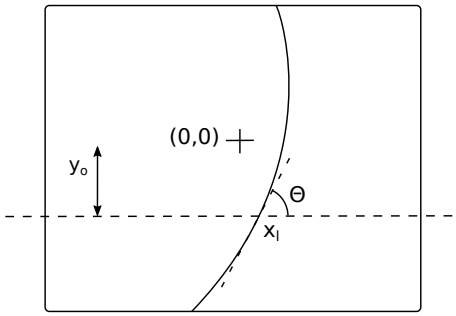
Figure 7: C.G. of the line would trigger the control system to start rotating left, which would cause the corner to be 'cut'. More precise following of the line is required.

3.1.3 Line fit

The centroid-based approach was demonstrated to be a first functional algorithm, while a more accurate algorithm was desired. Until now we have been looking at the centroid only, while more information of the line would be beneficial. Ideally, we would want to know the position and orientation of the line directly beneath the vehicle and translate that to yaw and roll commands. In that way, the vehicle is flying



(a) Side view of DelFly in flight showing the projected image on the ground.



(b) Sample resultant second order line fit in the image plane

Figure 8: Graphical depiction of error terms used for line following control.

above the line at all times and not cutting corners.

Using the points (pixels) generated before, a second order line fit can be done. As the FOVs, attitude and altitude of the camera are known, it is possible to extrapolate the line fit outside of the image.

After having implemented this strategy, it turned out that it was impractical to perform extrapolation. The fitted line would quickly diverge from the line in the real world and an unusable detection was the result. Because of this reason, it was chosen to compute the position and orientation of the line at some point in the lower half of the image. In other words, for a chosen value along the vertical y-axis of the image, we compute the position and orientation of the line. Depending on speed and camera orientation, this point can be moved along the vertical y-axis of the image.

One final improvement is that, for every pixel that is found on the line, the algorithm will search in all 4 directions to find more pixels that meet the colour filter. In this way, more pixels will be found at low cost. This is the strategy that was finally used in the IMAV 2018 indoor competition.

3.2 Control

Figure 8 shows a sample resultant second order line fit as described in the previous section. This figure depicts a DelFly flying at height h and body pitch angle θ. With this, we can define the parameter y_0 as the angular vertical offset from the center of the image plane which, when projected onto the ground will coincide with a target distance d_0 away from the vehicle. Note that a potential camera offset can be added to this computation.

$$y_o = \theta - tan^{-1}\left(\frac{h}{d_o}\right) \tag{1}$$

From this we can compute our lateral offset x_l from the line at some point ahead of the vehicle which we can track.

$$x_l = ay_o^2 + by_o + c \tag{2}$$

This angular attitude error can be projected onto the ground plane to determine the metric lateral offset from the line. This lateral error can then be minimized with a simple PID controller coupled to the vehicle roll angle ϕ_{sp}.

Additionally, we extract the gradient of the line fit (Θ) at this target offset to determine our alignment error.

$$\Theta = \begin{cases} 2ay_o + b, & \text{if } 2ay_o + b \leq 2 \\ 2, & \text{otherwise} \end{cases} \tag{3}$$

With this we can then set our desired heading as

$$\psi_{sp} = \psi + \tilde{\Theta} \tag{4}$$

4 FLIGHT THROUGH AND DETECTION OF CIRCULAR GATES

Algorithms described in this section allow the MAV to perform precise flight through a set of increasingly small hoops. A test setup was created in TU Delft's Cyberzoo, depicted in Figure 9.

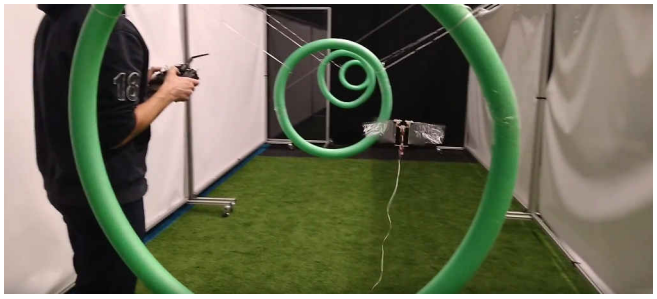


Figure 9: Flight through hoops - test setup.

4.1 Perception

To detect the gates, we find circles in the image using a probabilistic Hough transform based on the bisector between pixel pairs. The Hough transform provides robustness against segmentation errors (for instance by uneven lighting) and against errors in the shape of the gates, while the probabilistic sampling keeps the method computationally lightweight.

Pixels are randomly sampled from the image; YUV color thresholding is then used to test whether these pixels belong

to the gate or the background. Only pixels belonging to the gate are considered for further processing. The thresholds were tuned conservatively, as only a small number of inliers is required to locate the gate in the image while false positives are likely to degrade the result.

We use the bisector between pixel pairs to find the midpoint of the gate. For all pairs of pixels lying on a circular gate, their bisector should intersect the gate's midpoint. This is used as follows: each time a new gate pixel is found, it is paired with all previously found pixels. For each new pair of pixels, the bisector is constructed and all accumulator bins along this line are incremented (Figure 10). This procedure is repeated until a fixed number of pixels is sampled; we sample 20 pixels, leading to a total of 190 bisectors. Once enough pixel pairs have been evaluated, the bin with the highest inlier count is selected as the gate's midpoint (x_g, y_g).

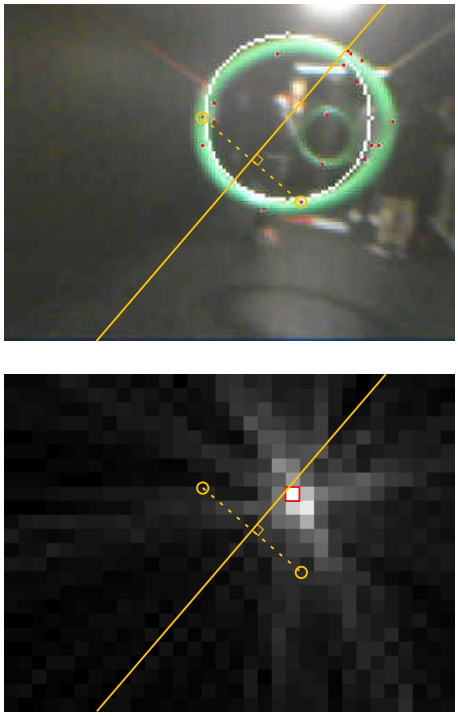


Figure 10: Circular gate detection using the probabilistic Hough transform. *Top:* input image with the sampled pixels highlighted in red and the detected gate shown in white. An example bisector between two sampled pixels is shown in yellow. *Bottom:* the accumulator belonging to this image. Brighter pixels indicate a higher likelihood of the gate's midpoint lying at that position; the bin highlighted in red has the highest inlier count and is selected as the gate's midpoint. The example bisector is also overlaid on the accumulator and is shown to intersect the gate's midpoint.

Estimation of the gate's radius was deliberately left out of the Hough transform to reduce the size of the accumulator. The accumulator was also made four times smaller than the input image to further reduce memory consumption and processing time. Instead of estimating the radius during the Hough transform, we perform this in a later stage where the median distance between the estimated midpoint and a small number (11) of inlier pixels is used to measure the gate's apparent radius or aperture β_g.

4.2 Control

We run a very simple iterative algorithm to localize our position along the tunnel (p) using the predefined location ($\mathbf{D_g}$) and width ($\mathbf{w_g}$) of the gates. When we start the flight attempt, we reinitialize the localization algorithm at an assumed start distance from the first gate. We use two estimates to update our estimated position, the first based on odometry and the second on the perceived location of the gates.

The odometry estimate is obtained using the estimated vehicle air speed generated by a linear transform (c) from the vehicle pitch angle $v_{est} = c\theta$. c was identified as -0.049 by performing a simple line fit through the measured pitch and speed using a motion tracking system as ground truth. Due to the relatively large profile drag of the DelFly at small pitch angles, this linear transform is generally quite accurate. If we assume no external drafts, we can equate the vehicle air speed and its ground speed.

The identified angular aperture of the gate obtained from the perception algorithm described earlier, is used to generate a relative position estimate to the gates. There is, however, an ambiguity as to which gate was identified. To address this, we compute the estimated metric distance to each gate given the size of each gate and the identified angular size. Given our current estimate of the position computed using our odometry estimate ($\tilde{p}(t) = p(t-1) + c\delta t$) and the gate positions from our map, we extract the gate index which result in the smallest estimation error.

$$i = argmin(\mathbf{D} - \mathbf{w_g} tan\frac{\beta_g}{2} - \tilde{p}) \tag{5}$$

Now that we know which gate we are looking at, we use the estimation error to update our position estimate with a filter using a discount factor $\alpha = 0.25$.

$$p(t) = \tilde{p}(t) + (\mathbf{D}(i) - \mathbf{w_g}(i) tan\frac{\beta_g}{2} - \tilde{p}(t)) * \alpha \tag{6}$$

We then set our desired lateral and vertical position for the control system which drive the vehicle roll and thrust respectively. The body pitch is kept constant to have constant forward airspeed.

$$Lat_{sp} = \mathbf{w_g}(i) tan\frac{\beta_g}{2} x_g \tag{7}$$

The height set-point is computed as:

$$h_{sp} = h + \mathbf{w_g}(i) tan\frac{\beta_g}{2} y_g \tag{8}$$

5 CONCLUSION

In this paper we have presented augmented versions of traditional computer vision and control algorithms. With these, we participated in the 2018 International Micro Air Vehicle (IMAV) competition with the DelFly Nimble. First of all, robust flight of the platform was assured, by implementing attitude stabilization and battery level dependent height control. Then, we demonstrated how a second order line fit was performed using a sub-sampling method, which was then used to control roll and yaw. Finally, flight through and detection of circular gates in a tunnel was discussed. A probabilistic Hough transform in conjunction with a position estimate was used to control the vehicle.

With these algorithms we could now, for the first time, run several non-trivial perception tasks on the DelFly Nimble. In a further iteration of the algorithms, increased robustness should be the main focus, to allow for application in more challenging environments. For example, the control algorithms can be altered to work better in outdoor environments and a higher dynamic range in the camera would be desirable.

Even though novel algorithms could further improve mission capabilities, we believe that more computing power is necessary for increasingly autonomous flight with such limited power and weight budgets. Custom SoC (System on Chip) design with accelerators specific to the application have the potential to enhance the autonomous capabilities of flapping wing MAVs.

REFERENCES

[1] Syaril Azrad, Farid Kendoul, and Kenzo Nonami. Visual servoing of quadrotor micro-air vehicle using color-based tracking algorithm. *Journal of System Design and Dynamics*, 4(2):255–268, 2010.

[2] Daniel Eberli, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. Vision based position control for mavs using one single circular landmark. *Journal of Intelligent & Robotic Systems*, 61(1-4):495–512, 2011.

[3] Abel Gawel, Mina Kamel, Tonci Novkovic, Jakob Widauer, Dominik Schindler, Benjamin Pfyffer von Altishofen, Roland Siegwart, and Juan Nieto. Aerial picking and delivery of magnetic objects with mavs. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5746–5752. IEEE, 2017.

[4] Justin Thomas, Giuseppe Loianno, Kostas Daniilidis, and Vijay Kumar. Visual servoing of quadrotors for perching by hanging from cylindrical objects. *IEEE robotics and automation letters*, 1(1):57–64, 2015.

[5] Cooper Bills, Joyce Chen, and Ashutosh Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *2011 IEEE International Conference on Robotics and Automation*, pages 5776–5783. IEEE, 2011.

[6] Hanoch Efraim, Amir Shapiro, Moshe Zohar, and Gera Weiss. Position-based visual servoing of a micro-aerial vehicle operating indoor. *Journal of Dynamic Systems, Measurement, and Control*, 141(1):011003, 2019.

[7] Oualid Araar and Nabil Aouf. Visual servoing of a quadrotor uav for autonomous power lines inspection. In *22nd Mediterranean Conference on Control and Automation*, pages 1418–1424. IEEE, 2014.

[8] 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.

[9] GCHE De Croon, M Perçin, BDW Remes, R Ruijsink, and C De Wagter. *The DelFly*. Springer, 2016.

[10] GCHE De Croon, KME De Clercq, Remes Ruijsink, Bart Remes, and Christophe De Wagter. Design, aerodynamics, and vision-based control of the delfly. *International Journal of Micro Air Vehicles*, 1(2):71–97, 2009.

[11] Christophe De Wagter, Sjoerd Tijmons, Bart DW Remes, and Guido CHE de Croon. Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4982–4987. IEEE, 2014.

[12] Sjoerd Tijmons, Guido CHE de Croon, Bart DW Remes, Christophe De Wagter, and Max Mulder. Obstacle avoidance strategy using onboard stereo vision on a flapping wing mav. *IEEE Transactions on Robotics*, 33(4):858–874, 2017.

[13] Kirk YW Scheper, Matěj Karásek, Christophe De Wagter, Bart DW Remes, and Guido CHE De Croon. First autonomous multi-room exploration with an insect-inspired flapping wing vehicle. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.

[14] Karl Martin Kajak, Matěj Karásek, Qi Ping Chu, and GCHE de Croon. A minimal longitudinal dynamic model of a tailless flapping wing robot for control design. *Bioinspiration & biomimetics*, 2019.

[15] Guido C. H. E. de Croon, C. De Wagter, Bart D. W. Remes, and Rick Ruijsink. Sub-sampling: Real-time vision for micro air vehicles. *Robotics and Autonomous Systems*, 60:167–181, 2012.

Using Prior Information to Improve Crop/Weed Classification by MAV Swarms

Federico Magistri*, Daniele Nardi† and Vito Trianni ‡
ISTC-CNR, Via San Martino della Battaglia 44, 00185 Rome, Italy
DIAG, Sapienza University of Rome, Via Ariosto 25, 00185 Rome, Italy

ABSTRACT

Precision agriculture can benefit from the usage of swarms of drones to monitor a field. Crop/weed classification is a concrete application that can be efficiently carried out through collaborative approaches, whereby the information gathered by a drone can be exploited as prior to improve the classification performed by other drones observing the same area. In this study, we instantiate this concept by exploiting state-of-the-art deep learning techniques. We propose the usage of a shallow convolutional neural network that receives as input, besides the RGB channels of the acquired image, also an additional channel that represents a probability map about the presence of weeds in the observed area. Exploiting a realistic, synthetic dataset, the performance is assessed showing a substancial improvement in the classification accuracy.

1 INTRODUCTION

Use of aerial robots has been steadily increasing over the past decade, thanks to improved remote sensing abilities, better motion control and even onboard manipulation abilities. Such systems constitute a natural fit for tasks related to monitoring and inspection. In particular, small micro aerial vehicles (MAVs) are very well suited to such operations, even indoor, as they can navigate in narrow spaces, get close to the target objects and safely operate around humans. MAVs constitute an extremely attractive option for a number of practical use-cases—e.g., within application areas such as agrifood or infrastructure inspection and maintenance—opening up a wide range of market opportunities. However, for MAVs to realize their potential and get deployed in unstructured environments (outside the lab, without support from any external infrastructure to operate), a number of technical and scientific challenges related to navigation, perception and cognition must be solved. In addition, while MAVs small size is key to operational settings, it also gives rise to a number of limitations, for instance in terms of useful payload and power autonomy. Payload and power limitations do not support the installation onboard of powerful computing devices, high-resolution cameras and heavy optics, and in any case the battery lifetime may be severely limited. Such limitations make it difficult to address inspection and monitoring tasks over extensive areas, a necessary requisite for applications in precision agriculture—both outdoor and within greenhouses—or in large industrial settings.

The above limitations can be gainfully addressed by means of multi-robot systems, and notably MAV swarms, that can improve efficiency through parallel operation over large areas [1]. By exploiting a swarm of small drones, it is possible to acquire data at higher resolution, exploiting their ability to hover close to a given target and to navigate narrow cluttered environments. Additionally, the ability for members of the swarm to actively support each others enables collaborative localisation and collision avoidance [2]. Finally, MAVs in a swarm can collaborate to improve the quality of exteroception and sensory data interpretation, as a result of the collective intelligence of the group.

In our work, we propose the exploitation of drone swarms for precision agriculture applications [3]. Specifically, we consider the problem of identification and mapping of weeds within a crop field. This is a very relevant application in the precision agriculture domain, because the detailed knowledge of the position and type of weeds within a field can support advanced weed control techniques, from variable-rate herbicide application—a practice that can reduce herbicide usage by more than 80%—to mechanical removal of weeds, possibly automatically performed by ground robots. Assuming that the weed distribution within a field is non-homogeneous, inspection of extensive fields by drone swarms can be efficiently performed by means of non-uniform coverage strategies, which deploy resources (i.e., drones) only towards portions of the field with high relevance, while areas of low interest receive much less attention [4]. To this end, an estimation of the utility of each area must be performed first, and on such basis a more or less detailed inspection can be executed. Utility estimation is performed by a high-altitude/low-resolution inspection, while detailed inspection is performed through low-altitude/high-resolution inspection, and the latter can be exploited to continuously update the former. Hence, for non-uniform coverage strategies to be implemented by an autonomous decentralised system, it is necessary that MAVs are capable to communicate and adapt their mission on the

*federico.magistri@gmail.com
†nardi@diag.uniroma1.it
‡vito.trianni@istc.cnr.it

basis of what observed on the field, hence requiring suitable algorithms for online/onboard feature detection.

While high-altitude/low-resolution inspection can be performed by individual drones with standard estimation techniques based on common indexes used in precision agriculture, low-altitude/high-resolution inspection requires the identification and classification of individual plants, so as to determine their type and position within the field. In this study, we address the latter aspect, proposing a framework for collaborative classification of relevant environmental features based on state-of-the-art deep neural networks. We assume here that detailed inspection is performed by a MAV swarm by flying at a relatively low altitude (e.g., 3m from the ground) so that images of the field are taken at a sufficient resolution even with low-end and lightweight cameras. Classification of crop and weed can be carried out with state-of-the-art techniques making use of convolutional neural networks (CNNs) for object detection, which return the position and class type for all relevant objects identified within an image [5]. However, CNNs are computation-hungry methods that are not suitable for the limited devices available onboard MAVs. Therefore, for MAV swarms to be efficient and accurate, it is necessary to reduce the computational complexity of the algorithms running onboard the single MAV, while exploiting collaboration among MAVs that can support each other on the classification task.

We propose to exploit the fact that different MAVs can inspect the same area of the field at different times and from different perspectives, therefore having redundant information about the same plants that can be exploited to improve the classification accuracy. Each MAV is endowed a streamlined version of a deep CNN. On the first passage over an area of the field, a MAV independently makes a classification of the different plants it can perceive. Such classification is geolocalised exploiting onboard devices (e.g., RTK-GNSS) or self-localisation techniques, and then broadcasted to all other MAVs in the swarm, possibly using a simple re-broadcasting protocol to widely diffuse newly available information. Successive passages exploit prior knowledge by building probability maps about the existence of crops and weeds on the current portion of the field. Such probability maps are fed as additional input channels to the CNN (similarly to what proposed in [6] for foreground/background segmentation), so as to improve the classification accuracy on all the relevant elements in the inspected area. To validate this proposal, we developed a realistic 3D simulation of a sugar-beet field in which two types of weed are present. This allowed us to generate a synthetic dataset of field images as gathered from a MAV flying at a low altitude, simulating multiple independent passages over the same area by changing position and illumination parameters. We reduce the depth and complexity of a state-of-the-art CNN and increase the input channels to include also the possible availability of probability maps. We test several training approaches by varying the likelihood of providing the additional probability maps with respect to simple RGB channels. We show that across multiple passages, the performance of the classification substantially improves, validating the proposed concept and calling for further refinements as well as for tests with real-world datasets.

The paper is organised as follows. In Section 2, we briefly review the available techniques for classification in a weed management domain. In Section 3, we describe the experimental setup detailing the synthetic dataset generation, the proposed CNN architecture and the training methods devised. In Section 4, we discuss the testing procedure and the results obtained, comparing our iterative method with one-shot approaches. Section 5 concludes the paper.

2 CROP/WEED METHODS

In recent years, the interest in robotics applications for precision agriculture raised constantly [7]. Among the most important problems tackled through automatic techniques, weed control represents an important case study as it requires both advanced vision to recognise weed type and fine mechanical control to spray or remove the identified plants. As a consequence, several approaches to the crop/weed classification problem have been attempted using both unmanned ground (UGVs) and aerial vehicles (UAV). On the one hand, UGVs are generally large powerful tractors adapted from traditional agricultural machinery, and can be equipped with performing hardware, thus allowing on-board classifications even with modern deep neural networks. However, large UGVs must carefully manoeuvre to avoid damage to the crop field and to reduce soil compaction. Furthermore, the close-up view of a camera mounted on a UGV does not allow to exploit the geometric pattern of a typical field. On the other hand, UAVs have the possibility to quickly cover large crop fields and to perceive a wide area at once. However, due to payload limitations, they cannot exploit hardware with the same computing capability as for the UGV case.

In recent years, efforts have been made to provide reliable crop/weed classification methods. Object-based classification methods exploit the sowing pattern to classify as weed plants that lay outside the crop rows [8]. In [9], a random forest classifier was adapted to UAV imagery, using, as input, a large set of hand computed features including also the main row direction of the crop field. The same authors [10] exploited also a very shallow neural network to classify plant species; before feeding the neural network, a vegetation mask based on the popular NDVI index is computed, and single plants are extracted from this mask and passed to the CNN. In [11], feature learning is exploited for weed classification from UAV images. In [12], a deep auto-encoder architecture composed by 26 convolutional layers (the encoder) and 5 up-sampling layers (the decoder) obtained a pixel-wise semantic segmentation, using as input RGB images with NIR informations. The same approaches have been deployed to UGV based systems [13, 14, 15, 16]. In all these examples, the classification

is performed offline after the collecting stage is finished.

3 EXPERIMENTAL SETUP

As described in Section 2, most current approaches tackle the crop/weed classification task by means of semantic segmentation solutions, while our goal is to use state-of-the-art object detection algorithms so that each plant can be individually classified. This will make it possible, once each plant is detected, to take action within the field on a per-plant basis, e.g., with mechanical removal or spraying of individual plants. For this purpose, there is no large publicly available dataset that can be exploited. Additionally, the proposed swarm-based technique requires multiple images of the same portion of the field taken at different times and possibly from slightly different positions. While work is being performed to collect a suitable dataset with the required features, the validation of the concept can be more flexibly performed on synthetic datasets that can be generated through modern computer graphics engines [17]. We describe the dataset generation in Section 3.1. Thanks to such a dataset, we are able to train CNNs for object detection. The chosen CNN architecture and the technical choices to provide prior knowledge from previous passages as input to the CNN are detailed in Section 3.2. Finally, the training methods used to obtain an efficient object detection are discussed in Section 3.3.

3.1 Synthetic dataset generation

The proposed method for crop/weed classification relies on multiple passages over the same area of the field, hence on multiple images with different illumination and possibly different perspective. Given the complexity of acquiring a similar dataset in the field, a synthetic dataset has been generated using the advanced computer graphics features provided by the game engine Unity 3D (https://unity.com). As a bonus, the ground truth labelling is obtained with precision and low effort directly from the simulator, hence removing one of the main difficulties in machine vision research.

Starting from the 2D texture of leaves belonging to the target plant species, it is possible to generate a large variety of individual 3D plants by assembling multiple leaves and realistically bending the texture [17]. In the simulation environment, each plant is generated with several parameters which are individually tuned for each species to resemble as much as possible the aspect of the real counter-part (see Figure 1). To each plant, independently from its species, a vertical growth axis is associated which is slightly perturbed by a random noise. To simulate a uniform growth stage for all the plants that have been generated, each plant has a number of layers up to two. Each plant, moreover, has an associated number of leaves per layer which is different from species to species. At each layer, the leaves are homogeneously spread around the main growth axis, again with a small random disturbance. In order to simulate succesive visits of the same region of the field, once a set of plants has been placed on the scene, the illumination parameters have been randomly changed, moving the position and the intensity of the light sources illuminating the scene, hence also casting different shadows on the ground. As a last step to create a realistic environment, the soil is generated starting from various real world textures. Every time a fragment of terrain is created, two textures are chosen and blended together using Perlin noise based linear interpolation (LERP). Similarly to the plants generation, by changing the parameters of the Perlin noise, it is possible to create a large variety of soil textures.

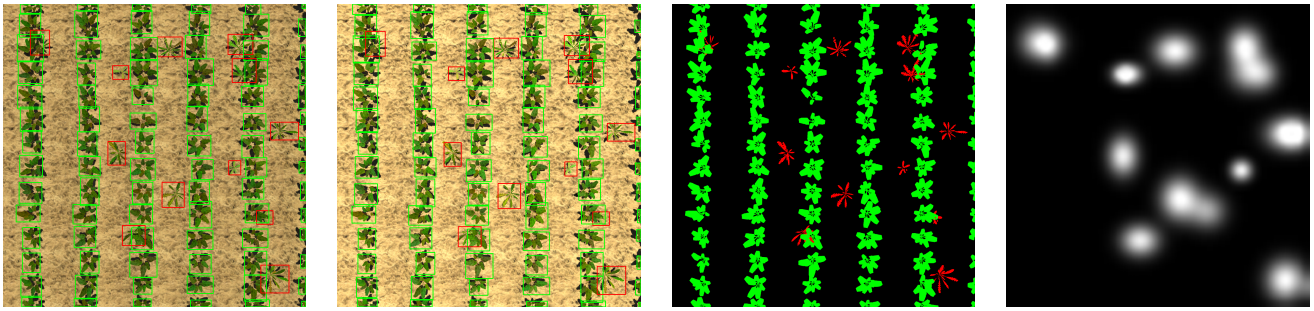
With this method, images of different plants taken at different altitudes can be generated at will. Here, we consider images taken at about 3m altitude, containing about 80 simulated sugar beets as crop, while two types of weed are present in variable number, having up to 20 plants per image (see Figure 1). A training set with 500 images has been generated, while the validation set is composed of 100 images. Regarding the testing set, 10 blocks of 40 images have been generated. Each block is composed of the same 40 plant patterns, but the illumination parameters, such as light source orientation and intensity, randomly change for each image within a block. This results in a testing dataset with overall 400 images that we refer to as dataset A. Additionally, a second testing set has been generated featuring also a small, random perturbation of the camera position within the same image in different blocks. In this way, the position error of the MAVs flying over the same region is simulated. This second dataset is also made of 40 fields and 10 blocks for a total of other 400 images, and is referred to as dataset B.

3.2 CNN architectures

In the literature, several architectures have been proposed for classification purposes, either for image segmentation or for object detection [5, 18]. Competitions and benchmarks have contributed to establish an objective methodology to determine performance and direct choice of the best approach, given the task demands. In our case, to perform the plant detection, Faster R-CNN has been chosen [19], which can

Table 1: CNN Architectures

5 layers	9 layers
7×7, 64, stride 2	
3×3, max pools, stride 2	
$\begin{bmatrix} 3\times 3, & 64 \end{bmatrix} \times 1$	$\begin{bmatrix} 3\times 3, & 64 \end{bmatrix} \times 2$
$\begin{bmatrix} 3\times 3, & 128 \end{bmatrix} \times 1$	$\begin{bmatrix} 3\times 3, & 128 \end{bmatrix} \times 2$
$\begin{bmatrix} 3\times 3, & 256 \end{bmatrix} \times 1$	$\begin{bmatrix} 3\times 3, & 256 \end{bmatrix} \times 2$
$\begin{bmatrix} 3\times 3, & 512 \end{bmatrix} \times 1$	$\begin{bmatrix} 3\times 3, & 512 \end{bmatrix} \times 2$



(a) (b) (c) (d)

Figure 1: Synthetic dataset generation. The same field with crop rows and weeds is displayed. Panels (a) and (b) show different illumination conditions on the same field, and also display the automatically generated ground truth. A small difference in the plant position within the image is also present. Panel (c) shows a segmentation on the image obtainable as ground truth directly from the simulation. Finally, panel (d) shows a probability map computed on the ground truth data.

be considered the state-of-the-art method for object detection tasks. In its standard version, it is composed of two stages, in which the first stage—referred to as the backbone—is a deep CNN responsible for generating bounding boxes around objects to be proposed to the second stage as potentially containing relevant features. Considering the computing capabilities of a MAV, it is not possible to use the standard Faster R-CNN backbone such as ResNet50 [20], which is way too demanding in terms of computational power. Therefore, we have chosen to implement two shallow networks with much reduced demands, removing several layers from the standard backbone. In both cases, the first initial layers are the same as the ResNet architectures, namely a 7×7 convolutional layer with 64 filters and stride 2, followed by a 3×3 max-pooling layer. After that, a sequence of 3×3 convolutional layers is presented as described in Table 1. We will refer to the first as FCN5 and to the latter as FCN9. Here, FCN stands for Fully Convolutional Network.

In order to exploit detections previously made by other agents, the input of the CNN is composed of a fresh RGB image together with an auxiliary channel encoding a probability map based on previous classifications. Well-known object detection algorithms usually output 6 values for each detection i, that is, the class of the detected object c_i, a confidence score s_i, and 4 values representing bounding box coordinates encoding the coordinates x_i, y_i of the center and the width w_i and height h_i of the bounding box. From this values a probability $P(x, y)$ for each point x, y is computed as follows:

$$P(x,y) = \sum_{c_i=W} s_i \cdot e^{-\left(\frac{(x-x_i)^2}{2w_i^2}+\frac{(y-y_i)^2}{2h_i^2}\right)}. \qquad (1)$$

In other words, each detection belonging to the class $c_i = W$—standing for weed—provides a probability increment proportional to the confidence score s_i, and decaying from the center of the bounding box x_i, y_i as a 2D gaussian with a spread that depends on the bounding box dimensions w_i and h_i. The resulting probability map is practically null when far from any bounding box, indicating that the probability of finding a weed plant in that position is extremely low. Peaks are visible in correspondence of detected weeds, as shown in Figure 1d. Note that we decided to focus on weed classification only, as it turns out that the performance on crop classification is already very high (see Section 4.1), hence requiring a specific method only for improving the weed detection.

3.3 Training methods

The CNN that we have devised must be capable of performing two tasks at the same time. On the one hand, it must observe the RGB channels alone to identify the presence of crops or weeds. This will output a list of detections that can be used to compute a probability map for subsequent passages by other MAVs. On the other hand, the CNN must prove capable of using—when available—the prior information to improve the classification and reduce errors. Possibly, the NN must also identify and remove conflicts between the newly available RGB image and the prior information encoded in the probability map. This turns out to be an important choice to achieve better results, since the network has to learn to balance the information coming from other agents and the fresh image. Therefore the network will not only rely on information coming from the auxiliary channels but it will be able to make a valuable initial classification and to correct possible misclassifications. A correct training of the network is therefore key to obtain both these abilities within a single CNN.

First and foremost, we have devised three different training strategies in terms of the frequency with which the probability map is presented. We used 25%, 50% and 75% of the training cases, hence pushing more or less towards the usage of the information encoded into the probability map. Additionally, to compute the probability map, instead of using the available ground truth we decided to use realistic labelling as produced from a CNN classification. To this end, we trained a FCN5 architecture with the only RGB channels

Table 2: Crop/weed classification performance with FCN5 and FCN9, with only the RGB input channels.

Dataset A

	FCN5		FCN9	
	Crop	Weed	Crop	Weed
Precision	0.96	0.98	0.96	0.98
Recall	0.87	0.65	0.91	0.73
F1	0.91	0.78	0.93	0.83

Dataset B

	FCN5		FCN9	
	Crop	Weed	Crop	Weed
Precision	0.99	0.98	0.99	0.98
Recall	0.94	0.64	0.96	0.73
F1	0.96	0.78	0.97	0.83

on the available dataset, and we used the detections obtained by the FCN5 to generate the probability map, without filtering out bounding boxes with low confidence. As a consequences the neural network will learn to deal with errors in the auxiliary input as generated by a similar CNN architecture.

4 RESULTS

The trainings and testings are performed with a NVIDIA Quadro P6000, a 24 GB GPU with 3840 CUDA cores. Each training was performed with 50000 iterations, a learning rate of 0.01, weight decay of 0.0001 and batch size 4. Testing of the trained networks has been perfomed on the two testing datasets, with and without position error. In order to evaluate our approach, precision, recall and F1-score have been computed. As in many detection tasks, a detection is considered a true positive if the Intersection Over Union (IoU) between the detected box and the ground truth is above a certain threshold (here: 0.5). Otherwise, it is considered a false positive.

4.1 Crop/weed classification with simple RGB images

First of all, we discuss the classification performance on the synthetic dataset when no a priori information is provided, hence no additional input channel is used besides the RGB channels of the input image. The FCN5 and FCN9 networks have been trained and tested on both datasets A and B. In this case, each testing set is composed of 400 images. The performance for the precision, recall and F1 metrics is shown in Table 2. Note that, not using any a priori knowledge, every image is processed independently and the differences observable between dataset A and dataset B are only due to the 40 different synthetic fields generated for each.

Specifically, the *crop* class achieves high scores even with the shallower FCN5 network, and dataset B appears easier to classify, possibly due to the relative positioning of crop and weeds, or border effects (e.g., a crop line partially included into an image because appearing on the border). The performance on the weed class is instead lower, especially for the recall, meaning that several weed plants go undetected. The FCN9 achieves better results here, meaning that there is room for improvement over the FCN5 results by including prior information with additional channels. Considering that the testing datasets are organised in blocks representing the same field but varying the illumination conditions, it is interesting to analyse how performance varies across different blocks, so as to determine how much the illumination matters on the final results. Figure 2 shows that there is indeed some non-negligible variability in performance among the different blocks, hence further motivating the use of prior information for more stable and reliable classification. Given that the performance on the *crop* class is already very high with the FCN5, we decided to use only one auxiliary channel representing a probability map for the *weed* class obtained from previous classifications.

4.2 Crop/weed classification with probability maps

To evaluate the performance achievable over multiple passages on the same field, we perform 10 classifications in a sequence using the output of the current stage to compute the probability map of the following stage (see Figure 3). As it is possible to note, while the first passage has an empty probability map, successive passages can exploit the prior knowledge to improve the classification of weeds. As a matter of fact, it can be noted that in the successive passages, more plants are correctly detected.

A proper performance evaluation is carried out on dataset A, where no position error is included (corresponding to the same condition experienced during training). Considering that each of the 10 blocks in dataset A have independent illumination conditions, we compute 100 different sequences by random permutation of the 10 blocks, and use them to have

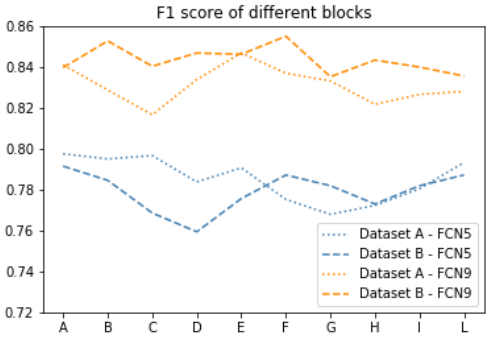


Figure 2: F1 performance of FCN5 and FCN9 across different blocks of images, which differ only in the scene illumination.

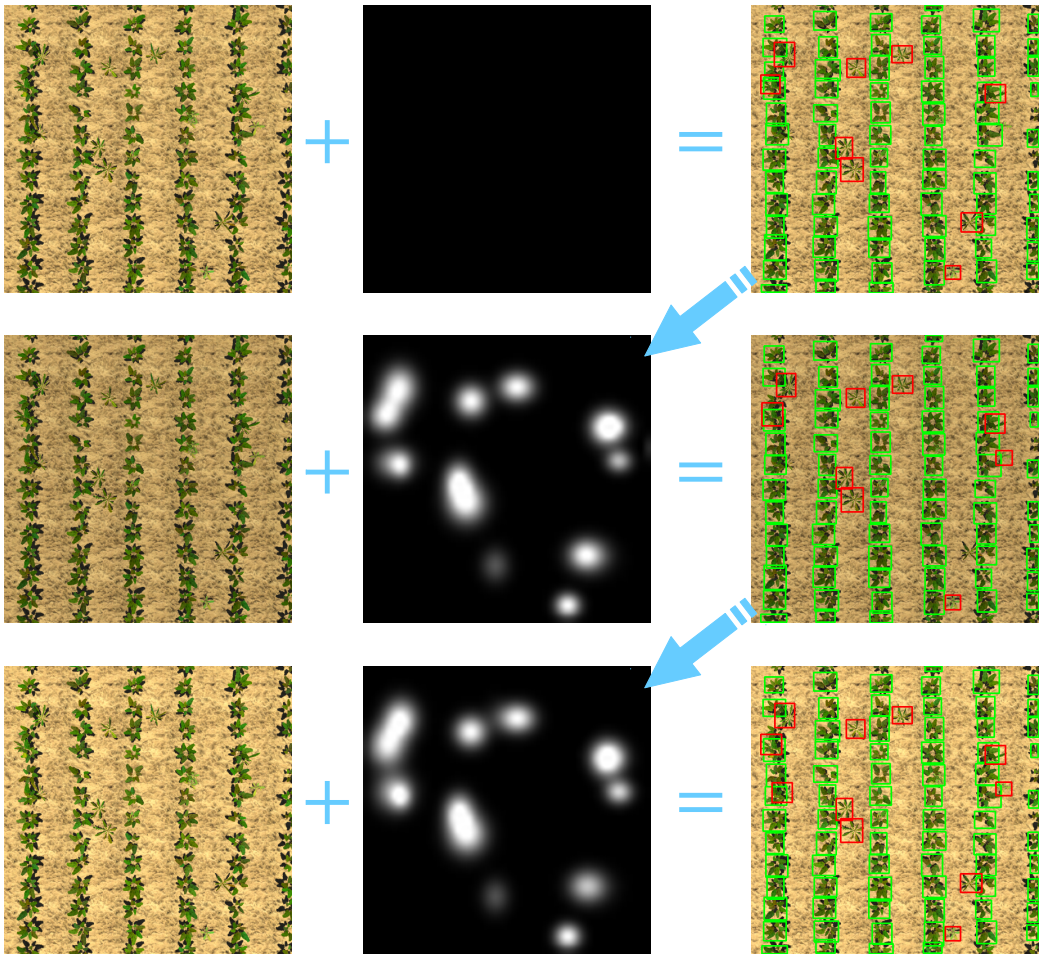


Figure 3: Classification over multiple passages. Each row correspond to a single passage over the same portion of the field, but with different illumination conditions (first column). The RGB image is coupled with a probability map derived from previous passages, when available (second column). The combined input provides an improved object detection across passages (third column). In the second passage, a weed is discovered in the center-right part of the image, but one in the bottom right is lost. In the third passage, the latter weed is detected again, and an additional weed is discovered among the first crop row on the left.

an average performance that is independent as much as possible from the specific sequence observed. The results for precision, recall and F1 on the *weed* class are presented in Figure 4. It can be noted that the overall classification accuracy increases when exploiting the probability maps coming from previous passages. More specifically, the recall is the measure most affected by the auxiliary input, while the precision can undergo a slight degrade, which is observed especially for networks trained with 75% probability of having a probability map in input. The training strategy is indeed very important to obtain a substancial improvement in the classification through multiple observations. When only 25% or 50% of the training examples are provided with a probability map, the improvement in the weed classification is only mild. Instead, with a 75% probability, the neural network learns to properly exploit the additional input when available, reaching comparable levels of performance as the more complex FCN9 network. The proposed approach is intrinsically robust against position errors, as shown by the testing performed on dataset B (see Figure 5). Even though position errors where never presented during the training phase, it is possible to note that a performance improvement is still visible through successive observations of the same region of the field. This improvement is not as considerable as for Dataset A but it is still possible with FCN5 and the auxiliary probability map to approach the performances of FCN9.

5 Conclusions

We have proposed an approach to exploit knowledge available on portions of the field coming from previous observations to iteratively improve the performance of classification by a shallow neural network, to be executed onboard

lightweight MAVs with limited payload and constraints in the computational power. We obtained a substancial improvement in performance, that makes a shallow architecture achieve similar performance of a double-size network.

These results validate the concept proposed here for the first time, and open the way for a thorough analysis of the design space to identify possible improvements that can further boost performance. Future work will be dedicated to this as well as to test the methodology on real-world images. To this end, a dataset with multiple passages on the same area has already been collected, and studies are on the way to provide new grounds for the analysis of the proposed framework.

ACKNOWLEDGEMENTS

This work has been conducted within the project SAGA (Swarm Robotics for Agricultural Applications, see http://laral.istc.cnr.it/saga), an experiment founded within the ECHORD++ EU project (GA: 601116). Vito Trianni acknolwdges the NVIDIA GPU Grant Program providing the Quadro P6000 GPU used in this work.

REFERENCES

[1] K McGuire, M Coppola, C De Wagter, and G de Croon. Towards autonomous navigation of multiple pocket-drones in real-world environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems IS -*, pages 244–249, 2017.

[2] Mario Coppola, Kimberly N McGuire, Kirk Y W Scheper, and Guido C H E de Croon. On-board communication-based relative localization for collision avoidance in Micro Air Vehicle teams. *Autonomous Robots*, 42(8):1787–1805, June 2018.

[3] Dario Albani, Daniele Nardi, and Vito Trianni. Field coverage and weed mapping by UAV swarms. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS*, pages 4319–4325. IEEE, 2017.

[4] Dario Albani, Tiziano Manoni, Daniele Nardi, and Vito Trianni. Dynamic UAV Swarm Deployment for Non-Uniform Coverage. In *AAMAS '19: Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, pages 523–531, Stockholm, Sweden, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[5] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

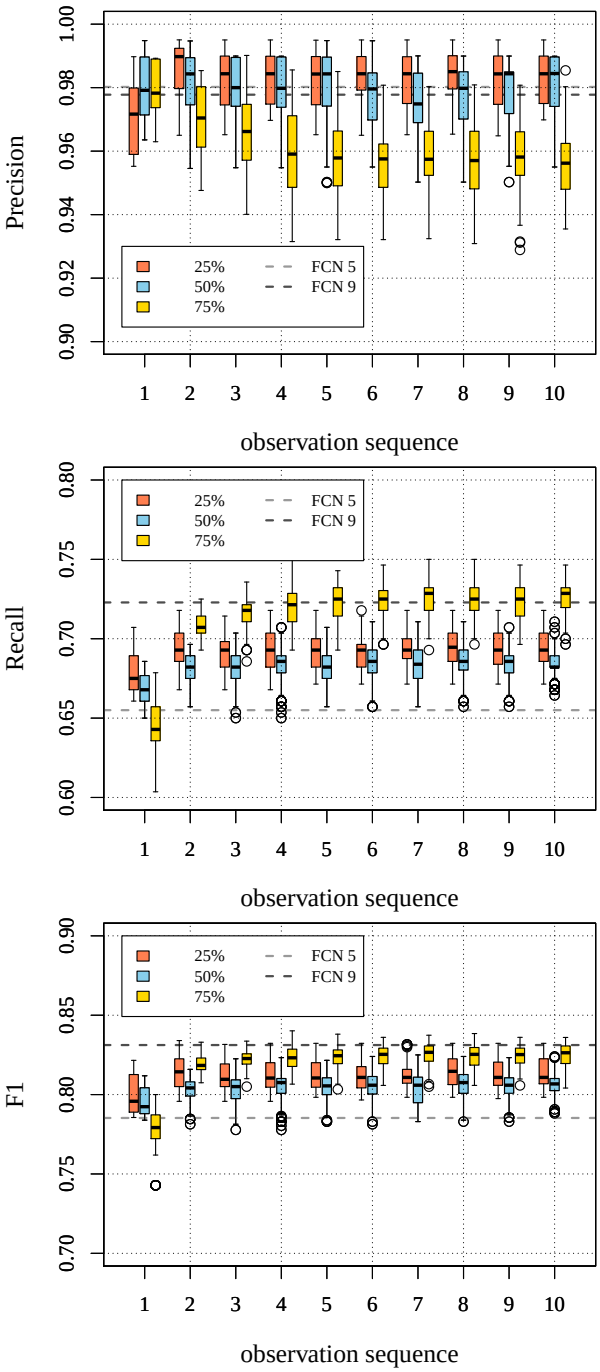


Figure 4: Precision, recall and F1-score on dataset A. The performance achieved with different training strategies (i.e., 25%, 50%, and 75% probability of having a probability map as input) is compared against the average performance of the FCN5 and FCN9 networks using only the RGB inputs.

[6] Kunming Luo, Fanman Meng, Qingbo Wu, Wen Shi, and Lili Guo. A cnn-based segmentation model for segmenting foreground by a probability map. In *2017 In-*

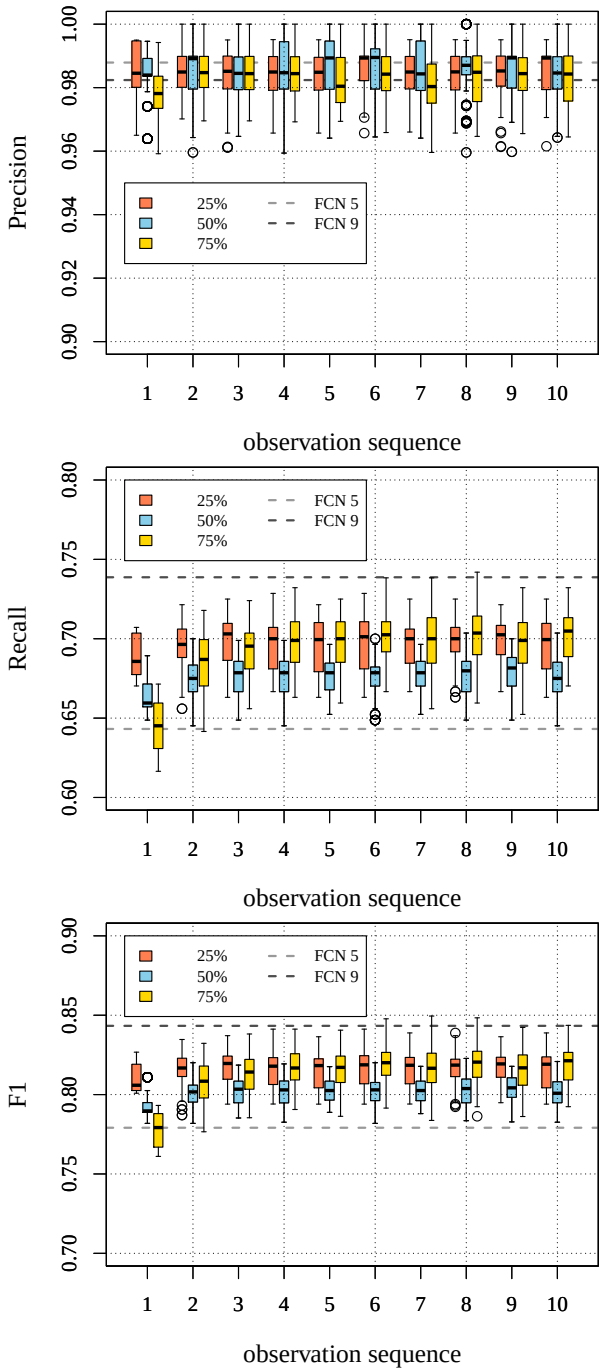


Figure 5: Precision, recall and F1-score on dataset B, where also a positioning error is included.

ternational Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), pages 17–22. IEEE, 2017.

[7] Anthony King. Technology: The Future of Agriculture. *Nature*, 544(7651):S21 EP —-S23, April 2017.

[8] José Manuel Peña, Jorge Torres-Sánchez, Ana Isabel de Castro, Maggi Kelly, and Francisca López-Granados. Weed Mapping in Early-Season Maize Fields Using Object-Based Analysis of Unmanned Aerial Vehicle (UAV) Images. *PLoS ONE*, 8(10):e77151 EP —-11, October 2013.

[9] Philipp Lottes, Raghav Khanna, Johannes Pfeifer, Roland Siegwart, and Cyrill Stachniss. UAV-based crop and weed classification for smart farming. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3024–3031. IEEE, 2017.

[10] Andres Milioto, Philipp Lottes, and Cyrill Stachniss. Real-time blob-wise sugar beets vs weeds classification for monitoring fields using convolutional neural networks. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4:41, 2017.

[11] Calvin Hung, Zhe Xu, and Salah Sukkarieh. Feature Learning Based Approach for Weed Classification Using High Resolution Aerial Images from a Digital Camera Mounted on a UAV. *Remote Sensing*, 6(12):12037–12054, December 2014.

[12] Inkyu Sa, Zetao Chen, Marija Popović, Raghav Khanna, Frank Liebisch, Juan Nieto, and Roland Siegwart. weednet: Dense semantic weed classification using multispectral images and mav for smart farming. *IEEE Robotics and Automation Letters*, 3(1):588–595, 2017.

[13] A T Nieuwenhuizen, L Tang, J W Hofstee, J Müller, and E J van Henten. Colour based detection of volunteer potatoes as weeds in sugar beet fields using machine vision. *Precision Agriculture*, 8(6):267–278, November 2007.

[14] S Haug, A Michaels, P Biber, and J Ostermann. Plant classification system for crop /weed discrimination without segmentation. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 1142–1149. IEEE, 2014.

[15] P Lottes, M Hoeferlin, S Sander, M Müter, P Schulze, and Lammers C Stachniss. An effective classification system for separating sugar beets and weeds for precision farming applications. In *2016 IEEE International Conference on Robotics and Automation (ICRA*, pages 5157–5163. IEEE, 2016.

[16] Ciro Potena, Daniele Nardi, and Alberto Pretto. Fast and Accurate Crop and Weed Identification with Summarized Train Sets for Precision Agriculture. In *Intelligent Autonomous Systems 14*, pages 105–121. Springer, Cham, Cham, July 2016.

[17] M Di Cicco, C Potena, G Grisetti, and A Pretto. Automatic model based dataset generation for fast and accurate crop and weeds detection. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS*, pages 5188–5195, 2017.

[18] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[19] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Towards drone racing with a pixel processor array

Colin Greatwood[1], Laurie Bose[1], Thomas Richardson[1], Walterio Mayol-Cuevas[1], Robert Clarke[1]
Jianing Chen[2], Stephen J. Carey[2] and Piotr Dudek[2]

Abstract

Drone racing is an interesting scenario for an agile MAV due to the need for rapid response and high accelerations. In this paper we use a Pixel Processor Array (PPA) demonstrating the marriage of perception and compute capabilities on the same device. A Pixel Processor Array (PPA) consists of a parallel array of processing elements, each of which features light capture, processing and storage capabilities allowing for various image processing tasks to be efficiently performed directly on the sensor itself. This paper presents the use of a PPA for gate detection and location in a typical drone racing scenario. Conventional sensing techniques typically require significant processing overheads on separate hardware, resulting in lower frame rates and higher power consumption than is possible to achieve with a PPA. The results given here demonstrate gate detection and location with real-time planning to account for uncertainty in the gate location. Additionally, the PPA only needs to output specific information such as the estimated target location variables, rather than having to output entire images. This significantly reduces the bandwidth required for communication between the sensor and on-board computer, further enabling a high frame rate, low power operation.

1 Introduction

Autonomous Drone Racing (ADR) requires a Micro Air Vehicle (MAV) to fly with high speed, agility and accuracy. This agile control also requires suitable perception that can enable flight through small racing gates and around obstacles. An aircraft with such capabilities would be of great use in many future robotics applications. In such a vehicle it is important to minimise the size, mass and power consumption of on-board components such as sensors and processors. Image sensors must also be able to cope with scenes that move at high speed without suffering from motion blur. Sensing must be rapid, accurate, robust and take place with minimal delay in order to allow for the rapid control decisions required for precision flying and obstacle avoidance. Pixel Processor Arrays (PPAs) are a great fit for ADR and many other robotics applications due to their size, speed and low power requirements.

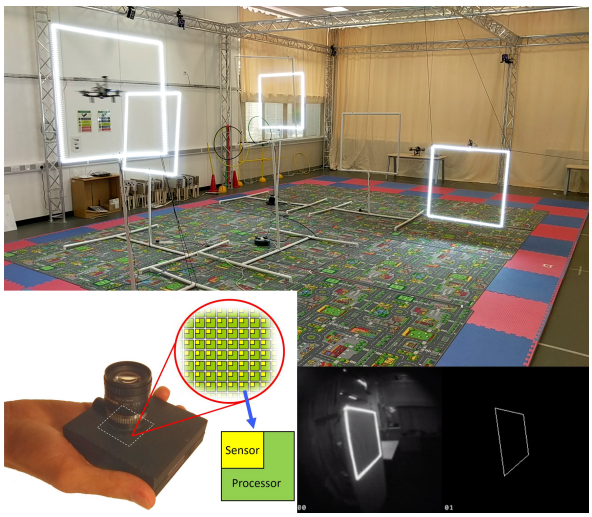


Figure 1: Racing layout; PPA structure; and the SCAMP-5 system tracking a single gate.

Autonomous navigation for aerial robotics has historically leveraged Simultaneous Localisation and Mapping (SLAM), where powerful single board computers or FPGAs process information from vision sensors such as stereo cameras and RGB-D cameras e.g. [1–5]. More recently in ADR competitions however, focus has shifted to high frame rate estimation of the platform's motion, rather than rapidly generating a detailed map of the environment. This rapid frame estimation allows for agile control and manoeuvring of the vehicle and enables accurate map generation to take place, both at a lower frame rate and through post flight data processing.

Another direction recently under development is the use of delta sensors or dynamic visual sensors such as the DVS, reporting only pixel event locations that changed intensity. Such sensors have very low latency and energy usage, however their data often needs to be deeply processed and events pooled to construct whole images before being usable for navigation [6].

Targeting the current interest in ADR, this paper presents an autonomous drone racing strategy using a Pixel Processor

*This work was conducted at the Bristol Robotics Laboratory
[1]Faculty of Engineering, Aerospace and Computer Science, University of Bristol, Bristol, England
thomas.richardson@bristol.ac.uk
[2]School of Electrical and Electronic Engineering, The University of Manchester, Manchester, England p.dudek@manchester.ac.uk

Array (PPA) camera to estimate gate locations at an average frame rate of 500 Hz. The approach taken here is similar to the previous competition winning method used at IROS Autonomous Drone Racing 2018 [7]; A key difference being the onboard sensing. Where the authors of 2018 [7] used deep-learning to estimate relative gate pose at 10 Hz on an Intel UpBoard, this work instead uses a PPA sensor to detect the racing gates and provide information to the control systems.

In contrast to many conventional image sensors, PPA sensors, for example the SCAMP-5 system used in this work, are capable of high frame rate, low latency, vision processing workloads. This has the potential to greatly reduce the workload exacted on any associated on-board computer. PPA sensors consist of a parallel array of processing elements, each featuring light capture, processing and storage capabilities allowing for various image processing tasks to be efficiently performed directly on the sensor [8, 9]. Crucially, the PPA is capable of outputting only required information such as the estimated location of a target, rather than having to output entire images. This vastly reduces the bandwidth required per frame in communication between the sensor and on-board computer, enabling high frame rate, low power sensing.

Figure 1 shows the overall layout of the gates and their brightness relative to the environment. The lower portion of the figure shows the raw SCAMP-5 system image captured alongside the binarized image containing the extracted gate.

The following section outlines the strategy taken for gate detection and vehicle control. Section 3 provides the experimental setup in the flight arena at the Bristol Robotics Laboratory; and Section 4 provides the test results from autonomous flight tests of the drone with the on-board SCAMP-5 system sensing the gates, with real-time planning to allow for uncertainties in their position.

2 METHOD

The system presented here follows recent autonomous drone racing strategies, splitting the problem into perception, and combined planning and control. The novelty, contribution and focus of this paper is in the programming and use of the SCAMP-5 system for high frame rate detection and localization of the racing gates. Detected gates given in the vehicle's frame of reference are combined with the vehicle's state estimate to produce a filtered estimate of the gate poses. The planning and control is subsequently performed for the flight tests using the Perception Aware Model Predictive Controller (PAMPC) framework presented by Falanga et al. [10].

2.1 Gate Sensing using the SCAMP-5 System

Gate detection is performed using a SCAMP-5 system attached to the front of the vehicle. The system is programmed to detect potential gates within each frame, sending only their size and location within the image frame back to the on-board computer, hence consisting of only a short stream of bytes per frame. This significantly reduces both the computation overhead in the controller, and bandwidth required for communication with the sensor. Additionally the gate detection algorithm is designed to exploit the parallel features of PPAs, which when combined with the small data transfer per gate, allows the algorithm to be performed at frame rates of up to 1500 Hz. Performing detection at this frame rate has the added benefit of allowing for a short exposure time, effectively eliminating motion blur, and improving gate detection accuracy under rapid camera motion.

2.2 PPA Algorithms

The approach used to detect gates makes heavy use of two different pieces of functionality on the SCAMP-5 system. First the ability to locate a set pixel (ie. pixel of value of +1) within a binary image stored upon the PE array, and secondly the ability to perform a parallel flood fill upon such a binary image. A second binary image is used to control this flood fill operation, restricting flooding propagation to only pixels which are set in this control image.

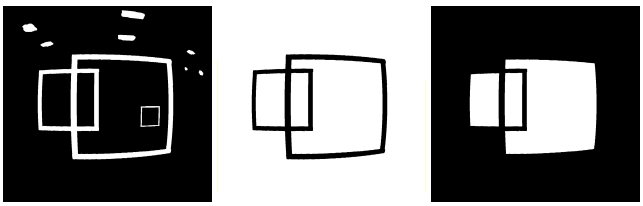


Figure 2: Left to right, a white shape is extracted from the binary camera image, inverted, and then flooded black from the image bounds leaving only the contained shapes within.

Using these features gate detection proceeds by extracting separate shapes from within a binary thresholded camera image, and then extracting the shapes contained within each of these as shown in Figure 2. Different combinations of the shapes contained within each extracted shape are the tested to determine if they constitute a potential gate. This simply involves extracting approximations of the four corners of the gate and evaluating how well the polygon spanning these vertices fits the shapes as illustrated in Figure 3. Pseudo Code for this algorithm is listed in Algorithm 1.

2.3 Gate Positioning

The PPA's output is transformed into an estimated gate pose relative to the drone through knowledge of the camera's intrinsic parameters. The vehicle's state estimate is then used to transform these relative gate poses into the world frame. The world frame positions are compared with a set of prior estimates provided to the vehicle before flight. Only gate updates that fall close enough to the existing gate predictions are accepted as valid gate updates used to drive the flight path.

2.4 Control

The control architecture follows that of Falanga et al. [10]. With the PAMPC controller, there are two objectives; the first is to have the vehicle follow a reference trajectory; and the

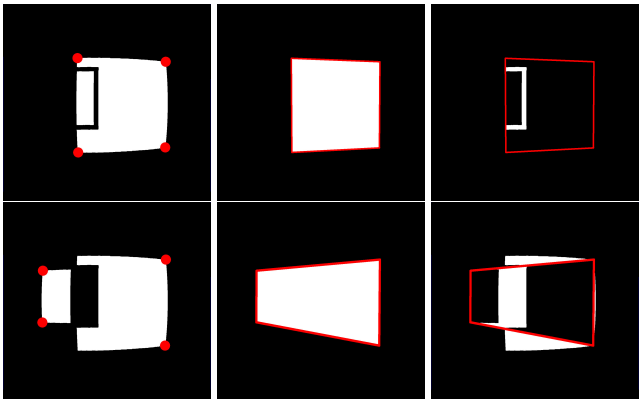


Figure 3: Two examples of combining and testing contained shapes from Figure 2 as potential gates. Approximate gate corners are extracted and a filled polygon fitted to these vertices. An XOR is then performed between these two images, with the number of remaining pixels indicating how closely this polygon fitted the shape.

second is to bring the next gate into the field of view of the SCAMP-5 system. In this work, the controller acts over a 2 s horizon.

The reference trajectory is a simple linear progression along a set of way-points, and for this work maintains a constant speed along the whole path. This can result in physically unrealisable accelerations at the transitions between each pair of way-points. This can in turn limit the maximum speed that the reference trajectory can be generated for while having the vehicle successfully follow it. The way-points are generated in pairs either side of each gate, such that the reference trajectory passes through the centre of each gate, to ensure as tight tracking as possible to the centre of each gate. The offset of the way-points either side of the gate was set at 1.5 m, which worked well with the chosen reference trajectory speed. In addition to these generated way-points, a start and end way-point were added to the overall list.

In this work, the PAMPC controller's point of interest, which controls where the vehicle attempts to point the PPA, is set to be the way-point after the next gate. This provides the fastest possible acquisition of the next gate of interest using the onboard PPA. These gate updates are subsequently used in real-time to update the reference trajectory by updating the endpoint for the reference trajectory generation. This rather coarse approach to the control results in some sharp discontinuities in the reference trajectory when a gate update is incorporated. However, this was found to be sufficient for these tests. Future control work will focus on smoothing out the transitions in the reference trajectory when a gate estimate is incorporated, and generating a smoother reference trajectory accounting for the gate topology.

Algorithm 1 $Extract\_Gates(A, \alpha)$

Input and Output
A *//Binary Camera Image*
$\alpha \in \mathbb{N}$ *// Gate Error Threshold In Number Of Pixels*
G *// List of detected Gates as corners*

while $Global\_OR(A)$ **do**
  $n = 0$ *//Reset inner Shapes Counter*
  $B = Extract\_Shape(A)$ *//Extract White Shape*
  $C = Flood\_From\_Edges(NOT(B))$
  $B = NOT(OR(B, C))$ *//Get inner shapes*
  while $Global\_OR(B)$ **do**
    $S_n = Extract\_Shape(B)$ *//Extract inner Shape*
    $n++$ *//Increment inner Shape Counter*
  end while
  for $i = 0$ to n **do**
    for $j = 0$ to n **do**
      $S = OR(S_i, S_j)$ *//Combine inner shapes*
      $corners = Extract\_Gate\_Corners(S)$
      $C = Draw\_Filled\_Polygon(corners)$
      $C = XOR(C, S)$ *//Generate Error Image*
      $Err = Count\_Set\_Pixels(C)$
      if $Err < \alpha$ **then**
        $G = G \cup \{corners\}$ *//Add corners List G*
      end if
    end for
  end for
end while
return $Gates$

3 EXPERIMENTAL SETUP

3.1 SCAMP-5 System

Gate detection was performed using SCAMP-5 system [9, 11, 12] specifically programmed for the task. No other device was used in directly processing visual data. The SCAMP-5 system integrated circuit features an array of 256×256 processing elements (PEs), each capable of light capture, storage and processing of visual data - effectively putting a small "microprocessor" inside every pixel of the sensor array. The pixels feature a photosensor, local analogue and digital memory, and the ability to perform various logic and arithmetic operations. The SCAMP-5 system is attached to the front of the vehicle as shown in Figure 4. Each PE may also communicate with its four neighbouring elements in the array, making it possible to transfer register data across PEs. A programmable controller chip issues identical instructions to each PE, which then all perform said instruction simultaneously. In this way processing follows the standard single instruction multiple data (SIMD) approach and allows for efficient parallel processing. Vision algorithms can then be performed directly upon the pixel array, without ever transmitting the images out of the sensor.

By only sending the meaningful data such as the values relating to gate locations, there is a significant decrease in the bandwidth and hence power required during operation. This approach allows many visual tasks to be conducted at very high frame-rates (such as at 100 000 fps in [9]), something typically not possible using the standard visual processing pipeline. SCAMP-5 system is also low power, requiring below 2 W, which compares well with GPU-based approaches that while parallel, require 10s-100s of Watts.

3.2 *Flight Hardware*

A custom quadrotor, shown in Figure 4, was designed and built to carry the SCAMP-5 system, it weighs 1kg with the PPA installed and measures 400 mm diagonally between rotors. In the work presented in this paper, the sensor was mounted facing forwards with a 4.5 mm lens providing a 107° field of view.

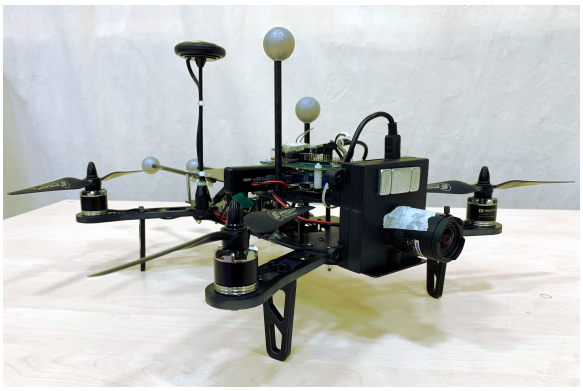


Figure 4: Custom quadrotor used for experiments. SCAMP-5 system facing forwards for gate sensing.

An ODROID XU4 single board Linux computer is fitted to the top of the quadrotor and enables the SCAMP-5 system and 'Pixhawk' autopilot to both be integrated within the Robot Operating System (ROS) for rapid development and system testing. Data is passed from the SCAMP-5 system over USB to the ODROID, whilst flight data from the Pixhawk is sent via a serial UART link. These communication links are summarised in Figure 5. If the ROS system was not used, the SCAMP-5 system has an M4 processor that could be used to carry out the computations currently programmed on the ODROID, and it could talk directly to the Pixhawk with the available serial link. The final overall mass of the system could therefore be reduced significantly if the requirement for rapid development were removed.

The outer loop control system runs on the ground, with only the low-level attitude controller running on-board the vehicle. Control inputs are sent over a Laird RM024 whilst data from the SCAMP-5 system is sent back to ground over WiFi using TCPROS. Position information for the vehicle is provided by a series of Vicon cameras and associated tracking software. The same system is used to track the gate positions, thereby providing a ground truth for the gate position estimates.

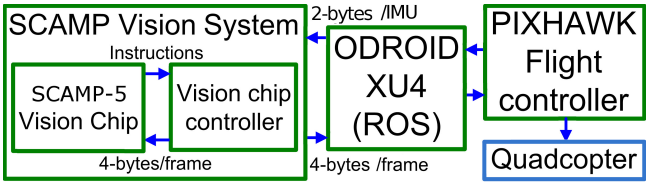


Figure 5: Block diagram of hardware. ODROID is used for rapid development and debugging with ROS and passes data between flight controller, SCAMP-5 system and the ROS system. It does not do any further computation.

3.3 *Initial Testing*

Six square gates were constructed with an array of LEDs around the outer edges and a width and height of 1 m. These gates were then positioned at various heights and locations to form a course, representative of the one used in IROS 2018. No particular consideration was given to the visibility of the gates while traversing the course. In this work, they were always vertical, although the method could be extended to deal with inclined gates. Figure 6 shows a top-down map of the course, highlighting the overall size, direction and numbering of each of the gates.

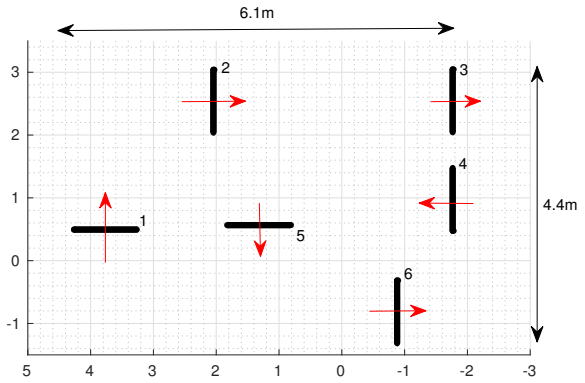


Figure 6: Map of the course

Initial testing of the system was carried out using simulated SCAMP-5 system output feeding back to the the PAMPC controller. This simulator used the output of the Vicon tracking to generate idealised inputs for the gate estimation. The system was found to be highly susceptible to lag, increasing the motivation for the low latency image processing pipeline provided by the SCAMP-5 system. In parallel with the testing of the controller, the SCAMP-5 system output was compared to the ground truth reference points measured using Vicon

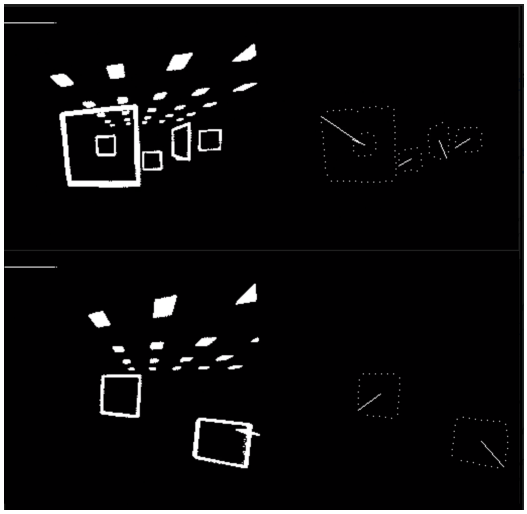


Figure 7: SCAMP-5 system simulated output for multiple gates

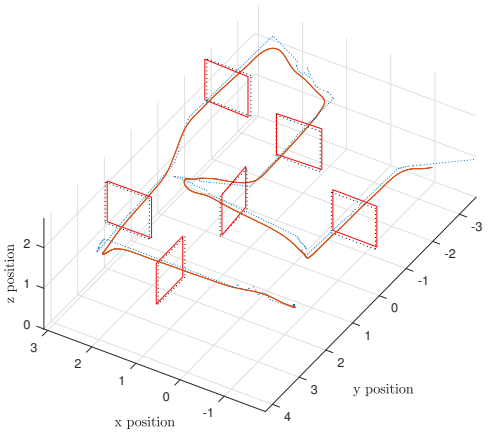


Figure 8: Plot showing track taken by drone through gates. No intentional shift of gates, reference speed $1.5\,\mathrm{m\,s^{-1}}$. Reference trajectory and prior gate estimates dotted, measured trajectory and positions solid.

Figure 7 shows some examples of simulated binarized frames captured by the SCAMP-5 system used for testing. From this image, the algorithm running on the system detects any gates present and transfers their image location and size to the on-board computer. This information is then utilised by the controller to update the vehicle's estimate of the gate positions. In these examples a number of gates are visible along with the ceiling lights of the arena, whose shapes are rejected by the gate detection algorithm.

4 RESULTS

Initial experimentation with the SCAMP-5 system output feeding into the control loop was carried out with the speed of the reference trajectory set to $1.5\,\mathrm{m\,s^{-1}}$. The gates were within $5\,\mathrm{cm}$ of the positions given in the prior estimates provided to the system. The vehicle successfully traversed the course completing it in approximately $17.5\,\mathrm{s}$. Figure 8 shows the reference trajectory and the measured vehicle position relative to the prior estimate and measured gate positions for this $1.5\,\mathrm{m\,s^{-1}}$ run. Of note are a number of sharp discontinuities in the reference trajectory caused by the gate updates shifting the end way-points for the linear trajectory generation. As mentioned previously, future work will focus on smoothing the effect of incorporating the updated gate estimates. There is very little shift between the measured gate positions and those provided as prior estimates as seen by the close correspondence of the dotted and solid gate positions.

For the following set of results, selected gates were moved prior to the flights. Three of the gates in the course were shifted laterally by up to 75% of a gate width, namely gates 2, 4 and 6. The gates remained in approximately the same plane as their pre-shift positions, though it should be noted that this is not required for the control strategy selected. The reference speed was also increased to $2.3\,\mathrm{m\,s^{-1}}$. With the current control implementation, this allowed for robust and repeatable completion of the course. Figure 9 shows successful completion of the course for this reference speed in approximately $12.5\,\mathrm{s}$. The shift in gate positions can be seen by the offset between the prior estimates provided (dotted) and those measured by the Vicon system (solid). The reference trajectory is noticeably noisier than that seen in Figure 8 which was at a refernce speed of $1.5\,\mathrm{m\,s^{-1}}$, but future smoothing of the gate updates will solve this problem. Variable reference trajectory speed and way-point offsets will also allow the overall speed to be increased.

Table 1 provides the maximum speed along all three axes and the maximum overall velocity during this run. The maximum roll and pitch angles experienced by the vehicle are also given.

Value	Absolute Maximum
x-velocity	$3.04\,\mathrm{m\,s^{-1}}$
y-velocity	$2.83\,\mathrm{m\,s^{-1}}$
z-velocity	$1.72\,\mathrm{m\,s^{-1}}$
Total velocity	$3.14\,\mathrm{m\,s^{-1}}$
Roll	$39.4°$
Pitch	$44.0°$

Table 1: Maximum values reached during the run shown in Figure 9

Figure 10 shows the location of the vehicle as it passes through each gate relative to the mean estimated gate position. The plot spans the overall cross-section of the gate. It can be seen that these are closely grouped, indicating that the vehicle is consistently passing through the gate at the targeted

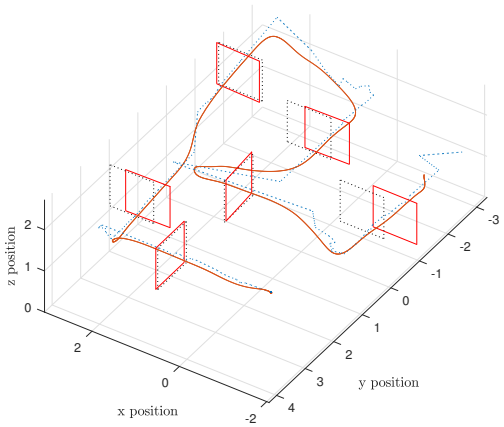


Figure 9: Plot showing track taken by drone through gates. Gates 2,4 & 6 shifted, reference speed $2.3\,\mathrm{m\,s^{-1}}$. Reference trajectory and prior gate estimates dotted, measured trajectory and positions solid.

location, as identified by the SCAMP-5 system tracking estimate. The vehicle trajectory has also been included for $0.5\,\mathrm{m}$ on both sides of each gate pass-through, indicating that the vehicle is consistent both in the approach and the departure for each gate. The pass-through locations for all six gates are within a $20\,\mathrm{cm}$ square, which together with the possible improvements identified above, indicate that there is still significant improvement possible in terms of maximum speed, acceleration and the minimum time to complete the course.

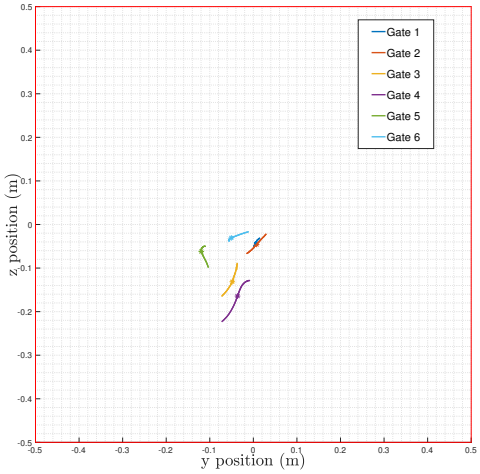


Figure 10: Accuracy of flight path relative to the estimated gate positions. Gates 2, 4 & 6 shifted. Plot spans overall size of gate, $0.5\,\mathrm{m}$ of the trajectories either side of the gate are plotted.

Figure 11 shows the location of the vehicle as it passes through each gate in terms of the measured position from the Vicon cameras. This is shown as a ground truth, and it shows little difference in terms of the grouping when compared to Figure 10. From these two plots, it can therefore be concluded that the dynamic SCAMP-5 system driven estimate of the gate position and the vehicle control are both sufficiently accurate for further speed increases. The small difference between the two plots, Figure 10 and Figure 11 could be due to a number of factors, namely a small offset in the orientation and/or position of the SCAMP-5 system on the vehicle; an error in the state estimate of the gate positions; or an error in the measurement of the true gate position. The combined errors though are very small and are not currently the limiting factor with regards to overall vehicle performance.

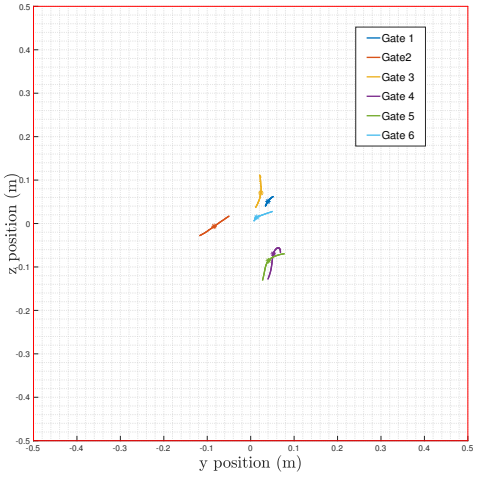


Figure 11: Accuracy of flight path relative to the measured gate positions. Gates 2,4 & 6 shifted. Plot spans overall size of gate, $0.5\,\mathrm{m}$ of the trajectories either side of the gate are plotted.

5 Conclusions

This work has shown that a novel Pixel Processor Array (PPA) device can be used to correct imperfect knowledge of a drone racing course in real-time. The high frame rate achievable with the SCAMP-5 system - i.e. an average of $500\,\mathrm{Hz}$ - has been shown to provide robust and reliable estimates of the true gate positions. This has been carried out on a representative drone racing course, with rapid and significant changes in vehicle trajectory. For the results shown, the position estimate based on the PPA sensing was not found to be the overall limiting factor for speeds and accelerations experienced.

Key limitations in the control strategy used have been identified and these will be addressed in future work to find the limits in terms of speed and acceleration for the scenario considered. These results show the PPAs are likely to be one

of a suite of sensors used on future small agile drones when manoeuvring rapidly in an unknown environment.

ACKNOWLEDGEMENTS AND DATA ACCESS STATEMENT

Supported by UK EPSRC EP/M019454/1 and EP/M019284/1. The nature of the PPA means that the data used for evaluation in this work is never recorded.

REFERENCES

[1] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor. In *Robotics: Science and Systems*, volume 1. Berlin, Germany, 2013.

[2] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 298–304. IEEE, 2015.

[3] Changhong Fu, Adrian Carrio, and Pascual Campoy. Efficient visual odometry and mapping for unmanned aerial vehicle using arm-based stereo vision pre-processing system. In *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, pages 957–962. IEEE, 2015.

[4] Reuben Strydom, Saul Thurrowgood, and Mandyam V Srinivasan. Visual odometry: autonomous uav navigation using optic flow and stereo. In *Australasian Conference on Robotics and Automation (ACRA)*, pages 1–10. Australian Robotics and Automation Association, 2014.

[5] Roberto G Valenti, Ivan Dryanovski, Carlos Jaramillo, Daniel Perea Ström, and Jizhong Xiao. Autonomous quadrotor flight using onboard rgb-d visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5233–5238. IEEE, 2014.

[6] A.I. Maqueda, A. Loquercio, G. Gallego, N. Garcia, and D. Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *CVPR*. IEEE, 2018.

[7] 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. *arXiv preprint arXiv:1810.06224*, 2018.

[8] Alexey Lopich and Piotr Dudek. A SIMD cellular processor array vision chip with asynchronous processing capabilities. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58(10):2420–2431, 2011.

[9] Stephen J Carey, Alexey Lopich, David RW Barr, Bin Wang, and Piotr Dudek. A 100,000 fps vision sensor with embedded 535gops/w 256× 256 simd processor array. In *VLSI Circuits (VLSIC), 2013 Symposium on*, pages C182–C183. IEEE, 2013.

[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)*, pages 1–8. IEEE, 2018.

[11] Julien NP Martel, Lorenz K Müller, Stephen J Carey, and Piotr Dudek. Parallel hdr tone mapping and autofocus on a cellular processor array vision chip. In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, pages 1430–1433. IEEE, 2016.

[12] Julien NP Martel, Lorenz K Mueller, Stephen J Carey, and Piotr Dudek. A real-time high dynamic range vision system with tone mapping for automotive applications. *CNNA 2016*, 2016.

Visual-Inertial Sensor Fusion with a Bio-Inspired Polarization Compass for Navigation of MAVs

Florian Steidle*, Wolfgang Stürzl, Rudolph Triebel
Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Germany

ABSTRACT

We present the integration of a polarization compass in a visual-inertial sensor fusion framework onboard a Micro Aerial Vehicle (MAV). The polarization compass estimates the position of the sun indirectly from the pattern of skylight polarization even in cases where the sun is not visible. It is based on a polarization sensor which consists of a standard RGB camera and a small polarizing unit that creates three polarization images on the camera sensor. Due to its low weight and compact size it is ideally suited for small aerial systems. The readings from the polarization compass are fused with angular rate and acceleration measurements from an Inertial Measurement Unit (IMU) and the 6 Degrees of Freedom (DOF) pose changes from the Visual Odometry (VO) in an indirect extended Kalman filter (EKF). Two different approaches to integrate the readings from the polarization compass in the filter are presented and compared. We show in experiments that adding a compass to visual-inertial sensor fusion does not only eliminate the drift of yaw angle estimates but also improves overall state estimation of the system.

1 INTRODUCTION

Due to the complementary information they provide, the fusion of visual and inertial data is widely used for state-estimation of MAVs, in particular in environments where global navigation satellite systems (GNSS) are unavailable or unreliable. While it is possible to estimate absolute roll and pitch angles based on acceleration measurements, the yaw angle is subject to drift as it can only be estimated by continuously integrating orientation differences. Therefore, magnetometers are often added. By measuring the magnetic field of the Earth, the absolute yaw angle can be estimated. In this case all degrees of rotation are observable, as well as the angular velocity and acceleration biases, which results in higher overall accuracy of the system. However, magnetic compasses can be disturbed by magnetic objects or electrical devices. Beside the magnetic field of the earth the position of the sun can be used as a compass cue and even if the sun is not directly visible its position can be estimated indirectly via the polarization pattern of the sky light. An other advantage compared to a magnetic compass is its insensitivity to interference fields caused, for instance, by electric devices. This motivated us to equip our multicopter "ARDEA" [7] with a bio-inspired polarization sensor and integrate its compass measurements in our Visual Inertial Navigation System (VINS).

Figure 1: Multicopter "ARDEA" with a frame in triangle shape, three pairs of counter-rotating rotors and a sensor suite mainly consisting of an IMU, two pairs of wide-angle stereo cameras and an insect-inspired polarization compass (highlighted by red ellipse).

In [1] a polarization compass was fused with IMU readings, but VOs were not used, the measurement equation for the polarization is different and instead of a indirect EKF a complementary filter was used. In [2] accelerations and angular rates are fused with readings from a polarization compass in a Kalman filter. But they also used readings from a GNSS and only estimated the orientation of their device.

The main contributions of our approach are a system for pose estimation onboard a MAV which does neither depend on external infrastructure nor readings of the magnetic field and nonetheless can provide a drift-free 3 DOF orientation estimate. Its accuracy is improved in comparison to a pure VINS and it avoids the usage of 3 DOF measurements of the direction vector to the sun with almost singular covariance matrix by projecting the measurement errors to different two dimensional subspaces.

In the following we describe the polarization compass in Section 2, the approach to combine data from different sensors in Section 3, the experiments in Section 4 and finally conclude the paper with Section 5.

*Email address: florian.steidle@dlr.de

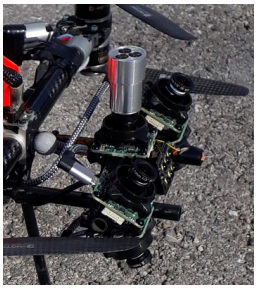



Figure 2: The polarization sensor, a standard camera with cylindrical polarizer unit mounted in front of the camera lens is positioned between the two stereo camera pairs (left). The sensor was inspired by the ocelli of orchid bees. As highlighted by the red circle in the inset of the right figure (shown is a close up view of the head of the bee *Euglossa imperialis*), bees have three simple eyes with polarization sensitive photoreceptors (photos courtesy of Emily Baird, Stockholm University). In orchid bees, the preferred polarization orientation is very similar within each eye but differs between eyes by somewhat less than $60°$ [3].

2 POLARIZATION COMPASS

We briefly describe the polarization sensor and summarize the computation of the sun vector. For more details see [4]. The polarization sensor utilized on our multicopter is identical to the one described in [4] except for the camera sensor. It is replaced by an USB3 camera with Sony IMX265 CMOS color sensor (IDS UI 3271LE-C).

2.1 Sky polarization pattern as compass cue

Scattering of sun light in the atmosphere creates a characteristic polarization pattern in the sky that is essentially symmetric with respect to the position of the sun. The degree of polarization is low close to the sun, increases with angular distance from the sun up to $90°$ and decreases for larger angles. Measuring the polarization, in particular its orientation, which is known to be more reliable than the degree of polarization [5], even just for small regions of the sky allows to estimate the sun position or at least its azimuth in cases where the sun is occluded by clouds, trees or buildings. Therefore, similar to the sun, the polarization pattern can be used as a compass. Interestingly, insects are known to use both, direct sun position and polarization pattern for orientation [6]. Bees and many other insect species, like desert ants, have a specialized region in the upper part of their compound eyes that are sensitive to polarization. In addition, there is recent evidence that the three simple eyes of bees located at the top of the head in between the two compound eyes, the "ocelli" (see right sub figure of Figure 2), might also play a role in polarization sensing. While each ocellum contains photoreceptors of similar preferred orientation, the preferred orientations of all three ocelli differ strongly. This arrangement of polarization sensitive photoreceptors in bees inspired the polarization sensor design and its use as compass cue on our multicopter ARDEA. As sky light is predominantly linearly polarized, i.e. contains almost no circular or elliptical polarization, three is the minimum number of linear polarizers sufficient for estimating all relevant polarization parameters.

2.2 Polarization sensor and multi-camera setup on MAV

As shown in Figure 1 and 2 the polarization sensor is placed between Ardea's "compound eyes" that consist of two wide-angle cameras on either side. The arrangement of these cameras provides a very large stereo FOV of approx. $240°$ vertically. As described in [7], each wide-angle camera is remapped to two virtual pinhole cameras to allow for efficient image processing.

The polarization sensor consists of a standard camera with a small-aperture lens to which the cylindrical polarizer unit is attached, see Figure 2. By means of this unit the camera image contains three basically identical images of the sky seen through three differently oriented linear polarizers (Figure 3 left). The preferred polarization orientations differ by $60°$.

In contrast to several devices based on photodiodes, e.g. [8, 9], the polarization sensor allows to estimate a large number of polarization vectors, which – in combination with a comparatively large field of view of approx. $56°$ – enables the estimation of the "sun vector", i.e. not only the azimuth of the sun but also its elevation angle can be inferred.

2.3 Remapping and polarization estimation

Raw images of the polarization camera of size 800×800 pixels are de-bayered, scaled down by factor 0.5 and then remapped to three polarization images (120×120 pixels) with constant radial resolution of $0.5°$ per pixel. From the intensity differences of corresponding pixels, i.e. pixels with same viewing directions as estimated by a three-camera-calibration using the DLR-CalDe/CalLab tool [10], the angle ϕ and degree of polarization δ can be determined for each pixel of the reference image (the remapped sub-image '1'), see [4] for details. By retracing the pixel rays, the polarization orientation on the sky sphere can be computed, which we describe by the 3D unit vector $\pm\mathbf{f}_i$ in the following, where i is the index of the pixel with image coordinates (u_i, v_i). If the multicopter is aligned with the north direction then the u-axis of the camera image points towards the west and the v-axis towards the south (see p-frame in Figure 4). The exact transformation between the polarization camera frame and the IMU or body frame of the multicopter was estimated based on an extrinsic calibration of the polarization camera and the topmost left virtual pinhole camera and an IMU-to-camera calibration between the reference pinhole camera and the IMU.

2.4 Sun vector estimation

As described in [4], the sun vector ${}^{\mathrm{p}}\mathbf{p}_{\mathrm{s}}$ can be estimated by minimizing

$$E({}^{\mathrm{p}}\mathbf{p}_{\mathrm{s}}) = \sum_i \tilde{w}_i (\pm \mathbf{f}_i^{\top} {}^{\mathrm{p}}\mathbf{p}_{\mathrm{s}})^2 = {}^{\mathrm{p}}\mathbf{p}_{\mathrm{s}}^{\top} \left(\sum_i \tilde{w}_i \mathbf{f}_i \mathbf{f}_i^{\top} \right) {}^{\mathrm{p}}\mathbf{p}_{\mathrm{s}} \quad (1)$$

under the constraint $\|{}^{\mathrm{p}}\mathbf{p}_{\mathrm{s}}\| = 1$. $\tilde{w}_i = (\sum_k w_k)^{-1} w_i$ are normalized weights. The weights w_i basically depend on the degree of polarization and the "blueness" of the corresponding pixel favoring "sky-pixels". Equation 1 is motivated by the fact that ideally all polarization vectors $\{\mathbf{f}_i\}$ are orthogonal to the observer-sun axis, i.e. the sun vector ${}^{\mathrm{c_p}}\mathbf{p}_{\mathrm{s}}$. Pre-whitening [11] of matrix $\mathbf{P} = \sum_i \tilde{w}_i \mathbf{f}_i \mathbf{f}_i^{\top}$ is used to reduce the bias that would result from solving the eigenvalue problem defined in Equation 1 directly. Assuming independent and identically distributed errors with standard deviation σ, the covariance matrix of the sun vector can be estimated,

$$\mathbf{\Sigma}_{{}^{\mathrm{p}}\mathbf{p}_{\mathrm{s}}} \approx \sigma^2 \, \mathbf{Q} \sum_i \tilde{w}_i^2 (1 - ({}^{\mathrm{p}}\mathbf{p}_{\mathrm{s}}{}^{\top} \mathbf{e}_i)^2) \mathbf{f}_i \mathbf{f}_i^{\top} \, \mathbf{Q}^{\top} \quad . \quad (2)$$

$\mathbf{Q}$ is a matrix describing rotation, scaling and projection onto the plane orthogonal to the estimated sun vector, and $\mathbf{e}_i$ is the viewing direction of pixel i.

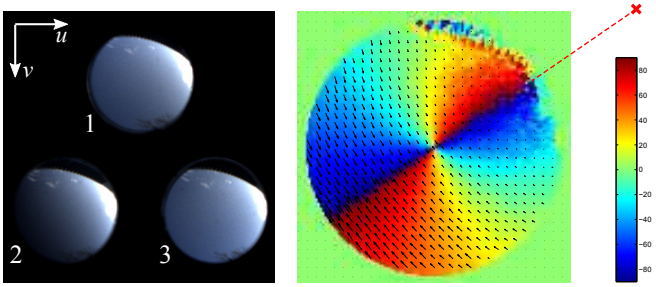


Figure 3: Estimation of sun position from the three images of the polarization sensor. Left: The camera image containing the three sub-images after de-bayering. In this example the sun is located outside the field of view of the camera. A bright cloud visible in the upper right corner of the sub-images indicates the approximate sun direction. Intensity differences between the three sub-images allow to estimate polarization degree and angle for each pixel. For example, quite strong intensity differences can be observed in the lower left corner of the three sub-images indicating high degree of polarization. Right: Shown are the sky polarization angles, i.e. the angles of the polarization vectors with respect to the local meridians (great circles of constant azimuth) in color code, ranging from $-90°$ (blue) to $+90°$ (red), and polarization vectors $\mathbf{f}_i$ with length scaled according to weight w_i (black arrows), projected onto the image plane. The red cross in the upper right corners depicts the estimated position of the sun (approx. $-34.5°$ azimuth and $+36°$ elevation angle with respect to the camera frame).

3 FUSION

3.1 Extended Kalman filter based visual-inertial odometry

In [12] and [13] an indirect, extended Kalman filter was introduced that combines the readings from an IMU and a single VO. In [7] the filter was extended to cope with multiple VOs.

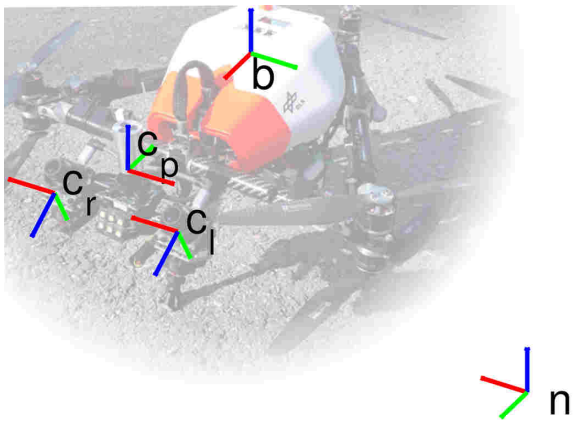


Figure 4: An image of ARDEA with the navigation frame (n-frame), the body frame (b-frame), the frames of one stereo pair ($\mathrm{c_l}$- and $\mathrm{c_r}$-frame) and the frame of the camera with the polarization compass (p-frame).

The main state $\mathbf{x}$ of the filter is defined by

$$\mathbf{x} = \begin{bmatrix} {}^{\mathrm{n}}_{\mathrm{b}}\mathbf{p}^{\top} & {}^{\mathrm{n}}_{\mathrm{b}}\mathbf{v}^{\top} & {}^{\mathrm{n}}_{\mathrm{b}}\mathbf{q}^{\top} & {}^{\mathrm{b}}\mathbf{b}_{\mathrm{a}}^{\top} & {}^{\mathrm{b}}\mathbf{b}_{\omega}^{\top} \end{bmatrix}^{\top}, \quad (3)$$

with the position ${}^{\mathrm{n}}_{\mathrm{b}}\mathbf{p} \in \mathbb{R}^3$ of the body frame (b-frame) relative to an earth-fixed, inertial frame (n-frame), the velocity ${}^{\mathrm{n}}_{\mathrm{b}}\mathbf{v} \in \mathbb{R}^3$, the orientation ${}^{\mathrm{n}}_{\mathrm{b}}\mathbf{q} \in \mathbb{R}^4$ represented by a unit quaternion and the acceleration ${}^{\mathrm{b}}\mathbf{b}_{\mathrm{a}} \in \mathbb{R}^3$ and angular rate ${}^{\mathrm{b}}\mathbf{b}_{\omega} \in \mathbb{R}^3$ biases of the IMU. The relationship between the main coordinate systems involved is shown in Figure 4.
If a raw measurement from a sensor is taken, its transmission and processing needs time and is therefore delayed when the results are available to the filter. For some sensors, e.g. IMUs the delay can often be neglected, for other sensors, e. g. cameras the delay usually has to be taken into account. Therefore, parts of the main state that are necessary to process the delayed measurements, when they arrive have to be augmented to the state. The final state consists of the main state $\mathbf{x}$ and an arbitrary number of augmented states $\mathbf{x}_{\mathrm{aug}}$.
A measurement from the VO that becomes available at time t_{k} can be described by

$$\mathbf{h}_k = \mathbf{h}(\mathbf{x}_{k-n}, \mathbf{x}_{k-m}), \quad (4)$$

where the states at time t_{k-n} and t_{k-m} must be part of the augmented state.
Instead of estimating the state directly, it is possible to estimate the errors of the state. This has several advantages, e.g. system dynamics can be decoupled from error dynamics, a

sophisticated model of the system is not needed and rotation errors can be locally described with a minimal representation. The indirect formulation is given by

$$\delta\mathbf{x} = \begin{bmatrix} {}^{\mathrm{n}}_{\mathrm{b}}\delta\mathbf{p}^\top & {}^{\mathrm{n}}_{\mathrm{b}}\delta\mathbf{v}^\top & {}^{\mathrm{n}}_{\mathrm{b}}\delta\boldsymbol{\phi}^\top & {}^{\mathrm{b}}\delta\mathbf{b}_{\mathrm{a}}^\top & {}^{\mathrm{b}}\delta\mathbf{b}_{\omega}^\top \end{bmatrix}^\top, \quad (5)$$

where all errors are in the form of ${}^{\mathrm{n}}_{\mathrm{b}}\hat{\mathbf{p}} = {}^{\mathrm{n}}_{\mathrm{b}}\mathbf{p} + {}^{\mathrm{n}}_{\mathrm{b}}\delta\mathbf{p}$, except the orientation error, which has an multiplicative error definition ${}^{\hat{\mathrm{n}}}\mathbf{q}_{\mathrm{b}} = {}^{\mathrm{n}}\mathbf{q}_{\mathrm{b}} \otimes {}^{\hat{\mathrm{n}}}\delta\mathbf{q}_{\mathrm{n}}$. The quaternion multiplication is denoted by $\otimes$ and ${}^{\hat{\mathrm{n}}}\delta\mathbf{q}_{\mathrm{n}}$ is the error quaternion corresponding to the angular error $\delta\boldsymbol{\phi}$.

3.2 Extending the EKF with readings from a polarization compass

The polarization compass determines the direction vector ${}^{\mathrm{p}}\bar{\mathbf{p}}_{\mathrm{s}} \in \mathbb{R}^3$ pointing to the sun expressed in the frame of the polarization camera (p-frame) and its corresponding covariance matrix $\boldsymbol{\Sigma}_s \in \mathbb{R}^{3\times3}$.
Using the convention to indicate the spherically normalized version of a vector $\mathbf{p}$ by $\bar{\mathbf{p}} = \frac{\mathbf{p}}{\|\mathbf{p}\|}$, the equation to transform the position of the sun in the navigation frame ${}^{\mathrm{n}}\mathbf{p}_{\mathrm{s}}$ to the camera frame ${}^{\mathrm{p}}\mathbf{p}_{\mathrm{s}}$ is given by (see [14])

$${}^{\mathrm{p}}\bar{\mathbf{p}}_{\mathrm{s}} = {}^{\mathrm{c}}\mathbf{R}_{\mathrm{b}}\,{}^{\mathrm{b}}\mathbf{R}_{\mathrm{n}}\,{}^{\mathrm{n}}\bar{\mathbf{p}}_{\mathrm{s}} \ . \quad (6)$$

The relation between the error of the expected measurement $\hat{\mathbf{h}}$ and the actual measurement $\mathbf{h}_{\mathrm{m}}$ as well as the error of the system state $\delta\mathbf{x}$ have to be defined in order to use them in the filter,

$$\begin{aligned}\delta\mathbf{h} &= \boldsymbol{\Pi}(\hat{\mathbf{h}} - \mathbf{h}_{\mathrm{m}}) \\ &= \boldsymbol{\Pi}({}^{\mathrm{p}}\mathbf{R}_{\mathrm{b}}\,{}^{\mathrm{b}}\mathbf{R}_{\hat{\mathrm{n}}}\,{}^{\mathrm{n}}\bar{\mathbf{p}}_{\mathrm{s}} - {}^{\mathrm{p}}\mathbf{R}_{\mathrm{b}}\,{}^{\mathrm{b}}\mathbf{R}_{\mathrm{n}}\,{}^{\mathrm{n}}\bar{\mathbf{p}}_{\mathrm{s}}) \\ &= \boldsymbol{\Pi}({}^{\mathrm{p}}\mathbf{R}_{\mathrm{b}}\,{}^{\mathrm{b}}\mathbf{R}_{\hat{\mathrm{n}}}\,{}^{\mathrm{n}}\bar{\mathbf{p}}_{\mathrm{s}} - {}^{\mathrm{p}}\mathbf{R}_{\mathrm{b}}\,{}^{\mathrm{b}}\mathbf{R}_{\mathrm{n}}(\mathbf{I}_{3\times3} + \lfloor\,\delta\boldsymbol{\phi}\,\rfloor_\times)\,{}^{\mathrm{n}}\bar{\mathbf{p}}_{\mathrm{s}}) \\ &= \boldsymbol{\Pi}\ {}^{\mathrm{p}}\mathbf{R}_{\mathrm{b}}\,{}^{\mathrm{b}}\mathbf{R}_{\hat{\mathrm{n}}}\lfloor\,{}^{\mathrm{n}}\bar{\mathbf{p}}_{\mathrm{s}}\,\rfloor_\times\delta\boldsymbol{\phi}\ .\end{aligned} \quad (7)$$

To solve Equation 7 the true rotation from the navigation frame to the body frame ${}^{\mathrm{b}}\mathbf{R}_{\mathrm{n}}$ is unknown and can be approximated by ${}^{\mathrm{b}}\mathbf{R}_{\mathrm{n}} = {}^{\mathrm{b}}\mathbf{R}_{\hat{\mathrm{n}}}(\mathbf{I}_{3\times3} + \lfloor\,\delta\boldsymbol{\phi}\,\rfloor_\times)$. The matrix $\boldsymbol{\Pi}$ is a projection matrix. It can be set to a constant value, e.g. $\boldsymbol{\Pi}_{\mathrm{s}} = \begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}$, which maps the angular error $\delta\phi$ to the x-y-plane of the polarization camera. If the deviation between the sun vector and the z-axis of the camera is sufficiently small, the performance will be satisfactory. Given the dynamics of the system and the fact, that the sun vector changes during the day, improvements can be expected by adapting $\boldsymbol{\Pi}_{\mathrm{d}}$ dynamically. By projecting the error between the predicted and measured sun vector onto the tangent space on the sphere at the predicted sun vector, the measurement error is invariant with respect to the estimated orientation [15]. The tangent space to the unit sphere is spanned by the column vectors of the matrix

$$\boldsymbol{\Pi}_{\mathrm{d}} = \mathcal{N}({}^{\mathrm{p}}\bar{\mathbf{p}}_{\mathrm{s}}{}^\top) = \begin{bmatrix}\mathbf{s}_{1\perp} & \mathbf{s}_{2\perp}\end{bmatrix}^\top, \quad (8)$$

where $\mathcal{N}({}^{\mathrm{p}}\bar{\mathbf{p}}_{\mathrm{s}}^\top)$ denotes the left null space of the vector ${}^{\mathrm{p}}\bar{\mathbf{p}}_{\mathrm{s}}$. The matrix $\boldsymbol{\Pi}_{\mathbf{d}}$ has to fulfill the property $\boldsymbol{\Pi}_{\mathrm{d}}\boldsymbol{\Pi}_{\mathrm{d}}^\top = \mathbf{I}_2$. One possible solution is given by

$$\begin{aligned}\mathbf{s}_{\mathbf{1}\perp} &= \frac{1}{\sqrt{\bar{p}_{\mathrm{s,x}}^2 + \bar{p}_{\mathrm{s,y}}^2}}\begin{bmatrix}-\bar{p}_{\mathrm{s,y}} & \bar{p}_{\mathrm{s,x}} & 0\end{bmatrix}^\top, \\ \mathbf{s}_{\mathbf{2}\perp} &= \frac{1}{\sqrt{\bar{p}_{\mathrm{s,x}}^2 + \bar{p}_{\mathrm{s,y}}^2}}\begin{bmatrix}-\bar{p}_{\mathrm{s,x}}\bar{p}_{\mathrm{s,z}} & -\bar{p}_{\mathrm{s,y}}\bar{p}_{\mathrm{s,z}} & \bar{p}_{\mathrm{s,x}}^2 + \bar{p}_{\mathrm{s,y}}^2\end{bmatrix}^\top.\end{aligned} \quad (9)$$

In the case of a static projection matrix, the covariance estimate $\boldsymbol{\Sigma}_{\mathrm{s}}$ can be projected to the subspace by the equation

$$\boldsymbol{\Sigma}_{\mathrm{s,r}} = \boldsymbol{\Pi}_{\mathrm{s}}\boldsymbol{\Sigma}_{\mathrm{s}}\boldsymbol{\Pi}_{\mathrm{s}}^\top. \quad (10)$$

In the case of a dynamic projection matrix, the static projection matrix $\boldsymbol{\Pi}_{\mathrm{s}}$ has to be replaced with the matrix $\boldsymbol{\Pi}_{\mathrm{d}}$ defined in Equation 8 and Equation 9. The reduced covariance matrix $\boldsymbol{\Sigma}_{\mathrm{s,r}} \in \mathbb{R}^{2\times2}$ is non-singular and can be used in the filter update equations.

4 EXPERIMENTS

Several experiments were carried out to test the different components of the system under varying conditions. The set of indoor experiments was done in a lab where high frequency ground truth data was available, but readings of the polarization sensor had to be simulated. For the set of outdoor experiments ground truth data was only available occasionally but real readings from the polarization sensor could be used.

The set of indoor experiments consists of a trajectory of ARDEA, which was augmented with simulated readings of the polarization compass to evaluate the influence of the polarization compass. The set of outdoor experiments consists of one experiment to evaluate the performance of the polarization compass itself and a second experiment to evaluate the performance of the overall system.

4.1 Test of the polarization compass

As an initial test, we placed the multicopter on a leveled turntable and recorded the estimated sun azimuth and elevation angles while turning the multicopter in steps of 30°. As illustrated in Figure 5, the sun position can be determined quite accurately with a standard deviation below 1° for azimuth and below 3° for elevation angle.

4.2 Indoor test of pose estimation with simulated measurements from the polarization compass

In the second experiment a trajectory of an indoor experiment in the lab was augmented with simulated measurements of the polarization compass. The measurements of the polarization compass were artificially corrupted by zero mean, white Gaussian noise. The noise levels were empirically determined. For the indoor datasets at each time stamp the ground truth pose of ARDEA is available with high precision.

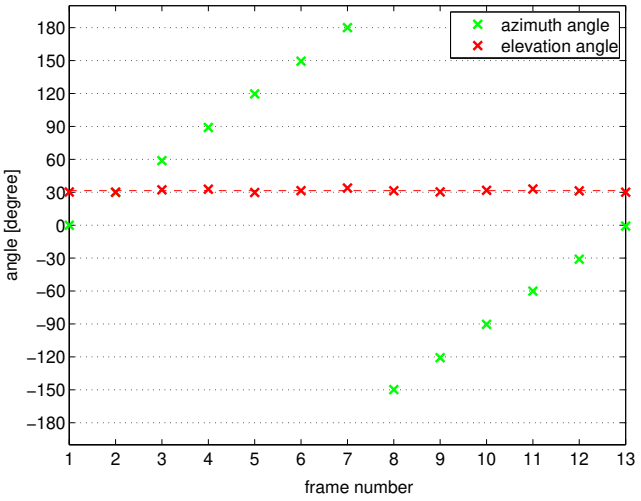


Figure 5: Test of the sun position estimation by turning the multicopter in 30° steps. Shown are the azimuth angle with respect to the initial orientation (green 'x') and the sun elevation angles (red 'x') as estimated by the polarization compass. The dashed line shows the true sun elevation angle ($\approx 32°$).

Therefore, Euler angle errors can be calculated. They are depicted in Figure 6. Roll and pitch errors stay limited for all three cases, while the yaw angle error grows unbounded with time if the polarisation compass is not used. Due to the use of the polarization compass its steady increase can be compensated.

4.3 Outdoor test of pose estimation with real measurements from the polarization compass

In Figure 7 the estimated position of ARDEA during an outdoor experiment is given. The start and final positions are at the origin. The polarization compass improves the estimates in the case of the dynamic projection matrix Π_d and also in the case of the static projection matrix Π_s. Slight differences between the static and dynamic projection approach can be seen for the z-direction, where the dynamic projection results in a lower error.

An often used error metrics for translational errors is the norm of the distance of the estimated final position to the true final position with respect to trajectory length. Given the length of the trajectory of approx. 132 m, the relative errors are 2.7%, 0.6% and 0.5% for the approach without the polarization compass, with the polarization compass and a static projection matrix and with the polarization compass and dynamic projection matrices.

Roll and pitch angles are globally observable when fusing accelerometer and gyroscope readings with the delta poses of a VO. Therefore, the polarization compass only slightly improves their estimation. But the slight improvement of roll and pitch estimation results in a lower vertical position error. Without the polarization compass, the yaw angle error grows

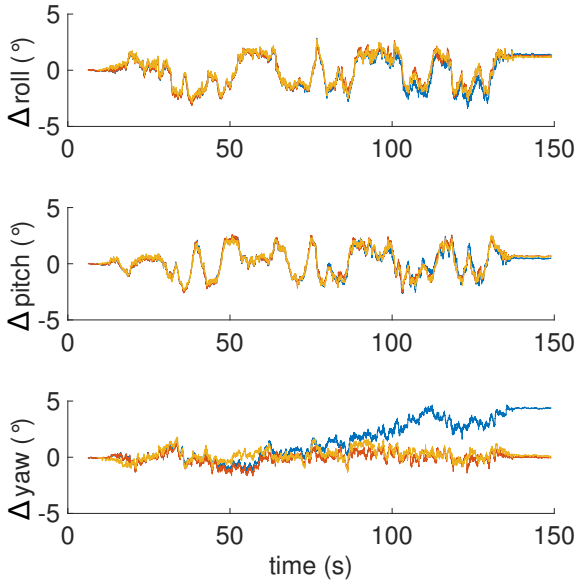


Figure 6: Roll, pitch and yaw error for a single run with simulated sun vector measurements. Blue: without polarization compass, red: with polarization compass using static projection, yellow: with polarization compass using dynamic projection.

unbounded with time. Due to the polarization compass this drift can be compensated, which results in improvements of x and y position estimates.

Multiple runs with different trajectories resulted in similar system behavior and similar values of the error metrics.

5 CONCLUSIONS

It was shown in the experiments that fusing the polarization compass with the data from an inertial measurement unit and a VO in an indirect EKF improves the accuracy of pose estimation. The differences are small between the approach with a static projection matrix and a dynamic projection matrix. While a polarization compass can obviously provide orientation estimations only outdoors, it is likely to improve state estimation also in mixed indoor/outdoor flights. Furthermore, using a polarization compass in conjunction with a VINS could also be beneficial in other applications, e.g. when matching maps from multiple robots.

References

[1] R. Jin, H. Sun, J. Sun, W. Chen, and J. Chu. Integrated navigation system for UAVs based on the sensor of polarization. In *ICMA*, pages 2466–2471, 2016.

[2] W. Zhi, J. Chu, J. Li, and Y. Wang. A novel attitude determination system aided by polarization sensor. *Sensors*, 18(1):158, 2018.

[3] G. J. Taylor, W. Ribi, M. Bech, A. J. Bodey, C. Rau, A. Steuwer, E. J. Warrant, and E. Baird. The dual func-

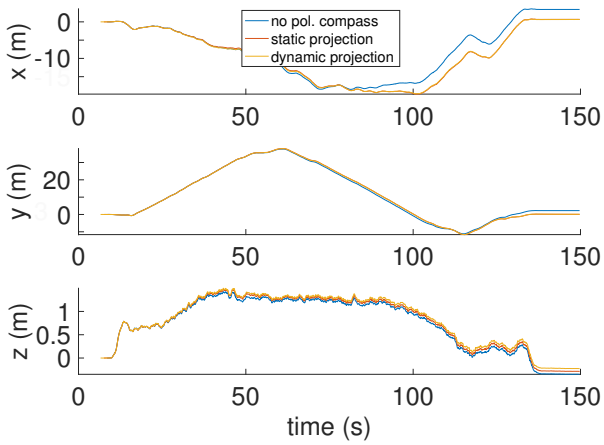


Figure 7: Estimated position of ARDEA during an outdoor experiment with and without the polarization compass.

tion of orchid bee ocelli as revealed by x-ray microtomography. *Current Biology*, 26(10):1319–1324, 2016.

[4] W. Stürzl. A lightweight single-camera polarization compass with covariance estimation. In *ICCV*, pages 5363–5371, 2017.

[5] G. Horváth and D. Varjú. *Polarized Light in Animal Vision: Polarization Patterns in Nature*. Springer, 2004.

[6] R. Wehner and M. Mueller. The significance of direct sunlight and polarized skylight in the ant's celestial system of navigation. *Proceedings of the National Academy of Sciences of the United States of America*, 103:12575–12579, 2006.

[7] M. G. Müller, F. Steidle, M. Schuster, P. Lutz, M. Maier, S. Stoneman, T. Tomic, and W. Stürzl. Robust visual-inertial state estimation with multiple odometries and efficient mapping on an MAV with ultra-wide FOV stereo vision. In *IROS*, 2018.

[8] D. Lambrinos, R. Möller, T. Labhart, R. Pfeifer, and R. Wehner. A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems*, 30:39–64, 2000.

[9] J. Chahl and A. Mizutani. Biomimetic attitude and orientation sensors. *IEEE Sensors Journal*, 12:289–297, 2012.

[10] K. H. Strobl, W. Sepp, S. Fuchs, C. Paredes, M. Smisek, and K. Arbter. DLR CalDe and CalLab, `www.robotic.dlr.de/callab/`. Institute of Robotics and Mechatronics, German Aerospace Center (DLR).

[11] W. J. MacLean. Removal of translation bias when using subspace methods. In *ICCV*, pages 753–758, 1999.

[12] K. Schmid, F. Ruess, M. Suppa, and D. Burschka. State estimation for highly dynamic flying systems using key frame odometry with varying time delays. In *IROS*, pages 2997–3004, 2012.

[13] K. Schmid, F. Ruess, and D. Burschka. Local reference filter for life-long vision aided inertial navigation. In *Fusion*, 2014.

[14] N. Trawny and S. Roumeliotis. Sun sensor model. *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, 1, 2005.

[15] W. Förstner. Minimal representations for uncertainty and estimation in projective spaces. In *ACCV*, pages 619–632, 2010.

A CNN-based Drone Localisation Approach for Autonomous Drone Racing

Jos Arturo Cocoma-Ortega *[1] and J. Martinez-Carranza[1,2]

[1]Instituto Nacional de Astrofisica, Optica y Electronica , Puebla, Mexico
[2]University of Bristol, Bristol, UK

ABSTRACT

In this paper we present a CNN architecture to automatically estimate the position of a drone, in metres, relative to a gate in a race track. The latter arises in the context of the autonomous drone racing competition where the challenge is to design a drone that can beat a human in a drone race. There have emerged different proposals to address this problem. Notably, localisation of the drone in the race track is one of the first capabilities that could lead to a solution. However, global localisation may require sophisticated methods such as odometry or SLAM that may become expensive to be computed on board. Furthermore, global localisation may drift as the drone runs the track. Motivated by the latter, we present a CNN architecture based on the Posenet network, which was designed for camera relocalisation in real time. Nevertheless, we have adopted, modified and re-trained such network to the context of relative localisation w.r.t to a gate in the track, which can be exploited by the autonomous navigation algorithms for the race. We report an average performance of 50 fps and a maximum up to 100 fps in a low budget computer with a modest GPU, thus outperforming similar works in the state of the art.

1 INTRODUCTION

Autonomous Drone Racing (ADR) is an open challenge that focuses on having to beat a human in a drone race. This task leads to various challenges, such as localisation and drone control navigation. To know where the drone is, represent a fundamental task in the planning for autonomous navigation, in the last decade several works were focused on estimating the pose of a robot by means of using a single camera and a techniques such as visual odometry or visual simultaneous localisation and mapping, with good accuracy in the estimation, but with the caveat that such estimates may be obtained at low frame rates (20 - 30 Hz). Pose estimation at high frequency is desirable as it could be exploited in agile flights, such as those expected in a drone race. Even proposals that have been employed in ADR competitions operates at 10 fps.

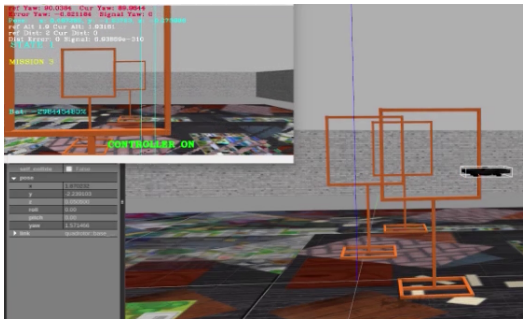


Figure 1: We design a method for Autonomous Drone Racing based on CNN for pose estimation and an algorithm for autonomous navigation. See `https://youtu.be/5rboqinFXYo`

Motivated by the above, in this work we proposed an algorithm for Autonomous Drone Racing based on Convolutional Neural Networks aiming at estimating the pose of the drone relative to the gate and at a high frequency. Similar works have achieved this but at a frame rate of 10 fps. In contrast, our proposal achieves an estimation speed of 100 fps on average with GPU and 20 fps on average with CPU.

To describe our approach, first we will discuss the related work in section 2, then we will describe the methodology used to design and train the network and how we use the pose estimation for autonomous navigation in section 3. Next, we will present the testing results showing that we can estimate the pose up to 100 fps, in section 4. Finally, conclusions are discussed in section 5.

2 RELATED WORK

In recent years, the problem of estimating the position of the camera has been widely studied. There are two main approaches to Visual Odometry: geometrical approach and deep learning approach.
Visual SLAM is one of the most used algorithms to known the robot (camera) position in navigation. V-SLAM solves the problem of localisation and mapping the environment by

*Department of Computer Science at INAOE. Email addresses: {cocoma, carranza}@inaoep.mx

landmarks and features from the frame observed [1]. Deep learning-based algorithms have explore different ways to estimate camera pose. We can found in literature works that uses CNN as main algorithm and shows the viability of the results instead geometric ones [2, 3], other ones resolves localisation via V-SLAM in where estimates VO and also generates a map of the environment [4, 5]. One relevant work is the reported in [6, 7] where they propose a Network they called Posenet. Posenet is based in GoogLeNet [8]. The main contribution of the work is the change of the softmax classifier in the last three layers by a regressor to estimate the pose of the camera. They report high accuracy in their results. Also, it is reported a real-time computation for pose estimation, a time of $5ms$.
There are some works, focus in the estimation of the pose of an object in the image, this is the scope of the works published in [9, 10, 11, 12].
Seminal works addressed the problem to autonomous navigation by using visual odometry or visual SLAM algorithms to resolve the drone's localisation [13] and then generate a planning based on the pose of the drone. In this same context, the works [14, 15] describe an algorithm to autonomous navigation by detecting the gate objective and develop a planning route for the drone flying. Using traditional computer vision, the works presented in [16, 17] propose a strategy based on colour pixels of the gate for detection (four corners) and subsequently, the problem of perspective n-point (PnP) is solved to estimate the relative position of the drone. Other approach based on gate detection by the use of deep learning is presented in [18], they propose a modified SSD network they called ADR-Net to gate detection and then they propose a guidance algorithm based on LOS vector guidance to performs autonomous flight to cross the gate.

3 Methodology

For autonomous navigation in drone racing, the principals approach have shown an efficient way to planning navigation knowing the position of the gate.

In this work we propose an approach to pose estimation based on gate position, this means, not to estimate the pose of camera based on the whole scene, instead take the gate as reference an estimate how far is the camera from the gate.

We propose a CNN solution based on Posenet [6]. Posenet allows to estimate 6D camera pose for a complete scene outdoors and indoors. We are only interested in 3D camera pose, it means only translation is required for this work, thus we modify the regressors layers to outputs only position (x, y, z) and set the euclidean distance only for translation for the learning algorithm. Also is eliminated a image normalization (mean subtraction) due the use in continuous video images (real-time). For this propose, this modifications are made in a complete network and also is designed a reduced one for increasing network rate predict (see figure 2).

The dataset was designed in simulated environment using gazebo. The scene is created with two gates only, the reason

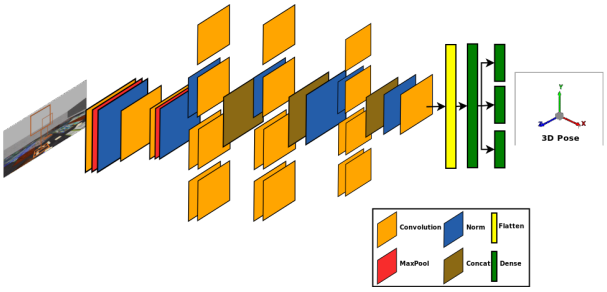


Figure 2: Reduced Posenet architecture.

for this is because when the drone is far enough of gate one, it can see the both gates. Then the drone flies towards the gate and when it is close to the gate one the camera will be in a blind point from that gate, this means that the drone will no be able to see the gate one. Thus, gate two will appear in the line vision producing a new estimation of the pose related to gate two. It is for that reason that the dataset is design in that way, figure 3 shows an example of the gates in the Gazebo scene.

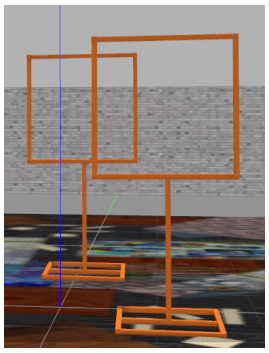


Figure 3: Example of gates used for training.

Using the gates designed as shown, the pose of the drone is calculated used gazebo model state, but not from scene origin, the pose is related to the gate one. Thus the groundtruth is created related the distance of the drone from the gate one, the figure 4 illustrates how is the pose of the drone taken in the simulator.

The pose estimation calculated by the CNN, is used to develop autonomous navigation. The algorithm developed calculates the trajectory adjustment necessary to fly through the gate. In the first step, the drone aligns its position to the center of the gate (y position), and then when it is centered the algorithm commands to fly the distance necessary to close the gate. As we describe early, when the drone is in the blind point of the gate, then predict the position to the next gate, with this new position, it is estimate the distance left to cross the gate. When the gate has been through, the algorithm restarts the process to fly and cross the next gate. The methodology described is illustrated in the figure 5.

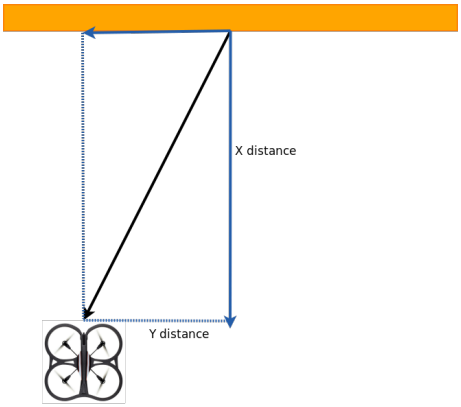


Figure 4: Pose of the drone related to the center of the gate, top view.

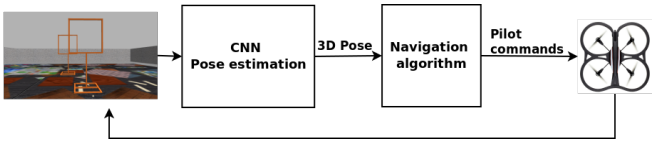


Figure 5: Proposed methodology.

4 Experiments and Results

The experiments were conducted by the use of simulated environment using gazebo. This section describe the results obtained in each experiment.

4.1 Pose evaluation

To evaluate the pose predicted. ROS framework was used to communicate simulated environment (Gazebo) with the Predictor (Modified Posenet) and RVIZ. The experiments performed showed that exist a precision zone for the prediction due to the design of the training dataset. Inside the precision area (this area has size 2.2m x 2.5m), the mean error decreases and prediction is close to the groundtruth, the figure 6 shows the evaluation of the pose predicted displayed in RVIZ, also is attached to figure a white rectangle indicating the precision area detected.

The error calculated inside and outside the area indicates that when the gate is the line vision of the drone the error decreases (inside area), but the error increases as the drone flies away leaving out of the line of vision to the window. The poses were compare calculating the distance between them (error). Figure 7 plots the errors in the navigation test. As the drone flies insider of the precision area, the error decreases to a mean of 0.16 m. Even if the drone is inside the area, the orientation also affects the pose estimation, the more oriented to the front of the gate, implies the less error in prediction.

To evaluate the performance of the Reduced Posenet, navigation tests are carried out in the same way as the Modified Posenet. It can observes from figure 9 that the pose predicted is close to groundtruth inside the precision area in the same

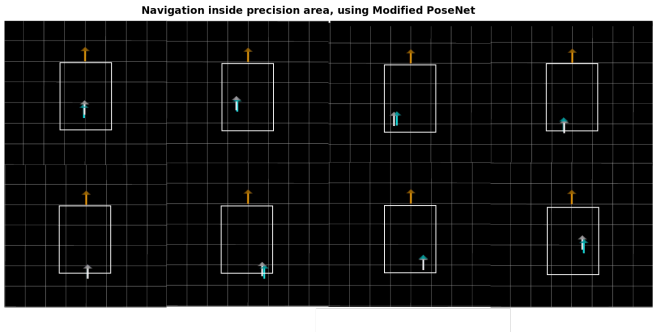


Figure 6: Predicted pose compared with groundtruth using RVIZ. White arrow shows Groundtruth and Blue arrow Predicted.

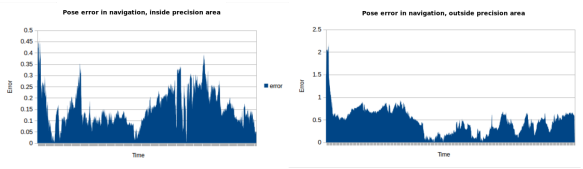


Figure 7: Error over time in navigation. Left graph shows the error while navigating inside the precision area. Right graph shows the error while navigating outside the precision area.

way that Modified Posenet, besides the error increases more outside the precision area. This is not really significant, because when the drone is flying towards the center of the window automatically will be placed inside the area and the prediction will be best for the navigation algorithm.

We have found a similar behavior for the error in the inside and outside area when we plot the error (position differences between predicted and groundtruth) over the time in a navigation. This is showed by figure 8.

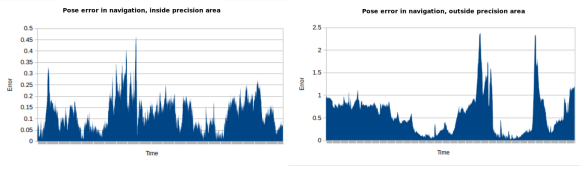


Figure 8: Error over time in navigation. Left graph shows the error while navigating inside the precision area. Right graph shows the error while navigating outside the precision area.

We have test the algorithm for autonomous drone racing. Using the scenario from Gazebo, via ROS framework, we communicate the pose prediction with the algorithm for navigation to the Gazebo world. In the world presented in figure 10, we put three gates in the line vision of the drone to evaluate at first the prediction in the autonomous navigation. The algorithm correctly estimate the pose from the gate and command the drone to center the gate to fly across. As the se-

quence shows, the drone flies satisfactorily through the gate and then stop and oriented to the next gate.

In the table 1, we summarise the error results of both approaches as well as the frequency of process of the pose prediction. The best performance is for the Reduced Posenet, that has minimum error inside the precision area and has the highest frame rate operation for prediction.

	Inside ε	Outside ε	Frame rate (GPU)
MPoseNet	0.1597 m	0.4866 m	50 fps
RPoseNet	0.1285 m	0.5867 m	100 fps

Table 1: Results in navigation testing Modified PoseNet (MPoseNet) and Reduced PoseNet (RPoseNet) for both inside and outside precision area of prediction. ε is the mean error of the predictions over the time of navigation.

All the test of the algorithm were conducted in a computer with a GTX 860m, 16Gb of RAM and an i7-4710HQ CPU.

5 CONCLUSIONS

Autonomous Drone Racing represents a big challenge to develop efficient algorithms that can beat a human pilot in navigation. Localisation at high-speed is still one of the principal problems to solve.

In this work, we have shown that it is possible to estimates 3D pose of a drone relative to a gate in real-time, and at high frame rate.

To achieve this, we have developed a dataset that allows the proposed CNN to learn the pose of the drone with respect to the gate with a low error. In addition, we have designed an algorithm for Autonomous Drone Racing based on the pose obtained from the CNN.

The tests performed in simulation shows goods results with low error.

We report the highest rate for pose prediction at 100 fps (20 fps with CPU) with our reduced Posenet for and with a low error of around 13 centimetres, which still enables our navigation algorithm to centre the drone w.r.t the gate to then command it to cross the gate.

As future work, we will improve the test for real-world scenarios to evaluate the pose estimation and the autonomous drone navigation.

REFERENCES

[1] H. Casarrubias-Vargas, A. Petrilli-Barcelo, and E. Bayro-Corrochano. Ekf-slam and machine learning techniques for visual robot navigation. In *2010 20th International Conference on Pattern Recognition*, pages 396–399, Aug 2010.

[2] Nolang Fanani, Alina Strck, Matthias Ochs, Henry Bradler, and Rudolf Mester. Predictive monocular odometry (pmo): What is possible without ransac and multiframe bundle adjustment? *Image and Vision Computing*, 68:3 – 13, 2017. Automotive Vision: Challenges, Trends, Technologies and Systems for Vision-Based Intelligent Vehicles.

[3] Alec Graves, Steffen Lim, Thomas Fagan, et al. Visual odometry using convolutional neural networks. *The Kennesaw Journal of Undergraduate Research*, 5(3):5, 2017.

[4] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Toward geometric deep slam. *CoRR*, abs/1707.07410, 2017.

[5] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6565–6574, July 2017.

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

[7] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4762–4769, May 2016.

[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[9] Patrick Poirson, Phil Ammirato, Cheng-Yang Fu, Wei Liu, Jana Kosecka, and Alexander C Berg. Fast single shot detection and pose estimation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 676–684. IEEE, 2016.

[10] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[11] EV Shalnov and AS Konushin. Convolutional neural network for camera pose estimation from object detections. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42, 2017.

[12] Thanh-Toan Do, Ming Cai, Trung Pham, and Ian Reid. Deep-6dpose: Recovering 6d object pose from a single rgb image, 2018.

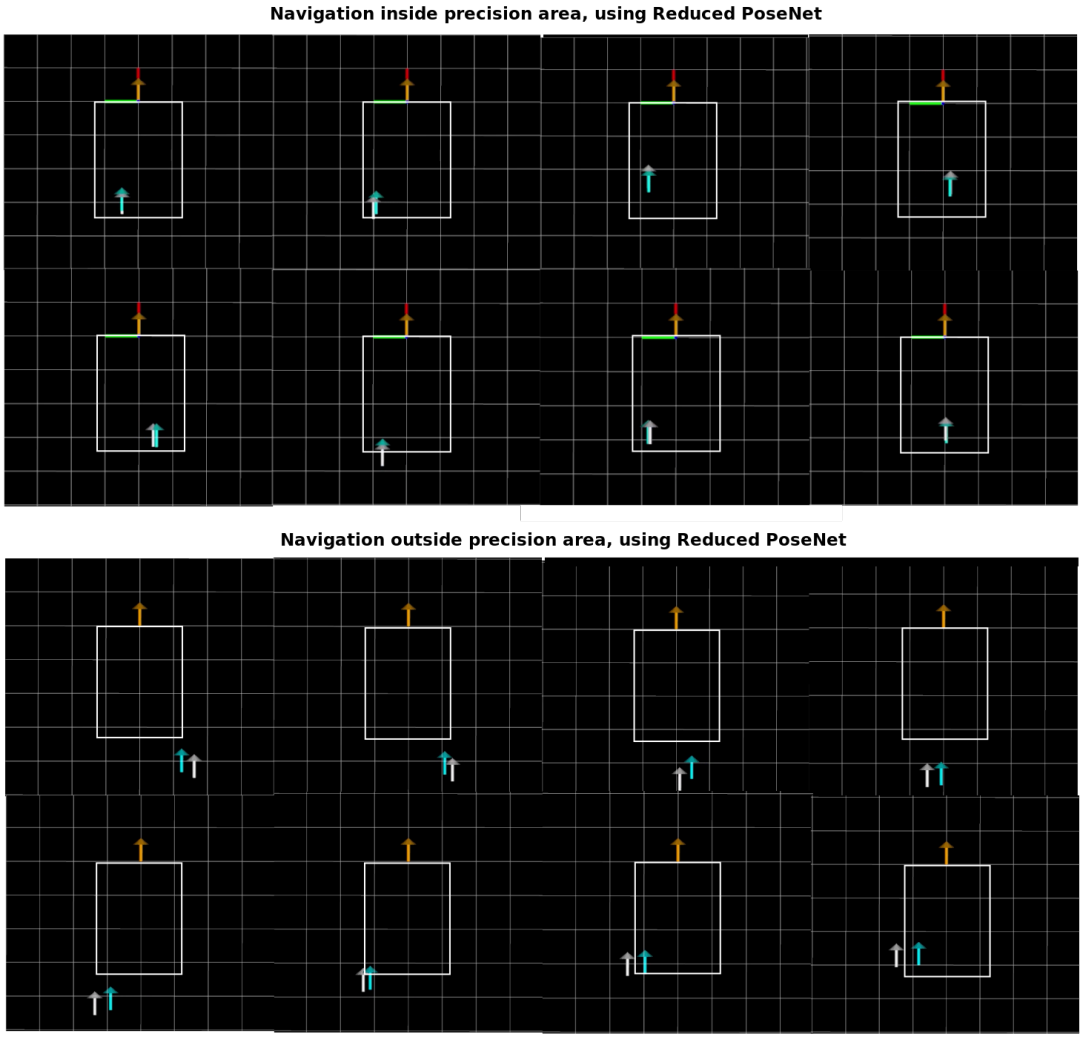


Figure 9: Predicted pose compared with groundtruth using RVIZ. First row shows Groundtruth (white) and Predicted (Sky) comparison results by Reduced PoseNet inside the precision area. Second row shows comparison results by Reduced PoseNet outside precision area.

[13] 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), Apr 2019.

[14] 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. *CoRR*, abs/1810.06224, 2018.

[15] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: Learning agile flight in dynamic environments. *CoRR*, abs/1806.08548, 2018.

[16] Shuo Li, Michaël MOI Ozo, Christophe De Wagter, and Guido CHE de Croon. Autonomous drone race: A computationally efficient vision-based navigation and control strategy. *arXiv preprint arXiv:1809.05958*, 2018.

[17] Shuo Li, Erik van der Horst, Philipp Duernay, Christophe De Wagter, and Guido CHE de Croon. Visual model-predictive localization for computationally efficient autonomous racing of a 72-gram drone. *arXiv preprint arXiv:1905.10110*, 2019.

[18] S. Jung, S. Hwang, H. Shin, and D. H. Shim. Perception, guidance, and navigation for indoor autonomous

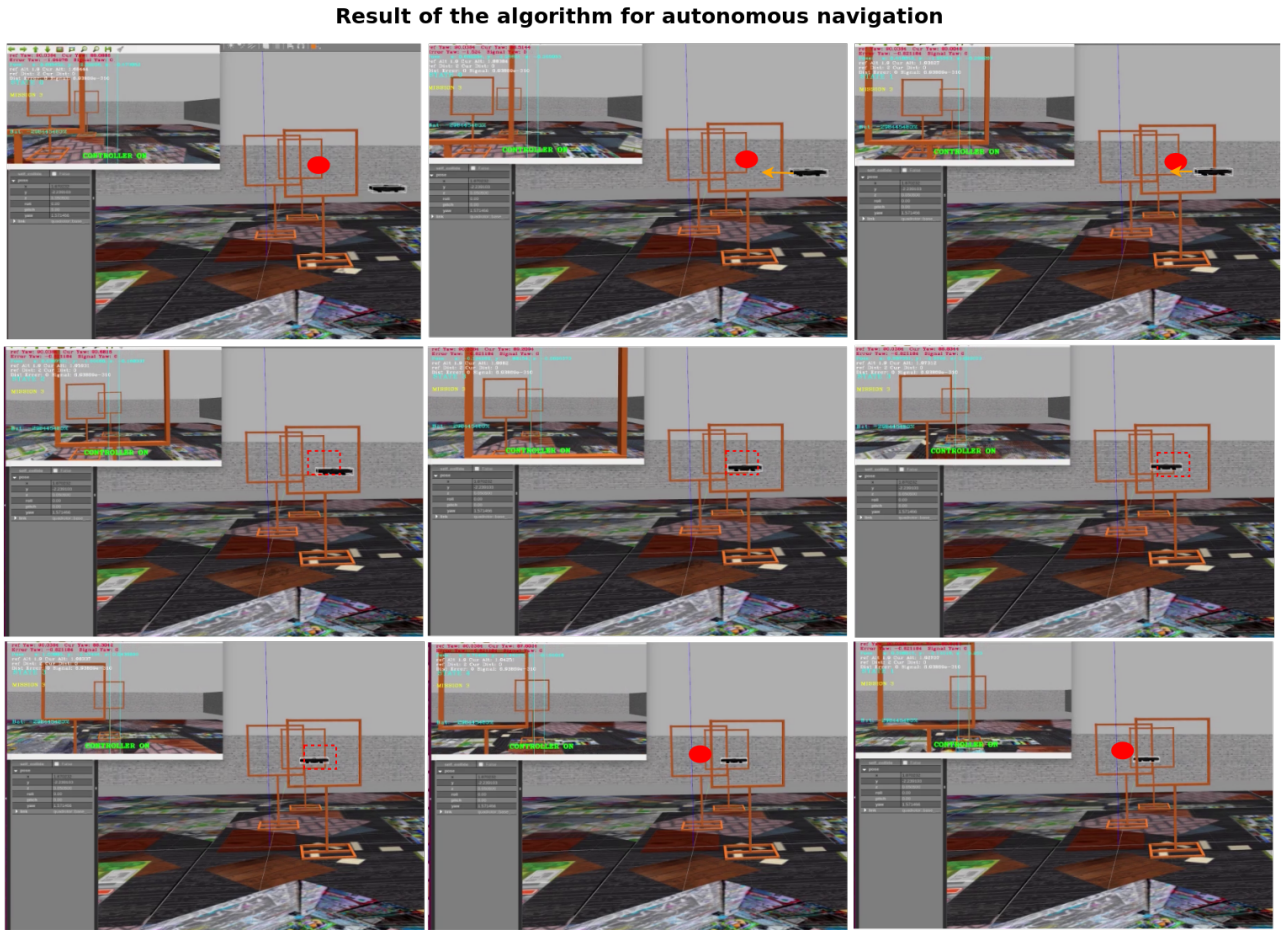


Figure 10: Autonomous navigation algorithm flying through one gate. In the first row, the algorithm centre the drone flying to the left, once the drone is centred, the drone flies to cross the gate (row two), this is marked with a dotted rectangle. Finally in row three when the drone complete cross the gate, the algorithms predict the pose of the drone and centres it again.

drone racing using deep learning. *IEEE Robotics and Automation Letters*, 3(3):2539–2544, July 2018.

Slipstream Deformation of a Propeller-Wing Combination Applied for Convertible UAVs in Hover Condition

Y. Leng*, M. Bronz, T. Jardin and J-M. Moschetta
ISAE-SUPAERO, Université de Toulouse, France
ENAC, Université de Toulouse, France

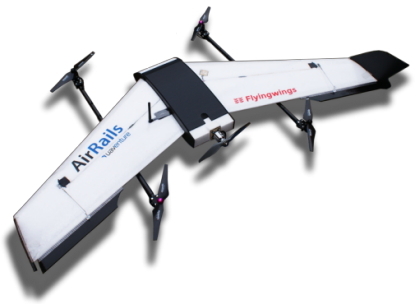
ABSTRACT

Convertible unmanned aerial vehicle (UAV) promises a good balance between convenient autonomous launch/recovery and efficient long range cruise performance. Successful design of this new type of aircraft relies heavily on good understanding of powered lift generated through propeller-wing interactions, where the velocity distribution within propeller slipstream is critical to estimate aerodynamic forces during hover condition. Current study analysed a propeller-wing combination with a plain flap. A 5-hole probe measurement system was built to construct 3 dimensional velocity field at a survey plane after trailing edge. The study has found that significant deformation of propeller slipstream was present in the form of opposite transverse displacement on extrados and intrados. The deformation could be enhanced by flap deflections. Velocity differences caused by the slipstream deformation could imply local variation of lift distribution compared to predictions from conventional assumptions of cylindrical slipstream. The research underlined that the mutual aspect of propeller-wing interaction could be critical for low-speed aerodynamic design.

1 INTRODUCTION

Small-scale unmanned aerial vehicle has recently attracted great amount of interests due to their autonomous capability to conduct highly repetitive or dangerous flight missions. This capability is realised through electrical propulsion system and improved autoflight system. The current UAV lifting systems are generally derived by down-scaling manned aircraft. The clear division of rotorcraft and fixed-wing aircraft can still be seen in most professional UAV applications.

It has been seen however that a hybrid design that combines the vertical take-off / landing capability and the efficiency of fixed-wing aircraft could improve mission performance of current UAV applications and eventually open up new type of missions. Rotor lifting system is inefficient for long-endurance flight, and thus mission range is limited. On the other hand, most current fixed-wing UAVs rely on crew and sometimes specific systems for launch and recovery, which limits the origin and destination to dedicated points where the aircraft can be accommodated by ground crew. To perform a fully autonomous long-range mission, a hybrid design called convertible drone is needed.



(a) Combination of quad-copter and flying wing [1]

(b) Darko developped by ENAC drone research group

Figure 1: Examples of convertible drone configurations

The key to an optimised design of convertible drone lies in the interaction between propulsion system and the lifting surfaces. An entirely independent design, such as shown in Figure 1a requires lifting propellers that aren't used in cruise flight, hence additional weight and drag are introduced. A

*Email address(es): yuchen.leng@isae-supaero.fr

fully hybrid approach (Figure 1b) takes advantage of arranging lifting surfaces within propeller slipstream for augmented lift from blown wing. In this way the propeller and wing are both used during hover and cruise flight, and their sizes must match to deliver the required aerodynamic performance while minimizing the weight of combined system.

Unlike an independent design, the hover lift is distributed between the vertical component of propeller thrust and wing lift augmented by rotor slipstream. Thus flow interference between the wing and slipstream must be well understood to ensure sufficient lift in hover.

To further augment wing lift and to provide flight control, trailing edge flap is typically installed, such as shown in Figure 2. Propeller slipstream can therefore be deflected at a certain angle to generate additional aerodynamic force and moment. Sufficient pitch and roll control authority can be achieved with appropriate flap design.

Figure 2: Convertible UAV Cyclone hovering with negative flap deflection

During preliminary design, reduced-order models such as panel method, vortex lattice method, to name a few, are preferred due to their capability of analysing large amount of candidate configurations at a relatively small computational cost [2]. Veldhuis et al. has identified two approaches in analysing propeller-wing systems : single approach and dual-coupling approach.

In single analysis mode, only the influence of propeller slipstream is taken into consideration. When calculating wing lift for sections immersed in propeller slipstream, the accelerated freestream velocity and sometimes the circumferential swirl velocity are applied to calculate local angle of attack and dynamic pressure. The velocities in the slipstream are computed from a free propeller model, such as one based on blade element momentum theory.

A dual-coupling mode is sometimes used to improve accuracy. The same calculation on wing sections still applies. A main difference is that the freestream condition of the propeller is also modified after the wing circulation distribution is solved, and induced velocity from the lifting surfaces is added to flight speed for propeller calculation. Ideally, an iterative approach is used until both solutions converge.

Both analysis modes require an empirical coefficient to attenuate propeller induced velocity before application in wing calculation [3, 4]. This suggests propeller induced velocity distribution might have changed due to the presence of wing. The effect was treated semi-empirically in [3], but a clear physical understanding is still absent.

Recent studies on tractor propeller wake measurements have found that the influence of wing to the propeller isn't limited to the flow upstream of the rotor disk. Deters et al [5] have used a seven-hole probe to make wake survey at different downstream locations after three different propellers. A flat plate wing is situated close to the propeller. The presence of wing is significant that the upper and lower halves of the slipstream translated in opposite direction by a distance up to 1 propeller radius at survey plane. The phenomenon was first observed and analysed by Witkowski et.al [6]. However neither studies provided quantitative analysis.

In this paper, a wake survey in static condition is presented at different rotation speeds and flap deflection angles. The test equipment and condition will be introduced in Section 2. Results and quantitative analysis will be shown in Section 3.

The test was also performed with flap deflection to investigate the slipstream development when the wing was generating lift.

2 Test set-up

2.1 *Test equipments*

The test was conducted in the indoor flight arena at Ecole National de l'Aviation Civile (ENAC). The flight arena's volume provides static ambient environment for simulating hover condition.

The test equipments were divided into three subsystems : 1) propeller-wing combination and their relevant motion control system ; 2) 5-hole probe and its data acquisition system ; 3) motion control system for 5-hole probe. The test setup in shown in Figure 3.

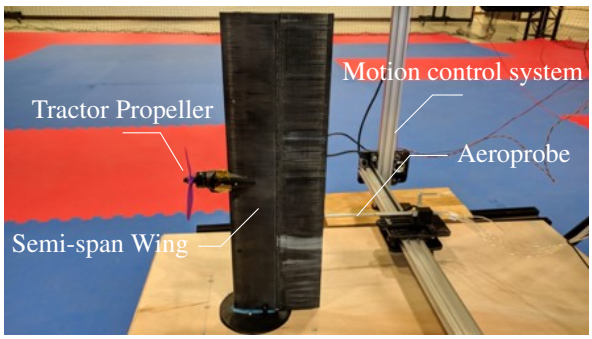


Figure 3: Test set-up in ENAC indoor flight arena

2.1.1 Propeller-wing model

The wing tested was a semi-span model with $500mm$ span. The straight wing had a constant chord length of $150mm$ and NACA0012 aerofoil section. A propeller nacelle was situated at $55mm$ from plane of symmetry, where a CM2206 direct current brushless motor was enclosed. A full-span plain flap was installed for the last 50% chord, and a servo allowed symmetrical flap deflection of $15°$ in either direction.

An APC 3-blade 5x4.6E propeller was tested. A tilt-rotor mechanism was designed to allow propeller install angle to change between $-10°$ to $10°$ with respect to wing chord line. The tilt mechanism was fixed at $0°$ for this experiment.

2.1.2 5-hole probe

The wake survey was conducted with an Aeroprobe 5-hole probe. The centre of probe head was located at $15mm$ behind trailing edge or 1.7 times propeller diameters downstream of rotor plane.

At the centre sphere, five holes were arranged in a cross pattern with one in the centre, a pair in vertical plane and another pair perpendicularly arranged. A series of static ports were situated after the probe head. When air is blown, the velocity, pitch and yaw attitude of probe will produce pressure difference between centre hole and static ports, vertical pair and side pair holes.

Honeywell analogue differential pressure sensors were used to measure the three pairs of pressure differences which were needed to resolve flow velocity. A calibration method proposed by Reichert et al [7] were used to take into consideration of cross-product terms and to correct alignment errors.

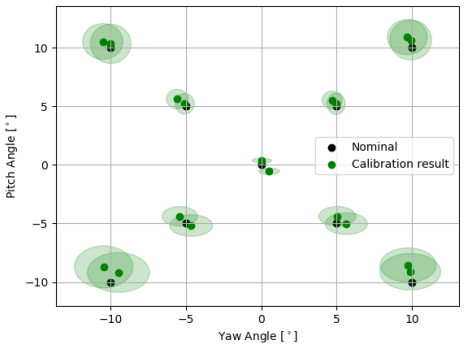


Figure 4: Flow angle measurement

The calibration were also analysed for measurement error. An uncertainty analysis were performed similar to the one described by Reichert et al, and fitting error as well as pressure fluctuations were considered in uncertainty propagation. A validation test were performed in the wind tunnel with known wind velocity and probe attitude. Flow angle measurement and its uncertainty is plotted in figure 4; flow speed measurement and its uncertainty is plotted in figure 5.

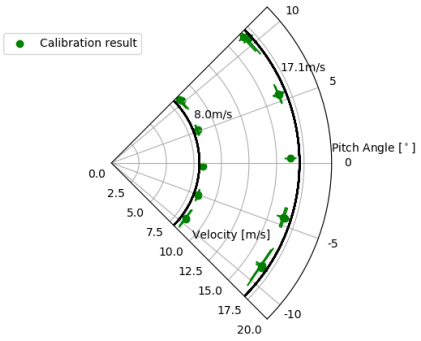


Figure 5: Flow speed measurement

From the validation case, uncertainty in flow speed was estimated at $\pm 0.3m/s$ and error in flow angle was estimated to be less than $2°$ below $20°$.

2.1.3 Motion control system

A 2-axis linear motion frame was constructed to allow automatic wake survey at a given plane perpendicular to propeller axis. Three stepper motors controlled by I2C bus were used to move a cart on which the 5-hole probe was mounted within the survey plane. The measurement was made on a 15×15 grid using alternating survey pattern as depicted in figure 6. Mean velocity data was obtained from sample recorded at $700Hz$ over a period of $5s$.

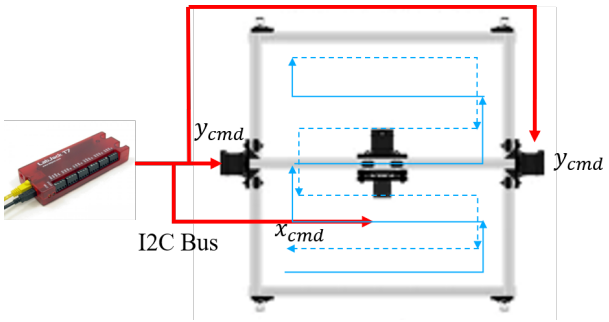


Figure 6: Motion control system and survey pattern

2.2 *Test conditions*

All tests were conducted at $V_\infty = 0$ to analyse flow condition at hover flight. Different propeller rotation speed and flap angle were tested, and the test matrix is given in Table 1

The rotation of propeller in front of a finite wing made the situation no longer symmetrical. Since lift must vanish at wing tip, spanwise lift distribution isn't uniform for a finite wing without propeller. Furthermore, an up-going propeller blade influences the wing section behind in a different

Test Variables	
Rotation Speed $[rpm]$	5770 / 8000 / 10000
Flap Deflection $[°]$	$0, \pm 15$

Table 1: Test Parameters

way from the down-going blade, hence the influence of a single rotating propeller isn't symmetrical. For this reason, both positive and negative flap deflections were tested.

3 RESULTS

In this section the results of $0°$ flap deflection will first be presented in subsection 3.1, where the effect of rotation speed as well as the general flow structure of propeller-wing interference will be discussed. Further discussion will continue in subsection 3.2 on the effect of flap deflection.

3.1 $0°$ Flap Deflection

The configuration at neutral flap setting excluded the effect of different velocity and pressure profiles on the extrados and intrados. The wake survey therefore was only influenced by the fact that propeller slipstream was separated by a solid surface.

The wake survey at $8000rpm$ is presented in Figure 7. The velocity field distribution in the survey plane is depicted as two components : the streamwise component u is perpendicular to the survey plane and the transverse component $V_t = \sqrt{v^2 + w^2}$ is situated within the survey plane. In Figure 7, the background contour shows u distribution while the transverse V_t is superposed by arrow symbols that give both magnitude and direction of V_t at sample points.

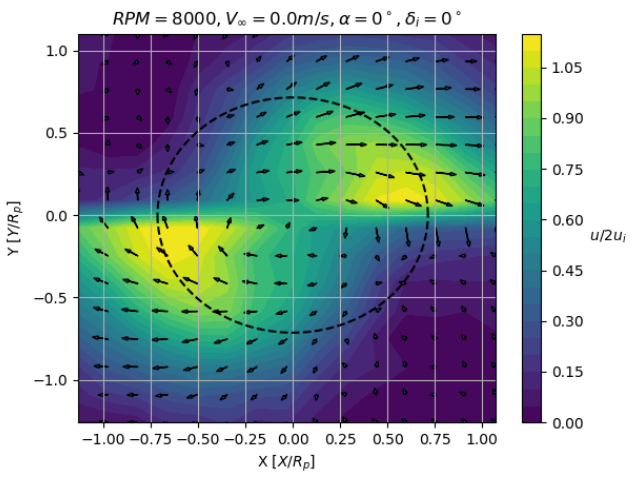


Figure 7: Velocity distribution at survey plane for symmetrical configuration at $8000rpm$

Above and below the wing, propeller slipstream can be identified as a semi-circular region of high energy airflow. Within the slipstream, both u and V_t are noticeably higher in magnitude than the surrounding flow region. The increase in axial velocity is expected as the propeller produces forward thrust by accelerating air in downstream direction. The transverse velocity is caused by the air resistance against blade rotation. Transverse velocity contains both induced velocity and viscous effect, and is commonly referred to as swirl in rotary wing terminology.

According to momentum theory [8], the induced axial velocity at propeller disk can be related to thrust coefficient.

$$u_i = nD\sqrt{\frac{2C_T}{\pi}} \tag{1}$$

where n is rotation speed in revolution per second and D is propeller diameter. Thrust coefficient is defined as $C_T = \frac{T}{\rho n^2 D^4}$, and was obtained as tabulated data from propeller manufacturer at different rotation speeds [9]. After the rotor plane, contraction of slipstream accelerates flow towards twice of u_i at downstream infinity. The flow survey is non-dimensionalised using the induced axial velocity at ultimate wake. The benefit of such normalisation is to remove the effects of thrust loading and rotational speed.

A circle in dashed line represents the undisturbed slipstream boundary obtained from vortex theory from McCormick [10], where

$$R(\bar{z}) = R_p\sqrt{1 + \bar{z}^2 - \bar{z}\sqrt{1 + \bar{z}^2}} \tag{2}$$

where $\bar{z}$ is the distance from propeller plane normalized by R_p and $\bar{z}$ is negative downstream. Through comparison with the actual high-speed regions, a distinct separation of flow structures between the extrados and intrados can be observed.

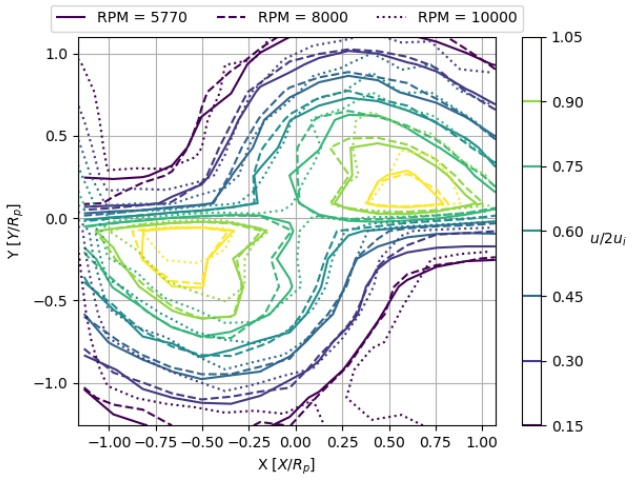


Figure 8: Comparison of axial velocity distribution at different rotation speeds

While increases in u_i and V_t can be reasonably explained by free propeller theory, movement of the two slipstream re-

gions can't be similarly explained. For a single propeller, the slipstream will stay together as in an approximate cylindrical shape. But when a wing is present, as seen in Figure 7, the upper slipstream exhibited a general displacement towards the right (outboard) while the lower slipstream region moves oppositely towards the left (inboard). The directions of movement is associated with the direction of propeller, where in the test case, the inboard blade was turning upward relative to the wing chord.

Axial velocity contours of cases from three different rotation speeds are plotted in Figure 8, where the solid line depicts u distribution at $5770rpm$, dashed line represents the one at $8000rpm$ and dotted line is for $10000rpm$.

Plotted in non-dimensional form, the contour lines of three different cases generally overlap for most flow region. Major differences lie close to the axial velocity peaks at intrados and extrados. The general agreement of flow topology suggests that at hover condition, the wake development is scalable with thrust loading and blade rotation.

A quantifiable measurement is made by determining the centres of extrados and intrados slipstreams. Due to the presence of wing wake, the slipstream centre cannot be easily defined. An indirect method was used to determine slipstream centre through shear stress at the boundary.

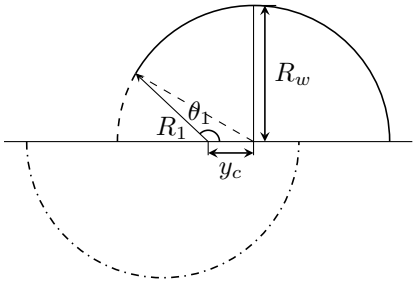


Figure 9: Geometry relations to determine slipstream centre

From turbulent jet theory, it can be concluded that the axial velocity profile of a round jet surrounded by static air can be approximated by Gaussian function. The jet boundary corresponds to where the extrema of shear stress exists. If streamwise partial derivatives ($\frac{\partial}{\partial x}$) are assumed to be small compared to cross flow derivatives, the cross-flow shear stress can therefore be determined.

$$\tau_{xy} = \mu \frac{\partial u}{\partial y} \tag{3}$$

$$\tau_{xz} = \mu \frac{\partial u}{\partial z} \tag{4}$$

The wake boundary was then determined to be the locus of maximum transverse shear stress, drawing analogy from conclusions of turbulent jet theory.

$$(y, z) : max \sqrt{\tau_{xy}^2 + \tau_{xz}^2} \tag{5}$$

The vertical extrema of the slipstream boundary were chosen as the radius of contracted wake R_w. The angular and radial position of the closest points of slipstream boundary to rotational axis were determined as R_1 and θ_1. From geometry relations, the slipstream centre can then be determined.

$$y_c = \sqrt{R_w^2 - (R_1 \sin\theta_1)^2} + R_1 \cos\theta_1 \tag{6}$$

The displacement of slipstream centre from propeller axis can therefore be found, and the results for three test cases can be found in Table 2.

RPM	C_T	y_c/R	Theoretical y_c/R	Error
5770	0.1907	0.4290	0.4252	0.9%
8000	0.1908	0.4086	0.4253	3.9%
10000	0.1906	0.4017	0.4252	5.5%

Table 2: Centreline displacement at different rotation speeds

$$\bar{y}_c(\bar{z}) = \begin{cases} 0, & \bar{z}_{LE} \le \bar{z} \\ \frac{2}{\pi}(\tan\phi - \sec\phi)(\bar{z} - \bar{z}_{LE}) & \bar{z}_{TE} \le \bar{z}, \\ \quad -\frac{2}{\pi}\sec\phi\left(\sqrt{1+\bar{z}^2} - \sqrt{1+\bar{z}_{LE}^2} + \ln\left|\frac{\bar{z}_{LE}}{\bar{z}}\frac{1-\sqrt{1+\bar{z}^2}}{1-\sqrt{1+\bar{z}_{LE}^2}}\right|\right), & \bar{z} < \bar{z}_{LE} \\ \frac{2}{\pi}(\tan\phi - \sec\phi)(\bar{z}_{TE} - \bar{z}_{LE}) & \\ \quad -\frac{2}{\pi}\sec\phi\left(\sqrt{1+\bar{z}_{TE}^2} - \sqrt{1+\bar{z}_{LE}^2} + \ln\left|\frac{\bar{z}_{LE}}{\bar{z}_{TE}}\frac{1-\sqrt{1+\bar{z}_{TE}^2}}{1-\sqrt{1+\bar{z}_{LE}^2}}\right|\right) & \bar{z} < \bar{z}_{TE} \\ \quad +\frac{2}{\pi}\left[\tan\phi(\bar{z} - \bar{z}_{LE}) - \frac{\bar{z} - \bar{z}_{TE}}{\bar{z}_{TE}\left(\sqrt{1+\bar{z}_{TE}^2} - \bar{z}_{TE}^2\right)}\sec\phi\right], & \end{cases} \tag{7}$$

From Table 2 it can be concluded that the three cases have nearly identical wake displacement. A theoretical result was also calculated for each case. This value is based on a potential flow method considering the mean chord surface as an imaginary plane, and thus a transverse velocity is induced from streamwise vortices in propeller slipstream, such idea was first introduced in [6] qualitatively and a quantitative model has been proposed by Leng et al [11].

The resulting model for centreline displacement $\bar{y}_c = y_c/R$ is a function of downstream location $\bar{z} = z/R$, with blade tip vortex shedding angle ϕ as a parameter. The centreline displacement is given in equation 7 at static condition.

Angle ϕ can be calculated from momentum theory using thrust coefficient, and $\bar{z}_{LE}$, $\bar{z}_{TE}$ are leading edge and trailing edge locations divided by propeller radius with origin at rotor centre and negative direction pointing downstream.

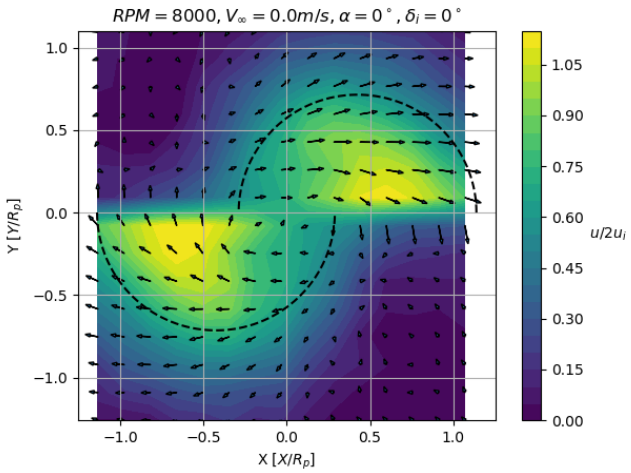


Figure 10: Velocity distribution at survey plane for symmetrical configuration at $8000rpm$, with displaced slipstream boundary

In Figure 10, slipstream boundary from momentum theory was displaced by the predicted amount from Table 2. The deformed boundary appeared to include both high-speed flow regions at extrados and intrados. The results confirm that at static condition, displaced centreline can be accurately calculated using the theoretical model. The results seem to affirm that the presence of wing serves as an imaginary plane for slipstream vortex system, and its induced transverse velocity component explains centreline displacement.

3.2 Effect of Flap Deflection

In subsection 3.1 the slipstream development in $0°$ flap deflection configuration was presented and analysed. In this condition the wing wasn't lifting, and thus the transverse slipstream displacement was purely caused by the presence of solid surface between the extrados and intrados parts of slipstream.

Results obtained at $8000rpm$ are included and discussed in this section, while the other results are included in appendix for simplicity. The effects discussed in this section are similar at a different tested rotation speed.

Figure 11 demonstrated the wake survey in a similar fashion as in Figure 7. The dashed line represents the flap trailing edge location when deflected. High speed region can still be observed in the velocity field, but the distribution took a different shape because of the deflection of flap. Besides the transverse displacement in left and right directions, the slipstream profiles also differ from each other in their vertical expansion. On the extrados, the slipstream was displaced towards the right and took a slightly narrower width. While the highest point of extrados slipstream stayed close to 1 propeller radius, the region spread lower and generally followed the deflected trailing edge flap. The extended vertical expansion is consistent with the reduced lateral width, since flow continuity must be satisfied.

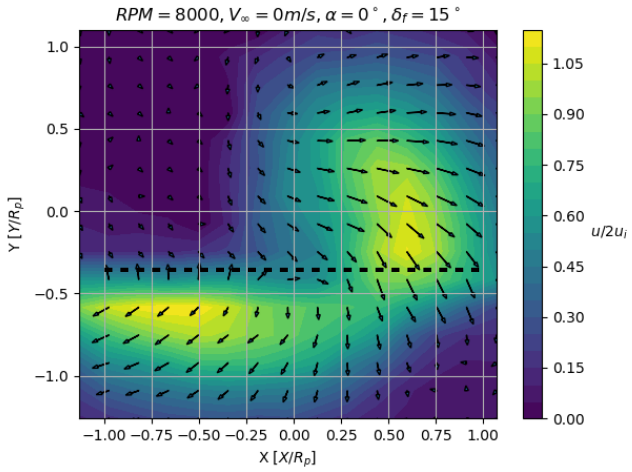


Figure 11: Velocity distribution at survey plane for with $15°$ flap deflection at $8000rpm$

The intrados slipstream was wider and flatter compared to the extrados slipstream and Figure 7. The combined effect produced a distinct velocity difference for the wing section after up-going blade (inboard section), while such difference was more subtle on the other side. The non-uniform velocity distribution could imply significant local lift variation in the surveyed section.

Wake survey for negative flap deflection is depicted in Figure 12. The velocity distribution is generally axial symmetric of Figure 11. However the vertical extent of the intrados slipstream is slightly larger than the extrados slipstream in positive flap deflection. In Figure 12, the wake boundary of intrados slipstream is shown lower than 1 propeller radius.

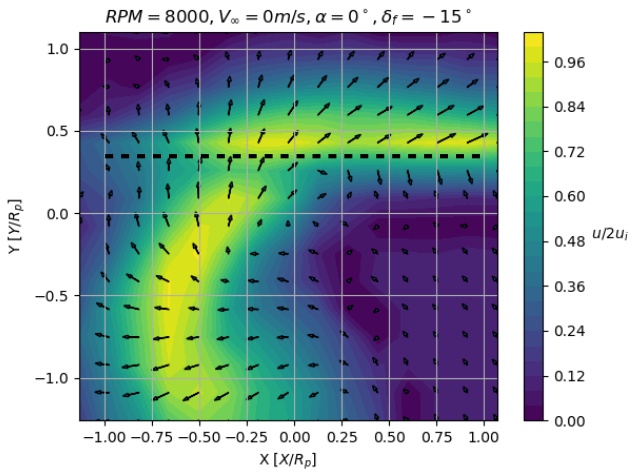


Figure 12: Velocity distribution at survey plane for with $-15°$ flap deflection at $8000rpm$

4 CONCLUSION

In this paper, a wake survey was presented immediately after a propeller-wing combination to investigate the flow interaction for a convertible UAV under hover condition. A symmetric wing profile was tested in ENAC indoor flight arena at calm wind condition. Velocity magnitude and direction were measured by a 5-hole probe at a plane perpendicular to streamwise direction and downstream of trailing edge. The test was conducted with zero flap deflection, as well as with flap deflection of $15°$ in either direction.

The results demonstrated that the presence of wing influences velocity distribution within propeller slipstream compared to a free propeller. In the experiment, the upper half slipstream was observed to translate towards outboard while the lower half slipstream translates towards inboard. The results contrast with most reduced-order model of propeller-wing interaction where propeller wake was assumed to keep its cylindrical shape.

Comparison with a theoretical model suggests that wing influence on propeller slipstream velocity distribution can be accurately modelled using method of reflection on slipstream streamwise vorticity.

The influence of wing on velocity distribution within slipstream was observed to be different between upper and lower surfaces when flap deflection was present, with the deformation being stronger on the wing surface opposite to flap deflection.

REFERENCES

[1] Quadplane. `http://uaventure.com/fcs/`, [Online; accessed 31-July-2019].

[2] LLM Veldhuis. Review of propeller-wing aerodynamic interference. In *24th International Congress of the Aeronautical Sciences*, volume 6, 2004.

[3] Gavin Kumar Ananda Krishnan, Robert W Deters, and Michael S Selig. Propeller-induced flow effects on wings of varying aspect ratio at low reynolds numbers. In *32nd AIAA Applied Aerodynamics Conference*, page 2152, 2014.

[4] Kitso Epema. Wing Optimisation for Tractor Propeller Configurations: Validation and Application of Low-Order Numerical Models Adapted to Include Propeller-Induced Velocities. Master's thesis, Delft University of Technology, Delft, Netherlands, 2017.

[5] Robert W Deters, Gavin K Ananda, and Michael S Selig. Slipstream measurements of small-scale propellers at low reynolds numbers. In *33rd AIAA Applied Aerodynamics Conference*, page 2265, 2015.

[6] Dave P Witkowski, Alex KH Lee, and John P Sullivan. Aerodynamic interaction between propellers and wings. *Journal of Aircraft*, 26(9):829–836, 1989.

[7] Bruce A. Reichert, Bruce J. Wendt, and United States. *A new algorithm for five-hole probe calibration, data reduction, and uncertainty analysis*. National Aeronautics and Space Administration ; National Technical Information Service, distributor [Washington, DC] : [Springfield, Va, 1994.

[8] Gordon J Leishman. *Principles of helicopter aerodynamics with CD extra*. Cambridge university press, 2006.

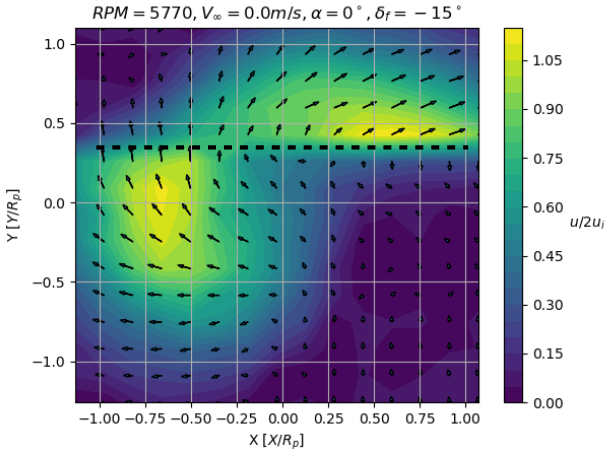
[9] Apc propeller performance data. `https://www.apcprop.com/technical-information/performance-data/`, [Online; accessed 21-May-2019].

[10] B.W. McCormick. *Aerodynamics, Aeronautics, and Flight Mechanics*. Wiley, 1994.

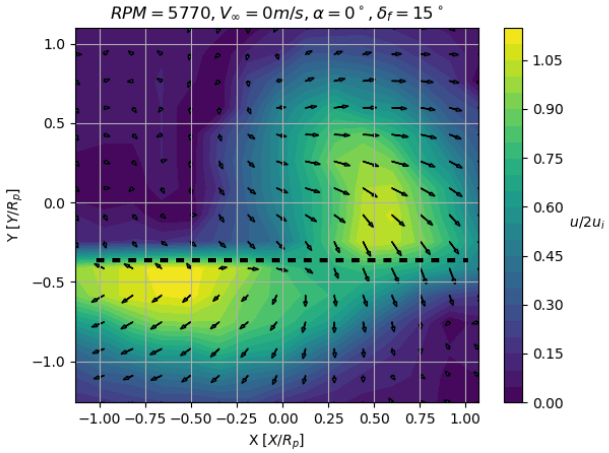
[11] Yuchen Leng, Murat Bronz, Thierry Jardin, and Jean-Marc Moschetta. Comparisons of different propeller wake models for a propeller-wing combination. In *8th European Conference for Aeronautics and Space Sciences*, 2019.

APPENDIX A: WAKE SURVEYS WITH FLAP

Wake surveys with $\pm 15^{\circ}$ flap deflections are depicted in Figure 13a and Figure 13b for propeller rotation speed at $5770rpm$.



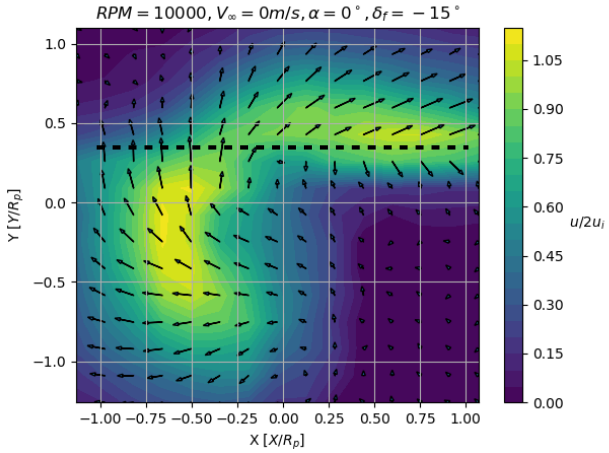
(a) -15° flap deflection



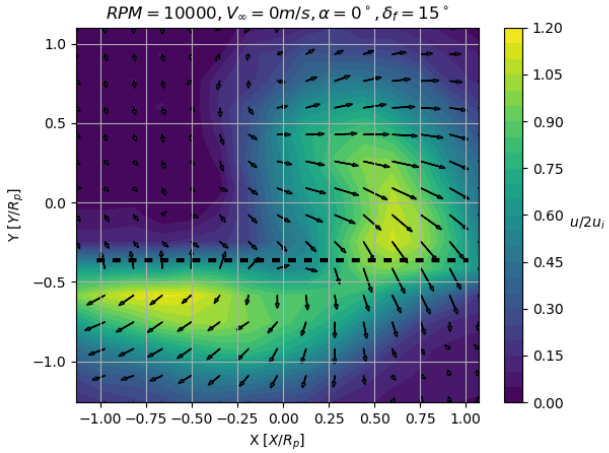
(b) 15° flap deflection

Figure 13: Velocity distribution at survey plane for with flap deflections at $5770rpm$

Wake surveys with $\pm 15^{\circ}$ flap deflections are depicted in Figure 14a and Figure 14b for propeller rotation speed at $10000rpm$.



(a) -15° flap deflection



(b) 15° flap deflection

Figure 14: Velocity distribution at survey plane for with flap deflections at $10000rpm$

Design of a high performance MAV for atmospheric research

Aurélien Cabarbaye, Titouan Verdu, Fabien Garcia, Michel Gorraz, Alexandre Bustico, Murat Bronz, and Gautier Hattenberger
ENAC, Université de Toulouse, France
firstname.lastname@enac.fr

ABSTRACT

This article presents the design of a mini UAV dedicated to atmospheric research with tight operational constraints coming from the end-users, which are the meteorologists associated to the project. Several aspects are covered in addition to the conceptual design of the frame itself and its manufacturing process. This includes the innovative launching system based on water rocket, the design of a 5-hole probe for wind and turbulence measurements, the new version of the on-board autopilot and finally the evaluation of a long range communication system. Preliminary results are presented to conclude the paper.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAV) are now commonly used for scientific research and especially in atmospheric research [1, 2] with a wide range of scale and mission. They have proved themselves to be cheaper and more agile than manned aircraft or balloons to probe the low-level atmospheric boundary layer. It is also important to be able to operate them in all sorts of weather conditions, including rain and strong winds [3]. However, flying a small UAVs at high speed, in harsh conditions and with a flight time as long as possible raises many technical challenges, especially to keep the overall system easy to operate.

One of the key measurement is the turbulence of the atmosphere which usually requires the use of 3D wind probes [4]. These type of sensors are already commercially available, like the μADC from Aeroprobe, but are pretty expensive and requires cumbersome electronic boards. It is also important to increase the sampling frequency with the speed of the aircraft in order to maintain a large usable bandwidth [5].

This article presents the design of a high performance small UAV designed for atmospheric turbulence measurements at high speed. After presenting the mission and the operational constraints in section 2, section 3 will describe the aircraft conceptual design and its innovative launching system, the next section will then focus on the on-board system integration and the design of a 3D wind probe. Finally preliminary flight and sensors tests will conclude the paper.

2 MISSION OVERVIEW AND CONSTRAINTS

The mission considered in this article is the measurement of the turbulence of the atmosphere with a small UAV. This is part of the NEPHELAE project, which involves atmospheric scientists as well as researchers in aerial robotics.

From the long experience of the authors in similar UAV operations [6, 7], it has been decided that a particular focus will be made on the take-off and landing procedures, as well as the capability of the plane to fly in strong wind conditions.

The UAV should be able to fulfil the following requirements:

- Lift a maximum of 800 grams of scientific payload in a roomy enough cargo bay.
- Cruise more than 2 hours to give the entire observation period coverage without having to land for recharging.
- Cruise at an average altitude of 3000 m above sea level where weather phenomena take place.
- Cruise at a speed of around 25 m/s which is fast enough to both counter the violent turbulences accompanying observed weather phenomena and to move from one measurement point to another in an acceptable time.
- The whole, fully equipped, aircraft should not weight more than 2.5kg to be easily transportable even in remote operating areas.

One of the main issue that is arising from these requirements is coming from the rather high cruise flight speed which prevents the use of off-the shelves airframes that would not be able to fly as fast (or as long, if overpowered) as expected, as it will be seen in section 3.

Figure 1 shows a typical location for operating UAVs without any proper ground infrastructure such as a runway. There is not even a flat space for belly landing. The UAV system must therefore presents VSTOL capability, which will be detailed in sections 3.2 and 3.3.

3 AIRFRAME DESIGN

Aircraft conception is an iterative process that covers the conceptual design, the definition of the launching and recovering systems and the manufacturing choices.

Figure 1: Base camp for operations of MAV in harsh environment.

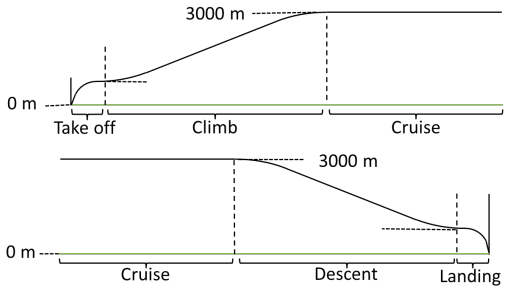


Figure 2: UAV typical mission.

3.1 *Aircraft conceptual design*

The conceptual design consist in optimizing the aircraft characteristics assessing its performances on a typical mission.

The basic architecture of the aircraft is predefined:

- A pusher configuration is adopted in order not to interfere with the sensors accommodated in the nose of the aircraft (cf. section 4).
- An electric power-plant is selected for its simplicity in operation. Sufficient autonomy can be achieved when combined with lithium-ion batteries. The battery is a homemade pack of 18650 type cells of 3500mAh capacity each.
- The aircraft is likely to withstood high vertical wind gradients. Therefore, a flying wing configuration is adopted in order to reduce turbulence by minimizing the longitudinal distance between the aerodynamic center and the lifting surfaces.
- The electric systems in the fuselage are air cooled by a flow entering through a particle separator located in the nose.

One of the design case mission considered is presented in Figure 2 which is a vertical profile up to the top of the cloud formation.

Main optimization parameters are the chord and the span of the wing, the motor and the propeller. The software used (*GENCAB*) for this study is based on genetic algorithms, differential evolution and non-linear simplex (Nelder Mead algorithm). This hybridisation of global and local techniques makes the convergence of the overall algorithm quicker and also increases the robustness of the tool over a variety of problems, and in particular problems involving alphanumeric parameters. The optimization process leads to a high wing loading to reduce the zero lift drag produced by the wing wetted area. High wing loading presents the other advantage to be less turbulence sensitive. However its drawback is a higher stall speed which has been limited to 13 m/s for take-off and landing operations.

Moreover, the propulsion system maximum power have been chosen much higher than what is required in cruise in order to reduce the climb time to 3000m.

3.2 *Launching systems*

The relatively high stall speed imposes the use of a launching systems since a whole prototype has been lost during a hand launched test.

Several existing options have been considered:

- Cataplut launching seems to be the most popular solution. Tests have been performed with two off the selves systems. It results that either they do not transmit sufficient energy to the aircraft (*SKYWALKERRC*) or the acceleration is much too high (*Gatewing*) for the structure, the autopilot and the payload. Figure 3 shows the position of a dummy UAV along the time obtained during a test. The final velocity is about $17.5m.s^{-1}$ which would be sufficient, but the acceleration is around $9.8g$ which is unacceptable. In addition, the maximum velocity is reached at less than one meter high which is

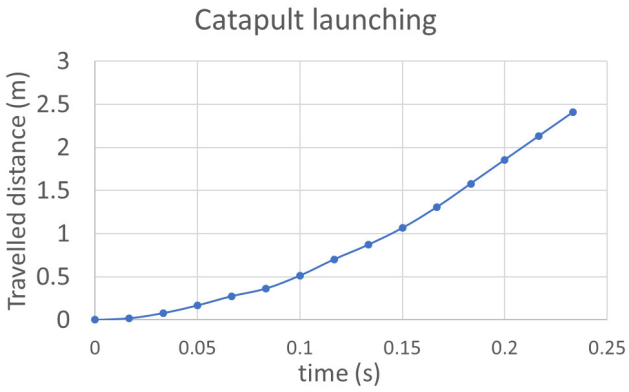


Figure 3: catapult launching performance

dangerous for operators and does not provide sufficient clearance.

- Winch launching using a bungee and placing the plane on a ramp at the beginning achieves good repeatability. It presents much lower accelerations, however this solution requires planes with rather low stall speed so that the plane doesn't touch the ground after leaving the ramp. If it is not the case, a longer ramp can be used at the expense of a possible uncontrolled yaw moment due to the friction of the wing on the ramp.
- Car (manned or RC) top launching do not present the latter defect but requires an even longer clear flat area to be operated.
- Trebuchet launching or multicopter lifting are more unusual solutions because they require relatively bulky facilities.
- Several UAVs are rocket launched following the idea of the Zero-length launch of manned aircraft developed in the 1950's. The main issues with this method is the fire hazard represented by the solid fuel rocket.

Therefore, a specific takeoff system based on a water rocket has been designed. Its main advantages are: limited acceleration, high height clearance, very high reachable airspeed, very compact facilities, only little hazard compare to the solutions previously presented. In addition, the launch pad consists in a tube inserted in the water rocket that guide it at the very beginning of the launch and reduce the waste of water acting like a piston. The maximal acceleration can be limited by reducing the nozzle diameter, while the internal volume of the water rocket is computed in order to store the required amount of energy. The acceleration has been limited here to 4g. The UAV is released at 15 m/s and 8 m high with an initial pressure of only 9 bars. The acceleration is thus much more gentle than with a catapult as it can be noticed in Figure 4.

However, its integration is more restrictive than for a classical Jet-Assisted Take-Off (JATO) system that is jettisoned in flight like a drop tank. As water rocket has indeed a much lower specific energy than solid fuel rocket, it cannot be fixed underneath the aircraft and must be placed at the rear of the drone like the first stage of a rocket. In this position, the CG is very far back. An additional lifting surface must therefore be added at the very back of the water rocket to restore the natural longitudinal stability. The additional lifting surface area is computed in order to move the aerodynamic center behind the center of gravity of the set with a static margin of 5% of the UAV aerodynamic chord. The UAV and the rocket are connected thanks to a cone shape coupler made of a male plug mounted on the motor shaft and a female plug put on top of the rocket. A parachute is added to the assembly through a cable ensuring the separation of the two components on time.

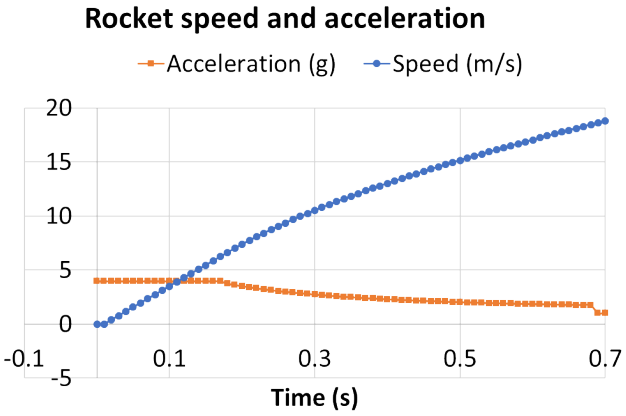


Figure 4: Water rocket performances

The parachute is laid on the floor beyond the drone and is naturally inflated when the UAV passes over it. A diagram illustrating the operation principle of water rocket launching is shown in Figure 5.

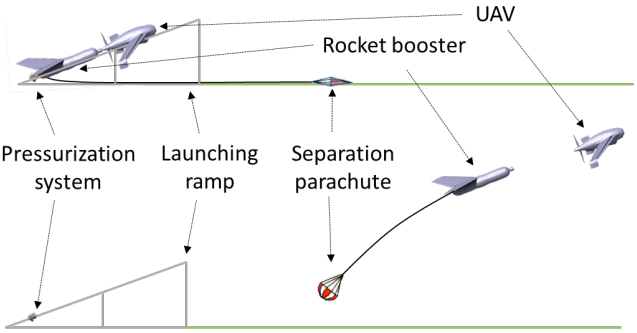


Figure 5: Water rocket launch phases.

3.3 *Recovering systems*

Several solutions can be implemented to obtain a short or vertical landing. The most popular solution consists in accommodating a ballistic parachute in the airframe. Despite the fact that very reliable systems are available on the market for the considered aircraft size, the additional weight is too high not to be detrimental.

Arresting hook is another solution but it still requires a piece of runway and constitute another overweight.

Therefore, a vertical arresting net has been selected since it can be used on almost every cleared ground and do not need any additional on-board system.

3.4 *Manufacturing process*

The manufacturing process is custom to deal with the unusual constraints of the chosen UAV operation mode. In addition, the manufacturing must be easy, fast, low cost in order to produce the whole fleet on time, on cost and with the re-

quired quality. To do so, several solutions have been adopted as shown in Figure 6.

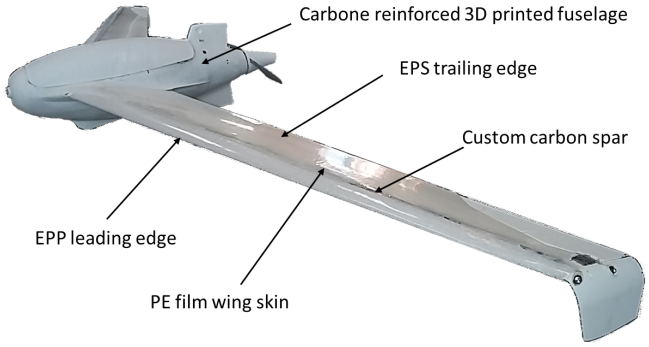


Figure 6: UAV construction.

The high aspect ratio combined with the high wing loading and the high expected turbulence impose the use of a spanwise carbon spar. In order to gain weight, the back half of the wing is built from an extruded polystyrene foam block with a hot wire CNC. This material has a high stiffness suitable for the sharp trailing edge. The front half is built with the same tool but from a block of expended polypropylene which offers a very good shape memory that is of great interest when the UAV impacts the arresting net. The joint between the two parts is made of a mixture of resin and glass micro-sphere. The obtained wing is covered with a high-density polyethylene film that protects it against abrasion and improves its polishing.

The planned number of UAVs is too low to justify the purchase of fuselage moulds. Therefore an innovative manufacturing process has been developed. The external skin of the fuselage is 3D printed in PET. The PET is chosen because of its very good inter-layer cohesion and its very low wrapping despite its higher density compared with ABS. However, the fuselage is built with the minimum possible thickness that ensures good printing result. The strength of the fuselage is obtained by laying carbon fibers on the inside surface of the printed part. The resulting fuselage presents the strength of a carbon molded part at the cost of a 3D printed part.

The water rocket is built by splicing two 2L soda bottle back to back. the maximum tolerable pressure is around the expected 9 bars. Therefore, a carbon fiber fabric is wrapped around the bottles assembly. Test have been performed up to 12 bars without showing any weakness. The wing is made of a carbon spar reinforced extruded polypropylene foam shape to withstood hard landings. The general layout of the water rocket booster is shown in Figure 7.

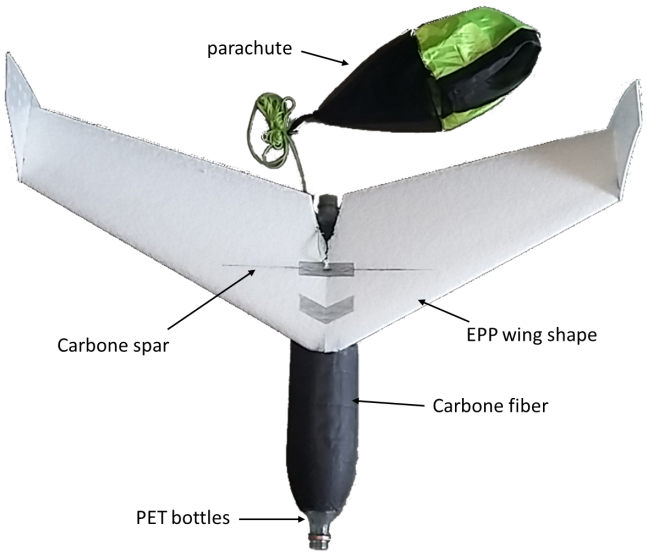


Figure 7: Rocket booster construction.

4 SYSTEM INTEGRATION

4.1 Autopilot

The open-source autopilot system used to equipped the UAVs is called *Paparazzi*[8]. The previous hardware boards used during past project was reaching end-of-life since some components were declared obsolete by the manufactures, so it was decided the design a new version.

The general guidelines were to keep the board small, eventually with the same footprint than the previous generation for easier upgrade, improve the connectivity to support more sensors and use latest available integrated components such as MCU and IMU. The main characteristics of this new board, named *Tawaki*, are listed in Table 1.

MCU	STM32F7
IMU	ICM20600 (accel, gyro) + LIS3MDL (mag)
Baro	BMP3
Serial	3 UARTS, I2C (5V + 3.3V), SPI
Servo	8 PWM/DShot output (+ ESC telemetry)
RC	2 inputs: PPM, SBUS, Spektrum
AUX	8 multi purpose auxiliary pins (ADC, timers, UART, flow control, GPIO, ...)
Logger	SD card slot
USB	DFU flash, mass storage, serial over USB
Power	6V to 17V input (2-4S LiPo) 3.3V and 5V, 4A output
Weight	12 grams

Table 1: General characteristics for the Tawaki autopilot board.

The Figure 8 shows the top and bottom 3D view of the final design of the Tawaki autopilot. A first prototype have already been assembled and successfully tested so far.

4.2 Communication system

The communication system on this UAV needs to provide at least a 10 km range and accommodate for several aircraft.

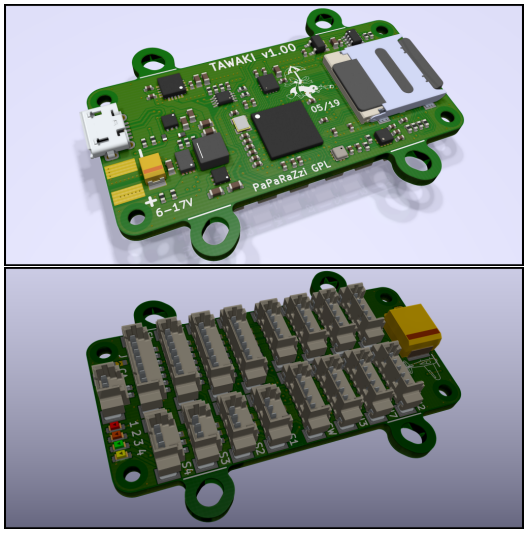


Figure 8: 3D view of the final design of the Tawaki autopilot

As the 2.4GHz *XBee* modems used in previous projects were a bit too short in range, the P2400 from *Microhard* has been selected in place. Its main characteristics are shown in Table 2. It must be noted that two types of Lora modems providing long range communication have been tested but the throughput of theses modems was two low to be usable for a UAV command and control link.

Frequency band	$2400 \leftrightarrow 2483.5MHz$
Spreading method	Frequency hopping
Link to autopilot or computer	UART up to 230.4 kbps
Link Rate	19.2 to 345 kbps
Emitted power	up to 1W
Power consumption	TX peak 6.6W
Size	26.5x33x3.5mm
Weight	5g

Table 2: General characteristics for the P2400 modem.

In order to estimate the communication range, a modem has been installed on board a UAV flying in circles at 150 m altitude while a car was driving away with another modem attached to a computer running a test program. To the day of writing this article, tests have been done with a wireless rate of 172.8 kbps resulting in a maximum range of 4.3km. This range is not sufficient for the planned mission, nevertheless the sensitivity of the modem (and hence its maximum communication range) increases when one configures a lower wireless rate allowing for a range increase. Through simple computation with the Friis free-space transmission formula, the feasible range have been evaluated for a 55 kbps wireless rate to be around 7.7 km. Furthermore, we plan to use a directive antenna which would give us a further boost in range up to 14 km which should provide enough range even counting the attenuation due to clouds.

4.3 Turbulence probe

In order to measure the turbulence of the atmosphere, a commonly used type of sensors are the multi-hole probes. Their primary usage are to measure the angle of attack and the sideslip angle, which doesn't require a very high sampling frequency, around 10 Hz. However, the dynamic of the turbulence requires a larger bandwidth with a minimum of 100 Hz.

For this project, a new integrated probe using fast differential pressure sensor is under development. The main characteristics are presented in Table 3.

MCU	STM32F7
Differential pressure	3 x SDP31
Absolute pressure	LPS33HW
IMU	ICM20600 (accel, gyro)
Logger	SD card slot
Data	UART, USB
Powering	5V
dimensions	⌀ 22 mm, L 110 mm min

Table 3: Main features of the 3D wind probe

Since it integrates a micro-controller and a SD card directly inside the probe, the recording and processing at high frequency can be done with minimum latency and independently of the main autopilot. A serial UART connection allows to forward the filtered and pre-processed data to autopilot, which will eventually forward the messages to the ground for real-time monitoring.

In addition to this, IMU data can be recorded for angle correction, or processed in real-time using attitude filter. The serial connection can also be used to receive external data such as GPS position, speed and heading for a better reconstruction of the local 3D wind field.

The Figure 9 shows the preliminary design of this new sensor. The expected sampling frequency is 500 Hz for the differential pressure and 50 Hz for the absolute pressure.

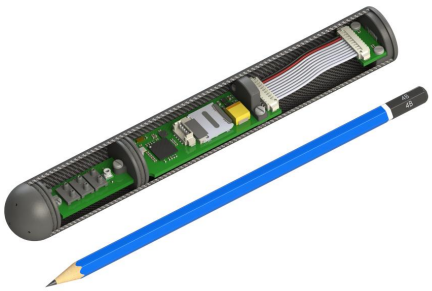


Figure 9: Inner view of the integrated 3D wind probe

4.4 Other sensors

In addition to the 3D wind probe, other meteorological sensors can be embedded inside the plane:

- Temperature and humidity, mostly used to build vertical profiles of the atmosphere the same way weather balloons would do.
- Cloud sensor [9] used to determine some of the characteristics of the water droplets and the liquid water content (LWC) of the atmosphere.
- Radiation sensors used to measure the albedo.

5 FLIGHT TESTS AND EXPERIMENTAL RESULTS

5.1 Flight patterns simulations

In order to efficiently sample the clouds, dedicated flight patterns have been defined based on the feedback from the scientists that will post-process them.

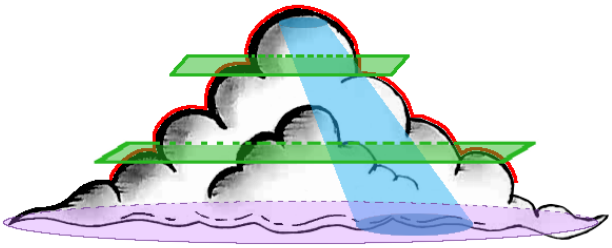


Figure 10: The different parts of the cloud that will require a specific focus during the mission.

The result is a set of predefined adaptive shapes that allows focusing the acquisition on certain parts of the cloud such as the border, the base or the core as shown on Figure 10. The real-time feedback from the sensors reduces the required time to sample a particular region corresponding to the physical process being studied, thus providing a better set of data to reconstruct the complete evolution of the cloud. The other benefit of this expert-based approach is that the computational load is limited compared to previous work on this topic [7] and can be performed in real-time by the on-board autopilot without heavy computations on the ground.

These strategy have been tested in simulations with realistic aircraft and cloud models [10]. The Figure 11 is showing the flight tracks of three UAVs where two of them are measuring the vertical wind intensity at two different altitudes to estimate the airflow inside the cloud, while the last one is covering the border in 3D to estimate the volume. The turnarounds at the edge of the cloud are triggered based on the liquid water content estimation in real-time.

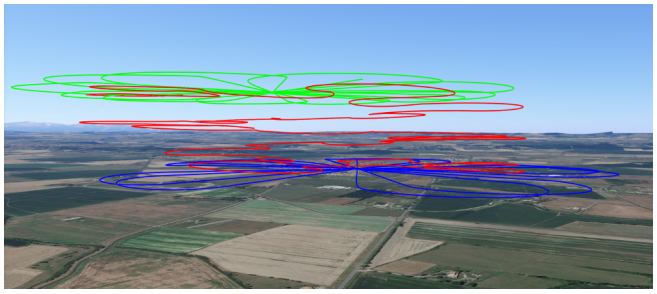


Figure 11: Simulation of 3 UAVs sampling 2 horizontal planes and the last one covering the outer 3D border.

5.2 Flight tests

Preliminary test flights have been performed with the optimized planes. At the very beginning, the issues during take-off using conventional techniques, leading to a loss of the plane, conduct to the design of the rocket-based launching procedure.

The second second session of flight tests have revealed an issue linked to the structural vibrations of the aircraft. Those vibrations, which disturb the attitude (AHRS) estimation filter, should be reduced prior to more flights. Two solutions have been implemented, which are a mechanical damping of the autopilot and a better filtering of the accelerometer data.

5.3 Five-Hole Probe Calibration (Prototype)

At the time of writing this draft paper, the final turbulence probe was not ready. In preparation for it, the calibration procedure has been done with prototype electronics, and an off-the-shelf 5-hole probe from AeroProbe. The prototype electronics consists of three differential pressure sensors (SDP31), an absolute pressure sensor, and STM32L4 microcontroller.

The calibration process has been done inside a low speed wind-tunnel facility, where the test section dimensions are 50cm by 50cm. The static ports on the probe are located at the very tip of the tube that has a spherical shape.

During the calibration process, it is important to keep the tip of the probe at the same location (fluid stream tube) in order to avoid possible uncertainties that may arise from nonuniform distribution of the speed. Therefore, a custom calibration bench has been designed and build that holds the tip of the probe in its center during the pitch and yaw movements. Figure 12 shows the calibration mechanism inside wind tunnel's test section. The bench has been designed to move from -20 deg, up to +20 deg both in angle of attack and side slip axes. During calibration, first the airspeed is fixed, and then for each angle of attack, side slip angle has been changed. Once all angle of attack and side slip angles are covered, the wind tunnel speed has been changed and the process is repeated for different airspeed (8m/s, 11m/s, and 14m/s).

A linear regression on the calibration data has been ap-

Figure 12: Calibration bench located inside the test section of ENAC's low speed wind-tunnel.

plied in order to estimate the angle of attack, side slip angle and the dynamic pressure by using the three on-board differential pressure sensors as

$$\begin{bmatrix} \alpha \\ \beta \\ q \end{bmatrix} = A \begin{bmatrix} P_1/P_3 \\ P_2/P_3 \\ P_3 \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix},$$

where A is a $[3 \times 3]$ coefficient matrix (obtained from the linear regression), and c_{1-3} are the offset values.

As the angle of attack and side slip angles should be estimated independent of the airspeed, the linear regression features has been selected in a normalized way being P_1/P_3 and P_2/P_3. It can be understood that P_3 is the differential pressure sensor that is connected to the total pressure and static pressure holes.

Figure 13 shows the result of the predicted angle of attack, side slip angle and the airspeed with respect to ground truth that comes from bench mechanism servo angles and wind-tunnel airspeed setpoint. The actual airspeed has been calculated by predicted dynamic pressure and the measured on-board absolute pressure and temperature values.

6 Conclusion and Ongoing Work

This paper has presented the overall design of a small UAV optimized for meteorological research, such as cloud and turbulence analysis. This includes the conception of the plane itself, taking into account the requirements of the atmosphere researchers, but also the adaptation of the manufacturing process to make the structure easily reproducible and adaptable and the launching system adapted to the specific performances of the aircraft. In addition, a special attention to the system integration has led to the conception of an integrated 3D wind probe based on 5-hole measurements, a new

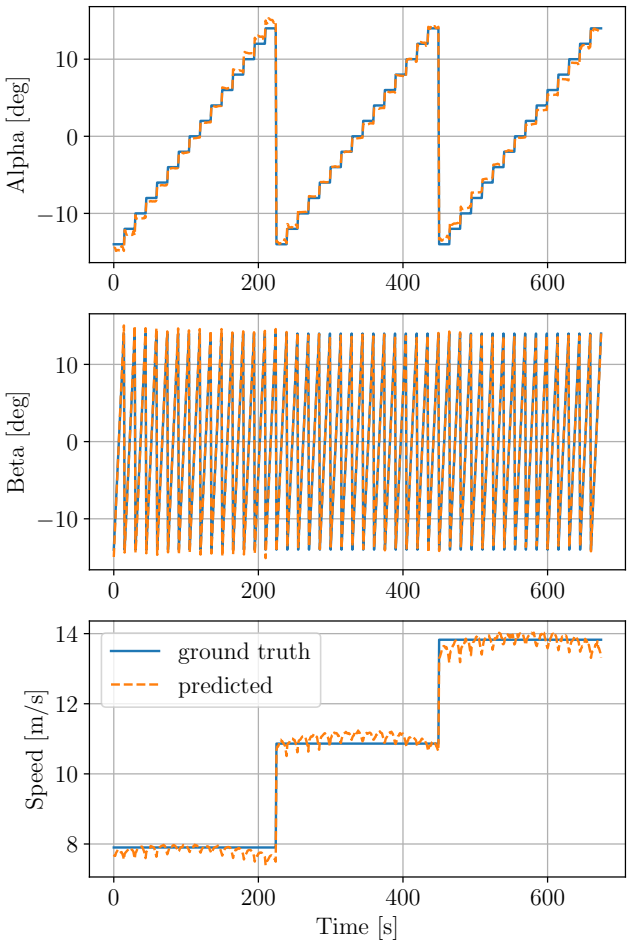


Figure 13: Comparison of predicted versus ground truth values of angle of attack, side-slip angle and the airspeed.

version of the main autopilot board and the integration of long range communication devices.

Future work will focus on the plane performances evaluation in real-flight, completing the final design of the 3D wind probe and manufacturing it in order to test it in flight.

Acknowledgments

The NEPHELAE research project is founded by the French National Research Agency (ANR).

The authors would like to thank the researchers from the LAAS-CNRS and Meteo-France involved in this project.

References

[1] Brandon M. Witte, Robert F. Singler, and Sean C. C. Bailey. Development of an unmanned aerial vehicle for the measurement of turbulence in the atmospheric boundary layer. *Atmosphere*, 8(10), 2017.

[2] Konrad Bärfuss, Falk Pätzold, Barbara Altstädter, Endres Kathe, Stefan Nowak, Lutz Bretschneider, Ulf

Bestmann, and Astrid Lampert. New setup of the uas aladina for measuring boundary layer properties, atmospheric particles and solar radiation. *Atmosphere*, 9(1), 2018.

[3] Keri Allan. Within limits, how are drones being used to improve weather predictions? *Meteorological Technology International*, pages 38–42, April 2019.

[4] Radiance Calmer, Greg Roberts, Jana Preissler, Solene Derrien, and Colin O'Dowd. 3D Wind Vector Measurements using a 5-hole Probe with Remotely Piloted Aircraft. *Atmospheric Measurement Techniques*, 2018.

[5] Dale A. Lawrence and Ben B. Balsley. High-resolution atmospheric sensing of multiple atmospheric variables using the datahawk small airborne measurement system. *Journal of Atmospheric and Oceanic Technology*, 30(10):2352–2366, 2013.

[6] Gautier Hattenberger, Grégoire Cayez, and Greg Roberts. Flight tests for meteorological studies with MAV. In *IMAV 2013, International Micro Air Vehicle Conference and Flight Competition*, page pp xxxx, Toulouse, France, September 2013.

[7] 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.

[8] 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.

[9] Gilles Harrison, Martin Airey, Graeme Marlton, Keri Nicoll, and Paul Williams. Volcanic ash detection. *Meteorological Technology International*, pages 54–56, September 2017.

[10] Titouan Verdu, Gautier Hattenberger, and Simon Lacroix. Flight patterns for clouds exploration with a fleet of uavs. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, Atlanta, 2019.

Effects of asymmetrical inflow in forward flight on the deformation of interacting flapping wings

D. N. W. M. Heitzig*, B. W. van Oudheusden, D. Olejnik, and M. Karásek
Department of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, 2629 HS Delft, the Netherlands

ABSTRACT

This study investigates the wing deformation of a flapping-wing micro air vehicle (MAV) in climbing and forward flight conditions. A measurement setup was developed that maintains adequate viewing axes of the wings for all pitch angles. Recordings of a high-speed camera pair are processed using a point tracking algorithm, allowing 136 points per wing to be measured simultaneously with an estimated accuracy of $0.25\,\mathrm{mm}$. Results of the climbing flight study show that although inflow is symmetric, the wing deformations are slightly asymmetric. Furthermore, it was found that an air-buffer remains present between the wing surfaces at all times, especially with increased freestream velocity. Apart from a minor camber reduction, the clap-and-peel motion remains mostly unchanged for changing velocities, while during the remaining cycle the incidence angle and camber ratio are reduced, together with the angle of attack. In forward flight the clap-and-peel motion is twisted around its contact area to align with the inflow direction, while the general deformation remains unchanged, suggesting similar effectiveness as in hover. Positive mean incidence angles are present for the entire cycle, especially for fast forward flight and stroke reversals. Furthermore, camber is positive during downstroke, while approaching zero for the upstroke in fast forward flight, which suggests low loading during the upstroke.

1 INTRODUCTION

Aerodynamic efficiency of flapping wing fliers was poorly understood until the later part of the 20th century. The low Reynolds number regime in which insects fly should not allow for sufficient lift production to fly. Nonetheless, relatively high lift coefficients are found for hovering insects [1]. Reason for this was later found to be the appearance of strong leading edge vortices (LEV), which remain attached to the wing surfaces and delay stall. Especially the 'clap-and-fling' mechanism was found to harness this effect strongly [2], where LEV are created between two separating wings.

*Email address: dorian.heitzig@me.com

The wings were thereby assumed to separate rigidly, without any deformation. Subsequent studies however showed that insect wings are highly flexible, which further increases efficiency due to passive deformations of the wing shape such as dynamic camber production and wing twisting [3]. Camber production is thereby assumed to be especially influential, as it delays the LEV detachments, which improves the delayed stall effect [4].

All effects occur due to the interaction of aerodynamic and structural, i.e. inertial and elastic, forces and must therefore be considered in all discussions. Inertial forces acting on the wing trailing edge were for instance found to result in a phase lag [5], which initiates a recoil effect after stroke reversal that is beneficial to thrust production [6]. Elastic forces built up over the stroke can lead to extended rotation of the wing trailing edge at the stroke end, while aerodynamic forces act as damping [7].

In this study, the specific interaction of the flapping wing micro air vehicle (MAV) 'DelFly II' [8], henceforth simply called 'DelFly', is investigated. This MAV features two wing pairs in an X-wing configuration, which 'clap-and-peel' [5] on each side. Several studies of the force production [9] and flowfield around the DelFly have already been carried out [10, 11, 12], however the wing deformation was treated comparatively little [7]. So far, for simplicity the deformation was considered to be purely symmetrical, as only a stationary hover case was studied. This study extends on this work and introduces a freestream velocity in which the DelFly is pitched to different angles, thus simulating forward flight. This problem is especially interesting, as the clap-and-peel deformation was investigated only very little outside its designed symmetrical condition [13] and potentially opposing effects such as asymmetrical camber and incidence angle deformations are seen to come into play as fast forward flight is approached [3, 6].

2 EXPERIMENTAL SETUP

The used DelFly MAV (Figure 1) consists of only the X-wing pair with half span, $s_{tip} = 140\,\mathrm{mm}$, the flapping mechanism enabling stroke angles of $\phi = 44°$, the electronic speed controller and a central airframe to which it is mounted. Similar as in other tethered studies of the DelFly, the tail is omitted, and power is supplied externally using a laboratory power-supply and servo tester to generate the flapping frequency signal. The wing material is chosen to be $15\,\mu\mathrm{m}$ thick

Mylar, together with the default stiffener setup featuring a D-shaped leading edge rod which increases the stiffness in the stroke plane [8].

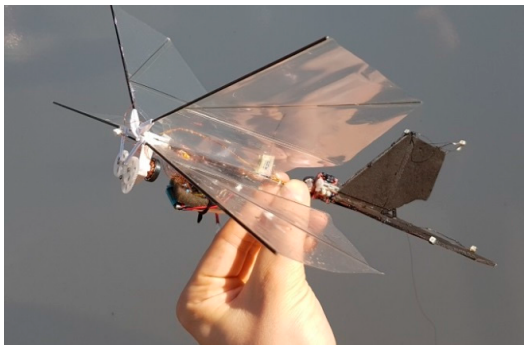
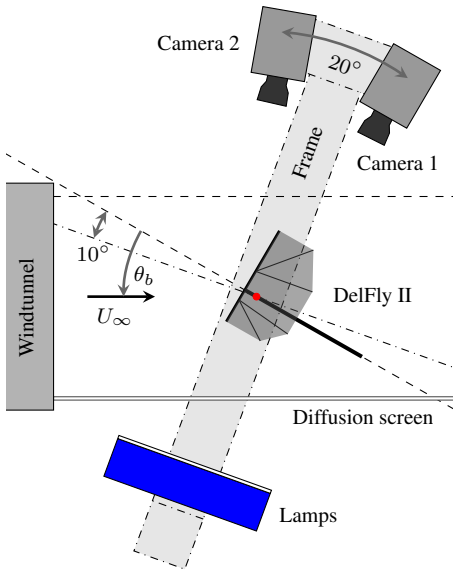


Figure 1: DelFly II MAV. The used model omits the tail and electronics. [11]

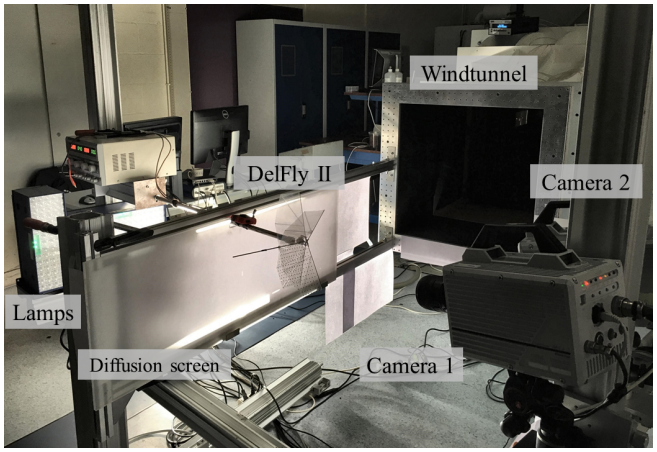
Several different optical measurement methods were considered for the measurement of the wing deformation. Ultimately, a back-light point tracking method [7] was chosen, that tracks distinct points placed on the wing over time, thus giving information of the overall wing deformation. Preliminary tests showed that compared to other methods such as digital image correlation [14] or fringe projection [15, 16], this method can be used to measure both wings simultaneously from one stereo view pair as it does not require opaque wings. Instead, the default transparent DelFly wings can be used, which allows points to be captured through an overlying wing. Methods that require opaque wings will either need optical measurement equipment on both sides of the object or separate measurements of the wings which must later be synchronized and aligned, likely increasing the measurement uncertainty.

2.1 Measurement setup

The basis of the measurement setup is formed by a frame, which is mounted on a rotating stage positioned below the center of an open $600\,\mathrm{mm} \times 600\,\mathrm{mm}$ windtunnel test section, as shown in Figure 2. The DelFly is mounted on its side, positioned so that the quarter wing chord is exactly over the rotational axis and at $10°$ pre-pitch relative to the plane of the frame. This proved to give the best optical access through the flapping cycle by the two cameras which are mounted at $\pm 10°$ relative to the frame. The used cameras are two Photron Fastcam SA 1.1 with a CMOS sensor with a 1024 pixel x 1024 pixel resolution and $20\,\mu\mathrm{m}$ pixel pitch capturing at $2\,\mathrm{kHz}$ and $1/2000\,\mathrm{s}$ exposure. Both are positioned around $600\,\mathrm{mm}$ from the DelFly and are fitted with a Nikon lens with $60\,\mathrm{mm}$ focal length and $f\# = 16$ mounted on a Scheimpflug adapter. The background illumination is provided by three LaVision LED-Flashlight 300 lamps also mounted to the frame. The lamps are pulsed in sync with the cameras with $10\,\%$ duty cycle. Although the lamps produce a relatively large and homogeneous light area, they are further diffused using a combination of a frosted acrylic screen and paper, mounted to the windtunnel nozzle.



(a) Top-down sketch on the test section with the DelFly pitched with θ_b around the rotation axis (red).



(b) Picture of the measurement setup showing the DelFly mounted in front of the windtunnel nozzle.

Figure 2: Measurement setup.

This setup allows the pitch angle, θ_b of the DelFly to be adjusted from $0°$ to $70°$ by simply manipulating the rotating stage. No readjustments of cameras or lamps are needed to maintain good visibility of the wing deformation, which would require frequent re-calibrations. The only exception is that for $\theta_b \geq 50°$ an additional halogen lamp is added on the camera side to provide sufficient illumination of the region close to the windtunnel nozzle where the background illumination no longer reaches.

The points are applied on the wing using a permanent marker as shown in Figure 3. Per wing, a total of 136 black markers of approximately $1\,\mathrm{mm}$ radius spaced at around $7.5\,\mathrm{mm} \times 10\,\mathrm{mm}$ are applied, thereby the grid is shifted between the upper and lower wing, so that overlapping of points during the contact phase is avoided. The marker position is exact to approximately $1\,\mathrm{mm}$, this has however only very little influence on the measurement process and in theory any arbitrary point spacing may be used.

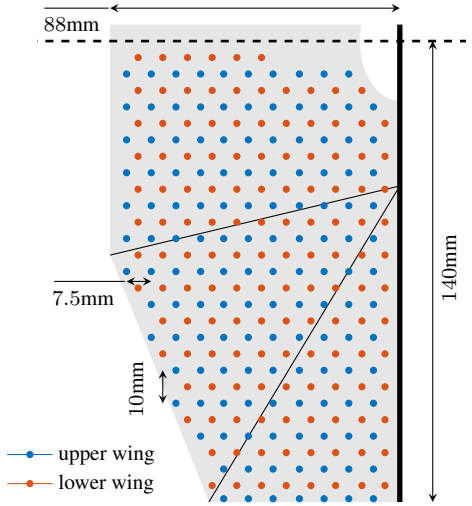


Figure 3: Schematic of DelFly wing half with point grids.

2.2 *Point tracking algorithm*

The recorded images are processed using a point tracking algorithm coded in MATLAB. Essentially, the algorithm uses a temporal tracking method to follow the image point movements, which are then triangulated to obtain the world locations. However, as points are not easily distinguishable, small errors due to for instance minor inaccuracies in the exact point location, can over time lead to larger errors such as snapping of the points to an incorrect one, especially as the path of points of the two wings often cross. Therefore, the known point spacing is used to reduce noise in the predictions and to detect errors. Once a full flapping cycle is measured, these measurements are used as predictions for the following cycles. This allows the algorithm to run fully automated, with only limited initial manual inputs. The following paragraphs explain the algorithm in more detail.

Initially, the images recorded in the LaVision software DaVis are imported into MATLAB and pre-processed, which includes distortion correction, background removal using separately recorded images, image inversion and Gaussian smoothing with a 7×7 kernel size. The camera model necessary for distortion correction and later triangulation was created using the MATLAB stereo camera calibration toolbox.

A two-stage Circular Hough Transform (CHT) method [17] is used to detect the wing points, starting with a recording where the wings are in contact and almost orthogonal to the camera view. One point per wing must be selected manually, the following will then be detected automatically using the known point spacing and an estimated magnification factor. With all points detected in all views, the stereo calibration is used to calculate the world positions.

For the subsequent timesteps, a temporal tracking method is used to predict the point locations. Therefore, an up to third degree polynomial is fitted to the growing time-series, which coefficients can be used to determine the point velocity components (after the first timestep they are assumed to be zero). The velocity vector multiplied by the timestep then gives an estimation of the point pixel shift. As the determined point locations contain some error, noise quickly accumulates in the determined velocities. Therefore, a spatial fit of the velocities is computed using radial basis interpolation using a C^2 compact support function, which is used instead of the calculated velocity if the difference between calculated and fitted velocity is larger than the velocity fit itself. It showed thereby that normalizing the velocities with the span-wise point location, analogous to the rotational velocity around the stroke axis, improves the spatial interpolation.

The true point locations are then determined as the simultaneous correspondence between all point predictions and CHT measurements which minimizes the total prediction error. This simultaneous matching of both wings avoids incorrect correspondence of points which easily occurs when points in close proximity are sequentially corresponded. The optimization is done using a mixed-integer linear programming algorithm, where duplicate use of a measured point is prevented using constraints. If no measurement that fulfills the set tolerance could be found, the point status is set to missing.

These points are neglected in the spatial predictions. Furthermore, in the following timesteps they are corresponded after successfully found points using the world re-projection instead of the temporal prediction. If the point status in one view is considered correct, the prediction is improved further by moving the re-projection onto the epipolar line.

Nonetheless, points correspondences can be incorrect, e.g. due to incorrect detection by the CHT method, or snapping to wrong points due to noise in the tracking. Therefore, a check for incorrect point measurements is also done in the world domain. Here, a spatial fit of the triangulated world points is created using the previously used radial basis interpolation for the in-plane location together with a polynomial fit for the out of plane location. Points which measured location lies more than $2.5\,\mathrm{mm}$ from the fit, or have a reprojection error above 1.5 pixel, are assumed to be incorrectly triangulated.

Once a point exceeds this tolerance by a factor of two, a correction of the triangulation is attempted. Therefore, the view with lower certainty is selected, which is the view where either no point measurement could be corresponded or where the reprojection of the spatial fit lies further from the measured point location. This incorrect point view is then handled like the missing points described above. The increased tolerance of the correction is used to prevent overuse of corrections without allowing incorrect triangulations to be considered in the spatial predictions.

After a full cycle is measured, the exact cycle length is determined which allows to combine the measurement series to a single cycle. This series is then resampled, thereby filling gaps where points could not be found, to create a prediction for the following cycles. This cyclic prediction is improved with each full cycle as new measurements are added.

The complete measurement series is low-pass filtered using a MATLAB function to remove noise. The cut-off frequency was set to the 10th flapping harmonic, i.e. between 100 Hz and 130 Hz, which is conservative compared to the influence limits found in other studies [18, 19].

The described algorithm works well in determining the DelFly wing deformation. On average, the reprojection error lies at around 0.21 pixel with 3.4 % point tracks are determined to be incorrect based on the mentioned criteria. The tracking quality is thereby lower for the second cycle half and the lower wing as point motions are possibly less favorable and larger points are more often occluded by stiffeners. Worse tracking results in isolated false positive point measurements, which increase the mean reprojection error.

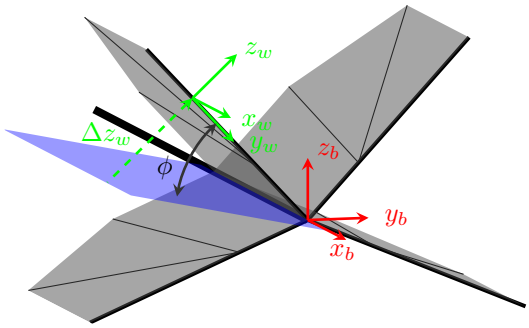
Measurements of a 150 mm diameter reference sphere were done to get a better understanding of the general setup accuracy. The 63 markers of 1 mm radius had an average reprojection error of 0.11 pixel, resulting in an average distance of 0.12 mm from the fitted sphere surface. Assuming a linear relation with the reprojection error, the deformation measurements can be said to be accurate to approximately 0.25 mm.

3 Results

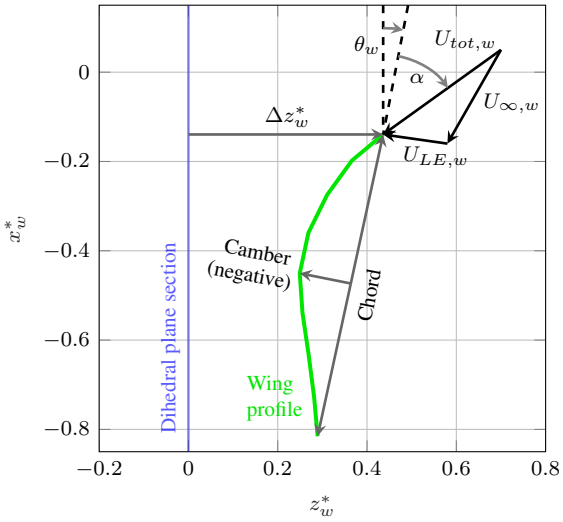
As forward flight of the DelFly results in simultaneous variation of pitch angle and flapping frequencies together with the inflow velocity, the effects of each parameter by itself should be understood. Previous studies have already addressed the effects of the flapping frequency variations [7]. Studies on the wing deformation due to increasing inflow velocity however have not yet been done, therefore this effect is analyzed before the forward flight results. Climbing flight is of further interest as recent tailless DelFly can sustain such condition more reliably [20].

The following discussion uses the measured points to represent the wing surface, i.e. the most forward and backward point are used as wing leading and trailing edge, respectively. To obtain equivalent parameters, the points on the lower wing are interpolated to match the upper wing. The total of 2000 measurements is allocated to 100 phase bins over the flapping cycle to calculate the deformation statistics. The phase is thereby indicated by the non-dimensionalized time, $t^* = t/T$, where period, $T = 1/f$, with $t^* = 0$ at the closest distance between the wing leading edge. Thus, the cycle starts with the outstroke and ends with the instroke.

The point measurements are transferred from the body coordinate system to a wing coordinate system fixed to the wing leading edge shown in Figure 4a. An exemplary $x_w - z_w$ cut at span location s_w is shown in Figure 4b. In this plot, the shown measurements are normalized by the mean chord, $c_{mean} = 80\,\text{mm}$, indicated by the asterisk and the origin is shifted by $\Delta z_w = tan(\phi) s_w$ to the intersection with the dihedral plane as done in [7] to visualize the stroke angle.



(a) Sketch of the DelFly including body (red) and local wing (green) coordinate systems and dihedral plane (blue).



(b) Definition of wing profile parameters in normalized $x_w^* - z_w^*$ plane at spanwise location s_w.

Figure 4: Used coordinate systems.

The wing coordinate system is also used to calculate different local wing profile parameters, later used for a quan-

titative description of the deformation. The camber ratio, ϵ is the ratio of camber and chord, where a curvature against z_w-direction is defined as negative, as shown. The incidence angle, θ_w is the angle between the chordline and the x_w-axis and used to describe wing twisting. The difference between incidence angle and angle of the inflow velocity, $U_{tot,w}$ is used to represent the angle of attack, α. Thereby the inflow direction is calculated from the sum of the freestream velocity in the wing reference frame, $U_{\infty,w}$ and leading edge velocity, $U_{LE,w}$, however neglects induced velocities.

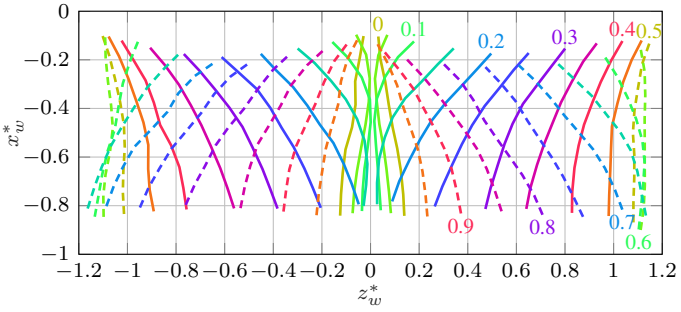
3.1 Climbing flight

In ideal hover or climb, the DelFly is orientated vertically remaining approximately in a fixed location in the horizontal plane. The vertical force is produced by the flapping wings, while conventionally a tail maintains stability. The frequency required to maintain hover lies above 13 Hz [9, 12] and must increase further to achieve climbing flight. Here, three cases were tested, ranging from hover with zero freestream velocity to U_∞ =1 m s^{-1} and 2 m s^{-1} at $\theta_b = 0°$. The flapping frequency was kept constant at 12 Hz, the results are therefore not representative for free flight, as the main objective is to understand the basic effects of non-zero inflow velocity.

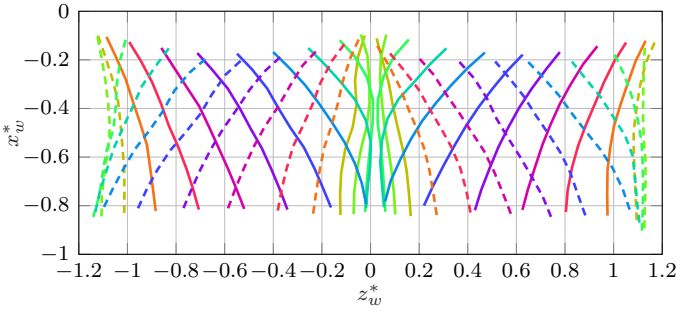
A representation of the temporal development of the wing deformation is given in Figure 5. The spanwise location, $s_w = 100\,\text{mm} = 0.71 s_{tip}$ is chosen as it showed to give a good representation of the average wing shape. In spanwise direction the deformation is relatively straightforward where the deformation magnitude typically increases towards the wing tips while maintaining the same temporal trends.

The clap-and-peel phase, where the wings are rolling off on each other can clearly be seen in the figure. The duration is slightly reduced from $\Delta t^* = 0.174$ to 0.193 (based on trailing edge detachment at $s_w = 100\,\text{mm}$) at faster climb speeds. As found by others, the leading edges thereby make no contact [7]. This gap appears larger in hover, and is increased especially towards the root, where a large gap remains for the entire first wing half. However, also for the remaining wing surface, minimal distances were found to remain. Opposed to the wing leading edge, the wing surfaces are closest at the hover case, with distances of around 0.7 mm close to the wing tip. At $U_\infty = 2\,\text{m s}^{-1}$ the minimal distance lies around 1.3 mm. The general presence of this 'air-buffer' between the wing surfaces is plausible, as viscous forces prevent large fluid accelerations close to the wing surfaces. Determining an exact reason for this behavior is however difficult, and is likely due to a combination of interactions over the entire wing cycle, such as pressure fields and elastic forces.

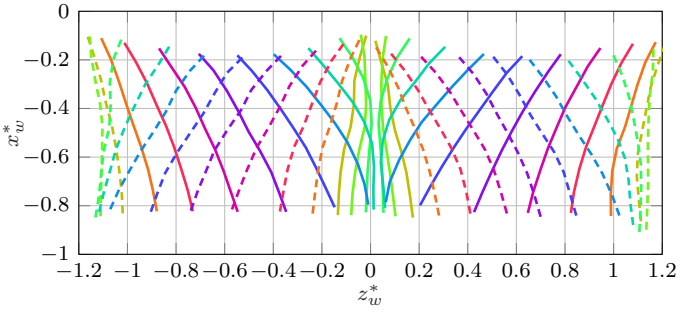
Even for this symmetrical inflow case, the wing deformations show clear asymmetries, which has not previously been noted. These asymmetries are visible especially during the end of the outstroke, where the lower wing displays



(a) Hover. t^* is indicated for the upper wing.



(b) 1 m s^{-1} climb.



(c) 2 m s^{-1} climb.

Figure 5: Wing profile deformation at $s_w = 100\,\text{mm}$ over the flapping cycle due to different climbing velocities. The upper wing can be seen in the right half, the outstroke profiles are dashed.

camber while the upper wing is mostly flat. This effect is reduced at higher flow-speeds. Also, the leading edges heave of both wings are clearly asymmetric. The upper wing heaves considerably more during the instroke than during the outstroke, while the lower wing heaves approximately identically during both strokes. Multiple reasons could result in this asymmetry. The dihedral angle of the DelFly already introduces a slight asymmetry, as the upper wings come closer to each other than the lower. This leads to minor differences in the aerodynamic behavior, as well as possible asymmetric wing tensioning. Inaccuracies in the manual manufacturing process of the wings can increase this effect. Further discrepancies may be introduced by the measurement

procedure, e.g. the support, the diffusion wall and possibly small uncertainties in the set pitch angle.

A closer investigation is done based on the wing profile parameters shown in Figure 6. Although asymmetries are again evident in the plots, as well as relatively large standard deviations (s.d.) at some instances, trends are still clearly visible and allow a discussion of the results.

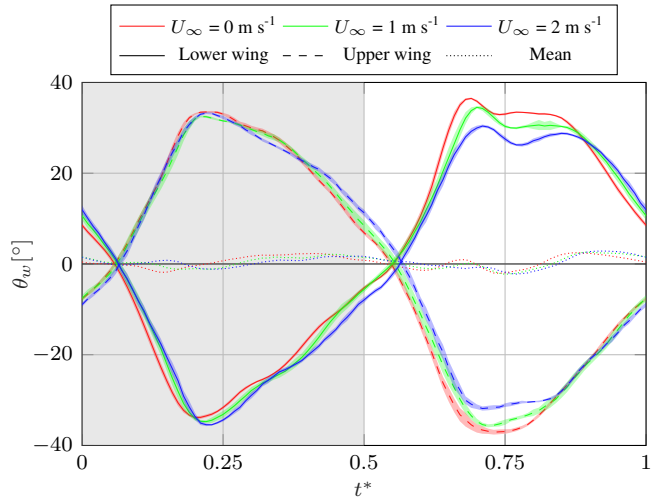
Through the clap and peel, the incidence angle changes with an almost equal rate for all cases. The wing in hover is thereby initially twisted least outwards, and therefore twisted most inwards at the end of the clap-and-peel phase.

In the outstroke phase, the wing incidence angle peaks at the time where the trailing edges detach, around $t^* = 0.18$ at this span location. Once detached, the trailing edge velocities become much larger than the leading edge velocities, which starts to reduce the incidence angle. The large acceleration is likely due to the elastic forces build-up during the clap-and-peel phase. The incidence angle of the hover case thereby start to decrease faster than that of the climb cases, which results in lower inwards wing twisting during the last part of the outstroke. Looking at the wing deformation plots, it can be seen that in fact the trailing edge location at the end of the outstroke is similar for all cases, while the leading edge moves further in the stroke plane direction.

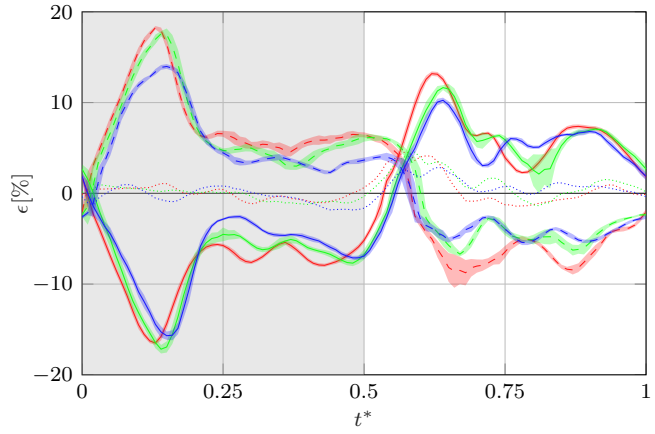
This can be explained by the aerodynamic forces, which are dominant during the high stroke velocity phase. Due to the high freestream component for the fast climb case, the inflow angle is quite low, in fact similar to the incidence angle. For $U_\infty = 2\,\mathrm{m\,s^{-1}}$, this results in a low angle of attack of $|\alpha| \approx 20°$ during the majority of the fast stroke phases, as can be seen in Figure 6c. Analysis of the spanwise distribution shows that this holds for most of the wing, only at the wing tip larger angles are found. This alignment of the inner wing surface with the inflow direction results in reduced wing loading, which in turn reduces the damping effect of the aerodynamic forces, thus allows the leading edge to stroke further.

At $U_\infty = 1\,\mathrm{m\,s^{-1}}$, the inflow angles are already considerably larger, resulting in larger angles of attack, and in the hover case even increases to $\alpha \approx 90°$, although the low velocity magnitude makes this phase irrelevant. The large spike in angle of attack around the stroke reversal occur due to small changes in the leading edge movement direction and should therefore also be neglected. Similarly, an incorrectly corresponded point of the hover case at $t^* = 0.42$ results in the downwards α spike.

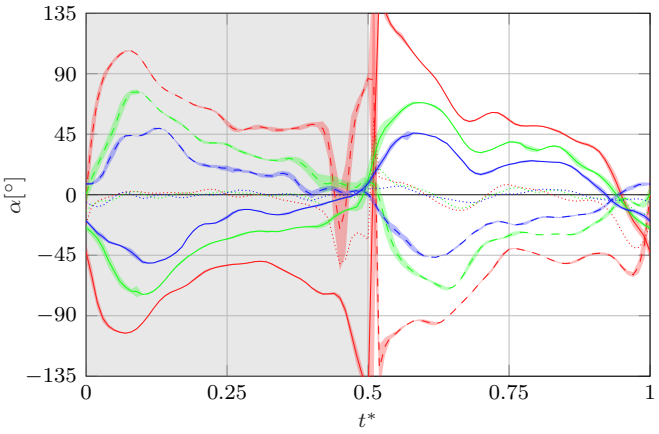
The instroke then behaves mostly as the outstroke, with the incidence angle of the climb cases again lagging the hover case. The incidence angle increases at an even higher rate, which can be linked to a torsional wave traveling down the wing span [6] seen in the 3D animation (Video 1). The



(a) Incidence angle.



(b) Camber ratio.



(c) Angle of attack.

Figure 6: Wing profile parameters at $s_w = 100\,\mathrm{mm}$ due to different climbing velocities indicated by U_∞. The outstroke phase is shaded in grey, the instantaneous s.d. is shaded in the respective color.

maximum incidence angle obtained by the hover case now reaches a considerably larger values compared to climbing flight, which can again be attributed to respective inflow direction, which results in low angles of attack and loading during climb. The incidence angle is now maintained for a longer period, where again the angle of the hover case reduces earlier, and the leading edges stop further apart when compared to the climb cases.

Apart from the wing twisting, camber or interchangeably camber ratio also plays a significant role in force production and is worth analyzing. The wing peel leads to a large camber production during the cycle start. On average, the wings of the hover case have thereby slightly larger camber ratios, which appears to be due to the faster detachment of the leading edges and a lower wing surface distance. As for the incidence angle, the maximum camber ratio is reduced quickly once the trailing edges separate. In the following out- and instroke, the camber ratio is lower for the climb cases. This is likely linked to the reduced inflow angle, which is assumed to lead to lower wing loading. At the start of the instroke, the camber shows also a peak, although lower than that during the clap-and-peel of the outstroke. This is similar to the recoil effect seen to increase thrust production in different insects [5, 6], which occurs due to inertial forces leading to trailing edge lag. Outside the clap-and-peel, the camber ratio decreases close to the wing tip, which is against the general spanwise trends and assumed to be due to the stiffener positioning [7]

3.2 Forward flight

During horizontal forward flight, the DelFly is pitched forward, which results a horizontal component of the forces produced by the flapping wings. The pitch angle and flapping frequency must then be matched to maintain horizontal flight at a certain velocity. To simulate this in the windtunnel, the parameters θ_b, f and U_∞ were set to replicate values measured in previous free forward flight investigations [9, 12]. The cases investigated here are shown in Table 1.

θ_b [°]	U_∞ [$\mathrm{m\,s^{-1}}$]	$U_{\infty,z}$ [$\mathrm{m\,s^{-1}}$]	f [Hz]
70	0.50	0.47	13.00
50	1.12	0.85	11.89
40	1.63	1.05	11.07
30	2.26	1.13	10.11

Table 1: Tethered flight settings representing free forward flight.

Changes of the wing deformation are therefore not purely dependent on the changing pitch or inflow direction, but are a result of the combination of the three parameters. Additionally to the terms in- and outstroke, which described the wing movement towards and from the dihedral plane, with the introduced horizontal orientation the terms downstroke and upstroke are now used. These describe the wing movement relative to the horizon, and correspond inversely to in- and outstroke for the upper and lower wing.

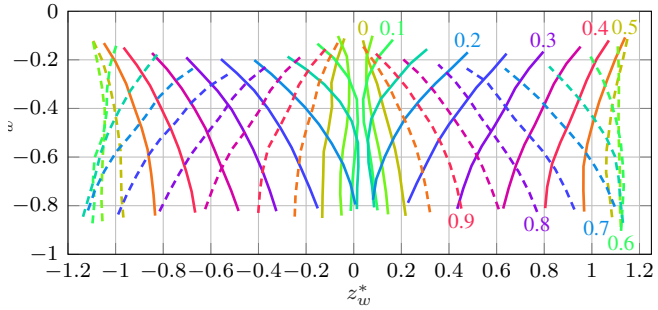
The wing profiles at $100\,\mathrm{mm}$ span location presented in Figure 7 show that the introduced asymmetry of inflow direction and force production has a large influence on the wing deformation and goes beyond the minor asymmetries found in the climbing cases. In all cases a mean incidence angle directed towards the inflow direction is present. This effect increases especially for low pitch angles, see for instance the wing contact region and the trailing edge location at the outstroke end. This suggests that the increased freestream velocity during these phases has a larger influence than the pitch angle alone. In fact, the asymmetry appears to be proportional to the freestream component in z_w direction, $U_{\infty,z} = sin(\theta_b) \cdot U_\infty$, listed in Table 1.

Interesting to see is that the core clap-and-peel deformation remains relatively unaffected by the pitch angle and appears to be simply rotated to a twisted wing contact plane. The leading edge path of the lower wing is therefore directed considerably more backwards, which results in an asymmetric heave of both wings, where the wings are heaved more during downstroke. This typically indicates higher loads during this phase, which is in line with the required lift production. The clap-and-peel duration at $s_w = 100\,\mathrm{mm}$ reduces slightly from $\Delta t^* = 0.185$ to 0.165 for faster forward flight.

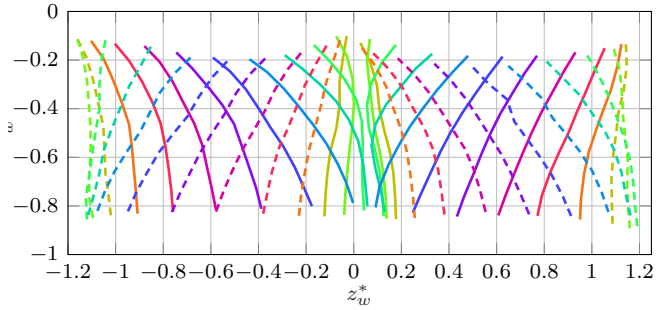
The wing parameter plots shown in Figure 8 give further insight into the deformations. For all cases, the mean incidence angle is positive over most of the cycle, especially for the cases with large normal freestream component. This trend is especially large during stroke reversal, where the inflow velocity is almost entirely made up by the freestream velocity. Also, the upstroke, where the incidence angle is negative, varies more with the pitch angle.

Little differences in incidence angle during the clap-and-peel phase show again that this phase is relatively unaffected by the asymmetric inflow. Difference is only a shift in the initial incidence angle, while the incidence angle increase rate remains identical. The final angle difference between the wings is in line with the effect of reduced flapping frequency [7].

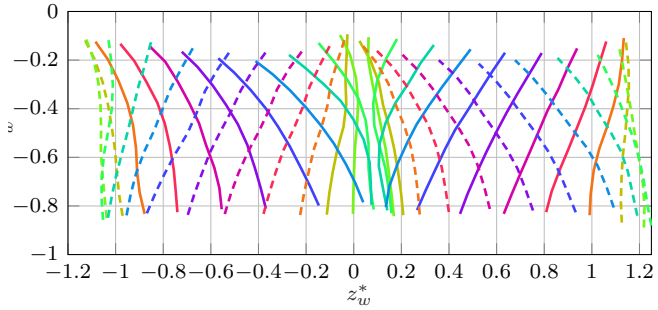
The assumption that the clap-and-peel behavior is largely unaffected is also supported by the measured camber deformation. While initially the both wings have positive camber ratios, between $t^* = 0.05$ and $t^* = 0.18$ the camber ratio is almost symmetric. The reduced magnitude can again be linked to the reduction in flapping frequency [7] and reduced freestream velocity. The $\theta_b = 50°$ case is the only outlier for



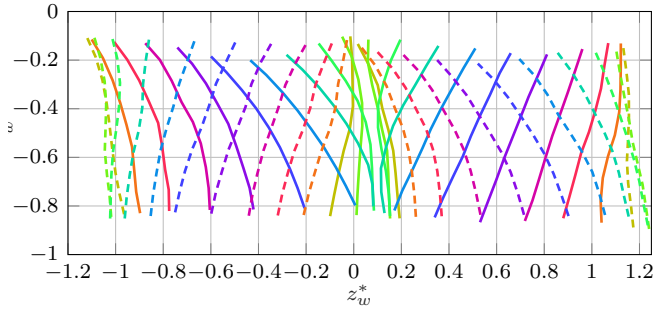
(a) $0.5\,\mathrm{m\,s^{-1}}$ forward flight at $\theta_b = 70°$. t^* is indicated for the upper wing.



(b) $1.12\,\mathrm{m\,s^{-1}}$ forward flight at $\theta_b = 50°$.

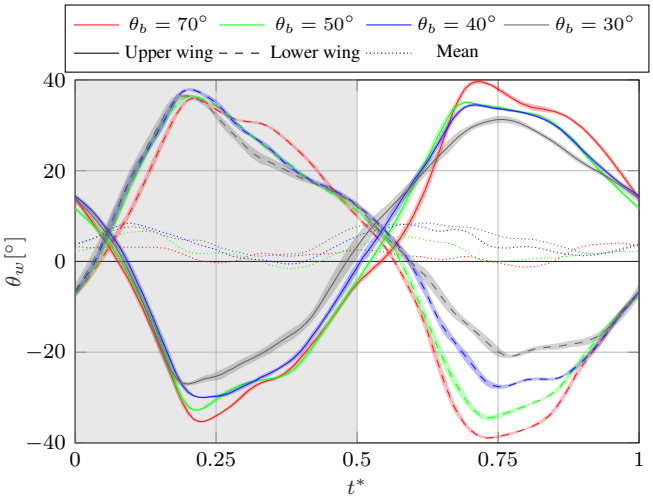


(c) $1.63\,\mathrm{m\,s^{-1}}$ forward flight at $\theta_b = 40°$.

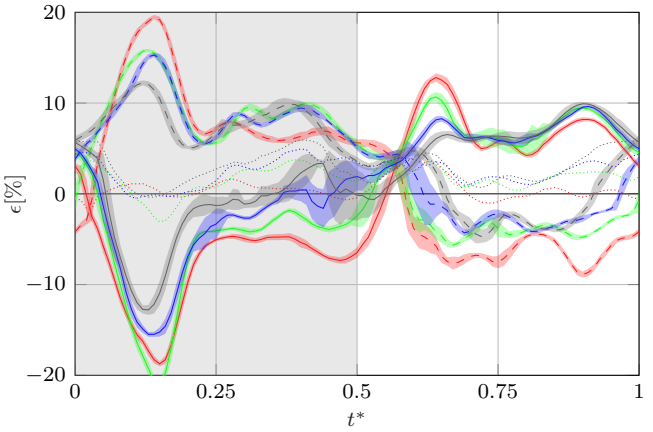


(d) $2.26\,\mathrm{m\,s^{-1}}$ forward flight at $\theta_b = 30°$.

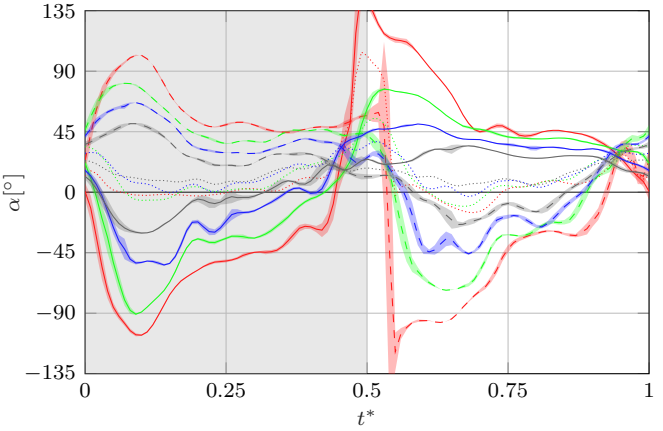
Figure 7: Wing profile deformation at $s_w = 100\,\mathrm{mm}$ over the flapping cycle due to different different forward flight velocities. The upper wing can be seen in the right half, the outstroke profiles are dashed.



(a) Incidence angle.



(b) Camber ratio.



(c) Angle of attack.

Figure 8: Wing profile parameters at $s_w = 100\,\mathrm{mm}$ due to different forward flight velocities indicated by θ_b. The outstroke phase is shaded in grey, the instantaneous s.d. is shaded in the respective color.

this assumption, the camber production during peel follows no clear trend. This appears to be due to the specific contact region rotation, which results in a sharper peel angle in this case.

For the $\theta_b = 70°$ case, the camber deformation is quite similar to the hover case. Larger asymmetries occur for lower pitch angles. Here, during the downstroke similar positive camber ratios occur for all cases, while during the upstroke the camber ratio reduces approximately proportional with the pitch angle, approaching zero for $\theta_b = 30°$, a behavior that is common in insect flight [3, 6]. Figure 7d shows that the upper wing profile has an S-shape towards the end of the outstroke, which makes the determination of the camber direction difficult, thus leads to large s.d.. The wing surface towards the leading edge is thereby already curved upwards, while the trailing edge is still curved downwards. Towards the wing tip this behavior is increased where the entire profile is inverted already before the stroke reversal. This suggests that the wing is here already beginning to produce lift and possibly a LEV starts to develop on the upper wing surface. This effectively moves the start of the lift producing phase of the upper wing forward.

This may be an explanation for the absences of a large initial camber production, previously called recoil, and the reduction of the torsional wave. Recoil remains visible only for $\theta_b = 70°$.

Further links can be drawn between the camber and twist development to the wing loading, represented by the angle of attack. As already seen for the camber development, during the clap-and-peel phase the angle of attack is relatively symmetric, and large angles of attack still indicate the production of LEV even in the fastest forward flight case. This, together with the alignment of the clap-and-peel motion with the flight direction, will likely result in the majority of the required thrust being produced during this phase. This could reduce the need for the wings to produce thrust during upstroke, as it is assumed to occur for single wing fliers [3].

Afterwards, the angle of attack is similar for both stroke halves, where during the downstroke the angle of attack remains mostly constant in time, while reducing considerably for the upstroke. This occurs due to motion of the wings relative to the freestream, which also leads to higher relative velocity of the wing during downstroke [19]. This change in loading directly corresponds to the change in camber, which partially speaks for a well working passive deformation properties of flexible flapping-wings. This can be seen particularly for the $\theta_b \leq 40°$ cases, where $\alpha \approx 0°$ during the last part of the upstroke of both wings. It was assumed that the wings already start producing lift at this span location, and indeed the sign change of α and ε coincide very closely. This is remarkable, especially considering the neglection of induced velocities in α and the likely presence of structural effects.

4 CONCLUSION

Previous studies on the wing deformation of the DelFly in hovering flight [7] were extended to climbing flight of up to $2\,\mathrm{m\,s^{-1}}$ and forward flight with $70°$ to $30°$ pitch. An optical measurement setup was developed which co-aligns a camera pair together with the DelFly and the background light to maintain adequate viewing axes of the wings which undergo large stroke angles. As the wings are transparent, 136 points applied to each wing could be measured simultaneously. The points were tracked using an in house developed point tracking algorithm, which uses known structural information to enhance the temporal tracking so that very few false point matches occur. The general accuracy lies around $0.25\,\mathrm{mm}$ based on reference sphere measurements. The developed setup and measurement algorithm may be useful in the future for investigating different flight states or other models and may also be an important tool for optimizing wing designs and generating validation data for numeric methods.

The carried out measurements show different general deformation behaviors of interacting flapping-wings, which to the best of our knowledge have not yet been noted in literature. Firstly, measurements show that the wing surfaces do not touch during the clap-and-peel phase, instead an air-buffer remains at all times. This air-buffer lies between $0.7\,\mathrm{mm}$ and $1.3\,\mathrm{mm}$ and increases with inflow velocity. Furthermore, slight asymmetries are found even in symmetrical inflow conditions. These are assumed to be to an extent inherent because of the dihedral angle which influences aerodynamics and wing tension, but may also be a result of measurement uncertainties.

Apart from this, the climbing flight study showed that the increase in freestream velocity has relatively little effect on the clap-and-peel, only minimal camber reductions were found. The incidence angle is mostly unaffected during the entire outstroke, as elastic energy stored during the clap-and-peel is released during the remaining cycle half. Otherwise, camber ratio and incidence angle are reduced with increasing climbing rate, which is assumed to be driven by lower wing loading, indicated by the angle of attack. Torsional waves traveling in spanwise direction were found, as well as recoil-like camber increase at the cycle start.

The forward flight study showed that asymmetries are especially large in fast forward flight, which suggests that asymmetric deformations are not proportional to pitch angle but to the normal freestream component. Clap-and-peel deformation is thereby again mostly unaffected, indicating that the produced LEV has a dominant effect over the asymmetrical freestream velocity and that the motion likely remains as effective as in hover. Varied is only the motion symmetry plane, which is now twisted to align more with the inflow. A positive mean incidence angle is present during the entire cycle, and especially large during the stroke reversal, where the tip velocities are low. The camber production now differs between

up- and downstroke: During the downstroke wings remain cambered upwards, while during the downstroke the negative camber is heavily reduced for fast forward flight. This, together with the angle of attack estimation indicates very low wing loading during upstroke. The calculation of α thereby neglects the presence of large unsteady aerodynamic effects, for instance due to LEV and wing rotation. This makes the estimation of the loads difficult and prevents extended discussion, which may be addressed in future work.

REFERENCES

[1] A. R. Jones and H. Babinsky. Unsteady lift generation on rotating wings at low reynolds numbers. *Journal of Aircraft*, 47(3):1013–1021, 2010.

[2] T. Weis-Fogh. Quick estimates of flight fitness in hovering animals, including novel mechanisms for lift production. *J. Exp. Biol*, 59:169–230, 1973.

[3] H. Wang. Measuring wing kinematics, flight trajectory and body attitude during forward flight and turning maneuvers in dragonflies. *Journal of Aircraft*, 206(4):745–757, 2003.

[4] G. Du and M. Sun. Effects of unsteady deformation of flapping wing on its aerodynamic forces. *Applied Mathematics and Mechanics*, 29(6):731–743, 2008.

[5] C. P. Ellington. The aerodynamics of hovering insect flight. iii. kinematics. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 305(1122):41–78, 1984.

[6] S. M. Walker, A. L. R. Thomas, and G. K. Taylor. Deformable wing kinematics in free-flying hoverflies. *J. R. Soc. Interface*, 7(42):131–142, 2010.

[7] M. Perçin, B. W. van Oudheusden, G. C. H. E. De Croon, and B. Remes. Force generation and wing deformation characteristics of a flapping-wing micro air vehicle 'delfly ii' in hovering flight. *Bioinspiration & Biomimetics*, 11(3):036014, 2016.

[8] G. De Croon, M. Perçin, B. Remes, R. Ruijsink, and C. De Wagter. *The DelFly: Design, Aerodynamics, and Artificial Intelligence of a Flapping Wing Robot*. Springer Netherlands, Dordrecht, 2016.

[9] M. Karasek, A. J. Koopmans, S. F. Armanini, B. D. Remes, and G. C. De Croon. Free flight force estimation of a 23.5 g flapping wing mav using an on-board imu. *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4963–4969, 2016.

[10] M. Percin, B. W. van Oudheusden, H. E. Eisma, and B. D. W. Remes. Three-dimensional vortex wake structure of a flapping-wing micro aerial vehicle in forward flight configuration. *Experiments in Fluids*, 55(9):729, 2014.

[11] A. del Estal Herrero, M. Percin, M. Karasek, and B. van Oudheusden. Flow visualization around a flapping-wing micro air vehicle in free flight using large-scale piv. *Aerospace*, 5(4):99, 2018.

[12] B. Martínez Gallar, B. W. van Oudheusden, A. Sciacchitano, and M. Karásek. Large-scale flow visualization of a flapping-wing micro air vehicle. *Proceedings 18th International Symposium on Flow Visualization*, 2018.

[13] T. Nakata, H. Liu, Y. Tanaka, N. Nishihashi, X. Wang, and A. Sato. Aerodynamics of a bio-inspired flexible flapping-wing micro air vehicle. *Bioinspiration & Biomimetics*, 6(4):045002, 2011.

[14] P. Wu, B. Stanford, W. Bowman, A. Schwartz, and P. Ifju. Digital image correlation techniques for full–field displacement measurements of micro air vehicle flapping wings. *Experimental Techniques*, 3:53–58, 2009.

[15] D. Song, H. Wang, L. Zeng, and C. Yin. Measuring the camber deformation of a dragonfly wing using projected comb fringe. *Review of Scientific Instruments*, 72(5):2450–2454, 2001.

[16] B. Li and S. Zhang. Superfast high-resolution absolute 3d recovery of a stabilized flapping flight process. *Optics Express*, 25(22):27270–27282, 2017.

[17] H. K. Yuen, J. Princen, J. Illingworth, and J. Kittler. Comparative study of hough transform methods for circle finding. *Image and Vision Computing*, 8(1):71–77, 1990.

[18] S. M. Walker, A. L. R. Thomas, and G. K. Taylor. Photogrammetric reconstruction of high-resolution surface topographies and deformable wing kinematics of tethered locusts and free-flying hoverflies. *J. R. Soc. Interface*, 6:351–366, 2009.

[19] S. Deng. *Aerodynamics of Flapping-wing Micro-Air-Vehicle*. PhD thesis, Technische Universiteit Delft, Delft, 2016.

[20] M. Karásek, F. T. Muijres, C. De Wagter, B. D. W. Remes, and G. C. H. E. De Croon. A tailless aerial robotic flapper reveals that flies use torque coupling in rapid banked turns. *Science*, 361(6407):1089–1094, 2018.

A Long Range Fuel Cell/Soaring UAV System for Crossing the Atlantic Ocean

N. Gavrilović*, D. Vinceković†, and J-M Moschetta‡
ISAE-SUPAERO, Université de Toulouse, 31400 Toulouse, France

Abstract

The design of a long-range unmanned aircraft system powered by fuel cells is investigated, with the study focusing on the feasibility of crossing the Atlantic Ocean. The motivation behind this aircraft is to demonstrate the capability of hydrogen fuel cells as an alternative fuel source and to create a case for future commercial and civilian aircraft. Existing hydrogen powered UAS are benchmarked and an in-depth analysis of the mission environment is conducted to ultimately determine reasonable requirements for the aircraft. The requirements of a 3000 km range, a maximum mass of 25 kg and being primarily powered by fuel cells, are used as inputs for the conceptual design phase. A classical design approach is conducted but modified appropriately for a fuel cell powered UAS. The aircraft mass, lift-to-drag ratio, wing loading and thrust-to-weight ratio are iterated until a baseline configuration that can feasibly cross the Atlantic Ocean is achieved. Some additional aspects oriented towards exploitation of atmospheric phenomena in the purpose of performance improvement will also be exposed.

1 Introduction

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.

This project aims to analyze the feasibility of an unmanned aircraft system powered by hydrogen fuel cells that has the capability of crossing the Atlantic Ocean, as detailed in Figure 1. This route has been selected as it has historical significance; it was used by the French aviation company, "Aéropostale" in the 1930's 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 and being sufficiently light-weight to be within the certification category allowing beyond line of visual sight.

Figure 1: Route from Dakar to Natal 3000 km. Source: Google Maps, 2019.

2 State of the Art

Hydrogen fuel cells generate energy through catalysis; a process that separates the electrons and protons of the reactant fuel and forces the electrons to travel through a circuit which generates an electric current as explained by Gundlach [1]. Another catalytic process then takes the electrons and combines them back with protons and oxygen from the ambient air to form water as a waste product. It is seen that a fuel cell resembles a battery because it provides a direct electrical current, however, it uses a separate fuel and oxidant that are not stored together. According to Osenar et al. [2] this makes a

*Postdoctoral researcher, Department of Aerodynamics and Propulsion (Email address: nikola.gavrilovic@isae.fr)
†Graduate student
‡Full Professor, Department of Aerodynamics and Propulsion

fuel cell system inherently safer than other advanced high energy density battery technologies. The fuel cell itself is a way to convert the fuel and oxidant but does not store any energy. In general, because air is used as the oxidant and not stored with the fuel, the energy density of the fuel system is able to exceed traditional battery systems. Additionally, this allows of the system to be scaled up easier, as only larger hydrogen fuel tanks need to be installed and not a larger fuel cell. In comparison, to increase the range of a battery powered UAS, additional batteries will need to be added which significantly increases the mass. The high electrochemical conversion efficiency of fuel cells combined with the high specific energy of H2 fuel makes them suited to extending the endurance of small UAS.

A state of the art evaluation of existing unmanned aircraft systems revealed several hydrogen fuel cell powered UAS which have been successfully designed and built. It also serves as a means to benchmark the project's objectives to a reasonable standard, ensuring a feasible project with worthwhile applications.

2.1 Sparkle Tech: Eagle Plus

The Eagle Plus is a small, VTOL, unmanned aircraft system built by Sparkle Tech based in Hong Kong and is shown in Figure 2. This UAS is a new generation hybrid fix wing and multi-rotor configuration which allows it to be versatile due to not requiring a runway and also have high endurance as the cruise phase occurs with the fixed wing. The length of the UAS is approximately 2m with the airframe made from composite carbon fibre. The Eagle Plus has a 3.5 m wingspan, a wing area of 0.7 m2 and its maximum take off mass is 21 kg. It utilizes a 500 W electric engine that is powered by a 9L liquid hydrogen fuel cell and also is equipped with a 10000mAh lithium polymer battery as a redundancy for flight safety. The total weight of the fuel tank, battery and equipment is 10 kg, which enables an endurance of 5 hours at a cruise speed of 28 m/s.

Figure 2: Eagle Plus VTOL UAS. Source: Sparkle Tech [3].

2.2 US Naval Research Laboratory: Ion Tiger

The United States Naval Research Laboratory's Ion Tiger features a conventional wing and stabilizer layout, as shown in Figure 3. The cruise velocity for the Ion Tiger is 13.9 m/s, features a wingspan of 5.2 m, a wing area of 1.57 m^2 and has a length of 2.4 m. The maximum take of mass of Ion Tiger is 16.1 kg, with a structural mass percentage of approximately 44% and cruise power of 300 W. The energy is provided by 550 W hydrogen fuel cells, stored at at pressure of 34.5 MPa. The Ion Tiger accomplished a 26 hour flight with a payload mass of 2.5 kg.

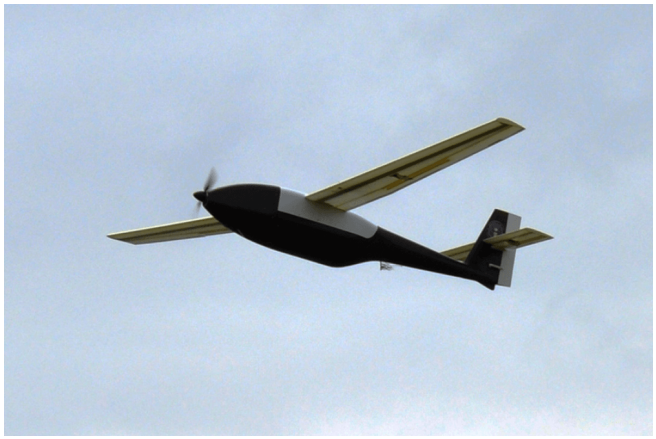


Figure 3: Ion Tiger UAS. Source: Swider-Lyons et al. [4].

2.3 H3 Dynamics: Hywings

The Hywings UAS, as shown in Figure 4, is developed by H3 Dynamics based in Singapore. It features a similar conventional wing and stabilizer layout to the Ion Tiger and has the same cruise velocity of 13.9 m/s. The maximum take off mass of Hywings is 7 kg and it features a 200 W hydrogen fuel cell system, pressurized between 30 and 35 MPa, that allows for quick hydrogen bottle refuelling. The UAS does not require a runway and can take-off via a hand launch and can achieve a range of up to 500 km and an endurance of up to 10 hours.

2.4 Insitu: Scan Eagle

Scan Eagle is developed is by Insitu, a subsidiary of Boeing, and is a small, long-endurance, low-altitude UAS that is used for reconnaissance. The original variant as shown in Figure 5, is powered by a petrol engine, however, there are currently efforts to incorporate a hydrogen fuel cell power system to drive the UAS. As the 1200 W fuel cell module fits within the existing airframe without modification, the Scan Eagle fuel cell variant aims to be a technology demonstrator with a flight time of 9 hours. The UAS maximum take off mass is 22 kg, with a length of 1.55 m and a wingspan of 3.11 m. In addition, it takes off using a runway independent launcher and lands via a hook recovery system so it does not

Figure 4: Hywings UAS. Source: H3 Dynamics [5].

Figure 5: Scan Eagle UAS. Source: Insitu [7].

require landing gear.

The main characteristics of these platforms are outlined in Table 1. The analyzed platforms all feature hydrogen fuel cell propulsion but were not limited to a specific mission. The

UAS	M (kg)	V (m/s)	b (m)	E (hr)
Eagle Plus	21	28	3.5	5
Ion Tiger	16	14	5.2	26
Hywings	7	14	1.9	10
Scan Eagle	22	25	3.1	9

Table 1: Summary of existing fixed-wing UAS powered by fuel cells.

following conclusions may be drawn from the above benchmarking analysis:

- All benchmarked hydrogen fuel cell powered aircraft have a mass between 5 kg and 25 kg, which puts them within the open UAS category under EASA's [6] classification.
- It is important to note that the leading aircraft configuration for this platform is a conventional wing design. It is also noticed that the multi-rotor configurations used with hydrogen fuel cells had significantly less range and endurance than the fixed wing configurations which is why they were not included in the state of the art.
- The endurance time varies significantly between platforms, with the Ion Tiger featuring the highest endurance. This shows that weight is not the only parameter important for endurance; it is inferred that wingspan contributes significantly as the Ion Tiger has the largest wingspan which in turn minimizes the induced drag due to the large aspect ratio.

3 Mission Specification and Equipment

3.1 Aircraft Requirements

The main design requirements for the unmanned aircraft system will be presented in this chapter. The first range requirement has been selected as this is the distance between the two cities, Dakar and Natal and the mass requirement is highly desirable to minimize the required certification for beyond visual line of sight flight. The third requirement has been selected because the motivation of the aircraft is to demonstrate the capability of enormous energy density of hydrogen fuel cells.

- Must be able to cross the distance of 3000 km.
- To have a total mass of less than 25 kg.
- Must use hydrogen fuel cells as primary energy source.

3.2 Feasibility Analysis

The initial feasibility study of basic aircraft parameters for completion of a defined mission has been performed to verify the requirements presented earlier. The iteration process has been set with basic parameters shown in Table 2. The objective of the iteration process is to achieve convergence in the total mass of the aircraft while satisfying the chosen parameters. According to a chosen speed of flight, lift to drag ratio and available energy density needed for cruise related to a fixed distance, the program will try to converge towards a total mass of the aircraft.

This analysis has been primarily performed for two sources of energy of which the lithium-polymer batteries and fuel cells. The sources of energy have been characterized through a value of specific energy density where the highly efficient lithium-polymer batteries provide around 250 Wh/kg, according to Gatti [8], which is 4 times less than for fuel cells. The analysis has revealed that the convergence process cannot be achieved with the usage of lithium batteries for such a required distance. On the other hand, the energy

Varying parameter	Value
V (m/s)	23
L/D	30
Energy Density (Wh/kg)	1015
Equipment mass (kg)	1
Structural mass (%)	40
Fixed parameter	**Value**
Distance (km)	3000
Propeller efficiency	0.8
Motor efficiency	0.8
Converter efficiency	0.9
Battery return rate	0.9
Ration motor mass/power (g/W)	0.3

Table 2: Parameters of feasibility study.

density of fuel cells achieves a steady convergence with a total mass of 20 kg for a relatively high required lift-to-drag ratio. This analysis has brought a conclusion that such a distance cannot be crossed with the usage of even most efficient lithium batters, while on the other side, the usage of fuel cells makes the journey possible.

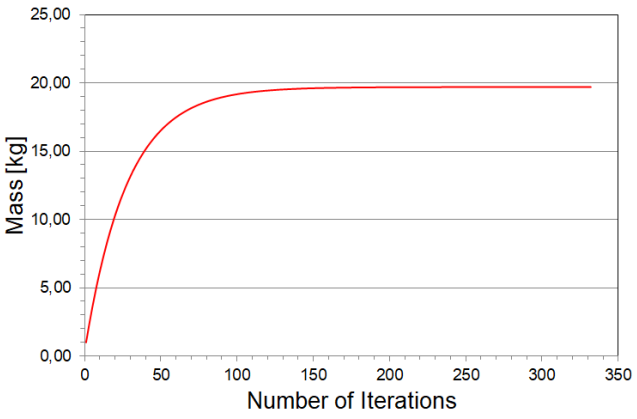


Figure 6: Mass convergence for chosen set of parameters presented in Table 2.

3.3 Fuel Cell Power System

Although there are nowadays several suppliers of modules and equipment related to the latest fuel cell technology, the present work is focused on products available from HES Energy Systems [9]. The specifications of the light weight commercially available Aerostak fuel cells (as an example shown in Figure 7) are summarized in Table 3, where the model number is the stack's rated power. Each fuel cell is cooled by an integrated fan and requires a hydrogen input pressure of 0.6 to 0.8 bar. Note that an additional 0.1 kg was considered for each fuel cell system specification for margin, as well as for the cylinder masses in Table 4.

Model	P (W)	Dimensions (mm)	M (kg)
Aerostak 250	300	110 x 120 x 124	0.83
Aerostak 500	600	194 x 105 x 166	1.4
Aerostak 1000 v1	1200	254 x 170 x 125	2.14
Aerostak 1000 v2	1300	194 x 127 x 193	1.9

Table 3: Summary of existing fixed-wing UAS powered by fuel cells.

Figure 7: Aerostak Fuel Cell. Source: HES Energy Systems [9].

The fuel cell model selected for the aircraft will be decided based on the conceptual design which will determine the required power. It is desirable to have the smallest possible fuel cell that can produce the required power as this will minimize the weight of the aircraft and improve the range capability. The hydrogen storage options are also to be provided by HES Energy Systems. The types of storage cylinders, shown in Figure 8, are summarized in Table 4 , where their model number indicates the storage volume in liters.

Model	M (kg)	Energy (Wh)	D (mm)	L (mm)
A2	1.3	784	102	385
A2.5	1.35	980	132	288
A3.5	1.75	1372	132	375
A5	1.95	1960	152	395
A9	2.95	3528	173	528
A12	3.6	4704	196	532
A20	7.10	7840	230	655

Table 4: Summary of existing fixed-wing UAS powered by fuel cells.

3.4 Mission Environment

Due to the relatively low cruise speed and mass of the UAS, the wind's speed and direction can highly impact its performance. As the cruise phase will take place over the Atlantic Ocean, the wind conditions in this region have been examined using a tool called Windy. It compiles GFS and ECMWF wind models from the Swiss company Meteoblue

Figure 8: Hydrogen Cylinders. Source: HES Energy Systems [10].

and displays the wind conditions worldwide. Figure 9 shows the wind map for an average day in March at different altitudes, with the black line marking the mission route.

It is observed that the majority of the wind's direction over the aircraft's design route is favorable under an altitude of 1 km, with wind speeds varying between 3 and 10 m/s. While the wind has the potential to reduce the power requirements during cruise and increase the range of the aircraft, it is decided to not factor the benefits of the wind into the design. Despite the observed trend over this region of the Atlantic Ocean, the wind is still considered unreliable as its direction and speed will vary depending on the day. Therefore, the favorable wind will be considered as an external bonus but the aircraft will be designed such that it is capable of achieving its mission without assistance from the wind.

4 Conceptual Design

The principal aim of the conceptual design phase is to determine the information required in order to decide whether the concept will be technically feasible and capable of meeting the design requirements. This process is primarily conducted using a classical conceptual design approach outlined by Raymer [11] and Roskam [12], but modified appropriately for a hydrogen fuel cell powered UAS. The expected outputs from this phase of the design are the aircraft mass, the required lift to drag ratio, the energy system requirements and a baseline geometric configuration.

4.1 Design Parameters

To determine the total aircraft mass, wing loading, thrust to weight ratio and aerodynamic requirements, an iterative procedure is used. The design flow, shown in Figure 10, illustrates the non-linear method, where the yellow icons represent inputs from the feasibility study of the aircraft and mission, the blue icons represent mass inputs that are iterable and the green icon represents the outputs which are the conceptual design parameters.

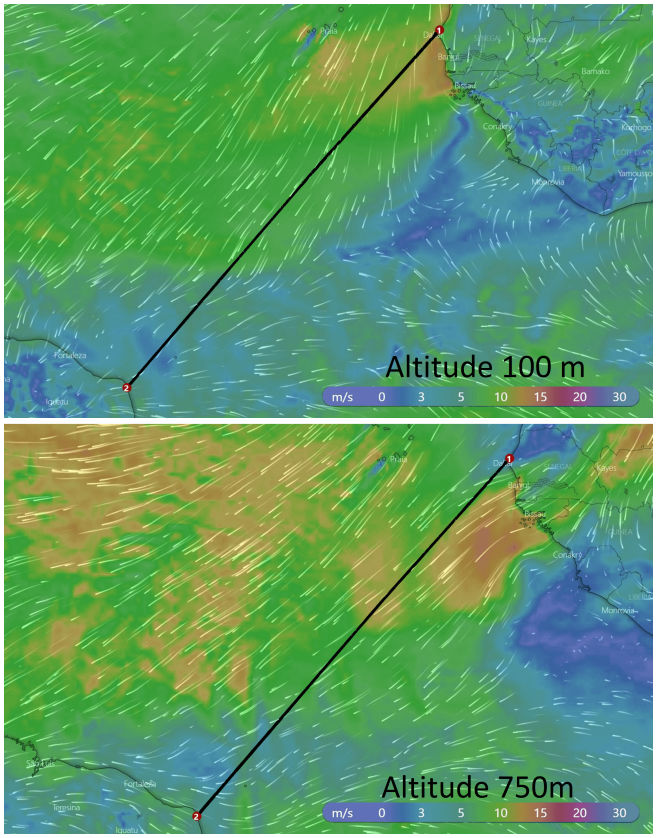


Figure 9: Wind direction at 100 and 750 m of altitude between Dakar in Senegal and Natal in Brazil.

4.1.1 Geometric Parameter Summary

The Table 5 summarizes the geometric aircraft parameters which have been determined for the initial conceptual design. These parameters will be used throughout the conceptual design of the aircraft to establish a baseline configuration, however, will be adjusted appropriately to maximize the lift to drag ratio in the preliminary design. As an example, a slight dihedral angle has been applied for stability purposes. The parasite drag coefficient has been obtained through the ratio between the surface area of the aircraft that interacts with the flow and the planar area of the wings and equivalent skin friction drag coefficient. For a light aircraft, the equivalent skin friction drag is estimated to be $C_f = 0.004$ according to Raymer [11] and the wetted area ratio is estimated from initial modems of the unmanned aircraft to be approximately 3.7. It is observed from Figure 11 that a lift to drag ratio of approximately 30 is common for powered sailplanes with as-

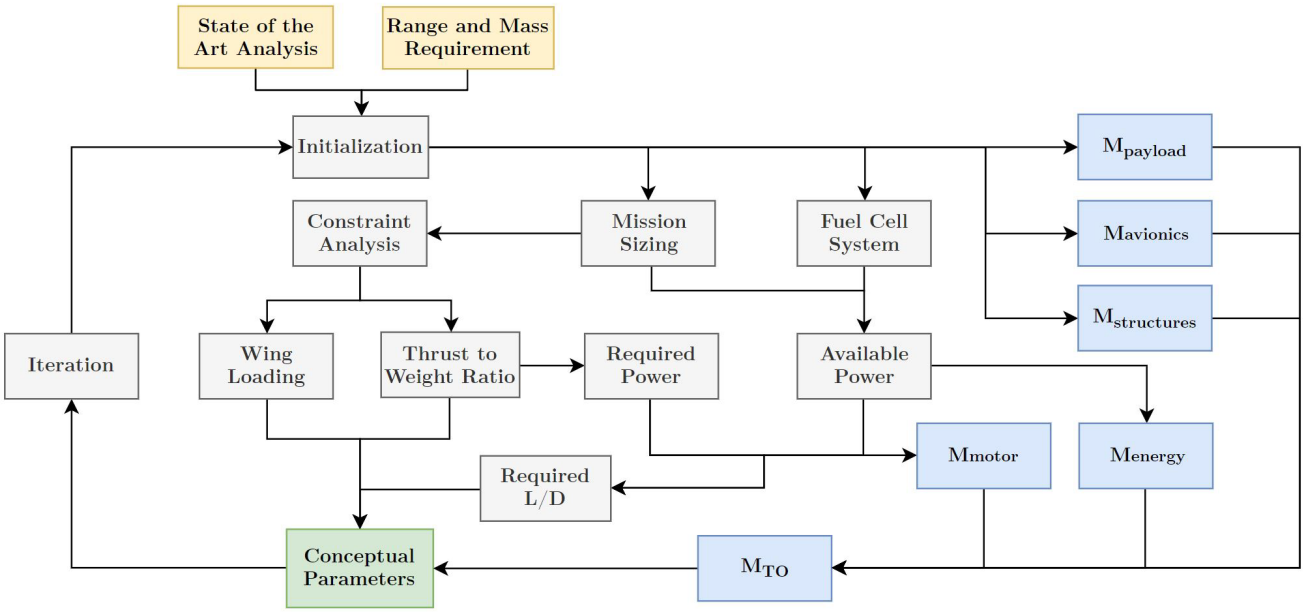


Figure 10: Conceptual Design Flow Chart.

pect ratios near 15. The Oswald efficiency factor is calculated for a trapezoidal wing with an aspect ratio of 15 without wing sweep.

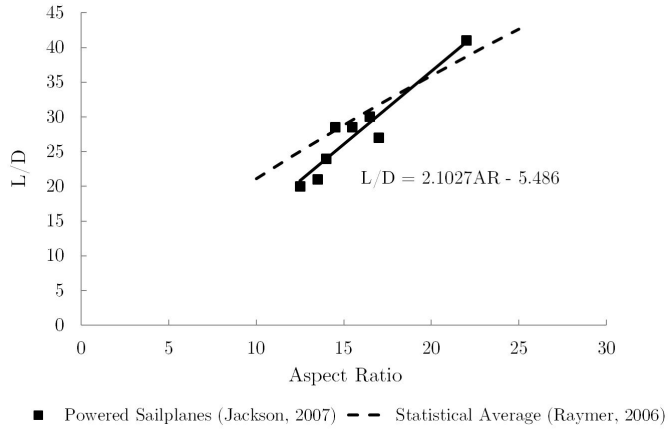


Figure 11: Statistical relationship between L/D and aspect ratio for sailplanes.

4.1.2 Initial Baseline Design Case

From the process of iterating the initialization variables, several feasible configurations are obtained. The cruise velocity of the aircraft is settled at 23 m/s at an operating altitude of 300 m and the maximum payload size is considered to be 1 kg as it results in an achievable L/D. Furthermore, from the state of the art and benchmarking hydrogen fuel cell

Parameter	Value
Aspect Ratio	15
Wing Sweep (°)	0
Dihedral Angle (°)	6
Taper Ratio	0.45
Parasite Drag Coefficient	0.0165
Oswald Efficiency Faactor	0.76

Table 5: Summary of Geometric Aircraft Parameters..

based UAS, a structural mass fraction of 0.38 has been set for the baseline aircraft and it is estimated that the in flight data transmission can be achieved with an avionics mass of 0.5 kg. However, a redundancy battery system will also be added to the avionics mass bringing the total to 1.22 kg. Gundlach [1] suggests that a depth of discharge, f_{DOD}, of 80% is the maximum allowable discharge before reducing the battery efficiency below 0.9. For a lithium-polymer battery, this efficiency is deemed reasonable from also consulting Gatti [8]. From Roskam [12], an energy reserve of 20% is considered sufficient to provide the extra power for the climb and descent segments as well as it serving as a reasonable safety margin to ensure that the mission's design range is achieved. The following Tables 6 and 7 summarize the input and state of the art parameters used for the initial design configuration.

Most notably though from the process, it is determined that the 250 W fuel cell is unable to provide a sufficient output power, whereas the 500 W system (Aerostak 500 from Table 3) is adequate. Therefore, the 1000 W system will lead to an

Parameter	Value	Unit
R	3 x 10^6	m
M	≤ 25	kg
$M_{payload}$	1	kg
$M_{avionics}$	1.2	kg
V_{cruise}	23	m/s
Rate of climb	1	m/s
Climb angle	3	$^\circ$

Table 6: Design Case Input Parameters.

Parameter	Value	Unit
MF_{struct}	0.38	–
M_{motor}/P_{shaft}	0.25	g/W
η_p	0.8	–
η_m	0.85	–
η_e	0.9	–
η_f	0.95	–
η_b	0.9	–
f_{DOD}	0.8	–

Table 7: State Of the Art Parameters.

unnecessary mass increase of the aircraft, resulting in the 500 W system being the most appropriate. With regards to the quantity of hydrogen cylinders (please note that we are considering the gas state of the hydrogen), only the two cylinder configuration is able to achieve the range requirement with a L/D below 30 while meeting the mass requirement, as shown in Table 8.

No. of Cylinders	MTOW (kg)	L/D_{req}
1	12.7	37.9
2	19.5	29.5
3	26.4	26.7

Table 8: Comparison between No. of Hydrogen Cylinder for a 500 W Fuel Cell System.

4.2 Configuration and Layout

The outputs of the configuration layout task are design drawings of the aircraft as well as geometric information required for further analysis in the preliminary design phase. For the aircraft, a T-tail configuration with a tractor propeller has been selected. The placement of the horizontal tail on top of the vertical fin provides increased leverage and assists with stability. Additionally, due to its higher placement, it will remain in undisturbed flow during a stall. As a consequence of the smaller required vertical tail, the T-tail can be lighter and due to the increased leverage, the horizontal tail can also be smaller. This reduces the friction drag, as well as there being less interference drag as a result of raising the horizontal tail. Due to the short nature of the fuselage and the low structural support of the introduced boom, a tractor propeller will be employed. Propellers mounted on the wings were also investigated, however, the idea was ultimately dismissed due to the increased complexity and weight increase, as two sets of smaller motors and propellers will be heavier than a single motor and propeller system (Gatti [8]). Shown in Figure 12 is the initial internal layout of the aircraft. Two detail views, A and B, are shown for the front and rear fuselage sections respectively. Each labeled component from Figure 12 is detailed in Table 9. Note that it is not an exhaustive drawing of all internal components, with components such as the avionics and additional battery system not being included. This schematic serves as an initial internal layout and will be refined in the preliminary design. The key finding from this drawing is that the hydrogen cylinders occupy majority of the internal space and that the fuselage may need to be extended in the preliminary design phase to embed all required internal components.

Component Description
1 Forward Hydrogen Cylinder - A12
2 Rear Hydrogen Cylinder - A12
3 Fuel Cell - Aerostak 500
4 Motor
5 Propeller and Hub
6 Beginning of Boom

Table 9: Conceptual Schematic Description.

4.3 Summary of Conceptual Design

The conceptual design of the transatlantic hydrogen powered UAS resulted in a 19.5 kg aircraft that with a lift to drag ratio of approximately 29.5 can feasibly cross the Atlantic Ocean. The wing, fuselage, empennage and propulsion system were appropriately sized to meet the mission objectives. Major aircraft and mission parameters are detailed in Tables 10, with wing and fuselage parameters shown in 11.

Parameter	Value	Unit
M_{TO}	19.5	kg
$M_{payload}$	1	kg
L/D_{req}	29.5	–
Altitude	300	m
V_{cruise}	23	m/s

Table 10: Aircraft and Mission Parameters.

The fuel cell and power plant characteristics are summarized in Table 12 and the empennage is detailed in Table 13, with an isometric drawing of the aircraft shown in Figure 13.

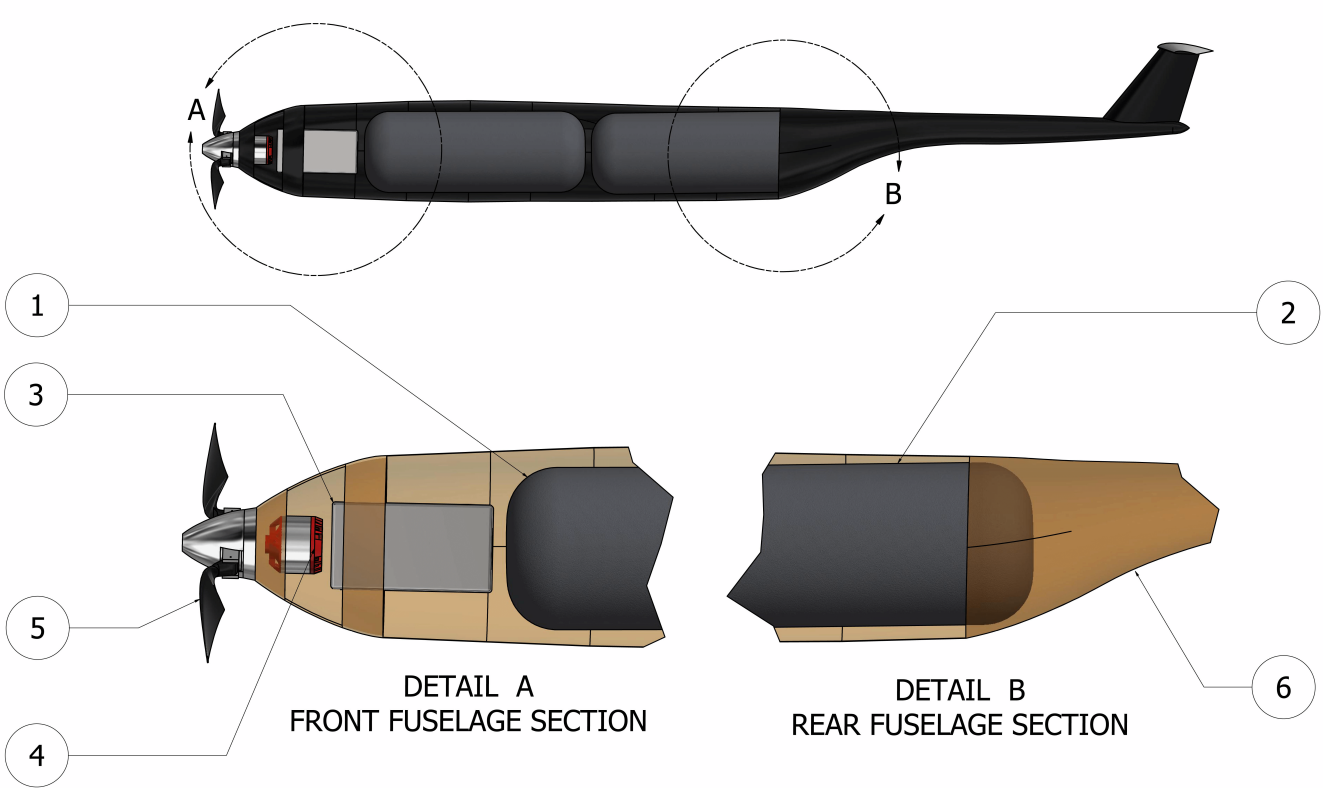


Figure 12: Internal Schematics of Conceptual Design.

Parameter	Value	Unit
S_{ref}	0.85	m^2
b_{ref}	3.6	m
AR	15	–
$L_{fuselage}$	2.3	m
$D_{fuselage}$	0.25	m

Table 11: Wing and Fuselage Design Parameters.

Parameter	Value	Unit
P_{fc}	500	W
E_{fc}	955	Wh/kg
T_O	9	N
$P_{inputcruise}$	274	W
$D_{propeller}$	0.375	m

Table 12: Fuel Cell and Power Plant Parameters.

5 Atmospheric Energy as a Potential for Performance Improvement

Various styles of flight could be noticed while bird watching. According to Scott and McFarland1 [13] birds use several strategies of energy harvesting, which serve as an inspiration for all the current improvements in the field of UAV long endurance performance. Interaction of wind and obstacles such as buildings, hills, or waves generates an ascending component of air motion. Many birds with knowledge of soaring techniques use these updrafts to power their flight instead of wing flapping.

In case of unequal heating of Earth's surface provoked, for example, by punctured cloud layer, implies uplift of hot air, known as thermal. Eagles, condors, vultures, and many other large birds use these updrafts with a technique called thermal soaring in order to extend their endurance while searching for a prey. Another example is sweeping flight within the gust pushed by the waves. Gulls and pelicans use these gusts to power their flight by flying along the wave cliffs. Gaining speed while wave slows down, they are able to pull up and glide to another wave where the process continues.

As there are many examples in nature, Albatrosses are particularly adept at exploiting wind gradients above sea level and can travel many thousands of kilometers using very little energy from flapping. Albatrosses and other birds that soar dynamically also have a skeletal structure that allows them to lock their wings when they are soaring, so the bird can continue flying almost indefinitely without having to put in much effort besides steering. In effect, it is harvesting energy

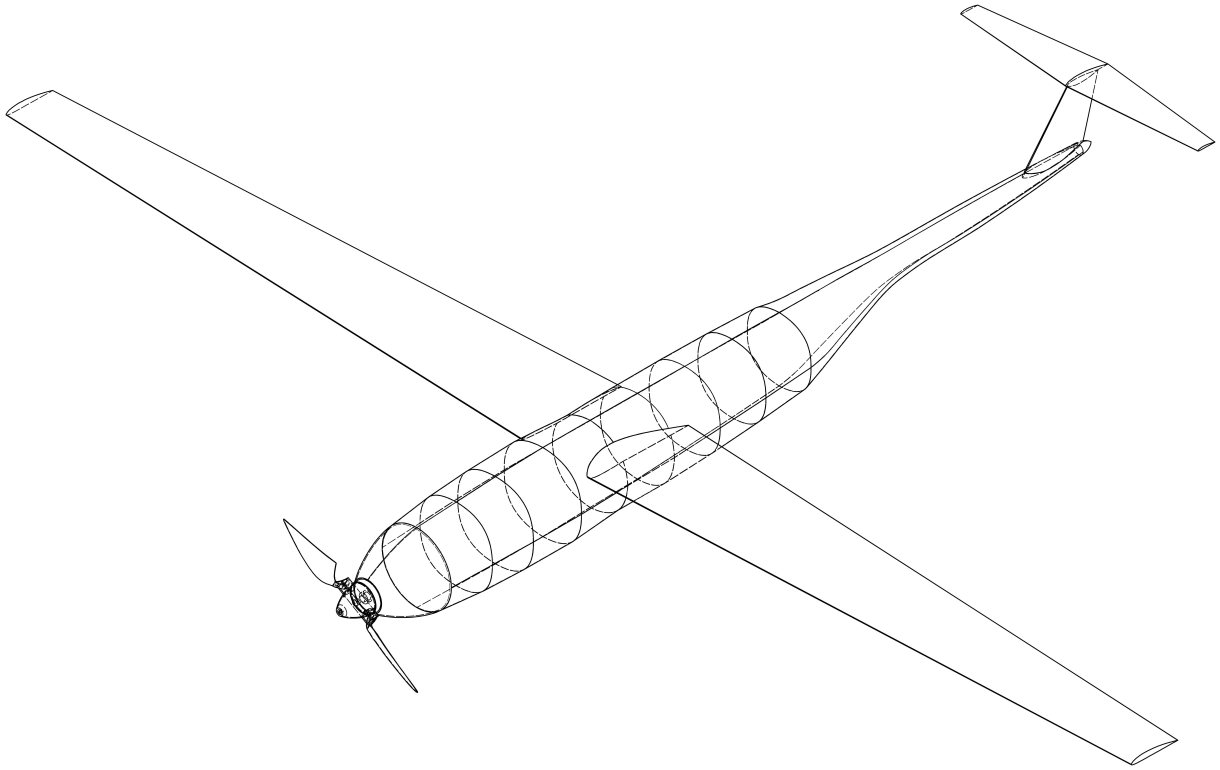


Figure 13: Isometric Drawing of Conceptual Transatlantic UAS.

Parameter	Value	Unit
S_{HT}	0.067	m^2
S_{VT}	0.039	m^2
AR_{HT}	8	–
AR_{VT}	1.6	–
$VT_{location}$	2.1	m

Table 13: Empennage Design Parameters.

from the wind gradient.

Those biologically inspired flight techniques would not be possible if the birds were not equipped with natural sensory systems for the detection of atmospheric phenomena. Severe turbulent flows will cause the feathers to vibrate and gyrate wildly. As the feathers are elevated by the air stream, mechanoreceptors increase their discharge frequency. However, it is clear that identical copies from nature to man-made technologies are still not feasible. It took millions of years for evolution to develop such extraordinary sensory systems and skills of natural yers. On the other hand, an imaginative inspiration and transformation into technology are often based on various steps of abstraction. Finally, it should also be pointed out that besides biologically inspired flight techniques, birds also serve as an inspiration for an extensive amount of extraordinary aerodynamic structures that are in service of performance improvement. An overview of aerodynamic structures for aircraft drag reduction inspired by wingtips of some natural fliers has been investigated by Gavrilovic et al [14].

5.1 Biologically Inspired Sensory Systems

5.1.1 Pressure Measurements on the Wing

The paper from Gavrilovic [15, 16] reveals a system for the local angle of attack estimation based on pressure measurements on the wing. The idea implies that a certain pair of pressure ports is located on the wing, as shown on Figure 8, where one port is on the upper surface of the chosen section, while the other is on the lower surface. Those points are recording a pressure difference with time which has to be normalized with dynamic pressure in order to enable the effectiveness of the system for various airspeeds. If a single location on the wing measures pressure difference then the estimation of the local angle of attack, including the influence of aileron deflection.

5.1.2 Multi-hole probes

These sensors consist of pressure-based multi-hole probes that can measure flow angles and airspeed in flight. A single tube provides real-time measurement of the local angle of attack and airspeed ahead of the wing. Therefore, several probes placed along wing-span can be used to determine the

gust length scale. This method of flow sensing has been previously demonstrated by Gavrilovic [17] If gust length scales were equal to or larger than the wingspan, then flight through such a large gust structure would result in only pitching and heaving motion. However, gusts smaller than the wingspan would provoke unequal load distribution along wingspan implying additional roll and yaw moments on the wing. Using multiple probes would ensure that aircraft reacts only within sufficiently large wind field.

5.1.3 Aerobooms

Aerobooms can be considered as custom-designed equipment for estimation of wind magnitude and direction, usually based on a single pitot tube which allows measurement of airspeed and a pair of wind vanes for the angle of attack and sideslip angle. The wind vanes are attached to magnetic encoders (acting as a potentiometer) which enable a digital signal to a controller. The advantage of using such equipment is the ease of manufacturing and relatively low cost when compared to sophisticated multi-hole probes or pressure surface systems. The comparison of the angle of attack estimated with Aeroboom shown in Figure 14 and angle from pressure measurements on the wing is investigated in the work of Gavrilovic [15].

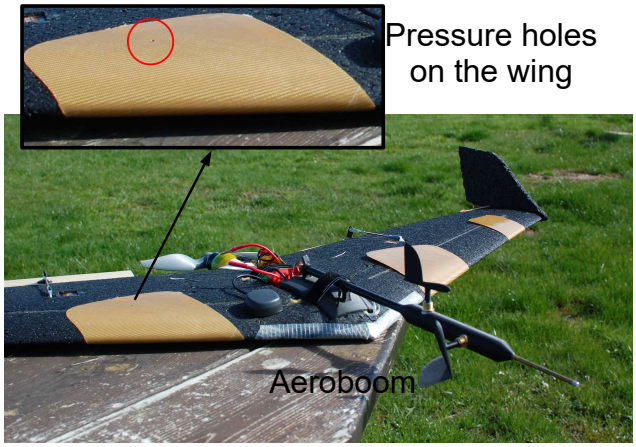


Figure 14: A small UAV equipped with wing pressure system and aeroboom.

5.2 *Energy-Harvesting from Atmospheric Phenomena*

The analysis that compared different flight strategies through a sinusoidal wind field showed that energy-harvesting flight represents the most favorable flight technique when compare to auto-stabilization or fixed-stick flight. It was also shown that energy-harvesting from wind fluctuations could potentially lead to a very significant savings in invested power that can even go up to 40 %, depending on the magnitude of wind field. Those results have been presented in the work by Gavrilovic [17]. Another flight technique which utilizes the energy of rising air has been demonstrated by Stroman and Edwards [18]. The previous work has demonstrated more than 100 km flight with a 4 m glider, all without a motor. On the other hand, a shorter cycle of climb within strong wind updraft and significant benefits in reduced invested power has been recorded in the work of Gavrilovic et al. [16]. Another very promising flight technique related to the exploitation of spatial wind gradients is dynamic soaring. This flight technique is usually related to the flight of Wandering Albatross as shown in Figure 15, which can cross enormous distances while being fueled by only couple of grains of food. A dynamic soaring mechanism in the ocean boundary layer has been previously presented by Bonnin [19]. A demonstration of gain in potential energy while exploiting a horizontal wind gradient with reduced power of a small unmanned aerial vehicle has been previously demonstrated by Gavrilovic et al. [16, 17]. Finally, it can be concluded that energy-harvesting represents an opportunity to significantly enhance the performance of a small unmanned aerial vehicle, through extended endurance and range. With equipment for detection of flow magnitude and the direction, the aircraft would be able to apply adequate maneuvers for increasing its energy state.

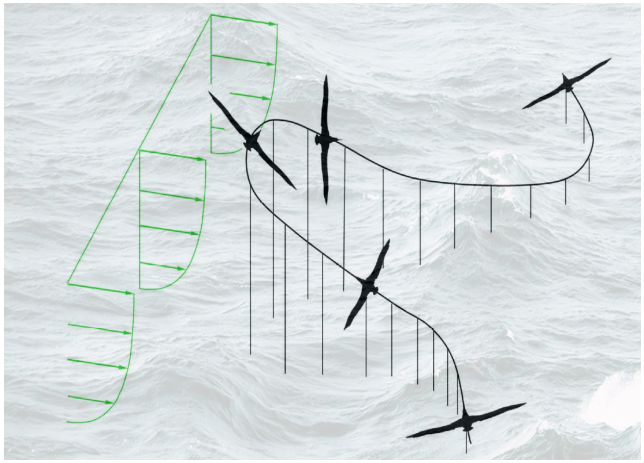


Figure 15: Albatross neutral energy cycle with dynamic soaring.

6 Conclusion

The conceptual design phase was conducted using a classical approach, but modified appropriately for a hydrogen fuel cell powered UAS. It involved taking the inputs from the mission objectives and iteratively outputting the aircraft mass, required L/D, wing loading and thrust to weight ratio until a baseline configuration that can feasibly cross the Atlantic Ocean was achieved. The initial analysis has also proved

that crossing a distance of 3000 km would not be possible with only Li-Po batteries. On the other side, a significant rise in specific energy density provided by fuel cells ensures that crossing the Atlantic Ocean is possible. The conceptual design phase resulted in an aircraft with a maximum take off mass of around 20 kg and cruise velocity of 23 m/s that is capable of carrying a 1 kg payload. The reference wing area of 0.85 m^2 and wing span of 3.6 m are sized according to a wing loading that met all operational constraints. Furthermore, the 500 W fuel cell unit and 274 W cruise motor power are sized from the thrust to weight ratio obtained from the constraint analysis. Future work for this project includes transitioning into the preliminary design phase, where the conceptual design will be refined. An optimal combination of airfoil and horizontal tail position are to be determined as well as an in-depth sizing of the heavily coupled motor and propeller. Further modifications to the aircraft will also be investigated such as incorporating geometric twist to the wings and adding wingtip devices to minimize the induced drag. These modifications are to be studied by using Vortex Lattice Method (VLM) and Computational Fluid Dynamics (CFD) programs. This work has also presented some of the bio-inspired systems for wind speed and direction measurements. Besides their primary function, those devices can also be used as stall-recovery systems, as they can locally estimate the angle of attack on the wing. Moreover, those bio-inspired systems can be also applied as a way of sensing atmospheric phenomena that can be exploited by an aircraft. In such a way the aircraft performance can significantly be enhanced by utilizing the energy of the atmosphere.

ACKNOWLEDGEMENTS

The authors are grateful to Mehmet Kesmez from HES Energy Systems for all the useful information related to the latest technology of hydrogen fuel cells.

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] EASA. Introduction of a regulatory framework for the operation of unmanned aircraft., 2015. European Aviation Safety Authority.

[7] Insitu. Scan eagle capabilities, 2018. https://www.insitu.com.

[8] M. Gatti. Complete preliminary design methodology for electric multirotor. *Journal of Aerospace Engineering*, 30(5):1–15, 2017.

[9] HES Energy Systems. Aerostak ultra-light fuel cells, 2019. https://www.hes.sg/aerostak.

[10] HES Energy Systems. Series-a tanks, 2019. https://www.hes.sg/accessories.

[11] D. Raymer. *Aircraft Design: A Conceptual Approach*. American Institute of Aeronautics and Astronautics, Reston, VA, 2006.

[12] J. Roskam. *Airplane Design Part I: Preliminary Sizing of Airplanes*. DARcorporation., Lawerence, KA, 2004.

[13] C. Scott and C. McFarland. *Bird feathers - a guide to North American species*. Mechanicsburg: Stackpole Books,, 2010.

[14] N. Gavrilovic, B. Rasuo, G. Dulikravich, and V. Parezanovic. Commercial aircraft performance improvement using winglets. *FME Transactions*, 43(1):1–8, 2015.

[15] N. Gavrilovic, M. Bronz, and E. Benard J-M. Moschetta. Bioinspired wind field estimation-part i: Angle of attack measurements through. *International Journal of Micro Air Vehicles*, 10(3):273–284, 2018.

[16] N. Gavrilovic, M. Bronz, and E. Benard J-M. Moschetta. Bioinspired energy harvesting from atmospheric phenomena for small unmanned aerial vehicles. In *AIAA SciTech Forum, San Diego, CA.*, 2019.

[17] 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):1–20, 2019.

[18] R. Stroman and D. Edwards. The hybrid tiger: A long endurance solar/fuel cell/soaring unmanned aerial vehicle. In *DOE Annual Merit Review, US Naval Research Laboratory, Washington DC*.

[19] V. Bonnin, E. Benard, C. Toomer, and J-M. Moschetta. Dynamic soaring mechanism in the ocean boundary layer. *Int. J. Engineering Systems, Modelling and Simulation*, 8(2):136–148, 2016.

Autonomous Navigation in Dynamic Environments using Monocular Vision

Liang Lu*, Javier Rodriguez-Vazquez, Adrian Carrio and Pascual Campoy
Universidad Politecnica de Madrid, Calle Jose Gutirrez Abascal 2, Madrid (Spain)

ABSTRACT

Autonomous navigation in dynamic unknown environments is a key research topic in robotics and has gained a lot of attention from the research community in the last years. In this paper, we propose a strategy for autonomous navigation in an environment with spheric obstacles. In this strategy, we use the YOLOv3 object detection model to detect the obstacles and an Iterative Perspective-n-Point (PnP) algorithm to estimate the center of the obstacle based on the result from the detector. Then using the obstacles' positions, a Receding Horizon Control (RHC) based planner is used to plan a trajectory using Covariant Hamiltonian Optimization (CHO) function, and a trajectory controller controller based on Model Predictive Control (MPC) is used to fly the planned trajectory. The proposed navigation strategy is evaluated with Rotors Gazebo simulator in a dynamic environment. Experimental results show that our autonomous navigation strategy is a valid approach for Unmanned Aerial Vehicle (UAV) navigation in dynamic environments.

1 INTRODUCTION

Navigation in dynamic environments is a key challenge in the area of autonomous navigation. It is still an unsolved problem because of the requirements of fast perception, planning and control algorithms when robots operating in a dynamic environment. During the last years, this problem has attracted a lot of researchers' attention and several methods have been proposed [1, 2, 3]. Among these methods, the following ones have gained a lot of interest within the research community, Artificial Potential Field (APF), geometry-based approach, Velocity Obstacle (VO), Partially Observable Markov Decision Process (POMDP), learning-based method and sampling-based strategy.

Our method can be thought of as a kind of geometry-based strategy because we describe the obstacles using geometric models (spheric obstacle in this case). In our method, first of all, we build a dataset of the obstacles we want to avoid and train the tiny version of the YOLOv3 model [4] to detect them. Next, the 3D positions of the centers of the obstaces will be computed using an Iterative PnP algorithm. Then, an online trajectory planner uses an RHC framework will be applied to plan a path. Finally, an MPC trajectory controller is employed to fly the planned trajectory.

The remainder of the paper is organized as follows. Section II presents problem formulation. In Section III, we introduce the proposed methodology. We show the experimental results and discussion in Section VI. And finally, Section IV concludes the paper and summarizes future research directions.

2 PROBLEM FORMULATION

2.1 Robot Model Assumption

A multirotor UAV has been used in this research. The multirotor UAV has 6 Degrees of Freedom (DoF), 3 DoF in translation and 3 DoF in rotation, and can fly freely in the 3-dimensional environment. The on-board sensors of the UAV are a front RGB camera and an Inertial Measurement Unit (IMU) sensor. The front RGB camera is used to detect obstacles within its the Field of View (FOV) and the IMU sensor is used for estimating the pose of the UAV. In this paper, the UAV will be modelled as a sphere that fully contains the UAV.

2.2 Environment Model Assumption

The operating environment in this paper is an environment with dynamic obstacles, modelled as spheres. The size of the obstacles is given but their positions are unknown. The initial point P_{init} and goal point P_g are given and the robot should move from P_{init} to P_g without collision. The robot will be thought as reaching P_g when the distance from the center of the robot to P_g is smaller than the threshold L_G. The environment model which is used in the paper is shown in Figure.1.

3 METHODOLOGY

The description of the proposed architecture for autonomous navigation is shown in Figure.2. There are 3 main parts in it, which are **robot localization** , **obstacles detecting and pose estimation** and **online RHC trajectory planner and controller**. The part of the robot localization is based on our previous work [5].

3.1 Obstacles Detection and Pose Estimation

In this paper, we train tiny YOLOv3 to detect the obstacles. It's an object detector that uses features learned by a deep convolutional neural network to detect an object. Given

*Email address(es): liang.lu@upm.es

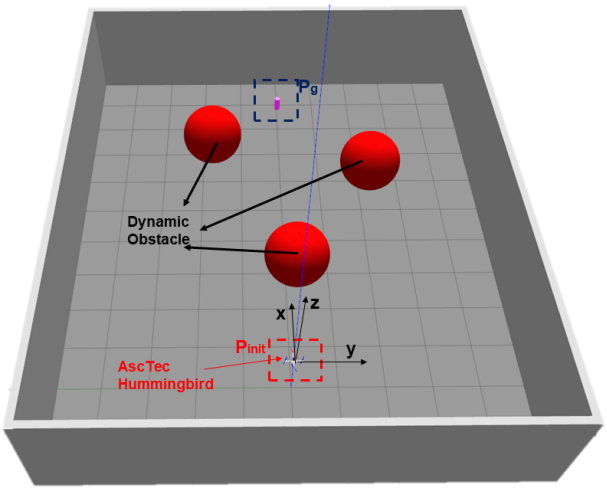


Figure 1: The environment model with dynamic obstacles.

the controlled experimentation conditions of this work, a very naive obstacle detection method (for example thresholding by color and circle detection) based on classical computer vision can be designed, but this could not work on future experimentation that includes real world flights. The reason of using YOLO instead of an easier method is to take un account the noise that would be introduced by the detector in the real world. The detector trained by tiny YOLOv3 has a very high frame rate and can be run real-time with a Graphics Processing Unit (GPU). The average processing time is 15ms using a GeForce GTX 1050Ti GPU. The output of the detector are the bounding boxes which enclose the detected obstacles. The dataset used to train the network has been derived from OpenImages V4.

In order to calculate the pose of the obstacles with sufficient accuracy, the bounding boxes should enclose the obstacle completely. And because the obstacles in our paper are spheric obstacles, the bounding boxes should have a square shape. As it can be seen from Algorithm 1, if the quotient between the bounding box's width w and height h is within a predefined range of $[\sigma_0, \sigma_1]$, the bounding box will be considered valid and used to compute the center of the obstacle.

With selected bounding boxes, we use an Iterative PnP algorithm [6] to compute the 3-dimensional position of the center of the obstacles. The average processing time of the iterative PnP is 14 ms when we run it a laptop with an Intel Core i7-8750H CPU..

3.2 Online RHC trajectory planner and controller

The RHC framework based trajectory planner is used in order to find the best trajectory from a trajectory library. In this paper, first, we use some ideas from Andreas Bircher *et al* [7] to generate online goal candidates and some ideas from Zheng Fang *et al* [8] to build the CHO objective function. Then the CHO objective function is used to generate a trajectory library. The initial point and goal point of the path are

Algorithm 1 Bounding Boxes Filter

Input: $Bounding\ Boxes\ From\ Detectors$
Output: $Bounding\ Boxes\ Selected$
1: $B_0 \leftarrow Bounding\ Boxes\ From\ Detectors$;
2: **for** $Bounding\ Boxes\ B\ in\ B_0$ **do**
3: **if** $\sigma_0 < B.width/B.height < \sigma_1$ **then**
4: $Bounding\ Boxes\ Selected \leftarrow B$;
5: **end if**
6: **end for**

the current robot position and the generated goal candidates, respectively. Finally, the trajectory with the lowest objective function value is selected as the best trajectory.

The Rapid Random Tree (RRT) framework [9] is used to generate the goal candidate nodes. As it can be seen from algorithm 1, first we set the number of maximum goal candidate nodes N_{max}, next a random node is generated in the predefined goal candidates searching area V_S, then we find a goal candidate by $Nearest$ and $Steer$ function from the RRT framework and store this goal candidate. The $Nearest$ function is responsible for searching for $P_{nearest}$ and $Steer$ function is used for generating P_{new}, detail information about these two functions can be found from [9]. At last, if the number of goal candidates is larger than N_{max}, the online goal candidates generation process is finished.

A modified objective function for CHO is built to create the trajectory library and search for the best trajectory. Our objective function measures four different aspects of the trajectory planning problem. First, in order to get a smooth path, we add a penalization based on dynamical criteria, like velocities and accelerations to the trajectory. Next, we penalize the trajectory by the distance from the trajectory waypoint to the objects to make the trajectory avoid obstacles. Then, the end of the trajectory is penalized by the distance from it to the final goal, which can help the trajectory planner plan a trajectory close to the final goal ξ_g. Finally, we penalize the trajectory by the distance from its waypoint to the ground to make the trajectory go far away from the ground. We describe these four items by f_s, f_o, f_g, f_a respectively, and define our objective function by summing their weights:

$$f(\xi) = w_1 f_s(\xi) + w_2 f_o(\xi) + w_3 f_g(\xi(1)) + w_4 f_a(\xi) \quad (1)$$

As described above, the trajectory is ξ and $\xi(s)$ is the function mapping the trajectory length s to the robot configurations, the initial and end configurations of the trajectory ξ is $\xi(0)$ and $\xi(1)$ respectively. The waypoints in the trajectory ξ are 3 DoF point $\{x, y, z\}$. w_1, w_2, w_3 and w_4 are the weights for each objective functions.

The objective functions of f_s, f_o, f_g are the same from [8]:

$$f_s(\xi) = \int_0^1 c_o(\xi(s)) \| \frac{\mathrm{d}}{\mathrm{d}t} \xi(s) \| \mathrm{d}s \quad (2)$$

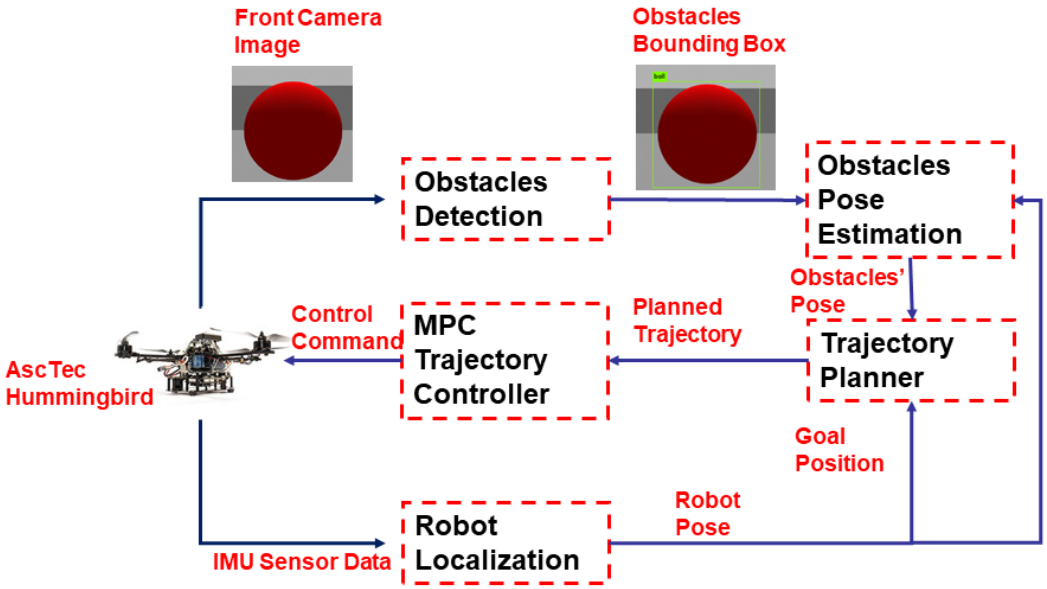


Figure 2: Architecture of the autonomous navigation method proposed.

$$f_o(\xi) = \frac{1}{2}\int_0^1 \|\frac{\mathrm{d}}{\mathrm{d}t}\xi(s)\|^2 \mathrm{d}s \tag{3}$$

$$f_g(\xi(1)) = \|\xi(1) - \xi_g\| \tag{4}$$

$c_o(\xi(s))$ in the objective function is the obstacle cost for the spheric obstacle obstacle:

$$c_o(\xi(s)) = \frac{w_0 R_o}{3}(1 - \frac{Dist(\xi(s))}{R_o})^3 \tag{5}$$

w_0 is the weight and R_o is the radius of the spheric obstacle obstacle. $Dist(\xi(s))$ is the distance from the waypoints in the trajectory to the center of the spheric obstacle.

The objective function of f_a is learned from [10]:

$$f_s(\xi) = \int_0^1 c_a(\xi(s))\|\frac{\mathrm{d}}{\mathrm{d}t}\xi(s)\|\mathrm{d}s \tag{6}$$

$c_a(\xi(s))$ in the objective function is the altitude cost:

$$c_a(\xi(s)) = \begin{cases} (Alt(\xi(s)) - \epsilon)^2, & Alt(\xi(s)) \leq \epsilon \\ 0, & Alt(\xi(s)) > \epsilon \end{cases} \tag{7}$$

$Alt(\xi(s))$ is the altitude of the waypoints of the trajectory ξ and ϵ is predefined the minimum value of the altitude.

The path library is generated by using the objective function $f(\xi)$ and the best trajectory is the trajectory with the lowest objective function value and optimized by minimizing the objective function $f(\xi)$.

After obtaining the trajectory, an MPC based trajectory controller which is similar to [11] is used to follow the robot to correctly follow the planned trajectory.

While the robot is flying along the trajectory, we will check the status of several trajectory points in front of the robot. If the distance from one of the trajectory points to the center of the obstacles is smaller than the obstacles' radius, the trajectory planner will search for a new trajectory to fly.

Algorithm 2 Online Goal Candidates Generate

Input: *Current Robot Position*
Output: *Goal Candidate Nodes*

$P_0 \leftarrow Current\ Robot\ Position$;
$T \leftarrow P_0$;
$N_T \leftarrow 0$;
while $N_T < N_{max}$ **do**
  $P_{rand} \leftarrow SampleFree(V_S)$;
  $P_{nearest} \leftarrow Nearest(T, P_{rand})$;
  $(P_{new}, T_{new}) \leftarrow Steer(P_{nearest}, P_{rand})$;
  $Put\ P_{new}\ in\ Goal\ Candidate\ Nodes$
  $N_T \leftarrow N_T + 1$;
end while

4 EXPERIMENTAL RESULTS AND DISCUSSIONS

4.1 Experimental Setup

RotorS Gazebo simulation environment and Robot Operating System (ROS) have been used under Ubuntu 18.04. The Rviz/Gazebo environment can use real physical param-

eters of the robot and environment. All the experiments run on a laptop with Intel Core i78750H at 2.2GHz, a GeForce GTX 1050Ti GPU. The simulation of the proposed navigation method is integrated into our open source framework Aerostack[1]. The used environments are 3D indoor environments. The UAV in the simulation is the AscTec Hummingbird, which is equipped with a front RGB camera. The front camera takes charge of receiving information from the environment in which the robot operates. The UAV can fly freely in the 3-dimensional environment.

4.2 Evaluation of Obstacle Pose Estimation

We build the environment which is shown in Figure 3 to evaluate the performance of our obstacle pose estimation algorithm. The size of the environment is $10\,m \times 12\,m$. The multirotor UAV hovers at the point $(0, 0, 1.1)\,m$, the proposed approach for obstacle detection and pose estimation strategy are used to calculate the center of the obstacles by using the images captured from the UAV's front camera. We test our obstacle pose estimation strategy with two different spheric obstacles. The radius of these spheric obstacles is $1\,m$ and $1.5\,m$, respectively. The position of obstacle will generate randomly in the obstacle area. The obstacle area is a cube area, the center of the area is $(5, 0, 1.125)\,m$. The length (L), width (W) and height (H) of the obstacle area is $6\,m$, $6\,m$ and $1.5\,m$ respectively. We run our algorithm 1000 times for the

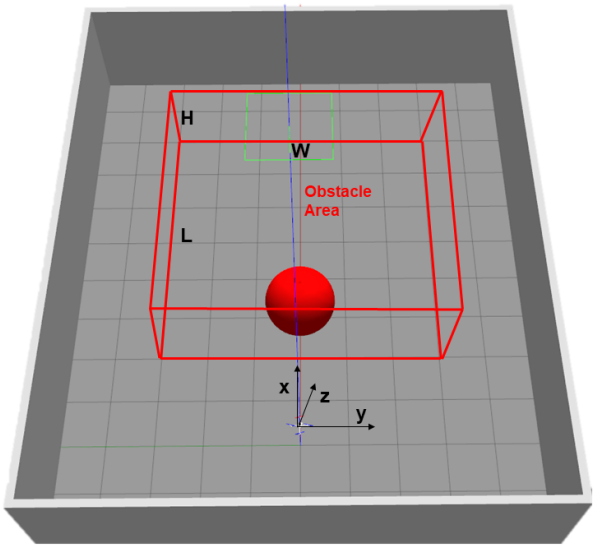


Figure 3: The environment used to evaluate the performance of obstacle pose estimation algorithm.

spheric obstacles mentioned above respectively, and compute the error which is the euclidean distance of the estimated obstacle center position and Gazebo ground truth. Then, Max Error (MaxE), Min Error (MinE), Mean Error (ME) and Root Mean Square Error (RMSE) is calculated and can be seen from Table 1

	obstacle radius = 1m	obstacle radius = 1.5m
MaxE (m)	1.9534	3.3951
MinE (m)	0.0105	0.0135
ME (m)	0.3061	0.2981
RMSE (m)	0.2315	0.3114

Table 1: Results of obstacle pose estimation.

4.3 Experiments of Autonomous Navigation

We use the environment shown in Figure 1 to test the performance of our autonomous navigation system. In the environment, there are 3 dynamic obstacles which have a sinusoidal trajectory. The initial point of the robot is $(0, 0, 0)\,m$. The coordinate x, y and z are randomly generated between $[8, 8.5]\,m$, $[-3, 3]\,m$ and $[0.75, 1.5]\,m$, respectively. We run our autonomous navigation algorithm 50 times in the environment. Table 2 shows the results after running the algorithm 50 times. In the table, the Successful Rate (SR), Path Length (PL), Time to the Goal (TG), Maximum Velocity (MaxV) and Mean Velocity (MeanV) express the performance of successful flight from the initial point to the goal point, the mean length of the path, the average time to reach the goal, the average maximum velocity, and the average velocity for the 50 runs.

SR (%)	82
PL (m)	9.8848
TG (s)	28
MaxV (m/s)	0.7012
MeanV (m/s)	0.3223

Table 2: Results after 50 runs of our navigation algorithm in the dynamic environment.

Figure 4 shows 3 flying trajectories from the 50 runs. The figure of top left, bottom left and bottom right correspond respectively to the top view, left view and normal view of the 3 flying trajectories in the test environment. In the figure, the red point is the initial point which is $(0, 0, 1.1)\,m$, the yellow, green and blue points are the goal points for different trajectories and the purple, brown and blue line lines correspond to the 3 different trajectories. The red spheric obstacles are the dynamic obstacles which move with a sinusoidal trajectory.

A video description of these flights for the 3 trajectories can be seen from https://vimeo.com/347564788.

4.4 Discussions

As it can be seen from Table 1, there are some errors in the obstacle pose estimation. However, the SR in Table 2

[1]www.aerostack.org

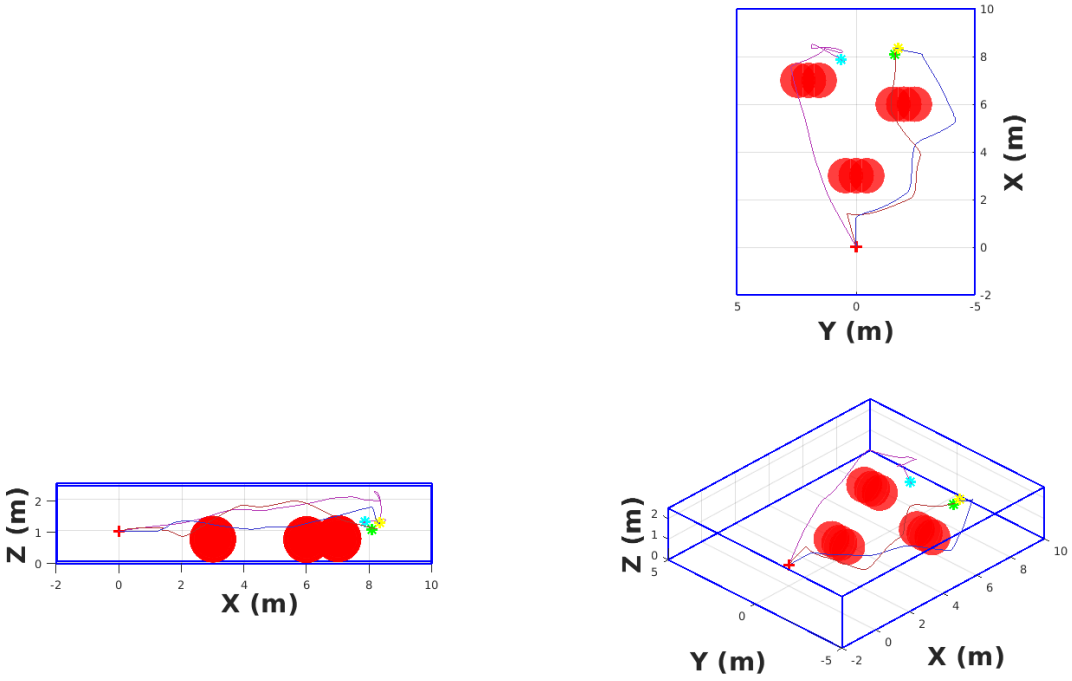


Figure 4: Three views of three flight trajectories.

shows that it is still valid for our proposed trajectory planner. The SR in Table 2 also shows that our navigation strategy can be used for navigation in an environment with dynamic spheric obstacles with a maximum velocity of $0.7012\,m/s$ and an average velocity of $0.3223\,m/s$.

In this paper, we build a Gazebo simulation environment and use Rotors Gazebo simulator to evaluate our algorithm, Rotors Gazebo model incorporates a good dynamic model for the aerial robot which can provide realistic flight behaviors and it is widely used in the research community [12, 13].

5 CONCLUSIONS AND FUTURE WORKS

In this paper, a method for autonomous UAV navigation in an environment with dynamic obstacles has been presented. An obstacle detector based on YOLOv3 and an iterative PnP algorithm are used to estimate the relative position of the obstacles. Then, an online RHC trajectory planner is used to plan a path and finally, the robot is controlled using an MPC controller in order to guide it to the goal waypoint. The experiment results show that this is a valid approach for UAV navigation in dynamic environments.

In the future, we will improve the performance of our detection and pose estimation algorithm to reduce the errors in obstacle pose estimation. An Extended Kalman filter or Unscented Kalman Filter will be also used to predict the future position of the obstacle and used in our trajectory planner to improve its performance. A real flight will also be implemented to evaluate the performance of the proposed algorithm.

Acknowledgements

The work reported in this paper is sponsored by the Chinese Scholarship Council (CSC).

References

[1] C. Sampedro, H. Bavle, A. Rodriguez-Ramos, P. de la Puente and P. Campoy, "Laser-Based Reactive Navigation for Multirotor Aerial Robots using Deep Reinforcement Learning," Intelligent Robots and Systems (IROS), *2018 IEEE/RSJ International Conference in.* IEEE, 2018, pp.1024-1031.

[2] C. Sampedro, H. Bavle, A. Rodriguez-Ramos, A. Carrio, R. A. Suarez-Fernandez, J. L. Sanchez-Lopez and P. Campoy, "A fully-autonomous aerial robotic solution for the 2016 International Micro Air Vehicle competition," Unmanned Aircraft Systems (ICUAS), *2017 IEEE International Conference in.* IEEE, 2017, pp.989-998.

[3] A. Carrio, C. Sampedro, A. Rodriguez-Ramos and P. Campoy, "A review of deep learning methods and applications for unmanned aerial vehicles," Journal of Sensors, 2017: 1-13.

[4] J. Redmon, A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv: 1804.02767, 2018.

[5] H. Bavle, J. L. Sanchez-Lopez, P. de la Puente, A. Rodriguez-Ramos, C. Sampedro and P. Campoy "Fast and Robust Flight Altitude Estimation of Multirotor UAVs in Dynamic Unstructured Environments Using 3D Point Cloud Sensors," aerospace, 2018, 5(3): 1-21.

[6] A. Rodriguez-Ramos, C. Sampedro, A. Carrio, H. Bavle, R. A. Suarez-Fernandez, Z. Milosevic and P. Pascual, "A Monocular Pose Estimation Strategy for UAV Autonomous Navigation in GNSS-denied Environments," Proceedings of the International Micro Air Vehicle Conference and Flight Competition, 2016, pp.17-22.

[7] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova and R. Siegwart, "Receding Horizon NextBestView Planner for 3D Exploration," on Robotics and Automation (ICRA), *2016 IEEE International Conference.* IEEE, 2016, pp.1462-1468.

[8] Z. Fang, S. Yang, S. Jian, G. Dubey, S. Roth, S. Maeta, S. Nuske, Y. Zhang and S. Scherr, "Robust Autonomous Flight in Constrained and Visually Degraded Shipboard Environments," Journal of Field Robotics, 2017, 34(1): 25-52.

[9] S. Karaman and E. Frazzli, "Sampling based algorithms for optimal motion planning," *International Journal of Robotics Research*, 30(7): 846-894, 2011.

[10] M. Zucker, N. Ratliff, A.D. Dragan, M. Pivtoraiko,M. Klingensmith, C.M. Dellin, J.A. Bagnell, and S.S.Srinivasa, " CHOMP: Covariant hamiltonian optimizationfor motion planning," *International Journal of Robotics Research*, 2012.

[11] M. Kamel, M. Burri and R. Siegwart, "Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles," *arXiv preprint arXiv: 1611.09240*, 2016.

[12] T. Cieslewski, E. Kaufmann and D. Scaramuzza, "Rapid exploration with multi-rotors: A frontier selection method for high speed flight," Intelligent Robots and Systems (IROS), *2017 IEEE/RSJ International Conference in.*, IEEE, 2017, pp. 2135-2142.

[13] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart and E. Galceran, "Continuous-time trajectory optimization for online UAV replanning," Intelligent Robots and Systems (IROS), *2016 IEEE/RSJ International Conference in.*, IEEE, 2016, pp. 5332-5339.

Detection of nearby UAVs using CNN and Spectrograms

Aldrich A. Cabrera-Ponce *[1], J. Martinez-Carranza[1,2], and Caleb Rascon[3]

[1]Instituto Nacional de Astrofisica, Optica y Electronica , Puebla, Mexico
[2]University of Bristol, Bristol, UK
[3]Universidad Nacional Autonoma de Mexico, Ciudad de Mexico, Mexico

Abstract

In this work, we address the problem of drone detection flying nearby another UAV. Usually, computer vision could be used to face this problem by placing cameras on board the patrolling UAV. However, visual processing is prone to false positives, sensible to light conditions and potentially slow if the image resolution is high. Thus, we propose to carry out the detection by using an array of microphones mounted with a special array on board the patrolling UAV. To achieve our goal, we convert audio signals into spectrograms and used them in combination with a CNN architecture that has been trained to learn when a UAV is flying nearby and when it is not. Clearly, the first challenge is the presence of ego-noise derived from the patrolling drone itself through its propellers and motor's noise. Our proposed CNN is based on the Google's Inception v.3 network. The Inception model is trained with a dataset created by us, which includes examples of when an intruder drone flies nearby and when it does not. We tested our approach with three different drone platforms, achieving a successful detection of 97.93% for when an intruder drone flies by and 82.28% for when it does not. The dataset used for this work will be available as well as the code.

1 Introduction

Recently, the autonomous drones have grown in popularity in aerial robotics since they are vehicles with multiples capabilities, with the help of on-board sensors such as Inertial Measurement Unit (IMU), laser, ultrasonics, and cameras (both monocular and stereo). Visual sensors can be used to generate maps, for 3D re-construction, autonomous navigation, search and rescue, and security applications. However, these applications face serious problems when attempting to identify another drone in circumstances where the visual range is lacking, which can cause collisions, putting bystanders at risk in public places. Thus, it is necessary to have strategies that employ other modalities other than vision to ensure the discovery of an intruder drone. One such modality can be audio.

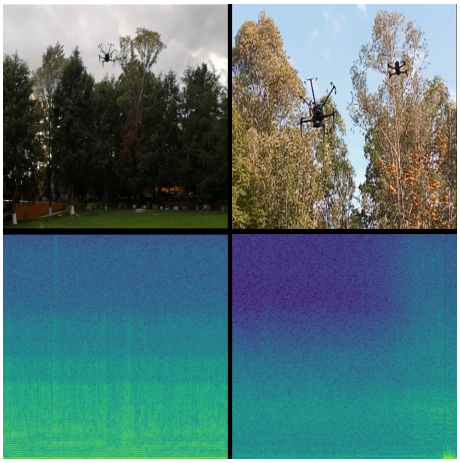


Figure 1: Classification of audio in two different environments. Left: spectrogram of an intruder aerial vehicle nearby. Right: spectrogram without an intruder vehicle nearby. `https://youtu.be/B32_uYbL62Y`

Audio processing has been a topic of research for years, which includes the challenge of recognising the source of an audio signal. In aerial robotics, the signals usually tend to present noise that disturbs the original signal, making the recognition an even more difficult task. However, if this is successful, it can be used to find the relative direction of a sound source (such as another drone) as well as identify other sounds in different distance ranges. A useful manner with which audio is represented in this type of applications is in the time-frequency domain, in which the spectrogram of the signal is manipulated as if it were an image. These images allow a detailed inspection of the noise of the rotors to analyse vibration and prevent future failures in the motors. By identifying features inside the spectrogram, sound source identification and localisation may be possible over a drone.

Recent works employ deep learning strategies (such as Convolutional Neural Networks, CNN) to classify sound sources, and many of these methods aim to learn features from a spectrogram. We propose to use a CNN to identify when there is or may not be an aerial vehicle near our drone from a given input spectrogram (See Fig.1).

We base our CNN-based classification model on the

*Department of Computer Science at INAOE. Email addresses: {aldrichcabrera, carranza}@inaoep.mx

Google's Inception v.3 architecture. The information is separated in two different classes: with and without a drone. Each class has 3000 spectrograms for training. Each spectrogram is manipulated as though it is an image, with each pixel representing a time-frequency bin, and its colour representing its energy magnitude. Moreover, our approach aims to classify with a high level of performance over different aerial platforms.

This paper is organised as follows: Section 2 provides related works which identify sources in the environment with aerial vehicles; Section 3 describes the hardware used; Section 4 provides a detailed description of the proposed approach; Section 5 describes the analysis of the spectrograms for each class; Section 6 presents the classification results using the proposed approach; and conclusions and future work are outlined in Section 7.

2 RELATED WORK

As mentioned earlier, drones that solely employ vision may be limited when identifying aerial vehicles in an environment near a flight zone. Thus, works with radars have used the micro-doppler effect to identify a target [1] or different targets [2]. This use, as the basis for classification, the change of the audible frequency due to changes in the velocity of the propellers [3,4], as well as other features [5]. Additionally, when this effect is represented by its cadence frequency (CFS), it can be used to recognise other descriptors like shape and size, achieving the classification of multiple classes of aerial vehicles [6].

As for audio processing techniques, they have been used in aerial robotics for classification, detection, and analysis of rotors, to analyse annoyances generated by the UAV's noise through psycho-acoustic and metrics of noise [7]. Likewise, they have been used for the localisation of sound sources [8], reducing the effect ego-noise of the UAV's rotors and localise the source in high noise conditions in outdoor environments [9]. These auditory systems have been used in conjunction with radars and acoustic sensors, showing good performances when identifying UAVs in public places [10] and detecting sound sources in spaces of interest [11]. Even though these alternatives have been developed, the audio processing area of research over a drone is a challenging task that still has considerable room to develop.

On the other hand, good acoustic identification using harmonic spectrums can help avoid collisions between two fixed-wing aircraft by increasing the detection range of an intruder UAV to 678 meters [12]. This localisation range can be further improved by 50% (while reducing computational costs) by using harmonic signals [13]. In [14], a design for positioning an 8-microphone array over a drone is presented, aimed to detect distinct nearby UAVs from a given drone. This design is useful for detection, localisation, and tracking intruder drones operating close to undesired areas such as airports or public environments.

There are several strategies that employ deep learning for sound classification. For example, the direction of a sound source was estimated using spherical bodies around a drone and microphones on-board in [15]. Furthermore, multiple targets were detected, localised and tracked using audio information and convolutional networks in [16]. Deep learning strategies have also been used to identify the presence of different drones in outdoor environments, by analysing and classifying their spectrogram-based sound signatures in [17] or by merging them with wave radar signals in a convolutional network [18]. However, these strategies are performed from ground stations. There isn't much developed when it comes to identifying a UAV from the audio data captured from microphones on-board another UAV.

3 SYSTEM OVERVIEW

The primary aerial vehicle from which all test are carried out is a quad-rotor "Matrice 100", manufactured by DJI. This platform is popular because it can carry multiple sensors for outdoor navigation and autonomous flight, as it can bare a load of up to 1000 grams.

The audio capture system is the 8SoundUSB system that is part of the ManyEars project [19]. It is composed of 8 miniature microphones and an USB-powered external audio interface. The microphones were designed for mobile robot audition in a dynamic environment, implementing real-time processing to perform sound source localisation, tracking, and separation.

For audio recording and processing, we mounted the microphones over the same 3D-printed structure used by [14] to record eight-channel audio in raw format. All of the hardware was driven by the on-board Intel Stick Computer, with 32 GB of RAM and Linux Ubuntu 16.04 LTS. The recordings were carried out in two different environments: with an intruder drone and without an intruder drone. The intruder drone was a Drone Bebop 2.0, manufactured by Parrot, which is known for its stability and ease of control.

The place where the recordings were made was in the Centre of Information of the Instituto Nacional de Astrofisica, Optica y Electronica (INAOE), where there is a considerable large area that is appropriate for flying multiple drones at once.

The recording process is shown in Figure 2. First, the microphones are placed in the Matrice 100, with microphones 1, 2, 3 and 4 mounted in the front and microphones 5, 6, 7 and 8 mounted in the back. Then, an expert pilot controls the drone while the audio is recorded on-board the Intel Stick Computer.

The specifications with which audios were recorded are:

- The sampling rate is 48 kHz to allow a considerable amount of original resolution which can later be reduced if need be.
- Recording time: 240 seconds in the environment

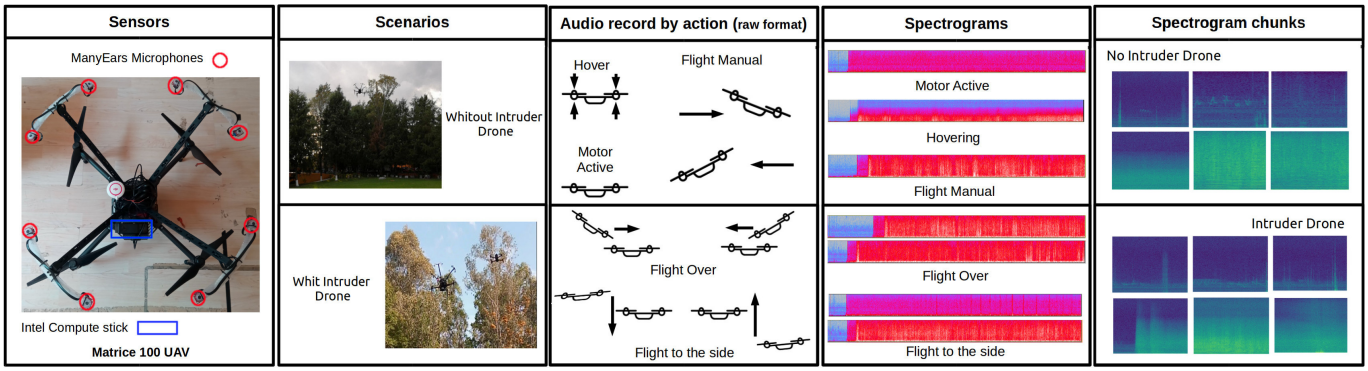


Figure 2: General overview to record the audio in two different environments and generate a dataset.

with an intruder drone, labelled as the class ”intruder drone”; and 198 seconds in the environment without it, labelled as the class ”no intruder drone.”

- The drones performed different actions while recording in both environments. In the environment ”intruder drone”, these actions were: on the ground with just the motors activated, hovering, and manual flight. In the environment ”intruder drone”, the actions were: the intruder drone flying on the side of the drone and over the top of the drone.

- The audio files were then manually transferred to a computer in the ground. This was done to avoid latency issues in the wireless transfer. The audio files were then transformed to the time-frequency domain, generating a spectrogram for each microphone (as detailed in the following section).

4 Training Dataset

The training dataset was created from the spectrograms generated by the recorded data, and apart from the training dataset, a testing dataset was created to validate the system. Each audio file was segmented in 2-second segments. The Short Time Fourier Transform was applied to each segment, with a 1024-sample Hann window (to avoid spectral leakage) and 75% overlap. The audio files in the negative class ”no intruder drone” include recordings of the air blowing through the trees, voices, cars, people and the noise of the Matrice’s motors. The positive class ”intruder drone” includes the recording of 200 seconds of the intruder drone flying on the side and over the Matrice 100. The Tables 1 and 2 show the spectrograms generated for each action which make up the whole of the training data set.

4.1 CNN architecture

We propose a convolutional neural network (CNN) as our classification model. This network is based on the architecture of the Google Net Inception v.3 (as shown in Figure 3)

Action	Time	Spectrograms (by mic)
Motor Activation	198.0 sec	98
Hovering	198.0 sec	98
Flight Manual	198.0 sec	98
Flight Manual 2	198.0 sec	98
	792.0 sec	**3168 (by all mics)**

Table 1: Spectrograms of the class ”no intruder drone”.

Action	Time	Spectrograms (by mic)
Flight over	200.0 sec	100
Flight to the side	200.0 sec	100
Flight over 2	200.0 sec	100
Flight to the side 2	200.0 sec	100
	800.0 sec	**3200 (by all mics)**

Table 2: Spectrograms of the class ”intruder drone”.

using Keras and Tensorflow. We employ a transfer learning strategy. Meaning, our system uses a model that was already trained on the ImageNet corpus [20], but we then augmented it with a new top layer to be trained with our recorded data. This is done so that the resulting model is focused in recognising the spectrogram-type of images relevant to our application: identifying an intruder drone flying near another.

The training data set was arranged in folders, each representing one class and baring approximately 3000 images. The model inherited the input requirement of the Inception V.3 architecture, receiving as input an image of size 224 x 224 pixels. The network was trained for 4500 epochs. Since the softmax layers can contain N labels, the training process corresponds to learning a new set of weights; that is to say, it is technically a new classification model.

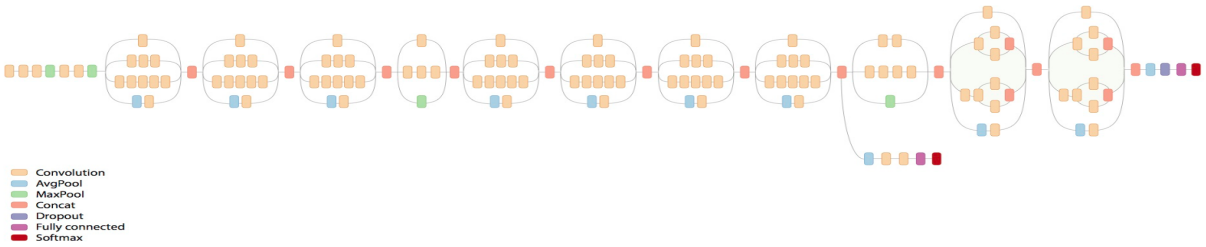


Figure 3: Schematic diagram of Inception V3.

5 Spectrogram Analysis

It is important to manually analyse the resulting spectrograms, to observe (in a preliminary fashion) if both classes are distinguishable to a human listener. The first analysis was made with the Audacity software [21] to visualise the audio data as a spectrogram. Then, the audio files were reproduced to see if a human listener was able to identify the intruder drone flight during the recordings. In Figure 4, the possible positions corresponding to these moments are marked in a circle.

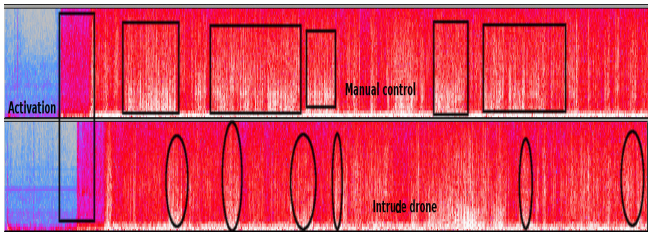


Figure 4: Comparison between spectrograms of manual control (top) and intruder drone (bottom).

Once it was shown that a human listener is able to identify the intruder drone, further analysis was carried out. 2-second time-frequency spectrograms were generated (as described in Section 4) for the two classes, and are shown in Figure 5. As it can be seen, there is an important amount high-frequency energy present in the ”intruder drone” class that is not present in the class ”no intruder drone.”

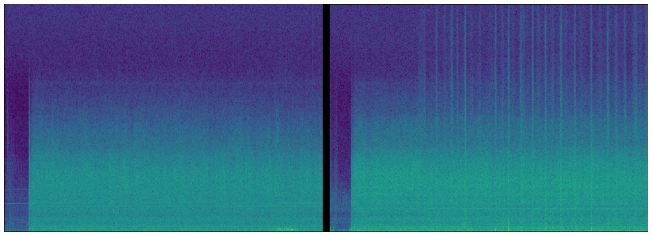


Figure 5: Spectrograms generated of activate motors (left) and intruder drone flight (right).

6 Results of the drone classification

The results shown in this section is that of the trained classifier. Its aim is to classify between two classes of input spectrograms. It could be argued that it is actually a verifier. However, we evaluated it as a classifier for future proofing.

6.1 Validation

We performed two experiments to evaluate the classification model. The first experiment shows the overall effectiveness of the model by testing it with 920 images for each class with an average inference time of 0.4503 sec. Table 3 presents the results of this test, and it can be seen that the class ”intruder drone” is consistently classified correctly, with only 19 wrong classifications out of 920 tests. However, the class ”no intruder Drone” gives a lower accuracy, with 163 images wrong classifications.

Class	Images	Incorrect	Accuracy
No intruder Drone	920	163	**82.28%**
Intruder Drone	920	19	**97.93%**

Table 3: Validation of classification network.

To a better understanding of the performance of the classifier we considered a binary classification where a nearby drone is considered as a positive sample in this way we have the true positive (Tp) = 901, true negative (Tn) = 757, false positive (Fp) = 19 and false negative (Fn) = 163. In Table 4, we show the result of Accuracy, Precision and Recall provide a better understanding of the performance of the classifier.

Accuracy	Precision	Recall
0.90108	0.97934	0.84680

Table 4: Accuracy, precision and recall result.

The second experiment measures the output of the model for each class, given a representative spectrograms to test with. 20 spectrograms were chosen (10 for each class), and the model outputs of each class are shown in Table 5. Although some outputs are below 70% (which implies some uncertainty of the model), the final classification is correct

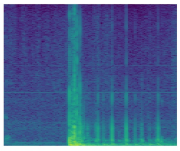
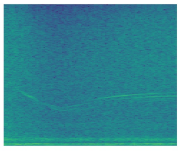
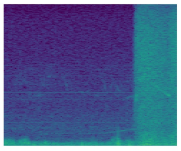
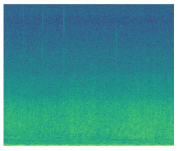
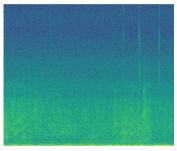
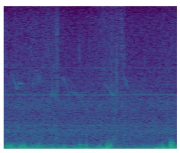
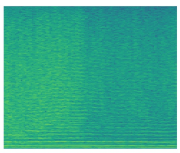
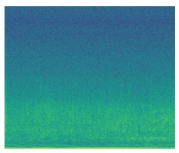
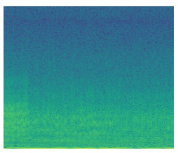
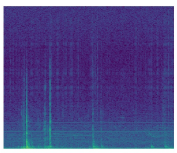
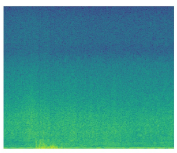
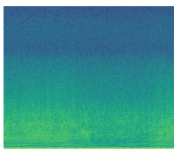
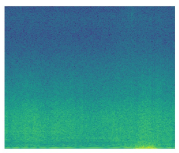
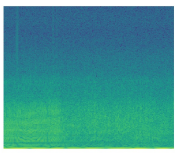
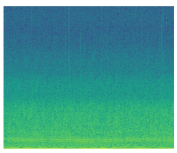
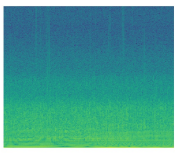
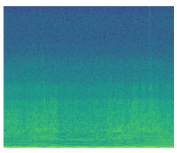
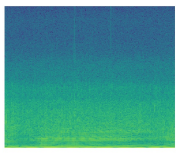
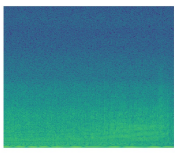
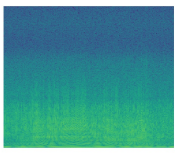
Class	Classification				
					
No intruder Drone	**97.82%**	**93.28%**	**78.82%**	**63.80%**	**76.59%**
Intruder Drone	2.18%	6.72%	21.18%	36.2%	23.41%
					
No intruder Drone	**77.54%**	**85.76%**	**84.66%**	**89.57%**	**82.31%**
Intruder Drone	22.46%	14.24%	15.34%	10.43%	17.69%
					
Intruder Drone	**88.23%**	**90.46%**	**94.70%**	**70.50%**	**91.57%**
No Intruder Drone	11.77%	9.54%	5.30%	29.50%	8.43%
					
Intruder Drone	**93.14%**	**91.10%**	**75.93%**	**65.27%**	**98.89%**
No Intruder Drone	6.86%	8.90%	24.07%	34.73%	1.11%

Table 5: Example of classification with CNN using some test images.

in all cases. These results give us a representative view of the expected performance of the model with the two classes that are relevant to our application: identifying an intruder drone flying near another.

7 Conclusion

In this paper, we proposed a CNN-based classifier of two types of environments: with and without an intruder drone, using only audio captured by a UAV. A time-frequency spectrogram was used as an the signal representation, which is compatible with known CNN-based architectures. We employed a transfer-learning strategy, with which the top layer of a pre-trained Google's Inception V.3 model was modified and trained, which made the training process very efficient. The classifier was evaluated in two experiments, and it achieved a good classification performance in most cases. The work can be further strengthened by using the eight microphones individually to detect the direction of the intruder drone. This would allow a fast enough detection, to give enough time to plan a strategy for collision avoidance.

References

[1] Matthew Ritchie, Francesco Fioranelli, Hervé Borrion, and Hugh Griffiths. Multistatic micro-doppler radar feature extraction for classification of unloaded/loaded micro-drones. *IET Radar, Sonar & Navigation*, 11(1):116–124, 2016.

[2] RIA Harmanny, JJM De Wit, and G Prémel Cabic. Radar micro-doppler feature extraction using the spectrogram and the cepstrogram. In *2014 11th European Radar Conference*, pages 165–168. IEEE, 2014.

[3] F Fioranelli, M Ritchie, H Griffiths, and H Borrion. Classification of loaded/unloaded micro-drones using multistatic radar. *Electronics Letters*, 51(22):1813–1815, 2015.

[4] JJM De Wit, RIA Harmanny, and P Molchanov. Radar micro-doppler feature extraction using the singular value decomposition. In *2014 International Radar Conference*, pages 1–6. IEEE, 2014.

[5] Pavlo Molchanov, Ronny IA Harmanny, Jaco JM de Wit, Karen Egiazarian, and Jaakko Astola. Classification of small uavs and birds by micro-doppler signatures. *International Journal of Microwave and Wireless Technologies*, 6(3-4):435–444, 2014.

[6] Wenyu Zhang and Gang Li. Detection of multiple micro-drones via cadence velocity diagram analysis. *Electronics Letters*, 54(7):441–443, 2018.

[7] Andrew W Christian and Randolph Cabell. Initial investigation into the psychoacoustic properties of small unmanned aerial system noise. In *23rd AIAA/CEAS Aeroacoustics Conference*, page 4051, 2017.

[8] Koutarou Furukawa, Keita Okutani, Kohei Nagira, Takuma Otsuka, Katsutoshi Itoyama, Kazuhiro Nakadai, and Hiroshi G Okuno. Noise correlation matrix estimation for improving sound source localization by multirotor uav. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3943–3948. IEEE, 2013.

[9] Takuma Ohata, Keisuke Nakamura, Takeshi Mizumoto, Tezuka Taiki, and Kazuhiro Nakadai. Improvement in outdoor sound source detection using a quadrotor-embedded microphone array. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1902–1907. IEEE, 2014.

[10] Seongha Park, Sangmi Shin, Yongho Kim, Eric T Matson, Kyuhwan Lee, Paul J Kolodzy, Joseph C Slater, Matthew Scherreik, Monica Sam, John C Gallagher, et al. Combination of radar and audio sensors for identification of rotor-type unmanned aerial vehicles (uavs). In *2015 IEEE SENSORS*, pages 1–4. IEEE, 2015.

[11] Prasant Misra, A Anil Kumar, Pragyan Mohapatra, and P Balamuralidhar. Aerial drones with location-sensitive ears. *IEEE Communications Magazine*, 56(7):154–160, 2018.

[12] Brendan Harvey and Siu OYoung. Acoustic detection of a fixed-wing uav. *Drones*, 2(1):4, 2018.

[13] Brendan Harvey and Siu O'Young. A harmonic spectral beamformer for the enhanced localization of propeller-driven aircraft. *Journal of Unmanned Vehicle Systems*, 7(2), 2019.

[14] Oscar Ruiz-Espitia, Jose Martinez-Carranza, and Caleb Rascon. Aira-uas: An evaluation corpus for audio processing in unmanned aerial system. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 836–845. IEEE, 2018.

[15] Kotaro Hoshiba, Kai Washizaki, Mizuho Wakabayashi, Takahiro Ishiki, Makoto Kumon, Yoshiaki Bando, Daniel Gabriel, Kazuhiro Nakadai, and Hiroshi Okuno. Design of uav-embedded microphone array system for sound source localization in outdoor environments. *Sensors*, 17(11):2535, 2017.

[16] Zeeshan Kaleem and Mubashir Husain Rehmani. Amateur drone monitoring: State-of-the-art architectures, key enabling technologies, and future research directions. *IEEE Wireless Communications*, 25(2):150–159, 2018.

[17] Sungho Jeon, Jong-Woo Shin, Young-Jun Lee, Woong-Hee Kim, YoungHyoun Kwon, and Hae-Yong Yang. Empirical study of drone sound detection in real-life environment with deep neural networks. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1858–1862. IEEE, 2017.

[18] Byung Kwan Kim, Hyun-Seong Kang, and Seong-Ook Park. Drone classification using convolutional neural networks with merged doppler images. *IEEE Geoscience and Remote Sensing Letters*, 14(1):38–42, 2017.

[19] François Grondin, Dominic Létourneau, François Ferland, Vincent Rousseau, and François Michaud. The manyears open framework. *Autonomous Robots*, 34(3):217–232, 2013.

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[21] D Mazzoni and R Dannenberg. Audacity [software]. *The Audacity Team, Pittsburg, PA, USA*, 2000.

Hear-and-avoid for UAVs using convolutional neural networks

Dirk Wijnker, Tom van Dijk, Mirjam Snellen, Guido de Croon and Christophe De Wagter*
Delft University of Technology, Kluyverweg 1, 2629HS Delft, the Netherlands

Abstract

To investigate how an Unmanned Air Vehicle (UAV) can detect manned aircraft with a single microphone, an audio data set is created in which UAV ego-sound and recorded aircraft sound are mixed together. A convolutional neural network is used to perform the air traffic detection. Due to restrictions on flying UAVs close to aircraft, the data set has to be artificially produced, so the UAV sound is captured separately from the aircraft sound. They are then mixed with UAV recordings, during which labels are given indicating whether the mixed recording contains aircraft audio or not. The model is a CNN which uses the features MFCC, spectrogram or Mel spectrogram as input. For each feature the effect of UAV/aircraft amplitude ratio, the type of labeling, the window length and the addition of third party aircraft sound database recordings is explored. The results show that the best performance is achieved using the Mel spectrogram feature. The performance increases when the UAV/aircraft amplitude ratio is decreased, when the time window is increased or when the data set is extended with aircraft audio recordings from a third party sound database. Although the currently presented approach has a number of false positives and false negatives that is still too high for real-world application, this study indicates multiple paths forward that can lead to an interesting performance. In addition, the data set is provided as open access, allowing the community to contribute to the improvement of the detection task.

1 Introduction

More and more UAVs are entering the air every day, both for professional as well as for recreational purposes. Safety and regulations are subjects undergoing intense study nowadays in the UAV industry, as UAVs form a hazard for people, other (air) traffic, buildings, etc. For this research, the focus is on the collisions between UAV and air traffic, which are still possible to occur. For example, emergency helicopters sometimes fly low in UAV-permitted airspace. Part of this problem can be solved by establishing (and following) good rules and laws, but also technology can help out. Technology becomes even more important when UAVs have to operate fully autonomously, as required by many future applications. A project initiated by Single European Sky ATM Research (SESAR) that aims to increase air traffic safety regarding to UAVs is called Percevite[1]. Using multiple lightweight, energy-efficient sensors obstacles should be avoided to protect UAVs and their environment. One such a sensor is a microphone, which fulfills the task of 'hear-and-avoid', meaning that it should detect and avoid air traffic by sound. The goal of this research is to create a safer airspace by creating this hear-and-avoid algorithm.

Figure 1: The acoustic camera on the runway of Lelystad Airport.

The first feasibility study for hear-and-avoid has been performed by Tijs et al [1]. In this research an acoustic vector sensor is used to detect other flying sound sources. Two co-authors, De Bree and De Croon [2], have used an acoustic vector sensor in order to detect sound recorded on a UAV for military purposes. However, neither works have used deep artificial neural networks to separate aircraft and UAV sounds. Moreover, there are two research groups that have tried to identify the position of other UAVs using sound recorded from a UAV. Basiri et al. [3, 4, 5, 6] try to determine the position of a UAV in a swarm of UAVs. The transmitting UAV sends a chirp sound in the air that has frequencies different than the UAV's ego-sound, which can be picked up quite well

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

[1] www.percevite.org

while flying. Also, they do tests with engines of the receiving UAV turned off and the transmitting UAVs not transmitting the chirp anymore. Also here, based on the engine sounds of the transmitting UAV its location can be determined. The hear-and-avoid algorithm can be seen as a follow up of these researches, as they have not managed to identify other air traffic by its original sound while also having the engines turned on. Harvey and O'Young [7] show that with two microphones, the detection of another UAV can be performed at such a distance that is double the distance to prevent head-on collision. Furthermore, research is performed focusing only on the UAV sound by Marmaroli et al. [8]. They have created an algorithm that is able to denoise the ego-sound of the UAV based on the knowledge about the propellers' revolutions per minute (RPM).

One of the reasons that there is not a large amount of research performed on audio analysis for UAVs is that there are alternatives that provide traffic information, such as ADS-B, GPS, vision, etc. However, all alternatives have their disadvantages and do not fully eliminate the chance of a collision. For example, ADS-B requires a system in an aircraft that is not always present or turned on. For vision based sense-and-avoid its images can be disturbed due to speed, rain, fog, darkness, objects, etc. Sound, on the other hand, is inevitable for motorized aircraft, so it is a promising method. Moreover, microphones are lightweight, easy to use, omnidirectional and only weakly influenced by weather. The challenge that sound brings in this application is that many different sounds are present, such as the UAV's ego-noise, wind, air traffic and environmental sounds.

In this research the following situation is studied: a UAV, which is carrying a single microphone, flies around and should detect incoming or passing aircraft based on sound. The detection of aircraft will be realized by means of a convolutional neural network (CNN) due to their promising performance on sound in [9],[10] and [11]. The representative data set that is needed, which consists of audio recordings taken on a UAV including aircraft sound, does not exist yet and therefore needs to be artificially created. The CNN uses three audio features as input: Mel Frequency Cepstral Coefficients (MFCCs), spectrograms and Mel spectrograms. Four variables are changed in the data sets to discover their influence: the window length, the amplitude ratio UAV/aircraft, the type of labeling and the use of third party database recordings.

The remainder of the article is structured as follows. The generation of the data set is explained in section 2, including how the individual sound recordings are obtained, how those are processed and mixed to recordings that include both UAV and aircraft sound. Secondly, the features and the model are described in section 3. The results for each of the models are shown in section 4 and discussed in section 5.

2 AUDIO ACQUISITION

This research needs a database that contains audio recordings, recorded on UAVs, of the UAV's ego-sound and closely approaching aircraft. Such a database does not exist yet and therefore it is created for this purpose. The database consists of (preprocessed) sound recordings (of UAVs, aircraft and rotorcraft) and labels, which indicate whether only UAV sound is present or UAV and aircraft sound are present.

2.1 Sound recordings

The laws on UAVs prevent the UAV to come in the vicinity of an aircraft. In order to still have a representative database of UAV sounds that include passing aircraft, the UAV sounds and aircraft sounds are recorded separately and mixed afterwards. Three types of recordings have been used: self-made recordings using a microphone on a UAV, general aviation aircraft recordings using a microphone array and aircraft recordings obtained from a third party audio database.

2.1.1 Recordings of the UAV sounds

The UAV sounds are recorded in the Cyberzoo of the TU Delft. This is a protected area for UAVs to be safely and legally flown at the university. An 808 micro camera[2] is placed under a Parrot Bebop UAV, so that its body already blocks part of the UAV's ego-sound. Between the UAV and the microphone, foam is used to absorb the mechanical vibrations. During the recordings, the UAV performed rotations and movements around its pitch, roll and yaw axes at different speeds. After recording, the data is cropped to remove the silences at the beginning and at the end. These recordings are complemented with audio recordings from a mobile phone that filmed the UAV from a close distance. Effectively a total of 20 minutes of UAV recordings are used.

2.1.2 Recordings of general aviation flights

Since the most probable group to come in contact with UAVs is general aviation (GA) rotor- and aircraft, flyover data has been obtained at the biggest GA airfield of the Netherlands, Lelystad Airport, in collaboration with the Aircraft Noise and Climate Effects (ANCE) section of the TU Delft.

As Lelystad airport is expanding to a larger airfield, the runway is extended, but the new part is not in use yet. This part of the runway is therefore a perfect place to obtain recordings as the aircraft would fly straight over the so-called "acoustic camera".

The acoustic camera, designed and built by the TU Delft [12], consists of an array with 8 bundles of 8 microphones[3]. The bundles are arranged in a spiral shape for optimal beamforming purposes. The microphones are covered in a foam layer to decrease the noise due to wind. Moreover, the array

[2] http://www.chucklohr.com/808/

[3] Model: PUI AUDIO 665-POM-2735P-R

is covered in foam in order to absorb ground reflections. All the bundles are connected to a Data Acquisition Box (DAQ) which samples the data at 50 kHz and sends it to the connected computer. Not only the DAQ is connected to the computer, but also an ADS-B receiver in order to receive aircraft position information. However, the ADS-B did not produce useful information as none of the GA aircraft broadcast ADS-B information. Moreover, a mobile phone camera is placed in the center of the array to capture the flyover on video, but this data is not used for this research. The setup of the acoustic camera is shown in Figure 1.

In total 75 recordings are obtained, which consist of background noise recordings and flyovers. One recording sometimes consists of more than one flyover. Effectively, 75 GA aircraft and 9 helicopter flyovers are captured. The background noise consists of microphone noise, noise due to wind, distant traffic and a distant motor race track.

For this research only the recording of one microphone is necessary, so from only one microphone the recordings are extracted. Every microphone is checked to make sure it worked correctly. One of the 64 microphones is faulty, so its data is not used.

2.1.3 Recordings obtained from a third party audio database

With regard to creating a data set that is representative for the possible air traffic sounds that a UAV could encounter, it had to consist of more than only flyover data. For example, other background noise could influence the detection performance. Therefore also a (free) audio database[4] is consulted to obtain helicopter and (propeller) aircraft sounds. Only the sound samples that are of sufficient quality and which are not mixed with (too much) other background noise are selected.

2.2 Data preprocessing

All the separate recordings are manually modified before adding them together. Some UAV recordings contained heavy vibrations of the tape that held the microphone. Those recordings are removed from the data set. For both the UAV recordings and the third party database recordings the silent/fading start and end are cut out. The recordings obtained at Lelystad airport do not require this as the parts that do not include aircraft sound are used as background noise. Instead, we manually labelled every second in the recording, indicating whether it consists of only background noise or include aircraft sound. The recordings from Lelystad Airport include noise introduced by the microphones and the wind. A first order Butterworth low-pass filter is used to remove most of the noise. Most of the time the aircraft sound information is in the frequency region lower than 100 Hz. Only during a flyover aircraft sound information comes above this value. In order to capture the higher frequency content during a flyover but also remove much of the noise during the rest of the time, the cut-off frequency is set on 2.5 kHz.

All the recordings are resampled to a sample rate of 8 kHz as there is no important information present above the Nyquist frequency of 4 kHz and it decreases the size of the data set significantly, which shortens the computational time. Secondly, the sound recordings are normalized by scaling the amplitude between -1 and 1, so that the amplitude of two recordings is similar. Before mixing aircraft and UAV sounds, also data augmentation is applied to all the separate aircraft and UAV recordings in order to increase the size of the data set, which increases the performance of the model. Three types of data augmentation are applied: addition of white noise, increase in pitch and decrease in pitch. The white noise is a randomly generated Gaussian distribution with mean 0 and a variance of 0.005. The pitch is increased and decreased by two semitones on the 12-tone. An increase of two semitones relates to $\sqrt[12/2]{2} \approx 1.12$ times the original frequency. After augmentation, the data set is four times its original size, one original data set plus three augmented data sets.

2.3 Mixing the recordings

In order to get sound samples that include both aircraft and UAV sound, the following mixing procedure is used.

First, the whole data set is split up in a test set and in a training set. All the augmented versions of a sound sample are always in the same set as their original sound sample to ensure that the two sets are uncorrelated.

Secondly, each recording from Lelystad airport is combined with a randomly selected UAV recording of the same set. In some (part of the) recordings only background noise is present. This background noise is necessary since without the noise, the model might classify every sound which is not UAV sound as aircraft sound. Mixing consists of adding a segment of the Lelystad airport sound sample, which has a random length, to one of the UAV recordings on a random starting position. If the starting position plus the length of the segment is longer than the length of the UAV sound sample, the added segment is cut off at the end of the UAV sound sample. The mixed sample therefore never exists of only aircraft sound. The total length of each mixed sample is equal to the length of the UAV recording, which is different for each recording.

Mixing the third party database recordings is done slightly different than the method described for the Lelystad recordings because the third party database recordings always exist fully of aircraft sound. The difference between the two mixing methods is that not only a part of the recording is added to the UAV sound sample, but the whole recording is added instead (at a random starting position).

The detection model in this paper requires the inputs to be of equal length (more on this in subsection 3.2). As this is not the case for the combined samples, the third step is to cut the combined samples to equal lengths. To maximize the amount

[4] https://freesound.org/

of data in the sets, the cutting length is set on 51 seconds, which is equal to the length of the shortest combined sound sample.

The amplitude ratio when mixing the UAV and aircraft sound is not always 1:1. In this work, four UAV/aircraft amplitude ratios will be used, namely 0:1 (which means no UAV sound), 1:1 (equal amplitudes), 1:4 (aircraft sound amplitude is four times larger) and 1:8 (aircraft sound amplitude is eight times larger). Most of the time, a ratio of 1:4 is used. This ratio is obtained as follows. Assuming the average Sound Pressure Level (SPL) of a UAV at one meter distance is 76 dB[5] and that of an aircraft at 300 meters distance is 88 dB[6], the difference between the SPLs of the two sounds is 12 dB. Equation 1 shows how the SPL is calculated from the pressure p_1 (which is the amplitude in the waveform) of a sound and a reference pressure p_0. Taking the amplitude of the UAV waveform as reference pressure and the aircraft waveform as p_1, an SPL of 12 is obtained when the aircraft waveform is 4 times larger. If the ratio 1:4 is corresponding to an airplane on 300 meters distance, 1:1 corresponds to a distance of 1200 meters and 1:8 to a distance of 150 meters, following Equation 2. In this equation, r_2 is the distance of interest, r_1 the original distance, SPL_1 the SPL at r_1 and SPL_2 the SPL at r_2.

$$SPL = 20 \log \frac{p_1}{p_0} \tag{1}$$

$$r_2 = r_1 \cdot 10^{\frac{|SPL_1 - SPL_2|}{20}} \tag{2}$$

2.4 Labels

Each second of a mixed sample is given a binary label, indicating whether there is other aircraft sound present (1) or not (0). The recordings from Lelystad airport are labeled manually before mixing. There are two types of labeling, called nearby detection labeling and distant detection labeling. Nearby detection labeling is partly based on listening to the sound, and partly on looking at the spectrogram. The spectrogram, which is shown in Figure 2 and elaborated on in subsubsection 3.1.2, shows the amount of frequency content over time. Nearby detection labeling gives label 1 when a peak is visible in the spectrogram. By ear this is noticeable as more high frequency content is heard.

Distant detection labeling is purely based on hearing. The frames in which a human is able to separate noise from aircraft sounds are labeled 1. This time it cannot be based on the spectrogram as the aircraft sound is either not visible on the spectrogram (when it is blended in too much with the background noise) or it is visible (as a line on a single frequency caused by the propeller's rotational speed) but the background noise is louder than the aircraft sound. An example of the latter is shown in Figure 3, at which the horizontal line around 100 Hz is also present when no label is given.

[5] https://www.youtube.com/watch?v=uprXhH6-FNI
[6] http://airportnoiselaw.org/dblevels.html

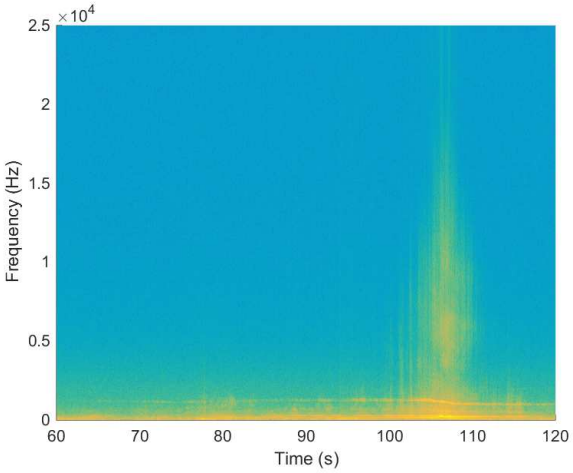


Figure 2: Spectrogram of a flyover recording. The exact flyover is between 100 and 110 seconds, which can be recognized by a yellow peak and a Doppler shift around 100 Hz. Also before and after the peak the aircraft sound is present, which is visible by the horizontal line around 100 Hz.

The time instances that are not labeled one are labeled zero, so also the background noise from the Lelystad recordings is given the same label as when there is no other aircraft sound present. In Figure 3, the areas in the spectrogram that are labeled as 1 are indicated in red for nearby detection labeling and green for distant detection labeling.

For the third party sound database, the whole aircraft recording is always labeled as a one, as each of the sound samples is selected on only having aircraft sounds. Again, all the time instances in the mixed recording that are not one are labeled zero.

3 AIRCRAFT AUDIO EVENT RECOGNITION

The aircraft sound will be detected by a framework that exists of a feature extractor and a classifier. The features capture important sound information and reduce the dimensionality of the data. They are the inputs for the classifier. Thereafter the classifier determines whether the sound sample contains aircraft sound or not.

3.1 Feature extraction

Three features are extracted from the combined sound samples using Python library Librosa [13]. First there are the Mel Frequency Cepstral Coefficients (MFCCs) [14], which are chosen because of their popularity in one of the biggest domains in machine hearing, Automatic Speech Recognition (ASR). The two other features, the spectrogram and Mel spectrogram, are visual representations of the sound samples. Content-based analysis of images is already quite developed [15], therefore the image of a sound might be a good starting point.

For every feature, each frame in the time dimension has a

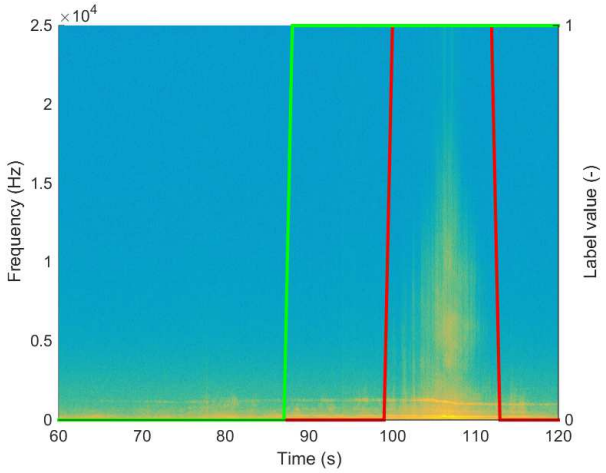


Figure 3: Spectrogram showing nearby detection labeling (red) and distant detection labeling (green).

length of one second. One second is a rather large frame but it chosen to reduce in dimensionality. The window moves over the sound sample with a step of one second. All the sound samples are 51 seconds long, thus from each sound sample 51 separate frames are obtained in the time dimension.

3.1.1 MFCC

The cepstrum is a domain which represents the rate of change in multiple frequency bands. MFCCs are the coefficients of which the cepstrum is composed. It has the ability to separate convoluted signals in the time domain[7]. This domain is therefore often used in speech recognition, to separate the vocal pitch and the vocal tract. The coefficients are obtained by taking the logarithm of the amplitude spectrum, converting this to the Mel scale and taking the Discrete Cosine Transform (DCT). The Mel scale, which is expressed as a function of frequency (f) in Equation 3, is a scale that approximates the human perception of frequency. This scale emphasizes the low frequencies (<1 kHz), which is also the frequency range in which most of the UAV/aircraft sound information is present. The full transformation from time domain signal to MFCC is shown in Equation 4 [16].

$$M(f) = 2595 \log\left(1 + \frac{f}{700}\right) \tag{3}$$

$$MFCC(d) = \sum_{k=1}^{K} (\log X_k) \cos\left[d\left(k - \frac{1}{2}\right)\frac{\pi}{k}\right] \quad \text{for} \quad d = 0, 1, ..., D \tag{4}$$

In this equation X_k is the Discrete Fourier Transform (DFT) obtained in Equation 5 of which the frequency belonging to each k is warped to the Mel scale by Equation 3. D is the total number of coefficients and N the number of data point in the time frame. The number of coefficients used in this research is 20.

$$X_k = \sum_{n=0}^{N-1} X_n e^{-\frac{2\pi i}{N} kn} \quad \text{for} \quad k = 1, 2, ..., N \tag{5}$$

3.1.2 Spectrogram

Spectrograms are visual representations of the energy per frequency plotted against time, of which the Mel spectrogram uses the Mel scale of Equation 3 on the frequency axis. A typical flyover spectrogram (without UAV sound), is shown in Figure 2. In this figure the point where the aircraft is passing the array is between 100 and 110 seconds, which is visible with the large yellow peak and a Doppler shift (the sigmoid-shaped line around 1 kHz). It also shows that when the aircraft is further away, it lacks in high frequency content (due to atmospheric attenuation). That means most of the time only the aircraft's low frequency content is heard by the UAV in combination with low frequency noise.

The spectrograms are calculated following Equation 6, which is the magnitude to the power p of the Short-Time Fourier Transform (STFT). Usually the Power Spectral Density (PSD) is chosen, for which $p = 2$. It uses a window function $w[n]$, in this case the Hann window of one second, of which m is the index of the position in the window function with length N, discrete frequency k, signal $x[n]$ at time n.

$$Spectrogram = \left| \sum_{n=-\infty}^{\infty} x[n] w[n-m] e^{\frac{-i2\pi kn}{N}} \right|^p \tag{6}$$

3.2 Model

The previously described features are the input for a deep artificial neural network: the convolutional neural network (CNN). It has shown best performance for sound event recognition tasks in [9],[10] and [11]. The basic CNN used in this research is shown in Figure 4. The network is created with the Python libraries Keras [17] and Tensorflow [18].

Even though the features consist of 51 second of UAV/aircraft sound, the input for the CNN is a smaller time window which slides over the time axis. The smaller time window is used as otherwise the detection output of a frame could be depended on data from later frames, due to the fully connected layer. Multiple window lengths are used, as shown in section 4. In the basis, however, the window size is three seconds. This window slides over the feature's time axis with a step of one second.

The first layers of the CNN are convolutional layers. There are two subsequent sets of layers, each consisting

[7] http://research.cs.tamu.edu/prism/lectures/sp/l9.pdf

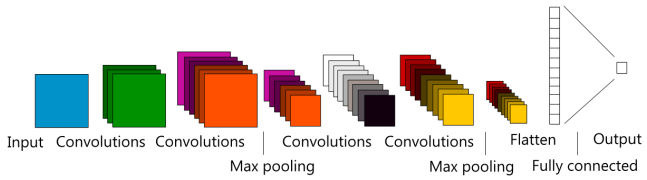


Figure 4: Architecture of the CNN. The input is a moving time window over the spectrogram, Mel spectrogram or MFCC. The output a binary value indicating whether aircraft sound is present or not.

Table 1: Model parameters of the CNN from Figure 4.

Parameter	CNN
Convolution units first set	32
Convolution units second set	64
Kernel size	3x3
Pooling size	2x2
Dropout probability 1	0.25
Dropout probability 2	0.5

of two convolutional layers, followed by a max pooling layer. The convolutional layers use the Rectified Linear Unit (ReLU) as activation function and it applies zero padding to the input. After the two sets, the output is flattened in order to be able to connect it with the output layer, a fully connected layer. For the output, a sigmoid activation function is used, which scales the output (as a float) between 0 and 1. The binary discrimination threshold determines whether this output becomes a 1 or a 0, so whether an aircraft is present or not, respectively. The network is based on [11] and its parameters are modified based on preliminary test results.

Training the network is performed by means of a binary cross-entropy loss function and the Adam optimizer [19]. The Adam optimizer parameters are the same as in the original paper, so a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and no decay. After each pooling layer, dropout is used in order to prevent overfitting of the training data. The parameters for the CNN are shown in Table 1.

4 Results

Each feature is combined with the CNN, so in total three models are tested. They are trained and tested on multiple data sets, which are listed in Table 2. To check the influence of certain parameters in the data set or in the model, four parameters are altered during the runs: the window length, the labeling type, the ratio in amplitude between the UAV and aircraft sound and whether third party database recordings and Lelystad airport recordings are used or only the Lelystad airport recordings.

There is one basis run, for which the window length is 3 seconds, the labeling is nearby detection labeling, the UAV/aircraft ratio is 1:4 and there are no third party database recordings involved. For all the other runs, only one variable of the basis run is changed each time.

Table 2: Overview of the variables that are changed for each run, including their corresponding values and the values of the standard case, the basis run.

Variables	Basis values	Variations
UAV/Aircraft ratio	1:4	0:1 1:1 1:8
Third party database used	No	Yes
Labeling type	Nearby detection labeling	Distant detection labeling
Window length (s)	3	10 15 20

Table 3: The number of each run with their corresponding changed variable and the corresponding value.

Run #	Variation
1	UAV/Aircraft ratio: 0:1
2	UAV/Aircraft ratio: 1:1
3	Basis run
4	UAV/Aircraft ratio: 1:8
5	Database used: Yes
6	Distant detection labeling
7	Window length: 10
8	Window length: 15
9	Window length: 20

The window length is either 3, 10, 15 or 20 seconds. The Lelystad airport recordings are labeled manually, in two manners, as explained in subsection 2.4. For distant detection labeling the training is performed with distant detection labeling and the testing is performed with nearby detection labeling. The idea behind this method is that the model could learn aircraft sound when it is not so obviously present, so that detection when the aircraft is obviously present is outstanding. The amplitude ratio between the UAV and the aircraft is tested when no UAV sound is present, and for the ratios 1:1, 1:4 and 1:8. Lastly, the third party database sounds are either added to the data set or omitted.

From here on, each specific run is indicated by the number of the run given in Table 3. The performance of the models is compared for each of the variables (window length, label type, etc.). This comparison is based on the Receiver Operating Characteristic (ROC) curve. The ROC curve shows the True Positive Rate (TPR) against the False Positive Rate (FPR) for all possible binary discrimination thresholds. The area under the curve (AUC) is a measure of accuracy of the binary classifier. In this research specifically, especially the region of low FPR is important, as it shows how many times the UAV would falsely decide to warn the operator or descend.

For each point on the ROC curve the desirable discrimination threshold can be extracted, which determines whether the output from the model is classified with label 1 or label 0.

4.1 *Influence of the UAV/aircraft ratio*

Runs 1, 2, 3 and 4 are simultaneously plotted for the CNNs in Figure 5. In general, the best performance is achieved for the cases where there is no UAV sound present (run 1). If the UAV's ego-sound is added to the aircraft sound with an amplitude ratio of 1:1 (run 2), the performance is the worst in all cases. The figures show that amplifying the aircraft sound increases performance, however, there is little increase between the ratio 1:4 and 1:8. The expected result is that the less UAV content is present, the more the performance would converge to the result of run 1. Only for the MFCC and Mel spectrogram this trend is visible in the lower FPR region. Looking at the AUC, the MFCC and the spectrogram show no convergence to the ratio of 0:1. In the case of the Mel spectrogram, there is only a difference visible between the ratio of 1:1 and the others.

4.2 *Influence of the third party database recordings*

In the basis run, only the recordings from Lelystad airport are used. This means that all the recordings have (fairly) the same background noise and types of airplanes and they use the same recording equipment. In order to check how much the models rely on these characteristics, they are trained and tested with the third party database recordings as well for this run.

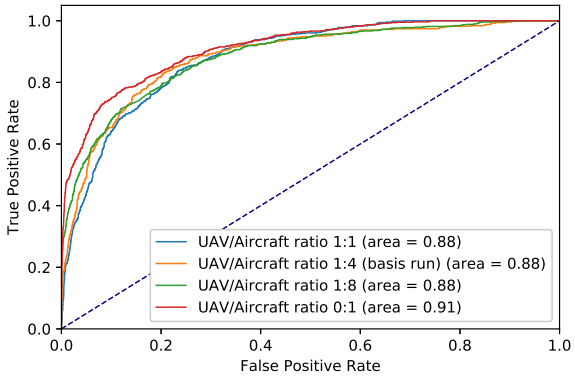
Figure 6 shows that for all the models, the addition of the third party database recordings improves the performance of the model. Only for the very low FPR (< 0.01), the basis run performs better for the MFCC-CNN and the Mel spectrogram-CNN.

4.3 *Influence of labeling*

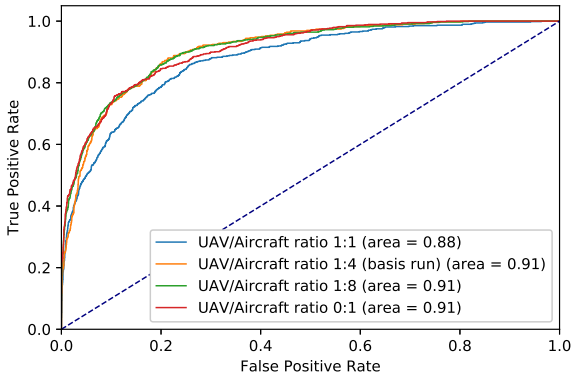
The third type of modification made in the data set relates to which labels are used for training. For all cases the nearby detection labeling is used for testing. For training, however, one run uses distant detection labeling and one run uses nearby detection labeling. When an aircraft is approaching, the lower frequencies of its generated sound reach the ear first. This low frequency content is in the same range as the background noise. It is therefore expected that for distant detection labeling a better separation is found in the model between drone and aircraft and therefore would also better perform for the nearby cases. Figure 7, however, does not prove this hypothesis. This time, for all features, the performance deteriorates when distant detection labeling is used.

4.4 *Influence of the window length*

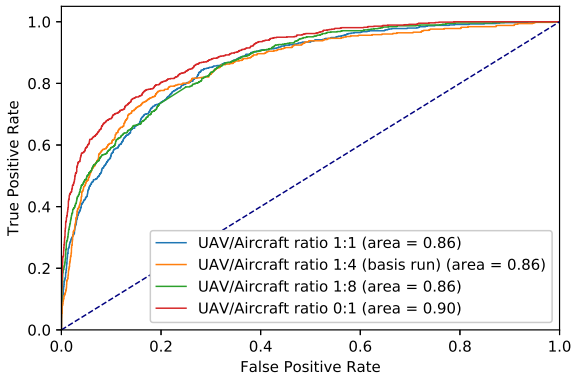
The window length of the CNN determines how many seconds of history are used to determine whether the sound contains aircraft sound or only UAV sound. The more history the sound contains, the better the development of (possible) aircraft sound can be captured. It is therefore expected that



(a) MFCC-CNN for different UAV/aircraft amplitude ratios.

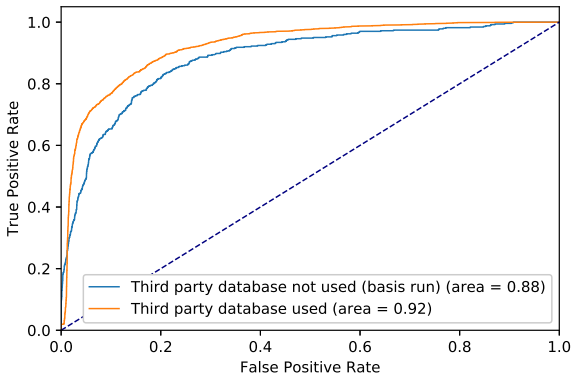


(b) Mel spectrogram-CNN for different UAV/aircraft amplitude ratios.

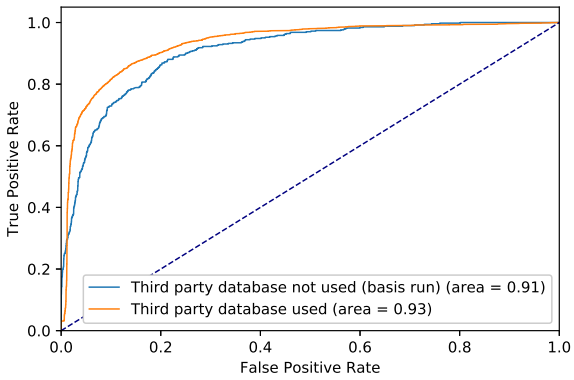


(c) Spectrogram-CNN for different UAV/aircraft amplitude ratios.

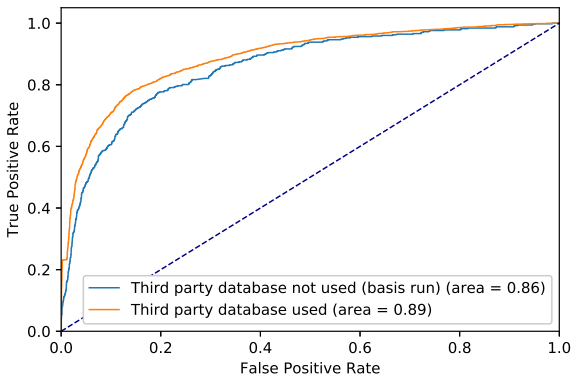
Figure 5: ROC curves showing the influence of the UAV/aircraft ratio for each feature. Best accuracy is achieved for the ratio 0:1 (no UAV sound present). The more UAV content is added, the worse the performance.



(a) MFCC-CNN with and without third party database recordings.

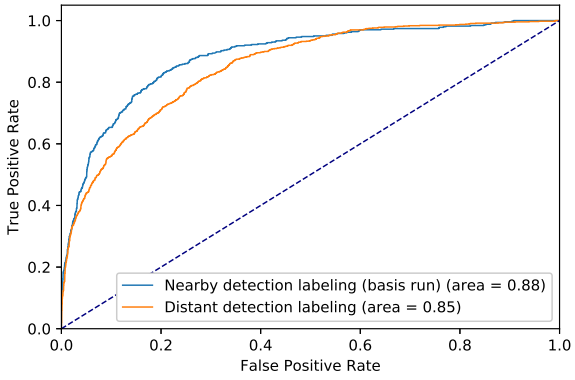


(b) Mel spectrogram-CNN with and without third party database recordings.

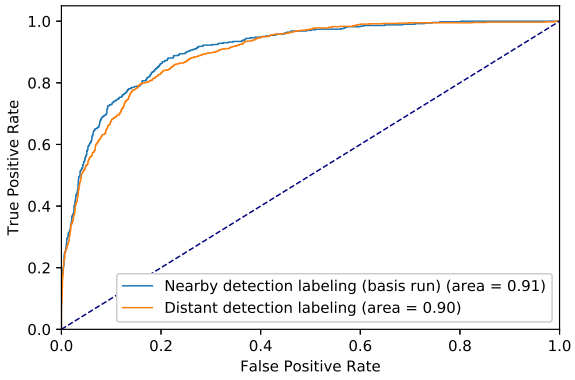


(c) Spectrogram-CNN with and without third party database recordings.

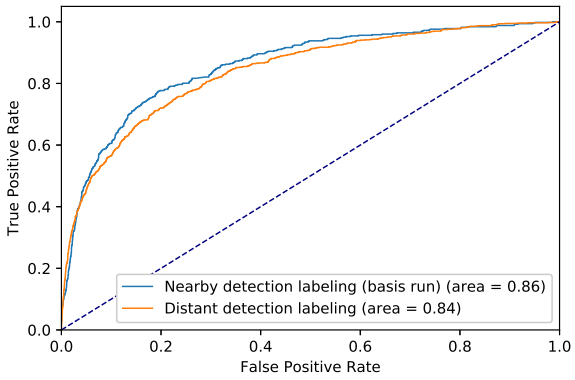
Figure 6: ROC curves showing the influence of the third party database recordings for each of the features. For all features, the performance increases using the third party database recordings.



(a) MFCC-CNN comparing the performance for different label types.



(b) Mel spectrogram-CNN comparing the performance for different label types.



(c) Spectrogram-CNN comparing the performance for different label types.

Figure 7: ROC curves showing the influence of labeling type for each of the feature. Each run is tested with nearby detection labeling. One run is using the nearby detection labeling for training as well and the other one uses the distant detection labeling during training.

with a larger window length a better performance is achieved. However, eventually the performance of longer time windows are expected to converge as history from long ago does not give useful information in detecting aircraft sound in the present.

This hypothesis is confirmed for the CNNs using Mel spectrogram, spectrogram and MFCC in Figure 8. Improvement in AUC between a three second window and a ten second window is shown in each of the subfigures. For window lengths of more than ten seconds, the AUC hardly changes. For the spectrogram-CNN there is a clear difference in the low FPR region between the 10 and 15 seconds.

4.5 Comparison of the features

So far, the results are only shown per feature. In order to show which feature works best, the features have been compared for the basis run in Figure 9. The results show that the Mel spectrogram performs best, followed by the MFCC. The spectrogram performs worst compared to the other two.

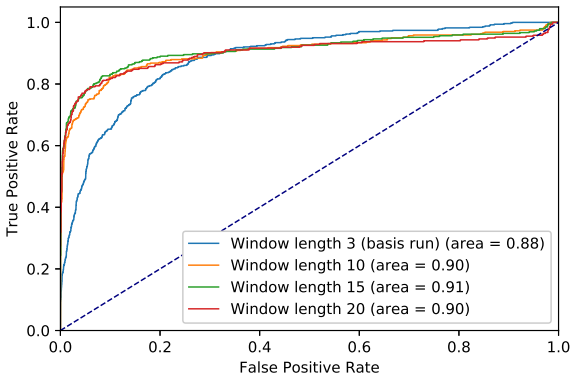
Even though the results are only set out for one run, this is true in general for the other runs. For the runs with a UAV/aircraft ratio of 0:1, 1:1, and distant detection labeling (run 1, 2 and 6) the MFCC is equally accurate as the Mel spectrogram. For the runs with an increase window size (run 7,8 and 9), the spectrogram is slightly better then the MFCC.

Moreover, a ROC curve with the binary discrimination threshold based on the pure energy of the signal is shown in Figure 9. This curve is used to see whether the model just checks the amount of energy in the signal or if it uses more elaborate features. The AUC gives away directly that the performance is significantly worse than the CNNs, so the model does not base its outputs simply on the amount of energy in the signal. Especially in the low FPR region (< 0.1) the TPR is significantly lower than for the CNNs.

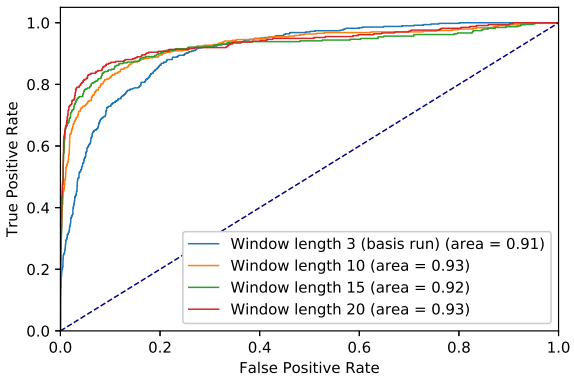
4.6 Visualization of the output

In order to clarify the output of the model, one of the runs is used to visualize the outputs. In Figure 10, the spectrogram of one sample of the basis run test set is shown, along with the expected label (in red), the output of the network (in black) and the binary discrimination threshold belonging to a FPR of 0.1 (in purple). This example shows a decent detection result in which the results in the time window for which the label is 1 (between 28 and 40 seconds) is correctly above the threshold (except for the first second). The rest of the output is always under the threshold and therefore not detected as an aircraft.

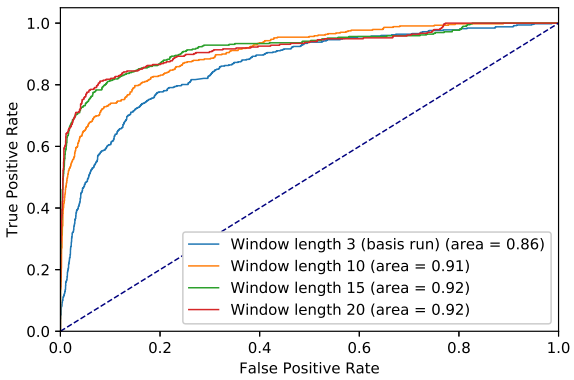
The correctness of the result of Figure 10, however, is not observed for all cases of the test set. False positives and false negatives are appearing as well, such as shown in Figure 11. In this figure the time span between 30 and 45 seconds should be given a label of 1, but but the model output is still under the threshold, except for 1 second. Also, the point at second 3 is just above the threshold, whereas it should be labeled 0. On the other hand, also for the human eye the presence of an



(a) MFCC-CNN for different window lengths.



(b) Melspectrogram-CNN for different window lengths.



(c) Spectrogram-CNN for different window lengths.

Figure 8: ROC curves showing the influence of the window lengths for each feature. In general, the increase in window length increases the performance, but it converges to the performance of a window length of 20 seconds.

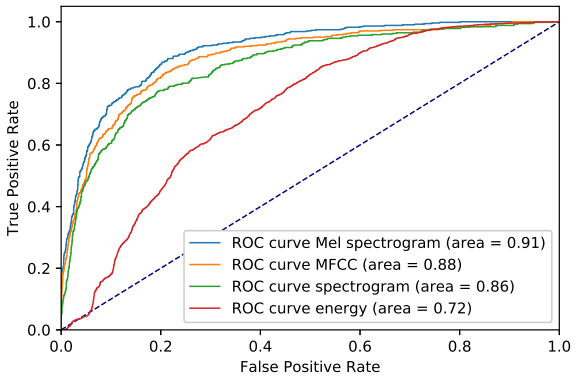


Figure 9: ROC curves of each feature for the basis run. Also the energy of the signal is used as an input for the ROC curve to show that the model does not base its output only on the energy in the signal. The Mel spectrogram is the best performing feature, MFCC second best, the spectrogram is the worst feature and energy performs significantly worse than all features.

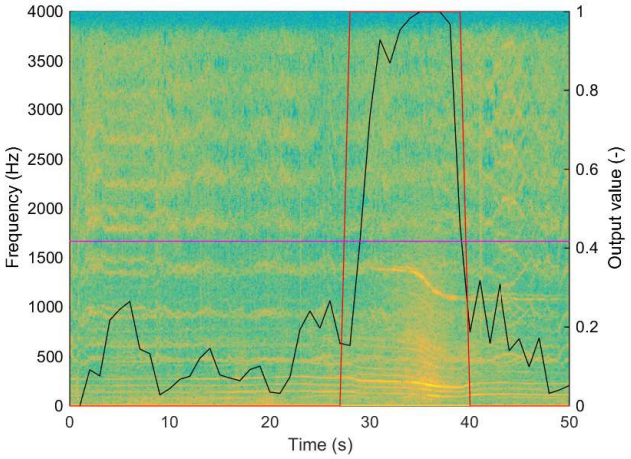


Figure 10: Correct classification example of a sound sample. In red is the expected label, in black the given output and in purple the discrimination threshold. The left axis belongs to the spectrogram only, the right axis belongs to the output, the label and the threshold lines. As the output is always under the purple line when the label is 0 and above the purple line when the label is 1 (except for 1 second), this sample is accurately classified.

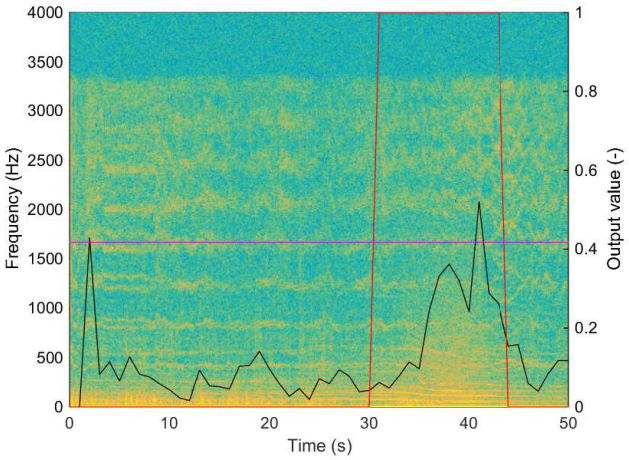


Figure 11: Partly wrong classification example of a sound sample. In red is the expected label, in black the given output and in purple the discrimination threshold. The left axis belongs to the spectrogram only, the right axis belongs to the output, the label and the threshold lines. A false positive is shown at 3 seconds and false negatives between 30 and 45 seconds (except second 40).

aircraft is better visible in the spectrogram of Figure 10 than in the spectrogram of Figure 11, due to the Doppler shift and the increase in energy (which can be seen by the increase of the yellow content) in Figure 10.

In order to confirm that the model can recognize Closest Point of Approach (CPA) such as shown in the spectrogram, all the audio samples of the test set of the basis run are centered around the CPA (if any). For each second in the range of 10 seconds before the CPA and 10 seconds after the CPA, the mean values and standard deviation of the model output are taken. Those values are shown in Figure 12. Each dot represents the value of the mean, each bar the standard deviation from the mean. This figure shows that at the CPA, the output value is usually the highest. Furthermore, the larger the time distance from the CPA, the lower the mean and standard deviation. There is, however, relatively much spread in the output of the network.

4.7 Precision and recall

The AUC gives a good overall indication for the accuracy of the model. However, in order to see how well the model performs per point on the ROC curve, precision and recall is used. Precision is defined in Equation 7, in which FP is the number of false positives and TP is the number of true positives. For recall, also the false negatives FN are used, such as shown in Equation 8.

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

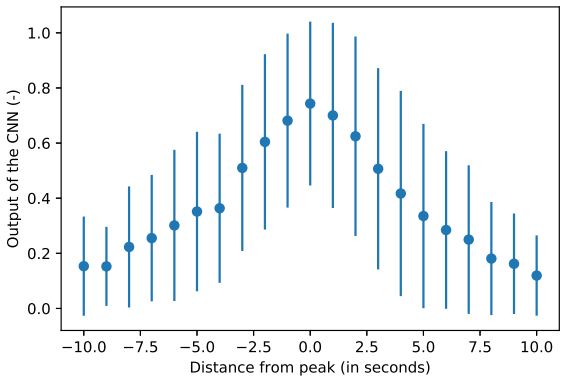


Figure 12: Means (dots) and standard deviations (bars) per time distance from the center of a CPA in the spectrogram. It shows that the closer the aircraft is, the better the detection performance.

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

In this research, an important value is $1-recall$ for the label 0. This value shows how many false positives are present, so how often the UAV would falsely perform an avoidance maneuver. The recall for the label 1 is the second most important. It shows how well the aircraft is detected when it is present. The reason that it is less important than the $1-recall$ for label 0 is because this value does not say when the false negatives appear. It is expected that the closer the aircraft gets, the better the detection performance. Figure 12 shows that this is actually the case for this model. So if the model does not detect the aircraft it is probably not too close, so it would not directly lead to a critical situation. Precision shows how many of the predicted labels are relevant, which is less important for this application than the recall.

An example of the precision and recall and the confusion matrix for the Mel spectrogram-CNN with the window length 20 are shown in Tables 4 and 5 respectively. As a very low FPR beneficial, but still aircraft should detected, the point on the curve for which the ROC curve just separates from the Y-axis is chosen (which is around an FPR of 0.01 and a TPR of 0.7).

Table 4: Precision and recall of the Mel spectrogram-CNN using window length 20.

	Precision	**Recall**
0	0.97	0.99
1	0.85	0.70

Table 5: Confusion matrix of the Mel spectrogram-CNN using window length 20.

		Predicted class	
		0	1
Actual class	0	2823	42
	1	101	234

5 Discussion

The results shown in section 4 are further discussed in this section. Starting with the different UAV/aircraft amplitude ratios, Figure 5 shows in the lower FPR region an expected trend, which is that the lower the UAV amplitude is compared to the aircraft amplitude, the better the aircraft is detected. That means, in order to use this model for real-world application, it is best to diminish the UAV's ego-sound as much as possible, for example by means of the method of Marmaroli et al. [8].

The addition of third party database recordings also improves the performance, such as shown in Figure 6. Those recordings consist of different background noise, which could be easier for the model to distinguish from the typical background noise from the Lelystad recordings. The basis run performed better in the very low FPR (< 0.01), but the corresponding TPR is to low to be a good detector.

The fact that the different type of labeling performs worse, which is shown in Figure 7, is unexpected. The labels that are 1 for the distant detection labeling consist of the ones from nearby detection labeling plus some extra ones before and after. In other words, the nearby detection labels are a part of the distant detection labels. As the distant detection labeling includes the nearby detection labels, it is expected that training with distant detection labeling at least performs the same as training with nearby detection labeling. However, the model performs worse (or equal, for any FPR lower than 0.05) which means that there is no benefit in using the distant detection labeling. The consequence of using nearby detection labeling over distant detection labeling is that the aircraft is closer to the UAV when it is detected.

The trends shown in Figure 8, at which the window length is increased, are not unexpected. The longer the window length, the more information the model uses to make a decision and therefore the performance is better. This only works up to a certain amount since sound information to far in the past can have nothing to do with the present sound. Based on the presented experiments, a window length between 15 and 20 seconds should be used to be as accurate as possible. Choosing a value above 20 seconds will not increase the performance and makes it computationally more expensive. Of course, also other forms of memory can be explored, such as Long Short Term Memory [20] or GRU [21].

In the ideal situation, no false positives or false negatives are present in the output of the detector. Since the ROC curves

in Figures 5, 6, 7 and 8 never have an AUC of 1, this is not possible. Therefore, we aim to have as little false positives and false negatives. In Table 4 and Table 5 a limit of one false positive in 100 seconds is set. If after a false positive a warning is send to the operator, once in 100 seconds he/she has to check whether there is really other air traffic present, which is not increasing the workload to much and therefore once in a 100 seconds is a reasonable limit. If the UAV has to descend (or even land) after a detection, a false positive once in a 100 seconds is too much, so for those cases a filter should be applied, which checks whether multiple positive detections are found in a short-time frame. The percentage of missed detections corresponding to a once in a 100 FPR, is 30%. Luckily, Figure 12 shows that the closer the aircraft is the better the accuracy, so the missed detections will be mostly appear in the early stages of the detection.

Alongside the conclusions drawn from the results, there are a few general comments to be made concerning the research method.

Firstly, the data set should be extended. The data set used in the basis case (run 3) only contains the recordings from Lelystad airport. This data set has in total 84 flyovers. The data augmentation increases the data set times four, so 336 flyovers are available for the data set. This is considered a relatively small data set for machine learning purposes such as this research. For comparison, ImageNet[8], a famous data set for image recognition, has 15 million examples in total. In addition, the ratio of the data set that includes aircraft sound and that only includes background noise is not 50/50, due to the fact that the cut-outs from the recordings are random. The ratio aircraft/background in this data set is approximately 20/80. The problem with this ratio is that the model could classify all the sound samples as background noise and still would have an accuracy of 80%. Another comment about the data set is that it is artificially mixed, so the UAV and aircraft sound are individually recorded. In the spectrogram, it is visible where the aircraft sound is added to the UAV sound by vertical lines at the stop and start. An example is shown in Figure 13, at which the aircraft recording part stops at 30 seconds. In order to avoid this effect, recordings should be taken on a UAV, which flies close to flying aircraft.

So far, the only different scale used is the Mel scale. Two features use this scale which mimics the way humans perceive frequency. The comparison of the Mel spectrogram and the spectrogram in Figure 9 shows that stretching the lower frequencies works well in combination with the CNN. One idea is to make a scale that stretches the lower frequencies even more. As most of the distant aircraft sound lies in the low frequency region, further stretching the lower frequencies could show more important low frequency sound information for the CNN.

What is more, is that there is not much difference in type

[8] http://www.image-net.org/

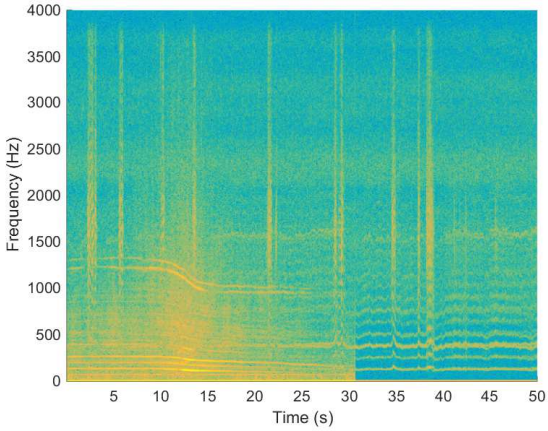


Figure 13: Spectrogram of a mix of UAV and aircraft sound. The end of the aircraft sound recording is visible on the spectrogram at 30 seconds by the vertical line (which is the sudden decrease in energy).

of background noise. Only two types of microphones are used, the 808 micro camera microphone and the microphone from the array. Different microphones could show different noise content. Further research in the quality of the microphones is demanded. Also, the background noise is pretty constant during the recordings, whereas on a flying UAV this could differ considerably. Other background noise, such as cars, trains, lawnmowers, etc., is not added.

Not only is there one composition of background noise, but also only one type of UAV sound has been used. In order to make a model for versatile applications, multiple UAV sounds should be included in the data set. If the model is applied to only one UAV, it is useful to use its specific model in training the detection network. In this process it is also important to check whether the ego-noise of the UAV is in the same order of loudness as the Parrot Bebop used in this research.

6 CONCLUSION

Detection of air traffic sounds on a UAVs could increase the safety of the airspace. This paper builds on existing sound features and classification methods, but this time applied to combined UAV and aircraft sound.

The three features used are the MFCC, spectrogram and Mel spectrogram, which are the input to a CNN classifier. The best performance of the model is obtained using the Mel spectrogram, which moves over the sound recording with a 20-second window length. The detection performance increases when the aircraft is closer to the UAV. Longer time windows give better performance up until a certain window length, but also decrease the potential reaction time for an avoidance maneuver. Secondly, the model works best if as little UAV sound is present as possible. Thirdly, the cur-

rent method still gives too many false positives for real-world application. Improvements may be expected from a better filtering over time (ignoring solitary peaks of the network's output), a more extensive data set, and potentially additional information such as the commanded RPMs of the UAV's propeller(s). Finally, a more realistic data set should include sound recordings of aircraft taken from a (moving) UAV.

REFERENCES

[1] E Tijs, GCHE de Croon, J Wind, B Remes, C De Wagter, HE de Bree, and R Ruijsink. Hear-and-avoid for micro air vehicles. In *Proceedings of the International Micro Air Vehicle Conference and Competitions (IMAV), Braunschweig, Germany*, volume 69, 2010.

[2] Hans-Elias De Bree and Guido De Croon. Acoustic vector sensors on small unmanned air vehicles. *the SMi Unmanned Aircraft Systems, UK*, 2011.

[3] Meysam Basiri, Felix Schill, Pedro U. Lima, and Dario Floreano. Robust acoustic source localization of emergency signals from Micro Air Vehicles. *IEEE International Conference on Intelligent Robots and Systems*, pages 4737–4742, 2012.

[4] Meysam Basiri and Felix Schill. Audio-based Relative Positioning System for Multiple Micro Air Vehicle Systems. *Robotics: Science and Systems*, (266470), 2013.

[5] Meysam Basiri, Felix Schill, Dario Floreano, and Pedro U Lima. Audio-based localization for swarms of micro air vehicles. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4729–4734. IEEE, may 2014.

[6] Meysam Basiri, Felix Schill, Pedro Lima, and Dario Floreano. On-Board Relative Bearing Estimation for Teams of Drones Using Sound. *IEEE Robotics and Automation Letters*, 1(2):820–827, 2016.

[7] Brendan Harvey and Siu O'Young. Acoustic Detection of a Fixed-Wing UAV. *Drones*, 2(1):4, jan 2018.

[8] Patrick Marmaroli, Xavier Falourd, and Hervé Lissek. A uav motor denoising technique to improve localization of surrounding noisy aircrafts: proof of concept for anti-collision systems. In *Acoustics 2012*, 2012.

[9] Huy Phan, Lars Hertel, Marco Maass, and Alfred Mertins. Robust audio event recognition with 1-max pooling convolutional neural networks. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 08-12-Sept(1):3653–3657, 2016.

[10] Ian McLoughlin, Haomin Zhang, Zhipeng Xie, Yan Song, Wei Xiao, and Huy Phan. Continuous robust sound event classification using time-frequency features and deep learning. *PLoS ONE*, 12(9):e0182309, sep 2017.

[11] Haomin Zhang, Ian McLoughlin, and Yan Song. Robust sound event recognition using convolutional neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2015-Augus, pages 559–563. IEEE, apr 2015.

[12] S. Doljé. Quantifying microphone array directivity. Master's thesis, Delft University of Technology, dec 2017.

[13] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25, 2015.

[14] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *ISMIR*, volume 270, pages 1–11, 2000.

[15] Richard F. Lyon. Machine hearing: An emerging field. *IEEE Signal Processing Magazine*, 27(5):131–136, 2010.

[16] Fang Zheng, Guoliang Zhang, and Zhanjiang Song. Comparison of different implementations of mfcc. *Journal of Computer science and Technology*, 16(6):582–589, 2001.

[17] François Chollet. Keras. `https://keras.io`, 2015.

[18] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Tara N. Sainath, Oriol Vinyals, Andrew Senior, and Hasim Sak. Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2015-Augus, pages 4580–4584. IEEE, apr 2015.

[21] Emre Cakir, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6):1291–1303, jun 2017.

Multi-UAV Specification and Control with a Single Pilot-in-the-Loop

Patricio Moreno*[1], Santiago Esteva[1], Ignacio Mas[3,4], and Juan I. Giribet[1,2,3]

[1]GPSIC - Facultad de Ingeniería, Universidad de Buenos Aires, Argentina
[2]Instituto Argentino de Matemática "Alberto Calderón" (IAM)
[3]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina
[4]Instituto Tecnológico de Buenos Aires, Argentina

ABSTRACT

This work presents a multi-unmanned aerial vehicle formation implementing a trajectory-following controller based on the cluster-space robot coordination method. The controller is augmented with a feed-forward input from a control station operator. This teleoperation input is generated by means of a remote control, as a simple way of modifying the trajectory or taking over control of the formation during flight. The cluster-space formulation presents a simple specification of the system's motion and, in this work, the operator benefits from this capability to easily evade obstacles by means of controlling the cluster parameters in real time. The proposed augmented controller is tested in a simulated environment first, and then deployed for outdoor field experiments. Results are shown in different scenarios using a cluster of three autonomous unmanned aerial vehicles.

1 INTRODUCTION

Unmanned autonomous systems, in general, is a topic of interest that has been growing steadily for some time. This raises from the diversity of applications in which these systems can be used. Examples of this are search and rescue missions [1], inspection of hazardous environments, goods delivery or object transportation, military and surveillance purposes, among others. Unmanned aerial vehicles (UAVs) are of special interest because of advances in technology, that have reduced cost and boosted the capabilities of all UAVs, particularly multicopters. This has raised the interest in formation control of multi-agent systems within academic and industry communities.

The theory used to design the control laws for these architectures feeds from different fields, such as game theory [2], biology [3, 4] or classic manipulator kinematic chains [5], [6]. Techniques derived from these studies include potential fields [7], behavioral primitives [8], swarm-like structures [9, 10], and leader-follower configurations [11].

*Email address: pamoreno@fi.uba.ar

All these techniques control a multi-agent system to operate in a cooperative fashion. Working with multiple unmanned aerial vehicles, the particular formation used in this work involves spatial constraints and impose physical limitations, such as communications range. Oh et al. [12] gave a detailed review on formation control and Yanmaz et al. [13] analyzed the communication network aspects of a formation.

In [5] Mas et. al. presented a cluster-space formulation for the coordinated control of a group of robots. The goal of the cluster-space approach is to promote the simple specification and monitoring of the motion of a multirobot mobile system, exploring a specific approach for formation control applications. This method considers the multirobot system as a single entity, or cluster, and desired motions are specified with respect to cluster attributes, such as position, orientation, and geometry. These attributes are the state variables that form the cluster space of the system. The method is flexible in the sense that these variables can be selected in different ways, favoring specific tasks or alternative implementations such as centralized or distributed control architectures [14]. Previous works showed results, both simulated and in real scenarios, for unmanned ground vehicles (UGVs)[5, 14] and autonomous surface vessels (ASVs) [15, 16], among others.

In this work we introduce a cluster space controller, where a feed-forward component is added to modify the trajectory in-flight. This formulation allows to naturally modify the position and geometric properties of the cluster in a way that enables a simple formation tele-operation by a single human pilot, using an intuitive remote control interface to command the motion of the formation. An alternative to this approach would be the specification of the trajectory of each vehicle or the relative position of each vehicle with respect to a neighbour.

To illustrate the benefits of such an architecture, in a task where multiple UAVs cooperatively transport a load, as in [17], if a tele-operated group of vehicles needs to pass through a narrow passage while keeping the load distribution constant, it may be of interest to momentarily modify the distance between vehicles without changing their spacial relative configuration. A single specification change such as "change the formation size" that can be commanded by an operator keeps the operation simple, regardless of the under-

lying complexity of the individual vehicles' motions.

Another benefit of a pilot-in-the-loop control arises when a multi-agent system is used for automated inspection. For example, electric power distribution lines may be located in areas of difficult access and unmanned vehicles can be used for inspection tasks [18]. The tele-operator may need an additional detailed view of a portion of a tower or cable, modifying a pre-loaded trajectory in-flight. Oil pipeline inspection [19] or civil engineering projects such as bridges or skyscrapers may also benefit from this approach.

This work is organized as follows. Section 2 shows the formation definition while the controller is described in section 3. The results of computer simulation and our experimental testbed are shown in section 4. Finally, section 5 draws the conclusions.

2 CLUSTER-SPACE FORMULATION

Cluster-space control [20] represents the state of a system as an articulated kinematic mechanism. The cluster is defined using variables which fully represent the pose and geometric structure of the formation. First, a cluster frame $\{C\}$ to represent the formation pose is defined. Then, each robot's pose, $\mathbf{r}_i \in \mathbb{R}^{m_i \times 1}$, is referenced to the $\{C\}$ frame. It is usually desired to define $\{C\}$ in a physically meaningful way, such as at the formation barycenter and oriented towards a particular vehicle. Additional cluster variables capture the formation shape and orientation, fully specifying the total number of degrees of freedom of the group. The formation motion is commonly defined using the cluster-space variables. Because of this, a formal set of kinematic transformations relating cluster-space variables and robot-space variables is needed. A cluster-space controller computes the compensation actions needed for the cluster and, using the defined kinematic transformations, converts the cluster compensation actions into robot compensation actions.

Consider an n-robot system, a cluster, where each of the robots has the same m degrees of freedom (although this is not necessary[1]). Let $\mathbf{r} \in \mathbb{R}^{mn \times 1}$ be a state vector comprised of the n robot poses, and $\mathbf{c} \in \mathbb{R}^{mn \times 1}$ a state vector corresponding to the cluster variables. These states are related through the following forward and inverse position kinematics transforms:

$$\begin{aligned}\mathbf{c} &= \text{FORWARD\_KINEMATICS}(\mathbf{r}) \qquad (1)\\ &= [\text{fwd}_1(r_1,\ldots,r_{mn}),\ldots,\text{fwd}_{mn}(r_1,\ldots,r_{mn})]^\intercal,\\ \mathbf{r} &= \text{INVERSE\_KINEMATICS}(\mathbf{c}) \qquad (2)\\ &= [\text{inv}_1(c_1,\ldots,c_{mn}),\ldots,\text{inv}_{mn}(c_1,\ldots,c_{mn})]^\intercal,\end{aligned}$$

where $\text{fwd}_k(r_1,\ldots,r_{mn})$ is the forward position kinematic equation that relates the kth cluster parameter with the robot poses, and $\text{inv}_k(c_1,\ldots,c_{mn})$ is the inverse position kinematic equation that related the kth robot state parameter with the cluster parameters.

Now, let $\mathbf{J}(\mathbf{r})$ be the jacobian matrix obtained from Equation 1, and $\mathbf{J}^{-1}(\mathbf{c})$, the jacobian matrix obtained from Equation 2, the mapping between the velocities are $\dot{\mathbf{c}} = \mathbf{J}(\mathbf{r})\dot{\mathbf{r}}$ and $\dot{\mathbf{r}} = \mathbf{J}^{-1}(\mathbf{c})\dot{\mathbf{c}}$, respectively.

Using generic kinematic transformations it is possible to envision a diagram of a system being controlled using the cluster-space formulation. Such an architecture is shown in Figure 1.

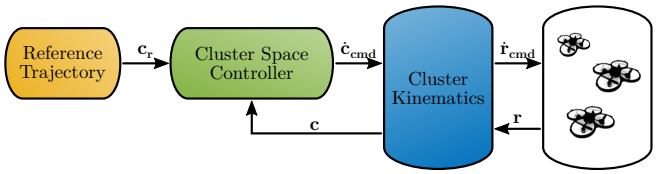


Figure 1: Control architecture for the cluster-space control method.

2.1 *Three-UAV Cluster Space definition*

The three-robot cluster state variables can be defined with the cluster reference frame located at the barycenter of the robots and the remaining variables describe a triangle with side lengths p and q and the necessary angles to articulate it and rotate it. Figure 2 shows all the parameters for the cluster of 3 UAVs. The equations for the forward position kinematics that define the cluster space are the following:

$$x_c = \frac{x_1 + x_2 + x_3}{3}, \qquad (3)$$

$$y_c = \frac{y_1 + y_2 + y_3}{3}, \qquad (4)$$

$$z_c = \frac{z_1 + z_2 + z_3}{3}, \qquad (5)$$

$$\theta_c = -\arctan\left(\frac{2x_1 - x_2 - x_3}{2y_1 - y_2 - y_3}\right), \qquad (6)$$

$$\rho_c = -\arctan\left(\frac{z_1 - z_c}{\sqrt{(x_1 - x_c)^2 + (y_1 - y_c)^2}}\right), \qquad (7)$$

$$\gamma_c = -\arctan\left(\frac{z_2 - z_3}{|x_2 - x_3|}\right), \qquad (8)$$

$$p = \tfrac{1}{2}\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}, \qquad (9)$$

$$q = \tfrac{1}{2}\sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2 + (z_1 - z_3)^2}, \qquad (10)$$

$$\beta = \arctan\left(\frac{(x_3 - x_1)\sin\alpha - (y_1 - y_3)\cos\alpha}{(x_3 - x_1)\cos\alpha + (y_1 - y_3)\sin\alpha}\right), \qquad (11)$$

where $\alpha = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right)$. Also, each UAV heading angle is a cluster parameter by itself, defined as the heading offset with respect to the cluster yaw angle. They have been omitted in the formulation above for simplicity.

Considering the heading angle of the UAVs, there are 12

[1] Considering an n-robot system where each robot has $m_i, i = 1,\ldots,n$ degrees of freedom, then the state vector $\mathbf{r}$ has $\sum_{i=1}^{n} m_i$ components.

cluster state variables for a formation of 3 stabilized UAVs, each with 4 degrees of freedom.

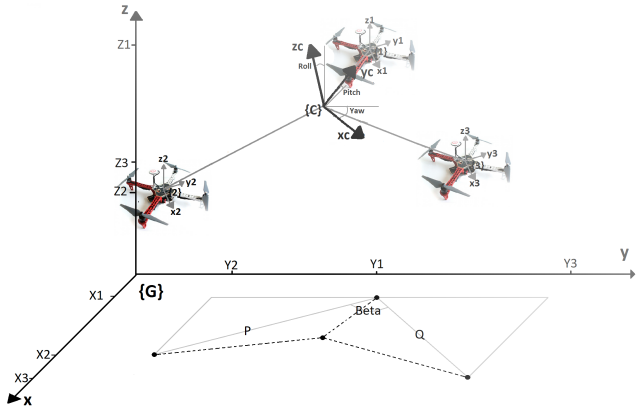


Figure 2: Cluster parameters definition for a formation of three UAVs.

3 Cluster-Space Controller

As shown in Figure 1, a classic PID controller was added for trajectory tracking. This controller receives the cluster state errors (or the cluster state reference and cluster state pose and computes the error) and generates a cluster state velocity control signal, using different proportional, integral and derivative gains for each cluster state variable. The PID output control signal is then multiplied by the inverse jacobian matrix to generate the compensation signal to be applied to each UAV.

To add the remote control operation, the addition of a feed-forward controller is proposed. This controller adds an external signal to the system. The external signal is a velocity command that can be readily sent to the cluster formation. It also modifies the trajectory to take into account the commanded velocity and integrates its value over time to modify the cluster space reference trajectory accordingly. Figure 3 shows a complete block diagram of the implemented controller.

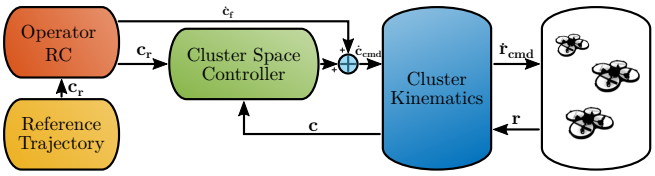


Figure 3: Implemented control scheme for the cluster-space control.

4 Results

The proposed system was first validated using a simulation environment. This environment consists of a computer running the XUbuntu 16.04 operating system with ROS Kinetic, the Robot Operating System, and Gazebo 7, a robot simulator. The multicopters are simulated using the firmware of the PX4 autopilot—version 1.5.1—, their Gazebo plugins, and a model of the IRIS drone from 3D-Robotics.

The experimental testbed consists of three commercially available UAVs built with DJI F450 frames, the Pixhawk 1 autopilot (FCU) from 3DR with the PX4 flight stack, and one Raspberry Pi 3B (RPi) for a pair of UAVs, the remaining one uses a 915 MHz link. Figure 4 shows a picture of one the UAVs with an RPi on top of the FCU. The communications network was build using a WiFi router, connecting all computer to it.

Figure 4: UAV with onboard computer and wifi link used for field experiments

The interface with the autopilot was through a mavros ROS node. The trajectory generator, the cluster kinematic equations and the controller were developed as ROS nodes, in the python programming language. The operator remote control was a gaming joystick with 14 buttons and 2 analog sticks.

The experiment consisted on a triangular-shaped formation having a predefined trajectory that would make one or more of the UAVs collide with objects placed in the environment. For this situation, two scenarios were proposed to overcome the conflict:

1. the formation changes it shape, becoming a line as it passes between the object, and

2. the formation scales down its size, maintaining its triangular shape as it moves between the obstacles.

In neither case, a collision avoidance maneuver is included *a priori* in the trajectory to evade the obstacles, nor an inter-vehicle collision avoidance; obstacle negotiation solely depends on the operators' commands executed on run-time.

Considering a position estimation error from the pixhawk's estimator of about $1.5\,\mathrm{m}$, with a standard deviation of $0.8\,\mathrm{m}$, a similar error is expected for the cluster's centroid, while it could be greater for the distance-based parameters.

If, for a test, the reference trajectory is followed with an error within the expected parameters, the result of the test is considered to be successful.

4.1 Simulation results

In both simulation scenarios the cluster has the same initial position: $z_c = 5\,\mathrm{m}, p = 7.1\,\mathrm{m}, q = 7.1\,\mathrm{m}, \beta = 60°$ and all other parameters with a zero value. The obstacles, of $20\,\mathrm{m}$ height, are placed at $(-4\,\mathrm{m}, 6\,\mathrm{m}, 0\,\mathrm{m})$ and $(-4\,\mathrm{m}, 6\,\mathrm{m}, 0\,\mathrm{m})$.

To evade the obstacles by switching from a triangle to a line, the varying parameters are p and β, while y_c vary just to go through the obstacles. This variation is shown in Figure 5.

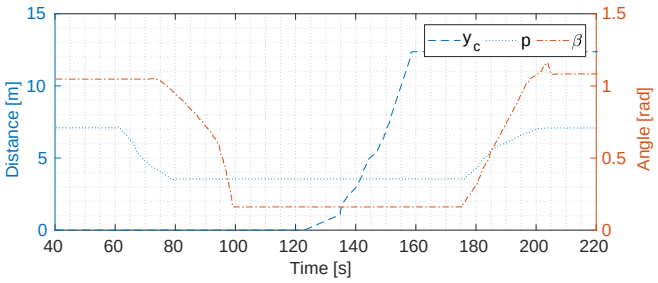


Figure 5: 3 UAV cluster varying parameters while evading obstacles by switching from a triangle to a line formation.

The obstacles positions and the cluster motion, on an XY plane, for the aforementioned cases are shown in Figure 6 and Figure 7. The first figure shows the trajectory while switching from a triangle to line, while the latter shows the formation while changing the triangle size.

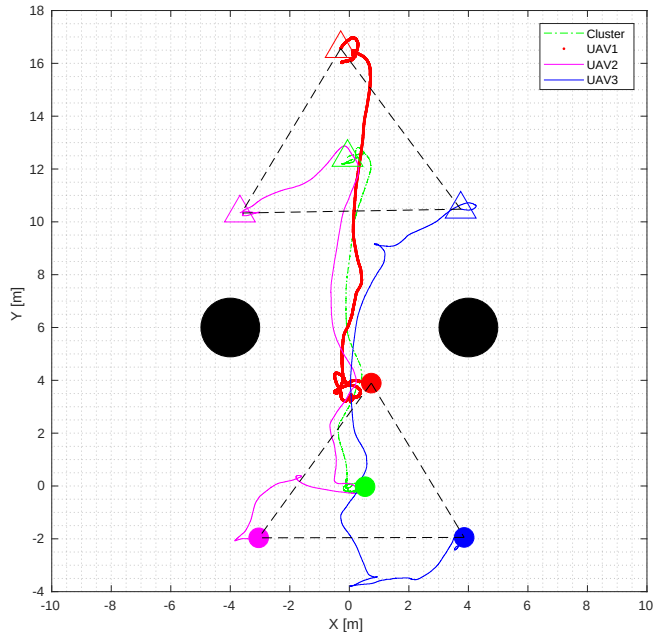


Figure 6: XY movement of the simulated 3 UAV cluster

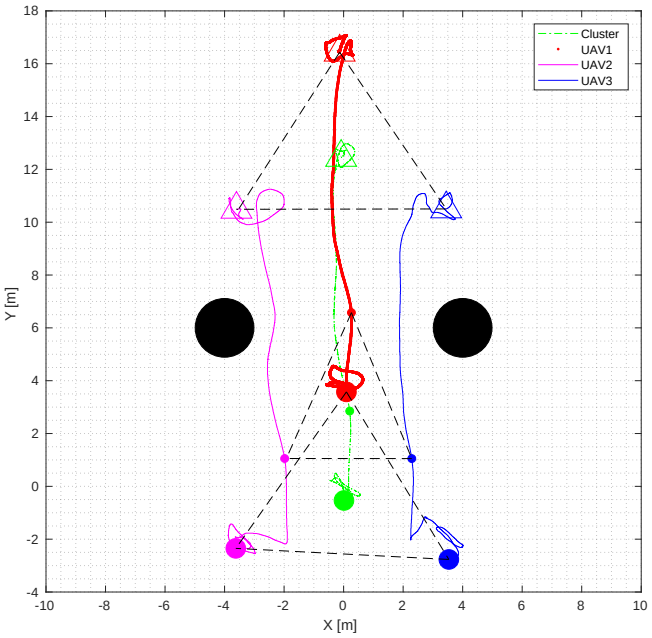


Figure 7: XY movement of the simulated cluster maneuvering through the obstacles reducing the area of coverage of the formation

Figure 8 shows the cluster state errors while maneuvering as a line formation. It can be seen that as soon as β approaches 0, the error of the roll parameter, γ_c, increases as there is a singularity when the agents are co-linear. Another error of importance can be seen for y_c near $t = 160\,\mathrm{s}$, which is due to fast varying parameters and a relative slow system response.

Figure 13 shows the cluster state errors while maneuvering as a triangle formation. It can be seen that the formation goes between the obstacles, staying further away of the singularities. This property gives the controller a better performance.

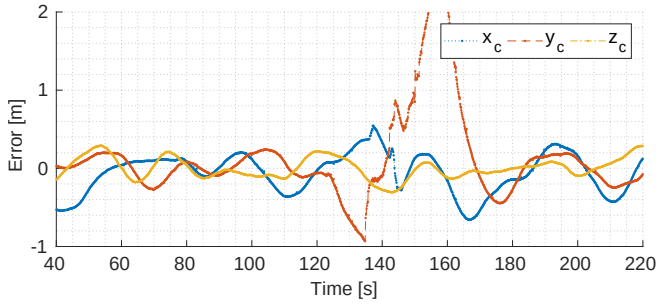
4.2 Experimental results

In these scenarios the cluster has the initial position: $z_c = 3\,\mathrm{m}, p = 7.1\,\mathrm{m}, q = 7.1\,\mathrm{m}, \beta = 60°$ and all other parameters with a zero value. The obstacles were at $(-6\,\mathrm{m}, -7\,\mathrm{m}, 0\,\mathrm{m})$ and $(2\,\mathrm{m}, -7\,\mathrm{m}, 0\,\mathrm{m})$.

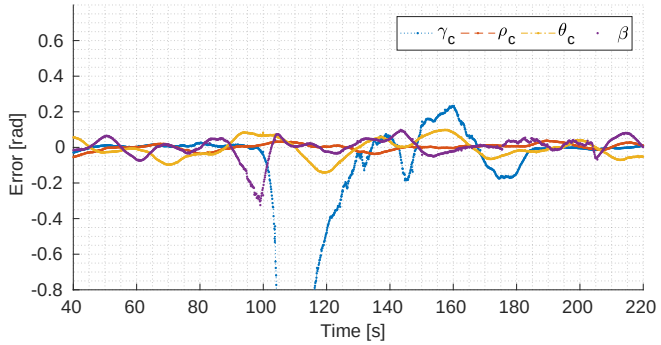
As in the simulation, for the first scenario the varying parameters are p and β, while y_c vary just to go through the obstacles. This variation is shown in Figure 9.

The obstacles positions and the cluster motion, on an XY plane, for both scenarios are shown in Figure 10 and Figure 11. The first figure shows the trajectory while switching from a triangle to line, while the latter shows the formation while changing the triangle size.

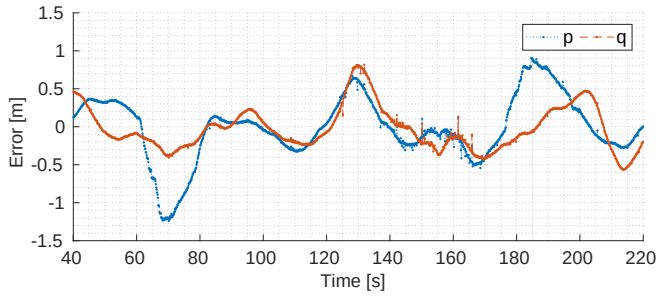
Figure 12 shows the cluster state errors while maneuvering as a line formation. It can be seen that β again approaches 0 and γ_c error increases as in the simulation case.



(a) Cluster position (x_c, y_c, z_c) error



(b) Cluster orientation $(\gamma_c, \rho_c, \theta_c, \beta)$ error



(c) Cluster vehicle distances (p, q) error

Figure 8: Cluster errors of a simulation using a joystick to control the formation (line shape obstacle avoidance)

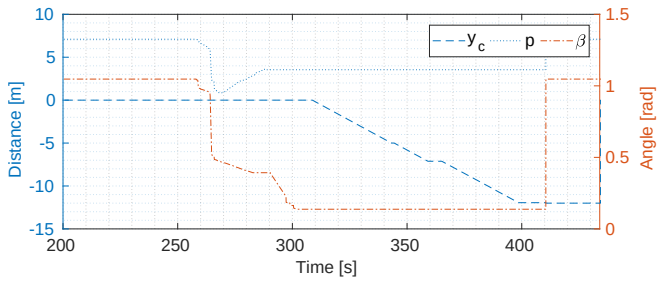


Figure 9: 3 UAV Cluster varying parameters while evading obstacles as a line formation during a field experiment.

The cluster state errors while maneuvering as a triangle formation, shown in Figure 14, present analogous results to those of the simulation.

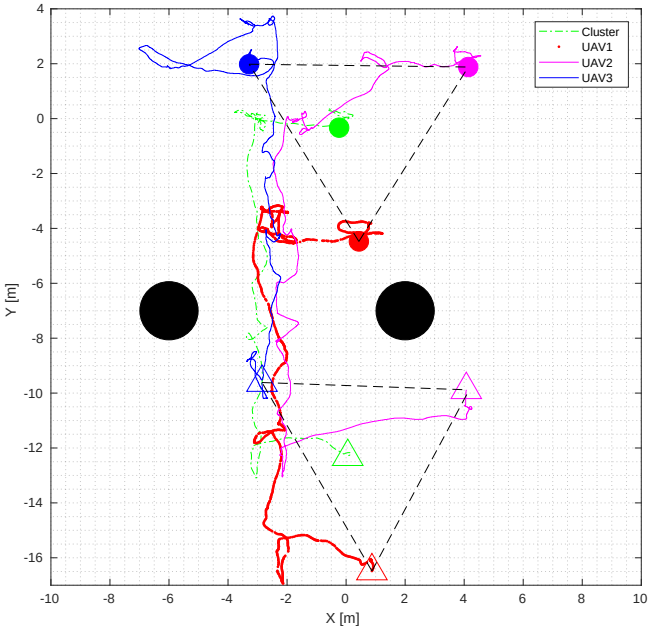


Figure 10: 2D movement of the real cluster evading the obstacles by switching the formation shape into a line

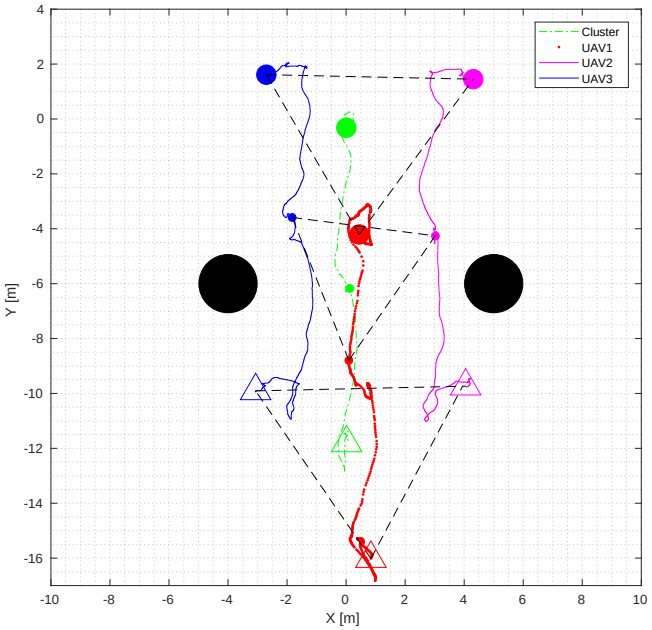
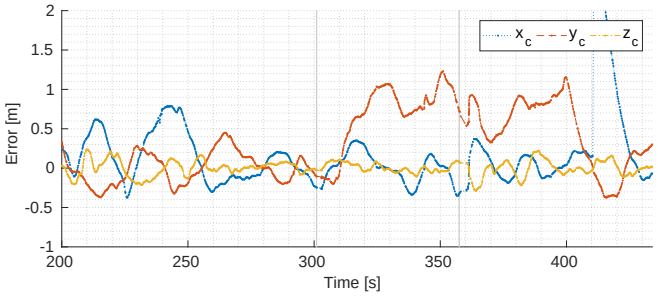


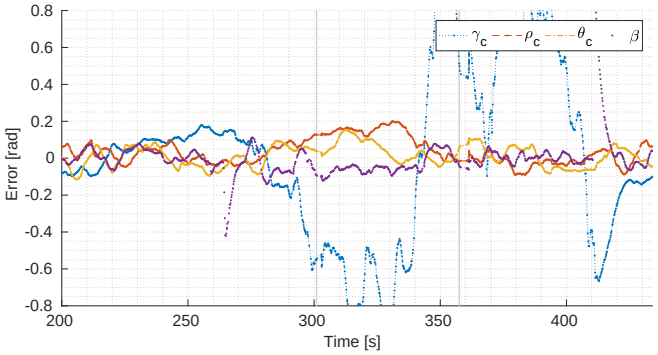
Figure 11: XY movement of the cluster maneuvering through the obstacles reducing the area of coverage of the formation (field experiment)

5 Conclusion

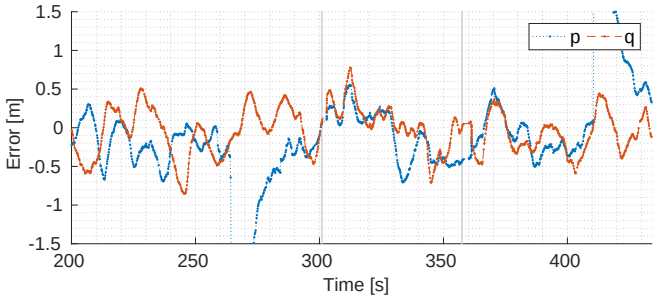
This work presented a cluster space controller with pilot-in-the-loop capability to allow for run-time actuation at the formation level, which provides the ability to modify a predefined trajectory to execute maneuvers such as collision avoid-



(a) Cluster position (x_c, y_c, z_c) error

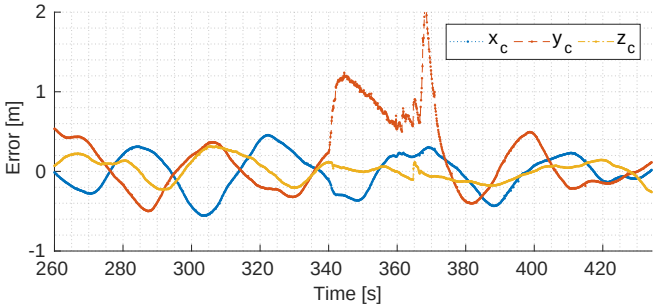


(b) Cluster orientation $(\gamma_c, \rho_c, \theta_c, \beta)$ error

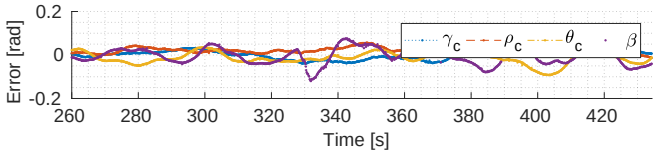


(c) Cluster vehicle distances (p, q) error

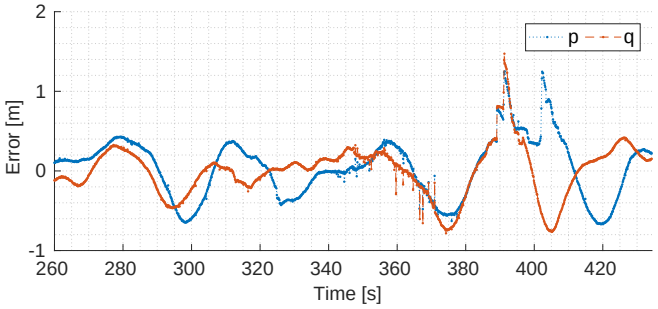
Figure 12: Cluster errors for an outdoor experiment using a joystick to control the formation (line shape obstacle avoidance)



(a) Cluster position (x_c, y_c, z_c) error



(b) Cluster orientation $(\gamma_c, \rho_c, \theta_c, \beta)$ error



(c) Cluster vehicle distances (p, q) error

Figure 13: Cluster errors for a simulation experiment using a joystick to control the formation (triangle shape obstacle avoidance)

ance. The proposed architecture was applied to a formation of three UAVs. By means of computer simulations and outdoor experiments the controller was shown to work and to be adequate for the case study applications. It was also shown that the multi-UAV formation could be intuitively operated using a single remote control, meaning that the operator can command the cluster as a whole in an abstracted manner that does not require to focus on the motions of the individual vehicles. Although the specification shows adequate results, this approach could be improved by adding an inter-vehicle collision avoidance mechanism, such as restrictions to cluster parameters or collision avoidance at the vehicle level.

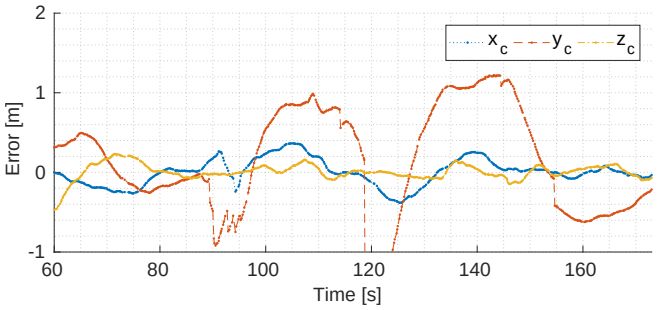
ACKNOWLEDGEMENTS

The authors would like thank the authorities of the School of Agriculture of the University of Buenos Aires for providing access to their campus to conduct the experiments. This work has been sponsored through the UBA-PDE-18- 2019 grant from Universidad de Buenos Aires, Argentina, Instituto Tecnolgico de Buenos Aires ITBACyT Interdiciplinario DT13/2018, and PICT 2016-2016 grant from Agencia Nacional de Promoción Científica y Tecnológica (Argentina). The experiments and research were done with a fellowship from the Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) from Argentina.

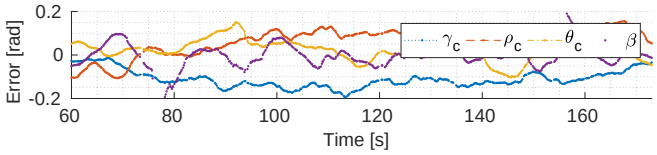
REFERENCES

[1] FJ Perez-Grau, R Ragel, F Caballero, A Viguria, and A Ollero. Semi-autonomous teleoperation of uavs in search and rescue scenarios. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1066–1074. IEEE, 2017.

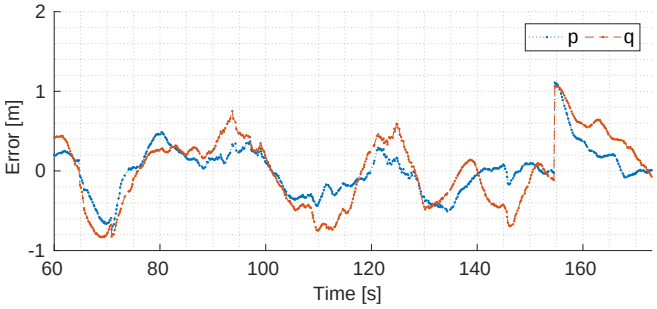
[2] J. Barreiro-Gomez, I. Mas, C. Ocampo-Martinez,



(a) Cluster position (x_c, y_c, z_c) error



(b) Cluster orientation $(\gamma_c, \rho_c, \theta_c, \beta)$ error



(c) Cluster vehicle distances (p, q) error

Figure 14: Cluster errors for an outdoor experiment using a joystick to control the formation (triangle shape obstacle avoidance)

R. Sanchez-Pena, and N. Quijano. Distributed formation control of multiple unmanned aerial vehicles over time-varying graphs using population games. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5245–5250, Dec 2016. doi: 10.1109/CDC.2016.7799072.

[3] V. Kumar, N. E. Leonard, and A. S. Morse. *Cooperative Control: A Post-Workshop Volume, 2003 Block Island Workshop on Cooperative Control*. New York: Springer-Verlag, 2005.

[4] H. M. La, R. Lim, and W. Sheng. Multirobot cooperative learning for predator avoidance. *IEEE Transactions on Control Systems Technology*, 23(1):52–63, Jan 2015. ISSN 1063-6536. doi: 10.1109/TCST.2014.2312392.

[5] I. Mas and C. Kitts. Dynamic control of mobile multirobot systems: the cluster space formulation. *IEEE Access*, 2:558–570, 2014.

[6] I. Mas and C. Kitts. Quaternions and dual quaternions: Singularity-free multirobot formation control. *Journal of Intelligent & Robotic Systems*, pages 1–18, 2016. ISSN 1573-0409. doi: 10.1007/s10846-016-0445-x.

[7] Peng Song and V. Kumar. A potential field based approach to multi-robot manipulation. *Robotics and Automation. Proceedings. ICRA. IEEE International Conference on*, 2:1217–1222, 2002. doi: 10.1109/ROBOT.2002.1014709.

[8] T. Balch and R.C. Arkin. Behavior-based formation control for multirobot teams. *Robotics and Automation, IEEE Transactions on*, 14(6):926–939, Dec 1998. ISSN 1042-296X. doi: 10.1109/70.736776.

[9] C. Belta and V. Kumar. Abstraction and control for groups of robots. *Robotics, IEEE Transactions on*, 20(5):865 – 875, oct. 2004. ISSN 1552-3098. doi: 10.1109/TRO.2004.829498.

[10] Martin Saska, Tomas Baca, Justin Thomas, Jan Chudoba, Libor Preucil, Tomas Krajnik, Jan Faigl, Giuseppe Loianno, and Vijay Kumar. System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization. *Autonomous Robots*, 41(4):919–944, April 2017. ISSN 0929-5593, 1573-7527. doi: 10.1007/s10514-016-9567-z.

[11] Zhao Yunyun, Wang Xiangke, and Zhu Huayong. Multiple uavs configuration formation control via the dual quaternion method. In *Control Conference (CCC), 2015 34th Chinese*, pages 7207–7211. IEEE, 2015.

[12] Kwang-Kyo Oh, Myoung-Chul Park, and Hyo-Sung Ahn. A survey of multi-agent formation control. *Automatica*, 53:424 – 440, 2015. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2014.10.022.

[13] Even Yanmaz, Saeed Yahyanejad, Bernhard Rinner, Hermann Hellwagner, and Christian Bettstetter. Drone networks: Communications, coordination, and sensing. *Ad Hoc Networks*, 68:1 – 15, 2018. ISSN 1570-8705. doi: https://doi.org/10.1016/j.adhoc.2017.09.001. Advances in Wireless Communication and Networking for Cooperating Autonomous Systems.

[14] I. Mas and C. Kitts. Centralized and decentralized multi-robot control methods using the cluster space control framework. *Advanced Intelligent Mechatronics (AIM), 2010 IEEE/ASME International Conference on*, pages 115 –122, July 2010. doi: 10.1109/AIM.2010.5695768.

[15] P. Mahacek, C. A. Kitts, and I. Mas. Dynamic guarding of marine assets through cluster control of automated surface vessel fleets. *IEEE/ASME Transactions*

on Mechatronics, 17(1):65–75, Feb 2012. ISSN 1083-4435. doi: 10.1109/TMECH.2011.2174376.

[16] C. Kitts, P. Mahacek, T. Adamek, and I. Mas. Experiments in the control and application of automated surface vessel fleets. In *OCEANS'11 MTS/IEEE KONA*, pages 1–7, Sept 2011. doi: 10.23919/OCEANS.2011.6106976.

[17] Konstantin Kondak, Aníbal Ollero, Ivan Maza, Kai Krieger, Alin Albu-Schaeffer, Marc Schwarzbach, and Maximilian Laiacker. Unmanned aerial systems physically interacting with the environment: Load transportation, deployment, and aerial manipulation. *Handbook of Unmanned Aerial Vehicles*, pages 2755–2785, 2015.

[18] Gino J Lim, Seonjin Kim, Jaeyoung Cho, Yibin Gong, and Amin Khodaei. Multi-uav pre-positioning and routing for power network damage assessment. *IEEE Transactions on Smart Grid*, 9(4):3643–3651, 2016.

[19] Huang Xiaoqian, Hamad Karki, Amit Shukla, and Zhang Xiaoxiong. Variant pid controller design for autonomous visual tracking of oil and gas pipelines via an unmanned aerial vehicle. In *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, pages 368–372. IEEE, 2017.

[20] I. Mas and C. Kitts. Dynamic control of mobile multirobot systems: The cluster space formulation. *Access, IEEE*, 2:558–570, 2014. ISSN 2169-3536. doi: 10.1109/ACCESS.2014.2325742.

Model Reference Adaptive and Gain Scheduling Control for variable payload UAV Quadcopters

Jorge Diaz Luengo*, Joel Bordeneuve-Guibé, and Francois Defay
ISAE-SUPAERO, Université de Toulouse

Abstract

This work addresses the flight control of a UAV subject to an important and sudden modification of its dynamics during the flight. Namely a UAV Quadcopter with variable payload during a mass drop mission is considered; at least 30% of the nominal weight of the drone is dropped. The control objective is to guarantee the same level of performance whatever the configuration of the UAV: loaded or unloaded. Two adaptive strategies are considered: Direct Model Reference Adaptive Control and Gain Scheduling Control. A feasibility study between both strategies is carried out, alongside a detailed comparison on the relative efficacy and ease of implementation of each. Finally, both controllers are integrated on ISAE-SUPAERO simulation facilities and tested in real flight. The superiority of Model Reference Adaptive Control is proven, not only in terms of dynamic behaviour, but also for its simplicity and robustness. The properties of autotuning and adaptation towards an unknown and disturbed flight make it a valuable solution for the flight control of UAVs.

1 Introduction

Quadcopters applications are growing since many years and the variety of application cases is increasing. In particular load transportation and mass dropping represent new challenging applications and can be used in many fields as rescue [1], delivery, medical assistance [2], agriculture, army... Thanks to the high payload and easily balanced center of mass, quadcopters are often used for transport missions [3]. Many others applications are focused on the use of multiple UAVs [4], a way of carrying heavy payloads with several small UAVs, mainly because of the actual regulation about maximum weight of UAVs.
Variable payload dropping has been addressed in [5] and remains a field with various interesting control problems.
Even if control and guidance of UAVs is an active and open domain of research, non standard configurations generate new needs in terms of performance keeping, recovery and safe flight guaranty. Although classical control methods are generally favoured, the need of more specific schemes able to "adapt themselves" becomes real.
This project is born from the interest of the ISAE Micro Drones Team to implement Adaptive Control algorithms on its competing UAVs, especially in those tasks involving a significant change in mass and/or payload of the UAV. The aim is to design an adaptive controller which is strong enough to properly cope with a drastic change of mass and inertia, while at the same time being flexible enough to be implemented in multiple UAVs.
Over the years, various controllers have been implemented for quadcopters as backstepping [6], Model predictive control [7], adaptive control [8] for different applications. In this paper it has been decided to compare Model Reference Adaptive and Gain Scheduling Control which are two possible candidates to solve the problem.
MRAC is a well known direct adaptive scheme, where the controller parameters are updated without any estimation of an open loop system model, like in indirect adaptive control. MRAC has proven to be very efficient and robust against model mismatches and uncertainties. Morevover it turns out that its implementation is very easy and adapted to demanding real time applications. Besides its adaptibility, the MRA controller can also be designed with a focus on autotuning[9].
On the other hand, gain scheduling control is very popular because basically the overall structure of an existing control scheme is preserved; only the controller parameters are scheduled as a function of an external variable. Many flight control systems rely on this solution. Obviously gain scheduling control belongs to the family of adaptive control.
This paper is organized as follows: in the first section two techniques of adaptive control are briefly presented: direct model reference adaptive control and gain scheduling control. Among their differences, the common factor between both techniques is the willingness of keeping the baseline controller structure already usually used for UAVs flight control. Then a detailed modelization of UAV dynamics is presented together with the global simulation and real-time control environment. The adopted methodology is also detailed and the flight results are shown and analyzed.

2 Adaptive Control

2.1 Direct Model Reference Adaptive Control

An adaptive system can be thought as having two loops (Figure 1): The first one being a classical feedback loop that includes the process and the control law, and the second one

*Email address(es): jorge.diaz-luengo@student.isae-supaero.com, joel.bordeneuve@isae-supaero.fr, francois.defay@isae-supaero.fr

being a parameter adjustment loop which makes it possible to compute the right parameters for the control law. Model Reference Adaptive Control is based on the parametrization of the controller as to obtain a plant whose behaviour mimics that of a reference system [10]. Therefore, the mechanism for adjusting the controller parameters is based on the comparison between the behaviour of the controlled system and of an explicit model reference, as seen in Figure 1 below.

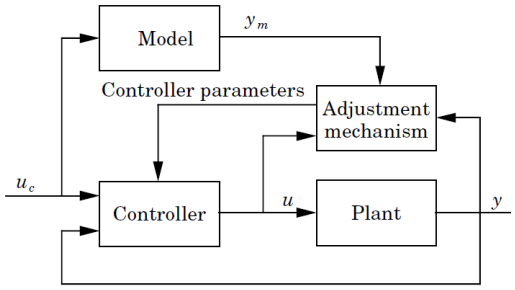


Figure 1: Block diagram of a MRA Controller

Considering the system output y and the reference model output y_m, the adaptive control law will explicitly be based on the output error $e = y - y_m$. For instance, the basic control law for a first order system can be written:

$$u(t) = \theta_1 u_c(t) + \theta_2 y(t) \tag{1}$$

where the controller gains θ_1 and θ_2 are being updated as:

$$\frac{d\theta_i}{dt} = -e\gamma\frac{\partial\theta_i}{\partial e} \qquad i = 1, 2 \tag{2}$$

where γ is an adaptation gain for the so called MIT Rule [11]. For a detailed proof of the convergence, especially Lyapunov Stability, please refer to [11]

2.2 *Gain Scheduling*

Gain Scheduling consists on designing a controller whose parameters change according to the operating conditions, but in a **pre-programmed manner**. This way, GS controllers provide a quick system reaction, whilst being reasonably straightforward to be implemented [11].

The general block diagram of a Gain Scheduling controlled process is seen in Figure 2 below:

Basically, the design of such a controller is performed through two main steps [11]:

1. **Identification of the auxiliary variables**. These are the variables whose dynamics change, affecting the behaviour of the system and demanding flexibility from the controller. In our case, appropriate variables could be the **mass** and **inertia** of the drone, since they quantify the difference between the 2 possible states of the plant: loaded and unloaded.

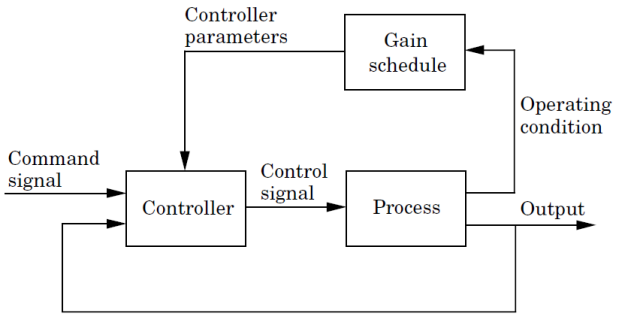


Figure 2: Block Diagram of a Gain Scheduling Controller

2. **Gain Scheduling tables:** These tables gather the different controller tunings, each controller being "optimized" for one state of the system.

3 UAV Modelization and Simulation

The following modelization was based on [7], by using the general equations proposed by [12] as the general equations of Flight Dynamics. The UAV is modelled as an unalterable (rigid) body. Solving its dynamics means computing, for each time instant t the attitude and position of its body frame with respect to an inertial frame (Figure 3).

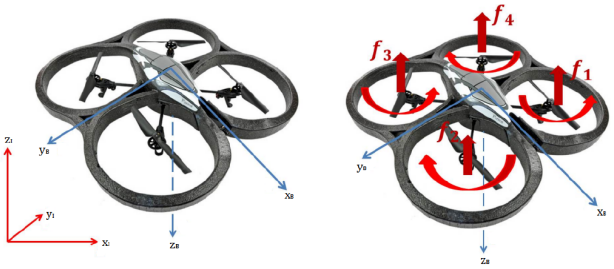


Figure 3: Inertial and Body frames

The aim is hence to compute the **attitude vector $\vec{\eta}^B$ and angular velocities vector $\dot{\vec{\eta}}^B$ expressed in the Body Frame** (^{B}), and the **position vector** $\vec{X}^I$ expressed in the Inertial Frame (^{I}).

$$\vec{X}^I = (x, y, z) \qquad \vec{\eta}^B = (\phi, \theta, \psi) \qquad \vec{\omega}^B = (p, q, r) \tag{3}$$

Now, applying the Rigid Body equations proposed by [12] and [7] and taking $\overline{\overline{I}}$ symmetric in all 3 axis, it comes:

$$\dot{p} = I_x^{-1}(M_\phi + qr(I_y - I_z)) \tag{4}$$

$$\dot{q} = I_y^{-1}(M_\theta + pr(I_z - I_x)) \quad (5)$$

$$\dot{r} = I_z^{-1}(M_\psi + pq(I_x - I_y)) \quad (6)$$

The corresponding equations for the position vector in the inertial frame are then:

$$\ddot{x} = Fm(sin(\phi)sin(\psi) + cos(\phi)sin(\theta)cos(\psi)) \quad (7)$$

$$\ddot{y} = (-sin(\phi)cos(\psi) + cos(\phi)sin(\theta)sin(\psi))Fm \quad (8)$$

$$\ddot{z} = Fm(-g + cos(\phi)cos(\theta) \quad (9)$$

The core structure of the Matlab/Simulink environment is depicted in Figure 4. This 8-block structure, suitable for integration with ROS/OROCOS, enables the environment to feature:

1. **Guidance and navigation indoors** (data given by an Optitrack system), and **outdoors** (GPS, Wifi or similar).

2. **Real time** simulations.

3. **Identical** structure for all drone models.

4. **Same environment** for simulations and flight tests (easy and quick switch between both).

Concerning the controller core design, the adopted strategy is driven by the following principles:

- Decentralized controllers for each axis (no cross coupling).

- Two cascade loops for each axis: **PI controller** for the position error, **P controller** for the velocity error. Derivative terms will not be considered due to Optitrack Noise effects. Hence 3 gains have to be tuned (K_P, K_I, K_{Pv}) per axis.

- **Attitude compensator:** the yaw angle will be kept null at all times.

4 CONTROL DESIGN METHODOLOGY

The mission defined to test and compare each controller covers a 120s flight composed of a first phase with a 400g extra mass (take-off, stabilization, vertical step changes), a second phase (horizontal circular motion) during which the mass is dropped, and a last phase with various vertical step changes and a final landing.

4.1 Model Reference Adaptive Controller

The 3 gains per axis are then adapted using the classical MRAC procedure: we first consider an initial rough tuning of the gains K_P, K_I, and K_{Pv}. Then the applied gains are computed as:

$$K_{Pa} = \theta_P K_P \qquad K_{Ia} = \theta_I K_I \qquad K_{Pva} = \theta_{Pv} K_{Pv} \quad (10)$$

where $\theta_P, \theta_I, \theta_{Pv}$ iare updated using (2).

As for the reference model, the dynamics on each axis have been modelled as a 2^{nd} order system, with the following requirements: **Natural frequency:** $4.5rad/s$, to enable a fast response and **overdamping** ($\zeta = 0.95$) to remove oscillations.

4.2 Gain Scheduling Controller

We have considered two models for the UAV: one with the extra load (before dropping it) and one without it. Thus two controllers have been designed independently in order to reach the aforementioned closed loop 2^{nd} order behaviour. Consequently the total mass of the UAV is the "switching" parameter between both controllers. Then it is critical to estimate in real time the mass of the drone, in order to identify the time at which the mass drop takes place, hence the appropriate switching time between both controllers.

The mass change is estimated from the following:

1. **Vertical velocity Vz**: this will be differentiated to compute the vertical acceleration.

2. **Throttle coefficient**: by using the motor model, the throttle enables calculation of the propeller speeds $w(rad/s)$, and hence, propulsion force $F(Newtons) = P_w * w^2$. The values for the thrust coefficient P_w are known.

The mass estimation is based on the following equation:

$$m(\ddot{z} + g) = F = 4P_w w^2 \quad (11)$$

where there are 2 approaches:

1. **Naive:** divide the force estimation by the $(\ddot{z}+g)$ signal. We obtain a **very reactive estimation**, but prone to error (e.g. the steps in Z are interpreted as mass changes).

2. **Recursive Least Squares:** with forgetting factor $\lambda = 0.98$. We obtain a **slower but more stable estimation**.

The actual estimation combines both ideas, in order to ensure both fast identification of the mass drop and error detection.

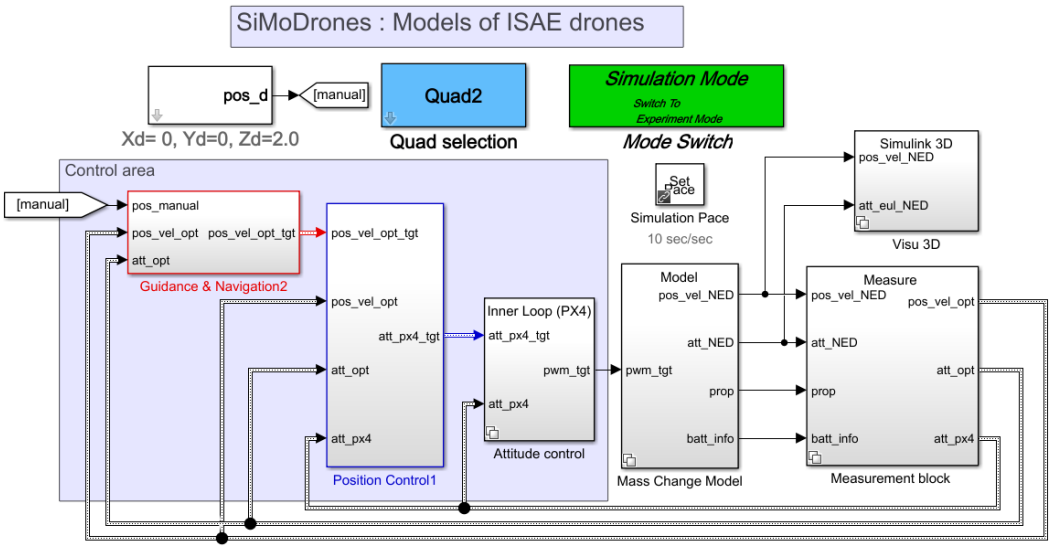


Figure 4: Full Control and Guidance drone model (MRAC)

5 Flight tests results with MRAC and Gain Scheduling

The results are exposed respectively in Figures 5 and 6 for both MRAC and Gain Scheduling controllers. Apart from a good tracking in the trajectory and each individual axis, it is extremely desirable to have a damped throttle behaviour, especially to save battery life. Notice that mass drop takes place after about 60s. Also included are the evolution of the adaptation gains and the mass estimation during the trajectory.

- **Loaded performance:** It is observed that the MRAC suffers from a slower take-off. This is due to the fact that around 5s are needed for the parameters to converge, because the initial gains K_P, K_I, and K_{Pv} are clearly badly tuned. Once in the air, the performance is very satisfactory, with a maximum 6 cm overshoot during the step sequence, and almost negligible static error during the circle/ellipse trajectory.

  The GS controller however, offers a better performance in terms of reactivity (faster take-off), as it is not subjected to the convergence of the MRAC parameters. Nevertheless, this greater reactivity comes at the price of a 30 cm oscillation in XY, corrected only 5s after take-off.

- **Unloaded performance:** As expected, once the parameters have converged again after the mass drop, the MRAC performance is almost identical to the loaded state, and follows that of the reference system.

  On the other hand, the GS controller offers a less impressive performance, characterized by 6 cm overshoots. This is due to the high integral gain present in the unloaded state. Unfortunately, this high gain is essential to ensure a fast convergence after the mass drop. An optimum had to be found between a fast convergence after the drop and a proper unloaded performance, resulting in the displayed trajectory.

- **Drop performance:** Both controllers prevent the drone from gaining more than 30 cm of altitude after the drop. However, while the GS takes **9s** to bring the drone back to its original altitude, the MRAC only takes **6s**. This 3s advantage is a great feature of the MRAC.

- **MRAC parameters convergence:** As a rule of thumb, it takes **10s** for the parameters to converge, either at the beginning of the mission and after the drop. The use of more noisy trajectories or higher adaptation gain would decrease the convergence time, but it would also result in a greater stress on the engines.

- **Mass estimation:** This is the main drawback of the GS. The ”naive” estimation proved to be extremely volatile, usually inducing false mass changes during the step trajectories. On the other hand, the RLS estimation, whilst better, resulted too slow.

As of today, the GS controller requires a precise knowledge of the drone mass and payload, only to deliver a performance that is only superior to that of the MRAC during the loaded segment of the mission. On the other hand, the MRAC offers a good performance throughout the entire mission, including better convergence after the drop, without any initial information about the drone mass or payload. Moreover, tests at very

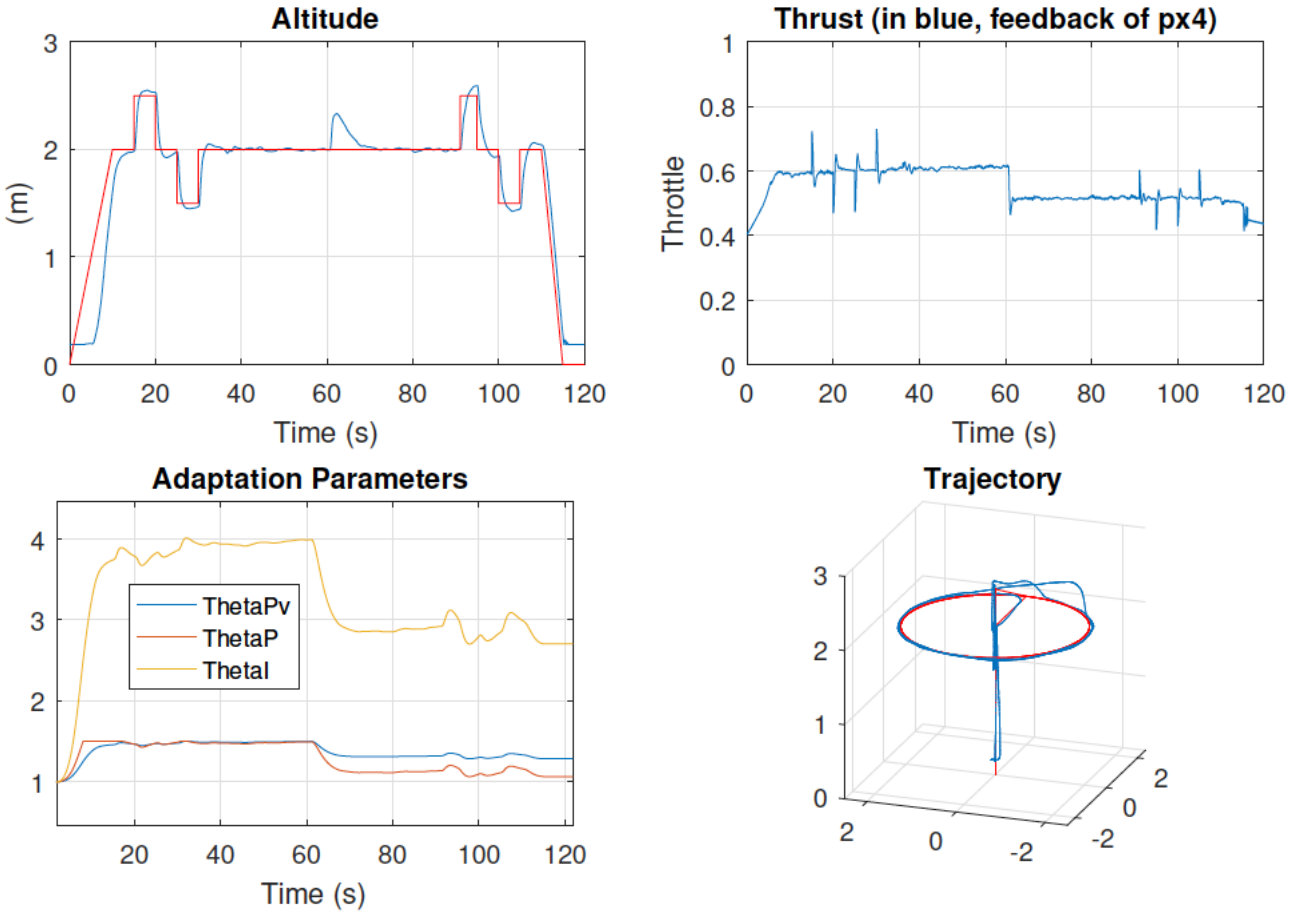


Figure 5: Circle mission results with MRAC (400g payload)

low altitude (subjecting the drone to ground effect) were also achieved with success, highlighting the flexibility and adaptability of the MRAC.

6 Analysis of the adaptation process

In order to highlight the potentialities of adapting the controller parameters, two experiments have been conducted. During the first one the adaptation has been frozen after some time to evaluate the expected loss of performances. For the second one, the initial UAV has been replaced by a completely different one but keeping the same controller.

6.1 Freezing the adaptation

The performance with full adaptation has already been presented in Figure 5. In the new test recorded in Figure 7, the convergence of the adaptation parameters is voluntarily **frozen** after 40s. Thus, convergence to the unloaded state will never be achieved.

By comparing Figures 5 and 7:

1. **Mass drop performance:** As expected, with the frozen adaptation the recovery is much slower, taking **18s** in comparison with the **6s** achieved with the full adaptation. Nevertheless, in both cases the drone only climbs **33 cm** before starting the recovery.

2. **Unloaded performance:** Due to the slow convergence from the mass drop, the drone does not have enough time to perform the full step sequence (since the mission is fixed at 120 s for battery reasons). However, on the final steps it can be seen that the behaviour of the drone is slightly better than the one recorded in the fully adapted case (Figure 5).

It can be concluded that the dynamic adaptation allows the drone to better **deal with the extreme disturbance** caused by the mass drop. The initial adaptation (after the take-off) provides however a **better tracking**.

6.2 Replacing the UAV

The formerly UAV (Mikrokopter Mk.6) has been replaced by a Parrot AR drone. The previous structure of the

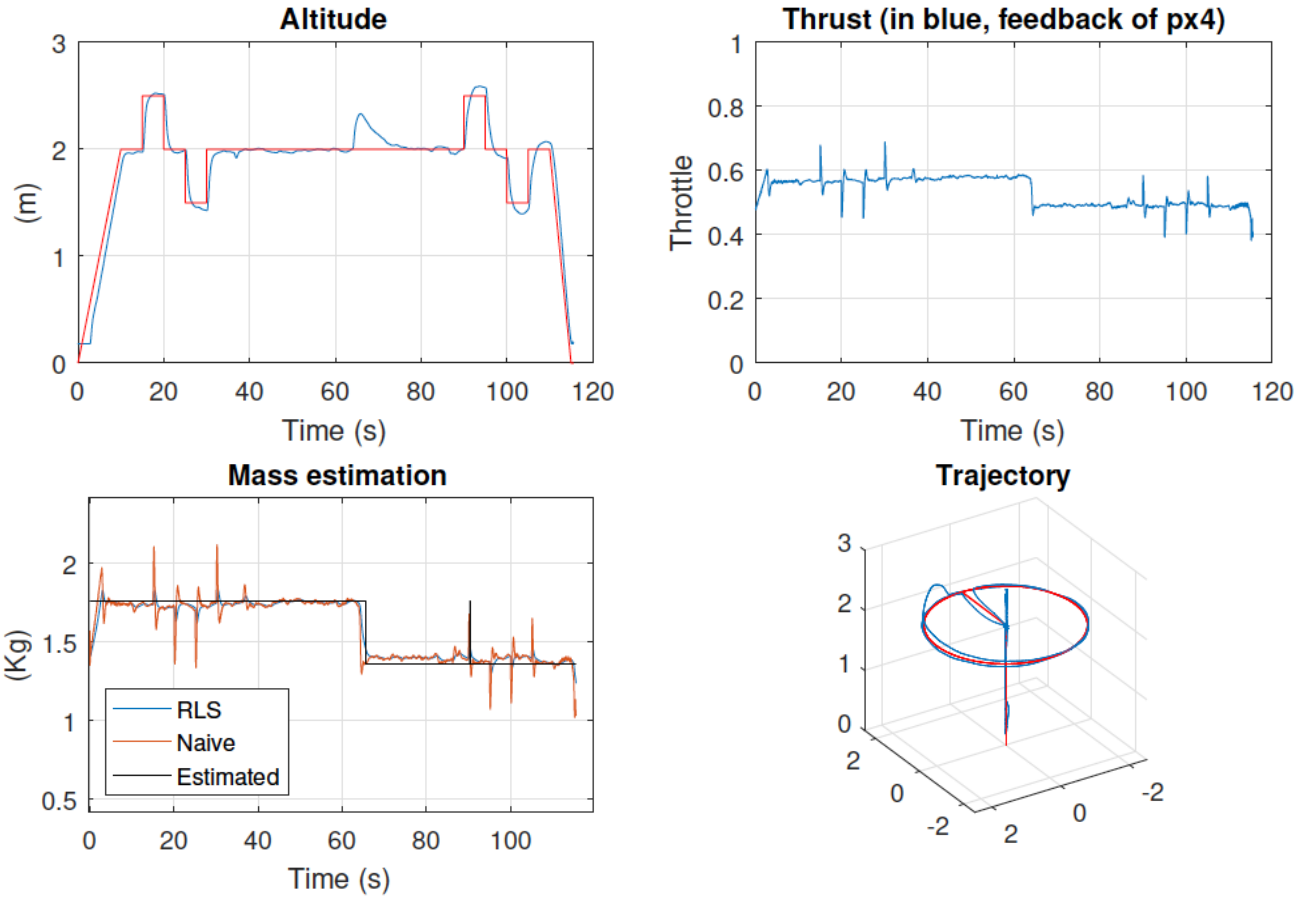


Figure 6: Circle mission results with GS (400g payload)

Specifications	Parrot AR	Mikrokopter Mk.6
Weight (kg)	0.713	1.360
Arm length (cm)	18	22.5
Propeller ⌀(inch)	8	10

Table 1: Parrot AR vs Mikrokopter Mk.6

controller is not modified, the initial tuning of the controller gains either. As seen in Figure 8, even when using a completely different drone, the MRAC ensures convergence of the controller gains, and a safe flight is possible.

A summary of the differences between the Parrot AR used in this flight and the Mikrokopter Mk.6 used elsewhere is given in Table 1 to illustrate the huge differences between the 2 drones; highlighting the MRAC convergence.

7 Conclusion

The objective of this work was to demonstrate that an adaptive controller represents a real alternative for the control of UAVs subject to severe changes of their dynamic behaviour. Model Reference Adaptive Control has proven to be a convincing and reliable solution. Indeed it allows to enrich a baseline controller with two important properties:

- autotuning of the controller gains from poor initial conditions.
- adaptation of the controller gains when needed.

Besides, the MRAC also proved itself very flexible, and may be used to control different drones which are forced into missions characterized by extreme perturbations. Future work may include more intensive testing of the behaviour of the current MRAC Controller for coping with specific conditions related to UAVs flight, like ground effect disturbances or engine failure.
Another interesting axis concerns the adaptation gains: Even if they represent an extra degree of freedom to improve the overall closed loop behaviour, their thin tuning may result time consuming. This is why we are actually working on the design of L1 type adaptive controllers, which belong to the

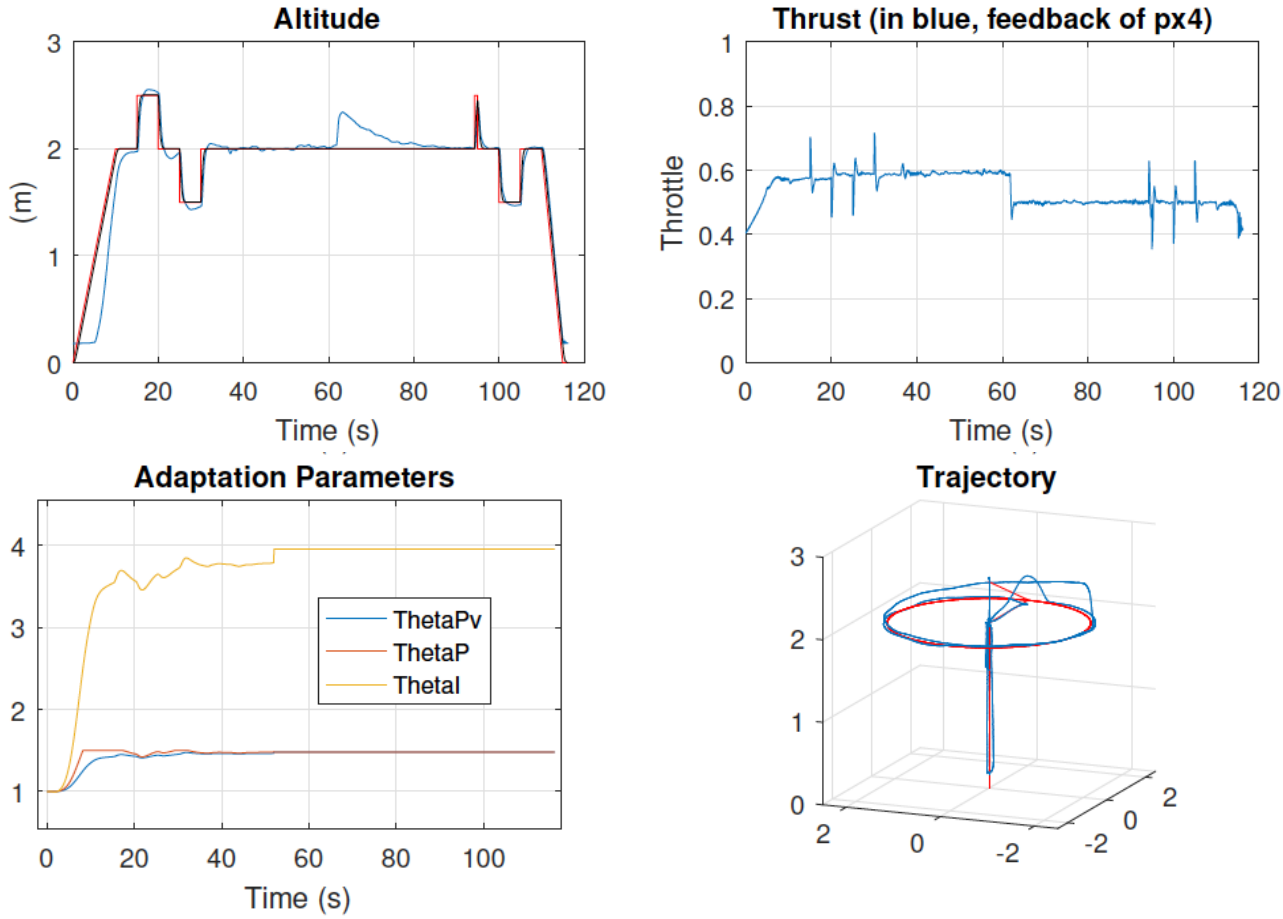


Figure 7: Fixed Adaptation Parameters with MRAC

same family while exhibiting interesting convergence properties when very high tuning gains are applied.

Finally, the Gain Scheduling controller also offers new windows of research. Since its performance is very sensible to the mass estimation, more elaborate methods of systems identification, like Kalman filtering, would be a great addition to the existing controller.

References

[1] J. Roberts, D. Frousheger, B. Williams, D. Campbell, and R. Walker. How the outback challenge was won: The motivation for the uav challenge outback rescue, the competition mission, and a summary of the six events. *IEEE Robotics Automation Magazine*, 23(4):54–62, Dec 2016.

[2] Saleen Bhattarai, Kushal Poudel, Nishant Bhatta, Sanjay Mahat, Sudip Bhattarai, and Kaman S Thapa Magar. Modeling and development of baseline guidance navigation and control system for medical delivery uav. In *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, page 0508. 2018.

[3] Daniel Mellinger, Michael Shomin, Nathan Michael, and Vijay Kumar. *Cooperative Grasping and Transport Using Multiple Quadrotors*, pages 545–558. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[4] Kimon P. Valavanis and George J. Vachtsevanos. *Future of Unmanned Aviation*, pages 2993–3009. Springer Netherlands, Dordrecht, 2015.

[5] B. Min, J. Hong, and E. T. Matson. Adaptive robust control (arc) for an altitude control of a quadrotor type uav carrying an unknown payloads. pages 1147–1151, Oct 2011.

[6] Maxime Lecointe, Caroline Ponzoni Carvalho Chanel, and Franois Defay. Backstepping control law application to path tracking with an indoor quadrotor. In *Proceedings of European Aerospace Guidance Navigation and Control Conference (EuroGNC)*, pages pp.1–19, Toulouse, FR, April 2015.

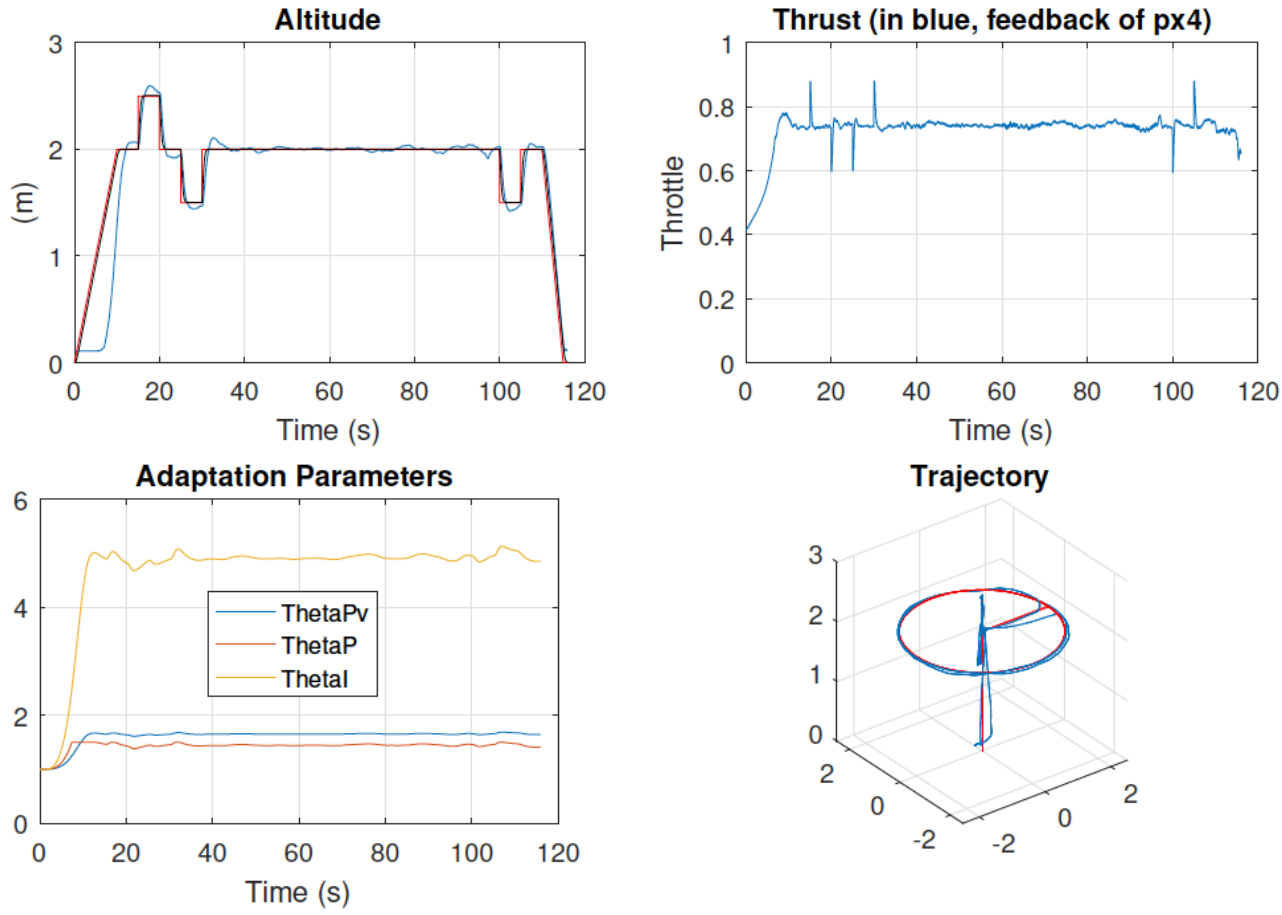


Figure 8: MRAC test with Parrot AR

[7] J. A. Maces Hernandez. *Landing an UAV in a moving platform, Master Research Project S2*. ISAE-SUPAERO, Toulouse, 2016.

[8] Yann Ameho, Fabien Niel, François Defaÿ, Jean-Marc Biannic, and Caroline Bérard. Adaptive control for quadrotors. In *ICRA*, pages 5396–5401, 2013.

[9] Teppo Luukkonen. Modelling and control of quadcopter. *Aalto University, Independent research project in applied mathematics*, pages 1400–1406, 2011.

[10] E Lavretsky and Kevin A. Wise. *Robust and Adaptive Control*. Springer London, London, 2013.

[11] Karl J. Astrm and Bjrn Wittenmark. *Adaptive Control*. Addison-Wesley, Reading, Massachusetts, 1995.

[12] M. A. Gmez Tierno, M. Prez Corts, and C. Puentes Mrquez. *Mecanica del Vuelo [Spanish]*. Garceta, Madrid, 2012.

Stability and Altitude Control of a Quadrotor Using Fuzzy Logic

R. L. S. M. Vilela [1,2] *, E. C. Silva [1,2] †
[1] Department of Electrical Engineering, Pontifical Catholic University of Rio de Janeiro, PUC-Rio, Rio de Janeiro, Brazil
[2] AeroRio, Pontifical Catholic University of Rio de Janeiro, PUC-Rio, Rio de Janeiro, Brazil

Abstract

The use of Unmanned Aerial Vehicles (UAVs) is becoming more common in many fields. The application spectrum varies from civil to military field, comprising environmental monitoring, border patrol, search and rescue operations, disaster relief, among others. In the next years, UAVs market is expected to provide an incoming of billions of dollars since it is rapidly growing in a lot of civilian and commercial industries such as agriculture, energy, utilities, mining, construction, real estate, news media and film production. Many of these applications require small and agile UAVs, capable to fly at low altitudes with a certain degree of maneuverability, controllability and stability, which requires well-tuned controllers. The most widely used controller for these applications is the PID (Proportional, Integral Derivative) controller. However, the tuning of this kind of controller can be very challenging. The objective of this paper is to present the development of a controller based on fuzzy logic to control the attitude angles and the altitude of a quadrotor UAV and compare its performance with a traditional PID control. The achieved results are shown and carefully discussed throughout the paper.

1 Introduction

Unmanned Aerial Vehicles (UAVs) have become an option to perform activities that would require a lot more effort when performed in traditional ways. The use of rotating wing UAVs have provided several benefits since they do not require a take-off and landing runway, with a VTOL (Vertical Take-off and Landing) vehicle system, and provide hovering capabilities. Also, the ease of control represents an advantage when compared to fixed-wing aircrafts. Furthermore, when it comes to financial value, the use of drones usually provide a viable alternative when compared to the traditional methods that are employed in the market[1].

Fixed-wings UAVs come in different geometries and sizes, that can be selected according to application and payload requirements. This paper discusses the modelling and control techniques for a quadrotor UAV in X configuration.

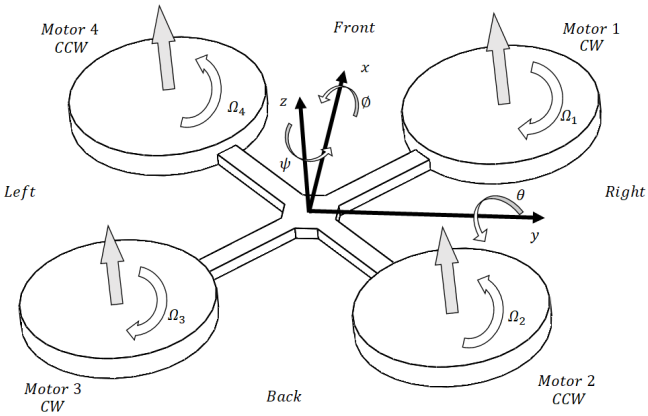


Figure 1: Quadcopter Representation

The principle of operation of multicopter vehicles consists in having their propellers generating enough lift to keep the vehicle in the air. To control the vehicle altitude, the amount of lift is increased or decreased, according to the desired movement. The position control in the xy plane is coupled with the roll and pitch angles, which means that a change in position is performed by changing the vehicles attitude. Finally, it is possible to control the heading of the vehicle with its yaw angle, that makes the drone rotate around its z-axis. Therefore, the vehicle dynamics can be described by changes in four main movements: altitude and roll, pitch and yaw angles.

The basis of the quadrotor dynamics consists in keeping the propellers spinning, aiming to generate enough lift to keep it in the air. Figure 1 shows a representation of a quadrotor configuration and will be used for a detailed explanation, throughout the paper. The pair of motors 1 and 3 spin on clockwise direction while motors 2 and 4 spin on counter-clockwise direction. All four propellers generate thrust in the same direction, which requires inverted pitch in the blades attached to motors 1 and 3 in relation to the pitch of the blades connected to motors 2 and 4. The difference in the spinning direction is to counter balance the torque generated by the rotating propellers[2].

To increase/decrease altitude, all four propellers in-

*Email address: renandelima95@gmail.com
†Email address: edusilva@ele.puc-rio.br

crease/decrease their rotation speed to increase/reduce the generated thrust, resulting a change in altitude. To change the roll angle, the pairs of motors 1 and 2 increase/decrease the rotation in comparison to the pair 3 and 4, generating a disbalanced thrust, and therefore changing the roll angle. To change the pitch angle, the process is analogous to the roll angle, however the pair of motors that changes the rotation speed are 1 and 4 or 2 and 3. On the other hand, to change the yaw angle, the pair of motors 1 and 3 changes its rotation in comparison to motors 2 and 4, increasing or decreasing the resultant torque applied to the vehicle.

2 QUADROTOR MODELLING

The quadrotor dynamic movements described in Section 3 are modelled based on Newton-Euler equations for 3D motion of rigid bodies and described in [3] and shown in Equation 1, where m is the drone's mass, I is a diagonal matrix with the drones inertia parameters around the x, y and z axis (I_{xx}, I_{yy}, I_{zz}), $\mathbf{v_B}$ is the vector with the drones velocity in the body-fixed reference system, $\omega_\mathbf{B}$ is the vector with the drones angular velocity in the body-fixed reference system and $\mathbf{f_B}$ and $\mathbf{m_B}$ are the vectors with the external forces and moments, respectively, applied to the drone.

$$\begin{cases} m(\dot{\mathbf{v_B}} + \omega_\mathbf{B} \wedge \mathbf{v_B}) = \mathbf{f_B} \\ I\dot{\omega_\mathbf{B}} + \omega_\mathbf{B} \wedge (I\omega_\mathbf{B}) = \mathbf{m_B} \end{cases} \tag{1}$$

The forces and moments generated by the rotation of the motors are described as in Equation 2 where l is the distance between the propeller and the vehicle CoG (center of gravity), b is the thrust coefficient of the motor-propeller setup, d is the aerodynamic drag coefficient, the Ω_n represent the rotation speed of the n-th motor, f_{Bx}, f_{By} and f_{Bz} make up the $\mathbf{f_B}$ vector and m_{Bx}, m_{By} and m_{Bz} make up the $\mathbf{m_B}$ vector, both from Equation 1.

$$\begin{cases} f_{Bx} = -mg\sin\theta \\ f_{By} = mg\cos\theta\sin\phi \\ f_{Bz} = mg\cos\theta\cos\phi + b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ m_{Bx} = \frac{\sqrt{2}}{2}lb(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \\ m_{By} = \frac{\sqrt{2}}{2}lb(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ m_{Bz} = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{cases} \tag{2}$$

The system kinematics is modelled using a ZYX Euler Rotation from the body fixed reference system to the inertial reference system.

Given both the dynamics and kinematics, the system equations can be arranged to form the state-space vector $x = [\dot{x} \quad \dot{y} \quad \dot{z} \quad \dot{\phi} \quad \dot{\theta} \quad \dot{\psi} \quad]^T \in \mathbb{R}^6$. Taking into consideration the simplification for small angles of movement, $[\dot{\phi} \quad \dot{\theta} \quad \dot{\psi}]^T = [p \quad q \quad r]^T$ [4], where p, q and r make up the vector $\omega_\mathbf{B}$, the state-space equations are described in Equation 3.

$$\begin{cases} \ddot{x} = -(\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)\frac{f_{Bz}}{m} \\ \ddot{y} = -(-\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi)\frac{f_{Bz}}{m} \\ \ddot{z} = (g - \cos\theta\cos\phi)\frac{f_{Bz}}{m} \\ \ddot{\phi} = \frac{I_{yy} - I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} + \frac{m_{Bx}}{I_{xx}} \\ \ddot{\theta} = \frac{I_{zz} - I_{xx}}{I_{yy}}\dot{\phi}\dot{\psi} + \frac{m_{By}}{I_{yy}} \\ \ddot{\psi} = \frac{I_{xx} - I_{yy}}{I_{zz}}\dot{\phi}\dot{\theta} + \frac{m_{Bz}}{I_{zz}} \end{cases} \tag{3}$$

3 TYPICAL CONTROL STATEGY

The quadrotors stability is achieved by using a closed loop control system, based on feedback from inertial measurements. Typically, UAVs have an embedded 9DOF IMU (9 Degrees of Freedom Inertial Measurement System), with three-axis accelerometers, three-axis gyroscopes and three-axis magnetometers that provide the attitude angles used to control its stability.

Since attitude estimation requires numerical integration based on measured angular velocities, the systems are subjected to cumulative errors due to noisy measurements. Therefore, control systems include fusion filters that merges data from multiple sensors, such as linear complimentary filters and Extended Kalman Filters [2].

Nested PID controllers are the most widely used strategy for stability control of rotating wing UAVs. They actuate on the desired moments of the vehicle, based on feedback from the sensors and attitude commands. The attitude commands can be given directly by the pilot or be provided by an autonomous trajectory control algorithm, that relies on other ways to measure the vehicle position (i.e GPS) and provide the specific attitude commands for the desired trajectory.

Since this paper is not focused on trajectory control, but on stability and altitude control, it will be based on the case of a manual flight mode,as highlighted in red in Figure 2.

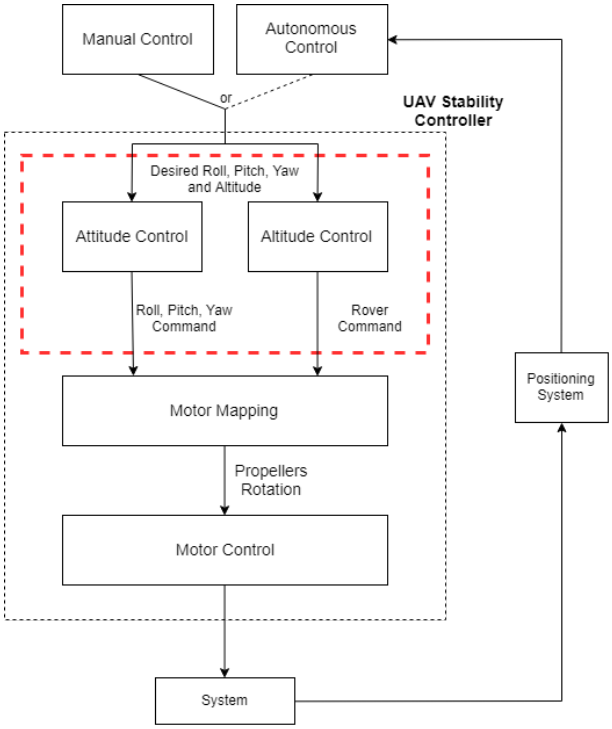


Figure 2: Overall Control Block Diagram

The altitude control, also known as altitude hold, is based on a PID controller actuating in the altitude error, that provides the base command for the motors in order to keep the quadrotor in hovering mode. Figure 3 shows the block diagram for altitude control.

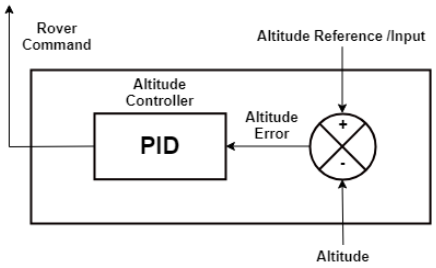


Figure 3: PID for Altitude Control

The stability control is based on a nested PID controller that actuates on the error of the attitude angles to provide the reference angular velocities and then actuates on the error of the angular velocities to provide a normalized actuation command for the motors. Figure 4 shows the block diagram for yaw angle control. Similar block diagrams are used for roll and pitch angles, varying only the inputs and controller gains.

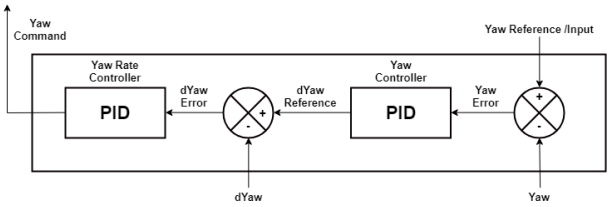


Figure 4: Nested PID for Yaw Control

4 Fuzzy Control

A fuzzy control system consists of 4 basic elements, shown in Figure 5. First, the fuzzification module, that is responsible for converting specific input values to fuzzy sets. The knowledge base is composed by the rules that define the control strategy for the system, which are usually extracted from a specialist. The inference system process the fuzzified inputs according to the rules from the knowledge base to infer the actions of the fuzzy controller. Finally, the defuzzification module converts the fuzzy sets, generated by the inference system, back to exact values that are used in the control process [5].

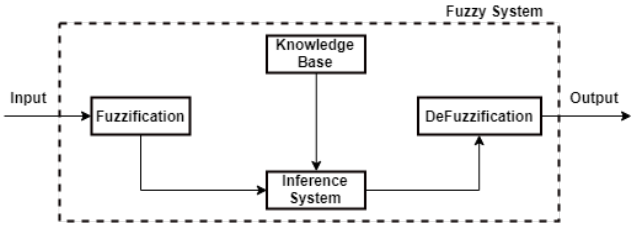


Figure 5: Fuzzy System Diagram

The fuzzy controller for the altitude control was developed based on the altitude error and the derivative of the altitude error. Figure 6 shows the block diagram of the fuzzy altitude controller.

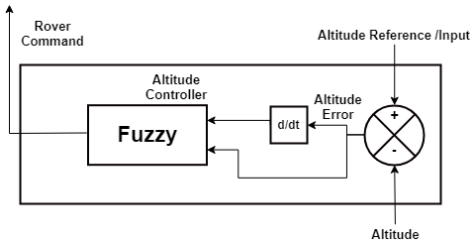


Figure 6: Fuzzy Control for Altitude Control

For the attitude angles, the same method of calculating the derivative of the error was used and only the first PID in the nested approach (Figure 4) was substituted for a fuzzy controller. Figure 7 shows the block diagram for the yaw angle fuzzy-PID controller. The same approach is used for roll and pitch angles as well, changing only the fuzzy controller.

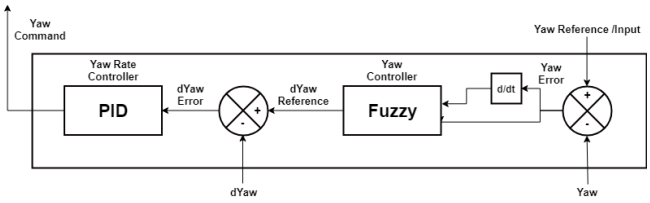


Figure 7: Nested Fuzzy-PID for Yaw Control

5 Implementation

The model was implemented in Simulink/Matlab, using the Simscape toolbox. The parameters used for the drone were extracted from the Crazyflie 2.0 Nano drone, a small quadrotor with open source software that allows the user to personalize its firmware and implement different control approaches. Therefore, developing the fuzzy control for this drone is a first step to implement it in a real setup.

The physical drone parameters were extracted from [6] and the PID gains, used during the initial simulations, were extracted from Crazyflie firmware. The Simscape toolbox creates a 3D simulation environment that allows proper visualization of the vehicles movements.

Fuzzy membership functions and rules were implemented by using the Fuzzy Logic Design toolbox, that provides visual interfaces for easy tuning of the parameters.

Sections 5.1, 5.2 and 5.3 shows the tuned membership functions and rules for the controllers developed. For all controllers, the centroid was used as defuzzification method while minimum was used as implication method.

5.1 *Altitude Hold*

The error in altitude input was divided in 5 membership functions: Negative Big (NB), Negative (N), Zero (Z), Positive (P) and Positive Big (PB). Its derivative was divided in 3 membership functions: Negative (N), Zero (Z) and Positive(P)

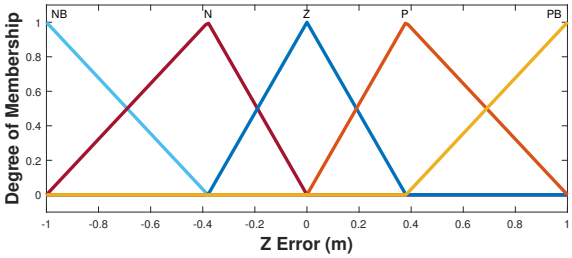


Figure 8: Input Membership Function - Z Error

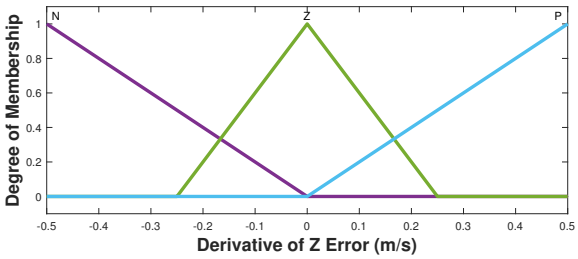


Figure 9: Input Membership Function - Derivative of Z Error

The output for the altitude controller was divided in 5 membership functions: Down Big (DB), Down (D), Maintain (M), Up (U) and Up Big (UB).

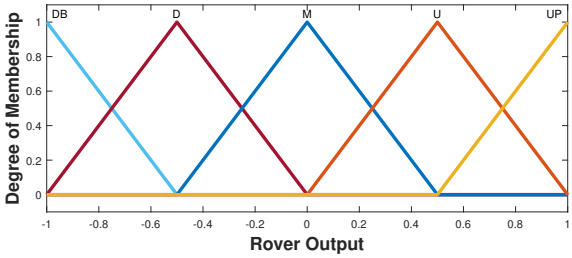


Figure 10: Output Membership Function - Rover

Table 1 show the set of rules used in the knowledge base of the fuzzy inference system for the altitude controller.

Derivative \ Error	NB	N	Z	P	PB
N	DB	D	D	M	U
Z	DB	D	M	U	UB
P	D	M	U	UB	UB

Table 1: Fuzzy Rules for Altitude Control

5.2 *Roll/Pitch Angle*

Since the model being used is symmetrically in the x and y direction, the roll and pitch movements presents the same dynamic. Therefore, the same controller was capable of controlling both angles.

The error in the roll and pitch angle input was divided in 5 membership functions: Negative Big (NB), Negative (N), Zero (Z), Positive (P) and Positive Big (PB). Its derivative was divided in 3 membership functions: Negative (N), Zero (Z) and Positive(P)

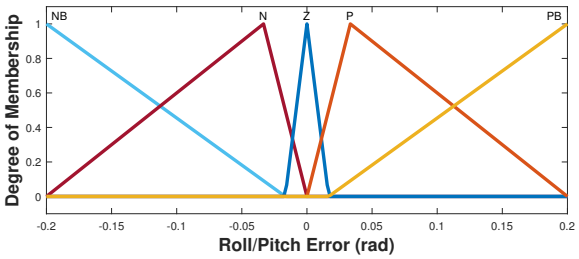


Figure 11: Input Membership Function - Roll/Pitch Error

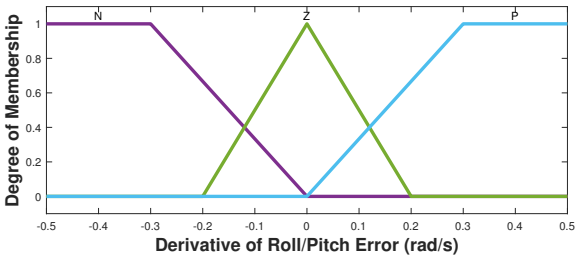


Figure 12: Input Membership Function - Derivative of Roll/Pitch Error

The output for the roll/pitch derivative reference was divided in 5 membership functions: Negative Big (NB), Negative (N), Zero (Z), Positive (P) and Positive Big (PB).

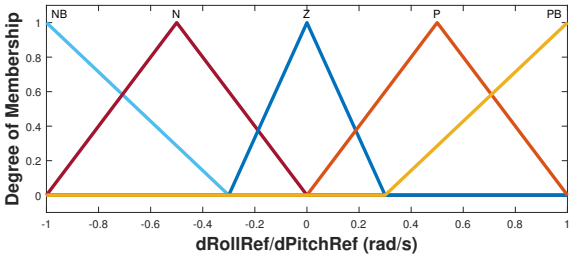


Figure 13: Output Membership Function - Reference for Derivative of Roll/Pitch

Table 2 shows the set of rules used in the knowledge base of the fuzzy inference system for the roll and pitch controller.

Derivative \ Error	NB	N	Z	P	PB
N	NB	NB	N	Z	P
Z	NB	N	Z	P	PB
P	N	Z	P	PB	PB

Table 2: Fuzzy Rules for Roll and Pitch Control

5.3 *Yaw Angle*

The error in the yaw angle input was divided in 5 membership functions: Negative Big (NB), Negative (N), Zero (Z), Positive (P) and Positive Big (PB). Its derivative was divided in 3 membership functions: Negative (N), Zero (Z) and Positive(P)

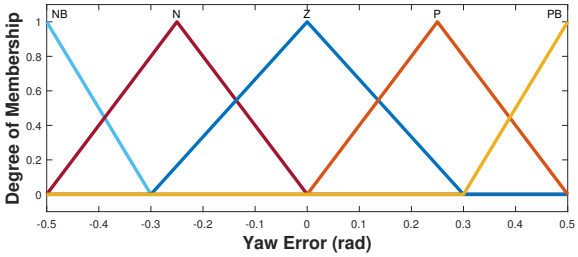


Figure 14: Input Membership Function - Yaw Error

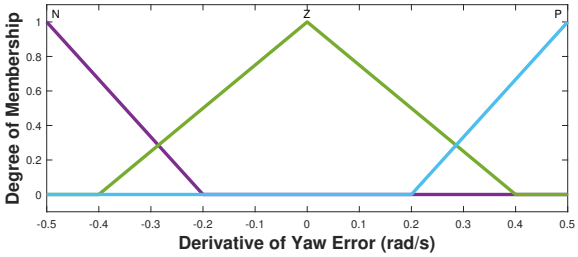


Figure 15: Input Membership Function - Derivative of Yaw Error

The output for the yaw derivative reference was divided in 5 membership functions: Negative Big (NB), Negative (N), Zero (Z), Positive (P) and Positive Big (PB).

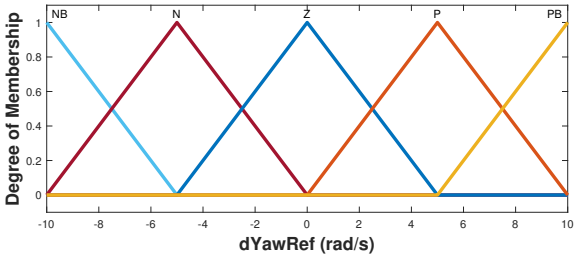


Figure 16: Output Membership Function - Reference for Derivative of Yaw

Table 3 shows the set of rules used in the knowledge base of the fuzzy inference system for the yaw controller.

Derivative \ Error	NB	N	Z	P	PB
N	NB	NB	N	Z	P
Z	NB	N	Z	P	PB
P	N	Z	P	PB	PB

Table 3: Fuzzy Rules for Yaw Control

6 Results

After completing the tuning for the Fuzzy membership functions and rules, simulations were performed by giving reference values for the 4 variables simultaneously. The values given for reference were selected considering reasonable maneuvers for a quadcopter. Figures 17, 18, 19 and 20 shows the results obtained.

To provide a benchmark to evaluate the performance of the designed fuzzy controller, simulations were performed using the traditional PID controller with gains set according to the standard values that come out-of-the-box with the Crazyflie 2.0 Nano.

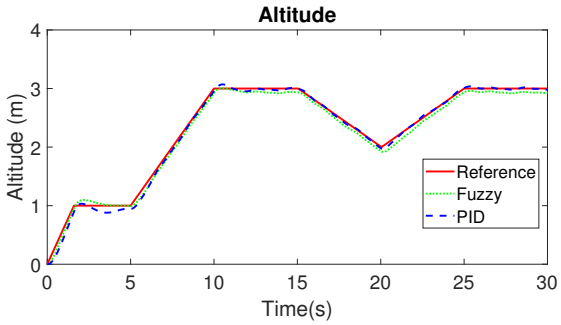


Figure 17: Altitude Control

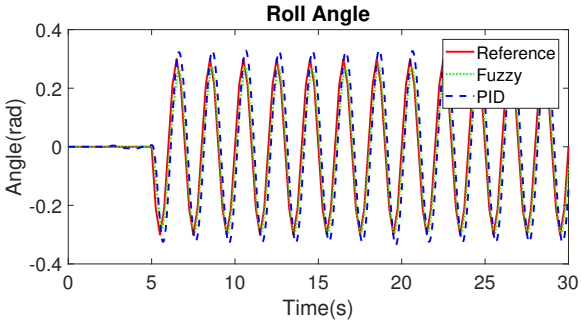


Figure 18: Roll Angle Control

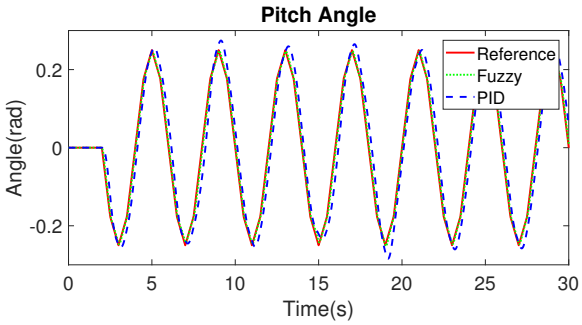


Figure 19: Pitch Angle Control

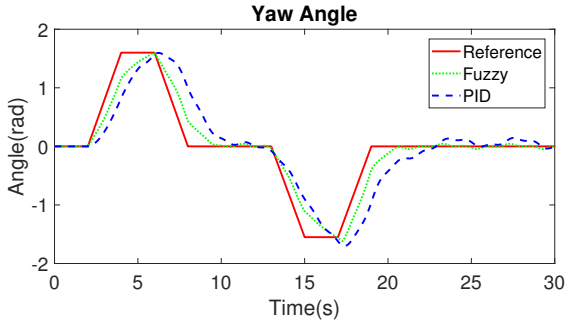


Figure 20: Yaw Angle Control

7 Discussion and Conclusion

The implementation of the fuzzy controller in the altitude and attitude control of a quadrotor UAV was successful, outperforming the standard PID controller, used as benchmark, in several cases.

For the altitude controller, there was a reduction in the over/undershoot performance compared to the PID controller. The roll and pitch controllers provided very good results, regardless of being based on PID or fuzzy control techniques, with different reference derivatives however. The yaw fuzzy controller obtained a significantly better performance compared to the PID performance, reducing the error during the whole control process.

As a future work, still in the stability control, it is possible to substitute the second PID controller for the derivative of the attitude angles for Fuzzy controllers. Also it is possible to expand this work to enable a full Fuzzy autonomous trajectory control. Besides, considering the good results predicted by the performed simulations, a next step would be to implement the fuzzy controller in the real Crazyflie 2.0 Nano quadcopter, aiming to evaluate its experimental behavior and also compare to the results obtained in the simulations.

Aiming to further enhance the simulation, a proper modelling of the sensors can be performed to analyze the influence of the sensor noise in the controllers performance. Also, one thing that needs to be taken into consideration when it comes to hardware performance is the implementation of the

fuzzy controller, since it normally requires more calculations than the PID controllers.

ACKNOWLEDGEMENTS

We would like to thank the Brazilian funding agencies CNPq, FAPERJ and CAPES for continued support and supplied resources.

REFERENCES

[1] S. Maharana. Commercial drones. *IJASCET*, 5:96 – 101, 2017.

[2] Q. Quan. *Introduction to multicopter design and control*. Springer, Singapore, 2017.

[3] F. Sabatino. *Quadrotor control: modeling, nonlinear control design, and simulation*. PhD thesis, 2015.

[4] A. Das, K. Subbarao, and F. Lewis. Dynamic inversion with zero-dynamics stabilisation for quadrotor control. *IET Control Theory Applications*, 3(3):303–314, March 2009.

[5] A. C. Gomide, F. Gudwin, and R. Tanscheit. Conceitos fundamentais da teoria de conjuntos fuzzy, logica fuzzy e aplicacoes. 12 2018.

[6] J. Forster. *System Identification of the Crazyflie 2.0 Nano Quadrocopter*. PhD thesis, 2015.

UAV control costs mirror bird behaviour when soaring close to buildings

A. Guerra-Langan*, S. Araujo-Estrada and S. Windsor
University of Bristol, Bristol, United Kingdom

ABSTRACT

Small unmanned aerial vehicles (SUAVs) are suitable for many low-altitude operations in urban environments due to their manoeuvrability; however, their flight performance is limited by their on-board energy storage and their ability to cope with high levels of turbulence. Birds exploit the atmospheric boundary layer in urban environments, reducing their energetic flight costs by using orographic lift generated by buildings. This behaviour could be mimicked by fixed-wing SUAVs to overcome their energy limitations if flight control can be maintained in the increased turbulence present in these conditions. Here the the control effort required, and energetic benefits, for a SUAV flying parallel to buildings whilst using orographic lift was investigated. A flight dynamics and control model was developed for a powered SUAV and used to simulate flight control performance in different turbulent wind conditions. It was found that the control effort required decreased with increasing altitude and that the mean throttle required increased with greater radial distance to the buildings. However, the simulations showed that flying close to the buildings in strong wind speeds increased the risk of collision. Overall, the results suggested that a strategy of flying directly over the front corner of the buildings appears to minimise the control effort required for a given level of orographic lift, a strategy that mirrors the behaviour of gulls in high wind speeds.

1 INTRODUCTION

The high manoeuvrability of small unmanned aerial vehicles (SUAVs) makes them particularly suitable for many low-altitude operations in urban environments. They have application in many different fields [1], such as border control, search and rescue, surveillance [2, 3, 4, 5], medical supply or parcel delivery [6, 7], and natural disaster response [8, 9]. Limited on-board energy [10, 11] and the effect of the atmospheric turbulence at low altitudes [12, 13, 14, 15] are two challenges that have a major effect on flight performance of SUAVs.

On-board energy storage is limited due to the size and weight constraints of SUAVs. Batteries can constitute up to 40% of the vehicle's mass, giving a nominal flight time of approximately 60 minutes for small fixed-wing vehicles [10, 16, 11]. This restricts the endurance and range of these aircraft, which may compromise their missions. Therefore, energy management constitutes an important challenge in the design of these vehicles.

SUAVs typically operate at low altitude, in the atmospheric boundary layer (ABL). The challenge of this resides in the increase in turbulence intensity closer to the ground, where the flow is dominated by horizontal transport of atmospheric properties and wind speeds increase because of the pressure gradients caused by buildings and obstacles [13]. This can result in wind disturbances that are the same order of magnitude as the vehicle's flight speed. As the flight of these vehicles is characterised by low Reynolds number, low inertia, low flight speed and low stability [17], this creates a substantial challenge for flight control in some conditions.

One method for reducing the energetic cost of flight is to make use of environmental wind flows. When wind is deflected upwards by the presence of obstacles, such as hills or buildings, this creates opportunities for orographic soaring, where the sink rate of the aircraft is offset by the vertical motion of the air. Langelaan *et al.* [18] designed a path planner for UAVs which could make use of orographic soaring through optimising a particular cost function based on knowledge of the wind field. Simulations were conducted for two unmanned aircraft, showing improvements for both optimal minimum time and optimal maximum energy trajectories compared to a constant speed trajectories. White *et al.* [19] studied the feasibility of SUAV soaring in urban environments based on wind-tunnel experiments with scaled model buildings. The aim of this testing was to measure the relationship between the vertical component of the wind and the oncoming mean wind speeds, showing values between 15 and 50%. A further study, [20], measured the sink rate of a soaring MAV and concluded that depending on the wind strength and direction, it is feasible for a MAV to exploit orographic soaring next to buildings. A MAV platform and control system was later designed to try to mimic the kestrel's "wind-hovering" strategy, holding the MAV's longitudinal and lateral position [21]. Flight tests were conducted around two locations: a hill and a building. The former resulted in consis-

*Email address: ana.guerra-langan@bristol.ac.uk

tently successful soaring flights while tests around the latter could not be sustained for over 20 seconds, which was attributed to gustiness. A CFD model to simulate the turbulent wind flow conditions surrounding buildings was designed by Mohamed *et al.* in [22], with the aim of providing the potential energy available for harvesting. This model was used to locate suitable areas of lift, which were then tested in flight trials [23].

Birds fly in the same conditions as SUAVs and face similar challenges in energy management and flight control. Birds frequently reduce their energy expenditure by exploiting environmental wind fields such as tailwinds, wind gradients and updraughts. Of particular interest for SUAV flight control is how birds exploit orographic updraughts generated by man-made structures, with birds being observed soaring on the upwind side of ferries [24] and buildings [25]. Shepard *et al.* [25] studied gulls exploiting orographic updraughts by soaring parallel to the face of buildings. The two main findings of this work were that birds used the updraughts to maintain height rather than to gain it and that they positioned themselves in specific regions of the wind field depending on the strength of the wind. It was hypothesised that the gulls flight control requirements in gusty conditions were reduced in these specific regions of the wind field.

Following the work from Shepard *et al.* [25], the aim of this study is to investigate the control requirements for a powered SUAV to take advantage of orographic soaring when flying along a row of buildings. This is done by simulating a SUAV holding position relative to buildings whilst flying through different wind fields measured in [25]. Control effort is used to compare the control demand of different simulations and ultimately, of flights in different regions of the wind field. First, a flight guidance, navigation and control (GNC) framework is described in Section 2. The specific atmospheric conditions, a description of the SUAV and an overall view of the metrics and simulations used is then given in Section 3. Results are shown and discussed in Section 4; and conclusions drawn in Section 5.

2 FRAMEWORK

Simulations were carried out using a 6 DOF flight GNC framework implemented in Simulink (MathWorks, MA, USA). This model was used to simulate the behaviour of a powered SUAV flying in gusty, windy conditions.

2.1 Guidance

The flight guidance system calculated the changes in pitch and yaw angle required to follow a predefined trajectory based on the SUAVs inertial position.

- Altitude control - Pitch angle desired

  Altitude was controlled by means of a pitch controller adjusting the elevator. The desired pitch angle was calculated following the block diagram in Figure 1.

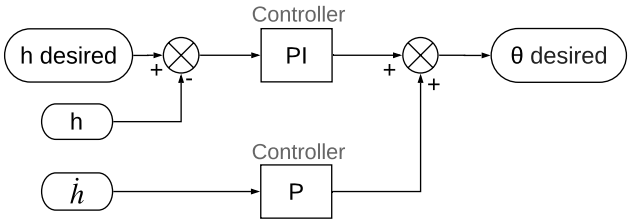


Figure 1: Altitude controller block diagram. h is the altitude, $\dot{h}$ is the vertical speed and θ is the pitch angle of the aircraft

- Lateral control - Yaw angle desired

  The lateral position was controlled by means of the ailerons and rudder. The desired yaw angle required to follow the path was calculated following the block diagram in Figure 2.

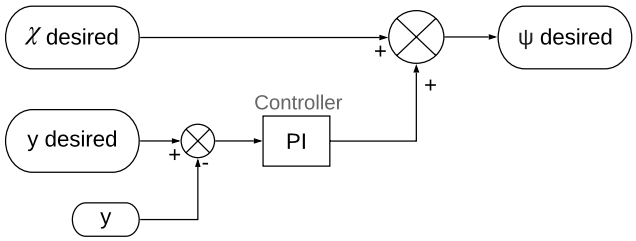


Figure 2: Lateral position controller block diagram. χ is the course angle, y is the lateral position of the vehicle and ψ is the yaw angle of the aircraft

  The course angle was defined as the angle between North and the direction of movement of the vehicle. The desired course angle was obtained by looking at a point in the trajectory which was at a distance $L1$ from the vehicle. $L1$ was set as $25\,\mathrm{m}$ in this study. Under no wind perturbations, the course angle is equal to the heading angle (yaw angle). However, to counteract the wind effects, a PI controller was used with the lateral position error to correct the drift caused by the wind.

2.2 Navigation

Following the work of Langelaan *et al.* [26] and Depenbusch [27], the 6DOF flight dynamic equations used in this work are presented in Appendix A. Figures 3 and 4 show the general structure used to model the flight dynamics.

x and **u** are the state and control vectors defined in Equations 1 and 2 below. V_a is the airspeed, α is the angle of attack, β is the sideslip angle, [p, q, r] are the roll, pitch and yaw Euler angle rates respectively and [ϕ, θ, ψ] are the corresponding Euler angles. δ_i for *i = ail, ele, rdd and thr* is the deflection angle of the aileron, elevator and rudder and the commanded value of throttle, respectively.

$$\mathbf{x} = [V_a, \alpha, \beta, p, q, r, \phi, \theta, \psi] \quad (1)$$

$$\mathbf{u} = [\delta_{ail}, \delta_{ele}, \delta_{rdd}, \delta_{thr}] \quad (2)$$

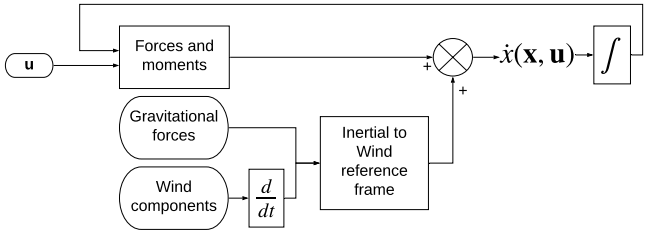


Figure 3: Block diagram of the flight dynamic equations

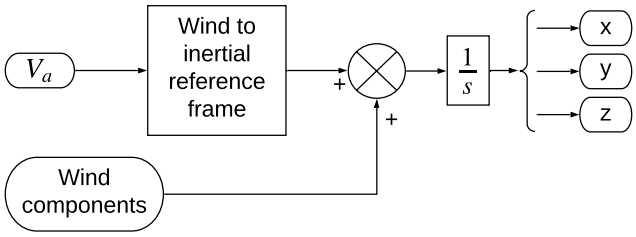


Figure 4: Block diagram of the inertial position of the aircraft

$[x, y, z]$ represented the position of the aircraft expressed in the inertial NED reference system fixed to the ground.

2.3 Control

The flight control framework was composed of a series of controllers which allowed the aircraft to keep steady-level-flight and hold its lateral position. Block diagrams of the controllers are presented in Figures 5-7. These controllers were designed following [28].

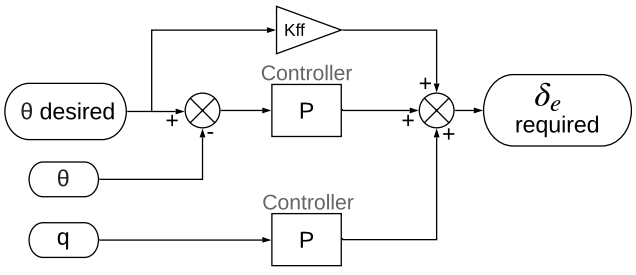


Figure 5: Pitch controller block diagram

Note that U_0 is the equilibrium airspeed in stability axes and τ_ψ and τ_r are time constants.

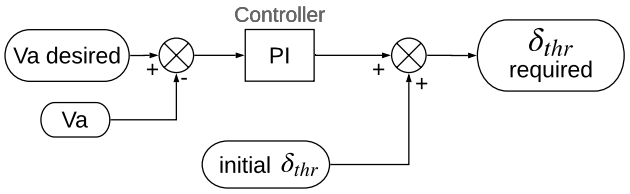


Figure 6: Airspeed controller block diagram

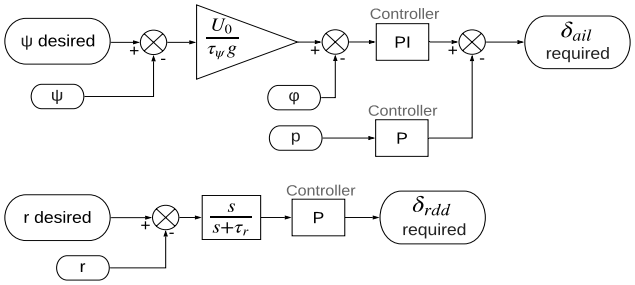


Figure 7: Course controller block diagram

2.4 Limitations

The model did not account for some of the physical limits of the aircraft or the environment. Phenomenon such as stall and ground effect were not taken into account. The aircraft was modelled as a point mass, with the distribution of the wind across the wing span not being considered in the flight dynamics of the aircraft.

3 SIMULATIONS

The ultimate goal of this work was to study the control costs of a SUAV when soaring close to buildings, determining if there was a benefit to flying in particular regions of the wind field. This was investigated by simulating a SUAV flying close to a simulated row of buildings in a range of different wind conditions. The aircraft's controllers were set to hold airspeed, height and lateral position. In particular, the desired airspeed was kept constant and equal to $12.7\,\mathrm{m/s}$ throughout the simulations and the 26 combinations of desired height and lateral positions studied are defined in Figure 8.

3.1 Wind field

The urban wind field used in this study was generated by Shepard *et al.* [25] for periods of onshore winds in Swansea Bay, UK. Here the wind came in over the open sea before meeting a row of four-storey buildings, which deflected the air upwards causing orographic lift. Shepard *et al.* found that the total number of gulls observed soaring by the sea-front in this area increased with wind strength and varied with wind direction. There was a peak in the total number of birds observed for winds from around 150° (SE), which coincided with the wind being perpendicular to the front face of the buildings.

The wind field data were simplified for the simulations by averaged along the direction of flight. This simplification allowed the simulation to run indefinitely, modelling the SUAV flying parallel to a long row of buildings. The two wind fields used in this study had considerably different nominal wind speeds at 20 feet ($W20$), $W20 = 2.26\,\mathrm{m/s}$ and $W20 = 9.34\,\mathrm{m/s}$, but similar wind directions, $140.8°$ and $137°$ respectively. Figures 8a and 8b show a cross section of the wind fields used along with the desired SUAV positions tested.

The Dryden model was used to add a continuous level of disturbance to the steady state wind field already described. This mathematical model representing the frequency spectrum of continuous gusts was integrated into the flight dynamic equations of motion as an atmospheric disturbance. In this work, the MIL-F-8785C [29] specification was applied through the "Dryden Wind Turbulence Model (Continuous)" block in Simulink for low-altitude applications. The input required in each simulation was the nominal wind speed at 20 feet ($W20$) and the wind angle with respect to North. The equations defining the model can be found in Appendix B. Three low-altitude disturbance levels are studied, defined as: $WFDi$ with $i = 75, 100$ and 125% of $W20 = 2.26\,\mathrm{m/s}$ and $W20 = 9.34\,\mathrm{m/s}$. Figures 8c and 8d show the variation of the wind field due to the Dryden disturbance model on its own. The Dryden model does not take into consideration natural or artificial obstacles in the environment. Because of that, the disturbance level and standard deviation are only affected by the vertical distance from the ground. The standard deviation of the disturbance added by the Dryden model reaches values of up to 20% of $W20$ for u and v, and up to 10% of $W20$ for w.

3.2 *Control effort*

In this work, control effort (CE) refers to the amount of control necessary to keep steady-level-flight whilst holding a specific lateral and vertical position parallel to the buildings. This term is used as a parameter to compare the control demand of different simulations and ultimately, of flights in different regions of the wind field.

Considering the rate of change in the control surface deflection and in the throttle demand as a measurement of how much these are being used to control the vehicle, Equations 3-5 define the control effort parameter used in this study.

The deflection angle for the three control surfaces and the demanded throttle value are normalised by their maximum achievable value. Deflection limits are gathered in Appendix C.

$$\delta_{i_{norm}} = \frac{|\delta_i|}{max\{\delta\}} \tag{3}$$

$$\delta_{i_{rate}} = \frac{d}{dt}\delta_{i_{norm}} \tag{4}$$

The deflection rate is defined as a timeseries and its root-mean-square (RMS) is reported here.

$$CE_i = RMS(\delta_{i_{rate}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}{\delta_{i_{rate}}}^2} \tag{5}$$

3.3 *SUAV platform*

A non-linear flight dynamic model of an instrumented WOT 4 Foam-E Mk2+ (Ripmax, Enfield, UK) (Figure 9) has been derived from outdoor flight tests using the output error method and has been integrated in the model. This vehicle is $1.345\,\mathrm{kg}$ and has a $1.205\,\mathrm{m}$ span with an aspect ratio of 4.85. It has three control surfaces: elevator, rudder and ailerons.

Figure 9: Ripmax WOT 4 Mk 2 UAV

The servo motors and the electric engine responses have been simplified and defined as a linear second-order transfer function:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{6}$$

The main physical and aerodynamic characteristics of this platform, the control PID gains for each one of the controllers and the natural frequency (ω_n) and damping ratio (ζ) of the transfer functions are given in Appendix C.

4 RESULTS AND DISCUSSION

In this section, results are presented for the different atmospheric effects described in Section 3. Flights under these conditions were simulated for a desired airspeed of $12.7\,\mathrm{m/s}$, and for 26 different paths.

The results showed that the CE required to maintain steady-level-flight and lateral position strongly depended on the height at which the SUAV was flying. Figure 10 shows the trend of this parameter for the three control surfaces and the commanded throttle. The CE with respect to the imposed height is presented for the six different atmospheric conditions studied. The control effort required decreased as the imposed height increased for all control surfaces and the

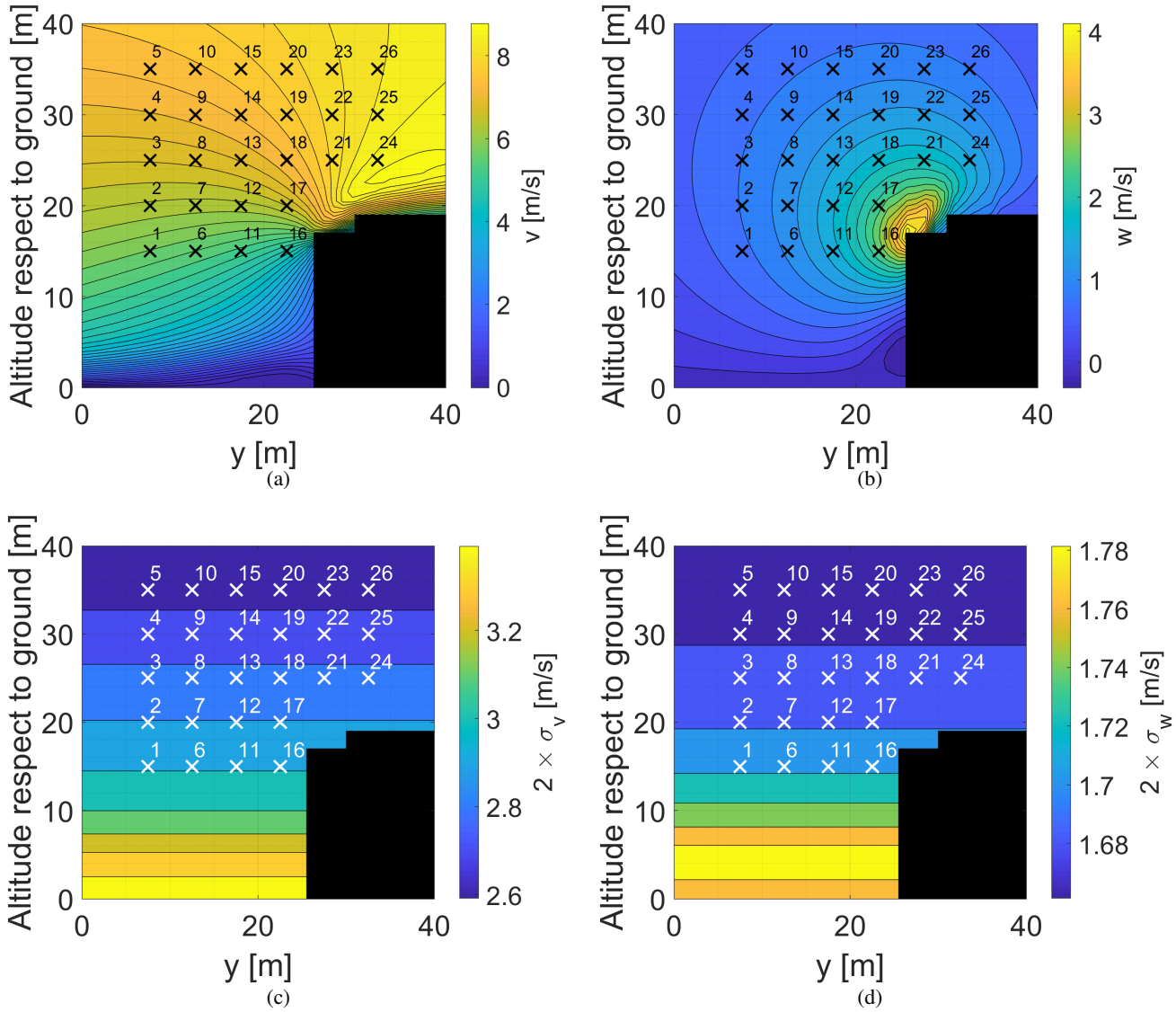


Figure 8: Contour plots of lateral (v) and vertical (w) components of the wind together with the imposed vertical and lateral positions studied ('x' symbols). 8a and 8b give the distribution of the v and w wind vector components in relation to the buildings for $W20 = 9.34\,\mathrm{m/s}$. The strength of the wind components is illustrated with the colour scale, with a resolution of $0.2\,\mathrm{m/s}$ between contour levels. 8c and 8d show the distribution of the v and w wind vector components of the Dryden disturbance model for $W20 = 9.34\,\mathrm{m/s}$. These are given as $2 \times \sigma_v$ and $2 \times \sigma_w$ respectively, which corresponds to the 95% confidence interval range for the change in wind speed. The strength of the Dryden noise wind components is illustrated with the colour scale, with a resolution of $0.1\,\mathrm{m/s}$ for v and $0.02\,\mathrm{m/s}$ for w.

throttle in all wind conditions. The slight difference in the pattern shown for the elevator for the most turbulent wind field (WFD125 9.34) was due to the aircraft crashing for some paths close to the buildings. There are two factors that could have an important role in the explanation of this trend. Firstly, the disturbance intensity was correlated with the inverse of the height (Figures 8c and 8d) and more control effort is required at higher disturbance intensities. Secondly, the mean wind fields close to the buildings had a greater horizontal variation at lower heights (Figures 8a and 8b), which would also have increased the wind gradient experienced by the UAV if it moved laterally. Figure 10 also shows that as the wind speed increased ($W20$), the control effort required also increased. This was as expected because the disturbance intensities of the Dryden model is proportional to the mean wind speed.

Comparing two points in the wind field at the same radial distance to the buildings, Shepard *et al.* suggested that the

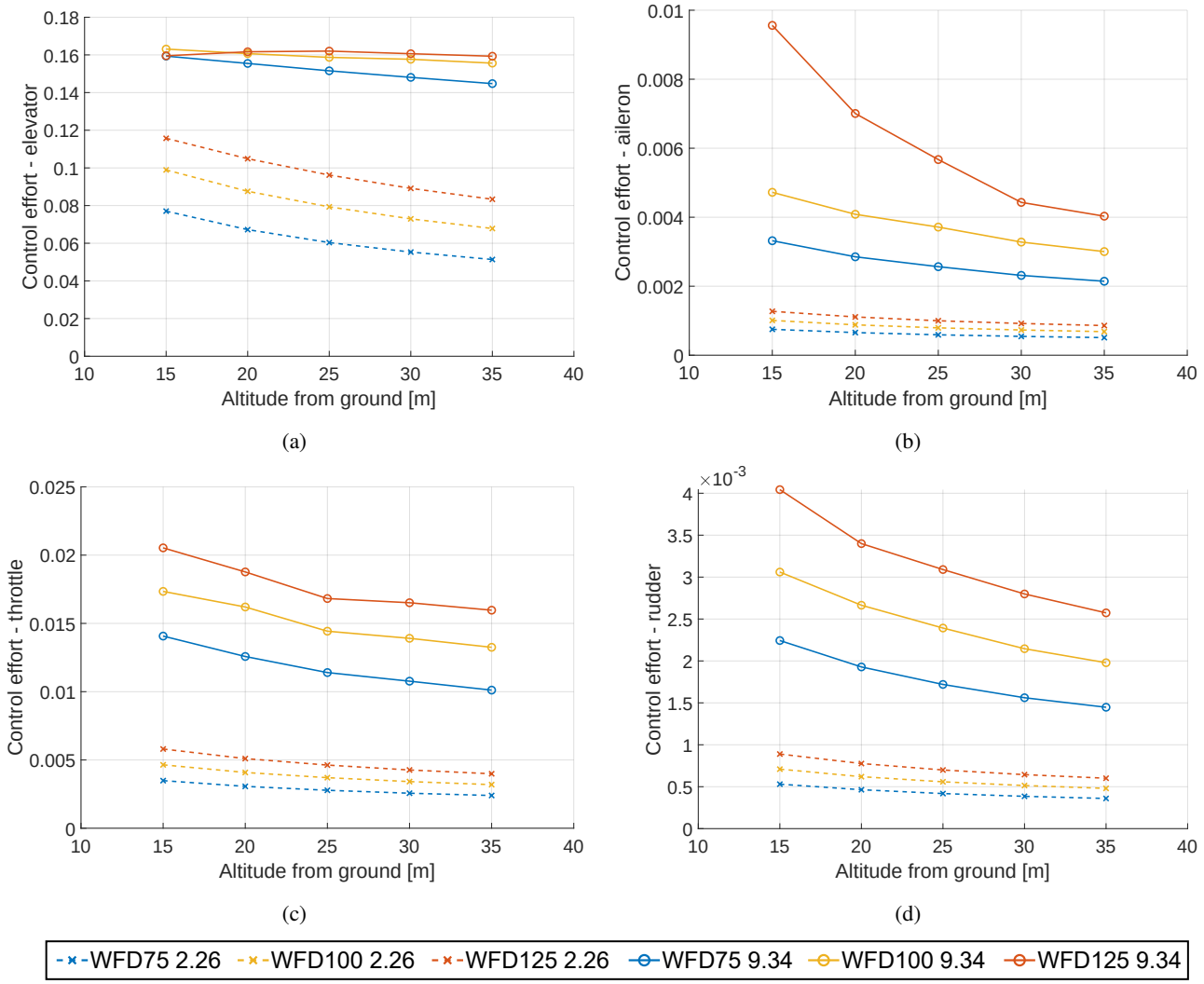


Figure 10: Control effort pattern for the control surface and throttle command versus the imposed flight height for six different atmospheric conditions: WFDi for $i = 75, 100, 125$ is the wind field with the Dryden noise, i being the percentage of W20 used for the simulation. Dashed lines represent the data for $W20 = 2.26\,\mathrm{m/s}$ and solid lines, $W20 = 9.34\,\mathrm{m/s}$. The mean CE value at each imposed height is shown in these figures. Figures 10a and 10c on the left-hand side show the effects in the longitudinal dynamics while Figures 10b and 10d present the patterns in the lateral dynamics.

birds' flight control requirements may be reduced at higher angle positions than at lower angles. It is important to highlight the fact that as the angle decreases for a constant radial distance, so does the height. Their hypothesis was that at higher angles, birds find a position with a greater velocity stability: lateral displacements do not have a strong effect because they remain in the same contour level and vertical displacements could appear to be self-stabilising. The results shown in Figure 10 are consistent with this hypothesis.

The CE definition used in this work takes into consideration the rate of change of the control surfaces and throttle input. This parameter allows for different paths and wind fields to be compared from a control point of view. However, it is important to note that the throttle value at which the SUAV is flying is also an important factor in the energy consumption of the vehicle and hence, a meaningful parameter to consider to improve SUAV flight performance. Unlike the control effort trends, the mean throttle value shows a strong dependency with the radial distance to the buildings. This trend indicates that there is an effect caused by the orographic lift (Fig. 8b). Figure 11 shows this trend for the six different wind conditions studied. These curves have been obtained by curve fitting the data from the successful paths: where the SUAV has not crashed against the ground or the

buildings during simulation. Because of this, the data points used to fit the curves in the case of $W20 = 9.34\,\mathrm{m/s}$ are limited in close proximity to the buildings. Note that the throttle command is defined in the limits [0, 1]. The curves in this figure suggest that the SUAV required less throttle input the stronger the wind field and the closer it was to the buildings. For the slower mean wind speed the level of disturbance had no significant effect on the mean throttle, but for the stronger wind field, a greater level of disturbance correlated with a lower mean throttle. However, as shown in Figure 12, for the faster wind field the closer the SUAV path was to the buildings the greater the risk of collision with increased levels of turbulence, with positions 16 and 17 crashing for WFD75 of $9.34\,\mathrm{m/s}$, positions 12, 16 and 17 for WFD100 and positions 11, 12, 16, 17 and 21 crashing for WFD125. These failed flights indicate the risk of flying close to the buildings under strong wind conditions.

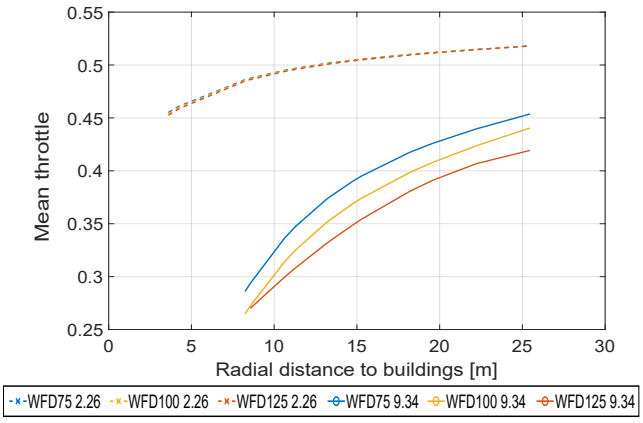


Figure 11: Mean throttle value respect to the radial distance to the buildings for six different atmospheric conditions: WFDi for $i = 75, 100, 125$ is the wind field with the Dryden noise, i being the percentage of W20 used for the simulation. Dashed lines represent the data for $W20 = 2.26\,\mathrm{m/s}$ and solid lines, $W20 = 9.34\,\mathrm{m/s}$.

Figure 13 is a visual representation of what has been shown and described above in Figures 10 and 11. Because all control surfaces show the same CE trend with the altitude, only the aileron parameter is presented in Figure 13b. These colour maps have been obtained as an interpolation of the successful flights for $WFD100$ of $W20 = 9.34\,\mathrm{m/s}$. The comparison with Figure 8 suggests that the mean throttle value is strongly dependant on the orographic updraughts induced by the buildings and that the CE of the control surfaces is mostly affected by the Dryden noise and the wind gradients in v at low altitudes and close around the buildings. A comparison between flight paths in Figure 13a indicates that flying at a short radial distance from the buildings could reduce the throttle command to up to 15% if compared to the farthest path studied, in the top left of the figure. However, as was stated previously, the closer the SUAV is flown to the buildings, the higher the chances of losing control and crashing. This probability is higher the stronger the wind field and the greater the level of turbulence (Figure 12).

Shepard *et al.* discussed the behaviour of gulls flying at different angles with the same radial distance to the buildings. Figure 14 presents two examples of this for WFD100. The resolution of the contour is $0.2\,\mathrm{m/s}$ in all sub-figures and the colorbar is kept constant for both wind fields, which is important to correctly interpret the results. The pairs A and B are situated at the same radial distance to the buildings, at different angles. These points correspond to positions 7 and 22 from Figure 8, respectively.

The comparison between the top and bottom figures ($W20 = 2.26\,\mathrm{m/s}$ and $W20 = 9.34\,\mathrm{m/s}$ respectively) highlights the difference between the spatial variation of the paths for the two wind fields. The wind gradient due to the horizontal displacement in the wind field is lower for the weakest wind field. This is due to the Dryden disturbance intensities being low. Therefore, the control effort required to fly in this wind field is lower than the one required with $W20 = 9.34\,\mathrm{m/s}$. However, the wind gradient is proportional to the nominal wind speed, meaning that the wind field will have the same effect on the wind gradient in both cases but at different scales.

Figures 14c and 14d together with the discussion above, suggest that position A requires more control effort to hold lateral and vertical position, in comparison to position B. The contour figures show that the lateral position of the SUAV moves into more levels when flying at lower angles compared to at a higher altitude. This variation in the contour levels results in a greater wind gradient which is added to the Dryden disturbance model in the flight dynamics model. These results suggest two things. Firstly, they confirm that flying at a lower altitude and under stronger wind conditions requires more control effort. Secondly, the magnitude of the wind components in the top figures compared to the bottom indicate that there is little benefit in terms of the required control effort to flying in any specific region of the wind field at weaker nominal wind speeds, which was also shown in Figures 10 and 11 when looking at the magnitude of the CE required.

Shepard *et al.* [25] hypothesised that the gulls were flying at higher angles relative to the building in stronger winds in order to reduce their control effort. This hypothesis is supported by the results of this study, with the control effort required for the SUAV reducing with altitude and the mean throttle required increasing with radial distance from the front corner of the building. With this pattern of effects the best compromise between reduced control effort and reduced requirement for thrust production is to fly directly above the front corner of the building where the altitude is maximised for a given radial distance. However at lower wind speeds

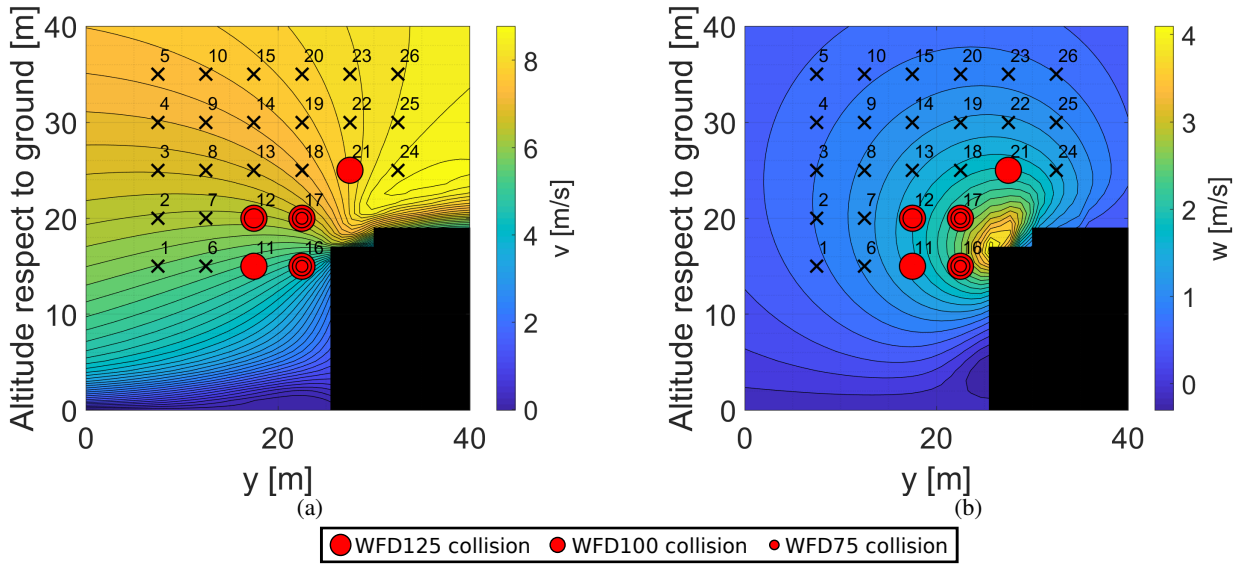


Figure 12: Contour plots of lateral and vertical components of the wind for $W20 = 9.34\,\mathrm{m/s}$ with the imposed vertical and lateral positions studied ('x' symbols) and the red circles symbolise the flights that crashed against the ground or the buildings during simulation.

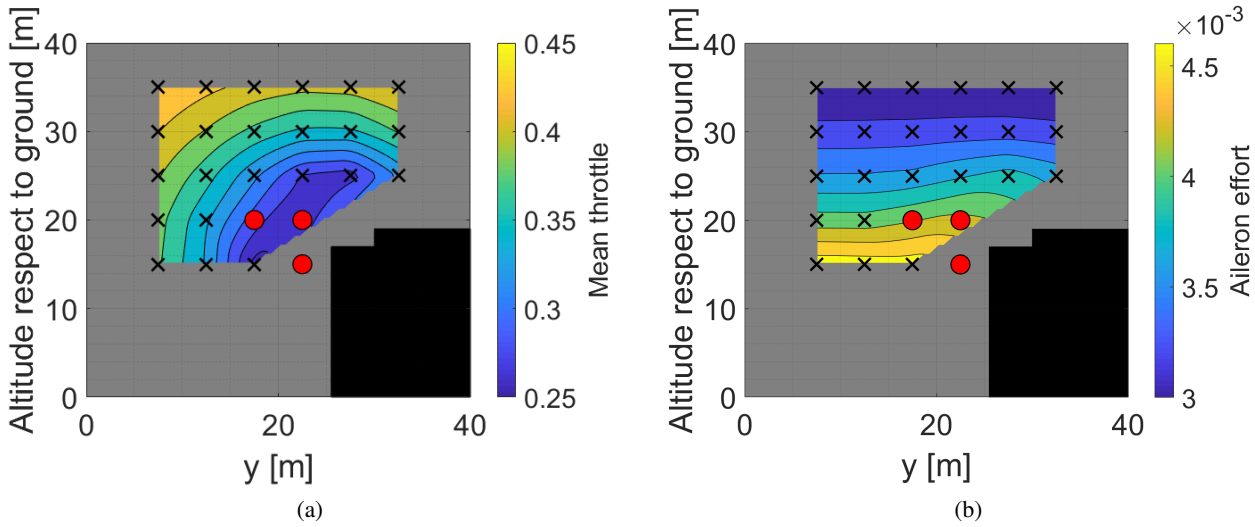


Figure 13: Contour plot of the mean throttle command and the CE of the aileron for $WFD100$ of $W20 = 9.34\,\mathrm{m/s}$. The colour maps are based on the interpolation of the successful flights. The red circles represent the flights that crashed against the ground or the buildings during simulation.

the gulls flew at lower angles out in front of the buildings and this behaviour is not explained by this optimisation. When looking at the simulation results it is apparent that in absolute terms there is not much of a change in control effort required with height at lower wind speeds so it may be that the gulls flew at lower angles to forage by the seafront, exploiting orographic lift whilst gaining a better view of the ground and find possible sources of food. This strategy may then become increasingly energetically costly as wind speed increases, along with having an increasing risk of collision with the buildings at higher wind speeds.

5 Conclusions

The work presented investigated the control effort required for a SUAV to fly parallel to buildings whilst affected by orographic soaring. The aim was to assess if there are any energetic benefits to flying in specific regions of the

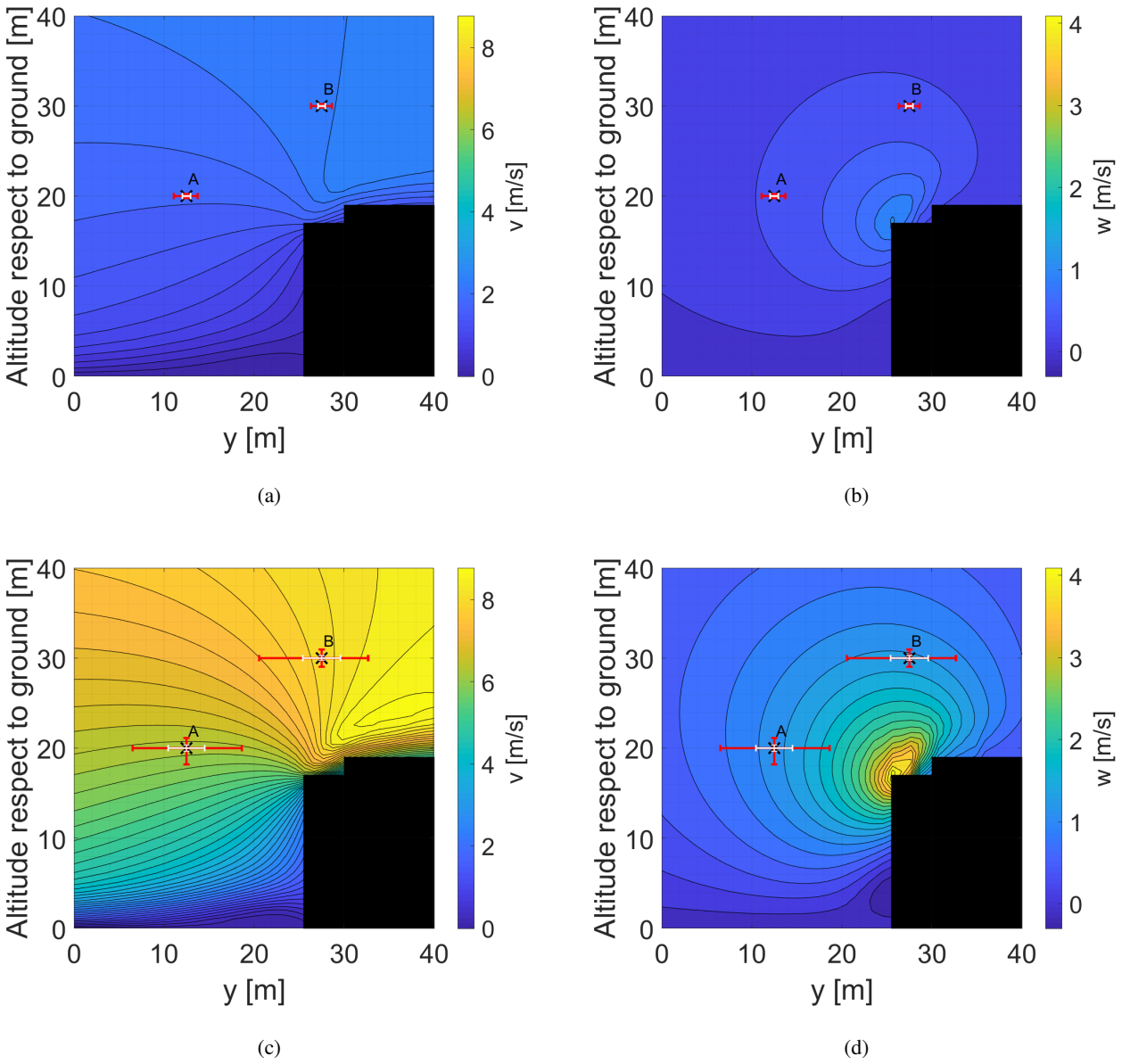


Figure 14: Positions A and B are given with the distribution of the lateral (v) and vertical (w) wind vector components in relation to the building. Positions A and B are at the same radial distance from the buildings. Red error bars indicate the horizontal and vertical flight range of the SUAV during the simulation; the white bars show the standard deviation of the position error during flight. 14a and 14b present the effects of $WFD100$ of $W20 = 2.26\,\mathrm{m/s}$ with a resolution of $0.2\,\mathrm{m/s}$ between levels. 14c and 14d show $WFD100$ of $W20 = 9.34\,\mathrm{m/s}$ with a resolution of $0.2\,\mathrm{m/s}$ between levels.

wind field. The WOT 4 Foam-E Mk2+ SUAV was simulated in Simulink to fly parallel to buildings by the seafront in Swansea. A range of different flight paths were imposed for a constant airspeed, and the control effort and path displacement were calculated and discussed for two different wind fields affected by the Dryden disturbance model. Overall, the findings of this study indicate that:

- The control effort is correlated with the nominal wind speed and the level of disturbance added with the Dryden model.
- The control effort is mostly correlated with the inverse of the height at which the SUAV is flying.
- Both strong and weak wind fields show a trend between the control effort and the imposed height; however, at

stronger nominal wind speeds, there is a greater advantage to flying at higher altitudes.

- The mean throttle command varies with the radial distance to the buildings, showing a benefit of up to 15% in the throttle command when flying next to them as opposed to flying in the farthest path defined in this study. However, the simulations have shown that flying in this region of the wind field for strong nominal wind speeds increases the risk of collision with the buildings.

- A strategy of flying directly over the front corner of the buildings appears to minimise the control effort required for a given level of orographic lift and mirrors the behaviour of gulls seen at higher wind speeds.

ACKNOWLEDGEMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 679355) and was supported by the EPSRC Centre for Doctoral Training in Future Autonomous and Robotic Systems (FARSCOPE) at the Bristol Robotics Laboratory.

REFERENCES

[1] G. Cai, J. Dias, and L. Seneviratne. A Survey of Small-Scale Unmanned Aerial Vehicles: Recent Advances and Future Development Trends. *Unmanned Systems*, 02(02):175–199, 2014.

[2] A.R. Girard, A.S. Howell, and J.K. Hedrick. Border patrol and surveillance missions using multiple unmanned air vehicles. *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, pages 620–625, 2004.

[3] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka. Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. *IEEE Robotics and Automation Magazine*, 19(3):46–56, 2012.

[4] CBP Public Affairs. U.S. Customs and Border Protection.

[5] S. Qazi, A. S. Siddiqui, and A. I. Wagan. UAV based real time video surveillance over 4G LTE. *ICOSST 2015 - 2015 International Conference on Open Source Systems and Technologies, Proceedings*, pages 141–145, 2016.

[6] CA. Thiels, JM. Aho, SP. Zietlow, and DH. Jenkins. Use of unmanned aerial vehicles for medical product transport. *Air Medical Journal*, 34(2):104–108, 2015.

[7] CC. Murray and AG. Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109, 2015.

[8] C Yuan, Y Zhang, and Z Liu. A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques. *Canadian Journal of Forest Research*, 45(7):783–792, 2015.

[9] M Erdelj and E Natalizio. UAV-assisted disaster management: Applications and open issues. *2016 International Conference on Computing, Networking and Communications (ICNC)*, pages 1–5, 2016.

[10] M. Bronz, J.M. Moschetta, P. Brisset, and M. Gorraz. Towards a Long Endurance MAV. *European Micro Aerial Vehicle Conference and Flight Competition*, pages 241–254, 2009.

[11] 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.

[12] S. Watkins, J. Milbank, B.J. Loxton, and W.H. Melbourne. Atmospheric Winds and Their Implications for Microair Vehicles. *AIAA Journal*, 44(11):2591–2600, 11 2006.

[13] S. Watkins, M. Thompson, B. Loxton, and M. Abdulrahim. On Low Altitude Flight through the Atmospheric Boundary Layer. *International Journal of Micro Air Vehicles*, 2(2):55–68, 6 2010.

[14] S. Watkins, A. Fisher, A. Mohamed, M. Marino, R. Clothier, and S. Ravi. The Turbulent Flight Environment Close to the Ground and Its Effects on Fixed and Flapping Wings at Low Reynolds Number. *5th European Conference for Aerospace Sciences (EUCASS)*, (May 2014):1–10, 2013.

[15] A. Mohamed, P. Poksawat, S. Watkins, and A. Panta. Developing a Stable Small UAS for Operation in Turbulent Urban Environments. In *International Micro Air Vehicle Conference and Flight Competition (IMAV)*, pages 184–189, 2017.

[16] L.W. Traub. Range and Endurance Estimates for Battery-Powered Aircraft. *Journal of Aircraft*, 48(2):703–707, 2011.

[17] W. Shyy, M. Berg, and D. Ljungqvist. Flapping and flexible wings for biological and micro air vehicles. *Progress in Aerospace Sciences*, 35(5):455–505, 1999.

[18] J.W. Langelaan. Long Distance / Duration Trajectory Optimization for Small UAVs. *AIAA Guidance, Navigation and Control Conference*, (August):1–14, 2007.

[19] C. White, E. W. Lim, S. Watkins, A. Mohamed, and M. Thompson. A feasibility study of micro air vehicles soaring tall buildings. *Journal of Wind Engineering and Industrial Aerodynamics*, 103:41–49, 2012.

[20] C. White, S. Watkins, E.W. Lim, and K. Massey. The Soaring Potential of a Micro Air Vehicle in an Urban Environment. *International Journal of Micro Air Vehicles*, 4(1):1–13, 2012.

[21] A. Fisher, M. Marino, R. Clothier, S. Watkins, L. Peters, and J.L. Palmer. Emulating avian orographic soaring with a small autonomous glider. *Bioinspiration and Biomimetics*, 11(1), 2015.

[22] A. Mohamed, R. Carrese, D. F. Fletcher, and S. Watkins. Scale-resolving simulation to predict the updraught regions over buildings for MAV orographic lift soaring. *Journal of Wind Engineering and Industrial Aerodynamics*, 140:34–48, 2015.

[23] S. Watkins, A. Mohamed, A. Fisher, R. Clothier, R. Carrese, and D.F. Fletcher. Towards autonomous MAV soaring in cities: CFD simulation, EFD measurement and flight trials. *International Journal of Micro Air Vehicles*, 7(4):441–448, 2015.

[24] H. Tennekes. *The Simple Science of Flight: From Insects to Jumbo Jets*. MIT Press, Cambridge, Massachusetts, 2009.

[25] E.C. Shepard, C. Williamson, and SP. Windsor. Fine-scale flight strategies of gulls in urban airflows indicate risk and reward in city living. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 371(1704), 2016.

[26] JW. Langelaan. Gust Energy Extraction for Mini and Micro Uninhabited Aerial Vehicles. *Journal of Guidance, Control, and Dynamics*, 32(2):464–473, 2009.

[27] N.T. Depenbusch. *Atmospheric Energy Harvesting for Small Uninhabited Aircraft by Gust Soaring*. PhD thesis, Pennsylvania State University, Pennsylvania, USA, 2011.

[28] J. How. *Aircraft Stability and Control*. Massachusetts Institute of Technology: MIT OpenCourseWare, License: Creative Commons BY-NC-SA.

[29] Department of Defense. Flying Qualities of Piloted Aircraft. *Department of Defense Handbook Mil-HDBK-1797B*, 2012.

[30] B. Etkin. *Dynamics of Flight. Stability and Control*. John Wiley & Sons, second edition, 1982.

APPENDIX A: FLIGHT DYNAMIC EQUATIONS AND PARAMETERS

The flight dynamic equations used in this work are presented below expressed in the wind reference system and following the nomenclature in [30] with the exception of the roll moment components which are discriminated with a caret ($\widehat{L}$).

$$\dot{V_a} = -\frac{qw - vr}{V_a} + \frac{Tcos(\alpha)cos(\beta) - D}{m} + g_1 - \frac{dw_x}{dt}d_1 - \frac{dw_y}{dt}d_2 - \frac{dw_z}{dt}d_3 \tag{7}$$

$$\dot{\beta} = \frac{pw - ur}{V_a} - \frac{-C + Tcos(\alpha)sin(\beta)}{V_a m} + \frac{g_2}{V_a} - \frac{\frac{dw_x}{dt}d_4 + \frac{dw_y}{dt}d_5 + \frac{dw_z}{dt}d_6}{V_a} \tag{8}$$

$$\dot{\alpha} = -\frac{pv - uq}{V_a} - \frac{Tsin(\alpha) + L}{V_a m} + \frac{g_3 - \frac{dw_x}{dt}d_7}{V_a} - \frac{\frac{dw_y}{dt}d_8 + \frac{dw_z}{dt}d_9}{V_a} \tag{9}$$

$$\dot{p} = \frac{I_{zz}\widehat{L} + I_{xz}N - (I_{xz}(I_{yy} - I_{xx} - I_{zz})p)q}{\tau} + \frac{((I_{xz}^2 + I_{zz}(I_{zz} - I_{yy}))r)q}{\tau} \tag{10}$$

$$\dot{q} = \frac{M - (I_{xx} - I_{zz})pr - I_{xz}(p^2 - r^2))}{I_{yy}} \tag{11}$$

$$\dot{r} = \frac{I_{xz}\widehat{L} + I_{xx}N + (I_{xz}(I_{yy} - I_{xx} - I_{zz})r)q}{\tau} + \frac{(I_{xz}^2 + I_{xx}(I_{xx} - I_{yy}))p)q}{\tau} \tag{12}$$

$$\dot{\phi} = p + (qsin(\phi) + rcos(\phi))tan(\theta) \tag{13}$$

$$\dot{\theta} = qcos(\phi) - rsin(\phi) \tag{14}$$

$$\dot{\psi} = (qsin(\phi) + rcos(\phi))sec(\theta) \tag{15}$$

The transformation matrix from body reference system to an inertial (NED) reference system and the transformation matrix from wind to body reference systems are defined in Equations 16, 17 respectively.

$$L_{ib} = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{16}$$

$$L_{bw} = \begin{bmatrix} c_\alpha c_\beta & -c_\alpha s_\beta & -s_\alpha \\ s_\beta & c_\beta & 0 \\ s_\alpha c_\beta & -s_\alpha s_\beta & c_\alpha \end{bmatrix} \tag{17}$$

s_k and c_k are the sine and cosine of k, where k can be an Euler angle: roll ϕ, pitch θ and yaw ψ or angle of attack or side-slip, α and β respectively.
The transformation matrix from the NED inertial system to the wind reference frame:

$$L_{wi} = L_{bw}^T \cdot L_{ib}^T = \begin{bmatrix} d_1 & d_2 & d_3 \\ d_4 & d_5 & d_6 \\ d_7 & d_8 & d_9 \end{bmatrix} \tag{18}$$

Parameters *d* and *g* in the flight dynamic equations of motion (Eqs 7 - 9) are defined in Eq 18 and 19.

$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = L_{wi} \cdot \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{19}$$

Airspeed expressed in the body reference system:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = L_{bw} \cdot \begin{bmatrix} V_a \\ 0 \\ 0 \end{bmatrix} \tag{20}$$

The position of the aircraft expressed in the inertial NED reference system is defined as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = L_{wi}^T \begin{bmatrix} V_a \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{21}$$

The aerodynamic forces [L C D] and moments [$\hat{\text{L}}$ M N] and thrust (T) are defined as:

$$\begin{bmatrix} L \\ C \\ D \end{bmatrix} = \frac{1}{2}\rho V_a^2 S \begin{bmatrix} C_l \\ C_Y \\ C_d \end{bmatrix} \tag{22}$$

$$T = \frac{1}{2}\rho S C_{t_k} C_t \tag{23}$$

$$\begin{bmatrix} \widehat{L} \\ M \\ N \end{bmatrix} = \frac{1}{2}\rho V_a^2 S \begin{bmatrix} b \cdot C_{\widehat{l}} \\ c \cdot C_m \\ b \cdot C_n \end{bmatrix} \tag{24}$$

The aerodynamic coefficients:

$$C_l = C_{l_\alpha}(\alpha + \alpha_0) + C_{l_q}\frac{c}{2V_m}q + C_{l_{\delta_e}}\delta_{ele} \tag{25}$$

$$C_Y = C_{Y_\beta}\beta + C_{Y_{\delta_a}}\delta_{ail} + C_{Y_{\delta_r}}\delta_{rdd} \tag{26}$$

$$C_d = C_{d_0} + C_{d_\alpha}\alpha + C_{d_{\alpha_2}}\alpha^2 \tag{27}$$

$$C_t = C_{t_{dt_2}}\delta_{thr}^2 \tag{28}$$

$$C_{\widehat{l}} = C_{\widehat{l}_\beta}\beta + C_{\widehat{l}_p}\frac{b}{2V_m}p + C_{\widehat{l}_r}\frac{b}{2V_m}r + C_{\widehat{l}_{\delta_a}}\delta_{ail} + C_{\widehat{l}_{\delta_r}}\delta_{rdd} \tag{29}$$

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}\frac{c}{2V_m}q + C_{m_{de}}\delta_{ele} \tag{30}$$

$$C_n = C_{n_\beta}\beta + C_{n_p}\frac{b}{2V_m}p + C_{n_r}\frac{b}{2V_m}r + C_{n_{da}}\delta_{ail} + C_{n_{dr}}\delta_{rdd} \tag{31}$$

where V_m is the equilibrium velocity used to estimate the coefficients.

APPENDIX B: DRYDEN WIND DISTURBANCE EQUATIONS

Dryden Model MIL-F-8785C [29] equations for low-altitude are defined as:

- Power spectral densities:

$$\Phi_{u_g}(\Omega) = \sigma_u^2 \cdot \frac{2L_u}{\pi}\frac{1}{1 + (L_u\Omega)^2} \tag{32}$$

$$\Phi_{v_g}(\Omega) = \sigma_v^2 \cdot \frac{2L_v}{\pi}\frac{1 + 12(L_v\Omega)^2}{(1 + 4(L_v\Omega)^2)^2} \tag{33}$$

$$\Phi_{w_g}(\Omega) = \sigma_w^2 \cdot \frac{2L_w}{\pi}\frac{1 + 12(L_w\Omega)^2}{(1 + 4(L_w\Omega)^2)^2} \tag{34}$$

- Turbulence intensities:

$$L_w = h \tag{35}$$

$$L_u = L_v = \frac{h}{(0.177 + 0.000823 \cdot h)^{1.2}} \tag{36}$$

- Turbulence scale lengths:

$$\sigma_w = 0.1 \cdot W20 \tag{37}$$

$$\sigma_u/\sigma_w = \sigma_v/\sigma_w = \frac{1}{(0.177 + 0.000823 \cdot h)^{0.4}} \tag{38}$$

- Transfer function of the Dryden Model

$$G_{u_g}(s) = \sigma_u \sqrt{\frac{2L_u}{\pi V}} \frac{1}{1 + \frac{L_u}{V}s} \tag{39}$$

$$G_{v_g}(s) = \sigma_v \sqrt{\frac{2L_v}{\pi V}} \frac{1 + \frac{2\sqrt{3}L_v}{V}s}{(1 + \frac{2L_v}{V}s)^2} \tag{40}$$

$$G_{w_g}(s) = \sigma_w \sqrt{\frac{2L_w}{\pi V}} \frac{1 + \frac{2\sqrt{3}L_w}{V}s}{(1 + \frac{2L_w}{V}s)^2} \tag{41}$$

APPENDIX C: WOT4 CHARACTERISTICS

The simulation parameters and aerodynamic coefficients used in this work are gathered in this section in Tables 1 and 2, respectively. Table 3 and Table 4 contain the control gains and the servo and electric motor natural frequencies and damping ratios.

- Pitch and control surfaces maximum and minimum angle permitted.

$$\begin{gathered} -40^\circ \leq \theta \leq 40^\circ \\ -15^\circ \leq \delta_{ele} \leq 15^\circ \\ -18^\circ \leq \delta_{ail} \leq 18^\circ \\ -29^\circ \leq \delta_{rdd} \leq 29^\circ \end{gathered}$$

Table 1: WOT 4 simulation parameters

Physical constants		
Parameter	Value	Units
g	9.81	$\mathrm{m/s^2}$
ρ	1.225	$\mathrm{kg/m^3}$
Aircraft model parameters		
Parameter	Value	Units
V_m	18	$\mathrm{m/s}$
S	0.3	$\mathrm{m^2}$
c	0.254	m
b	1.206	m
m	1.345	kg
I_{xx}	5.1×10^{-2}	$\mathrm{kg\,m^2}$
I_{yy}	7.8×10^{-2}	$\mathrm{kg\,m^2}$
I_{zz}	1.12×10^{-1}	$\mathrm{kg\,m^2}$
I_{xz}	1.5×10^{-3}	$\mathrm{kg\,m^2}$

Table 2: WOT 4 aerodynamic coefficients

Parameter	Value	Parameter	Value
C_{t_k}	5	$C_{t_{dt2}}$	10.60
C_{d_0}	0.03	C_{Y_β}	-4.31×10^{-1}
C_{d_α}	0.48	$C_{Y_{\delta_a}}$	2.03×10^{-2}
$C_{d_{\alpha 2}}$	1.26	$C_{Y_{\delta_r}}$	3.71×10^{-2}
C_{α_0}	4.44×10^{-3}	C_{m_0}	4.22×10^{-3}
C_{l_α}	3.89	C_{m_α}	-1.01×10^{-1}
C_{l_q}	1.04×10^{-1}	C_{m_q}	-4.84
$C_{l_{\delta_e}}$	-4.24×10^{-1}	$C_{m_{\delta_e}}$	-3.02×10^{-1}
$C_{\widehat{l}_\beta}$	-7.74×10^{-3}	C_{n_β}	4.04×10^{-2}
$C_{\widehat{l}_p}$	-5.09×10^{-2}	C_{n_p}	-1.26×10^{-2}
$C_{\widehat{l}_r}$	3.13×10^{-2}	C_{n_r}	-1.65×10^{-1}
$C_{\widehat{l}_{\delta_a}}$	-2.11×10^{-2}	$C_{n_{\delta_a}}$	-6.39×10^{-4}
$C_{\widehat{l}_{\delta_r}}$	-2.54×10^{-3}	$C_{n_{\delta_r}}$	-4.13×10^{-2}

Table 3: WOT 4 PID gains

Altitude guidance	
Parameter	Value
K_{p_h}	0.3
K_{i_h}	0.3
$K_{p_{\dot{h}}}$	-0.1455
Lateral guidance	
Parameter	Value
K_{p_y}	0.03
K_{i_y}	0.03
Pitch angle controller	
Parameter	Value
K_{p_θ}	-3
K_{p_q}	1
K_{ff}	-0.09
Velocity controller	
Parameter	Value
K_{p_V}	0.4118
K_{i_V}	0.0343
Lateral controller	
Parameter	Value
K_{p_ϕ}	-4.896
K_{i_ϕ}	-1
K_{p_p}	-2
K_{p_r}	0.0912

Table 4: Servo and electric motor characteristics

Parameter	ω_n [rad/s]	ζ
Aileron servo	100	0.9
Elevator servo	23	0.9
Rudder servo	23	0.9
Electric motor	15	0.9

Aerial Interaction Control Using Gain-Scheduling and PID for a Drone with a 2-DOF Arm

Aaron Lopez Luna*, Jose Martinez Carranza, and Israel Cruz Vega
National Institute of Astrophysics, Optics and Electronics

ABSTRACT

One of the significant challenges of unmanned aerial vehicles (UAV) is the physical interaction with an object or a rigid structure. There are numerous potential benefits of physical interaction with the environment. However, the process of approaching a UAV to an object or a surface also brings challenging control problems. This paper addresses the control problem of a UAV endowed with a robotic arm to physically contact a rigid structure. The proposed control technique is based on a Gain-Scheduled Proportional-Integral-Derivative (GS-PID) algorithm. Our previous work [1] based on a simple Gain-Scheduling (GS) approach for the stability of the robotic aerial system is insufficient to counteract the disturbances during the interaction successfully. Therefore, the proposed GS-PID control can respond to the disturbances induced by the wall-effect, the arm's movement, and the contact with the rigid structure. Experimental testing results demonstrate satisfactory performance of the proposed control strategy.

1 INTRODUCTION

Aerial manipulators (robotic manipulator arms attached to aerial vehicles) research has grown in recent years due to the importance and potential applications of this useful systems in industrial and commercial fields. [2, 3]. Nevertheless, this new configuration represents a new problem in the stability control of the aerial vehicle. The movements of a manipulator attached to a UAV during flight mode are considered as disturbances, which can cause instability and the loss of the entire system. New models and control algorithms have been proposed to prevent this situation [4, 5]. The UAV physical interaction may provide excellent solutions as well as reliable operations, e.g., the inspection of a surrounding environment. Moreover, the concept called flying hand is a unique and leading technology. However, the interaction control imposes inherent nonlinearities due to not only the dynamic behavior of the UAV but also the interaction between the surface and the system [6, 7]. In all the situations in which contacts between the aerial vehicle and the environment occur, the dynamics of the system may dramatically change, and the development of a robust control law able to handle all the possible interactions becomes a challenge [7, 8].

In this work, we consider the proposal of taking into account dynamical changes in the UAV due to three essential factors. i)The movement of the arm attached to the aerial vehicle, ii) the wall-effect disturbance when the system is approaching the rigid structure, and iii) the effect of the contact with the surface. We also propose an experimental study to determine the variation in plant dynamics with the three factors and establish a set of controllers which allows approximating the real trajectory of the system to the desired trajectory. Therefore in this work, we employ a novel manipulator arm of two degrees of freedom (DOF) especially developed for a commercial quadcopter parrot bebop-2. We also propose to incorporate a GS approach to the Classical PID control to ensure the stability of the proposed aerial manipulator. We determine the gain values of a set of PID controllers by the experimental study; this represents a novel technique to deal with the drawbacks of perturbations in the aerial manipulation systems.

This paper is organized as follow: In section 2, the proposed system is described. The GS-PID control technique developed for this work is presented in section 3.1. To prove the effectiveness of the proposed strategy, a set of experiments are implemented and are described in section 4; the experimental results are also shown in this section. Finally, the main contribution, conclusions, and future direction are described in section 6.

2 DESCRIPTION OF PROPOSED SYSTEM

Nowadays exist different configurations of UAV systems used for several missions in commercial and industrial tasks. Each configuration has distinct advantages and disadvantages according to its design. Vertical Take-Off and Landing (VTOL) vehicles have been taken into account especially for aerial manipulation due to specific aspects of their flight mode which are used to achieve the primary goal of maintaining a manipulator robot in the desired point. In this work, we designed an aerial, consisting of two main subsystems: The aerial vehicle of four rotors and a robotic arm of two DOF. In the following subsections, each one of this system is presented.

*Email address(es): aaron.eleazar@inaoep.mx

2.1 ROBOTIC ARM

For manipulation and interaction task, robotic arms can provide the necessary degrees of freedom to achieve the objective [9, 10]. In contact with the environment, for example, an n-DOF arm could supply the stiffness and versatility to the vehicle to accomplish the goal involving contact with a rigid structure. The n-DOF arm could also provide a safe distance between the aerial system and the structure.

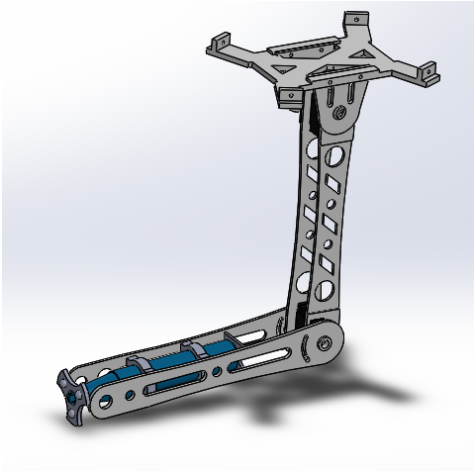


Figure 1: CAD model of proposed robotic arm.

The robotic arm of Fig. (1) was developed thinking in two main aspects - first, the physical task. The task consists in to exert a force on a rigid surface. For this objective, the robotic arm must provide the movements necessaries to contact the surface successfully and also must provide adequate distance between the vehicle and the surface to reduce the disturbances induced by the proximity of the rigid structure with the rotors of the aerial vehicle. Second, the dimensions of the proposed design must maintain a relationship with the physical capabilities of the system in order to guarantee the ideal performance in flight mode. Even with this consideration, the behavior of the robotic arm can alter the efficiency of the vehicle, for this reason, a control technique considering the perturbations of the robotic arm must be implemented to achieve the proposed task. The length of the extended arm designed for this work provides sufficient distance between surface and the aerial system to maintain a safety flight and reduce the wall effect disturbance. the arm structure is light enough to allow the aerial system to take off suitably and keep a stable flight, however, the poor stiffness of the arm it must be considered in the control design to prevent exceed supported force of the arm. In the following sections, the problem of perturbations and the proposed solution is handled.

2.2 END EFFECTOR

To prove the satisfactory performance of the interaction control, we proposed an experimental scenario in which the aerial manipulator marks the points where the end-effector of the arm and the surface are in contact. A set of pieces were designed, allowing the arm to hold a pencil and mitigate the friction and contact forces. Fig. (2) shows the proposed end-effector for the experimental task. The design of the contact end-effector permit to reduce the interaction area to a three simple points, reducing at the same time the friction in the contact phase.

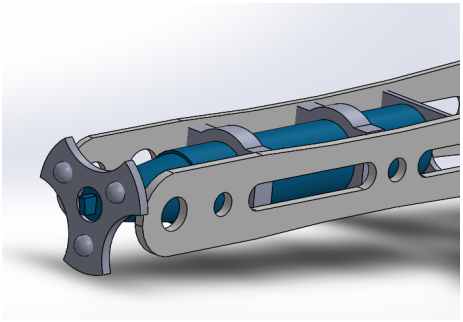


Figure 2: CAD model of end effector for contact whit a surface.

2.3 PHYSICAL SYSTEM

Due to the four rotors, the quadcopter has more lifting power than a helicopter of the same size, allowing carry a more massive payload. The interest in this kind of configuration comes not only from its dynamics, which represent an attractive control problem but also from the design issue [11, 12, 13, 14]. The rotors of a quadcopter work together to lift the weight of the quadcopter airborne. Quadcopters have being used in search and rescue missions, surveillance, inspection, mapping, and law enforcement [15]. Fig. (3) shows the complete physical system, the bebop-2 with the 2-DOF arm.

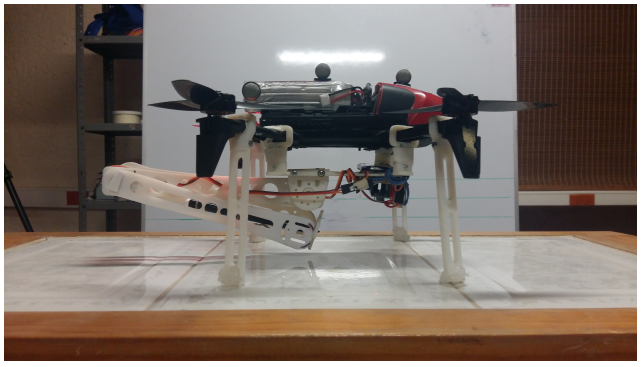


Figure 3: Aerial manipulator, physical implementation.

3 AERIAL INTERACTION CONTROL

In this work, the PID control and the Gain-Scheduling approach are developed to design and implement an interaction control for a bebop-2 vehicle with a 2-DOF arm. The following section describes the architecture of the controls and the system structure.

3.1 PID CONTROL

A PID controller continuously calculates an error value $e(t)$ as the difference between the desired set point and the measured process variable and applies a correction based on proportional, integral, and derivative terms, (sometimes denoted P, I, and D respectively). The following equation (1) describes the PID algorithm:

$$u(t) = k_p e(t) + k_I \int e(\tau)d\tau + k_D \frac{d}{dt}e(t) \qquad (1)$$

where k_p, k_I, k_D are the PID control gains, $u(t)$ is the control signal and $e(t)$ is the error signal. The integral, proportional, and derivative parts are interpreted as control actions based on the dynamics of the signal. The PID gains can be designed based upon the system parameters with a certain precision. In this work, a PID controller is designed for the x and y position and yaw orientation of the aerial vehicle.

3.2 GAIN SCHEDULING CONTROLLER

To compensate the disturbances of the robotic arm, we incorporate the Gain-Scheduling technique into the PID control, enhancing the performance of the flight vehicle with a robotic arm. The gain-scheduling method uses measurable variables correlating changes in the dynamic process to define controller parameters. The gain schedule technique is an acceptable approach to control nonlinear systems using a set of linear controllers, providing adequate control responses to various operational points of the system. To tune the controller, we need to select one or more adjustment variables.

After the selection of these variables, the regulator parameters are calculated for several operation points. In this work, for the designing of the GS-PID controller, a set of pre-tuned gains are applied to the controllers. No rule specifies the number of zones or operation points for the division in the range of operation of the plant, the designer decides in this respect [16, 17, 18]. To implement the GS-PID controller, we follow the next steps: first, to chose auxiliary variables. In this work, these variables are the angles of the links of the robotic arm (θ_1, θ_2). Second, after choosing the auxiliary variables, the operation points P_n are determined. These operation points are the position P_e in (X, Y, Z) of the end effector of the arm which depends on the values of θ_1 and θ_2 (More detail in [1]). The controller actions readjusted for each operating condition. The calculated parameters are the gains K_p, K_I, and K_D of the PID control, which now depends on the values of θ_1 and θ_2. The equation (1) with the gain-scheduling method now is rewritten as follow:

$$u = k_p(\theta_{1,2})e + k_I(\theta_{1,2}) \int e(\tau)d(\tau) + k_D(\theta_{1,2})\frac{d}{dt}e(t) \quad (2)$$

where $k_P(\theta_{1,2})$, $k_I(\theta_{1,2})$, $k_D(\theta_{1,2})$ are the PID control gains for every operational condition P_n. The following section describes the implementation of GS-PID controller for the aerial vehicle.

3.3 WALL-EFFECT AND CONTACT DISTURBANCE

When a UAV in flight mode is close to a wall (vertical rigid structure) a disturbance force induced by the propellers of the aerial vehicle affects the stability of the system, this disturbance increase when the distance between the vehicle and the surface is smaller. Due to the length of the arm, it is not sufficient to maintain the aerial vehicle far enough to decrease the wall-effect, so such disturbance must be considered to achieve the interaction task satisfactorily. The contact with the surface also generate a rejection force affecting the flight stability; this rejection force depends on the contact force exerted by the arm. To simplify the interaction control, the contact force of the arm is considered to stay at a minimum value. We conducted experiments to obtain both the wall-effect and the contact rejection force.

4 EXPERIMENTAL SETUP

The proposed control strategy was proved via experimental tests in a controlled environment. The manipulator incorporated to bebop-2 is composed of 2 links with a dimension of 10 cm (L), 0.3 cm (W) and 4 cm (H), each one with 0.12 kg (m). In order to compensate the disturbance of the arm with the GS-PID controller for the position (X, Y, Z) of the system, a set of ten operation points were chosen carefully for each controller. Fig. (4) shows the operation points of the system depending on the angles of the links.

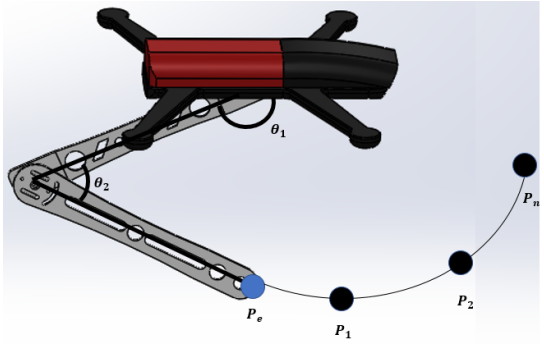


Figure 4: Representation of operating points of the system [1].

Just like in the previous work [1] the VICON cameras were used to get the actual position (X, Y, Z) of the system continually and monitoring the variables θ_1 and θ_2 of the links of the robotic arm.

Fig. (5) depicts the block diagram for this research. In this representation $E(t)$ is the error between the desired position and the actual position, $Y(t)$ represents the control signals sent to the aerial vehicle to control the (x, y, z) position. The real position of the aerial vehicle during flight mode is taken from the VICON system providing information used to calculate $E(t)$. The different positions of the robotic arm produce different disturbances which can be calculated using the error as a reference to the displacement of the system; this instability is represented as noise in the block diagram affecting the position of the vehicle directly. The objective is to compensate for this displacement to maintain the aerial vehicle in the desired position.

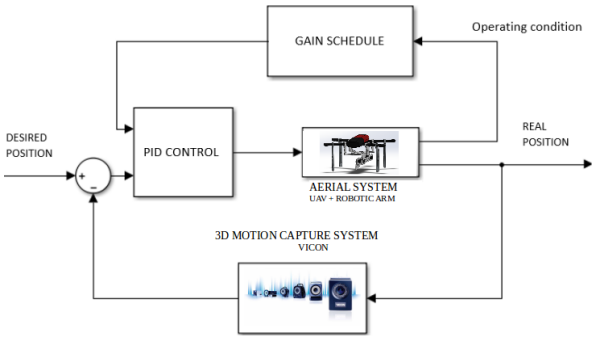


Figure 5: Block diagram of the proposed system with GS-PID control [1].

A set of gains were obtained experimentally to compensate the disturbances and upgrade the performance of the control maintaining the system in the desired position. Table 1 shows the value of K_p, K_I and K_D for each operating point.

θ_1/θ_2	K_P	K_I	K_D
0/0	0.22	0.12	0.03
30/0	0.45	0.37	0.09
30/30	0.92	0.64	0.17
60/30	1.3	0.9	0.22
90/30	1.6	1.2	0.56
120/30	1.72	1.34	0.78
150/30	2.1	1.66	1.05
150/90	2.22	1.79	1.12
150/120	2.45	1.89	1.2
150/150	2.52	1.98	2.31
near to wall	4.56	2.02	2.62

Table 1: set of gains for the operating points

Additionally, to the gains obtained in the previous work [1], a set of experiments were made to calculate the wall-effect affecting the stability of the system. Fig. (6) shows the experiment where the system is near to a vertical surface.

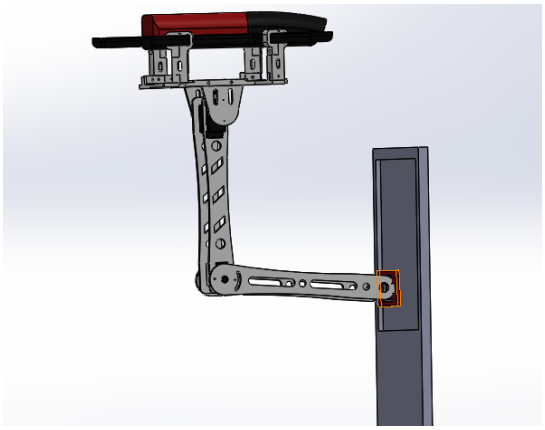


Figure 6: Representation of desired interaction task.

5 RESULTS

This section contains the experimental results of the proposed system. Three different experiments were designed, to demonstrate the effectiveness of the GS-PID interaction control. First, analyzing the behavior of the system when it is trying to approximate the surface employing a standard PID control. The x-axis of the graph represents time in milliseconds (ms), and the y-axis of the graph represents the position in millimeters (mm). The desired position of the system is (15,0, 1), at 20 cm near the surface. Fig. (7) shows the response of the system. The system is unable to reach the references in the x-axis due to the wall-effect.

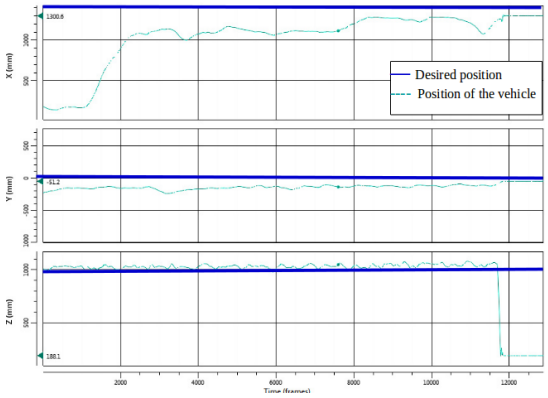


Figure 7: Behavior of the aerial vehicle carrying the robotic arm with standard PID control.

In the second experiment, GS-PID control is implemented [1]. This time, the control includes the set of gains required to attenuate the wall-effect, Fig. (8) shows the behavior of the system, the GS-PID control can lead the system to the reference.

In the third experiment, the end-effector is now in contact with the rigid surface, and the system executes a horizontal movement to follow a linear trajectory in the surface. The

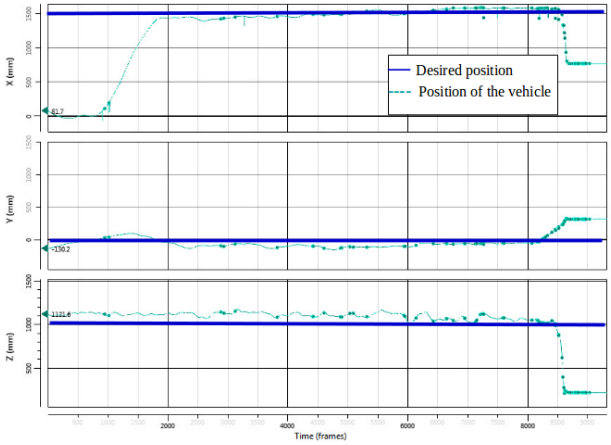


Figure 8: Behavior of the aerial vehicle with the robotic arm extended near to rigid surface, GS-PID control

GS-PID control maintains the system at the required distance to the surface to achieve the task. Fig. (9) shows the successful contact of the system with the surface.

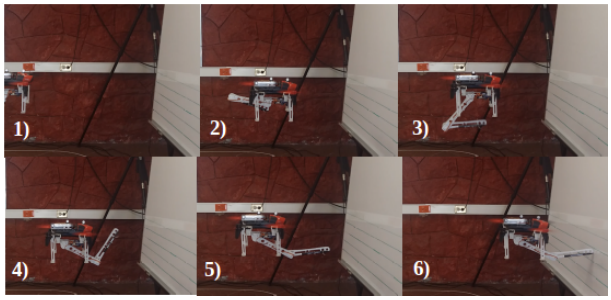


Figure 9: Sequence of successful contact of the aerial manipulator with the rigid surface.

Fig. (10) shows the response of the system in full contact with the surface. The proposed control maintains a constant distance between the system and the surface while the manipulator moves in the y-axis, drawing the continuous line over the surface.

The final result proves the effectiveness of the proposed control to interact with the environment. For quantitatively compare the control performance, we defined the following function:

$$mse = \frac{1}{n}\sum_{i=1}^{n}(p_d - p_r)^2 \tag{3}$$

this equation describes the mean squared error (mse) between p_d, as the desired position, and p_r the real position of the system in the step time i. The quantitative results for each controller are given in Table 2. We can observe an upgrade in the performance and response in comparison with the classical PID control.

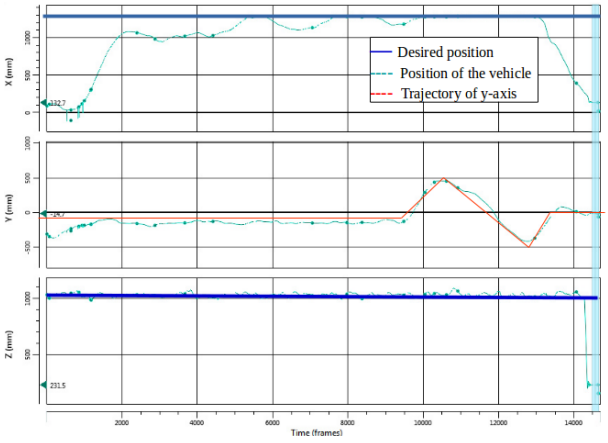


Figure 10: Behavior of the aerial manipulator in full contact with the rigid surface, GS-PID control.

mse	PID control	GS-PID control.
X	42.56	2.12
Y	26.44	1.23
Z	7.65	0.17

Table 2: Mean squared error.

In Figure 11 a line drawn by the drone is shown. The line proves the effectiveness of the proposed control to maintain the system in constant contact with the structure. We aim to upgrade the drawn line as future work.

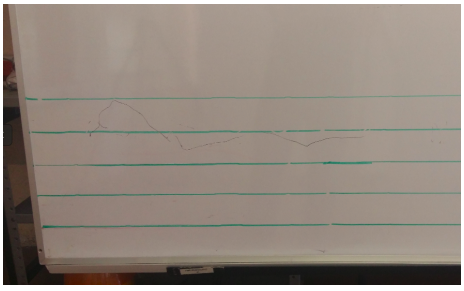


Figure 11: Line drawn by the drone.

6 CONCLUSIONS

This paper presents results regarding a control strategy to maintain a stable flight of the vehicle during the interaction task. The present work focuses on dealing with the stability problem of the arm, the wall-effect, and the contact with a surface. The strategy involves the use of GS- PID controller, where a set of operating points are carried out to create a group of controllers designed to eliminate the disturbances induced on the vehicle by the arm's motion. The conducted experiments prove that wall-effect degrades the efficiency of the GS-PID control, but the proper measure of

these dynamic changes are considered in the control scheme to upgrade the performance of the system. The results indicate that the proposed approach is an improvement step towards the development of specific tasks of aerial manipulation systems. The main contribution of this work is the design of a GS-PID control technique for aerial interaction. The next step is to achieve a complete trajectory over the surface and replicate the results in outdoor scenarios. A video with the experiments and results is available the following link: https://youtu.be/mOFIo2YJJTE,

REFERENCES

[1] A. L. Luna, I. C. Vega, and J. M. 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.

[2] C. D. Bellicoso, L. R. Buonocore, V. Lippiello, and B. Siciliano. Design, modeling and control of a 5-dof light-weight robot arm for aerial manipulation. In *Control and Automation (MED), 2015 23th Mediterranean Conference on*, pages 853–858, June 2015.

[3] D. Bazylev, A. Kremlev, A. Margun, and K. Zimenko. Design of control system for a four-rotor uav equipped with robotic arm. In *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2015 7th International Congress on*, pages 144–149, Oct 2015.

[4] 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.

[5] Fabio Ruggiero, Jonathan Cacace, Hamid Sadeghian, and Vincenzo Lippiello. Passivity-based control of {VToL} {UAVs} with a momentum-based estimator of external wrench and unmodeled dynamics. *Robotics and Autonomous Systems*, 72:139 – 151, 2015.

[6] Basaran Bahadir Kocer, Tegoeh Tjahjowidodo, and Gerald Gim Lee Seet. Model predictive uav-tool interaction control enhanced by external forces. *Mechatronics*, 58:47 – 57, 2019.

[7] B. B. Kocer, G. S. G. Lee, and T. Tjahjowidodo. Nonlinear predictive uav-elastic tool interaction control in real-time. In *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 472–477, July 2018.

[8] Lorenzo Marconi, Roberto Naldi, and Luca Gentili. Modelling and control of a flying robot interacting with the environment. *Automatica*, 47(12):2571 – 2583, 2011.

[9] S. Kannan, M. Alma, M. A. Olivares-Mendez, and H. Voos. Adaptive control of aerial manipulation vehicle. In *Control System, Computing and Engineering (ICCSCE), 2014 IEEE International Conference on*, pages 273–278, Nov 2014.

[10] S. Bouabdallah and R. Siegwart. Full control of a quadrotor. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 153–158, Oct 2007.

[11] F. Huber, K. Kondak, K. Krieger, D. Sommer, M. Schwarzbach, M. Laiacker, I. Kossyk, S. Parusel, S. Haddadin, and A. Albu-Schffer. 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.

[12] M. I. Sanchez, J. A. Acosta Acosta, and A. Ollero. Integral action in first-order closed-loop inverse kinematics. application to aerial manipulators. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5297–5302, May 2015.

[13] I. Palunko, A. Faust, P. Cruz, L. Tapia, and R. Fierro. A reinforcement learning approach towards autonomous suspended load manipulation using aerial robots. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4896–4901, May 2013.

[14] Juhyeok Kim, Hai-Nguyen Nguyen, and Dongjun Lee. Preliminary control design on spherically-connected multiple-quadrotor manipulator system. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2015 12th International Conference on*, pages 206–207, Oct 2015.

[15] Jing-Jing Xiong and En-Hui Zheng. Position and attitude tracking control for a quadrotor {UAV}. {*ISA*} *Transactions*, 53(3):725 – 731, 2014.

[16] I. Sadeghzadeh, A. Mehta, A. Chamseddine, and Y. Zhang. Active fault tolerant control of a quadrotor uav based on gainscheduled pid control. In *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–4, April 2012.

[17] Yeunduk Jung and David Hyunchul Shim. Development and application of controller for transition flight of tail-sitter uav. *J. Intell. Robotics Syst.*, 65(1-4):137–152, January 2012.

[18] Nour Ben Ammar, Soufiene Bouallgue, and Joseph Haggge. Fuzzy gains-scheduling of an integral sliding mode controller for a quadrotor unmanned aerial vehicle. *International Journal of Advanced Computer Science and Applications*, 9(3), 2018.

Flight Coordination of MAVs in GPS-denied Environments using a Metric Visual SLAM

L. Oyuki Rojas-Perez *[1] and J. Martinez-Carranza[1,2]

[1]Instituto Nacional de Astrofísica, Óptica y Electrónica , Puebla, Mexico
[2]University of Bristol, Bristol, UK

ABSTRACT

Flight coordination of Micro Air Vehicles (MAVs) has become an essential task in the autonomous fight of swarms of drones. In recent years, we have seen demonstrations performed by giant companies such as Intel, where hundreds of drones perform a coordinated flight in the open sky. Although impressive, these demonstrations strongly depend on the use of GPS in order to effectively deploy MAVs in a coordinated fashion. In contrast, in this work, we present an approach for GPS-denied scenarios where localisation is resolved by using a well-known technique in robotics: visual simultaneous localisation and mapping. For this technique to be utilised, a single camera is mounted onboard the MAV, and even when a monocular camera is used to perform visual SLAM, if the camera angle w.r.t to the base of the drone and altitude are known, then the scale of the MAV's pose can be estimated. Rather than having a central controller, we have implemented a single individual controller for each drone involved in the coordinated flight. Thus, each drone knows its current drone's position as much as the position of its partner. This information is used in a PID controller with a consensus strategy to perform a coordinated flight defined by a set of waypoints. We showcase the effectiveness of our approach in an application where two drones have to carry an object from one location to another in a coordinated manner.

Figure 1: We present a methodology to perform an autonomous coordinate flight indoors using the same controller for both MAVs. We use the onboard monocular camera to metric localisation. A video of this work is found at `https://youtu.be/Tcox2MpRGrY`

1 INTRODUCTION

Currently, MAVs are employed for the acquisition data from image areas with the purpose of generating 3-D models of an environment to evaluate infrastructures or contribute to cartographic information. There are some works that focus on complex tasks, where one MAV is insufficient to complete an assignment such as exploration and inspection in vast territories, transportation of heavy loads or dangerous material. The latter calls for the use of more than one MAV, but also that such MAVs can cooperate during a mission flight.

The implementation of multiple MAVs requires the interaction between them to perform tasks in a coordinated manner. In this sense, MAV localisation is essential for cooperative tasks. For this reason, a research topic is that of estimating the drone's pose in 6D, which can be done by the use of GPS. However, GPS may not be accessible or reliable in some environments namely in urban canyons, forest environments or indoor scenes. [1], [2]. Motion capture systems are an alternative to GPS in terms of external localisation systems, however, this is not a general solution and therefore, there have been many efforts to develop systems to localise the drone via processing of sensor data acquired from onboard cameras, laser or similar sensors. The set of techniques relying on visual data for localisation are known as visual Odometry or visual Simultaneous Localisation and Mapping (SLAM), meaning that localisation is resolved while a map of the scene is also built.

Recently, visual SLAM has been used in the context of collaborative flight [3], where the authors present a cooperative system, a first drone navigates and maps the scene, while a second drone flies over the same place and recognises the

*Department of Computer Science at INAOE. Email addresses: {oyukirojas, carranza}@inaoep.mx

scene using the shared map by the first drone. In the other hand, the authors of [4] propose a method to fuse the IMU data and the monocular camera to build sub-maps for each MAV to get a robust communication between MAV's to perform efficient data exchange.

Motivated by the above, we present an autonomous system for multiple MAVs that uses a visual SLAM system to obtain the camera pose estimates with scale in centimetres for each MAV. The pose estimates are used by a PID controller with a consensus strategy to perform a coordinated flight defined by a set of waypoints.

The processing is carried out off-board since the MAV transmits the image and altimeter data in real time to a ground control station. Each MAV generates an individual metric map of the environment and shares its current position. For the flight, a predefined path in 3-dimensional coordinates is given, this path is followed by the coordinate MAV. We performed a series of experiments in indoor environments for the autonomous flight at 1.5 metres in height to transport a load in a straight line and for formation flight following a series of reference points, see the example in the Figure 1.

To present our proposed approach in detail, this paper has been organised as follows: section 2 describes the related work; section 3 describes our proposed methodology; 4 describes our experiments; finally, our conclusions are discussed in section 5.

2 Related Work

The problem of collaborative or coordinated flight has been studied for several years now. One of the most common techniques is based on a leading-follower architecture. Some works have proposed the use of geometric relationships, speed ratio, minimum turning radius or potential fields as main strategy to implement the coordination [5]. Some works [6] present simulation experiments of the control of multiple unmanned aerial vehicles using the relative position of the leader, where two controllers modify the behaviour of the vehicles, the first one controls the trajectory that the vehicles must follow and the second controls the height of flight [7]. However, the vehicles maintain the formation under certain conditions, for example, constant speed and trajectory angle not greater than 20°[8].

Flight training has also been implemented based on global positioning and telemetry [9]. In its control station, information is monitored individually by the telemetry of each vehicle, which allows changing the parameters of the system to keep them aligned with their neighbours. Vehicles fly in two predefined courses at separate altitudes, where MAVs wait for others to join. When all the MAVs are in the arena, they fly to the predefined area [10]. Regarding the control algorithms, the leading vehicle receives commands of speed and angles of orientation and trajectory, while the follower follows the manoeuvres of the leader maintaining a distance of separation to avoid collisions, where the system of coordinates are centred in the leading vehicle[11, 12]. In [13], the authors use a pilot to control the leading vehicle remotely, the control scheme consists of having to follow the leader under a separation distance. The primary condition of these works is that of maintaining the global position to remain in the formation.

On the other hand, the authors of [14], presage coordinated flight in interiors making complex trajectories. However, they depend on an external location system so that the vehicles stay aligned [15, 16, 1, 17, 18]. Besides, these works depend on the size of the arena to be able to carry out the training. In the works mentioned above the responsibility of the formation rests directly on the leading vehicle and this only receives information from the work station to update the trajectory, for the adjustment of parameters each follower vehicle receives updates individually.

In contrast to the works described before, in this work, an autonomous system is proposed to perform coordinated flight without dependency on the GPS. A strategy of consensus governs the proposed system. This enables the vehicles to make the flight without depending on a leader. Both vehicles receive the same parameters of speed and trajectory, as well as the same control system.

3 Methodology

The proposed autonomous system for multiple MAVs is based on two main components: (1)a metric monocular SLAM [19, 20] to obtain the camera pose estimates with scale in centimetres for each MAV;(2) and a Controller for Flight Coordination. Figure 2, shows the pipeline processing of our approach. The processing is carried out off-board since the MAV transmits the image and altitude data in real time to a Ground Control Station (GCS). Each MAV generates an individual metric map of the environment and shares its current position. For the flight, a predefined path in 3-dimensional coordinates is given, this path is followed by the MAVs.

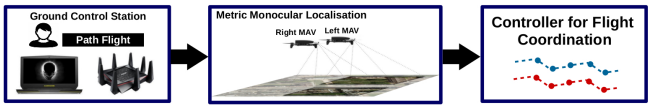


Figure 2: Schematic representation of our proposed approach. Two MAVs localise themselves by using our metric monocular system, where the camera looks down at the ground, while the camera images are processed to obtain the MAV's pose estimates.

3.1 Visual Metric Localisation

The Metric monocular SLAM is a modified version of RGB-D of ORB-SLAM. This method generates a synthetic depth image based on the line-plane intersection problem by formulating a geometric configuration where it is assumed that the ground is plane. The Bebop's altimeter is used to obtain an estimate of the camera's height h in centimetres and

the camera angle is obtained through the SDK is used to calculate the angle at which a vector n would be located with respect to the origin in the camera's coordinate system with length h. This vector n is perpendicular to the planar ground; therefore, it can be used to know a point lying on this planar ground with normal n. Figure3 illustrates a side view of this geometric configuration when the bebop's camera is foveated to the angle of -30 ° with respect to the horizon. The line-plane intersection equations are used to find α , depth corresponding to the pixel x, y.

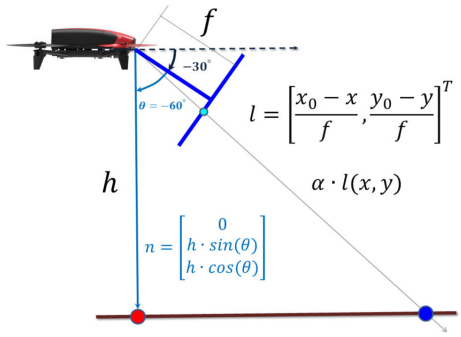


Figure 3: Geometric configuration used to generate a synthetic depth image to be used by ORB-SLAM in its RGB-D version, to obtain a pose estimation with metric. Image taken from [19].

3.2 Controller for Flight Coordination

The proposed controller uses rotation matrices to calculate the orientation of the current point with respect to the reference point, for which control commands are sent in yaw. The translation is calculated using the vector generated between the current point towards the reference point and to reach the desired point, and control commands are sent in pitch. To achieve a coordinated flight, the controller receives the position of the vehicle on the left and the one on the right, later a consensus strategy is implemented, which allows them to moderate their speed to keep them aligned during the flight. Once the vehicles arrive at the reference point, they are oriented towards the next one and later they are moved to the designated point. This will be calculated depending on the number of points of reference. The Robotic Operating System (ROS) communication system allows both vehicles to use the same control, under the same conditions, speed and path flight.

4 Experiments

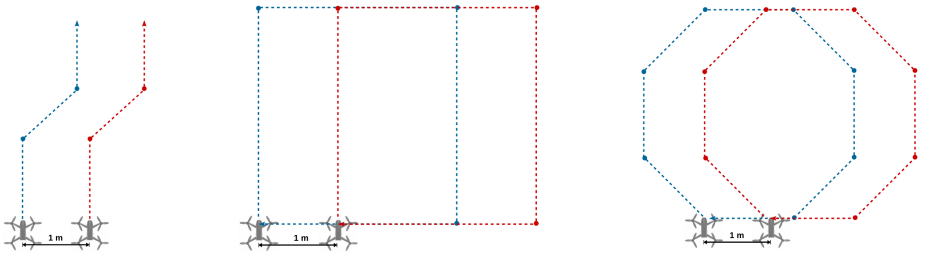
We present three different sets of experiments where we evaluated the performance of our proposed methodology. In these experiments, the vehicles take off and perform autonomous flight in an indoor environment by following a trajectory defined by a user. We evaluated the accuracy of the coordinate flight, for these, we used the motion capture system Vicon to obtain the measurements of the MAV's position.

For our experiments, we used two Parrot Bebop 2.0 Power Edition. We used the images captured with the onboard camera transmitted via WiFi with a resolution of 640 x 368 pixels at 30 Hz and the altimeter data transmitted at 5 Hz. Communication is possible using a router to communicate both vehicles to the GCS. For the programming of the controller, we used the Software Development Kit (SDK) known as the bebop autonomy SDK. This package run on a GCS: an Alienware-Dell Laptop with Intel Core-i7, with 16 GB in RAM. We used the ROS, Kinetic version, for implementation of our approach and communication with the other nodes; the Bebop driver, metric monocular SLAM and our controller. Figure 5, shows a scheme of our software architecture. Our approach uses the same flight controller for both MAVs. For the latter we exploited the capabilities of node reconfiguration offered by ROS through the use of the launcher files. In addition, ROS also facilitated the communication, transmission and consumption of all the data involved in our system, which led to carry out coordinated successful flights.

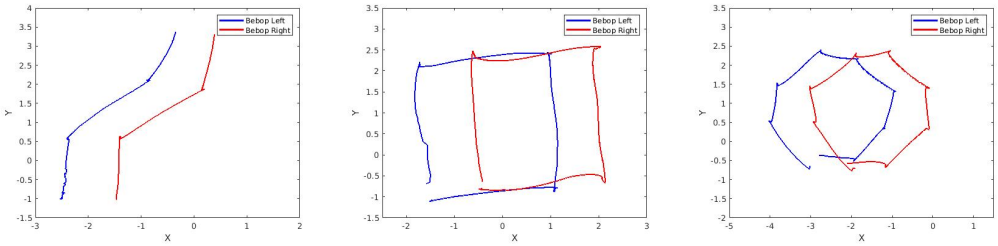
Figure 4(a), shows three plan flights: Line, Square and Octagon. The first trajectory is composed of three waypoints, the second trajectory is composed of four waypoints and the third trajectory has eight waypoints. The vehicles start 1 metre apart. After takeoff, each vehicle confirms if its partner is ready to start the flight. First, each vehicles changes heading in the direction of the next waypoint, once oriented, the PID controller calculates the error w.r.t. the next waypoint and sends control commands to pitch in order to fly towards this waypoint. For the consensus strategy, the controller receives information about the position of its partner and calculates the difference in the X coordinate (front axis). This difference value is added up to the pitch controller in order to regulate the speed of each drone in order to wait for each other or speed up, aiming at maintaining the same distance towards the waypoint. When the waypoint is reached, the controller turns the vehicle in the direction of the next waypoint. This will continue until each MAV reaches the last waypoint and then each one will land. Figure 4(b), shows the trajectories performed by the MAVs, the trajectories demonstrate that the vehicles follow the path symmetrically.

Ten runs of each trajectory were made. Table 1 shows that, on average, vehicles maintain the initial separation distance. The evaluation made of the trajectories with the VICON motion capture system was divided into two tables, table 2 corresponds to the vehicle on the left and table 3 corresponds to the vehicle on the right. In them, it can be seen that the error of the line path is high, due to the disturbances that are generated between them. However, in the square and octagon trajectories, it can be seen that the average error is less than 2%.

Figure 6, shows the trajectories generated with the coordinated flight controller, first shows the results of the vehicle on the left, followed by the results of the vehicle on the right. The first two columns show 5 line trajectories, column three



(a) Tracks designed to evaluate the autonomous coordinate flight.



(b) Trajectories performed by the MAVs evaluated using VICON.

Figure 4: Tracks predefined to perform coordinate flight, the blue line represent the left MAV trajectory and the red line correspond to the right MAV.

and four show 5 square trajectories and the last two columns show five octagon trajectories. The red line indicates the trajectory followed by the vehicle and the green line indicates the ground truth (VICON). Although there are disturbances generated between them, the vehicles are able to carry out the desired trajectory successfully.

Finally, the Figure 7 external views of the autonomous flight execution are displayed using the coordinated flight controller. The first row shows the performance of the line trajectory. The second row corresponds to the square trajectory; the third row shows the performance of the octagon trajectory. Finally, an example of cooperative flight is shown using the coordinated flight controller proposed in this work, Figure 8.

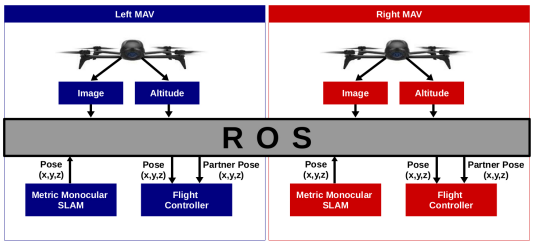


Figure 5: The architecture of the processing of our approach using ROS as a communication channel. Each PID controllers know the metric position of the partner to compensate the differences of the current pose to the reference point, this allows the flight are coordinate.

Table 1: Average Distance Between MAVs During Coordinated Flight.

Trajectory	Average distance [m]	Std [m]
Line	1.032	±**0.110**
Square	1.084	±**0.197**
Octagon	1.002	±**0.171**

Table 2: Left MAV

Trajectory	Average Error [m]	Std [m]	Average traversed distance [m]	Error in %
Line	0.120	±**0.067**	5.825	2.532
Square	0.159	±**0.089**	13.111	1.213
Octagon	0.218	±**0.110**	11.636	1.871

5 CONCLUSION

We have presented the implementation of a single individual controller for each MAV involved in a coordinated flight. So each MAV knows its current position as well as the position of its partner. This information is used in a PID controller with a consensus strategy, which commands each drone to follow a flight plan made of a set of waypoints. The results of the evaluation show that the average error of the estimated trajectories followed by each MAV is, in average, below 2% of the total trajectory, this in comparison to ground truth obtained with a motion capture system. In addition, we have presented an illustrative application where two MAVs are co-

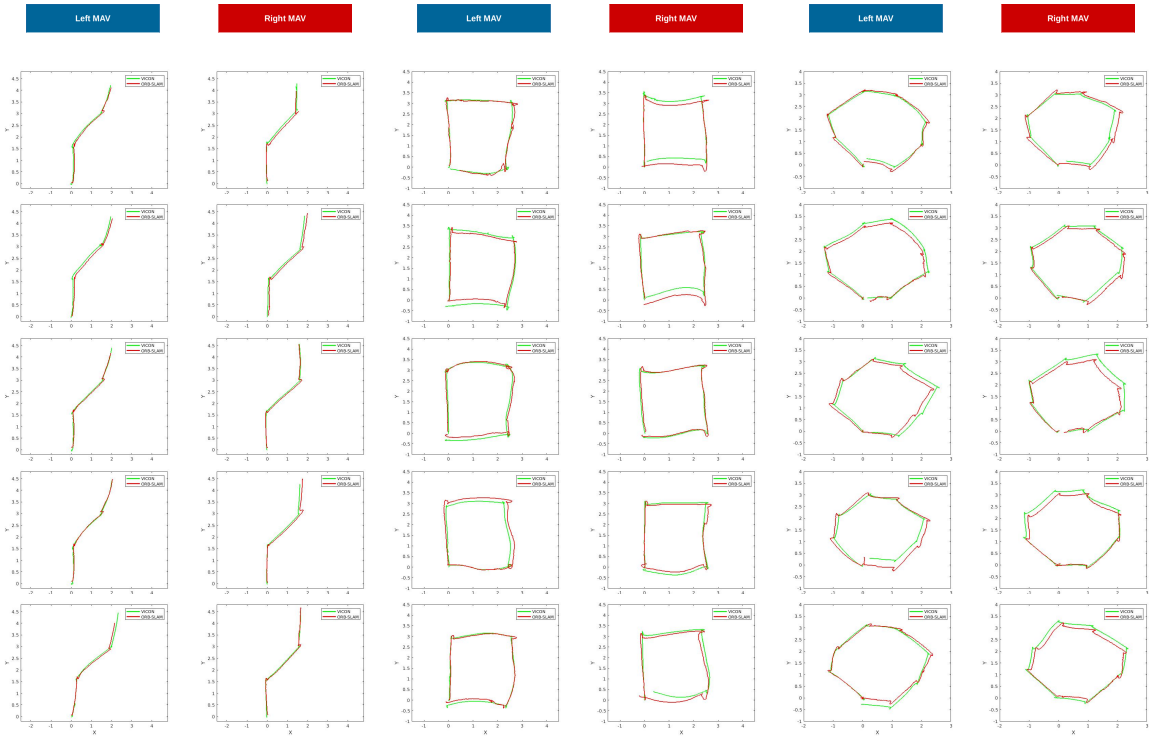


Figure 6: Examples of the trajectories generated. The green line represents the ground truth (VICON), and the red line is the path travelled by the vehicle. First, the graph of the vehicle placed on the left is shown, depending on the vehicle placed on the right.

Table 3: Right MAV

Trajectory	Average Error [m]	Std [m]	Average traversed distance [m]	Error in %
Line	0.155	$\pm$**0.080**	5.662	2.732
Square	0.179	$\pm$**0.107**	12.008	1.493
Octagon	0.162	$\pm$**0.078**	11.156	1.456

ordinated, using our approach, to carry out a collaborative flight to transport a load without risk of collision. We believe that the obtained results are promising and we will continue working on expanding the control for more than two MAVs, improving the communication system between them and the control algorithm.

REFERENCES

[1] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, pages 287–300, 2013.

[2] G. Vásá, Cs. Virágh, G. Somorjai, N. Tarcai, T. Szörényi, T. Nepusz, and T. Vicsek. Outdoor flocking and formation flight with autonomous aerial robots. In *IEEE/RSJ IROS*, pages 3866–3873. IEEE, 2014.

[3] P. Schmuck. Multi-uav collaborative monocular slam. In *Robotics and Automation ICRA*, pages 3863–3870. IEEE, 2017.

[4] N. Mahdoui, E. Natalizio, and V. Fremont. Multi-uavs network communication study for distributed visual simultaneous localization and mapping. In *ICNC*, pages 1–5. IEEE, 2016.

[5] T. Paul, T. Krogstad, and Jan T. Gravdahl. Modelling of uav formation flight using 3d potential field. *Simulation Modelling Practice and Theory*, pages 1453–1462, 2008.

[6] F. Giulietti, L. Pollini, and M. Innocenti. Autonomous formation flight. *IEEE-Control Systems Magazine*, pages 34–44, 2000.

[7] Q. Zhang and H. H. T. Liu. Robust nonlinear control of close formation flight, 2019.

[8] M. Pachter, J. D', Azzo, and Andrew W. Proud. Tight formation flight control. *Journal of Guidance, Control, and Dynamics*, pages 246–254, 2001.

[9] G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szörényi, T. Nepusz, and T. Vicsek. Outdoor flocking and formation flight with autonomous aerial robots. In *2014 IEEE/RSJ IROS*, pages 3866–3873, 2014.

(a) Line Trajectory

(b) Square Trajectory

(c) Octagon Trajectory

Figure 7: Examples of autonomous flight performed in a real-time on the indoor environment. The first row corresponds a line trajectory, the second row, correspond a square path and the third row, correspond an octagon trajectory.

Figure 8: Examples of autonomous collaborative flight performed in a real-time on the indoor environment.

[10] W. Meng, Z. He, R. Su, P. K. Yadav, R. Teo, and L. Xie. Decentralized multi-uav flight autonomy for moving convoys search and track. *IEEE Transactions on Control Systems Technology*, pages 1480–1487, 2017.

[11] D. Casbeer, S. Rasmussen, D. Kingston, D. Baker, and E. Garcia. Implementation of leader-wingman flight formation with sampled-data controller. In *AIAA Scitech Forum*, page 1453, 2019.

[12] C. Schumacher and S. Singh. Nonlinear control of multiple uavs in close-coupled formation flight. 2000.

[13] Y. Gu, B. Seanor, G. Campa, M. R. Napolitano, L. Rowe, S. Gururajan, and S. Wan. Design and flight testing evaluation of formation control laws. *IEEE Transactions on Control Systems Technology*, pages 1105–1112, 2006.

[14] M. Turpin, N. Michael, and V. Kumar. Trajectory design and control for aggressive formation flight with quadrotors. *Autonomous Robots*, pages 143–156, 2012.

[15] J. Welsby, C. Melhuish, and C. Lane. Autonomous minimalist following in three dimensions: a study with small-scale dirigibles. *Proceedings of Towards Intelligent Mobile Robots Manchster*, 2001.

[16] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin. Precision flight control for a multi-vehicle quadrotor helicopter testbed. *Control Engineering Practice*, pages 1023 – 1036, 2011. Workshop on Dependable Control of Discrete Systems.

[17] M. Turpin, N. Michael, and V. Kumar. Decentralized formation control with variable shapes for aerial robots. In *2012 IEEE Robotics and Automation*, pages 23–30, May 2012.

[18] T. Stirling, J. Roberts, Jean-C. Zufferey, and D. Floreano. Indoor navigation with a swarm of flying robots. *Robotics and Automation*, pages 4641–4647, 2012.

[19] H. Moon. Challenges and implemented technologies used in autonomous drone racing. *Intelligent Service Robotics*, 12(2):137–148, Apr 2019.

[20] L. O. Rojas-Perez and J. Martinez-Carranza. Metric monocular slam and colour segmentation for multiple obstacle avoidance in autonomous flight. In *RED-UAS*, pages 234–239, Oct 2017.

Autonomous Robotics Competition Club (ARCC)

Rachel Axten, Wen-Yu Chien, Jacob Crouse, Oliver Dunbabin, Venkatakrishnan Iyer, Kalki Sharma, and Vidullan Surendran*
Pennsylvania State University, University Park, Pennsylvania

ABSTRACT

In this paper, the technical approach to achieve the objectives for the indoor and outdoor missions of the International Micro Air Vehicle (IMAV) competition are presented. The ARCC team plans to utilize a single drone for the indoor competition and multiple drones to achieve the outdoor mission objectives. A scaled up version of the indoor drone would be utilized for the outdoor competitions to increase endurance and carry a larger sensor payload. A fully autonomous and primarily vision focused approach was taken for both missions with all computations being performed on-board excluding the 2D mapping which is performed off-board for the outdoor mission, whereas the indoor mission leverages off-board computing to reduce vehicle mass. Computer vision tasks are performed using a combination of the ZBar library for QR codes, semantic segmentation using the UNet architecture, object recognition using an RCNN, and classical image processing techniques such as ORB features and Hough transforms.

1 INTRODUCTION

The Autonomous Robotics Competition Club (ARCC) is a student run organization formed in the spring of 2018 and advised by Dr. Eric Johnson. The club consists of graduate and undergraduate students from the Pennsylvania State University. The club was founded with the purpose of participating in autonomous aerial vehicle competitions. The club consists primarily of students studying in the fields of acoustics, aerospace engineering, mechanical engineering, and electrical engineering but is open to anyone interested in such activities. A detailed description of the organisation and prior work can be found at the the club's website: https://sites.psu.edu/arcc/.

In this paper the previous work of Penn State's ARCC team, the technical approach for the indoor and outdoor competitions, and hardware and software specific elements are outlined.

2 PREVIOUS WORK

Thus far, the ARCC organisation has participated in the Vertical Flight Society's 2019 Mico-Aerial Vehicle Challenge (videos related to the competition are available on the club's website). The competition required the vehicle to pickup a package, navigate a course avoiding obstacles while flying over certain zones, and finally drop off the payload at a specified location while staying within the arena boundaries. The problems addressed were localisation, image recognition, and decision making. Initially an optical flow device paired with a Lidar was used to aid in localisation, but the mono-chromatic competition floor was featureless and thus we adopted a vision based approach using the Intel RealSense. A downward-facing camera was used to detect targets marked by April Tags. The images were transferred to a ground station via WiFi for processing. For obstacle detection a sonar sensor was used. The total weight of the vehicle was 500 grams. For package retrieval a pick-up mechanism was 3D printed doubling as the legs of the vehicle. To pick up the package the vehicle would hover above the package, lower itself, and close its servo actuated legs. The vehicle followed a pre-programmed flight path which was fine-tuned during run time through visual identification of landmarks reactionary behaviour to the world state. For example, if the vehicle was set to go straight but the sonar detected an obstacle, it would perform an obstacle avoidance maneuver prior to continuing on its path. An image of the vehicle is shown in Fig. 1.

Figure 1: Vehicle used for the Vertical Flight Society's competition with the optical flow (later replaced by RealSense).

3 HARDWARE

Since the competition is split into an indoor and outdoor portion separate drones have been built for the two missions.

This section outlines the hardware components chosen for each mission along with an estimate for the component and total weight.

Due to the increased image processing requirements of the outdoor competition, in addition to the Aaeon compute board, the outdoor drones will be outfitted with a much more capable Jetson Nano board, which is equipped with a GPU allowing us to speed up CNN inference. The specific components are outlined in Table 1. Since the outdoor vehicle is larger, the frame, power train, gripper, and battery are sized to reflect this. We chose to reuse the rest of the sensor payload from the indoor vehicle on the outdoor one. The indoor vehicle currently weighs about 470 grams whereas the outdoor vehicle weighs about 931 grams.

The open source px4 flight stack was chosen for this competition and thus the Pixhawk flight controller was utilized. The device has an on-board magnetometer and external GPS module, but these are disabled/removed on the indoor flight vehicle. The GPS sensor is removed to reduce weight and the magnetometer was disabled due to the potential of magnetic interference indoors. The flight controller is used to stabilise the vehicle whereas the Intel Realsense, Lidar, and camera make up the perceptual system.

4 Indoor Mission

The indoor mission requires the drone to navigate a warehouse environment autonomously. The mission profile involves recognition of key features (QR codes, flags, landing pad, etc), avoidance of obstacles (e.g. shelving), package retrieval and delivery. In order to accomplish these tasks autonomously several problems need to be addressed and solved. These are the problems of localisation, image recognition, control, path planning, communication, and package delivery. Prior to addressing the problems an overview of the hardware components and how they interact is presented in the next section.

4.1 Architectural Overview

The control architecture of the vehicle is comprised of three modules as shown in Fig. 2. The first module is the manual control. The manual control sends commands directly to the fight controller on the vehicle via a human controlled transmitter operating in the 2.4GHz band. This link allows us to take over manual control or disable the vehicle in case of emergencies. The second module is the vehicle. The vehicle is comprised of on-board computer and micro controller, 5GHz WiFi data link, flight controller, Intel realsense, and a camera used to identify mission elements using computer vision. The inertial sensors and motors are connected to the flight controller which forwards telemetry data to the on-board computer. The camera feed is sent to the on-board computer which transmits it over the data link to a ground station for processing. The on-board computer receives control commands from the ground station to be forwarded to the flight controller which executes the commands.

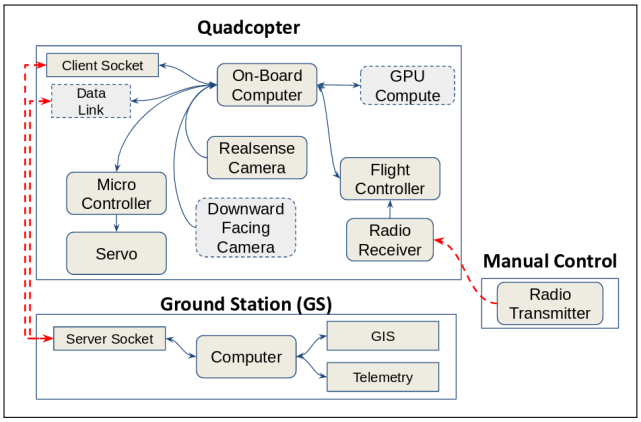


Figure 2: Overview of sub systems and how they are connected. Red links represent wireless communication. Grey dashed boxes are specific to the outdoor vehicle only.

In the event of a loss of connection with the ground station, the on-board computer is able to maintain static stability of the craft while it tries to reconnect to the ground station. The micro-controller is used to actuate the pickup mechanism and read/write to any sensor payloads that may be used.

4.2 Localisation

Localisation will be performed visually using the RealSense T265 camera. Since the camera does not provide a map, we will be post processing camera images to generate a map. This map could help us correct deviations in local position estimates if the ground truth locations of the static environment are known or estimated during flight using stereo vision/distance sensor.

In order to effectively command the Micro Aerial Vehicle (MAV), its current position in inertial space must be known within some degree of accuracy. Inside a warehouse environment, the drone is likely denied a GPS signal, and hence must look towards other sensors to provide the pose measurements required to estimate the vehicle's state. To this end, visual Simultaneous Localisation and Mapping (V-SLAM) is used, in the form of the RealSense T265, a camera-CPU suite capable of performing V-SLAM.

SLAM algorithms estimate vehicle pose by identifying features in a camera image plane, and tracking it frame-to-frame. The movement of the feature in the image between frames, coupled with knowledge of camera movement provided through accelerometer and gyroscope sensors, allow for the estimation of the location of that feature point in 3D inertial space with-respect-to the camera. Simultaneously, the accelerometers and gyroscopes can be used to estimate the camera's pose via some filtering technique such as a Kalman Filter in this case. This estimate is subject to accelerometer bias and gyroscopic drift. The feature point state estimates are then used to correct for these sensor drifts. The result is an accurate estimate of both the camera and feature point

Component	Quantity	Weight - Indoor (g)	Weight - Outdoor (g)	Description
Frame	1	80	150	Composite
ESC	4	10	10	4 in 1
Motor	4	16	16	Emax motors
Flight Controller	1	10	70	Outdoor incl. GPS
Onboard Computer	1	60	N/A	Aaeon Up
GPU Compute	1	N/A	130	Jetson Nano
Intel RealSense	1	60	N/A	T265
Lidar	1	12	N/A	Range sensor
Camera	1	N/A	15	Downward facing
Propeller	4	5	8	APC
Payload Gripper	1	40	30	3D printed
Battery	1	120	300	LiPo
Miscellaneous	N/A	50	100	Wires, connectors,...
Total		470g Approx.	931g Approx.	

Table 1: Vehicle components and weight. N/A represents components not applicable to that version of the vehicle.

state. A LIDAR distance sensor is used for accurate altitude measurement and as is common for UAV applications, an Extended Kalman Filter is used to combine the measurements from these sensors, with vehicle state equations, to yield an estimate of the vehicle state (position and pose).

4.3 Image Recognition

QR code recognition will be done using the ZBar library. This library requires a cropped image containing the QR code in order to recognise it. We trained a Regional Convolutional Neural Network (R-CNN) using the inception v3 backbone on synthetically generated images containing various QR codes superimposed on a wide variety of indoor backgrounds. It is hard to quantify accuracy as our synthetic data set is not standardised, but we obtained an accuracy of approximately 92% on the held out test set. The bounding box generated was then cropped out and recognised using ZBar. A similar strategy was employed to detect the boxes and the flag.

To circumvent the issue of being unable to predict the effectiveness of the RCNN in the actual competition environment and since we have ample computing power due to offboard processing, we validate our results by performing semantic segmentation using the UNET architecture [2]. The goal of a segmentation problem is to section the pixels of an image into the classes that the network has been trained on. The data for these classes would be obtained from an image search on the internet thereby augmenting our dataset with labelled data that represents the classes we are interested in such as the shelves, mailboxes and other objects of interest. The segmented pixels would then be extracted and then cross-validated against the CNN classifier to confirm that the correct class has been detected. A weighted average of the the R-CNN and UNET prediction would be considered as the final result.

For the boxes on the shelves we had to train a regional network as the architecture we used for semantic segmentation does not segment multiple objects of the same class, only providing a blob that contains all the objects. Since there are many boxes on the shelf, we are training a faster-RCNN [3] to draw bounding rectangles around them. The boxes are non-descript and of various shapes with the identifying factor that they contain QR codes. The training dataset for this particular case was generated from images of cardboard boxes that were super imposed with a QR code and warped, rotated, and skewed using classical image processing techniques.

In the case of the flag, during take off we hope to detect the flag pole and store the image of the flag as a ground truth image. When we encounter the image during the drop-off phase, we would compare it to the ground truth image using ORB [4] features. A backup plan is to use transfer learning to quickly train a CNN in the 2 training days preceding the competition when we are made aware of the flags that would be used as these would only be a subset of all the flags in the world.

Finally, to detect the landing zone which has an 'H' on it, we will reuse code from our previous competitions where we use probabilistic Hough transforms to identify the 3 lines which are oriented in a known geometric shape.

4.4 Controls

Control of the MAV must be robust enough to handle the coupled dynamics of the MAV and payload package slung load; to this end, control is provided by the PX4 firmware running atop the Pixracer flight controller. As the lion's share of vehicle control is being provided by the PX4 firmware, only a brief treatment of the controller software will be presented. The PX4 firmware estimates the state of the vehicle (from accelerometer, gyroscope, LIDAR, V-SLAM, and GPS sensors) via an Extended Kalman Filter (EKF). It then compares the

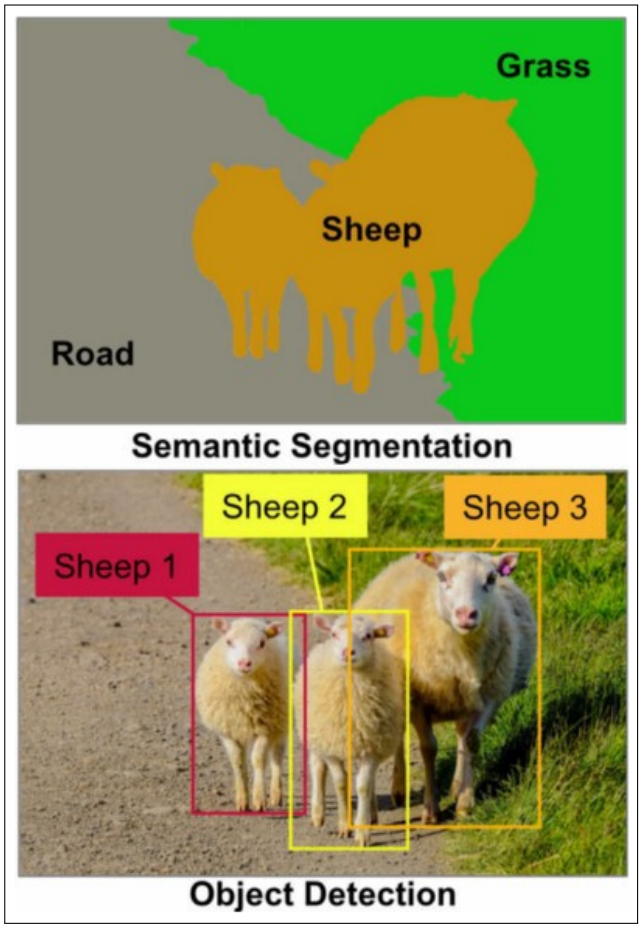


Figure 3: Illustrates the difference between image segmentation and object recognition, the two approaches adopted in our algorithms. [1]

vehicles current state estimate (position, velocity, pose, and pose rates) to that which is commanded by trajectory planning. Error between current and desired state, prompts the controller to send signals to the four motors, eliciting collective, lateral and longitudinal cyclic (governing pitch, roll, and yaw of the vehicle - ϕ, θ, ψ - and body angular rates p, q, r) response of the vehicle.

The magnitude to which this state error influences the motor inputs is regulated via a simple proportional-integral-derivative (PID) controller.

$$U(s) = (K_p + \frac{1}{s}K_i + K_d s)E(s) \quad (1)$$

Here, $E(s)$ is the tracking error, in the Laplace domain, K_p gain minimises tracking error, where the higher the gain value the faster the response; K_d gain dampens the response of the vehicle, reducing overshoot; K_i gain aids in reducing steady-state error.

PID controllers are used to control the vehicle's rate, attitude, velocity, and position command. Despite the dynamics of the system changing after the package pickup, the PX4 controllers are assumed robust enough (with large enough margins), as to observe minimal controller performance degradation. These gains are tuned for our particular flight needs, and will likely be slightly different for indoor and outdoor missions.

4.5 Path Planning

A feature point in the camera image plane is defined as a point which may correspond to an obstacle in the environment. The process of tracking a feature point involves identifying a point of interest within the camera image frame and then tracking those points between frames (correlating a measured feature point with its counterpoint in a database of tracked features). The first step is to identify which objects within a frame are of interest. Two methods being utilised aboard our vehicle are classical image processing techniques (Harris Corner and Canny Edge detection algorithms)[5] as well as more contemporary/deep-learning techniques (training CNNs to identify objects likely to be encountered by the drone). Once these features are found within the image plane, these measurements need to be corresponded to those features seen in prior frames (stored in a database) in such a way that the same features are tracked between frames. To this end, the statistical Z-test is employed; a method which, when presented a new image of measured features/objects, can match measurements with objects stored in the database, with highest probability[6]. It is necessary to store these observed features in a database (and subsequently estimate their state - i.e. position), so that we can track their relative location, even in their absence from camera frames. To this end, the vehicle's state (i.e. pose), and the feature point's movement within the image plane, is used to estimate the object's location[7]. This is done using an EKF, whereby the vehicle's state equations, the feature's state equations, and the entire suite of vehicle sensors are fused to provide an accurate estimate of its location relative to the vehicle[8]. These estimates can then be used to avoid (and in some cases track towards - c.f. using image recognition to identify points of interest) objects within field of view of vehicle.

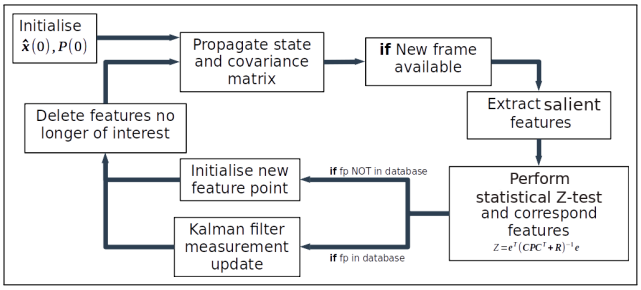


Figure 4: The feature point (fp) tracking EKF process. N.B: this flowchart is invariant of the image processing method.

4.6 Payload Delivery

For the indoor vehicle, an alternative, weight-saving approach for the pickup mechanism will be considered since the package weighs 25 grams for vehicles weighing under 500 grams. The proposed mechanism will be a small hook like device which can be extended to secure the loop on the package. The hook will be mounted on a pivot such that it minimises visual interference during flight, and allows the vehicle to land on its 'belly' eliminating the need for landing gear. A common 3.5 gram micro servo actuates the hook the device and is capable of moving 60 degrees in 0.05 seconds. The servo arm is approximately 15cm and exploits the structural strength of the 3D printed plastic in tension to lift the package. This minimises the load on the servo compared to a gripper design which uses servo torque to hold the package. This allowed us to reduce weight and power consumption.

4.7 Communication

This section outlines how the vehicle and ground station communicate with one another. Two types of data formats are sent and received by both the vehicle and the ground station. The data formats are Mavlink messages and video. The vehicle transmits telemetry data to the ground station, and the ground station logs the information along with displaying the current state for purposes of debugging and testing. The ground station sends commands to the vehicle via user input during testing. Both the telemetry and commands are sent as Mavlink messages. The other data format is that of video. Video is transmitted from the vehicle to the ground station, where the ground station displays the feed and uses it to feed the object detectors.

The communication protocol used for Mavlink data transmission is TCP/IP and for video is UDP/IP. Bo protocols utilize WiFi at 5GHz for data transmission, but the TCP and UDP have differing structures. The benefit of TCP is that it performs rigorous checking to ensure the data transmitted is received and in the format which it was sent. While UDP streams data are connection less meaning there is no acknowledgement that a package has been successfully transmitted. Therefore TCP is better suited for sending telemetry and commands where the transmission of a corrupted message or missing a message all together can prove fatal, and UDP is adequate for video transmission where missing a frame is not detrimental to the vehicle. When setting up socket communication a server and client must be specified. In this case the ground station serves as the server which waits for a client, the vehicle, to connect.

Apart from the vehicle and the ground station, a transmitter is also able to communicate with the vehicle in order to take over manual control. The transmitter operates at a frequency of 2.4GHz. In order to minimize interference and increase available bandwidth, Wi-Fi communication will utilize the 5GHz band.

5 OUTDOOR MISSION

For the outdoor competition, we plan to use a GPS module to obtain the current position of the drone in addition to fusion of vision data. As time is a crucial factor, three drones would be utilized to simultaneously deliver the three packages. As the location of the houses and post boxes are not provided, a random grid search would be performed to locate these targets. Similar to the indoor approach, we will leverage R-CNNs to identify the targets, namely the houses, post boxes, the crashed drones, and the missing parcels. The system will be equipped with a Raspberry Pi camera for use with mapping and target detection. From pixel coordinates, aircraft attitude and altitude, along with the GPS information, the pinhole camera model would be used to estimate the location of the detected objects in the world coordinate system.

Finally, the task of generating a 2D map of the area would be done using a Geographic Information System library, in our case GeoPython as our off-board computational stack is coded using python 3. Features in the images would be identified and tagged with global positioning information to stitch them together to generate a 2D map of the area. The process of feature tracking and determining the latitude and longitude in the images is automated such that the on-board computer tags each image taken with the current GPS location at capture so that the stitching of the images can be completed in the 20 minutes duration provided at the end of the mission. Further, vision would be used to detect the balloons in the takeoff and landing area but since a quad-rotor affords vertical take off and landing, with careful flight planning we do not anticipate an encounter with the obstacles.

In contrast to the indoor mission, the outdoor vehicle will communicate with the ground station using a 2.4GHz wireless broadcast link with a 900MHz control link for emergency manual control. These frequencies provide adequate range and bandwidth for video and control signals respectively. This would allow for the transmitting of camera data from the vehicle to the ground station for 2D map generation. The vehicles would still be autonomous with all processing done off-board, so loss of link would only restrict its ability to map the environment and send telemetry data.

The decision to perform all computation on-board the vehicle unlike the indoor competition is driven by the risk of connection loss on the 2.4GHz band at the allowed 25mW power limit at maximum operational range. As the vehicle all up weight is not as critical in the outdoor mission, we chose to include a more powerful compute board on-board for robustness and safety reasons.

5.1 Architectural Overview

The proposed approach to the outdoor competition is to use two identical 500 gram quadrotors similar to the indoor competition mainly for package drop off with a larger (approximately 1Kg) vehicle that identifies drop off locations, and maps the environment. This is to optimize the weights of

the vehicles to maximize the weight multipliers while creating a vehicle which will be powerful enough to lift the avionics as well as a post box package with a reasonable endurance.

The on-board avionics system for the two smaller vehicles will be the same as the indoor vehicle with the addition of a GPS module. The larger vehicle will use the Pixhawk 2.1 flight controller which supports autonomous waypoint tracking, as well as compatibility with all other avionics peripherals. To localize the system in the global coordinate frame, the team will use the Here GNSS GPS system, which is compatible with the Pixhawk and has been found to be accurate up to 3.7 meters of the desired destination [9]. Additionally for communication, the team will utilize 915MHz transmitters with a common ground station. All antennas will be circular polarized (2-5 dBi gain) to ensure uniform converge regardless of azimuth angle.

The vehicles have been designed to optimize the scoring system of the competition. The weight has been selected as an ideal point for a high weight scale factor, as well as allowing sufficient vehicle weight to support the proposed avionics set. Additionally, the system is optimized for target detection over mapping because the scoring system of the outdoor competition heavily favors detecting the points of interest over creating a high quality map.

5.2 Localization

The plan to localize each identical system is using the Here GNSS GPS system which is compatible with the Pixhawk 2.1 flight controller.

5.3 Object Detection

A Regional Convolutional Neural Network (R-CNN) is trained with synthetic databases generated to identify the house, mailboxes, lost packages and crashed drones similar to the indoor mission. As the mailboxes and lost packages are of unique colors, color identification schemes can also be used to identify these objects. However, owing to the ground color being a possible mixture of yellow and green, identification of the yellow mailbox using color detection schemes may yield false positives. Further, as the same network architecture can be trained to identify different objects using different databases, it is more convenient to use the same deep learning approach for detection of all objects. Transfer learning approach will be used to train the RCNN at a lower computational cost.

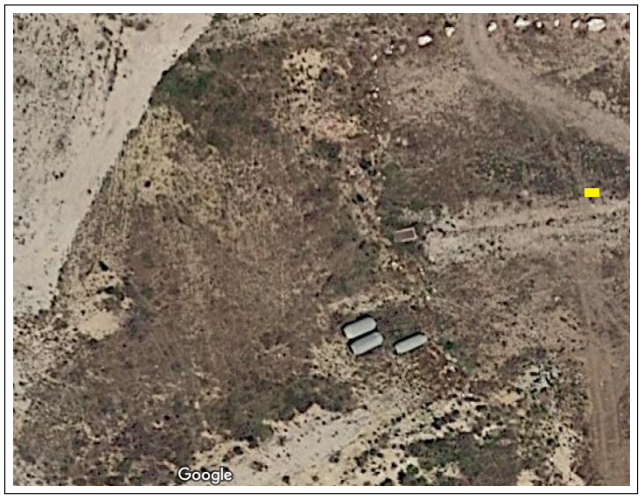


Figure 5: Example of an image for training the neural network. The superimposed mailbox can be seen in yellow.

Characteristic	Value
f	1.14 mm
X_0	0.507 mm
Y_0	0.395 mm
k_1	-0.013
k_2	0.1764
k_3	-0.6391
p_1	-0.0032
p_2	-0.0072

Table 2: Raspberry Pi camera parameters.

The CNN is fed images from a Raspberry Pi camera mounted on-board for inference. Since the multirotors will be flying at a height of 25 $[m]$, it is possible to determine the ground dimensions of the image since the internal parameters of the camera is known. The internal parameters of the camera is not available from the manufacturer and hence the same needs to be determined through camera calibration. A similar approach is adopted by Piras et al [10] where the camera parameters have been determined using camera calibration. The camera internal parameter as determined in the cited paper have been used. The parameters are listed in table 2.

In order to train the neural network, a database of images is generated. These images are generated by superimposing a colored box over an image of the outdoor competition's terrain in a random location. This colored box is scaled to be of the correct size relative to the predetermined height of flight of 25 $[m]$. An example of one of the training images can be seen in Figure 5. From this example we observe that the size of the objects are much smaller than the overall image which prompted us to use an R-CNN over other architectures such as YOLO [11] as emperical tests have confirmed that it has superior performance for smaller objects with careful tuning of anchor size and aspect ratio hyper parameters.

5.4 2D Mapping

Two approaches are proposed to generate the 2D map of the area. In the first approach, multiple images with sufficient overlap as explained in the Path Planning section are obtained. Image stitching is then performed based on keypoints generated and using the RANSAC algorithm. This approach is currently implemented in the ground station. Figure

Figure 6: 2D mapping using image stitching.

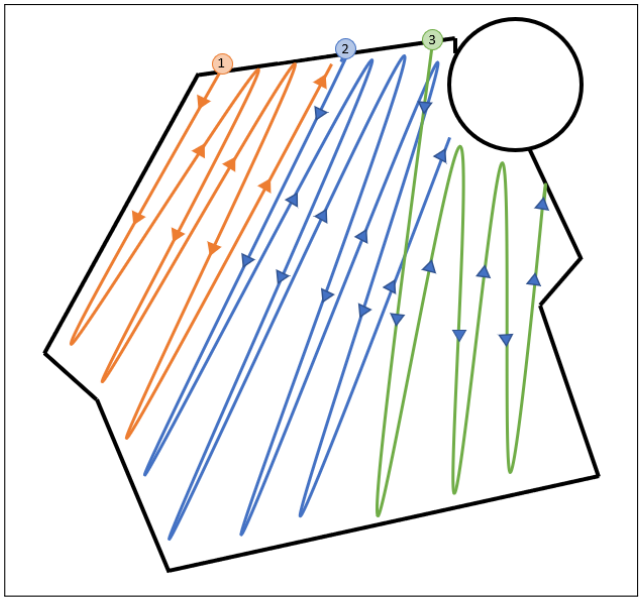


Figure 7: Predetermined path of each outdoor quadrotor.

6 illustrates the same based on multiple overlapping images obtained of an area from an elevation.

While the image stitching approach might yield acceptable results, there exists a possibility of artifacts in the stitched image owing to 3D objects. Thus, the alternative approach proposed is to generate orthomosaics. The orthomosaics are then geo-referenced using tools such as Open Drone Map, etc. This will generate the required 2D map of the operational area of the outdoor mission.

5.5 *Path Planning*

The path design for the outdoor competition will take advantage of all three quadrotors at the same time. The two smaller vehicles will transmit their image data to the ground station for off-board identification of targets whereas the larger drone will perform this on-board. The common ground station ensures all vehicles will be kept in sync regarding recognised targets. Using this approach, the area will be mapped in the shortest time possible.

The default path for the vehicles will imitate the ”mowing of a lawn” by starting at the starting location and sweeping back and forth within the competition area, moving in one direction until the boundary of the area is reached, and then shifting some distance horizontally while rotating 180° to move back toward the starting area. This behavior can be seen in the Figure 7 In case of a failure in Wi-Fi link at range, the two smaller drones will be unable to perform this task. In this scenario, the larger drone will be commanded to complete this behaviour slowing down the search, but still retaining the capability of using the smaller drones to deliver packages using GPS as this is not dependent on a high throughput Wi-Fi link.

In order to ensure that the entire area is mapped, sufficient overlap between images need to exist so that the image stitching can be performed correctly. Based on the geometry as detailed in Kraus [12], the percentage overlap in the forward direction is set to 60% and 30% in the sides. This will ensure that there are sufficient DoG and Harris keypoints available to ensure effective image stitching. The homography matrix for the matched vectors is determined using the RANSAC algorithm. With the desired percentage overlap, images in the forward direction need to be captured at every 40 m and the distance between two parallel paths should not be greater than 30 m.

5.6 *Payload Delivery*

Time is a crucial factor in scoring of the outdoor phase of the competition. Each of the three multirotors will be loaded with a single package. Once one of the multirotors locates a post box, this information will be sent to the ground station and relayed to the multirotor with the corresponding package.

After the multirotor is aware of the location of the correct postbox, it will use the onboard Pixhawk 2.1 flight controller along with the Global Positioning System (GPS) location of the post box to navigate to the post box and deliver the package. Given the competition’s requirement of landing the package within a 5 meter radius of the post box and the accuracy of the Here GNSS GPS module being reported to be within 3.7 meters we did not employ any additional vision correction. As the competition does not require the package to be picked up by the drone autonomously, the payload mechanism used is a mainly a package drop mechanism as shown in figure 8

Figure 8: Package Drop Mechanism.

As can be seen in figure 8, the drop mechanism is in the retracted position. This ensures that the drop mechanism does not obstruct landing of the drone. The mechanism will be extended after takeoff and the package can be attached to the hook when the vehicle is hovering. The package is dropped at a desired location by retracting the mechanism.

5.7 Communication

The method of communication is similar to the indoor portion with the major difference being the transmission frequencies. Due to the range required of the outdoor vehicle, the WiFi operates at 2.4GHz as opposed to 5GHz. The manual control transmitter is operated at 915MHz to ensure it does not interfere with the WiFi communication.

6 CONCLUSION

The details presented in the paper is a brief insight into the technical approach that the team has envisioned to successfully complete the IMAV 2019 challenge. With many modules already implemented and ready, the team is on track to fly, test and validate all modules well in advance of the actual competition. With the feature tracking methodology, the warehouse mission can be successfully completed considering the fact that the real sense module provides localization with tolerances of within 3cms observed during testing. Finally, the deep learning framework proposed will enable real time target detection and identification owing to CNN inference being computationally efficient compared to classical vision processing.

ACKNOWLEDGEMENTS

The team is thankful to the Department of Aerospace Engineering, Pennsylvania State University for providing necessary support and infrastructure. The team would also like to thank the Pennsylvania State University for the resources in terms of funding. Finally, the team is indebted to Dr. Eric Johnson who, as the academic advisor of the club, has been a constant source of technical guidance and support.

REFERENCES

[1] Ravindra Parmar. Detection and Segmentation through ConvNets,*https://towardsdatascience.com/detection-and-segmentation-through-convnets-47aa42de27ea*, Sep 2018.

[2] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. UNet++: A Nested U-Net Architecture for Medical Image Segmentation. *CoRR*, abs/1807.10165, 2018.

[3] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR*, abs/1506.01497, 2015.

[4] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An Efficient Alternative to SIFT or SURF. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

[5] Daniel Magree, John Mooney, and Eric Johnson. Monocular Visual Mapping for Obstacle Avoidance on UAVs. *Journal of Intelligent Robotic Systems*, 74(1-2):17–26, 04 2014.

[6] Girish Chowdhary, Eric N. Johnson, Daniel Magree, Allen Wu, and Andy Shein. GPS-denied Indoor and Outdoor Monocular Vision Aided Navigation and Control of Unmanned Aircraft. *Journal of Field Robotics*, 30(3):415–438, 2013.

[7] J. Civera, A. J. Davison, and J. Montiel. Inverse Depth Parametrization for Monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, 2008. Date revised - 2009-07-01; Last updated - 2011-11-14; SubjectsTermNotLitGenreText - Parametrization; Inverse; Standards; Representations; Localization; Uncertainty; Mapping; Position (location); Robotics.

[8] Autonomous Flight in GPS-Denied Environments Using Monocular Vision and Inertial Sensors. *Journal of Aerospace Information Systems*, 10(4):172–186, 2013.

[9] Titan Elite. Pixhawk compatible gnss receiver comparison,*https://files.zubax.com/products/com.zubax.gnss/gnss-receiver-performance-report-april-20184.pdf*, 2018.

[10] Macro Piras, Nives Grasso, and Ansar Abdul Jabbar. UAV Photogrammetric Solution Using A Raspberry Pi Camera Module and Smart Devices: Test and Results.

[11] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[12] K. Kraus, I. Harley, and S. Kyle. *Photogrammetry: Geometry from Images and Laser Scans*. Number v. 1 in De Gruyter textbook. Bod Third Party Titles, 2007.

Warehouse Management Using Real-Time QR-Code and Text Detection

Debjoy Saha*, Ganesh Shiridi Balaji Udayagiri†, Parakh Agarwal, Biswajit Ghosh, Somesh Kumar‡
IIT Kharagpur, West Bengal, India

ABSTRACT

The objective of this project is developing the computer vision tools for efficient inventory management of packages of a warehouse. Packages are identified by unique QR-codes(Quick Response codes) and are required to be matched with the alphanumeric codes of the shelves on which they are kept. Dimensions of the shelves are pre-known. Figure 1 shows the setup available to us at the competition. QR codes are pasted on the body of the packages and alphanumeric codes are put on the shelves. QR-code identification relies on OpenCV(Open-source computer vision) library ZBar which provides quite reliable and robust output. Corresponding alphanumeric code identification is done using deep learning text detection library Tesseract. All modules were integrated into a ROS(Robot Operating System) architecture and the output preserved in a CSV file. In addition to it, we developed an algorithm for robust and precise warehouse management. The novelty of our approach lies in the detection of text in Tesseract computationally inexpensively using pre-known information. In general, this paper can be used as a working guide for text detection using Tesseract under similar conditions. All libraries used are explained in detail.

1 INTRODUCTION

Real-time text detection from visual input is a useful aspect for the development of autonomous robots. Most of the information that we receive from our surroundings is in the form of images. And a major part of those images is text. Text identification and decoding is thus essential for the development of fully autonomous drones. This project concerns with text identification part only. It also concerns with the decoding of another kind of visual input, QR codes which are comparatively in more machine recognizable format, given the standardized representation of these codes. As already explained, the problem which we are concerned with involves a warehouse containing boxes on shelves. We need to make an inventory, listing QR-codes pasted on the objects and matching it with the alphanumeric codes pasted on their respective shelves. The last part of this paper deals with the algorithm for warehouse management. The warehouse management is done using a quadcopter autonomously with an attached gimbal for image stabilization, implementation similar to [1]. We produce all the steps that we took to improve the performance of our algorithm, the test inputs and their results. A basic flow chart of the developed algorithm is shown in Figure 2.

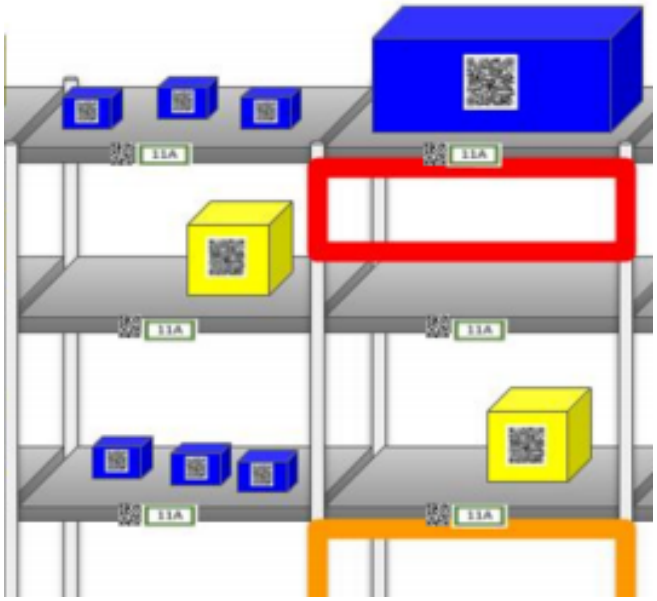


Figure 1: Sample shelf (Source: www.imav2019.org)

2 PRIMARY PRE-PROCESSING FOR QR-CODE DETECTION

Images obtained from the video feed is from a fish-eye camera. A fish-eye camera is used owing to the wide-angle that it captures. The images obtained from this camera are distorted. To utilize the data and properly decode QR-codes and text in the image we have to undistort the images. We use camera_calibration package of ROS [2] and OpenCV [3] to calculate the camera matrix, distortion coefficients and other parameters using multiple checkerboard images at different positions, scale and skew angles.
Having calculated all the intrinsic and extrinsic components,

*Email address(es): sahadebjoy10@gmail.com
†Email address(es): balajiatvizag@gmail.com
‡Supervising professor

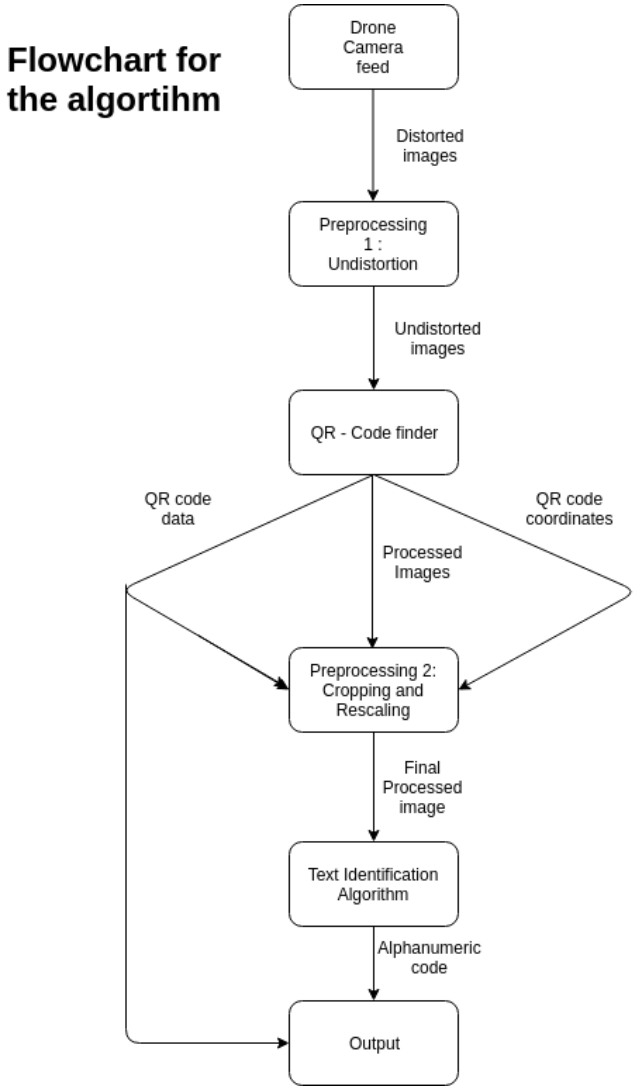


Figure 2: Flowchart for the algorithm

(a) Input

(b) After undistortion

Figure 3: Image before and after undistortion

we undistort the images and publish them for further usage. An example of the undistortion process is given in Figure 3. Some more pre-processing steps are used on the undistorted images using information extracted from QR codes, which are explained in section 4. For now, we skip the remaining pre-processing steps, since they are computationally expensive and not necessary for QR code detection. So, unless QR codes are observed from camera feed, we do not perform those steps.

3 QR-CODE DETECTION

We use the OpenCV ZBar library [4] to detect QR-codes. We pass the images obtained from the undistorted image feed(as referred to in section 2), to the ZBar class object created [5]. It scans the two-dimensional image to produce a stream of intensity samples. The decoder searches a stream of intensity values obtained by processing the images from the video feed for recognizable patterns and produces a stream of completely decoded symbol data from which the QR code text are decrypted. We store the obtained QR-code data and its coordinates for further usage and move on to Alphanumeric code detection.

4 ALPHANUMERIC CODE DETECTION

The most challenging hurdle to the task was detecting the alphanumeric codes accurately. For this task we used Google's open-source library Tesseract[6]. Tesseract is an OCR(Optical character recognition) engine with support for Unicode and the ability to recognize more than 100 languages. It can be trained to recognize other languages. Tesseract is efficient only in the case of extremely well defined text in a page (like we obtain for a .pdf (portable document format) file). It shows quite poor performance for detection of text from noisy images, such as those obtained from a drone camera. Parameters need to be properly tuned and images properly pre-processed to produce optimal output. To obtain this, we follow a step-by-step approach. First, using information extracted from the QR-code (center and dimensions), we attempt to further process the input image and create more identifiable images. Then we apply Tesseract text

recognition on it. We list the various parameters that we modify to obtain perfect solution. Finally, we propose a method to further improve the result obtained using Tesseract.

4.1 Secondary Image pre-processing for alpha-numeric code detection

After the initial pre-processing step (section 2), further processing needs to be done on input feed for better detection of alpha-numeric characters. The methods mentioned below helped improve performance.

- Rescaling: Tesseract gives useful results with specific text size in the image. To improve the result obtained by Tesseract, we need to resize the image appropriately.
- Cropping desired area: Cropping reduces the ROI(Region of interest), thus improves the text detection by increasing confidence in the prediction.
- Blurring: The noise in the image gets reduced and results thus get better.
- Thresholding: It can be used to binarize the image to improve the performance and eliminate shadows. [7].
- Rotation / Deskewing: The image could be rotated to align the text which helps gain better results. [8].

Out of the listed methods, only rescaling and cropping improved output quality, so we developed algorithms that used those methods appropriately.

4.1.1 Cropping the image

For the text recognition to work well we observed that the results are better when the domain on which search for the text has to be done is smaller. The sides of the image is cropped to reduce the search area. We intend to crop our image with respect to the QR-code. We do not crop the image from top or bottom as this may lead to loss of information of the text and QR-code. Our cropping assumes the following:-
1. Both the QR-code and the alpha-numeric code occur in the same frame.
2. The QR-code is positioned along the left or right edge of the shelf and the alphanumeric text is positioned at the center of the shelf.
3. Only one shelf was detected per image.

Assumptions 2 and 3 were taken as limiting cases of the alphanumeric detection problem. Our objective is to ensure that the qr code and text both appear in the final cropped image irrespective of their relative separation. Since we know the alphanumeric text is placed at the center of the shelf, the position of maximum separation is if the QR-code is near the edge. Also, the maximum separation in pixels can only be obtained if we consider one shelf per image.
Consider the length of the label of the alpha-numeric code be a and the length of the shelf be L. Position of the center of the QR-code label be x. And the length of the QR-code is l_q. The part of the image in between the position x. (refer to Figure:4)

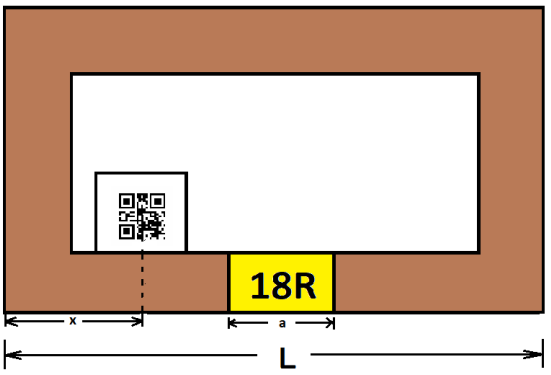


Figure 4: Showing the parameter of the shelf

Distance between farther ends of QR-code and alphanumeric code -

$$d = ((L + a)/2) - x \tag{1}$$

When $x = 0$ (The extreme case),

$$d\_max = (L + a)/2 \tag{2}$$

This implies that in a d_max neighbourhood of the QR-code, the alpha-numeric code exists. Now, since the image is undistorted, we can directly compare the ratio of distances to obtain the part of the image we need to retain. The ratio we obtain is

$$k = (L + a)/(2 * L) \tag{3}$$

In our current problem, we find this ratio to be 0.56 which we approximately 60%. So we crop of 2/5th of the distance of both ends of the image, measured from the center of the QR-code. So, in the final cropped image, we retain 60 % of the frame on either side of the QR-code. Using this we can be sure that the alphanumeric text is detected and at the same time reduce the region of interest considerably.

4.1.2 Re-scaling the image

The image obtained after cropping does not always yield good results. This reason is attributed to the size of the the text in the image, which has to be close to a particular value for best results. For proper detection, therefore, we need to estimate the text size before-hand to scale the image accordingly. There are many machine learning frameworks for detection of text regions but usually, they are computationally very expensive. We observed that the text recognition occurs optimally at a particular text size in the image (174 X 74 pixels using the test drone). This text size in the image occurs at a particular distance of the camera (here the drone) and the

object that contains text (here the shelf). When the camera is at some other distance, the text size in the image is different from the optimal size and the output is not desirable. We propose a method for text size approximation in the image that resizes the image using the information from the camera matrix [9] (obtained through camera calibration) [10, 11] and the distance of the shelf from the drone. The drone is maintained at a constant distance from the shelves using a depth estimate from a stereo camera mounted on parrot bebop drone. We make use of known text frame size (on the shelf) and the depth at which it occurs. Our objective in this method is to find the scaling factor by which we would scale our image.

Camera matrix =

$$\begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix}$$

where fx and fy are focal-length times a scaling factor.

$$fx = f * mx \tag{4}$$

$$fy = f * my \tag{5}$$

(where mx and my are the scaling factors along corresponding axes). In the camera we used, the scaling factors along both axes are the same hence we take the focal length f as the average of fx and fy.

$$y' = y * f/z \tag{6}$$

$$y'' = y' * m \tag{7}$$

where,
y' = object_size_in_image_sensor
y = object real size
f = focal length of camera
z = distance from object containing text
y'' = object size in pixel
m = pixels per millimeter

Final step is re-scaling by a factor such that the final output image used for detection has optimal text pixel size. Scaling by:

$$k = a/y'' \tag{8}$$

a = optimal text size in image(in pixels) = 74 px
k = scaling factor

To re-scale, we use OpenCV's [3] built-in re-scaling function and the obtained ratio k. The representation of the algorithm is provided in Figure 5.

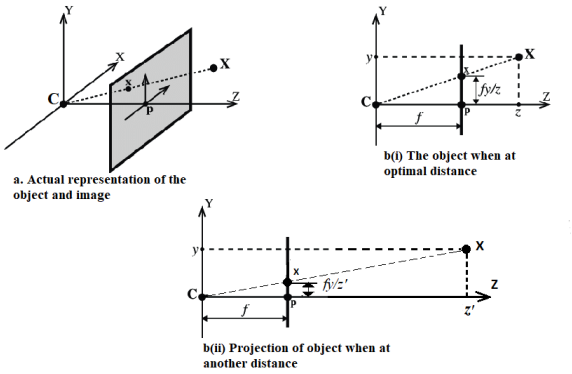


Figure 5: Re-scaling : The image (a) shows the actual representation of the object in 3-dimensions. The image b(i) shows the projection of object on the image. And the character recognition at this distance of the object is optimal. (ii) shows the projection of object on image at another distance that is not the optimal distance for character recognition. Our objective is to process this image so that it gives optimal output for character recognition, i.e. achieve situation(i).

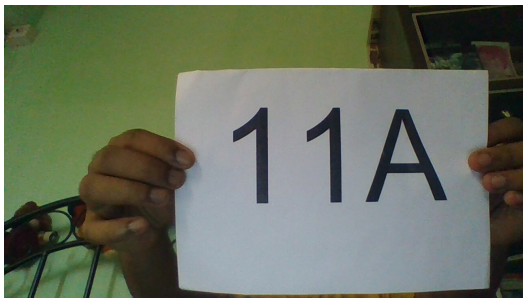


Figure 6: Sample testing image

4.2 Improving Tesseract performance

Furthermore, Tesseract contains many intrinsic parameters whose values can modify output results as well as its frequency. In the following section, we list those parameters and their observations one by one. All provided results are sampled from a larger output of the algorithms, as tested on sample Figure 6, therefore they are only indicative of the actual results and the effects observed on modifying those parameters. The final results are provided in section 6.

4.2.1 Setting appropriate Page segmentation method

Page segmentation mode(psm) is the first of those parameters. By default, Tesseract expects a page of text when it segments an image. How Tesseract will read the page will depend upon how it segments the image. And we can specify that using psm. Tesseract provides 13 supported page segmentation modes for different scenarios. PSM_SINGLE_WORD

(i.e Treating the image as a single word) was considered due to better performance at detection over other methods, PSM_SINGLE_LINE (i.e treating the image as a single text line) and PSM_DEFAULT.

S.no.	PSM_SINGLE_WORD	PSM_AUTO
1	11A	
2	11A	11
3	11A	114
4	11A	41h
5	11A	11A

Table 1: Results for different page segmentation modes

Increased frequency and accuracy of prediction, given predefined knowledge as to the type of text present (Refer to Table 1).

4.2.2 Disabling Tesseract built-in dictionaries and including custom word-lists and patterns

Modifying the load_system and load_freq parameters allows us to enable/disable Tesseract dictionaries, which are responsible for producing output close to the general dictionary words. Disabling the dictionaries, Tesseract should increase recognition since the text we need to detect isn't dictionary words but alphanumeric codes having a specific pattern.
Common character patterns were added (/d/d/c for digit, digit, character) to further trim down image output to more accurate results. It should be noted that this method only increases the probability of correct predictions. We need to perform a further trim(section 4.2.5) to ensure it. The following table summarizes the improvements.

S.no	Using Custom word-patterns	Default
1	11A	141A
2	J1A	J1A
3	11A	111A
4	11A	A1A
5	11A	1A

Table 2: Custom word pattern results

Notice the improvement in detection of both digits instead of just one (row 5) and of correct text-pattern (row 4) (refer to Table 2).

4.2.3 Using White-list of characters

White list is another parameter which can be assigned the value of a string containing all characters which we want to be used for recognition purposes. Since we are sure to encounter only single alphanumeric codes for detection, we can safely omit special characters and blank spaces from the list of characters. Thus we can add alphabets and digits under Tesseract white-list, only they will be used for prediction of text.

S.no.	With White-list	Without white-list
1	11A	
2	11A	11
3	114	114
4	11A	11&
5	11A	!1A

Table 3: Results for character white-list

Using white-list has suppressed occurrence of special character '!' (row 5) and '&'(row 4) and has improved confidence in prediction (Refer to Table 3).

4.2.4 Using an iterator object to examine sub-strings

Owing to background noise, a lot of stray text is also detected, which is not a part of the text in the input image. To suppress those occurrences, we use an iterator object to scan each string detected separately and select the text sorted on basis of confidence in prediction and word-pattern. We modify the code to involve an iterator object to achieve this result.

4.2.5 Trimming the output further

The next step is parsing the output for the desired result and selecting the most probable alphanumeric code corresponding to a specific QR-code. We run a loop over the obtained text to find for data of type (int)(int)(character) having the maximum number of observations.

Algorithm 1 explains the full algorithm.

Algorithm 1 Overall Algorithm per frame

Ensure: QR code & Text Matching
Input: Camera feed image, Depth
Undistorted_image←Preprocess(INPUT)
(QR_dims, QR_data) ← QR_detection(Undistorted_image)
Cropped_image←Cropping(Undistorted_image, QR_dims)
Resized_image ← Rescaling(Cropped_image, depth)
Text ← Text_detection(Resized_image)
Alphanumeric_Code ← Trim(Text, QR_data)
Output: (QR_data, Alphanumeric_Code)

5 WAREHOUSE MANAGEMENT ALGORITHM

Using the above pre-processing steps guarantees a good recognition using very minimal computation but during use, to reduce any possible errors, we improvised on the motion

of the quadcopter to improve efficiency and quality of results. The improvisations are listed below:

- The current warehouse consists of various shelves, and each of the shelf is used to store various packages. The shelves are further divided into various rows and columns to store the packages. We observed that detection when traversing along columns was considerably slower than when traversing along rows. So a row-wise traversal of the shelves was used finally.
- Also, since Tesseract gives optimal result at only a definite text size, we implement an algorithm, in addition to section 4.1.2, to maneuver forward and backward from the mean position. This to and fro swaying motion perpendicular to the shelf changes the text size by a small amount thus correcting any drift errors in the position of the drone and thus assuring an appropriate output even at an erroneous distance from the shelves. It also reduces the scope of error in re-scaling of the image. An oscillation of 10 cm was observed to produce maximum results.
- Having observed that Tesseract gives better results when the text is on the bottom side of the page, we align it more that way to get a better result using the pre-known position of the QR code from section 3.
- Once a QR-code is detected, we initiate a hover time so that sufficient detection results are obtained thus minimizing error during final processing.
- Also, the presence of gimbal ensures that the QR-code is viewed normally at all times, reducing perspective distortion in text.

6 TESTING

Figure 7: Drone used for testing (Source: www.parrot.com & www.amazon.in)

We used a Parrot Bebop 2 drone (Figure 7) for testing purposes which has a built-in gimbal camera, to eliminate all hardware limitations. The result obtained was quite satisfactory. QR code and corresponding alphanumeric text could be correctly recognized from a wide range of distances from the shelf, depending upon the quality of the camera feed of the drone. Experiments on Bebop drone gave impressive results for distances up-to 3m, after which the camera feed deprecates due to re-sizing (section 4.1.2). The Average computation time observed was low, thus the algorithm was capable of real-time detection. Output frequency and accuracy was sufficiently high, thus the hover time required before each package was low (Results listed in detail in Table 4).

Parameter	Freq(min^{-1})	Acc(%)	Frame-Rate(fps)
None	95	96.8	7.10
Config-File	68	95.4	5.96
White-List	97	42.2	7.31
PSM	11	51	7.21

Table 4: Result: Excluding different parameters

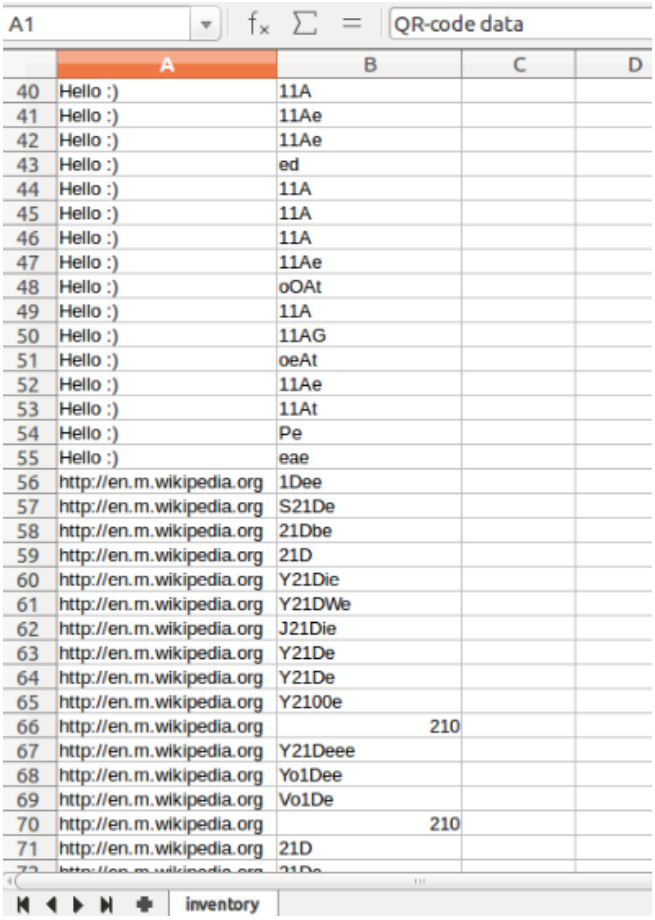
Table 4 lists in detail the effect of the absence of any of the tuned parameters tested in section 4.2. The first column is the parameter excluded. The second column represents the number of correct observation observed per minute. The third column is the percentage of correct observations in prediction. The fourth lists the frame-rate observed during detection. These results are obtained during testing on a desktop PC.

Major takeaways from the results table are -

- An impressive 95 observations/minute were observed on the final algorithm and an accuracy of 96.8 % without the final trimming step (section 4.2.5).
- Removing config-files (specifying custom user-patterns and dictionaries, section 4.2.2) decreases the frame rate as well as the observation frequency. However, on the limited observations, the accuracy observed is quite high (95.4%).
- Removing white-list (section 4.2.3) or changing page-segmentation-mode (psm) (section 4.2.1) has quite drastic results on detection results.

7 CONCLUSION

This article documents all steps and corresponding results of the warehouse management sub-module of the IMAV competition 2019. In this article we have mentioned the ways to optimize the two major modules of warehouse management i.e QR-code detection and alpha-numeric detection.The QR-code detection module is simpler and provides accurate results even with no modifications. However, the text detection module has higher complexity and has to be dealt with



A1 | QR-code data

	A	B	C	D
40	Hello :)	11A		
41	Hello :)	11Ae		
42	Hello :)	11Ae		
43	Hello :)	ed		
44	Hello :)	11A		
45	Hello :)	11A		
46	Hello :)	11A		
47	Hello :)	11Ae		
48	Hello :)	oOAt		
49	Hello :)	11A		
50	Hello :)	11AG		
51	Hello :)	oeAt		
52	Hello :)	11Ae		
53	Hello :)	11At		
54	Hello :)	Pe		
55	Hello :)	eae		
56	http://en.m.wikipedia.org	1Dee		
57	http://en.m.wikipedia.org	S21De		
58	http://en.m.wikipedia.org	21Dbe		
59	http://en.m.wikipedia.org	21D		
60	http://en.m.wikipedia.org	Y21Die		
61	http://en.m.wikipedia.org	Y21DWe		
62	http://en.m.wikipedia.org	J21Die		
63	http://en.m.wikipedia.org	Y21De		
64	http://en.m.wikipedia.org	Y21De		
65	http://en.m.wikipedia.org	Y2100e		
66	http://en.m.wikipedia.org	210		
67	http://en.m.wikipedia.org	Y21Deee		
68	http://en.m.wikipedia.org	Yo1Dee		
69	http://en.m.wikipedia.org	Vo1De		
70	http://en.m.wikipedia.org	210		
71	http://en.m.wikipedia.org	21D		

inventory

Figure 8: Initial output

great precision. We have described various methods of using Optical character recognition. To improve the results, we used undistortion, re-scaling of the input image, cropping of the input image. We obtained a frequency of 6-7 results per second and the correct alphanumeric code was embedded in about 90-95% of identified strings. Refer to Figure 8: Initial Output, without the final trim(steps up to section 4.2.3) and Figure 9: Final Output. The results were quite satisfactory. Given a hover time of 4-5 seconds (modifiable during final testing of the algorithm at the venue), we obtain enough test data to predict the correct QR-code with >90% accuracy on test input. We look forward to reproducing this performance at the competition too.

Acknowledgements

We would like to thank our team members and supervisors at ARK (Aerial Robotics Kharagpur) to provide us with valuable insights on this project and all necessary equipments. We would also like to acknowledge Parrot for the Bebop-2 drone that we have used for testing. We would also like to thank the authorities at Indian Institute of Technology, Kharagpur to provide us the funds and support for the presented work.

	A	B	C
1	QR-code data	AlphaNumeric Code	Shelf Code
2	Hello :)	11A	
3	http://en.m.wikipedia.org	21D	
4			
5			
6			
7			

Figure 9: Final output

References

[1] A. Anand, S. Agrawal, S. Agrawal, A. Chandra, and K. Deshmukh, "Grid-based localization stack for inspection drones towards automation of large scale warehouse systems," 2019.

[2] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

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

[4] Wikipedia contributors, "Zbar — Wikipedia, the free encyclopedia," 2019. [Online; accessed 5-June-2019].

[5] I. Szentandrási, A. Herout, and M. Dubská, "Fast detection and recognition of qr codes in high-resolution images," in *Proceedings of the 28th spring conference on computer graphics*, pp. 129–136, ACM, 2013.

[6] R. Smith, "An overview of the tesseract ocr engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, pp. 629–633, IEEE, 2007.

[7] Z.-K. Huang and K.-W. Chau, "A new image thresholding method based on gaussian mixture model," *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 899–907, 2008.

[8] Y. Zhang, J. Zhong, H. Yu, and L. Kong, "Research on deskew algorithm of scanned image," in *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 397–402, Aug 2018.

[9] O. Semeniuta, "Analysis of camera calibration with respect to measurement accuracy," *Procedia CIRP*, vol. 41, pp. 765–770, 12 2016.

[10] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, 2000.

[11] J. Weng, P. Cohen, and M. Herniou, "Camera calibration with distortion models and accuracy evaluation," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 10, pp. 965–980, 1992.

Heterogeneous Multiple Vehicles Cooperation Approach for Smart Roads

Abdulla Al-Kaff [1]*, Ángel Madridano[1], Ahmed Radwan[2], Francisco Miguel Moreno[1], Ahmed Hussein[3] and Arturo de la Escalera[1]
[1]Intelligent Systems Lab (LSI), Universidad Carlos III de Madrid, Madrid, Spain
[2]Multi-Robot Systems Lab (MRS), German University in Cairo, Cairo, Egypt
[3]IAV GmbH, Berlin, Germany

ABSTRACT

Intelligent vehicles are equipped with multiple on-board sensors for environment perception. Moreover, with the increasing number of these vehicles on the roads, the more cooperation and coordination among them is becoming more crucial. Accordingly, this paper presents multiple heterogeneous vehicles cooperation approaches, to be used in smart roads to improve driving safety. The heterogeneity aspect is based on the use of Unmanned Aerial Vehicle (UAV) to scout the surrounding of the Unmanned Ground Vehicle (UGV), thus increasing the perception efficiency. Two approaches were proposed for this cooperation, vehicle and pedestrian detection. The algorithms are implemented in the on-board computers. In order to evaluate the proposed approaches, different scenarios were selected and multiple experiments were carried out. The obtained results show the high performance of the algorithm in almost real-time detection and classification, moreover the ability to communicate the outcomes to the UGV, thus improving the automated navigation process for out of the line of sight pedestrians.

1 INTRODUCTION

The advances in Intelligent Transportation Systems (ITS) are exponentially improving over the last century. The objective is to provide intelligent and innovative services for the different modes of transportation, towards better, safer, coordinated and smarter transport networks. The ITS focus is divided into two main categories; improve existing components of the transport networks, and develop intelligent vehicles which facilitate the transportation process [1]. In recent years, interest in self-driving vehicles has significantly increased. Accordingly, the necessity of cooperation with all road entities becomes more crucial. The ITS consists of three main entities: vehicles, infrastructure and pedestrians [2].

Accordingly, an intelligent vehicle on the road must cooperate with all road entities, to ensure road safety, especially the safety of pedestrians and other Vulnerable Road Users (VRU). Therefore, the VRU recognition and avoidance in intelligent vehicles are essential tasks. However, due to sensor limitations and several blind spots surrounding the vehicles, researchers are studying different possibilities improving the perception to detect out of the line of sight obstacles.

This paper presents a heterogeneous cooperative approach to tackling the problem of obstacle detection and avoidance with intelligent vehicles. In particular, an Unmanned Aerial Vehicle (UAV) is used to help an autonomous vehicle detect pedestrians located in blind or low visibility areas for the car. To do this, it embarks on the UAV, a monocular camera and a computer, which can process visual information and determine both the position of the vehicle and that of pedestrians. The information generated by the vision algorithms is shared with the vehicle to be incorporated into its perception of the environment through intervehicular communication. In this way, a method is presented that allows providing the terrestrial system with safer navigation.

The remainder of this paper is organized as follows; Section 2 presents the background overview of previous carried-out work in this field. Afterwards, Section 3 introduces the proposed algorithms for detecting and tracking pedestrians and vehicles. In Section 4, the experimental work is illustrated with the selected platforms and scenarios, followed by the discussion of the obtained results in Section 5. Finally, in Section 6 conclusion and future work are summarized.

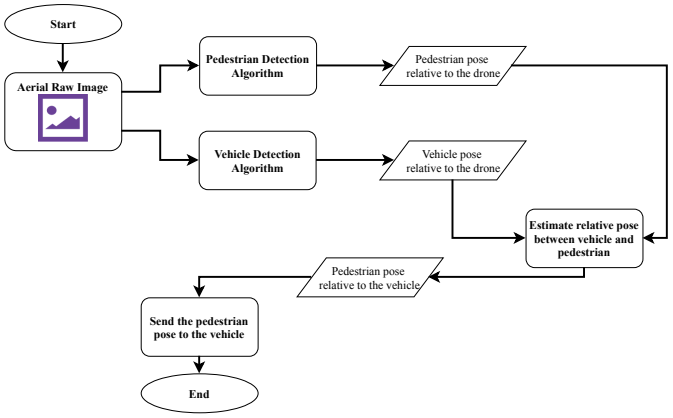


Figure 1: Proposed Approach

*Email address(es): akaff@ing.uc3m.es

2 BACKGROUND OVERVIEW

Pedestrian detection using computer vision is considered as a challenging problem in traffic environments, and most of the solutions presented in this field are based on a common approach, which uses a Histogram of Oriented Gradients (HOG) descriptor, and a Support Vector Machine (SVM) classifier [3]. In [4], a monocular camera is used to detect pedestrians from a UAV by applying a HOG descriptor. Thereafter, based on three image sequences, the distance to the pedestrian is estimated. Other works, such as [5], prefer a Haar-Like based algorithm for pedestrian detection, followed by a template-based tracking.

Furthermore, a detecting and tracking feature-based method, from UAVs was presented in [6]. First, the features are extracted using Harris detector, then the pyramidal Lucas-Kanade (LK) optical flow model and Least Median Square Estimator (LMedS) are used; to classify the movement of the detected features. Then, a Kalman filter and a template matching algorithm are used to track the targets.

Lately, with the advances in deep learning, new methods for object detection and classification are used. For instance, in [7], a deep Convolutional Neural Network (CNN) is trained to classify moving vehicles, showing promising results. In addition, the heterogeneous cooperation between ground and aerial vehicles has been explored in applications such as search and rescue. Recently, a heterogeneous robot collaboration of UGV-UAV has been presented in [8]; in order to collect observations in cluttered urban environments. In this approach, the robot team is able to map the environment while following predefined waypoints. First, the UGV builds the 3D map of the environment using a LIDAR, then, the UAV performs the data gathering. Moreover, the UAV estimates its location by detecting and tracking the UGV.

Furthermore, authors in [9] introduced a method for pedestrians detection and localization based on perception for cooperation between a team of UAV and UGV. The geographic information systems localization system considered that the UGV as a moving landmark for a perspective transformation; to convert the image locations of the targets.

3 PROPOSED APPROACH

In this section, the proposed approach is divided into two algorithms: vehicle and pedestrian detection, as it is shown in Figure 1. These algorithms are explained below.

3.1 Vehicle Detection

The main objective at this point is to be able to detect, by computer vision and in real time, a characteristic pattern located on the roof of an autonomous vehicle, and knowing its position with respect to the UAV while it flies over an area, in which is located the Vehicle and a set of pedestrians.

The procedure of detecting and estimating the position of the autonomous UGV consists of analyzing each frame captured by the camera equipped in the UAV. Once the pattern is located, the algorithm estimates the position of the UGV with respect to the UAV. Once the position of the UGV and the pedestrians with respect to the UAV is estimated, the relative position of pedestrians to the UGV is estimated.

In this work, the UGV is equipped with a pattern, placed on the roof, as it is shown in Figure 2. The detection of this pattern will make it possible to know the position of the vehicle with respect to the UAV, and to be able to know the relative position of the vehicle with respect to the pedestrians located in the vehicle environment.

In the case of the circumference, it is defined by Equation 1 and is described by three parameters: coordinates x and y to the center of the circumference (a, b) and the radius (r).

$$(x-a)^2 + (y-b)^2 = r^2 \tag{1}$$

In this algorithm, Hough is used to detect the circumference of the pattern. Since the flight altitude of the UAV is variable, it can be found in different sizes depending on the height, at which it is found, so no search size will be set for the radius. The only parameter established will be the distance between centres of the different circumferences to be found within the image, so that in each image only one circle is detected, given that in the proposed scenarios there will only be one UGV.

As it is shown in Figure 2, the pattern is formed by a circle of $790mm$ diameter, and X-shape inside.

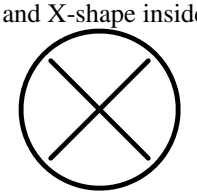


Figure 2: Landing Pattern

The procedure used to try to detect and know the position of the autonomous vehicle consists of analyzing each frame captured by the camera and, through computer vision algorithms, detecting the described pattern on the roof of the vehicle. Once located, it will be possible to know the position of the land vehicle with respect to the UAV.

Once the position of the autonomous vehicle with respect to the UAV and the position of pedestrians with respect to the air vehicle is known, the relative position of persons with respect to the autonomous vehicle may also be known.

The algorithm is based on the localization of circles within the image and, in the subsequent analysis of the regions of interest (ROIs) generated from the circles detected in the frame. The algorithm will be taking frame to frame by trying to carry out positive locations of the model in each of the captured images, and performing a tracking process in case there is no detection in the consecutive frame to a positive detection. The steps within this algorithm for the detection process are detailed below:

Circle detection: it is carried out using the transformation of Hough.

Creating a region of interest: to create a new image from the original frame, extracting the section of the frame in which the circle has been detected, provided that the pattern is completely inside the image. Next, a filtering by size is carried out to ensure the successful completion of the remaining steps of the algorithms. The location of the small-sized pattern would negatively affect all other tasks.

Resetting pattern size: an image of the model to be searched is created that matches the size of the circle being analyzed at this time.

Match between model and detection: Once the pattern is reset, a correlation is established between the model and the ROI created around the detected circle. If the correlation value is above a threshold, which has been set experimentally, it is considered that the detected circle corresponds with the pattern placed on the vehicle, so that it is proceeded to accept that detection (Figure 3) and to calculate the position with respect to the UAV.

If the correlation value is below the set value, the algorithm starts the tracking process as long as the pattern has been detected in the previous frame.

Before moving to a new detection or to carry out the tracking process it is verified that the detection is not incorrect by finding the model rotated with respect to the original, thus repeating this step by making turns of 30^o in the reset pattern. This process is repeated as much 2 times before leaving this step, because when you reach the third iteration the pattern will have rotated 90^o and therefore reach its original position.

Figure 3: Detection of the Landing-Pad

Position calculation: Finally, the calculation of the position is performed. For this, as it is collected in the following equations, it is necessary to know the size of the pattern, both real and in the image, as well as the focal length of the camera. In addition, the value of the x and Y coordinates of the image, as well as the value of the x and Y coordinates of the center of the detected circle, must be used, all in pixels

$$z[m] = \frac{RealPatternSize * FocalLength}{ImagePatternSize * 1000} \quad (2)$$

$$x[m] = \frac{|CenterImgX - CenterDetecX| * z}{Focal\_LengthX * 1000} \quad (3)$$

$$y[m] = \frac{|CenterImgY - CenterDetecY| * z}{Focal\_LengthY * 1000} \quad (4)$$

As indicated above, in the case that there is no detection after a correct location, a tracking process of the pattern is started using the position of the pattern in the previous image. The tracking is carried out using the OpenCV library and follows the steps as follow:

Tracker initialization: to carry out the creation and initialization of the tracker. To do this, it is necessary to pass to the function an image and an area or region of the element to be followed during the tracking process, so this operation is performed by passing to the function the previous frame and the region of interest in which the pattern has been correctly detected. Each time the detection algorithm concludes with a positive result, the MedianFlow tracker is initialized.

Tracker update: to update the tracker; in order to carry out the detection of the model in the current frame, in which the detection has not achieved a satisfactory result. In this way, the tracking updates the location of the pattern from the last known position of the pattern in the new captured image.

Checking the tracking: It establishes a new region of interest obtained from the frame being analyzed and the position of the pattern obtained by the tracker. The pattern is readjusted to the size of the region of interest and again a matching process is performed where the correlation between the ROI obtained from the tracking process and the model being sought is checked. If the correlation is above the established threshold, the tracking is correct and the position of the last detected pattern is updated, whereas if the correlation result is below the threshold, the tracker is considered to have failed and the pattern is lost, so that the algorithm will loop back the detection process until a good location (Figure 4) is obtained.

From the results obtained from the tracking algorithm, it has been decided to set a threshold value in the tracking algorithm lower than the threshold value in the detection algorithm. Which resulting the increament the number of frames in which the location of the vehicle is known.

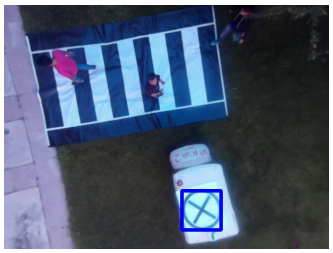


Figure 4: Tracking of the Landing-Pad

Position calculation: If the tracking is correct, the position is calculated using the Equations 2, 3, and 4 previously set.

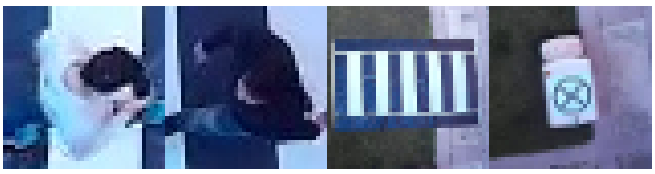
3.2 Pedestrian Detection

The main objective at this point is the detection of pedestrian, by HOG in real time, and to locate their positions with respect to the UAV.

HOG descriptor uses a global feature to represent an object rather than a collection of local features. An entire object is represented by a single feature vector, as opposed to many individual vectors representing smaller parts of the object. Typically, HOG descriptor converts an RGB image of size $(width \times height \times 3)$ to a single feature vector n.

Pedestrian detection is done by a camera housed on a UAV. The procedure is to process each frame from the camera stream to detect pedestrians within the image. Detection steps within this algorithm for the detection process are as follows:

1. ***Training HOG descriptor:*** the descriptor training is performed using linear SVM. The training set is a balanced one; number of positive set equal number of negative set. The positive set consists of two hundred 40x40 pixels images of the object of interest (Figure 5a and 5b). The negative set consists of two hundred 40x40 images from the background of the object of interest (Figure 5c and 5d).



(a) (b) (c) (d)

Figure 5: Positive and Negative Training Set

2. ***Setup HOG descriptor:*** as stated earlier in subsection (3.2), the HOG descriptor detection depends on several parameters that were tuned through trial and error. WinStid size is set to 8x8 pixel step between each sliding window location. Padding of size 8x8 was used in the detection. The scale parameter was set to 1.01, this value provided a sufficient factor by which the image is resized at each layer and number of levels in the image pyramid. The hit-threshold proved to be the most important parameter in HOG detection. In the results section, the performance of HOG detection under three different hit-threshold values will be discussed.

3. ***Detections Filtering:*** the detection suffered from noise at the edges of the frame to be processed. A small number of false positives appeared at the edges of the processed frame. These false detection are filtered and omitted from the detection. Furthermore, the detection was filtered against size. Detections with size greater than 60x60 pixels are not counted.

4. ***Position Calculation:*** the positions of detected pedestrians is calculated using Equations 2, 3, and 4. When the positions of both pedestrians and the vehicle are calculated relative the UAV, the relative position between pedestrians and vehicle can be calculated.

3.3 Inter-Vehicular Communication

In this work, an approach for inter-vehicular communication for the broad off-road environment is proposed. The approach objective is to maintain a continuous connection among the vehicles in the system. Accordingly, a Virtual Private Network (VPN) is created, which requires secure connection via the use of authentication keys and certificates. The platforms connect to the VPN via any suitable internet connection using the proper authentication credentials. In reference to platform ROS software architecture, the approach utilizes the multi-master presented in [10]. This enables the platform to have a separate ROS core, thus it is self-dependent and does not operate in a centralized paradigm. The proposed scheme allows the platform to access two networks. One for the vehicular communication schemes, and another for any other types of communication.

Accordingly, for the cooperation among the UAV and the intelligent vehicle, the proposed communication approach assures continuous connection among the vehicles, which was verified in the previous work for cooperative driving in [11].

4 EXPERIMENTAL WORK

The approach presented in previous sections has been validated by performing real tests with UAV and UGV platforms. The following subsections will describe the research platforms used, the scenario designed for the experiments.

4.1 Platforms

On the one hand, the experiments have been carried out on a ground autonomous vehicle under the iCab project (Intelligent Campus Automobile). This vehicle consists of an electric golf cart, which has been modified mechanically and electronically to satisfy the goal of autonomous navigation from one point to another within campus vicinity, as shown in Figure 6a.

(a) (b)

Figure 6: Research Platforms

On the other hand, the UAV platform used in the experiments consists of a 3D printed quadcopter as shown in Figure 6b, with a total weight of 1.5Kg. The autopilot used with this quadcopter is the *Pixhawk*, equipped with GPS, magnetometer, IMU and barometer sensors. For the perception purposes, SJCAM SJ4000 camera is used and mounted on Walkera G-2D gimbal, which provides 640×480 RGB images at 30 frames per second. All the processing is performed

on-board by an ODROID-XU4 embedded computer. Finally, both platforms are running Ubuntu 16.04 operating system, and the software architecture has been integrated into ROS middleware. Moreover, in order to avoid a centralized approach and guarantee their operations in an independent way, each vehicle has its own ROS master.

4.2 Scenario

The tests have been performed in outdoor environments, emulating a zebra crossing area, with an autonomous vehicle approaching and several pedestrians crossing. A UAV is hovering while the pedestrian is crossing, detecting both the vehicle and the pedestrians in the area. The relative distance from the vehicle to each pedestrian is computed in the UAV and shared with the UGV to perform safer navigation tasks.

5 RESULTS

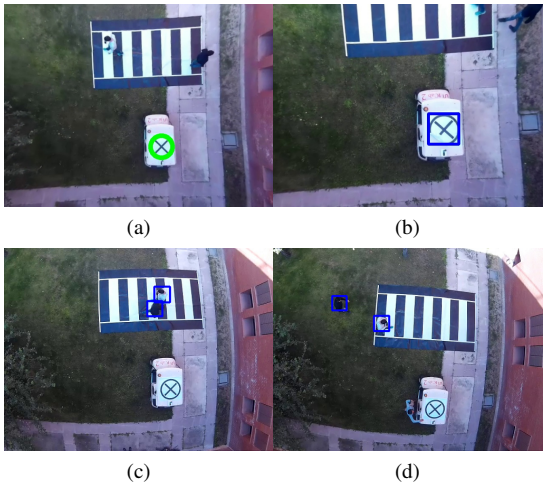
In this section, the results from both vehicle and pedestrian detection are discussed.

5.1 Vehicle Detection Results

Table 1 collects the results obtained in terms of detections of the vehicle are concerned (Figures 7a and 7b). Each of the tables corresponds to each of the two sequences used for the test of the algorithm.

In the first sequence, the lighting conditions are adverse for the perception systems, as the UAV is flying over areas with shadows and light changes. In the second sequence, the tests are performed in a shaded area, where the level of illumination is low, but without light changes.

Both sequences have been tested with three different correlation threshold values, where the vehicle appears is constant and what varies is the number of detections of the vehicle and the false positives.



(a) (b) (c) (d)

Figure 7: Vehicle and Pedestrain Detection results

In the case of false positives, a precise knowledge of the altitude at which the UAV is flying, would allow applying a filter of size of the circle of the pattern. Knowing the height at which you are flying, and since you know the size of the actual model, you can know the size with which the pattern should be detected in the image, which would allow to carry out a filtering by the size of the detected circles, thus decreasing the likelihood of false positives appearing.

It is shown that slightly decreasing the threshold significantly increases the percentage of correct detection, but also increases the number of false positives, that is, there are times when it is detected as good something it is not the vehicle. In addition, it can be seen how the algorithm improves as the light conditions become more suitable for the vision system.

As for the value of the threshold, if you want to avoid false positives, and whenever a detection is known to the 100% that is being detected the vehicle, it will be necessary to establish a threshold of at least 0.94. This may interest you if you want to carry out control actions on the UAV's flight depending on the detection, either to follow the vehicle or to carry out landing maneuvers on it. If instead, what you want is to get a greater number of detections even if you lose reliability, you can reduce the threshold value below the 0.94 set above.

Although sequence 2 was performed under better lighting conditions, the percentage of detections decreased. This is because the number of frames in which the vehicle appears is very low, which causes any detection failure to be penalized more than in the case of sequence 1. Even so, it can be seen that the number of times the pattern is located is over 90%.

5.2 Pedestrian Detection Results

Table 1shows the results obtained in pedestrian detection (Figures 7c and 7d). Both sequences have been tested with 3 different hit-threshold values. In each sequence the number of frames in which the pedestrian pass zebra crossing area is constant. The table illustrates the variation of the detection rate and number of false positives, depending on the threshold with which the detection works. Detection rate is the ratio of pedestrian detected correctly to the total number of pedestrians detected in the frame, and is defined by Equation 5. False positives can be filtered knowing the height at which the UAV is flying and average pedestrian size as stated earlier.

$$DetectionRate = \frac{TruePositive}{TruePositive + FalseNegative} \quad (5)$$

A slight decrement in the hit-threshold value significantly increases the detection rate, but also increases the number of false positives detected. Furthermore, the results in this table support the relation between algorithm detection improvement and light conditions. In sequence 2, the UAV is maintained at a high altitude which provided a challenge in training and detection. The training file size had to be increased in order to maintain a good detection rate.

6 CONCLUSIONS

This paper presents a heterogeneous vehicles cooperation approach to cope with cutting-edge UAVs technology

Table 1: Vehicle and Pedestrian Detection Results

Vehicle Detection				
Sequence	**Threshold**	**Vehicle frames**	**Detection rate %**	**False positive**
Seq. 1	0.92	301	100	65
	0.93	301	100	26
	0.94	301	100	0
Seq. 2	0.92	36	100	7
	0.93	36	91.6	2
	0.94	36	91.6	0

Pedestrian Detection				
Sequence	**Threshold**	**Pedestrian frames**	**Detection rate %**	**False positive**
Seq. 1	0.73	301	98	71
	0.9	301	94.7	32
	0.98	301	89	0
Seq. 2	0.73	308	96.75	43
	0.9	308	94.48	15
	0.98	308	87.01	0

for smart roads. This approach consists of real-time pedestrian and UGVs detection and tracking. These algorithms are studied as a complex and essential task for intelligent vehicles in transportation systems. The proposed algorithms take the advantages of on-board camera for sensing and detecting the two important components in the road (pedestrian and vehicle), and providing information about its position and velocity; to increase the safety in the smart roads.

Different scenarios are evaluated and difficulties have been successfully overcame by means of monocular camera on-board processing, where the pedestrians and UGVs are detected with a total accuracy (>92%).

Future works will focus on the increasing of the environment understanding; which refers to detecting more components in the road, and the combinations of all this information with on-line context information; such as digital maps.

ACKNOWLEDGEMENT

This work was supported by the Comunidad de Madrid Government through the Industrial Doctorates Grants (GRANT IND2017/TIC-7834), and the Spanish Government through the CICYT projects (TRA2016-78886-C3-1-R and RTI2018-096036-B-C21), and the Comunidad de Madrid through SEGVAUTO-4.0-CM (P2018/EMT-4362)

REFERENCES

[1] Fei-Yue Wang, Daniel Zeng, and Liuqing Yang. Smart cars on smart roads: an ieee intelligent transportation systems society update. *IEEE Pervasive Computing*, pages 68–69, 2006.

[2] George Dimitrakopoulos and Panagiotis Demestichas. Intelligent transportation systems. *IEEE Vehicular Technology Magazine*, 5(1):77–84, 2010.

[3] Junping Zhang, Fei-Yue Wang, Kunfeng Wang, Wei-Hua Lin, Xin Xu, Cheng Chen, et al. Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639, 2011.

[4] Kim Insu and Yow KinChoong. Object location estimation from a single flying camera. In *Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM), 2015 9th International Conference on*, pages 82–88. IARIA, 2015.

[5] Chen Yan-yan, Chen Ning, Zhou Yu-yang, Wu Ke-han, and Zhang Wei-wei. Pedestrian detection and tracking for counting applications in metro station. *Discrete dynamics in nature and society*, 2014, 2014.

[6] Mennatullah Siam and Mohamed ElHelw. Robust autonomous visual detection and tracking of moving targets in UAV imagery. In *Signal Processing (ICSP), 2012 IEEE 11th International Conference on*, volume 2, pages 1060–1066. IEEE, 2012.

[7] Y. Qu, L. Jiang, and X. Guo. Moving vehicle detection with convolutional networks in uav videos. In *2016 2nd International Conference on Control, Automation and Robotics (ICCAR)*, pages 225–229, April 2016.

[8] Christopher Reardon and Jonathan Fink. Air-ground robot team surveillance of complex 3d environments. In *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*, pages 320–327. IEEE, 2016.

[9] Sara Minaeian, Jian Liu, and Young-Jun Son. Vision-based target detection and localization via a team of cooperative uav and ugvs. *IEEE Transactions on systems, man, and cybernetics: systems*, 46(7):1005–1016, 2016.

[10] Sergi Hernandez Juan and Fernando Herrero Cotarelo. Multi-master ros systems. *Institut de Robotics and Industrial Informatics*, pages 1–18.

[11] Ahmed Hussein, Pablo Marín-Plaza, David Martín, Arturo de la Escalera, and José María Armingol. Autonomous off-road navigation using stereo-vision and laser-rangefinder fusion for outdoor obstacles detection. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 104–109. IEEE, 2016.

Bibliography

[1] Bart Theys and Joris De Schutter. Forward flight tests of a quadcopter uav with various spherical body diameters. In P. Campoy, editor, *11th International Micro Air Vehicle Competition and Conference*, number IMAV2019-1, pages 12–18, Madrid, Spain, Sep 2019. 8

[2] Sergey V. Serokhvostov and Tatiana E. Churkina. Optimal flight altitude for the small solar-powered airplane. In P. Campoy, editor, *11th International Micro Air Vehicle Competition and Conference*, number IMAV2019-2, pages 19–24, Madrid, Spain, Sep 2019. 8

[3] Fabian Binz and Dieter Moormann. Actuator modeling for attitude control using incremental nonlinear dynamic inversion. In P. Campoy, editor, *11th International Micro Air Vehicle Competition and Conference*, number IMAV2019-3, pages 25–31, Madrid, Spain, Sep 2019. 8

[4] Ana Carolina Dos Santos Paulino, Antoine Murie, Thomas Pavot, Martin Lefebre, Renaud Kiefer, Edouard Laroche, and Sylvain Durand. Experimental versus computational determination of the dynamical model of a glider. In P. Campoy, editor, *11th International Micro Air Vehicle Competition and Conference*, number IMAV2019-4, pages 32–41, Madrid, Spain, Sep 2019. 8

[5] Muhammad Sadeq Ale.Isaac, A. Naghash, and S. H. Mirtajedini. Control and guidance of an autonomous quadrotor landing phase on a moving platform. In P. Campoy, editor, *11th International Micro Air Vehicle Competition and Conference*, number IMAV2019-5, pages 42–48, Madrid, Spain, Sep 2019. 8

[6] Christophe De Wagter, Bart Remes, Rick Ruisink, Freek van Tienen, and Erik van der Horst. Design and testing of a vertical take-off and landing uav optimized for carrying a hydrogen fuel-cell with pressure tank. In P. Campoy, editor, *11th International Micro Air Vehicle Competition and Conference*, number IMAV2019-6, pages 49–54, Madrid, Spain, Sep 2019. 8

[7] Claudio D. Pose, Francisco Presenza, Ignacio Mas, and Juan I. Giribet. Trajectory following with a mav under rotor fault conditions. In P. Campoy, editor, *11th International Micro Air Vehicle Competition and Conference*, number IMAV2019-7, pages 55–59, Madrid, Spain, Sep 2019. 8

[8] Diana A. Olejnik, Bas P. Duisterhof, Matej Karásek, Kirk Y.W. Scheper, Tom van Dijk, and Guido C.H.E. de Croon. A tailless flapping wing mav performing monocular visual servoing tasks. In P. Campoy, editor, *11th International Micro Air Vehicle Competition and Conference*, number IMAV2019-8, pages 60–66, Madrid, Spain, Sep 2019. 8

[9] Federico Magistri, Daniele Nardiyand, and Vito Trianni. Using prior information to improve crop/weed classification by mav swarms. In P. Campoy, editor, *11th International Micro Air Vehicle Competition and Conference*, number IMAV2019-9, pages 67–75, Madrid, Spain, Sep 2019. 8

[10] Colin Greatwood, Laurie Bose, Thomas Richardson, Walterio Mayol-Cuevas, Robert Clarke, Jianing Chen, Stephen J. Carey, and Piotr Dudek. Towards drone racing with a pixel processor array. In P. Campoy, editor, *11th International Micro Air Vehicle Competition and Conference*, number IMAV2019-10, pages 76–82, Madrid, Spain, Sep 2019. 8

[11] Florian Steidle, Wolfgang Stürzl, and Rudolph Triebel. Visual-inertial sensor fusion with a bio-inspired polarization compass for navigation of mavs. In P. Campoy, editor, *11th International Micro Air Vehicle Competition and Conference*, number IMAV2019-11, pages 83–88, Madrid, Spain, Sep 2019. 8

[12] José Arturo Cocoma-Ortega and Jose Martinez-Carranza. A cnn-based drone localisation approach for autonomous drone racing. In P. Campoy, editor, *11th International Micro Air Vehicle Competition and Conference*, number IMAV2019-12, pages 89–94, Madrid, Spain, Sep 2019. 8

[13] Yuchen Leng, Murat Bronz, Thierry Jardin, and Jean-Marc Moschetta. Slipstream deformation of a propeller-wing combination applied for convertible uavs in hover condition. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-13, pages 95–102, Madrid, Spain, Sep 2019. 8

[14] 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 P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-14, pages 103–110, Madrid, Spain, Sep 2019. 8

[15] Dorian N.W.M. Heitzig, Bas W. van Oudheusden, Diana A. Olejnik, and Matej Karásek. Effects of asymmetrical inflow in forward flight on the deformation of interacting flapping-wings. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-15, pages 111–120, Madrid, Spain, Sep 2019. 8

[16] Nikola Gavrilovic, David Vincekovic, and Jean-Marc Moschetta. A long range fuel cell/soaring uav system for crossing the atlantic ocean. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-16, pages 121–131, Madrid, Spain, Sep 2019. 8

[17] Liang Lu, Javier Rodriguez-Vazquez, Adrian Carrio, and Pascual Campoy. Autonomous navigation in dynamic environments using monocular vision. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-17, pages 132–137, Madrid, Spain, Sep 2019. 8

[18] Aldrich A. Cabrera-Ponce, J. Martinez-Carranza, and Caleb Rascon. Detection of nearby uavs using cnn and spectrograms. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-18, pages 138–143, Madrid, Spain, Sep 2019. 8

[19] Dirk Wijnker, Tom van Dijk, Mirjam Snellen, Guido C.H.E. de Croon, and Christophe De Wagter. Hear and avoid for uavs using convolutional neural networks. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-19, pages 144–156, Madrid, Spain, Sep 2019. 8

[20] Patricio Moreno, Santiago Esteva, Ignacio Mas, and Juan I. Giribet. Multi-uav specification and control with a single pilot-in-the-loop. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-20, pages 157–164, Madrid, Spain, Sep 2019. 8

[21] Jorge Diaz Luengo, Joel Bordeneuve-Guibé, and Francois Defay. Model reference adaptive and gain scheduling control for variable payload uav quadcopters. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-21, pages 165–172, Madrid, Spain, Sep 2019. 8

[22] Renan L.S.M. Vilela and Eduardo Costa da Silva. Stability and altitude control of a quadrotor using fuzzy logic. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-22, pages 173–179, Madrid, Spain, Sep 2019. 8

[23] Ana Guerra-Langan, Sergio Araujo-Estrada, and Shane Windsor. Uav control costs mirror bird behaviour when soaring close to buildings. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-23, pages 180–193, Madrid, Spain, Sep 2019. 8

[24] Aaron Lopez Luna, Jose Martinez Carranza, and Israel Cruz Vega. Aerial interaction control using gain-scheduling and pid for a drone with a 2-dof arm. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-24, pages 194–199, Madrid, Spain, Sep 2019. 8

[25] Leticia Oyuki Rojas-Perez and Jose Martinez-Carranza. Flight coordination of drones in gps-denied environments using a metric visual slam. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-25, pages 200–205, Madrid, Spain, Sep 2019. 8

[26] Rachel Axten, Wen-Yu Chien, Jacob Crouse, Oliver Dunbabin, Venkatakrishnan Iyer, Kalki Sharma, and Vidullan Surendran. Autonomous robotics competition club (arcc). In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-26, pages 206–213, Madrid, Spain, Sep 2019. 8

[27] Debjoy Saha, Ganesh Shiridi Balaji Udayagiriy, Parakh Agarwal, Biswajit Ghosh, and Somesh Kumar. Warehouse management using real-time qr-code and text detection. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-27, pages 214–221, Madrid, Spain, Sep 2019. 8

[28] Abdulla Al-Kaff, Ángel Madridano, Ahmed Radwan, Francisco Miguel Moreno, Ahmed Hussein, and Arturo de la Escalera. Heterogeneous multiple vehicles cooperation approach for smart roads. In P. Campoy, editor, *11^{th} International Micro Air Vehicle Competition and Conference*, number IMAV2019-28, pages 222–227, Madrid, Spain, Sep 2019. 8