

http://www.imavs.org/

Proceedings of the 2016 International Micro Air Vehicle Conference and Competition

IMAV 2016

北京理工大學
BEIJING INSTITUTE OF TECHNOLOGY

NUS National University of Singapore

Contents

- *Hailong Qin, Mo Shan, Feng Lin, Yingc ai Bi, Jiaxin Li, Menglu Lan, Yunfeng Zhang and Ben M Chen. A 3D Rotating Laser Based Navigation Solution for Micro Aerial Vehicles in Dynamic Environments.*
- *Zonglin Ye, Shuo Yang, Shuo Zhu, Yazhe Tang, Hui Cao and Yanbin Zhang. A destination swap scheme for multi-agent system with agent-robots in region search problems.*
- *Sam Prudden and Simon Watkins. A Flying Anemometer Quadrotor: Part 1.*
- *Alejandro Rodríguez-Ramos, Carlos Sampedro, Adrian Carrio, Hriday Bavle, Ramon A. Suarez-Fernandez, Zorana Milosevic and Pasc ual Campoy. A Monocular Pose Estimation Strategy for UAV Autonomous Navigation in GNSS-denied Environments.*
- *Yijie Ke, Kangli Wang and Ben M. Chen. A Preliminary Modeling and Control Framework for a Hybrid UAV J-Lion.*
- *Jaime Junell and Erik-Jan van Kampen. Adaptive Path Planning for a Vision-Based quadrotor in an Obstacle Field.*
- *Randa Almadhoun, Tarek Taha, Lakmal Seneviratne, Jorge Dias and Guowei Cai. Aircraft Inspection Using Unmanned Aerial Vehicles.*
- *Yingc ai Bi, Jiaxin Li, Hailong Qin, Menglu Lan, Mo Shan, Feng Lin and Ben M. Chen. An MAV Localization and Mapping System Based on Dual Realsense Cameras.*
- *Thien Minh Nguyen, Abdul Hanif Zaini, Kexin Guo and Lihua Xie. An Ultra-Wideband-based Multi-UAV Loc alization System in GPS-denied environments.*
- *Ruan van Breda and Willie Smit. Applicability of Vec tor Field Histogram Star (VFH*) on Multicopters.*
- *Ye Zhou, Erik-Jan van Kampen and Qi Ping Chu. Autonomous Navigation in Partially Observable Environments Using Hierarchical Q-learning.*
- *Xiaolong Zhang, Zhihong Peng and Hongyang Liu. Autonomous Positioning and Navigation for UAV in GPS-Denied Environments.*
- *Nikita Klyuc hnikov, Anastasia Uryasheva and Nikita Rodichenko. Commercially Suitable Hybrid Multi-Visual, Inertial and Satellite Navigation Unit for Small Unmanned Aerial Vehicles.*
- *Yuying Qian, Kan Liu, Lijun Zhang, Junzhe Lin, Zixuan Lu and Swee King Phang. Design and Implementation of a Configurable Multi-rotor UAV.*
- *Eugenie Soerensen. Development of a Quadrotor with an original shape in order to do rescue applications.*
- *Zhen Li, Jie Guo, Tianyue Chen and Zhenc hang Liu. Deviation Correction and Stability Augmentation Control for UAV Taxiing.*
- *Hann Woei Ho, Guido de Croon and Qi Ping Chu. Distance and velocity estimation using optical flow from a monocular camera.*
- *Gautier Hattenberger, Murat Bronz and Antoine Drouin. Electric Propulsion System Charac terization through Experiments.*
- *Dong Yang, Zhen Liu and Chen Bu. Experimental and Computational Investigation on Interac tion between Nano Rotor and Aerodynamic Rudder.*
- *Niloofar Parhizkar, Abolghasem Naghash and Mehrdad Naghshineh. Experimental Investigation of Rotational Control of a Constrained Quadrotor Using Backstepping Method.*

http://www.imavs.org/

http://www.imavs.org/

- *Quoc -Viet Nguyen, Woei Leong Chan and Marc o Debiasi. Experimental Investigation of Wing Flexibility on Force Generation of a Hovering Flapping Wing Micro Air Vehic le with Double Wing Clap-and-Fling Effects.*
- *Guowei Cai, Adnan Saeed, Fatima Alkhoori, Ahmad Younes, Tarek Taha, Jorge Dias and Lakmal Seneviratne. First-Principles Modeling of A Miniature Tilt-Rotor Convertiplane in Low-Speed Operation.*
- *Kun Li, Ao Chen, Kun Zhang and Ben M. Chen. Improved Efficiency Electronic Speed Controller Development for 3-Phase Brushless DC Motor in Unmanned Aerial Systems.*
- *Ye Zhou, Erik-Jan van Kampen and Qi Ping Chu. Incremental Model Based Heuristic Dynamic Programming for Nonlinear Adaptive Flight Control.*
- *M. ElSalamony, S. Serokhovostov, A.Epikhin and K.Zaripov. Measurement and Calculation of Natural Frequencies of the Longitudinal Flight Modes for Small UAVs.*
- *Li Liang, Zhihong Peng and Xinxing Li. Mixed differential game in Target-Attacker-Defender problem.*
- *Peng Lv, Jianming Fu, Zhenfeng Xia, Fazila Mohd-Zawawi, Emmanuel Benard and Jean-Marc Mosc hetta. Numerical Investigation of a Proof-of-Concept Rotor in Martian Atmosphere.*
- *Alireza Mohamadifard, Abolghasem Naghash, Ali Jamei. Optimal Distributed Cooperative Control for Multiple Quadrotors: Heading Consensus Experiment with Paparazzi Autopilot and AR. Drone 2.*
- *Bart Theys and Joris De Sc hutter. Parameter selection method and performance assessment for the preliminary design of electrically powered transitioning VTOL UAVs.*
- *Torbjørn Cunis, Matěj Karásek and Guido C.H.E. de Croon. Precision Position Control of the DelFly II Flapping-wing Micro Air Vehicle in a Wind-tunnel.*
- *Jiaxin Li, Yingc ai Bi, Menglu Lan, Hailong Qin, Mo Shan, Feng Lin and Ben M. Chen. Real-time Simultaneous Localization and Mapping for UAV: A Survey.*
- *Kexin Guo, Zhirong Qiu, Wei Meng and Lihua Xie. Relative Localization for Quadc opters using Ultra-wideband Sensors.*
- *Mo Shan, Feng Lin and Ben Chen. Salient Object Detection Using UAVs.*
- *Jaco Coetzee and Willie Smit. Simulation of an Obstacle Avoidance Algorithm in a Dynamic 2D Environment.*
- *Kirill Stremousov, Maksim Arkhipov and Sergey Serkhvostov. Small Height Duct Design for 17" Multicopter Fan Considering Its Interference on Quad-copter.*
- *Martin Molina, Adrian Diaz-Moreno, David Palacios, Ramon A. Suarez-Fernandez, Jose Luis Sanc hez-Lopez, Carlos Sampedro, Hriday Bavle and Pasc ual Campoy. Specifying Complex Missions for Aerial Robotics in Dynamic Environments.*
- *Yao-Wei Chin, Yu-Kai Luo, Zi-Yuan Ang and Gih Keong Lau. Spring for clapping X-wing Micro Air Vehicles.*
- *Limin Wu, Yijie Ke and Ben M. Chen. Systematic Modeling of Rotor Dynamics for Small Unmanned Aerial Vehicles.*
- *Woei Leong Chan, Quoc Viet Nguyen and Marc o Debiasi. Tailless Control of a Double Clap- and-Fling Flapping Wing MAV.*
- *Alexander Kern, Markus Bobbe and Ulf Bestmann. Towards a real-time aerial image mosaicing solution.*
- *Gerald van Dalen, Kimberly Mc Guire and Guido de Croon. Visual Homing for Micro Aerial Vehic les using Scene Familiarity.*

A 3D Rotating Laser Based Navigation Solution for Micro Aerial Vehicles in Dynamic Environments

Hailong Qin[2], Mo Shan[1], Yingcai Bi[2], Jiaxin Li[2],Menglu Lan[2]
F. Lin[1], Y.F. Zhang [3] and Ben M. Chen[2]

Abstract— **In this article, a 3D rotating laser-based navigation framework for a micro aerial vehicle (MAV) to fly autonomously in the dynamic environment is presented. The proposed navigation framework consists of a 6-degree of freedom (DoF) localization module and a 3D dynamic mapping module. By extracting and aligning 3D point cloud features from a dense point cloud which generates by a self-designed rotating 2D laser setup, we are able to solve the laser distortion issue while estimating the 6-DoF pose of MAV. In addition, the dynamic mapping module could further eliminate the dynamic trail so that a clean dense 3D map can be reconstructed.**

Our proposed navigation framework detects the dynamic target based on the spatial constraints and propagates without dense point cloud clustering. Through filtering the detected dynamic obstacles, the proposed localization approach is proven to be robust to the environment variations.

We demonstrate the utility of our proposed framework in both real indoor environment with highly dynamic obstacles using a customized MAV platform.

http://www.imavs.org/

I. INTRODUCTION

The applications of MAV on autonomous tasks have been intensively studied [1] [2] [3]. The fundamental requirements for automation of MAV are (1) Onboard sensing capability. The MAV should utilize the onboard sensors such as Laser scanner [4], vision [5]and other sensors (sonar and radar)[6] to obtain the information about an unknown environment. (2) Onboard processing capability. Based on the sensing information, the onboard processor should achieve state estimation for control and environment mapping for planning [7]. (3) Onboard state feedback capability. the state estimation from external sensors should be fused with onboard inertial measurement unit (IMU) or the other sensors for a closed feedback control loop [8]. The above capabilities are critical to the safety navigation of MAV. More importantly, they are in fact interdependent, performance deterioration of single component could degrade the whole autonomous system. We can formalize the mentioned capabilities into sensing, perception(includes state estimation and mapping) and control modules correspondingly.

[1]Mo Shan and F. Lin are with the Temasek Laboratories, National University of Singapore, Singapore. E-mail: {shanmo, tsllinf}@nus.edu.sg

[2]Hailong Qin, Jiaxin Li, Yingcai Bi, Menglu Lan and Ben M. Chen are with the Department of Electrical & Computer Engineering, NUS, Singapore. E-mail: {mpeqh, jli, yingcaibi, lanmenglu, bmchen}@nus.edu.sg

[3]Y.F. Zhang is with the Department of Mechanical Engineering, National University of Singapore, Singapore. E-mail: {mpezyf}@nus.edu.sg

The sensing and perception modules are challenging for MAVs because of the size and computational power constraints. To full fill the autonomous navigation requirements, the following sensor setup is selected in general:

1) Laser, usually known as laser range finder (LRF) is a highly precise active sensor which can provide planarly range measurement to a large range field. Comparing with the passive sensor such as cameras, the laser range finder is still effective in low illumination environment. And the commercial product such as Hokuyo UTM-30LX-EW 2D laser scanner is able to achieve a 30 Hz measurement update which is beneficial to the state estimation. However, the 2D LRF itself cannot measure the 3D environment completely due to the limited vertical measurement [9]. This disadvantage could lead to the failure of MAV autonomous tasks because of the absent of information about the environment for state estimation and obstacle avoidance.
2) Camera, either monocular or multiple camera system is considered as an alternative to the active LRF sensor. The camera system is low in both weight and power consumption makes it promising to the MAV platform. And the camera system is a natural 3D sensing device which full of texture and color information [10]. Yet, the information from camera system is still insufficient to the MAV platform in certain conditions such as illumination variation and low texture environment [11].

Regarding the execution of autonomous task of MAVs, the key requisite is to locate itself in a GPS-denied environment. Moreover, the obstacles should be reliably detected in real time to generate an obstacle-free path for flying [12]. However, it is still challenging to achieve these requirements by the previously mentioned sensor setup individually. Therefore, a combination of these two sensors [13]or the RGBD camera [14]is proposed. Among all the proposed solutions, the vision-based approach is selected as the basic module in the algorithm which still could not resolve the illumination issue [15].

Beyond the sensor setup choice, the perception module is not longer dealing with the static environment alone. Traditionally, the basic assumption for the simultaneous localization and mapping (SLAM) is the static environment due to the reason that the current map measurement should be registered into previous global map. The realistic condition includes a lot of moving target which could confuse the

http://www.imavs.org/

perception algorithm. Even though the vision based approach could reject the moving target in the motion estimation phase through iterative process [16]. However, this approach is not very effective to the slow moving target. In addition, this process is computationally expensive. To the whole autonomous system, the moving target could degrade the performance in following conditions: (1) If the perception module still works (can estimate the ego-motion of MAVs), then the map cannot update properly in the dynamic environments and leave dynamic trails [17]. The dynamic trails could further affect the planning since the obstacles cannot be properly detected. (2) The worst condition is the failure of perception module because of the environment is no longer static, the ego motion of MAVs could not be estimated anymore. Burgard et al. [18] propose to differentiate the dynamic and static cells in 2D grid map by evaluating the expectation maximization. This approach is straightforward to extend to 3D but expensive in computation due to the ray-tracing.

In this article, we present a complete customized and integrated system consisting of a MAV platform with onboard processor, a rotating laser-based localization module and a dynamic mapping module. The MAV platform could provide onboard computation and flying capability. The localization module could utilize the continuous rotating laser scans to achieve a high frequency motion estimation update. And the dynamic mapping module could detect and eliminate the moving target for both localization and further real-time mapping. The developed techniques allow for reliable motions estimation and efficient dynamic map update. The robustness of the proposed framework is evaluated in multiple dynamic environments.

II. RELATED WORKS

The demanding of MAVs to fly autonomously in typical GPS-denied environments such as indoor and forest are rapidly growing. Based on the different platforms and sensors for perception, several research groups have carried out some demonstrations for practical applications. Stephan Weiss et al. [7] use an MAV from ASL and onboard monocular camera together with IMU achieved a flight height of $70m$. However, in contrast to our work this computationally constrained solution could not handle the 3D navigation properly since that the obstacle cannot be reliably detected from sparse points. Similarly, Forster et al. [19] use same sensor setup but a semi-dense image registration approach. The proposed approach for MAV high altitude fly shows that the motion estimation of UAV can be efficient and robust through pixel intensity based alignment. However, computational expensive dense mapping module is required for a complete 3D navigation [20]. To observe the points depth directly, some groups utilize the stereo camera [21]. However, this approach could fail in low feature or texture environment. The RGBD camera is well-suited for the MAVs' indoor navigation task which measures the depth from projecting infrared pattern [2]. Comparing with passive camera systems, the infrared pattern can estimate the depth on the textureless condition such as the white wall. Yet the view field of the infrared pattern is limited which could cause problems when the obstacles are out of MAV view field. To cope with this issue, it is straightforward for the MAVs to equip with multiple RGBD cameras. However, the computational cost rises correspondingly.

The laser-based state estimation technologies are largely adopted by unmanned ground vehicle (UGV). For instance, the utilization of 2D LRF on UAV and small size UGV - Hector SLAM which estimates the 2D motion [22]. Together with the height measurement, a 3D Octomap can be constructed for obstacle detection. However, this approach relies on the reliable and fast height measurement or else will result in an inconsistent 3D map. Similar to this approach, Morris et al. [23] utilize the sparse visual features together with the 2D LRF for motion estimation. Yet, it is still not a direct 3D measurement. In contrast to this approach, our motor driving 2D LRF can directly measure the depth of 3D environment.

Different from 2D LRF, the 3D LRF could provide an accurate 3D dense range measurement regardless of the illumination variation. The sensing capability of 3D LRF makes it possible for UGVs to detect obstacles in all directions [17]. Combined with camera systems for state estimation, the sensing and perception modules are robust and effective. Considering the size and weight limitations, the 3D LDFs are rarely utilized on UAVs. Besides that, the power consumption of 3D LRF could lead to the reduction of endurance.

A combination of visual for localization and laser for obstacle detection is considered as a good choice: Either monocular camera or stereo camera can provide an 6 DoF so that the state estimation is not limited to 2D planner space. Based on the state estimation information, either the static or rotating LRF can construct a 3D map. A similar approach is proposed by Cover et al. [15] use vision for localization and laser for obstacle detection. However, this combination could not work in dark environment and obstacles out of view.

Similar to our approach, Hovermap [24] utilizes a continuous rotating laser on a Micro Aerial Vehicle for state estimation and mapping. However, the GPS signal is fused for a better motion estimation. And more importantly, scenario with moving target is not considered. Matthias et al. [4] build a MAV platform with the camera system for odometry and rotating laser to detect obstacles. Similar to vision-based approach, illumination variation and textureless cause the failure of motion estimation.

The other important issue is dynamic map update. Hahnel et al. [25] consider the moving target as outliers and removed. Yet, this approach is not a correct representation of the map at each time instant due to the lack of moving target. Similar strategy has adopted by [26] [27].

III. SYSTEM CONFIGURATION

The MAV platform is designed by NUS UAV Group with the capability of wind resistance and heavy payload. The platform has a 128 cm tip to tip length and a maximum take-off weight with the 2 kg payload. A self-designed power distributor module provides multiple power sources including 3.3 V, 5 V and 12 V for on-board electronics. The on-board

http://www.imavs.org/

flight controller is Pixhawk with customized flight control algorithms. For the mission control computer, we use the Intel-NUC board. The powerful NUC board equipped with a 5th Generation Intel Core i7-5557U processor. The optimized size is and the optimized weight is only 200 g including a cooling fan. The platform is shown in Fig. 1.



Fig. 1: MAV platform.

IV. TERMINOLOGY AND NOTATION

In this article, we define the body frame of UAV platform as U and local laser coordinate as L. For the rotating laser device, one rotation of the laser range finder is from -90° to 90° in CW or 90° to -90° in CCW as denoted (θ) in Fig. 2 while the x-y plane is the 0° planar plane. Thus for kth rotation, the laser coordinate is expressed as L_k. The global coordinate is O. For a single laser point $p(x,y,z)$ in kth rotation, the coordinate is expressed as $X^U_{k,l,p}$, $X^L_{k,l,p}$ and $X^O_{k,l,p}$ correspondingly.



Fig. 2: Rotating angle illustration in front view and isometric view.

V. ALGORITHM OVERVIEW

In this article, a sensing and a perception modules for MAV in dynamic environments are presented. The proposed algorithm contains following components:

- Feature extraction and alignment: extract the defined feature from 2D laser scan and align the scans in 3D point cloud sequence.
- Motion estimation: estimate the motion based on the optimization of feature alignment error metric.
- dynamic mapping: update the 3D map for dense mapping and motion estimation refinement.

The following section will describe the above components in detail.

VI. FEATURE EXTRACTION AND FEATURE ALIGNMENT

To solve the motion estimation problem efficiently, we select the feature points $p(x,y,z)$ in each scan instead of all the points to build the correspondence between neighbouring scans. The natural properties of laser scan in real environment is large amounts of data, noisy and contains of outliers which makes the feature extraction quite difficult as described by Abdul [28] . In their work, outlier rejection and robust fitting is applied to extract reliable feature from point cloud. For 2D laser scan, variation of a scanning point in a local neighborhood is used to select the feature point. For a point p in a single scan l of kth rotation, the average length of the edges $l_{avg,p}$ incident to it in a local neighborhood:

$$l_{avg,p} = \frac{1}{N} \sum_{q \epsilon N(i)} ||X^L_{k,l,p} - X^L_{k,l,q}|| \tag{1}$$

Where N stands for total number of points adjacent to p and q stands for the neighbor point of p. A relative variation ratio $R_{k,l,p}$ is defined to select edge point $e_{k,l,i}$(with a variation ratio larger than threshold) or flat point $f_{k,l,i}$ (with a variation ratio smaller than threshold).

$$R_{k,l,p} = \frac{l_{avg}}{||X^L_{k,l,p}||} \tag{2}$$

In our work, since we have extracted feature points as edge points and flat points, two Euclidean distance based relationship will be built accordingly to associate the feature points in different rotations.

- To a edge point p in kth rotation, we can search for the nearest neighbor point of p, m in $(k+1)th$ rotation from the beginning of $(k+1)th$ rotation through reprojection. In order to build a point to line correspondence, we further search for a neighbor point of m in consecutive scan and denote it as n. We can calculate the distance from the edge point p to its corresponding line by the following equation:

$$d_e = \frac{|v_l \times v_e|}{|v_l|} \tag{3}$$

  Where v_l stands for $(X^L_{k+1,l,m} - X^L_{k+1,l,n})$ and v_e stands for $(X^L_{k,l,p} - X^L_{k+1,l,m})$ as shown in Fig. 6.
- To a flat point q in kth rotation, we need to search for the nearest neighbor point of q, r in $(k+1)th$ through reprojection. After that, another two neighbor points of r, s and t need to be searched so that three non-collinear points can construct a plane. The objective is to minimize the distance from the flat point q to its corresponding plane. The unit normal of plane can be calculated as :

$$n = \frac{v_{p1} \times v_{p2}}{|v_{p1} \times v_{p2}|} \tag{4}$$

Where v_{p1} is $(X^L_{k+1,l,s} - X^L_{k+1,l,r})$ and v_{p2} is $(X^L_{k+1,l,t} - X^L_{k+1,l,r})$. The distance from flat point q to its corresponding plane can be calculated as

$$d_p = n \cdot v_p \tag{5}$$

where v_p stands for $(X^L_{k,l,q} - X^L_{k,l,r})$.

VII. MOTION ESTIMATION

Translation and rotation of a 3D point p in $(k+1)th$ rotation $X^L_{k+1,l,p}$ with respect to last laser frame L_k is expressed by,

$$X_{k,l,p} = \widehat{X^L_{k+1,l,p}} = R^k_{k+1} X^L_{k+1,l,p} + P^k_{k+1} \tag{6}$$

Where

$$R^k_{k+1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\alpha & -s\alpha \\ 0 & s\alpha & c\alpha \end{bmatrix} \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \begin{bmatrix} c\gamma & -s\gamma & 0 \\ s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P^k_{k+1} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

α, β and γ are the UAV body Euler angle correspondingly. Based on the transformation relationship$T^k_{k+1} = (R^k_{k+1}, P^k_{k+1})$.We can further express the transformation relationship as a nonlinear function f,

$$f(X^L_{k+1,l,p}) = X_{k,l,p} \tag{7}$$

Consider the point to line correspondence in equation (3). To estimate the motion between $(k+1)th$ rotation and kth rotation, we need to minimize the distance d_e to 0 as described in following function.

$$d_e = w_1(f(X_{k+1,l,m}), f(X_{k+1,l,n})) \to 0 \tag{8}$$

Similarly, the point to plane distance d_p can be expressed by,

$$d_p = w_2(X^L_{k,l,q}), f(X_{k+1,l,r}), f(X_{k+1,l,s}), f(X_{k+1,l,t})) \to 0 \tag{9}$$

Through combining the feature point correspondence, we can build a nonlinear function $w(T^k_{k+1}) = d$ to express (8) and (9). Therefore, the square error function can be established as

$$S = (w(T^k_{k+1}))^2 \tag{10}$$

Since the objective value of $w(T^k_{k+1})$ is 0. To minimize the objective function, mainly two methods can be adopted for real time application, Levenberg-Marquardt(LM) [29] method and Trust-Region-Reflective method (TRR)[30]. Compared with the LM approach, TRR is more accurate and less costly when the incrementally updated result is far away from solution. The update step $\sigma_i = (T^k_{k+1})_{i+1} - (T^k_{k+1})_i$ can be obtained by solving

$$\min_{\sigma \in N} \psi_i(\sigma) \tag{11}$$

$$s.t. \|\sigma\|_2 \leq \Delta_k \tag{12}$$

Where $\psi_i(\sigma) = g^T\sigma + \frac{1}{2}\sigma^T H\sigma$ and g and H are the gradient and Hessian, respectively of w evaluated at $(T^{k+1}_k)_i$ and $\Delta_k > 0$ is the trust region radius. We can further define the acceptable ratio by

$$r_k = \frac{(\psi(0) - \psi_i(\sigma_i))}{(w(T^k_{k+1}) - w(T^k_{k+1} + \sigma_i))} \tag{13}$$

Which is used to decide the iteration of trial step σ_i.

VIII. DYNAMIC MAPPING

The safety of autonomous navigation of MAVs requests the 3D map updating with dynamic obstacles detection. Once when the dynamic obstacles are properly detected and removed, the perception module could perform a good localization based on the static objects. In the proposed framework, an efficient structure is kept for dense 3D map so that it could maintain a consistent dense representation in long term. In this article, we only discuss the 3D map without dynamic objects for perception.

The fundamental assumption to identify the dynamic objects is the consistency of visibility: in the continuously updating environment, if a point along the laser ray is visually blocked by a point that we previously observed, then the previously point could be further considered as dynamic moving target. Instead of directly adopting the ray-tracing for visibility validation [31], the point cloud representation for the map is utilized to ensure a smooth information flow. Moreover, the spherical coordinates representation is introduced to represent the geometrical information. The usage of spherical coordinate allows to store the depth of point in a 2D distance map format [32] so that the dynamic obstacles can be efficiently detected. To a point *p(x,y,z)* in one point cloud set, its corresponding spherical coordinate form is

$$\rho = \sqrt{x^2 + y^2 + z^2} \tag{14}$$

$$\theta = atan2(y, x) \tag{15}$$

$$\phi = arccos(\frac{z}{\rho}) \tag{16}$$

To a point cloud set which generated by the kth rotating of laser V^o_k in global coordinate, its associated spherical coordinate form is V^s_k. Initially, we assume that most of the points in V^s_k are static. The spatial difference between a point $X^s_{k+1,l,p}$ and $X^s_{k,l,p}$ is calculated by

$$\left\| X^s_{k+1,l,p} - X^s_{k,l,p} \right\| \tag{17}$$

The spatial difference serves to distinguish between the dynamic points and static points. Moreover, the angle between $X^s_{k+1,l,p}$ and upcoming new point in local spherical coordinate could further propagate the knowledge on the points. The greater the angle difference, the less the previous knowledge changed. Through this approach, the dynamic points are removed so that the motion could be correctly updated. The whole localization and dynamic mapping workflow can be described by Fig. 3.

http://www.imavs.org/

http://www.imavs.org/

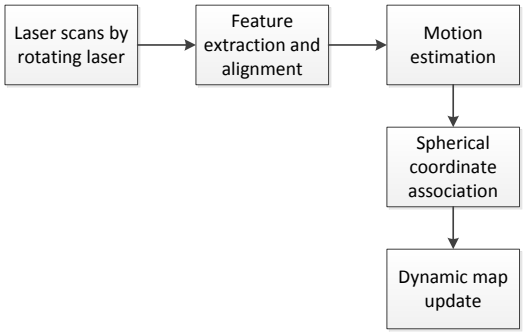


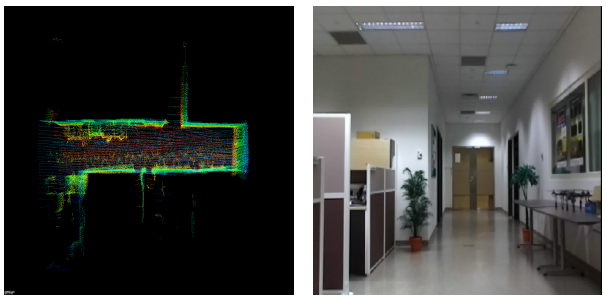
Fig. 3: System workflow.

IX. EXPERIMENTAL RESULTS

The proposed framework is verified in the real environment directly instead of simulated environments to show the robustness and efficiency of the solution. In the designed experiment, the MAV has a maximum flying speed of $1.5m/s$ in clustered indoor corridor. The onboard stereo camera is setup for online visualization of the scene.

A. Flying in corridor environment

This experiment is designed to verify the accuracy of the motion estimation module and mapping module without dynamic obstacles. Since the ground truth of the motion estimation is not available, the accuracy of the motion estimation is evaluated by the reconstructed environment. For instance, the vertical wall should keep upright. The result is shown in Fig. 4.



(a) (b)

Fig. 4: Reconstructed corridor environment.

From the reconstructed environment we can see that the accuracy of motion estimation is good since the reconstructed environment well represented real condition.

B. Hovering with dynamic obstacles

This experiment is designed to test the dynamic module specifically. The MAV is controlled by the information from the motion estimation module in the presence of moving obstacles. The failure of the map updating could lead to the dynamic trails as shown in Fig. 5. Based on the experiment result, the dynamic mapping module removes the continuous moving obstacle(target in black rectangle) successfully and keeps a static map for further motion estimation refinement as shown in Fig. 6 .

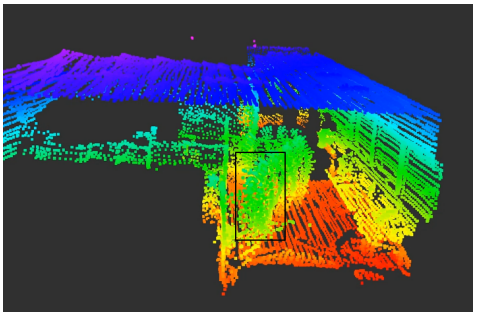


Fig. 5: Moving obstacles caused dynamic trails.

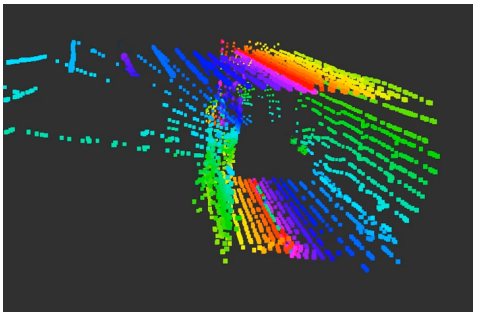


Fig. 6: Dynamic obstacles filtering.

X. CONCLUSION

In this article, we present a novel rotating laser based motion estimation and dynamic mapping framework to achieve MAV autonomous navigation in a GPS-denied environment. The proposed framework solves the laser distortion issue by feature-based motion estimation and dynamic trails through by point based filtering instead of voxel grid based ray-tracing. Further, we show the accuracy and robustness of our proposed framework in the experiments. The rotating laser based framework estimates the ego-motion of MAV while filtering the dynamic obstacles successfully.

REFERENCES

[1] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained mav," in *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE, 2011, pp. 20–25.

[2] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an rgb-d camera," in *International Symposium on Robotics Research (ISRR)*, vol. 2, 2011.

[3] L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing," in *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE, 2011, pp. 2472–2477.

[4] M. Nieuwenhuisen, D. Droeschel, M. Beul, and S. Behnke, "Autonomous navigation for micro aerial vehicles in complex gnss-denied environments," *Journal of Intelligent & Robotic Systems*, pp. 1–18, 2015.

[5] M. Blösch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based mav navigation in unknown and unstructured environments," in *Robotics and automation (ICRA), 2010 IEEE international conference on*. IEEE, 2010, pp. 21–28.

[6] J. Müller and W. Burgard, "Efficient probabilistic localization for autonomous indoor airships using sonar, air flow, and imu sensors," *Advanced Robotics*, vol. 27, no. 9, pp. 711–724, 2013.

[7] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 957–964.

[8] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Autonomous Robots*, vol. 33, no. 1-2, pp. 21–39, 2012.

[9] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, U. Klingauf, and O. von Stryk, "Hector open source modules for autonomous mapping and navigation with rescue robots," in *Robot Soccer World Cup*. Springer, 2013, pp. 624–631.

[10] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor mav," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4557–4564.

[11] S. Yang, S. A. Scherer, K. Schauwecker, and A. Zell, "Onboard monocular vision for landing of an mav on a landing site specified by a single reference image," in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*. IEEE, 2013, pp. 318–325.

[12] S. Lai, K. Wang, H. Qin, J. Q. Cui, and B. M. Chen, "A robust online path planning approach in cluttered environments for micro rotorcraft drones," *Control Theory and Technology*, vol. 14, no. 1, pp. 83–96, 2016.

[13] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2174–2181.

[14] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 4973–4980.

[15] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, and S. Singh, "River mapping from a flying robot: state estimation, river detection, and obstacle mapping," *Autonomous Robots*, vol. 33, no. 1-2, pp. 189–214, 2012.

[16] R. Mur-Artal, J. Montiel, and J. D. Tardós, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[17] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale, and R. Siegwart, "Long-term 3d map maintenance in dynamic environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3712–3719.

[18] W. Burgard, C. Stachniss, and D. Hähnel, "Mobile robot map learning from range data in dynamic environments," in *Autonomous Navigation in Dynamic Environments*. Springer, 2007, pp. 3–28.

[19] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 15–22.

[20] M. Pizzoli, C. Forster, and D. Scaramuzza, "Remode: Probabilistic, monocular dense reconstruction in real time," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2609–2616.

[21] C. De Wagter, S. Tijmons, B. D. Remes, and G. C. 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)*. IEEE, 2014, pp. 4982–4987.

[22] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 155–160.

[23] W. Morris, I. Dryanovski, J. Xiao, *et al.*, "3d indoor mapping for micro-uavs using hybrid range finders and multi-volume occupancy grids," in *In RSS 2010 workshop on RGB-D: Advanced Reasoning with Depth Cameras*. Citeseer, 2010.

[24] L. Kaul, R. Zlot, and M. Bosse, "Continuous-time three-dimensional mapping for micro aerial vehicles with a passively actuated rotating laser scanner," *Journal of Field Robotics*, vol. 33, no. 1, pp. 103–132, 2016.

[25] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1557–1563.

[26] D. Hähnel, D. Schulz, and W. Burgard, "Mobile robot mapping in populated environments," *Advanced Robotics*, vol. 17, no. 7, pp. 579–597, 2003.

[27] D. F. Wolf and G. S. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Autonomous Robots*, vol. 19, no. 1, pp. 53–65, 2005.

[28] A. Nurunnabi, G. West, and D. Belton, "Robust methods for feature extraction from mobile laser scanning 3d point clouds."

[29] S. Roweis, "Levenberg-marquardt optimization," *Notes, University Of Toronto*, 1996.

[30] Y.-x. Yuan, "A review of trust region algorithms for optimization," in *ICIAM*, vol. 99, 2000, pp. 271–282.

[31] F. Ferri, M. Gianni, M. Menna, and F. Pirri, "Dynamic obstacles detection and 3d map updating," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 5694–5699.

[32] P. Gohl, D. Honegger, S. Omari, M. Achtelik, M. Pollefeys, and R. Siegwart, "Omnidirectional visual obstacle detection using embedded fpga," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 3938–3943.

http://www.imavs.org/

A destination swap scheme for multi-agent system with agent-robots in region search problems

Zonglin Ye, Shuo Yang, Shuo Zhu, Yazhe Tang*, Hui Cao†, and Yanbin Zhang ‡

ABSTRACT

As the multi-agent robots can share information and cooperate with each other, they are applied in the region search problems like odor source localization. In this kind of problems, the detection result of the agent-robot at the current destination will influence the location of its next destination. So the route planning of the agent-robots cannot be made in advance. This paper proposes a dynamic destination allocation scheme for the agent-robot group to reduce the total moving distance of the agent-robot group and thus save energy. When the agent-robots start to search the target region, the scheme calculates the first destinations for the agent-robots and allocates the destinations to the agent group to minimize their total moving distance. Then as the distances between the agent-robots and their destinations are different, the agent-robots will reach their target destinations at different moments. In the following searching process, whenever an agent-robot arrives at its current destination and calculates its next destination, the scheme will then find out the best destination allocation plan for the agent-group to minimize the total moving distance of the group. The simulation is performed with particle swarm optimization (PSO) on benchmark functions with agent-robots. The experiments verify the effectiveness of the proposed scheme.

http://www.imavs.org/

1 INTRODUCTION

The multi-agent system (MAS) consists of a number of agents which can interact with each other and react to the environment. Due to its superiority in dealing with complex problems, the notion of MAS has been applied in many aspects like commercial bargain modelling [1], optimization problems [2], wireless sensor network (WSN) [3], and distributed surveillance system [4].

In this kind of application, the agent is a programme sent by the sink node in the WSN or the central computer in the distributed surveillance system when a mission is generated. That is to say, the agents in the application above are virtual agents, not entities. They do not occupy any space, will not be worn, and cost nearly no energy.

However, for multi-agent robots, like robots used for region search operations [5] (e.g. odor source localization [6]), the situation is different. In these two applications, agent-robots consume energy while moving. So the designer should not only make sure that the problem can be solved, but also reduce the energy consumption. Decreasing the total moving distance of the agent-robot group is an effective approach and is meaningful for all the group-search problems with agent-robots.

There are many works focus on the task allocation problems of the agent-robot group or the UAV group [7, 8]. However, in these works, the locations of all the targets are known before the allocation. As in the region search problems like odor source localization, the locations of the targets to be detected are calculated real-timely with the information obtained from the former detection results, other methods are needed to find an optimal allocation scheme for the region search problems.

This paper proposes a dynamic destination allocation scheme for multi-agent system with agent-robots in the region search problems to reduce the total moving distance of the agent-robot group. When the agent-robots start to search the target region, the scheme first calculates the first destinations under the predefined searching algorithm for the agent-robots and allocates the destinations to the agent group to minimize their total moving distance. Then as the distances between the agent-robots and their destinations are different, the agent-robots will reach their target destinations at different moments. In the following searching process, whenever an agent-robot arrives at its current destination and calculates its next destination, the scheme will then find out the best destination allocation plan for the agent-group to minimize the total moving distance of the group. Generally speaking, the destination re-allocation scheme is triggered whenever an agent-robot reaches calculates its next destination. As particle swarm optimization (PSO) [9] is widely used in the region search problems, a simulation of 2-D PSO benchmark function optimization problem with agent-robots using PSO is performed to show the effectiveness of the proposed scheme. As the conventional PSO is a synchronous algorithm and the agent-robots will reach its distance and different moment, an asynchronous version of PSO is used. In this asynchronous PSO, the particles are treated as the real-world robots and

*Yazhe Tang is with the Temasek Laboratories, National University of Singapore, Singapore (email: yztang2008@yahoo.com)

†Corresponding author. Tel.: + 86 13991193207; fax: + 86 029-82668666-2113. Email address: huicao@mail.xjtu.edu.cn

‡Zonglin Ye, Shuo Yang, Shuo Zhu, Hui Cao and Yanbin Zhang are with State Key Laboratory of Electrical Insulation and Power Equipment, School of Electrical Engineering, Xian Jiaotong University, Xian, Shaanxi, 710049

http://www.imavs.org/

their movements cost time. As the distances between the locations of the particles and their destinations are different and we assume that the velocities of the particles are identical, the particles will arrive at their destination at different moments. Whenever a particle arrives at its current destination, its next destination is calculated by its personal best and the current global best. Thus, in this version of PSO, the different particles will have different update times at a specific moment. The experiments verify the effectiveness of the proposed scheme. The influence of the number of the agent-robot to the scheme is also analysed.

The rest of the paper is organized as follows. Section II is the proposed scheme. Section III is the experiment and section IV gives the conclusion.

2 THE PROPOSED SCHEME

In region search problems, the agent-robots need to move in the searching region to find the problem-related target under the predefined algorithm. No matter what the problem and the algorithm are, the agent-robots are always in the iteration composed of three parts: detecting the destination, calculating the next destination, and moving to the next destination. The detection part is accomplished by the sensors equipped on the agent-robot. Then the agents can exchange the information of the problem, like the optimal value being detected and the locations of the obstacles. When calculating the next destination for every agent, the agent-robots are treated as a group in the sense that the knowledge gathered by all the agent-robots is used. However, when moving to the next destination, the agent-robot just moves to the destination calculated by itself. In another word, in the moving process, the agent-robots are treated as individuals and there is no cooperation among them. The following part introduces a dynamic destination allocation scheme for the agent-robot group.

The destination allocation scheme is based on the fact that the next destination of an agent-robot may be closer to another agent-robot. In that situation, if the two destinations are re-allocated between the two agent-robots, the total moving distance and the energy consumption of the two agent-robots can be decreased. In the scheme, the parts of detecting the destination and calculating the next destination are the same as the conventional scheme in the region search problems. Now we show the proposed destination allocation scheme for the process of moving to the next destination. Assume that there are m agent-robots $r_1, r_2, ..., r_m$ in the searching region to detect the target which is at a specific location *X*. The searching process is controlled by a specific algorithm *C*. The process of the scheme can be described as follows:

Step 1: The calculation of the initial locations and the first destinations of the agent-robots. The initial locations of the agent-robots, donated as $x_1^0, x_2^0, x_3^0, ..., x_m^0$ are obtained by *C*. The subscript is the label of the agent-robot and the superscript is the iterations of the agent-robot. For example, x_3^5 is the third location of the 5th agent-robot. The second location of agent-robot *i*, donated as x_i^1 , is also obtained by *C*.

Step 2: The first destination allocation process of the agent-robot group. The purpose of this step is to minimize the total moving distance of the robot group. At the beginning, the agent-robots are at $x_1^0, x_2^0, x_3^0, ..., x_m^0$, and their destinations are $x_1^1, x_2^1, x_3^1, ..., x_m^1$. In conventional algorithms without considering the cooperation in the moving process, the agent-robots will move to their destination, respectively. And the total moving distance of the robot group is $\sum dist(x_i^1 - x_i^0)$, where *dist* is the Euclidean distance. The destination allocation process is used to minimize the total moving distance.

For example, if there are agent-robots *i*, *j*, and *k* with $x_i^1 = p_i$, $x_j^1 = p_j$, $x_k^1 = p_k$, where p_i, p_j, and p_k are the locations of the destinations after step 1. Then after Hungarian algorithm gives an allocation plan, the allocated destinations may be like $x_i^1 = p_i$, $x_j^1 = p_k$, $x_k^1 = p_j$. In this case, the agent-robots *j* and *k* have exchanged their destinations to decrease the total moving distance.

Step 3: The agent-robots with the least moving distance moves to its current destination and gets its new destination.

For an agent-robot *i*, after step 2, its moving distance will be $dist(x_i^1 - x_i^0)$. For the agent-robot group, there will be a robot, labeled as least, with the least moving distance. As we assume that the velocity of all the agent-robots are 1, then robot least will first reach its destination x_{least}^1 after $t_1 = dist(x_i^1 - x_i^0)$. Here t_1 is the time interval between the first and the second allocation. Then based on the information obtained from location x_{least}^1 , the predefined searching algorithm *C* calculates the next location of the agent-robot *least*, which is donated as x_{least}^2 .

Step 4: The destination allocation process of the agent-robot group. When the next destination of the agent with the least moving distance, $x_{least}^{t_{least}+1}$ is obtained, another destination allocation for the agent-robot group is performed. The *l*th destination allocation can be formulated as follows:

$$\arg\min(\sum_{i=1}^{m} dist(x_i^1 - k_{ij}x_j^0))$$

$$s.t. \begin{cases} \sum_{i=1}^{m} k_{ij} = 1 \quad j = 1, 2, 3, ..., m \\ \sum_{j=1}^{m} k_{ij} = 1 \quad i = 1, 2, 3, ..., m \\ k_{ij} = 1 \, or \, 0 \quad i, j = 1, 2, 3, ..., m \end{cases} \tag{1}$$

where x_{jc}^{nj} is the current location of agent-robot *j* when the *l*th destination allocation happens. n_j is the number of times that agent-robot *j* is the first to reach its destination, which also can be interpreted as the times that agent-robot *j* gets its destination by *C*, not by the allocation process. This is a combinatorial optimization problem and can be solved by

Hungarian algorithm which solves the assignment problem in polynomial time [10]. The algorithm can give an allocation plan to the agent-robot group.

$$x_{jc}^{n_j} = x_j^{n_j} + \frac{x_j^{n_j+1} - x_j^{n_j}}{\|x_j^{n_j+1} - x_j^{n_j}\|} \cdot \|x_{least}^{n_{least}+1} - x_{least}^{n_{least}}\| \quad (2)$$

$$x_{least\,c}^{n_{least}} = x_{least}^{n_{least}+1} \quad (3)$$

where $x_{least}^{n_{least}+1}$ least is updated by *C*.

In our scheme, Hungarian algorithm is also used to solve (2) and gives a destination re-allocation plan for the agent-robot group.

Step 5: Iterate step 3 and 4, until algorithm *C* ends.

The following section will illustrate the performance of the proposed scheme on some benchmark functions of PSO algorithm, which is widely used in region search problems.

3 EXPERIMENTS

In this section, six test functions (Bohachevsky, Griewank, Michalewicz, Rastrigin, Rosenbrock, and Schwefel) are used to verify the effectiveness of the proposed scheme. The concept of PSO was first suggested by Kennedy and Eberhart [9]. The algorithm uses a group of particles to search the feasible region and updates the locations of the particles by considering the current location of the particle, the history best location of the particle, and the global best location of the particle group. For the *i*th particle P_i, let v_{id}^t donate the velocity on the *d*th dimension at the *t*th iteration. Similarly, x_{id}^t is its location on the *d*th dimension at the *t*th iteration. Let p_{id}^t and p_{gd}^t be the particle best location and swarm best location till the *t*th iteration. Then the velocity of the *i*th particle on the *d*th dimension is updated according to the following equation [9]:

$$v_{id}^t = wv_{id}^{t-1} + c_1 r_1 (p_{id}^t - x_{id}^t) + c_2 r_2 (p_{gd}^t - x_{id}^t) \quad (4)$$

where *d* = 1, 2, ..., *D*, and *D* is the dimension of the searching space. $c_1$and c_2 are acceleration factors indicating the degree that the particle is attracted by its personal best position and the global best position. These two constants are set as 2 in order to make the average velocity change coefficient close to 1 [9]. r_1 and r_2 are uniformly generated random numbers with a scope of [0, 1]. w is the weighting factor controlling the searching region of the particles. Then the location of the *i*th particle on the *d*th dimension is updated as follows:

$$x_{id}^{t+1} = x_{id}^t + v_{id}^t \quad (5)$$

For the experiments with real robots, moving from one place to another costs time. Usually, the agent-robots will not arrive at their destinations at the same time. Comparing with the strategy that updating the new destinations synchronously when each member of the agent-robot group reaches its destination, we would rather do the destination allocation process whenever an agent-robot reaches the destination. Thus, unlike the conventional PSO, the PSO used in the experiment is an asynchronous PSO. The asynchronous PSO is

$$v_i^{n_i} = wv_i^{n_i-1} + c_1 r_1 (p_i^{n_i} - x_i^{n_i}) + c_2 r_2 (p_g^l - x_i^{n_i}) \quad (6)$$

$$x_{id}^{n+1} = x_{id}^n + v_{id}^n \quad (7)$$

The experiment uses the asynchronous PSO described above and the asynchronous PSO with the destination allocation scheme on six benchmark functions to compare the total moving distance of the agent-robot group in the entire searching process. For each function, the asynchronous PSO algorithm and the asynchronous PSO with the proposed scheme both run for 100 times. The number of the iterations of the algorithm is set to be 1000 before it ends in each run. The average value of the total moving distance of the agent-robot group in the two methods, the percentage of the distance decreased by the proposed scheme, and the results of the Wilcoxon rank-sum test are listed in table 1. From Table 1, it is clear that the proposed scheme can significantly reduce the moving distance of the agent-robot group for all the benchmark problems. For all the functions, the proposed scheme helps to decrease the total moving distance of the agent-robot group by nearly one quarter except one function. The maximum reduction happens on the Rastrigin function, which is 24.2 %. With a significance level of 5%, the p-value also indicates that the superiority of the proposed scheme does not happen by chance. And the superiority also holds over Bohachevsky function.

Another experiment is performed to investigate the relationship between the number of the robots and its effect on the proposed scheme. For Griewank function, the experiment is performed for 100 times with the number of the robots increasing from 6 to 12. The average values of the total moving distance of the two schemes in 100 experiments of Griewank function with the number of the robots ranging from 6 to 12 are in Table 2. From this table, it can be found that the total moving distance of the two schemes change with the number of the robots. The total moving distance of the proposed scheme is always smaller than that of the conventional PSO. The percentage of distance decreased varies, ranging from 5.7% to 47.4%. It can be seen that the number of the robots is not the only element that affects the total moving distance. More experiments are needed to discover the other elements.

4 CONCLUSIONS

This paper proposes a dynamic destination allocation scheme for the agent-robot group to reduce the total moving distance of the agent-robot group and thus save energy. As PSO is widely used in region search problems, experiments

http://www.imavs.org/

Benchmark functions	Asynchronous PSO with proposed scheme	Asynchronous PSO	Decreased percentage (%)	P-value
Bohachevsky	3.0040E+03	3.1384E+03	4.4	4.9304E-04
Griewank	1.3593E+04	1.7271E+04	22.3	1.6611E-16
Michalewicz	682.0137	883.4735	22.8	1.4568E-18
Rastrigin	1.0352E+04	1.3666E+04	24.2	1.5763E-20
Rosenbrock	9.8706E+03	1.2909E+04	23.6	3.2185E-19
Schwefel	3.9774E+05	5.2008E+05	23.5	1.9068E-15

Table 1: The average of the total moving distance on six benchmark functions of the two schemes

The number of the robots	The proposed scheme (1E+03)	Asynchronous PSO (1E+03)	Decreased percentage (%)
6	0.6375	0.7141	10.7
7	0.8610	0.9892	13.0
8	1.3593	1.7271	21.3
9	1.2319	1.5703	21.6
10	1.6122	2.0451	21.1
11	1.6041	3.0515	47.4
12	1.6042	1.7009	5.7

Table 2: The total moving distance with the changing of the number of the robots for the two schemes

http://www.imavs.org/

are performed on the benchmark functions to verify the effectiveness of the proposed scheme. From our research, several conclusions can be obtained. First, the proposed scheme can significantly reduce the total moving distance of the agent-robot group. Second, like the conventional PSO, the asynchronous PSO can find the optimal value of the region search problem. Finally, the asynchronous PSO with the proposed scheme can reduce the total moving distance of the agent-robot group compared to the asynchronous PSO itself. In the future, the simulations and experiments on practical region search problems like odor source localization need to be implemented. More properties of the asynchronous PSO also deserve further analysis.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China under Grant 61375055, the Program for New Century Excellent Talents in University under Grant NCET-12-0447, the Natural Science Foundation of Shaanxi Province of China under Grant 2014JQ8365, State Key Laboratory of Electrical Insulation and Power Equipment EIPE16313 and the Fundamental Research Funds for the Central University.

REFERENCES

[1] M.H. Chen and L. Wang. Impact of individual response strategy on the spatial public goods game within mobile agents. *Applied Mathematics and Computation*, 251:192–202, 2015.

[2] X.F. Xie and J.M. Liu. Multiagent optimization system for solving the traveling salesman problem (tsp). *IEEE Transactions on Systems Man And Cybernetics Part B*, 39(2):489–502, 2009.

[3] D.Y. Ye, M.J. Zhang, and D. Sutanto. Self-organization in an agent network: A mechanism and a potential application. *Decision Support Systems*, 53(3):406–417, 2012.

[4] J.M. Gascuena and F.C. Antonio. On the use of agent technology in intelligent, multisensory and distributed surveillance. *The Knowledge Engineering Review*, 26(2):191–208, 2011.

[5] D.W.Gong and C.L. Qi. Modified particle swarm optimization for odor source localization of multi-robot. In *IEEE Congress on Evolutionary Computation (CEC)*, 2011.

[6] A.T. Hayes, A. Martinoli, and R.M. Goodman. Swarm robotic odor localization: Off-line optimization and validation with real robots. *Robotica*, 21(4):427–441, 2003.

[7] M. Alighanbari, Y. Kuwata, and J.P. How. Coordination and control of multiple uavs with timing constraints and loitering. In *Proceeding of the IEEE American Control Conference*, 2003.

[8] S. Rathinam and R. Sengupta. Lower and upper bounds for a multiple depot uav routing problem. In *Proceeding of the IEEE American Control Conference*, 2003.

[9] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, 1995.

[10] Pentico and W. David. Assignment problems: A golden anniversary survey. *European Journal Of Operational Research*, 176(2):774–793, 2007.

http://www.imavs.org/

A Flying Anemometer Quadrotor: Part 1

S. Prudden*, A. Fisher, A. Mohamed, and S. Watkins.
RMIT University, Melbourne, Australia

http://www.imavs.org/

ABSTRACT

The feasibility of using a Multi-rotor Unmanned Aerial System (MUAS) as flying anemometers for measurement of wind in urban environments was investigated. Flow mapping was conducted around the MUAS using smoke flow visualisation and multi-hole pressure probes to analyse the effects of the propellers on the measured flow speed and direction and determine a suitable mounting location for an on-board multi-hole pressure probe sensor system. It was determined that propeller-induced effects extended beyond a feasible on-board probe length. Therefore correction factors were developed from experimental measurements for application to the output of the on-board multi-hole pressure probe.

NOMENCLATURE

X	Longitudinal measurement position (mm)
α	Propeller-induced flow deflection angle (degrees)
Φ	MUAS angle of attack (degrees)
u	Flow speed (m/s)
V	Flow velocity (m/s)
ω	Angular velocity (rad/s)
λ	Tip speed ratio
r	Propeller radius (m)

1 INTRODUCTION

Knowledge of wind flow around buildings has a number of useful applications in architecture and construction industries. Wind data can be used to optimise the locations of wind turbines and photovoltaic cells on buildings in urban and city environments, assess effects of flow separation on structural integrity as well as investigate the feasibility of Micro Aerial Vehicles (MAVs) soaring at the tops of tall buildings [1, 2, 3, 4].

Measuring wind flow around buildings presents a number of practical challenges due to the size and complexity of the structures and frequently wind tunnel small scale models and wind tunnel are used, however this method can be expensive and time consuming [5]. When analysing complex turbulent flows, assumptions and simplifications must be made. Computational Fluid Dynamics (CFD) has an advantage in that a large number of combinations of tests can be conducted at a small portion of the cost of using a wind tunnel. However, they require extensive computing power and the resolutions are limited [6].

In-situ measurements of flow around buildings have been traditionally undertaken using static sensors fixed to masts and/or the structure being analysed. Atmospheric measurements for flow mapping require sample times of the order of minutes at discrete point locations in order to resolve for low frequency wind oscillations. Some forms of atmospheric boundary layer measurements also require multiple simultaneous measurements [3]. This introduces practical limitations, as multiple probes require significant time and resources to mount. LIDAR can be used to provide ground-based remote wind measurement of altitudes up to 200m [7]. However these are expensive, suffer from limited resolution and produce spatially-averaged results over a relatively large sensing area [8]. There is an opportunity to utilise Multi-rotor Unmanned Aerial System (MUAS) as "flying anemometers'.

MUAS have the advantage over fixed-wing UAS due to their ability to move in all directions, hover at a fixed location in space and take off and land vertically. This makes them more suitable for taking point measurements close to buildings and flying in confined urban environments. MUAS have demonstrated the ability to carry sensors for precision 3D mapping in confined spaces of both forests and mineshafts [9, 10].

Existing research projects have investigated measuring the MUAS dynamic behaviour or differential power measurements to estimate wind fields to a limited resolution [11, 12, 13]. The miniaturisation of flow sensors, such as pressure or ultrasonic-based sensors, have allowed these payloads to have a minimal impact on the MUAS airframe size, weight and performance. Distributed pressure sensors and sonic anemometers have been investigated as onboard sensors for path planning and atmospheric measurements respectively [14, 15]. However the effects of propeller wake on on-board wind measurements are not well understood. A major focus of existing propeller wake studies in hover and forward flight has been on the regions underneath or downstream of the rotors, which are obviously unsuitable for flow sensors. Mounting positions above, or in front, of the MUAS airframe might remain upstream of the major rotor wash when using a suitable control system. Therefore the potential rotor-induced effects in these regions must be well-understood and characterised in order to accurately measure atmospheric flows.

In this paper we explore the feasibility of using an MUAS as a wind-sensing platform to accurately resolve atmospheric

*Email address: sam.prudden@gmail.com

http://www.imavs.org/

flow vectors at point locations. MUAS have potential advantages over ground- and pole-mounted sensors because they can be quickly relocated to measure wind at multiple locations. Advances in autonomous control systems may also allow for swarms of MUAS to sense multiple locations in real time for flow mapping, such as measuring vertical velocity or intensity profiles. Additional applications for on-board flow sensing on MUAS may extend to measuring the upstream flow to assist reducing pertubations in gust, such as the phase-advanced attitude control system demonstrated for fixed-wing MAVs by Mohamed [16].

In this paper there is a preliminary analysis of the induced effects of MUAS propellers on the incident flow field in front of the airframe using Multi Hole Pressure Probes (MHPP). Qualitative flow visualisation and quantitative flow mapping were used to determine the induced flow angle and speed in a range of flight profiles. These measurements could then be used to identify a suitable mounting location for an on-board forward facing multi-hole pressure probe (MHPP) as well as identify limitations and required corrections. The research in this paper forms part of a series of studies that include development and integration of a purpose-built lightweight MHPP system as well as airborne wind measurement testing.

2 EXPERIMENTAL SETUP

2.1 Test Platform

The MUAS platform used in this project has been designed to accommodate both wind tunnel and flight testing. The platform is a quad-rotor configuration with diagonal rotor spacing of 800mm and a take-off weight of 2.4kg. The propellers have a diameter of 330mm and pitch of 119mm. The airframe has been designed to mount a forward-facing mount for a Multi-Hole Pressure Probe (MHPP) in line with the longitudinal axis of the Inertial Measurement Unit (IMU) to enable the measurements to be taken upstream of the rotor wash and reduce the complexity of attitude corrections when resolving the MHPP in the body-fixed coordinate frame (BCF) to the global coordinate frame (GCF). Methods used to resolve the flow vectors to the GCF during flight will be covered in Part B of this research. The MHPP has been developed at RMIT and has a 90 degree cone of acceptance to allow the large turbulence fluctuations in urban environments to be captured, as found by Milbank et al [17]. It is assumed that in the flying configuration the closed loop control system will adjust the heading angle to ensure that the mean wind vectors remain with the sensor cone of acceptance. The MHPP longitudinal mounting position can be adjusted to make use of the flow mapping results and place the probe in the position where it is least severely affected by propeller wash but not impeded by structural stiffness or vibration, as demonstrated by de Boisblanc [15].

The implications and method of resolving wind measurements from a moving platform will be covered in a future paper.

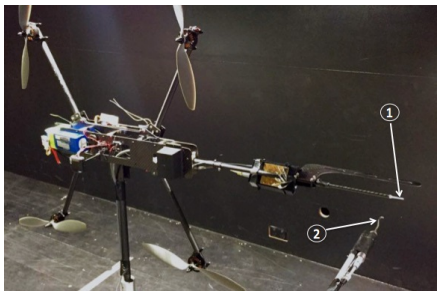


Figure 1: The mounted MUAS showing the on-board MHPP mounting configuration (1), and the Cobra Probe MHPP used for flow mapping (2)

2.2 Wind Tunnel Setup

The RMIT Industrial Wind Tunnel uses a return cycle configuration and has a test section of 9 x 3 x 2m. The base turbulence intensity of the tunnel is 1.5% [18]. The MUAS was held on its right side on a static sting. This orientation allowed the MUAS to be held out of ground effect while minimizing the length of the sting, thus optimizing the stiffness of the rig. The sting could also be rotated axially to investigate the effects of pitch and roll on flow measurements, thus replicating hover conditions in a constant wind speed or forward flight.

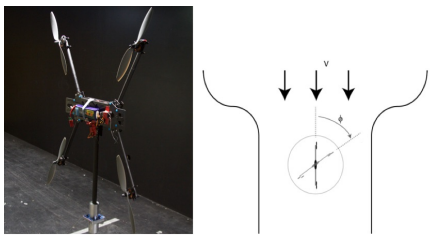


Figure 2: Wind tunnel mounting configuration

A TFI Cobra Probe [19] was used to measure local flow speed, angle and turbulence intensity at a range of longitudinal positions.

3 RESULTS AND DISCUSSION

Extensive wind tunnel testing was conducted to map the flow field along the longitudinal axis in front of the MUAS airframe. This was done to investigate the minimum distance required to avoid propeller-induced effects. This information was utilised to identify the most suitable position for the MHPP in front of the airframe. Quantitative flow measurements were conducted using a Cobra Probe at 1000Hz for 28 seconds in order allow both high and low frequency fluctuations to be analysed. Measurements were taken at least three times per configuration for repeatability and flow angles measured in each configuration were found to have an error within ± 1 degree.

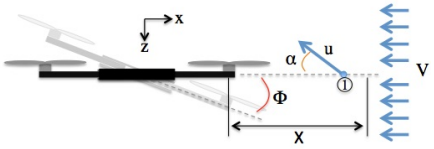


Figure 3: Parameters analysed in flow mapping experiments

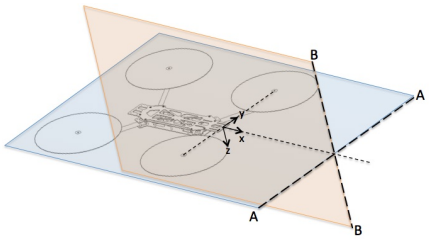


Figure 4: Flow visualisation planes

The parameters analysed in these experiments are shown in Figure 3. X represents the longitudinal distance for flow measurements forwards of the rotor hub axis. Φ is the MUAS angle of attack in degrees and V is the tunnel velocity in metres per second. If we consider any point in the flow, such as point 1 on Figure 3, α and u represent the propeller-induced deflection angle (degrees) and flow speed (metres per second) measured by the Cobra Probe. As shown in Figure 4, AA represents the smoke wire mounting orientation in the lateral plane, where the wire is level with the propeller axis. BB represents the smoke wire in the vertical plane, where the wire is at the mid-point between the rotor hubs. Both wires were placed at approximately X=2r. The MHPP was designed to be mounted in line with the longitudinal axis of the flight controller and level with the propeller hubs in the z-axis. Therefore, due to the scope of this project, quantitative flow measurements were not conducted at a range of positions in the z-axis and instead focused purely on the longitudinal axis. Analysis of flow fields at additional locations may be conducted in future research.

3.1 Effects of Reynolds Number

The deflection angle, α, is a function of tip speed ratio, λ, and Reynolds number. λ is a dimensionless ratio between the angular velocity of the propellers and the linear velocity of the air, as shown in Equation 2. An initial experiment was conducted to assess the effect of the Reynolds numbers within the test matrix. This was conducted by varying the tunnel flow speed and propeller RPM to produce the same tip speed ratio and measuring the flow deflection. Measurements were taken at longitudinal positions of X=200, 300, 400 and 500mm. The ratios are shown in Table 1. The ratios were selected to operate between the limits of the Cobra Probes minimum velocity and the maximum thrust able to be safely contained on the MUAS sting. The flow deflection was found by measuring the flow speed and angle with the propellers stationary and spinning in order to account for the initial mounting angle.

$$\alpha = f(\lambda, Re) \tag{1}$$

$$\lambda = \frac{\omega r}{V} \tag{2}$$

Wind Speed (m/s)	Propeller RPM	Tip Speed Ratio (λ)
4.17	2500	5.18
6	3600	5.18
7	4200	5.18

Table 1: Variables used to assess the effects of Reynolds number

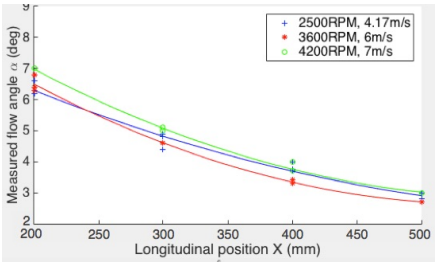


Figure 5: Flow measurements at three combinations of V and ω for λ=5.18

With a constant λ, the resultant deflection angle measured at each X position demonstrated less than 1 degree of variation between the three test configurations. This indicates that the Reynolds numbers did not significantly influence the deflections, thus allowing a uniform RPM to be used for all tests. In order to identify the propeller RPM at hover, the motor/propeller system used in the MUAS was first analysed using a force balance and the thrust curve was identified. The thrust of four propellers required to lift the weight of the MUAS platform corresponded to 3600RPM. This value was subsequently used for all further testing.

3.2 Propeller Flow Visualisation

Flow visualisation was used to qualitatively measure the induced effects of the propellers on the incident flow upstream of the airframe, which is a region that has seen very little analysis in previous research. The aim was to gain an understanding of the maximum distance significantly influenced by the propellers, the regions most severely affected as well as the amount of induced turbulence and flow mixing along the longitudinal axis where an on-board MHPP would be mounted. Two planes were analysed, as demonstrated in

http://www.imavs.org/

Figure 4. In both tests the MUAS had zero pitch and yaw angle relative to the tunnel flow and equal thrust was generated from all four propellers. Smoke was generated using a heated wire covered in mineral oil. The tunnel flow speed was held at 5km/h to prevent excessive smoke dissipation. The turbulence intensity of the wind tunnel combined with vortex shedding from the smoke wire caused some visible turbulence in the baseline tests with stationary propellers. The propellers were spun at 830RPM in order to maintain a tip speed ratio of 5.18, which corresponded with all quantitative tests.

The streamlines in the lateral plane, shown in Figure 6, demonstrated significant lateral displacement, with the angle, θ, increasing along both the longitudinal and lateral axes towards the airframe and propellers respectively. The streamlines approximately $\pm$30mm from the centreline of the airframe demonstrated very little deflection, indicating that the wakes of the left and right propellers do not interact in this configuration. Despite being turned outwards, the streamlines also appeared to remain relatively laminar until reaching downstream of the vortices caused by the propellers and front arms.

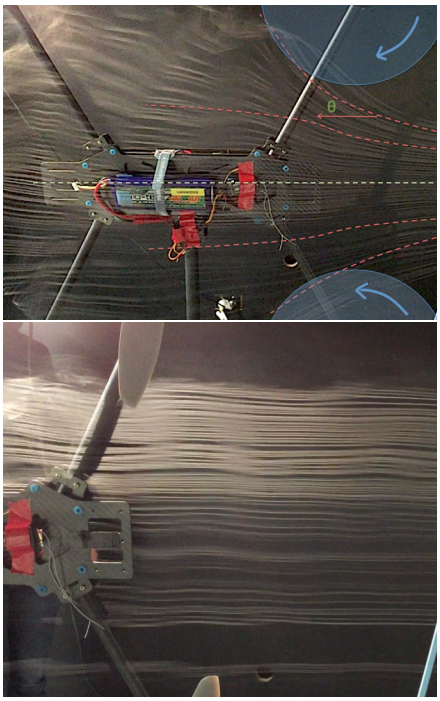


Figure 6: Smoke flow results in the lateral plane with propellers spinning (top) and stopped (bottom)

The streamlines in the vertical plane, shown in Figure 7, also demonstrated significant vertical displacement within a distance of 300mm from the rotor hub axis. The streamlines approximately 100mm above and below the propeller axis demonstrated similar deflection angles. It can also be noted that because the smoke streamlines passed equidistant between the tandem rotors, the maximum amount of flow deflection was experienced at a position towards the rear of the rotor, unlike a single rotor that creates a stream-tube near the leading edge of the rotor. Relatively laminar flow was observed upstream of the rotor hub axis. The flow also appeared to develop a boundary layer over the airframe.

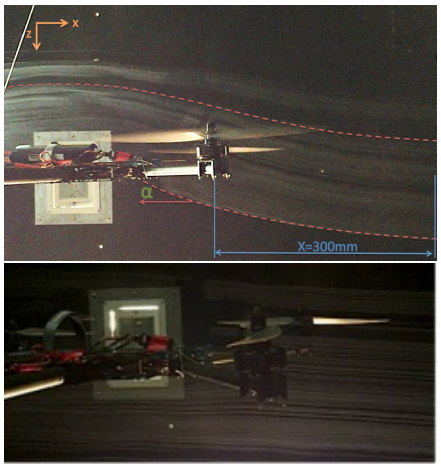


Figure 7: Smoke flow results in the vertical plane with propellers spinning (top) and stopped (bottom)

These images indicated that the region approximately two rotor radii upstream of the rotor hub axis is unsuitable for mounting a MHPP due to the significant propeller-induced flow angle. Therefore a MHPP cannot be mounted directly to the airframe on the longitudinal axis and will instead require an extended mount to place it further forwards. The smoke flow did not provide an indication of whether the propellers had increased the flow speed, however it did demonstrate that the propellers caused little increase in turbulence upstream of the rotor hub axis. This experiment also did not account for a range of incident wind angles in pitch and yaw.

3.3 Quantitative analysis of propeller-induced airflow deflection

3.3.1 Propeller Influence Region

An experiment was conducted to identify the maximum longitudinal distance in front of the MUAS where the flow was influenced by the propellers to determine the minimum probe length required to avoid these effects. This test was first conducted at V=6m/s and an MUAS angle of attack of 0, 15 and 30 degrees. Measurements were taken between X=0mm and X=800mm in 100mm increments.

The propeller-induced flow deflection, α, was found to increase exponentially as X was reduced and the probe was brought closer to the propellers. A position of X=0mm the deflection reached approximately 14 degrees. These quantitative measurements reflected the smoke flow images presented

http://www.imavs.org/

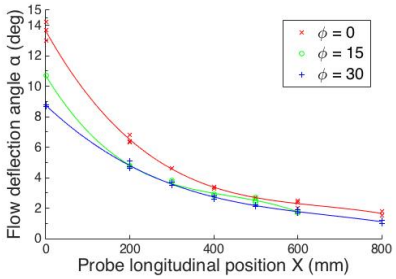


Figure 8: Flow deflection at λ=5.18

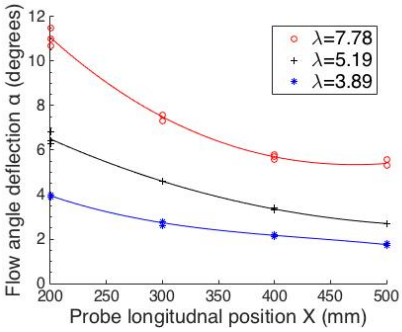


Figure 9: Flow deflection at Φ=0 deg

in Figure 3.2.

It was determined that the propellers influenced the flow beyond a distance of X=800mm albeit at a relatively small angle. X=800mm represents a probe length that is approximately 250% longer than the MUAS airframe and 6.7 radii forwards of the rotor hub axis. Mounting a probe this far in front of the MUAS would most likely introduce structural implications such as flex and a weight penalty caused by the probe mounting structure, difficulty achieving an ideal centre of gravity as well as impeding the manoeuvrability of the MUAS. Therefore measurements beyond this position were not tested. As a result, passively avoiding the propeller induced effects on flow measurements by placing the probe further forward was deemed impractical.

Additional measurements were taken at X=200, 300, 400 and 500mm to assess the effects of different wind speeds on the propeller-induced flow deflection. This was conducted at V=4, 6 and 8m/s, where λ =7.78, 5.19 and 3.89 respectively. Each of these flow speeds was tested at Φ =0, 15 and 30 degrees. These configurations represented a range of operating conditions that an MUAS would experience while hovering.

For a given λ, α was found to reduce as Φ was increased, which may be caused by the propeller thrust vector becoming more in-line with the wind flow vector at higher angles of attack and therefore causing less vertical deflection. This is noticeable between the results shown in Figure **??** and Figure **??**. However, at each value of Φ, α was found to become relatively constant beyond X=400mm, with the difference being within experimental error. As a result, it could be determined that the effect of Φ on beyond X=400mm could be regarded as being insignificant.

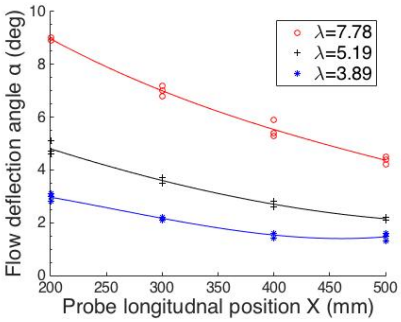


Figure 10: Flow deflection at Φ=30 deg

It can be seen in Figure **??** that α decreases as λ is decreased at a given value of X. As V is increased, the longitudinal component of the flow velocity, u, is increased. Therefore, for a given propeller-induced fluctuation of the vertical velocity component, w, an increase in u will reduce the perceived α angle measured by the probe. Despite the significant flow angle deflections, the flow acceleration caused by the propellers was found to be less than 0.1m/s, or less than 2%, at all longitudinal positions. As a result, flow speed was not used as a deciding factor in the probe mounting position.

3.3.2 Calibration Offset Functions

As it was determined that operating a MHPP outside the effects of propeller wash was structurally impractical, the feasibility of actively offsetting the propeller-induced effects on the measurements taken by a MHPP was investigated. The flow field measurements in the vertical plane had demonstrated good repeatability. Surfaces were fitted to the data points to allow the function α = f(Φ, λ) to be calculated for arbitrary Φ and λ. These surfaces were determined to be 2nd-3rd order polynomials. The surfaces could be calculated at different values of X, however ultimately only one value of X was required once the probe mounting location had been finalized. Therefore this offset method investigated using these calibration surfaces to interpolate the resultant effect of the propellers on the measured flow angle using sensor measurements, thereby allowing an angle offset to be applied in real time.

All of the required parameters could theoretically be measured onboard an MUAS using different sensors. It was found that the propellers did not have a significant impact on the measured flow speed, u. Therefore the tip speed ratio, λ, would be able to be determined using flow speed measured by the MHPP as well as RPM measured by onboard tachometers. The MUAS angle of attack, α, could be measured using the

http://www.imavs.org/

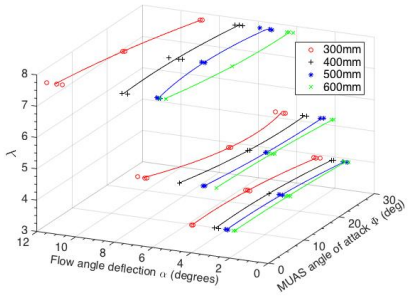


Figure 11: An example of the correction curves used to interpolate α from Φ, λ and X

onboard IMU. This would allow to be resolved for a MHPP mounting location, i.e. X as shown in Figure 3. The frequency at which corrections could be applied would depend on the sampling rate of each of the required sensors, and the accuracy would also be dependent on the coarseness of the calibration functions. For this project it was decided to apply the calibration offsets to mean angle measurements taken at fixed combinations of Φ, λ and X. Application and accuracy of these initial correction functions to the on-board MHPP sensor in both statically-mounted and flying configurations will be covered in the next edition of this research project. Incorporation of yaw angles and differential thrust into the correction functions will be conducted in future research in order to correct for wind measurements that are not closely aligned with the vertical plane of the MUAS. The current calibration model has been developed using time-average flow measurements at fixed attitudes with the assumption that the MUAS will hold relatively stable hover in wind. Therefore the effects of the dynamic motion of the MUAS in turbulent conditions on the accuracy of the resultant measurements will need to be investigated. These effects include the impact of the angular velocity of the probe tip on differential pressure measurements during rapid pitching due to the long probe moment arm, as well as the impact of rapidly changing the propeller tip speed differentially during changes in attitude.

4 CONCLUSIONS

In order to investigate the ability of a Multi-rotor Unmanned Aerial System (MAUS) to measure atmospheric winds, the flow around a typical MUAS was studied in a series of wind tunnel tests that replicated steady-state hover and forward flight. A multi-hole pressure probe (MHPP) was used to quantify propeller-induced flow angle as a function of tip speed ratio, MUAS angle of attack and longitudinal position forwards of the MUAS. It was determined that the on-board sensor would need to be mounted more than 7 rotor radii forwards of the hub axis to avoid all induced flow effects, which was considered to be structurally impractical. However, it was determined that the induced effects were relatively small at at X=400mm, or approximately 2.5 rotor radii, and could be corrected for. This position also presented a feasible sensor mounting location without impeding MUAS maneuverability or inducing adverse deflections. Correction functions were developed from the flow field measurements to be applied to the outputs of the on-board MHPP used in subsequent research projects. These functions will be applied to wind tunnel and flight tests in future research.

REFERENCES

[1] Malcolm A Heath, John D Walshe, and Simon J Watson. Estimating the potential yield of small building-mounted wind turbines. *Wind Energy*, 10(3):271–287, 2007.

[2] Sarah E Stenabaugh, Yumi Iida, Gregory A Kopp, and Panagiota Karava. Wind loads on photovoltaic arrays mounted parallel to sloped roofs on low-rise buildings. *Journal of Wind Engineering and Industrial Aerodynamics*, 139:16–26, 2015.

[3] RP Hoxey and PJ Richards. Flow patterns and pressure field around a full-scale building. *Journal of Wind Engineering and Industrial Aerodynamics*, 50:203–212, 1993.

[4] C White, EW 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.

[5] Simon Watkins, Ee Wei Lim, and Raj Ladani. The velocity field around the top of a tall building; model-scale and full-scale measurements. *6th European and African Conference on Wind Engineering*, 2013.

[6] Bert Blocken. 50 years of computational wind engineering: Past, present and future. *Journal of Wind Engineering and Industrial Aerodynamics*, 129:69–102, 2014.

[7] E.W. Lim, S. Watkins, R. Clothier, R. Ladani, A. Mohamed, and J Palmer. Full-scale flow measurement on a tall building with a continuous-wave doppler lidar anemometer. *Journal of Wind Engineering and Industrial Aerodynamics*, 154:69–75, 2016.

[8] B Canadillas, A Westerhellweg, and T Neumann. Testing the performance of a ground-based wind lidar system: One year intercomparison at the offshore platform fino1. *DEWI Magazine*, 38:58–64, 2011.

[9] Luke Wallace, Arko Lucieer, Christopher Watson, and Darren Turner. Development of a uav-lidar system with application to forest inventory. *Remote Sensing*, 4(6):1519–1543, 2012.

http://www.imavs.org/

[10] Pascal Gohl, Michael Burri, Sammy Omari, Joern Rehder, Janosch Nikolic, Markus Achtelik, and R Siegwart. Towards autonomous mine inspection. In *Applied Robotics for the Power Industry (CARPI), 2014 3rd International Conference on*, pages 1–6. IEEE, 2014.

[11] Javier Moyano Cano. *Quadrotor UAV for wind profile characterization*. PhD thesis, Universidad Carlos III de Madrid. Departamento de Informtica, 2013.

[12] Patrick Neumann, Matthias Bartholmai, Jochen H Schiller, Burkhard Wiggerich, and Manol Manolov. Micro-drone for the characterization and self-optimizing search of hazardous gaseous substance sources: A new approach to determine wind speed and direction. In *Robotic and Sensors Environments (ROSE), 2010 IEEE International Workshop on*, pages 1–6. IEEE, 2010.

[13] 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.

[14] Derrick W Yeo, Nitin Sydney, D Paley, and Donald Sofge. Onboard flow sensing for downwash detection and avoidance with a small rotor helicopter. In *Proc. AIAA Guidance Navigation and Control Conference*, 2015.

[15] Ian de Boisblanc, Nikita Dodbele, Lee Kussmann, Rahul Mukherji, Doug Chestnut, Stephanie Phelps, Gregory C Lewin, and Stephan de Wekker. Designing a hexacopter for the collection of atmospheric flow data. In *Systems and Information Engineering Design Symposium (SIEDS), 2014*, pages 147–152. IEEE, 2014.

[16] A Mohamed. *Phase-Advanced Attitude Sensing and Control For Fixed-Wing Micro Aerial Vehicles in Turbulence*. PhD thesis, RMIT University, 2015.

[17] J Milbank, B Loxton, S Watkins, and WH Melbourne. *Replication of Atmospheric Conditions for the Purpose of Testing MAVs: MAV Flight Environment Project Final Report*. Royal Melbourne Institute of Technology, 2005.

[18] Sridhar Ravi. *The influence of turbulence on a flat plate airfoil at Reynolds numbers relevant to MAVs*. PhD thesis, RMIT University Melbourne, 2011.

[19] Turbulent Flow Instrumentation. Cobra Probe, 2016, http://www.turbulentflow.com.au/Products/CobraProbe/CobraProbe.php.

http://www.imavs.org/

A Monocular Pose Estimation Strategy for UAV Autonomous Navigation in GNSS-denied Environments

Alejandro Rodríguez-Ramos[1], Carlos Sampedro[1], Adrian Carrio[1], Hriday Bavle[1], Ramon A. Suárez Fernández[1], Zorana Milošević[1], and Pascual Campoy[1]

***Abstract*—In this paper, an accurate, efficient, and simple vision-based pose estimation strategy for UAV navigation in GNSS-denied environments is presented. Using visual information and previous knowledge of 3D geometries present in the environment, the pose can be estimated accurately and used for autonomous navigation. The indoor mission in the IMAV 2016 competition has been chosen for developing and evaluating this approach. Three Perspective-n-Point (PnP) algorithms have been tested and benchmarked with the purpose of selecting the most suitable for navigating in this scenario. All of them have been tested in a realistic Gazebo-based simulation using our novel UAV software, Aerostack, which allows for a fully autonomous solution. A complete flight in a GNSS-denied environment has been successfully simulated, indicating that real flights are feasible with this approach.**

http://www.imavs.org/

I. INTRODUCTION

Nowadays, the field of micro aerial vehicles is evolving very rapidly due to its high versatility and the capability for performing complex maneuvers in structured and unstructured environments. Even though a lot of progress has been achieved during the recent years, research efforts are still focused on obtaining fully autonomous systems that can perform high level missions such as the autonomous exploration of a GNSS-denied environment.

Performing such maneuvers in GNSS-denied environments, requires highly accurate position estimation and environment mapping. For this purpose, some approaches are based on active sensors like lasers [1]–[3], or sonars [4], [5]. However these approaches sometimes derivate in heavy weight or high power consumption, affecting agility, endurance, and range of the UAV.

For addressing this problem, some approaches for localization using vision have been proposed, using only the information provided by a single camera [6], [7]–[9], or a stereo vision system [2], [10]–[12]. These kinds of localization strategies provide the advantage of a lightweight system which can be used in indoor as well as outdoor environments. However, using a vision based system for pose estimation and navigation is a very challenging task. For this purpose, some approaches have used visual markers for retrieving the pose of the UAV [13]–[16]. In addition, optical flow based approaches have demonstrated their good capability for indoor navigation with obstacle avoidance [17]–[19].

In this paper, we present a simple and robust monocular pose estimation system for entering a building in a GNSS-denied environment by using the front camera mounted on board the UAV as the main sensor for localization. In the proposed approach, a Computer Vision algorithm has been developed for image detection of the entrances of a building, i.e. door and window, marked with colours. Once the entrances have been detected, a PnP algorithm is used for computing the pose of the UAV based on the information provided by the detection module, and the 3D geometry of door and window in a global map, which is known. We compare the localization accuracy of three different pose estimation algorithms in order to select the most suitable for an specific GNSS-denied flight scenario.

The rest of the paper is structured as follows: Section II presents the problem statement and Section III introduces the proposed methodology. In Section IV the obtained results are described. Finally, Section V concludes the paper and points towards future research directions.

II. PROBLEM STATEMENT

The strategy presented in this paper has been proposed as part of CVG-UPM team solution for participating in The International Micro Air Vehicle Conference and Competition (IMAV[1]), an international reference challenge for UAV autonomous navigation. This yearly event combines a scientific conference with a robotics competition using Micro Air Vehicles (MAVs) and focuses on research that can be applied to real life scenarios. The indoor competition for the edition of 2016 requires one or multiple UAVs to enter a building in a drilling platform to move some important objects to safe locations in a GNSS-denied environment.

In our solution for this part of the competition, we propose a monocular vision approach for pose estimation outside the building. Monocular vision can be used for pose estimation by visually recognising landmarks in the surrounding environment which can serve as navigational aids for pose estimation, providing a cheap, lightweight and low consumption solution. In this case, we propose to use the front camera mounted on

[1] All authors are with the Computer Vision Group, Center for Automation and Robotics (UPM-CSIC), Calle José Gutiérrez Abascal 2, Madrid, Spain.

[1]http://www.imavs.org

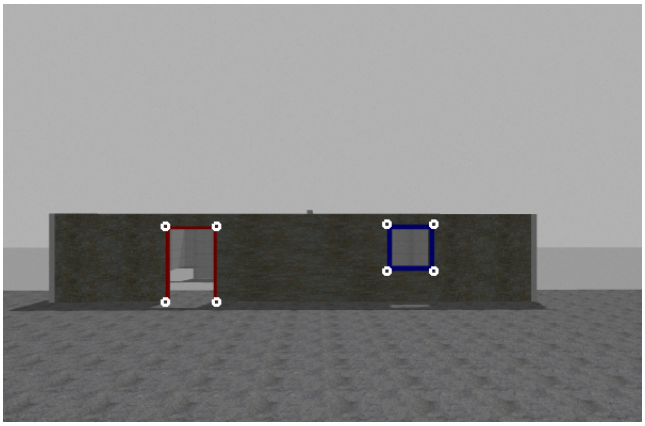


Fig. 1: Detections of door and window on the building façade.

board our UAV to capture RGB images of the building façade, detect door and window corners, whose frame of entrance will be marked with predefined colours for visual aid. The proposed detection strategy is based on two filters:

- Color filter: This filter is based on the rules proposed in the IMAV 16 competition for the entrances to the building. The door of the building has a red colored frame, while the frame of the window is blue. Taking this assumption into account the color filter of the proposed strategy is based on using the Opponent color space for enhancing the red and blue color. The proposed color channels are computed using equations 1 and 2.

$$O_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (1)$$

$$O_2 = \begin{bmatrix} 0 & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2)$$

  The next step is to apply a threshold in the corresponding color channels O_1 and O_2. Finally, an operation for finding contours is applied to the thresholded images in order to remove those contours with a small area.
- Shape filter: This filter is based on the geometry of the objects to be detected. First, an edge detector based on Canny operator is applied. After that, some morphologic operations (e.g. dilation) are applied in order to increase the detected edge contours. Finally, the geometry of the contours is taken into account for removing those which do not fit into a rectangle.

Once door and window corners have been detected, previous knowledge about their 3D geometry (in this case, their relative positions and sizes) can be used to estimate the camera pose and therefore the UAV pose, by using a Perspective-n-Point (PnP) algorithm.

The aim of the Perspective-n-Point problem is to determine the position and orientation of a camera given its intrinsic parameters and a set of n correspondences between 3D points and their 2D projections. In this case, the 3D points are the 3D coordinates of the door and window corners in an arbitrary reference system and their 2D image projections are obtained using the aforementioned detection strategy.

In the following section we present the methodology used to compare the localization accuracy given by three PnP algorithms, in an effort to determine the most suitable one for localization in this specific navigation scenario.

III. PROPOSED METHODOLOGY

In this section, the proposed PnP algorithms to be compared and the test bench structure of this analysis are presented.

A. Analyzed PnP algorithms

A set of three state-of-art PnP algorithms has been tested. As stated, these algorithms are sensitive to the number of points and the distribution of the points over the space. In the following subsection, each algorithm is presented, as well as their benefits and drawbacks in this specific scenario of study.

1) Iterative PnP: This method is based on an iterative solution based on Levenberg-Marquardt optimization. In this case, the function finds the pose of the corresponding object with respect to the camera that minimizes the re-projection error, that is, the sum of squared distances between the observed 2D projections and the projected 3D object points.

2) Robust PnP (RPnP): This algorithm is a robust non-iterative solution of PnP. This method works well for both non-redundant point sets ($n \leq 5$) and redundant point sets. In addition, this algorithm retrieves correct results robustly in the three configurations of points (Ordinary 3D case, Planar case and Quasi-singular case [20]), and its computational complexity grows linearly with n.

In our scenario of study, the points are co-planar and the number of points oscillates between 4 to 8. RPnP is capable of providing a robust and accurate solution in our case of study. Nevertheless, depending on the configuration, a non-iterative solution can lead to a lower accuracy compared to an iterative approach.

3) Efficient PnP (EPnP): [21] This method is based on a non-iterative solution to the PnP problem, in which the n 3D points of the object are expressed as a weighted sum of four virtual control points. The coordinates of these control points in the camera's reference system can be calculated by expressing these coordinates as a weighted sum of the eigenvectors of a 12×12 matrix and solving a small constant number of quadratic equations in order to obtain the right weights. This algorithm is applicable for $n \geq 4$ in planar and non-planar configurations, which make it very suitable for the problem proposed in this paper.

B. Test Bench

The test bench is based on the Aerostack architecture and a Gazebo simulated environment. All three algorithms are evaluated using a common time reference and the same set

http://www.imavs.org/

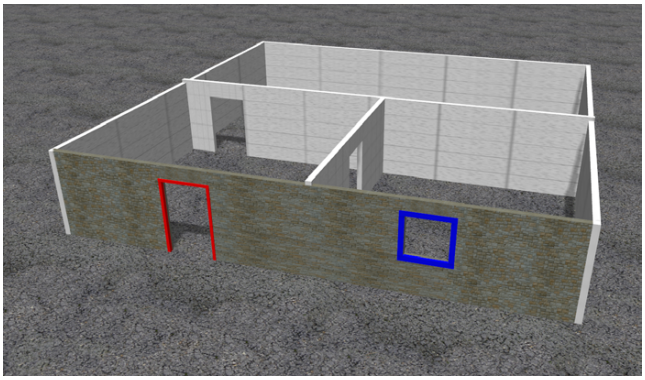


Fig. 2: IMAV'16 house replica, implemented in a Gazebo model.

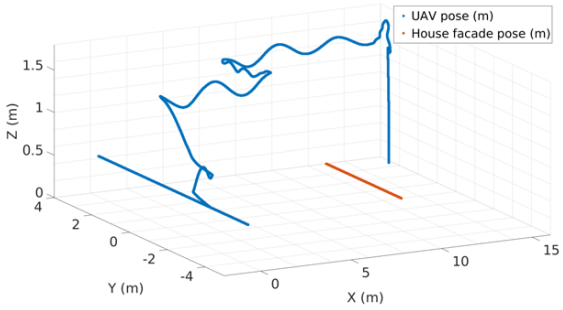


Fig. 4: UAV trajectory in the scenario of study

of 2D and 3D features. In the following subsections, the test bench is elucidated.

1) Aerostack Architecture: Aerostack (http://aerostack.org) is a software implementing fully autonomous navigation solutions for one or more heterogeneous UAVs. It is a versatile, robust and swarm-ready set of ROS packages. The autonomous navigation of the UAV in simulation has been carried out by Aerostack. A full description of Aerostack and its components falls out of the scope of this paper and can be reviewed in [22] and [23].

2) Gazebo Simulation Environment: Gazebo simulator (http://gazebosim.org) is a powerful tool to rapidly test algorithms, design robots, and perform regression testing using realistic scenarios. It provides a versatile set of models and plugins (ROS compatible), as well as an easy-to-use Application Programming Interface (API).

A replica of the IMAV'16 indoor competition set has been carefully designed in Gazebo simulator. All of the sizes of the mission elements, their colours (door and window frames) and their distribution in the scenario have been realistically modelled, in order to accurately test solutions before trying in the real world. This is best depicted in Figure 2.

Furthermore, the physical appearance and dynamics of a Parrot ARDrone 2.0 has been included, as well as different extra sensors (RGB camera, LIDAR, RGB-D camera, IMU and others). The autopilot has been simulated using a Software-In-The-Loop (SITL) simulation, in order to keep tests closer to the real world solution.

3) Simulation details: A flight from the take-off point to an area inside the house has been simulated. With a front RGB simulated camera and the simulated movements of the UAV, a realistic set of images is obtained and used to detect the corners of door and window. At any point of the flight, either the door and/or the window are seen by this camera.

In addition, Gazebo provides ground truth data of the pose of the UAV, which is used for evaluation. The trajectory followed by the UAV is shown in Figure 4.

PnP algorithms provide the pose of an arbitrary frame of reference in which the 3D points are specified with respect to the camera frame of reference. A transformation between this frame and the Gazebo global frame of reference has been taken into account for the PnP pose estimation results to be comparable with the ground truth data.

The position estimated by the three PnP algorithms and the ground truth data have been compared. Mean, standard deviation and maximum and minimum pose estimation errors have been analysed.

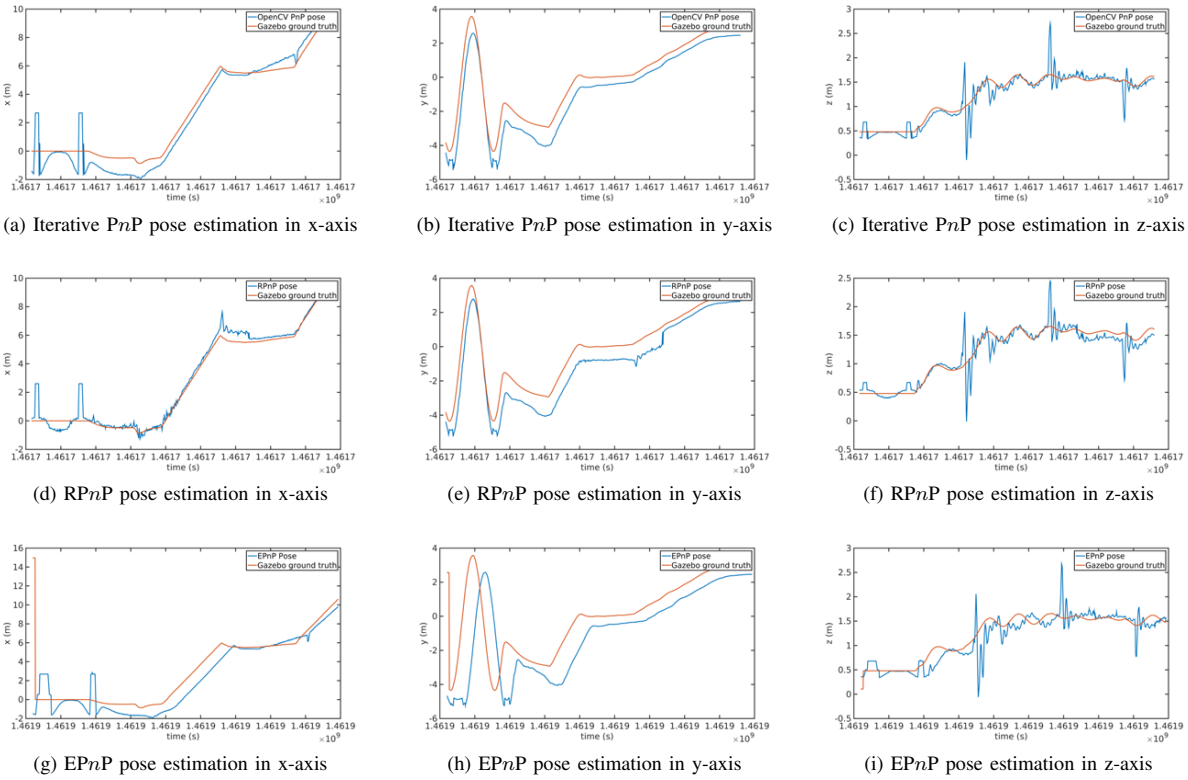
IV. RESULTS AND BENCHMARK

In this section, the benchmarking results are presented. Different error indicators have been considered in order to provide a suitable comparison among the whole set of PnP algorithms.

The difference between the ground truth data and the position estimated by the three PnP algorithms of study is best depicted in Figure 3. The whole set of algorithms shows a proper performance, regarding the ground truth data. The results are not filtered or modified by any offset, so that the PnP estimation remains unaltered. In the z-axis the results remain the closest to the ground truth. This is due to the planar configuration of the case of study, since the points share the same plane and they are perpendicular to the camera plane. This leads to an increased estimation error in x and y axes.

In addition, the absolute error for each PnP position estimation has been depicted in Figure 5. As shown, as the UAV gets closer to the house façade, the error decreases. This is due to the fact that, for a fixed camera resolution, the object size in the image increases and subsequently PnP algorithms can provide a better estimation in translation and rotation. The error in PnP algorithms is a function of the camera distance from the object space. In this planar configuration, where the points of the object space share the same plane, the error does not remain zero. PnP has some error in the estimation, due to different other sources of error, such as errors in the 2D feature detection, estimation, camera calibration, etc. Nevertheless, the error estimation is enough to properly locate the UAV in the whole environment and cross the window or the door for entering the building. This is possible by using an Extended Kalman Filter (EKF) to fuse the data of various on board

http://www.imavs.org/



(a) Iterative PnP pose estimation in x-axis (b) Iterative PnP pose estimation in y-axis (c) Iterative PnP pose estimation in z-axis

(d) RPnP pose estimation in x-axis (e) RPnP pose estimation in y-axis (f) RPnP pose estimation in z-axis

(g) EPnP pose estimation in x-axis (h) EPnP pose estimation in y-axis (i) EPnP pose estimation in z-axis

Fig. 3: Simulation results obtained in the stated experiment. (a), (b), (c), (d), (e), (f), (g), (h), (i) Pose estimation of every algorithm and Gazebo ground truth representation in every axis of the space.

http://www.imavs.org/

sensors, such as an Inertial Measurement Unit (IMU), with the pose estimation obtained using these techniques.

Other indicators, such as Squared Root Mean Quadratic (SRMQ), maximum and minimum position estimation error can be considered. This let the benchmark to be consistent in the comparative analysis. In table I different error indicators are best enumerated.

TABLE I: Benchmark of the set of PnP algorithms.

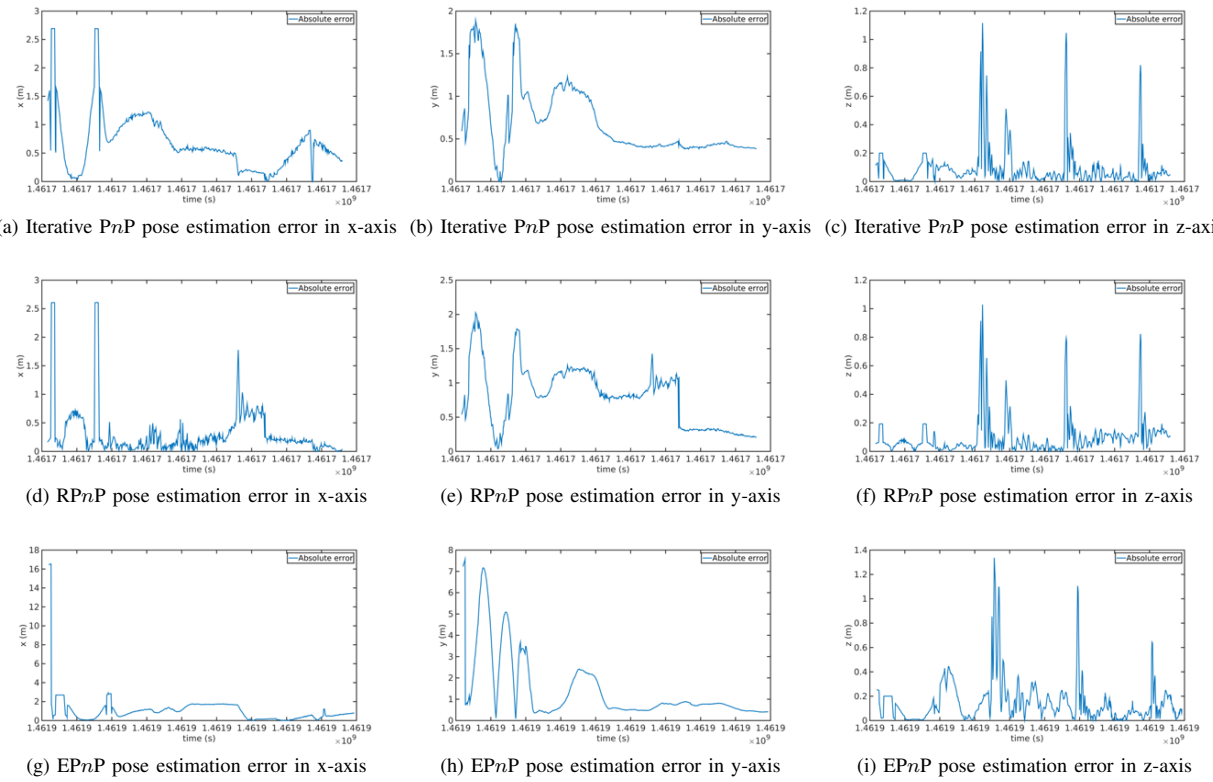
Algorithm	SRMQ (m)	Max (m)	Min (m)	PO
OpenCV PnP	0.6571	2.696 (x-axis)	5E-4 (z-axis)	No
RPnP	0.3711	2.6081 (x-axis)	4E-5 (z-axis)	No
EPnP	2.4342	16.5752 (x-axis)	3E-05 (z-axis)	No

The indicators for both Iterative PnP and RPnP resulted in being really similar, although RPnP shows slightly better performance. However, the error is acceptable, especially when filtering this data and fusing it with data from other sensors. Moreover, EPnP shows poorer performance, but still valid for this application. It can be observed that none of these algorithms implement internally a prediction and optimization post process (PO). This would lead to a finer performance with more complex objects (more points spread in the object space) and does not require any other feeds of pose estimation. The RPnP algorithm performed the best in the scenario of study. Further optimizations can be made in order to decrease the stationary error.

These three state-of-art PnP algorithms have demonstrated to provide a proper performance for monocular pose estimation in GNSS-denied environments. Each of them combined with sensor fusion techniques, such as EKF, provide a complete solution for pose estimation, taking advantage of known geometries of the surrounding environment. The benchmark has revealed that the error is acceptable for real applications, with RPnP method remaining closely the best.

V. CONCLUSIONS AND FUTURE WORK

Indoor and GNSS-denied environment navigation is an extremely challenging task. To achieve this goal inexpensively without extra sensors, monocular pose estimation algorithms can be used. PnP pose estimation is based on previous knowledge of 3D geometries in the scenario for localization. Three PnP algorithms has been tested and simulated. The simulation provided proper results, showing that an autonomous entrance to an indoor environment is completely feasible with this approach. PnP has some error in the estimation, due to different other sources of error but is accurate enough to provide a precise localization at real-time frame rates. Furthermore, these results can be greatly improved when combined with other filtering and state estimation techniques.



(a) Iterative PnP pose estimation error in x-axis (b) Iterative PnP pose estimation error in y-axis (c) Iterative PnP pose estimation error in z-axis

(d) RPnP pose estimation error in x-axis (e) RPnP pose estimation error in y-axis (f) RPnP pose estimation error in z-axis

(g) EPnP pose estimation error in x-axis (h) EPnP pose estimation error in y-axis (i) EPnP pose estimation error in z-axis

Fig. 5: Simulation results obtained in the stated experiment. (a), (b), (c), (d), (e), (f), (g), (h), (i) Absolute estimation error for each PnP algorithm of study.

http://www.imavs.org/

Improvements in the detection algorithm are under study, such as making it more versatile against other types of geometries. Also, other PnP algorithms, specific for planar case, should be included, benchmarked and optimized following this work flow. Finally, real flights are required to validate our algorithm, even though results have been obtained with a highly realistic simulation.

ACKNOWLEDGMENT

This work has been developed thanks to the financial support of the Spanish Ministry of Economy and Competitiveness in the project MICYT DPI2014-60139-R.

REFERENCES

[1] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a gps-denied environment," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1814–1820.

[2] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments," in *SPIE Defense, Security, and Sensing*. International Society for Optics and Photonics, 2009, pp. 733 219–733 219.

[3] S. Huh, D. H. Shim, and J. Kim, "Integrated navigation system using camera and gimbaled laser scanner for indoor and outdoor autonomous flight of uavs," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3158–3163.

[4] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 376–382, 1991.

[5] J. F. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, "Quadrotor using minimal sensing for autonomous indoor flight," in *European Micro Air Vehicle Conference and Flight Competition (EMAV2007)*, no. LIS-CONF-2007-006, 2007.

[6] A. D. Wu, E. N. Johnson, M. Kaess, F. Dellaert, and G. Chowdhary, "Autonomous flight in gps-denied environments using monocular vision and inertial sensors," *Journal of Aerospace Information Systems*, 2013.

[7] G. P. Tournier, M. Valenti, J. P. How, and E. Feron, "Estimation and control of a quadrotor vehicle using monocular vision and moire patterns," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2006, pp. 21–24.

[8] C. Bills, J. Chen, and A. Saxena, "Autonomous mav flight in indoor environments using single image perspective cues," in *Robotics and automation (ICRA), 2011 IEEE international conference on*. IEEE, 2011, pp. 5776–5783.

[9] S. Ahrens, D. Levine, G. Andrews, and J. P. How, "Vision-based guidance and control of a hovering vehicle in unknown, gps-denied environments," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2643–2648.

[10] C. Fu, A. Carrio, and P. 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*. IEEE, 2015, pp. 957–962.

[11] T. Tomić, K. Schmid, P. Lutz, A. Dömel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 3, pp. 46–56, 2012.

[12] L. R. G. Carrillo, A. E. D. López, R. Lozano, and C. Pégard, "Combining

stereo vision and inertial navigation system for a quad-rotor uav," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 373–387, 2012.

[13] J. L. Sanchez-Lopez, J. Pestana, P. de la Puente, A. Carrio, and P. Campoy, "Visual quadrotor swarm for the imav 2013 indoor competition," in *ROBOT2013: First Iberian Robotics Conference*. Springer, 2014, pp. 55–63.

[14] J. L. Sanchez-Lopez, J. Pestana, J.-F. Collumeau, R. Suarez-Fernandez, P. Campoy, and M. Molina, "A vision based aerial robot solution for the mission 7 of the international aerial robotics competition," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*. IEEE, 2015, pp. 1391–1400.

[15] P. Rudol, M. Wzorek, and P. Doherty, "Vision-based pose estimation for autonomous indoor navigation of micro-scale unmanned aircraft systems," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1913–1920.

[16] A. Amor-Martinez, A. Ruiz, F. Moreno-Noguer, and A. Sanfeliu, "On-board real-time pose estimation for uavs using deformable visual contour registration," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2595–2601.

[17] J.-C. Zufferey and D. Floreano, "Fly-inspired visual steering of an ultralight indoor aircraft," *Robotics, IEEE Transactions on*, vol. 22, no. 1, pp. 137–146, 2006.

[18] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, "Mav navigation through indoor corridors using optical flow," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3361–3368.

[19] F. Ruffier and N. Franceschini, "Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind reaction," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 3. IEEE, 2004, pp. 2339–2346.

[20] S. Li, C. Xu, and M. Xie, "A robust o (n) solution to the perspective-n-point problem," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 7, pp. 1444–1450, 2012.

[21] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, pp. 155–166, 2009.

[22] J. L. Sanchez-Lopez, J. Pestana, P. de la Puente, and P. Campoy, "A reliable open-source system architecture for the fast designing and prototyping of autonomous multi-uav systems: Simulation and experimentation," *Journal of Intelligent & Robotic Systems*, pp. 1–19, 2015.

[23] J. Sanchez-Lopez, R. Suarez-Fernandez, H. Bavle, C. Sampedro, J. Pestana, M. Molina, and P. Campoy, "Aerostack: An architecture and open-source software framework for aerial robotics," in *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, June 2016, p. 0.

http://www.imavs.org/

A Preliminary Modeling and Control Framework for a Hybrid UAV J-Lion

Yijie Ke*, Kangli Wang, and Ben M. Chen
Unmanned System Research Group,
Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117582

http://www.imavs.org/

ABSTRACT

J-Lion is a tail-sitter UAV platform developed to perform both VTOL and cruise flight missions. This paper presents a preliminary modeling and control framework for our hybrid UAV J-Lion. A unified model structure including comprehensive model components is derived for full envelop flight conditions. Currently, model-based controller has been specifically designed for V-TOL mode that can handle large angle deviations. Our method is verified by outdoor flight tests with existence of strong wind gust.

NOMENCLATURE

δ_p	Vectoring thrust angle in pitch channel. Positive value means that vectoring thrust propellers produce positive pitch torque
$\delta_{\text{fin1}}, \delta_{\text{fin2}}$	Deflection angles of two fins. Positive value means that fin deflection produces positive pitch torque
ω_1, ω_2	Rotating speed of two propellers
ϕ, θ, ψ	Euler angle
ρ	Air density
$\boldsymbol{\Omega}_b$	Vector of angular velocities expressed in body frame
$\boldsymbol{F}_{\text{aero}}$	Aerodynamic force expressed in inertial frame
$\boldsymbol{F}_{\text{fin\_b}}$	Drag force vector from fins expressed in body frame
$\boldsymbol{F}_{\text{surface}}$	Forces from control surfaces and fins expressed in inertial frame
$\boldsymbol{M}_{\text{aero}}$	Aerodynamic torque expressed in body frame
$\boldsymbol{M}_{\text{prop\_gyro}}$	Gyroscopic torque from rotating propellers expressed in body frame
$\boldsymbol{M}_{\text{prop}}$	Torque from rotating propellers expressed in body frame
$\boldsymbol{M}_{\text{surface}}$	Torque from control surfaces and fins expressed in body frame
$\boldsymbol{M}_{\text{vectorT}}$	Torque from vectoring thrust effect expressed in body frame
$\boldsymbol{R}_{\text{BE}}$	Rotation matrix from body frame to inertial frame
$\boldsymbol{r}_{\text{prop}}$	Vector from CG to center of propeller center, expressed in body frame
$\boldsymbol{R}_{\text{RB}}$	Rotation matrix from rotor frame to body frame
A_{prop}	Area of propeller actuator disk
AR_{fin}	Aspect ratio of two fins
e	Oswald efficiency factor, usually set to be between 0.7 and 0.85 [6]
g	Gravity acceleration
I_f	Moment of inertia matrix in diagonal form
$J_{\text{prop1}}, J_{\text{prop2}}$	Moment of inertial of two propellers
$M_{\text{prop1}}, M_{\text{prop2}}$	Torque from two rotating propellers
p, q, r	ngular velocity along three body axes
q_0, q_1, q_2, q_3	Quaternion elements, where q_0 is scalar part, and (q_1, q_2, q_3) is vector part
r_{fin}	Distance between CG and fin center in y_b axis direction
$Re_{\text{fin}i}$	Local Reynolds number of two fins in slipstream
S_{fin}	Single fin area
T	Total thrust from two propellers
u, v, w	Velocities expressed in inertial frame
V_∞	Axial inflow speed for two propellers
$V_{\text{slip1}}, V_{\text{slip2}}$	Slipstream velocity for two fins

*Email address: keyijie@u.nus.edu

http://www.imavs.org/

1 INTRODUCTION

Hybrid UAV, which combines the advantages of VTOL and cruise flight capabilities, is a design concept that can meet the requirement of missions in both open and clustered environments, and it's an attractive solution for applications with multiple flight envelopes that cannot rely merely on fixed-wing or rotorcraft UAVs. Design examples can be referred in [1–3].

In the view of system integration, development of a hybrid UAV platform is an iterative multidisciplinary process consisting of mechanical design, aerodynamics, system-on-chip, software and algorithm design. And the modeling and control framework plays a key role in the development cycle in order to modularly integrate various design elements such as modeling components, advanced flight control algorithms, and onboard flight health monitoring, etc.

Therefore, this paper will present a preliminary modeling and control framework for hybrid UAVs. A unified model structure will be derived to account for full AOA (angle of attack) envelop flight dynamics. Control design can then be carried out based on those model components in different flight conditions.

The paper is organized as followed: Section 2 will introduce briefly our hybrid UAV platform J-Lion. Section 3 will give the derivation of unified model structure. Section 4 will specifically cover the control design for VTOL mode based on proposed model structure. Implementation and results will be presented in Section 5 to verify our method. Conclusions are finally made in Section 6 with future work illustration.

2 PLATFORM

J-Lion is a hybrid UAV with tail-sitter configuration developed in NUS. The design methodology follows the approach in previous work [4]. The picture of real platform is shown in figure 1, and main components are illustrated below:

(1) Propeller:

Two propellers driven by brushless DC motors provide the thrust for J-Lion, and vectoring thrust is realized by using one DOF (degree of freedom) gimbal mechanism. The propellers can also provide pitch torque and differential thrust for pitch and roll stabilization.

(2) Fin:

Two fins are designed in the slipstream of propellers to enable yaw control in VTOL mode, and also enhance pitch control during transition.

(3) Control surfaces:

Same with conventional fixed-wing UAVs, elevator, aileron and rudder control surfaces are designed.

(4) Electronics:

Pixhawk autopilot is chosen as our main electronics device, together with various servos, GPS, and other sensors such as airspeed sensor. Note that the airspeed sensor data is utilized for transition and cruise flight only.

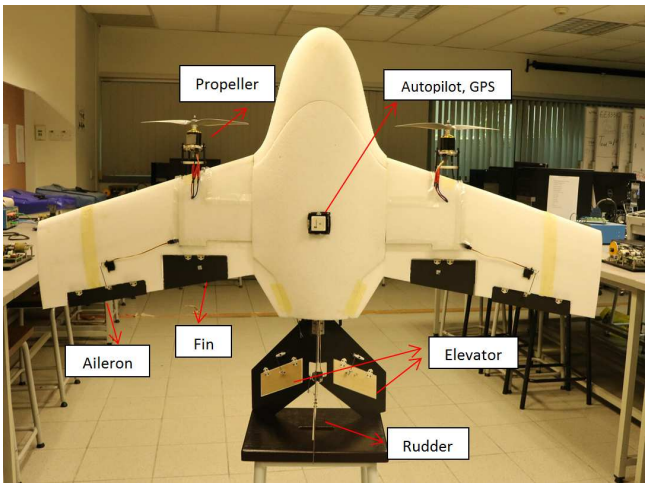


Figure 1: J-Lion platform

3 MODEL STRUCTURE

A unified model structure is advantageous in the sense that the structure is tailorable depending on flight conditions. For example, the fins provide yaw control torque during VTOL mode by deflecting in opposite directions, while they can also provide torque for pitch control in transition by deflecting in the same direction. A unified model can greatly ease mode-switching difficulties because flight modes sometimes have no clear boundaries during flight maneuvers.

Our model structure consists of three parts: kinematics equation, force equation and moment equation. Various components can be integrated into the structure based on whether they are in our interests or not, such as vectoring thrust torque, aerodynamic lift and drag in fixed-wing mode, etc.

3.1 Coordinate frames

Shown in figure 2 , three coordinate frames are defined.

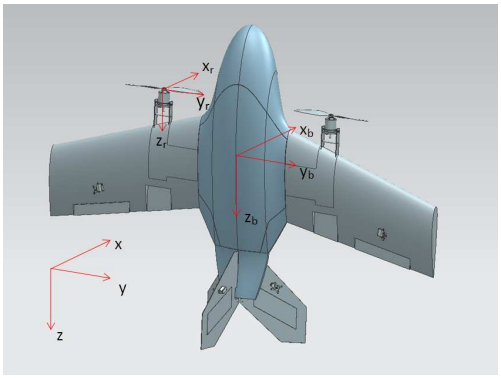


Figure 2: Coordinate frames

(1) Global inertial frame x, y, z is set to be NED frame.

(2) Body frame x_b, y_b, z_b is defined with its origin located at center of gravity (CG), and y_b, z_b direction pointing to its wingtip and tail respectively.

(3) Rotor frame x_r, y_r, z_r is also defined to describe the vectoring thrust direction. It is aligned with the body frame when no vectoring thrust angle is applied. The origin of rotor plane frame is located at the center of vectoring thrust mechanism.

3.2 Kinematics equation

The kinematics equation can be represented in Euler angle, rotation matrix or quaternion. Choosing which form to use depends on the control algorithm we'd like to use. Note that Euler angle form has intrinsic singularity problem.

1. Euler angle form

$$\begin{pmatrix}\dot{\phi}\\ \theta\\ \psi\end{pmatrix} = \begin{pmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & \text{-}\sin\phi\\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta\end{pmatrix}\begin{pmatrix}p\\ q\\ r\end{pmatrix} \quad (1)$$

2. Rotation matrix form

   The following strapdown equation holds:

$$\dot{\boldsymbol{R}}_{\text{BE}} = \boldsymbol{R}_{\text{BE}}\boldsymbol{\Omega}(\boldsymbol{\omega}) \quad (2)$$

   where $\Omega(\boldsymbol{\omega})$ is the skew symmetric matrix using angular velocity vector in body frame.

3. Quaternion form The differential form of quaternion is:

$$\begin{pmatrix}\dot{q_0}\\ q_1\\ q_2\\ q_3\end{pmatrix} = -\frac{1}{2}\begin{pmatrix}0 & p & q & r\\ -p & 0 & -r & q\\ -q & r & 0 & -p\\ -r & -q & p & 0\end{pmatrix}\begin{pmatrix}q_0\\ q_1\\ q_2\\ q_3\end{pmatrix} = -\frac{1}{2}\boldsymbol{\Omega_q}\mathbf{q} \quad (3)$$

3.3 Force equation

Let the position be $\boldsymbol{p} = (x, y, z)^{\text{T}}$, velocity be $\boldsymbol{v} = (u, v, w)$, the force equation is:

$$\begin{aligned}\begin{pmatrix}\dot{x}\\ y\\ z\end{pmatrix} &= \begin{pmatrix}u\\ v\\ w\end{pmatrix}\\ \begin{pmatrix}\dot{u}\\ v\\ w\end{pmatrix} &= \begin{pmatrix}0\\ 0\\ g\end{pmatrix} + \frac{1}{m}\boldsymbol{R}_{\text{BE}}\boldsymbol{R}_{\text{RB}}\begin{pmatrix}0\\ 0\\ -T\end{pmatrix} + \frac{1}{m}\boldsymbol{F}_{\text{surface}} + \frac{1}{m}\boldsymbol{F}_{\text{aero}}\end{aligned}$$

where $\boldsymbol{F}_{\text{surface}}$ refers to the forces produced by control surfaces and fins expressed in inertial frame, and the magnitude of the force is usually much smaller than others, and $\boldsymbol{F}_{\text{aero}}$ refers to the aerodynamic forces produced by the relative motion between UAV and surrounding air. In fixed-wing mode, $\boldsymbol{F}_{\text{aero}}$ is the main term in force equation.

The compact form is:

$$\begin{aligned}\dot{\boldsymbol{p}} &= \boldsymbol{v}\\ \dot{\boldsymbol{v}} &= g\boldsymbol{e_3} - \frac{1}{m}\boldsymbol{R}_{\text{BE}}\boldsymbol{R}_{\text{RB}}T\boldsymbol{e_3} + \frac{1}{m}\boldsymbol{F}_{\text{surface}} + \frac{1}{m}\boldsymbol{F}_{\text{aero}} \quad (4)\end{aligned}$$

where $\boldsymbol{e_3} = (0, 0, 1)^{\text{T}}$.

At current stage, we assume that two propellers have the same vectoring angle during flight, then R_{RB} is expressed below:

$$\boldsymbol{R}_{\text{RB}} = \begin{pmatrix}\cos\delta_p & 0 & \sin\delta_p\\ 0 & 1 & 0\\ -\sin\delta_p & 0 & \cos\delta_p\end{pmatrix}$$

3.4 Moment equation

Based on D'Alembert-Lagrange equation, the compact form of moment equation in body frame can be formulated as:

$$\begin{aligned}\boldsymbol{I_f}\dot{\boldsymbol{\Omega}}_b = -\boldsymbol{\Omega}_b \times (\boldsymbol{I_f}\boldsymbol{\Omega_b}) + \boldsymbol{M}_{\text{prop\_gyro}} + \boldsymbol{M}_{\text{surface}} + \boldsymbol{M}_{\text{prop}}\\ + \boldsymbol{M}_{\text{vectorT}} + \boldsymbol{M}_{\text{aero}} \quad (5)\end{aligned}$$

$\boldsymbol{M}_{\text{prop\_gyro}}$ is the gyroscopic torque from two propellers and the magnitude is small. $\boldsymbol{M}_{\text{surface}}$ is the torque produced by control surfaces and fins. In VTOL mode, the main component comes from fins for yaw control. In fixed-wing mode, all torques produced by control surfaces should be taken into account. $\boldsymbol{M}_{\text{prop}}$ is the torque produced by propellers which is effective in all flight conditions. $\boldsymbol{M}_{\text{vectorT}}$ is the torque from vectoring thrust which is also effective all the time. $\boldsymbol{M}_{\text{aero}}$ is the aerodynamic torque which is significant during transition and fixed-wing flight.

The expressions of those components consistent through full envelope flight are derived as:

$$\begin{aligned}\boldsymbol{M}_{\text{prop\_gyro}} &= -\begin{pmatrix}p\\ q\\ r\end{pmatrix} \times \left[\boldsymbol{R}_{\text{RB}}\begin{pmatrix}0\\ 0\\ J_{prop1}\omega_1 - J_{prop2}\omega_2\end{pmatrix}\right]\\ \boldsymbol{M}_{\text{prop}} &= \boldsymbol{R}_{\text{RB}}\begin{pmatrix}0\\ 0\\ M_{\text{prop2}} + J_{\text{prop2}}\dot{\omega_2} - M_{\text{prop1}} - J_{\text{prop1}}\dot{\omega_1}\end{pmatrix}\\ \boldsymbol{M}_{\text{vectorT}} &= \boldsymbol{r}_{\text{prop}} \times \left[\boldsymbol{R_{RB}}\begin{pmatrix}0\\ 0\\ -T\end{pmatrix}\right]\end{aligned}$$

4 CONTROL DESIGN IN VTOL MODE

Based on the model structure in previous section, we can perform control design with the consideration of corresponding model components for various flight conditions. Here, we will design specifically for VTOL mode with rotation matrix representation. First, the model components effective in V-TOL mode will be derived. Position control and attitude control layers are then designed respectively as outer-loop and inner-loop.

http://www.imavs.org/

http://www.imavs.org/

4.1 Model components in VTOL mode

Referring to force and moment equations 4 and 5, the model components in VTOL mode are:

1. Total thrust from propellers T

$$\begin{aligned} T &= T_1 + T_2 \\ &= a_t\omega_1^2 + b_t + a_t\omega_2^2 + b_t, \end{aligned} \tag{6}$$

where a_t, b_t are propeller thrust coefficients that can be determined by experiment.

2. Forces from control surfaces and fins $\boldsymbol{F}_{\text{surface}}$

The main component of $\boldsymbol{F}_{\text{surface}}$ is the drag force from two fins, hence is with the form below:

$$\boldsymbol{F}_{\text{surface}} = \boldsymbol{R}_{\text{BE}} \begin{pmatrix} 0 \\ 0 \\ F_{\text{fin\_b}} \end{pmatrix}$$

Using propeller momentum theory [5], the drag force $F_{\text{fin\_b}}$ in body frame is:

$$F_{\text{fin\_b}} = \frac{1}{2}\rho V_{\text{slip1}}^2 S_{\text{fin}} C_{\text{Dfin1}} + \frac{1}{2}\rho V_{\text{slip2}}^2 S_{\text{fin}} C_{\text{Dfin2}} \tag{7}$$

where

$$\begin{aligned} V_{\text{slip}i} &= \sqrt{V_\infty^2 + \frac{2T_i}{\rho A_{\text{prop}}}} \\ C_{\text{Dfin}i} &= 2 * 0.074 Re_{\text{fin}i}^{-1/5} + \frac{C_{\text{Lfin}i}^2}{\pi e A R_{\text{fin}}} \\ C_{\text{Lfin}i} &= 2\pi|\delta_{\text{fin}i}| \end{aligned}$$

for $i = 1, 2$.

Note that $C_{\text{Lfin}i}, C_{\text{Dfin}i}$ are estimated using flat plate assumption, and the drag force consists of friction drag and induced drag [6]. The propeller inflow velocity V_∞ is assumed to be zero in VTOL mode.

3. Torque from control surfaces and fins $\boldsymbol{M}_{\text{surface}}$

The main component of $\boldsymbol{M}_{\text{surface}}$ is the torque produced by two fins for yaw control, hence it's with the form:

$$\boldsymbol{M}_{\text{fin}} = \begin{pmatrix} 0 \\ 0 \\ M(\delta_{fin}) \end{pmatrix}$$

We assume here that the deflection angles of two fins are the same but in opposite directions, then the $M(\delta_{fin})$ is computed as:

$$M(\delta_{fin}) = \pi\delta_{\text{fin}}\rho(V_{\text{slip1}}^2 + V_{\text{slip2}}^2)S_{\text{fin}}r_{\text{fin}} \tag{8}$$

where δ_{fin} is the fin deflection angle following torque positive sign convention.

4.2 Outer-loop design

The position control layer consists of two parts. Firstly, the velocity reference is obtained using P-control scheme with position error:

$$\boldsymbol{v_{ref}} = K_{p1\_p}\boldsymbol{e_p} = K_{p1\_p}(\boldsymbol{p_{ref}} - \boldsymbol{p}) \tag{9}$$

where K_{p1_p} is the control gain, $\boldsymbol{p_{ref}}$ is the reference position.

Then desired thrust vector can be obtained in the second part with velocity information:

$$\boldsymbol{T_d} = K_{p2\_v}\boldsymbol{e_v} + K_{d2\_v}\frac{d}{dt}\boldsymbol{e_v} + K_{i2\_v}\int \boldsymbol{e_v}dt \tag{10}$$

where $K_{p2\_v}, K_{d2\_v}, K_{i2\_v}$ are velocity control gains, $\boldsymbol{e_v}$ is the velocity error vector.

Due to thrust saturation, the desired thrust is scaled within feasible range:

$$\boldsymbol{T_{dnew}} = \frac{\boldsymbol{T_d}}{|\boldsymbol{T_d}|}sat(|\boldsymbol{T_d}|, T_{max}) \tag{11}$$

where T_{max} is the maximum thrust available from propellers, $sat()$ is the saturation function.

With the desired thrust vector, we can achieve the desired $\boldsymbol{z_b}$ vector which is aligned with $\boldsymbol{T_d}$. The desired $\boldsymbol{y_{bd}}$ vector is computed using desired yaw angle ψ_d as:

$$\boldsymbol{y_b} = (-\sin\psi_d, \cos\psi_d, 0)^{\text{T}} \tag{12}$$

Thus the rotation matrix for desired attitude can be constructed based on desired body-axis vectors. The rotation matrix will be passed to inner-loop for attitude control.

4.3 Inner-loop design

The attitude control adopts the method from [7, 8], and the control scheme consists of two loops. Loop 1 is a simple P control loop to generate the attitude rate reference, while Loop 2 uses PID control law to generate required control torque in three body axes.

• Loop 1

In Loop 1, there are two routes that account for large and small angle deviations. The Z body axis vector is obtained from the third column of rotation matrix, which is expressed in inertial frame. The control algorithm itself follows the Euler sequence used in [7] which is roll-pitch-yaw sequence.

When the deviation angle between current $\boldsymbol{z_b}$ and desired $\boldsymbol{z_b}$ is less than 90 degrees, called ROUTE 1, the algorithm will first compute roll and pitch angle error because yaw channel response is slower. The roll and pitch motion can be seen as a rotation that can transform current Z body axis $\boldsymbol{Z_b}$ to desired Z body axis $\boldsymbol{Z_{br}}$, and the minimum distance movement is determined by the coplar of $\boldsymbol{Z_b}$ and $\boldsymbol{Z_{br}}$. Hence, the euler axis $\vec{n}$ and rotation angle θ can be obtained as:

$$\vec{n} = \frac{\boldsymbol{Z}_b \times \boldsymbol{Z}_{br}}{|\ \boldsymbol{Z}_b \times \boldsymbol{Z}_{br}\ |}$$

$$\theta = atan2(\boldsymbol{Z}_b \times \boldsymbol{Z}_{br}, \boldsymbol{Z}_b \cdot \boldsymbol{Z}_{br})$$

Then the intermediate rotation matrix is expressed as:

$$\boldsymbol{R_{rp}} = \boldsymbol{R(\theta, \vec{n})} \cdot \boldsymbol{R}$$

where $R(\theta, \vec{n})$ is the rotation matrix constructed from $\vec{n}$ andθ, R is current rotation matrix.

Roll and pitch angle errors are computed by projecting the rotation angle θ to current x_b, y_b axes, and yaw angle error is less weighted and computed with the deviation between R_{rp} and R_{ref}.

Similarly, when the absolute value of deviation angle between current $\boldsymbol{z_b}$ and desired $\boldsymbol{z_b}$ is greater than 90 degrees (ROUTE 2), the algorithm computes all three angle errors simultaneously using the projection of θ to current body axes.

The angle errors are finally computed by fusing the results from ROUTE 1 and 2 together, and the angular rate references can then be computed using a simple P control scheme.

• Loop 2

With the angular rate obtained from Loop 1, PID control scheme can be used to obtain desired control torque in three body axes. Using pitch channel as an example, the desired torque in pitch channel can be computed as:

$$M_{pitch} = K_p \boldsymbol{e_{\dot{\theta}}} + K_d \frac{d}{dt} \boldsymbol{e_{\dot{\theta}}} + K_i \int \boldsymbol{e_{\dot{\theta}}} dt \qquad (13)$$

where $\boldsymbol{e_{\dot{\theta}}} = \dot{\theta}_d - \dot{\theta}$ is the angular rate error.

Note that the gains in Loop 1 and 2 can be determined with typical LQR method for certain performance requirement, and also be tuned to satisfy the torque limit from actuators expressed in moment equation 5.

5 IMPLEMENTATION AND RESULTS

5.1 *Implementation*

Our modeling and control framework is implemented in opensource PX4 software environment. The main modules involved are shown in figure 3.

Except the outer-loop and inner-loop layers, navigator module manages flight mission control, and provides position reference. In the actuator layer, the actuator inputs are calculated with actuator models, for example, throttle input for motor ESC (electrical speed controller) is computed with fitting relationship of rotating speed to throttle that is determined from experiment, and deflection of fins is computed using equation 8 to realize desired yaw control torque.

5.2 *Flight results*

Both semi-auto and full-auto flights in VTOL mode were conducted for J-Lion with proposed framework. In semi-auto flight, attitude references are generated by human pilot with a transmitter, and only inner-loop controller is activated. In full-auto flight, J-Lion is required to perform fully autonomous flight through several waypoints.

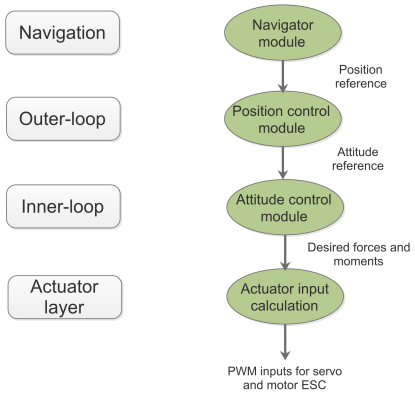


Figure 3: Framework implementation schematics

(1) Semi-auto test

The semi-auto test was conducted in indoor environment. Attitudes are plotted via their references, as shown in figures 4. From the results, we can see that the attitude stabilization of J-Lion in VTOL is well achieved despite of the offset from installation and assembly.

(2) Full-auto test

The full-auto test was conducted in outdoor environment with existence of strong wind. J-Lion was required to fly through four waypoints in VTOL mode. The four waypoints are set with the same altitude. When the UAV reaches one waypoint, the yaw reference will be set in the direction pointing to the next. The attitude curves are shown in figure 5. From the figure, the maximum pitch angle can reach up to 50 deg to counter strong wind gust in pitch channel. The flight trajectory of relative horizontal position is also shown in figure 6. It's observed that the tracking performance in VTOL mode is greatly affected by strong outdoor wind gust, but the attitude stabilization is still satisfactory. The position control performance can be improved by integrating wind gust observer in the future.

6 CONCLUSIONS AND FUTURE WORK

A preliminary modeling and control framework is proposed in this paper for our hybrid UAV J-Lion. The unified model structure can integrate full envelope flight dynamics which is usually derived separately in different flight modes, and it can serve as the basis for high nonlinear dynamics modeling in transition phase. Besides, model-based control design for VTOL mode is presented based on the proposed model structure. Our framework is verified by real flight tests. In the future, we will integrate the transition model into our model structure, and design unified flight control laws for all flight modes.

http://www.imavs.org/

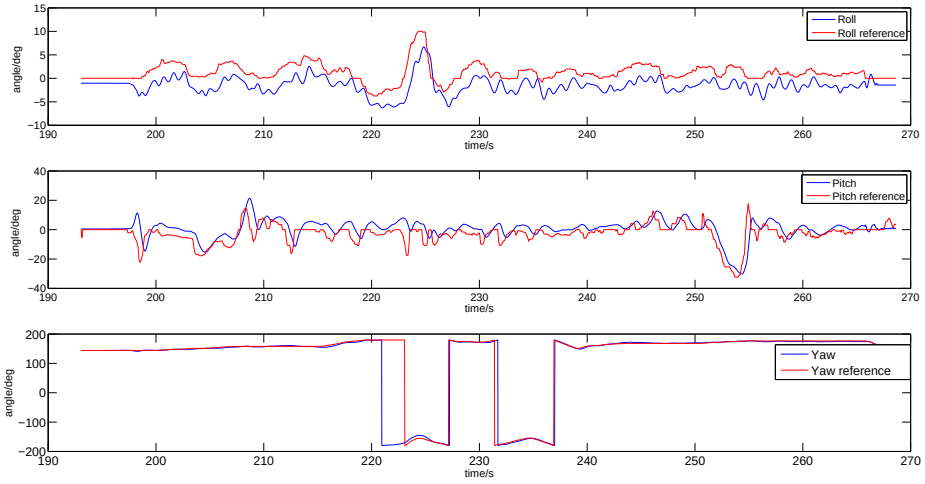


Figure 4: Attitude curve in semi-auto test

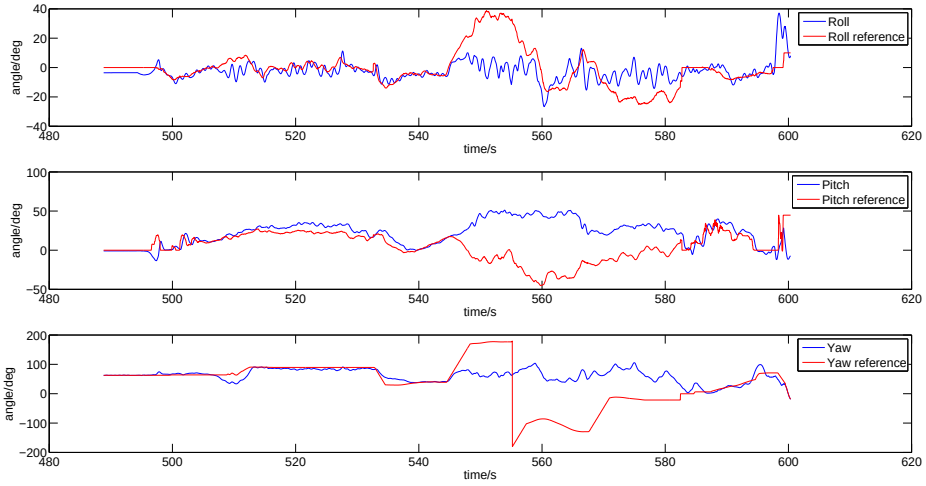


Figure 5: Attitude curve in full-auto test

http://www.imavs.org/

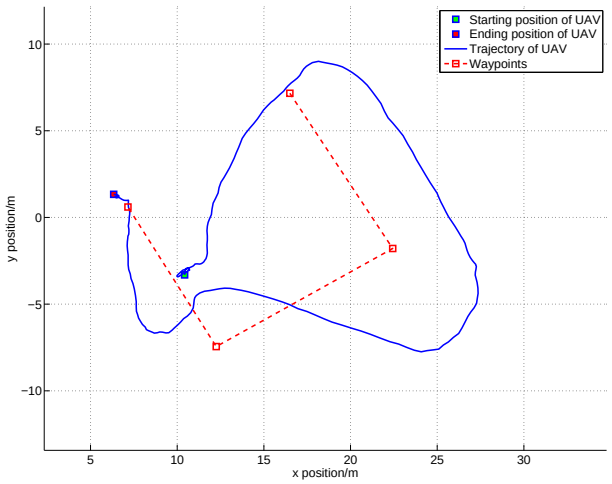


Figure 6: Flight trajectory of relative horizontal position in full-auto test

http://www.imavs.org/

REFERENCES

[1] Kendoul, Farid, Isabelle Fantoni, and Rogelio Lozano. Modeling and control of a small autonomous aircraft having two tilting rotors. *Proceedings of the 44th IEEE Conference on Decision and Control*, 2005.

[2] Cetinsoy, E., et al., Design and construction of a novel quad tilt-wing UAV. *Mechatronics*, 22(6): p. 723-745, 2012.

[3] Stone, R. Hugh. Control architecture for a tail-sitter unmanned air vehicle. *Control Conference, 2004. 5th Asian. IEEE*, 2004.

[4] Yijie Ke, Hui Yu, Changkai Chi, Mingwei Yue and Ben M. Chen. A systematic design approach for an unconventional UAV J-Lion with extensible morphing wings. *Proceedings of the 12th IEEE International Conference on Control & Automation*, 2016.

[5] McCormick, B. W. *Aerodynamics, Aeronautics, and Flight Mechanics*. John Wiley and Sons, Inc, New York, 1995.

[6] Anderson Jr, John David. *Fundamentals of aerodynamics*. Tata McGraw-Hill Education, 2010.

[7] Mellinger, Daniel, and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. *In IEEE Conference on Robotics and Automation (ICRA)*, 2011.

[8] Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. PIXHAWK: A system for autonomous flight using onboard computer vision. *In IEEE Conference on Robotics and Automation (ICRA)*, 2011.

Adaptive Path Planning for a Vision-Based quadrotor in an Obstacle Field

J. Junell*, and E. van Kampen†
Delft University of Technology Aerospace Engineering, Control and Simulation
Delft, The Netherlands.

http://www.imavs.org/

ABSTRACT

This paper demonstrates a real life approach for quadrotor obstacle avoidance in indoor flight. A color-based vision approach for obstacle detection is used to good effect conjointly with an adaptive path planning algorithm. The presented task is to move about a set indoor space while avoiding randomly located obstacles and adapting a path to prevent future confrontation with the obstacles all together. The goal is to complete this task with a solution that is simple and efficient. The result is an adaptive path planning algorithm that evades obstacles when necessary and uses these interactions to find an obstacle-free path with simple logic. The whole task is implemented within Paparazzi, an open source autopilot software. Flight tests are performed in an indoor flight arena with simulated GPS from a camera tracking system. Through these flight tests, the approach proves to be reliable and efficient.

1 INTRODUCTION

A wide range of applications for Micro Aerial Vehicles(MAVs) has created a demand for smaller vehicles with greater autonomy. Along with a decrease in size comes the need for lighter or fewer sensors and efficient utilization of the sensors onboard. Autonomy of the vehicle in many applications requires the ability to navigate in unknown environments and decide on paths or trajectories that ensures the safety of the vehicle in addition to the completion of the task at hand.

Collision avoidance is a broad topic and the problem approach inherently changes with differences in vehicle size, sensors available, type of aerial vehicle, and properties of the obstacles to avoid. In a survey of Collision Avoidance Systems (CAS), Albaker et al., has categorized all approaches into one of five key CAS functions; they are roughly: *sensing the environment*, *conflict detection*, *awareness*, *escape trajectory*, and *maneuver realization* [1]. From this all-encompassing view of CAS, this paper explores: 1) The combined functions *conflict detection and awareness* using a color-based detection method, and 2) The function *escape trajectory* using a simple adaptive path planning scheme within an obstacle field. This paper uses the terminology *Obstacle detection* and *Guidance*, for these 2 functions respectively.

Camera sensing is desirable because cameras are a lightweight sensor option and many MAV applications already require the video output as part of the main task. Detection of obstacles using vision methods has made much progress in MAV applications. Optical flow uses texture in the sensed environment to track points over time, ascertaining information about the distance of the lens from these points [2, 3, 4]. Another vision-based detection approach is stereo vision which uses two images with a known offset to create depth perception and recognize objects which are closer than the background [5, 6]. This approach is best used with a binocular camera setup, but can be implemented with a monocular vehicle by taking two side by side pictures with the same heading and a known offset in the vehicle position [7]. Finally, simultaneous localization and mapping (SLAM) has been used to fully map a room [8, 9], however not every task benefits from such a complex and information rich approach.

Approaches for guidance in collision avoidance tasks depend greatly on system and environment details. Looking only into static-obstacle avoidance and using a vehicle without communication with other agents there are a few guidance strategies popular among researchers. E-Field methods treat all agents/obstacles as a charged particle and model the repulsive forces between them. These forces drive the actions of the vehicle [1, 10]. Optimized, genetic, or learning approaches [11, 12] focus on finding optimal paths or policies. These can be computationally extensive and may still not guarantee convergence to an optimal path.

The conflict resolution guidance approach presented in this paper is categorized as one of predefined logic which is based on a set of predefined rules to avoid collisions. While these types of approaches do not attempt optimization, they are simple and therefore fast acting for collision avoidance. This research's specific guidance law takes advantage of the use of waypoints which is a strength of the chosen autopilot software. In addition, the logic is such that the planned path will adapt to obstacle placement and will find an obstacle-free path without the need of computationally heavy tasks of mapping or storing any obstacle information.

*j.l.junell@tudelft.nl. PhD Student
†Assistant Professor

http://www.imavs.org/

This paper presents an adaptive path planning logic which is then tested on a quadrotor micro aerial vehicle with a monocular camera sensor. The color filtering approach for obstacle detection is presented in Section 2 and is shown to be highly reliable for particular obstacles. The adaptive path planning and collision avoidance scheme is presented in Section 3. The complete system and experimental setup is detailed in Section 4. The results from test flights in different configurations are shown in Section 5. Finally, conclusions are drawn and discussed in Section 6.

2 OBSTACLE DETECTION

A color-based vision algorithm was used to identify objects within the quadrotor's 90° field of view. This approach is limited to obstacles of a chosen color which is distinguishable from the background, however the approach is very reliable within these limitations. This method was selected due to its reliability in practice and its ease of use as there is already a color filtering module implemented in the Paparazzi software for the AR Drone's camera (See Section 4.1.1 for details of the ARdrone quadrotor used for this research). Other vision approaches which are more robust to diverse obstacles could be attempted in place of this, including edge detection [13], optical flow [4, 3], or a combination method [13, 14]. Each of these vision approaches will have its own challenges, strengths, and weaknesses. The important thing to note is that the setup for this application is such that it would be easy to take another approach and attempt to implement it within the modular structure of the Paparazzi software.

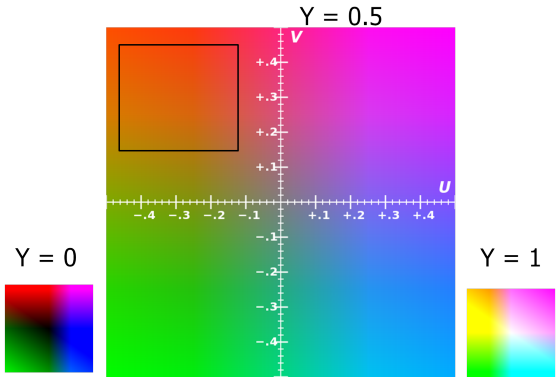


Figure 1: Color filtering detects colors within a specific range.

The algorithm works with the YUV images coming from the camera, downscaled to one quarter size for faster processing. The filter checks, pixel by pixel, if the three channels (one carrying the brightness and the remaining two the color) lie in the intervals defining the color shades of the obstacles. Figure 1 demonstrates the range interval of the color to detect.

The color filtering is already implemented in the AR.Drone 2 vision module of the Paparazzi software. The result is a binary image where "$1's$" mark the regions of the chosen color. Supposing first there is only one obstacle in the image, its position can be determined by finding the centroid of the detected pixels. As the vehicle approaches the obstacle, the number of detected pixels will increase. Thus, this number can be used as a metric of distance to the obstacle.

In practice, however, multiple obstacles can (and often will) be seen by the camera. Because the obstacles can be of any shape and can overlap, it would be difficult to detect them one by one. Thus, a different approach is taken. The image is split into five segments (Figure 2**a**, **b**) and the detected pixels are counted in each of them. If this value is greater than a certain threshold, that segment is considered blocked; otherwise it is free (Figure 2**c**). Finally if there exists three free neighboring segments then the quadrotor flies in the direction of the middle segment of the three (Figure 2**d**). In case of multiple options, the 0° heading has the priority. In situations where two or less neighboring segments are free (meaning there is not enough space for the drone between the obstacles), the drone turns by $+90°$ and repeats the detection.

The threshold value needs to be tuned depending on the size of the obstacles. Too low values will result in constant turning, as most segments will always be blocked. On the other hand, the threshold needs to be low enough, so that the thinnest obstacle is detected still at a safe distance.

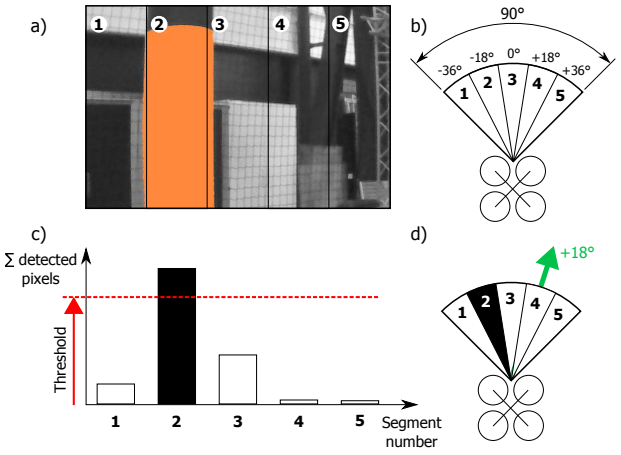


Figure 2: Obstacle detection based on color filtering: **a**) Video frame split into five segments, **b**) each segment corresponds to one fifth of the field of view, **c**) blocked segments are determined by a pixel threshold, d) a safe heading is planned if at least three neighboring free segments exist; otherwise the drone turns by $+90°$ and repeats the procedure.

3 GUIDANCE

In an unknown environment with randomly place obstacles, it is likely that an exploring vehicle will encounter an obstacle and need to avoid it. However, once those obstacles

have been detected and circumnavigated it would be advantageous to learn from the encounter and prevent taking that same path. Having to detect and avoid the obstacle again will only take more time than is necessary. Mapping of the entire flight area and/or storing information about the obstacle location in order to optimize a route takes a lot of computational resources. In this section, a guidance law is presented which learns from past obstacle encounters in order to plan an obstacle-free path without the use of obstacle information or complex algorithms.

The guidance approach was designed to be 1) simple to implement, 2) fail-safe against hitting obstacles, and 3) efficient in the sense that it would adapt to create an obstacle-free path. The final algorithm plays to the strengths of the Paparazzi software as it uses waypoints and logic within a flight plan in order to execute commands to the quadrotor. For more information on the paparazzi autopilot, see 4.1.3.

3.1 Adaptive path planning

The guidance concept is illustrated in Figure 3. An initial path is determined a priori without knowledge of the obstacle locations. This path and waypoint placement can be based on a specific task, for example: maximizing total distance traveled, traveling along the perimeter of an area, or trying to explore as much of the area as possible. Choosing initialization parameters will be discussed shortly in subsection 3.2.

Without knowing the environment, the initial path will likely have some points of collision with the obstacles. Although an obstacle may be detected earlier, the vehicle will continue on its path until an obstacle is nearby. An obstacle is determined "nearby" when the front segment and one of its adjacent segments are blocked. At this point, collision is within 1-2 meters if the obstacle is not thinner than approximately 20 cm. At this point, the vehicle will be commanded to stop and move the waypoint it was heading towards to a new location in a safe region. The waypoint will be placed 150cm in the direction of the safe heading output of the vision module and a predefined distance. If there is no safe heading or the waypoint is placed out of bounds, the vehicle will yaw and search for an acceptable place for its new waypoint. The logic is such that a last resort will be to go back the way it came and therefore should never get stuck in a corner.

Techniques for waypoint placement in difficult scenarios are put in place so that an in-bounds, safe heading can be found as quickly as possible. If the quadrotor avoids an obstacle and the resulting waypoint is placed outside the boundary, the vehicle will turn away from the boundary back towards the obstacle and search for another safe heading on the other side of the obstacle. If the logic were left here, there would be instances in the corner or in areas with many obstacles near the boundary where the vehicle could get stuck in yawing one way and then another in an infinite loop. To combat this worse case scenario, a counter, i, is used to count the number of times a waypoint is placed out of bounds. Each time a wp is placed out of bounds, $42 * i$ degrees will be turned (in the same direction). This will ensure that a 360° turn will be made if necessary. The value 42° was chosen rather than a value that divides into 360, so that if multiple turns were necessary to find a safe heading, the quadrotor camera will get as many viewpoints as possible.

The path will eventually converge to an obstacle-free path. In the example in Figure 3, it only takes two loops through the waypoints to come up with a converged path. If the next diagram in red is drawn by connecting the waypoints, it can be seen there are no longer any collision paths.

Managing special cases:

No safe zone is detected The quadrotor rotates its heading clockwise until it finds a safe area.

Waypoint is placed outside of a boundary The quadrotor searches for a safe zone on the other side of the obstacle by rotating incrementally in the opposite direction of the safe heading and searching again.

Between an obstacle and a corner There can be a case in a corner or when surrounded by obstacles where it will detect no safe zone, rotate clockwise, then rotate counter clockwise when it encounters a border. This has been addressed by setting a counter and incrementally turning more degrees each time a border is detected. A whole 360° turn is guaranteed if it is necessary. In a worst case scenario, the quadrotor can always go back the way it came.

There are two obstacles nearby each other The quadrotor will stop further away from obstacles because it merely detects if a threshold is surpassed. When a new waypoint is place only 2 meters away, it may not be enough to completely circumnavigate the obstacles and will result in the need for 2 waypoints needing to be relocated. This problem would be bypassed with a reasonable estimate of how far away the obstacles are. Then a variable waypoint displacement distance could be used.

3.2 Initial waypoint settings

As mentioned previously, the initial path and waypoint placement is chosen a priori and should be based on a specific task. In this example, the quadrotor could be following a mail route avoiding trees and sign posts; or it could be patrolling specific spots for crime. In another configuration, it could be trying to explore and video as much of the area as possible, in which case the path would cover more of the total area. The waypoint placement should reflect that of the task.

Secondly, the number of waypoints must be chosen. As shown in Figure 3, there are waypoints placed on top of each other. Two waypoints are placed at the end of a journey along the side of the square, and three waypoints are placed at the

http://www.imavs.org/

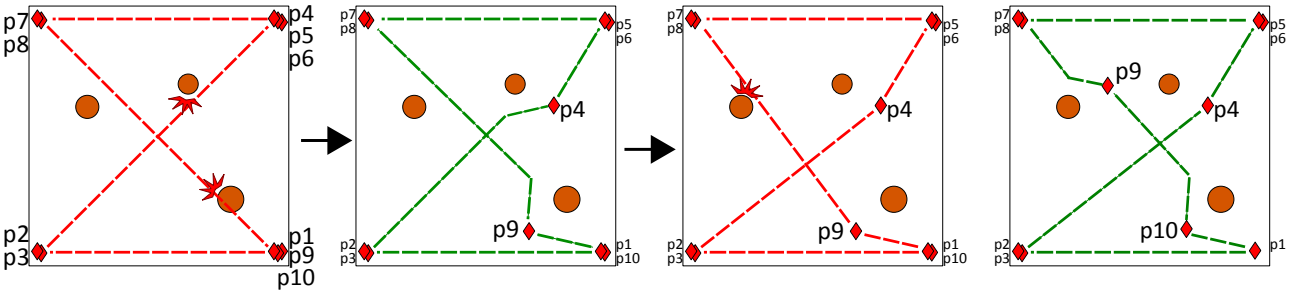


Figure 3: The path that an unobstructed vehicle would fly is in red and the actual path flown is in green. An unobstructed quadrotor would fly looping between waypoints. (eg. $p1 \rightarrow p2 \rightarrow p3... \rightarrow p_n \rightarrow p1...$). However, when an obstacle is detected in the path, the quadrotor stops and the waypoint it is currently moving toward is relocated to a safe zone N meters away from its current position. The obstacle is circumnavigated as the quadrotor continues to loop through the waypoints. This will happen each time an obstacle is encountered. The next round through the waypoint looping, the path will have changed so as to divert from previously encountered obstacles all together. However, the change can put other obstacles in the resultant path. The process then repeats. Given that there are enough waypoints created at the beginning, an obstacle-free path will be created.

http://www.imavs.org/

end of a diagonal journey. The more distance there is between two points, the greater the chance of multiple obstacle encounters, and the more need for waypoints to create a proportionally complex path. Therefore, an obstacle rich environment will require more waypoints to capture a complicated path to a desired point. However, it is not so simple as to just put a hundred waypoints in each location. If that spot is truly inaccessible, there should be few enough waypoints so that they will be moved away from that spot and the vehicle will stop trying in vain to get there.

4 FLIGHT TEST SETUP

4.1 *Resources*

This experiment utilized several resources which will now be described. For a more in depth description of these resources in a previous experiment, reference Junell et al. [11]. The most up to date information can be found at the websites in the footnotes.

4.1.1 AR-Drone 2 quadrotor

Since designing and building a quadrotor is not the main research intention for this experiment, it is very welcome to have a commercial quadrotor available which makes this research not only more convenient for this project but also more accessible to other researchers in the field. A quadrotor is an unmanned aerial vehicle consisting of 4 engines that spin independently of each other. Adjusting the thrust produced by each engine gives the necessary control to perform general motion as well as other aggressive maneuvers. The Parrot® AR-Drone 2 [1], is a commercially available quadrotor intended for recreational use. The AR-Drone 2 is also promising for research and educational purposes [15] as it is relatively inexpensive, and robust to damage. The AR-Drone features a forward facing camera, GPS, WiFi, and precision control. Most importantly for this project is that its built-in software can be overwritten.



Figure 4: The AR-drone 2 quadrotor MAV. Photo credits by Parrot.com

4.1.2 TU Delft Cyber Zoo

The TU Delft Cyber Zoo [2] was officially opened in March 2014 as a research and test laboratory for ground robots and aerial vehicles. Its space consists of a floor area that is 10 x 10 meters and extends 7 meters high. The space is contained inside an aluminum truss frame which holds nets to ensure the safety of people and surrounding equipment. Also supported on the truss structure are 24 cameras. These cameras are part of the *Optitrack* [3]*: Motive Tracker* optical tracking system. The quadrotor "wears" a special hull with reflective balls so that the cameras can track it. Within the Cyber Zoo boundaries, up to 32 rigid bodies can be tracked with high

[1] Parrot AR-Drone 2. http://ardrone2.parrot.com/

[2] TU Delft Robotics Institute. http://robotics.tudelft.nl

[3] Optitrack™. http://www.naturalpoint.com

precision accuracy. The system can also be used as a simulated GPS. Therefore, as long as an experiment is not using any feedback from the Cyber Zoo, everything done indoors can be translated to an outdoor environment.

This flight test will use the tracking system only for simulated GPS and for evaluating performance. The complete system as it is used is illustrated in Fig. 5. The computer running the OptiTrack software has a wired connection to the laptop running Paparazzi and from there can broadcast the GPS signal via WiFi to the quadrotor.

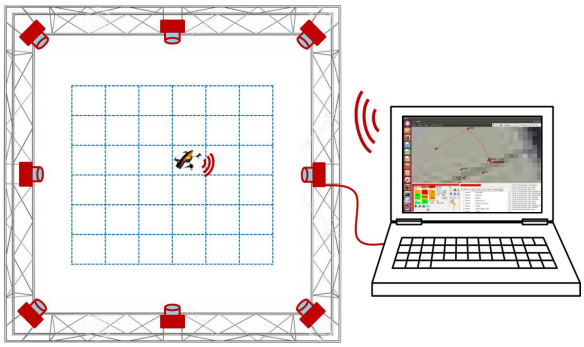


Figure 5: Cyber Zoo experimental setup: 24 cameras track the quadrotor. From this system, the quadrotor receives a simulated GPS signal. That position is transmitted via WiFi between a laptop and the quadrotor. Based on the location state and the memory state, a waypoint command will be sent to the quadrotor for it to execute.

4.1.3 Paparazzi Open Source Autopilot

Paparazzi [4] is a fully open source free autopilot for fixed wing and multi-copter UAVs [16]. A version of the software can be acquired at GitHub [5].

The quadrotor capabilities in Paparazzi include a model of a standard quadrotor that is close enough to the AR-Drone to function as a good inner loop controller given good tuning and standard non-aggressive maneuvers. Simulation of a flight plan in an intuitive GUI is available in addition to use as a Ground Control Station (GCS) for real flights. An example of the Paparazzi GCS interface can be seen in Fig. 6.

In the indoor setting, Paparazzi connects via WiFi to the quadrotor to give commands and via a wired connection to the Cyber Zoo computer to receive the position of the vehicle. In an outdoor situation, GPS would be picked up by the AR-Drone and used there directly.

Paparazzi is a modular platform and therefore has made it easy to add functionality by means of custom modules. Initializations, periodic and event functions can be added without effecting the main autopilot software. This makes customization very flexible and safe.

The high level position control of the vehicle is controlled by the flight plan. Waypoints and commands involving these waypoints are written into the flight plan. The autopilot software and the flight plan is compiled and uploaded onto the quadrotor computer. Once uploaded, the names and numbers of the waypoints and types of commands cannot be changed; however, the location of the waypoints can be modified at any time. Therefore, it is important to plan the method for which the waypoints will be used. Two new modules were created in this experiment. The first module processes the video from the forward facing camera on the AR-Drone and output the binary 5-tuple obstacle detection output as described in Section 2. The guidance is a set of commands which work together with the flight plan to move the waypoints based on predetermined logic.

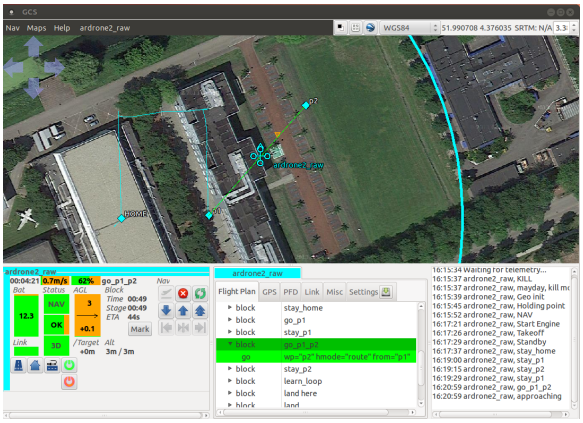


Figure 6: Paparazzi GCS (Ground Control Station) is the main interface for the user. In the GUI it shows a 2D map with a flight path (cyan line). The quadrotor status is shown in strips box (bottom-left) along with some buttons for common commands. Commands such as “go p1” or “stay p2”, can be seen in the Flight Plan tab in the Notebook frame (bottom-middle). Commands that have been made are printed in the console box (bottom-right).

4.2 *System Setup*

The controller consists of the Paparazzi autopilot with a vision module which continuously outputs the 5-tuple binary detection output representing the obstacles detected in each segment, and a guidance module consisting of functions used in the paparazzi flight plan. Once the software is uploaded to the quadrotor computer, the user can send commands from the GCS flight planvia WiFi to the quadrotor. By calling the block “Loop Route p2”, the quadrotor will begin a loop from p1 to p2 to p3, etc. Within each of these blocks are exceptions which will be triggered if an obstacle is detected and in the path of the quadrotor. If the exception is triggered, then the

[4] Paparazzi Free Autopilot. http://wiki.paparazziuav.org
[5] Paparazzi GitHub account. https://github.com/paparazzi/paparazzi

http://www.imavs.org/

quadrotor will stop, move the waypoint it was headed towards to a safe zone and continue towards it.

Paparazzi's inner loop controller, which is already uploaded onboard, will translate the waypoint command to specific low level control commands.

Figure 7 demonstrates the communication within the paparazzi controller. The vision module receives input from the camera and the GPS signal. It processes the images, and outputs information that will be used by both the flight plan in its exception block, and by the guidance module to determine the next action should an obstacle be encountered. The GPS signal is also used by many functions in the paparazzi software. The guidance module uses this information to determine the placement of waypoints and heading. The guidance module and flight plan, along with other standard functions within the software, provide the commands to the quadrotor which ultimately moves it to the desired heading and location.

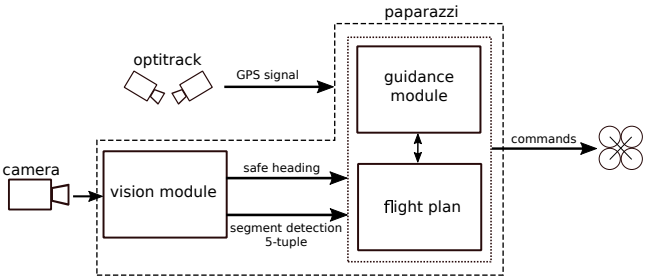


Figure 7: Communication between the modules and the flight plan within the paparazzi autopilot.

5 FLIGHT RESULTS

The results will now be presented and briefly discussed.

5.1 Easy configuration: Three obstacles spaced apart

The first test is configured just as the example from Section 3. Figure 8, shows the tracks of the flight test as received from the Optitrack tracking system. The final waypoint and obstacle positions are shown on the image.

The waypoints original configuration doesn't create an exact square because the original waypoints were placed in latitude/longitude coordinates based on the trackable area of the arena. The idea is still the same. It is shown on the tracks that the first approach toward an obstacles stops, backs up, and reroutes around the obstacle. The next route goes directly to the newly placed waypoint off to the side of the obstacle. This is seen easily in the routes from $p8 \rightarrow p9$. There are clearly 3 different angles leaving $p8$ from the three different positions that $p9$ was at.

The backtracking away from the obstacle is not seen for the $p4$ placement and the waypoint was placed far before the obstacle. This is because the obstacle was detected very early during the quadrotors flight from $p3 \rightarrow p4$. It is not known why the obstacle was detected from so far away, as it should not have been. One possibility is that a false positive detection was caused by seeing other obstacles during a quick yaw maneuver which, for an instant, triggered the detection conditions. This behavior is not necessarily undesirable since having a false positive detection does not result in a crash and in this case there was no harm done since the early waypoint placement still resulted in the desired effect.

5.2 Moving configuration: Adapting to a moved obstacle

The second configuration begins from the previously learned first configuration. As the the quadrotor is flying in its newly learned obstacle-free path, obstacleX is moved and placed in its path. The results show that the quadrotor successfully detects the obstacle and circumnavigates it in the same way as before. The result is a new obstacle-free path.

6 DISCUSSION AND CONCLUSION

This paper demonstrates a simple and effective obstacle avoidance approach within an indoor obstacle field. Past research is discussed for vision based obstacle detection and obstacle avoidance guidance tasks. The experimental approach and setup is presented in detail, and the results of the real-life flight tests are visualized. The results show that an obstacle field can be navigated and adapted to if an obstacle were to change location. However, limitations still do exist.

The benefits and limitations of this approach will now be itemized:

Benefits

- Both vision and guidance is simple to implement
- Does not use resources to map or store information about the obstacle location
- Converges to an obstacle-free path
- Color-vision detection is very reliable
- Logic implemented for special cases so that an inbound and safe heading can always be found if it exists
- If a spot is not accessible, waypoints will be moved and vain attempts to reach that spot will be abandoned

Limitations

- Constant waypoint displacement distance is used, thus wider obstacles would not be circumnavigated in one iteration of waypoint movement, but could take several.
- Number of waypoints must be chosen and placed a priori based on the desired task and number of anticipated obstacles.
- Limitations of the vision approach regarding type of obstacles:

http://www.imavs.org/

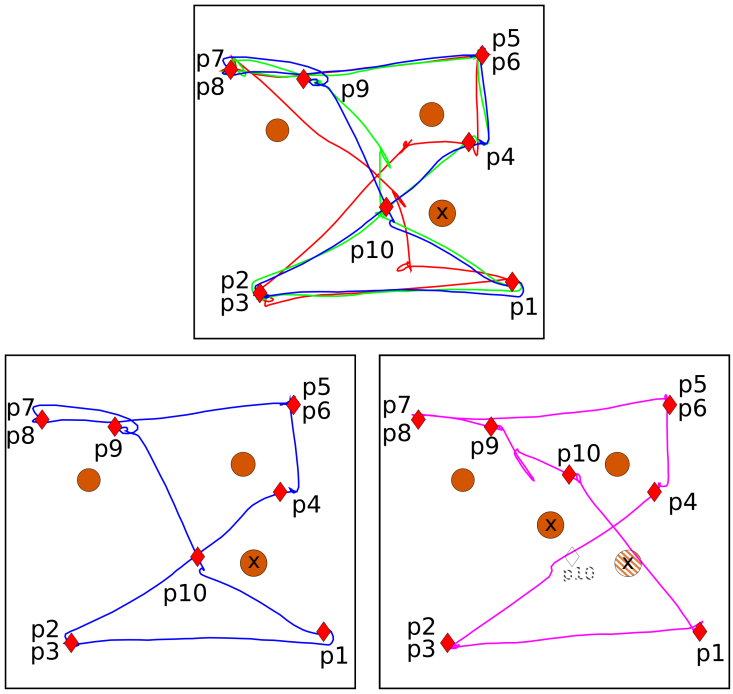


Figure 8: Tracks of the quadrotor during flights 1 & 2 from a birds-eye view. The final waypoint and obstacle positions are shown on the image. The waypoints are depicted as diamonds and are labeled. The obstacles are depicted as orange circles. *(Top figure):* Test 1 includes 3 rounds before a final adapted path is found. The first pass is in red, the second in green, and the third in blue. *(Bottom-left):* the final path learned in Test 1. *(Bottom-right):* After the path was solved in test 1, an obstacle was misplaced into the path. The path adapts to this new configuration. The tracks from the flight are shown along with the final waypoint positions.

- All obstacles must be of a similar (known) color and differentiable from background
- Cannot acquire information about obstacle width or distance

In order to address the limitations and to, for example, avoid obstacles of varying colors and texture, a more mature vision approach would need to be used. Luckily, much progress is being made in the realm of vision for guidance. With access to obstacle information such as width and distance, many of the guidance limitations could be eliminated.

For this research, the color-based vision approached works well. The necessary information is delivered reliably to the guidance algorithm for effective functionality. The guidance approach would only get better if more knowledge of the obstacles was acquired before hand or during flight from the vision algorithm. For example, if other types of detection methods were used separately or in collaboration, then the distance from the obstacle could be determined and the next waypoint could be placed in a more intelligent manner.

Overall, this guidance law is robust in that it is guaranteed to avoid any detected obstacles in its path as well as learn from its past encounters.

ACKNOWLEDGMENTS

Thanks to Matěj Karásek, Kimberly Mcguire, Sjoerd Tijmons, and Kevin Lamers.

REFERENCES

[1] B.M. Albaker and N.A. Rahim. A survey of collision avoidance approaches for unmanned aerial vehicles. In *International Conference for Technical Postgraduates (TECHPOS)*, pages 1–7, Dec 2009.

[2] Simon Zingg, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. Mav navigation through indoor corridors using optical flow. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3361–3368. IEEE, 2010.

[3] Ted Camus, David Coombs, Martin Herman, and Tsai-Hong Hong. Real-time single-workstation obstacle avoidance using only wide-field flow divergence. In *Proceedings of the 13th International Conference on*

http://www.imavs.org/

Pattern Recognition, volume 3, pages 323–330. IEEE, 1996.

[4] G.C.H.E. de Croon, H.W. Ho, C. De Wagter, E. van Kampen, B.D.W. Remes, and Q.P. Chu. Optic-flow based slope estimation for autonomous landing. *International Journal of Micro Air Vehicles*, 5(4):287–297, December 2013.

[5] Larry Matthies, Roland Brockers, Yoshiaki Kuwata, and Stephan Weiss. Stereo vision-based obstacle avoidance for micro air vehicles using disparity space. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3242–3249. IEEE, 2014.

[6] Sjoerd Tijmons, Guido de Croon, Bart Remes, Christophe De Wagter, Rick Ruijsink, Erik-Jan van Kampen, and Qiping Chu. Stereo vision based obstacle avoidance on flapping wing mavs. In Qiping Chu, Bob Mulder, Daniel Choukroun, Erik-Jan van Kampen, Coen de Visser, and Gertjan Looye, editors, *Advances in Aerospace Guidance, Navigation and Control*, pages 463–482. Springer Berlin Heidelberg, 2013.

[7] K. Lamers, S. Tijmons, K.N. Mcguire, J. Junell, and M. Karásek. Mantis inspired obstacle avoidance using monocular stereo vision. MAV course: Obstacle avoidance competition and report, April 2015.

[8] K. Celik, Soon-Jo Chung, M. Clausman, and A.K. Somani. Monocular vision slam for indoor aerial vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1566–1573, Oct 2009.

[9] Gim Hee Lee, F. Fraundorfer, and M. Pollefeys. Mav visual slam with plane constraint. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3139–3144, May 2011.

[10] J. Kim and P. Khosla. Real-time obstacle avoidance using harmonic potential functions. In *IEEE International Conference on Robotics and Automation*, pages 790–796 vol.1, Apr 1991.

[11] Jaime Junell, Erik-Jan van Kampen, Coen de Visser, and Qiping Chu. Reinforcement learning applied to a quadrotor guidance law in autonomous flight. In *AIAA Science and Technology Forum: Guidance, Navigation and Control Conference*, pages Paper No. AIAA 2015–1990, 2015.

[12] Stefan Wender and Ian Watson. Combining case-based reasoning and reinforcement learning for unit navigation in real-time strategy game ai. In Luc Lamontagne and Enric Plaza, editors, *Case-Based Reasoning Research and Development*, volume 8765 of *Lecture Notes in Computer Science*, pages 511–525. Springer International Publishing, 2014.

[13] A. Ohya, A. Kosaka, and A. Kak. Vision-based navigation of mobile robot with obstacle avoidance by single camera vision and ultrasonic sensing. In *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 704–711 vol.2, Sep 1997.

[14] Nguyen Xuan Dao, Bum-Jae You, and Sang-Rok Oh. Visual navigation for indoor mobile robots using a single camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pages 1992–1997, Aug 2005.

[15] Tomáš Krajník, Vojtěch Vonásek, Daniel Fišer, and Jan Faigl. AR-Drone as a platform for robotic research and education. In *Research and Education in Robotics - EUROBOT 2011*, volume 161 of *Communications in Computer and Information Science*, pages 172–186. Springer Berlin Heidelberg, 2011.

[16] B.D.W. Remes, D. Hensen, F. Van Tienen, C. De Wagter, E. Van der Horst, and G.C.H.E. De Croon. Paparazzi: How to make a swarm of Parrot AR Drones fly autonomously based on GPS. In *IMAV 2013: Proceedings of the International Micro Air Vehicle Conference and Flight Competition*, Toulouse, France, September 2013.

http://www.imavs.org/

Aircraft Inspection Using Unmanned Aerial Vehicles

Randa Almadhoun, Tarek Taha, Lakmal Seneviratne, Jorge Dias, Guowei Cai

Abstract— **In this paper, we propose a coverage planning algorithm for inspecting an aircraft, using an Unmanned Ariel Vehicle (UAV). Inspecting structures (e.g. bridges, buildings, ships, wind turbines, aircrafts) is considered a hard task for humans to perform, and of critical nature since missing any detail could affect the structure's performance and integrity. Additionally, structure inspection is a time and resource intensive task that should be performed as efficiently and accurately as possible. In this paper we introduce a search space coverage path planner (SSCPP) with a heuristic reward function that exploits our knowledge of the structure model, and the UAV's onboard sensors' models to generate resolution optimal paths that maximizes coverage. The proposed method follows a model based coverage path planning approach to generate an optimized path that passes through a set of admissible waypoints to fully cover a complex structure. The algorithm predicts the coverage percentage by using an existing model of the complex structure as a reference. A set of experiments were conducted in a simulated environment to test the validity of the proposed algorithm.**

http://www.imavs.org/

I. INTRODUCTION

The application of robotic inspection involves many robotic application components employing different types of robotic systems including: Unmanned Arial Vehicles (UAVs); Unmanned Ground Vehicles (UGVs); and maritime robots, all of which could be utilized for different inspection operations. The usage of UAVs particularly provides the flexibility of capturing visual information of the structure regions that are hard to reach, thus simplifying the reconstruction process which is essential for inspection. Inspecting large complex structures is particularly important in applications that require maintenance, fault traceability, anomaly and defects detection, and model digitizing. The presence of anomalies affects the performance and serviceability of complex structures, especially if these structures involve human lives such as an aircraft.

Technically, inspecting structures requires various robotic capabilities such as: localization in the environment where the structure exists; path planning and navigation in order to compute a set of achievable routes; sensing and perception in order to gather information about the structure from different viewpoints along the route. As such, it is important to equip the robot with intelligent sensing capabilities that enhance the quality of the information gathered in order to reconstruct, and inspect the structure of interest accurately.

Different research approaches have been followed in the past to perform inspection depending on the environment, the shape of the structure, and the level of the required details. The two main challenging research topics related to inspection are coverage path planning, and workspace 3D reconstruction. Coverage path planning is the process of computing a feasible path encapsulating a set of waypoints through which the robot must pass in order to completely scan the structure of interest. The two main coverage path planning categories include model based, and non-model based categories [1], [2]. The proposed algorithm follows a model based coverage path planning approach by which a set of waypoints is generated based on an existing reference model of the structure. The main aim of the model-based algorithms is to provide a set of waypoints that explores a structure so that every area of the structure is visible. Model based planning algorithms are further categorized into three categories based on the information embedded in the model of the structure including: set theory methods, graph theory methods, and computational geometry methods.

A number of structure-inspection algorithms that are suitable for robotics systems have been documented in the literature. In [3]–[6], an optimized path consisting of a set of stationary views providing full coverage of a ship hull was obtained by: using a polygonal mesh of the ship hull, and solving a set cover problem (SCP) that generates a redundant roadmap consisting of the waypoints. The path generation was preformed by solving a Travelling Salesman Problem (TSP) with lazy collision checking using Christofides approximation [7] and chained Lin-Kernighan improvement heuristic (LKH) [8]. A sampling based improvement using a modified Rapidly Exploring Random Tree algorithm (RRT*) [9] was used to reduce the path length. In [10], different search based algorithms including greedy variants and set cover with TSP algorithms were used with a known map in order to generate a set of sequenced waypoints that provide full coverage. It was found that using set cover method with TSP provides the best set of waypoints in terms of computation time and path cost [10]. Another model-based planning algorithm related to computational geometry methods was used in [11] in which a triangular mesh of the desired structure was utilized to determine the set of waypoints with the best configurations by solving it as an Art Gallery Problem (AGP) and connecting the waypoints by solving TSP using LKH. Additionally, Alexis et. al [12] proposed a Uniform Coverage of 3D Structures Inspection Path Planner (UC3D-IPP) that generates inspection paths by computing viewpoints and solving a TSP. An iterative strategy is used to improve the generated inspection path utilizing different remeshing techniques.

Moreover, a set of triangular meshes of objects of interest

Randa Almadhoun, Tarek Taha, Lakmal Seneviratne, Jorge Dias and Guowei Cai are with Khalifa University of Science Technology and Research, Abu Dhabi, UAE, e-mail: {randa.almadhoun,tarek.taha,guowei.cai,lakmal.seneviratne}@kustar.ac.ae

http://www.imavs.org/

were used in [13] in order to generate motion planning roadmap and coverage spaces, and find the inspection path using self-organizing neural network. Heng et. al [14] proposed a coverage and exploration algorithm that explores cluttered unknown environments by choosing goals that maximizes information gain and coverage in real-time using a Micro Arial Vehicle (MAV). The path was generated by solving a submodular orienteering problem, which was approximated to a modular problem, then an approximate solution was computed using Gurobi Optimizer [15]. The generated path is not the shortest path but it provides the maximum coverage since it incorporates a variety of yaw angles.

Due to the size, geometric complexity, and the application criticality (inspection) of aircraft structures, we propose an algorithm that explicitly targets the coverage, and accuracy requirements during aircraft inspection applications. The proposed work consists of three main components: viewpoints generation, path planning, and coverage evaluation. Our proposed SSCPP algorithm was developed to integrate the sensors Field of View (FOV), range limitations, and measurement errors in order to generate an optimized path encapsulating viewpoints that achieves the maximum coverage. We then evaluate the achieved coverage percentage, and the produced model resolution through these set of viewpoints. Unlike other approaches, the proposed algorithm is resolution complete [16] that generates optimal inspection paths which provides efficient 3D reconstruction of complex shapes with certain coverage guarantees in addition to an estimation of resolution accuracy.

An overview of our proposed algorithm is presented in section II, followed by a description of each of the main components of the algorithm including: viewpoints generation in section II-A, coverage path planning in section II-B, and coverage evaluation in section II-C. The simulated experiments used to verify the proposed algorithm are presented in section III. Finally, we draw our conclusions, and propose future enhancements in section IV.

II. SSCPP ALGORITHM

The proposed SSCPP algorithm was developed to utilize an existing mesh model of the complex structure and the models of the sensors mounted on the UAV. The algorithm consists of three components: viewpoints generation, coverage path planning, and coverage evaluation, each of which will be explained in details in the next sections. Algorithm 1 presents an overview of the proposed coverage planning procedure including the three components.

A. viewpoints Generation

Viewpoints generation is the process of generating a set of viewpoints from which the structure of interest is visible. Our viewpoints generation method starts by discretizing the structure's workspace using a specific grid resolution which generates a set of sample viewpoints defining an xyz position in 3D space. Next, an orientation based discretization step is performed on each position sample to produce a set of yaw angle orientations that the UAV can conduct inspection from. The generated viewpoints are represented by xyz coordinates and ψ yaw angle. Each sensor viewpoint is then generated by applying a 4×4 transformation matrix that defines the sensor location with respect to the UAV body frame. Figure 1 shows an example of discretizing the orientations by $\pi/4$, and generating the corresponding sensor viewpoints. The sensor is placed on the UAV with a translation of $6cm$ and a rotation of $5.5°$ around the y axis.

These sample viewpoints are then filtered using collision, distance and coverage based filtering approach. Collision based filtering eliminates the sample viewpoints that are inside the model or collides with the model. Distance based filtering keeps the sample viewpoints available within a defined distance from the structure model determined based

Algorithm 1: Coverage Planning Overview

Input : structure model, 3D grid resolution, angular resolution, sensor FOV and range limitations

1 Discretize workspace into a cubic grid with input resolution.
2 Discretize orientations at each grid with input angular resolution.
3 Preform transformation to generate the sensor viewpoints at each waypoints sample (position and angle).
4 Filter out the grid samples according to the sensor range limitations.
5 Perform path planning and trajectory generation using the filtered set of viewpoints.
6 Navigate through the waypoints, and perform 3D reconstruction
7 **foreach** *viewpoint in the generated path* **do**
8 Find the occlusion cull between the FOV and the model.
9 Add the occlusion cull to the predicted model.
10 **end**
11 Confirm model coverage completeness: predicted vs constructed

Output: waypoint trajectory, 3D reconstructed model, coverage percentage, model accuracy

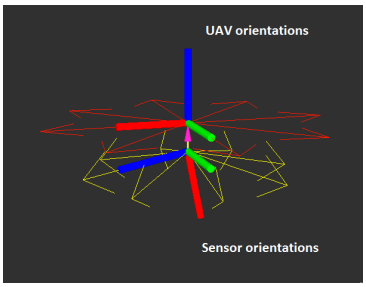


Fig. 1: Yaw orientation samples for the UAV body frame is shown in red, and the corresponding sensor orientation is shown in yellow

on the sensor's minimum and maximum effective range. To filter the viewpoints further, at each sensor viewpoint, the visible surface is extracted by performing frustum culling, which extracts the structure part that lies inside the sensor FOV frustum, and occlusion culling, which extracts the visible surface from the extracted frustum. Viewpoints that provide no coverage, as evident by the occlusion culling output, are then filtered out. The occlusion culling algorithm is presented in Algorithm 2 and Figure 2 shows an illustration of this process. Figure 3 shows the filtered pointcloud set, and sensor viewpoint.

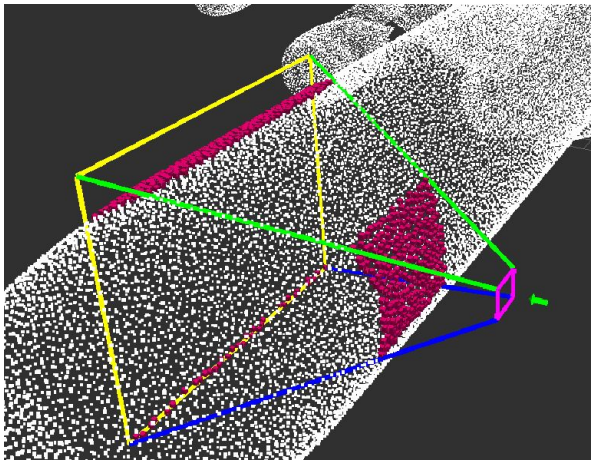
Algorithm 2: Occlusion Culling - RGBD Sensor

Input : sensor FOV, sensor origin O
Output: Visible surface point cloud
Identify the point cloud inside FOV frustum
Place the point cloud in a voxel grid
foreach *voxel in the voxel Grid* **do**
  Perform ray tracing from origin O to *voxel*
  if *voxel is not occluded by another voxel* **then**
    Store the *voxel* occluded free voxel point cloud
  end
end

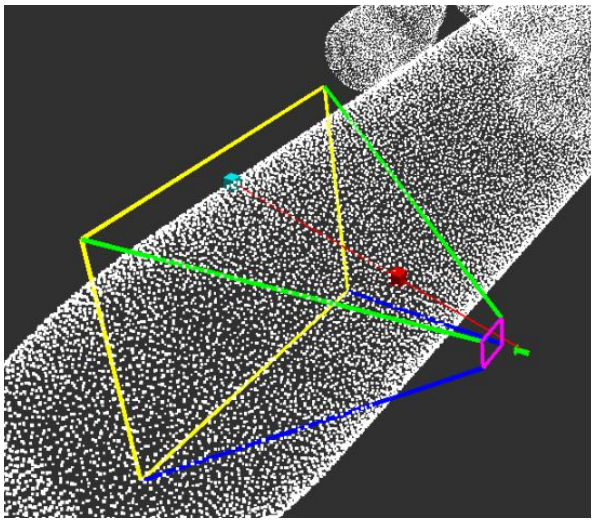
B. Coverage Path Planning

The next component of the algorithm is the coverage path planning. The discretized sample space at this stage consists of the set of filtered sample waypoints W, and the corresponding sensor viewpoints V. The search space is then generated by graphically connecting samples with their neighbors based on a pre-defined connection radius r as shown in Figure 4. The developed SSCPP, with a heuristic reward function, is then used to search this search space for an optimised path that achieves the desired coverage percentage. SSCPP is resolution complete graph search heuristic algorithm which increases the possibility of obtaining an optimal path as the discretization resolution increases. The heuristic reward function R minimizes travel distance δd and turns δa, and maximizes the coverage C computed at each step. The reward function is defined in equation (1), the first term is inversely proportional to the distance traveled, the longer the distance, the lower the ratio of the coverage C contribution to the reward. When the next waypoint involves only a rotation, then the reward is proportional to δa as shown in the second part of the equation. Algorithm 3 presents the search space coverage path planning algorithm which takes the set of waypoints and viewpoints, the connection radius r, the target coverage percentage, and the coverage tolerance as inputs and outputs the trajectory that will provide the target coverage.

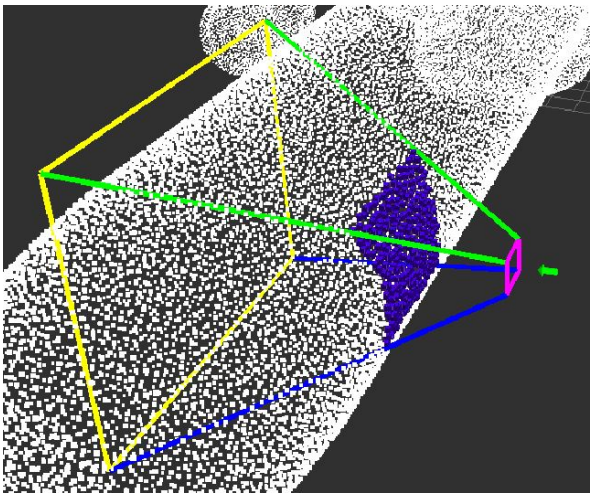
$$R = \begin{cases} \frac{1}{\delta d} \times C, & \text{if } \delta d > 0 \\ (1 - \frac{\delta a}{2\pi}) \times C, & \text{if } \delta d = 0 \end{cases} \quad (1)$$



(a) The visualization of the frustum cull (purple) of an aircraft model (white) from one sensor viewpoint



(b) The blue target voxel is occluded by the red voxel accros the red ray



(c) The visualization of the visible surface (blue) of an aircraft model (white) and the FOV planes from one viewpoint

Fig. 2: Visible surface extraction using occlusion culling

http://www.imavs.org/

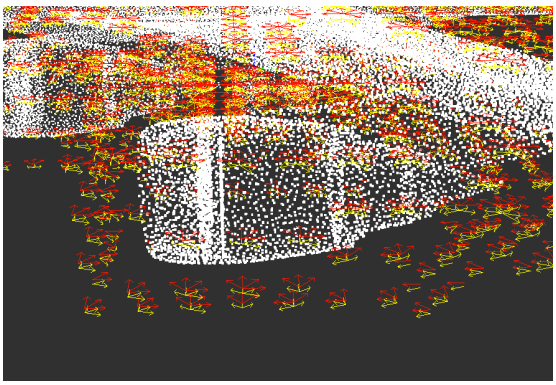


Fig. 3: Visualization of a set of filtered waypoints in red and the corresponding sensor viewpoints in yellow

Algorithm 3: Search Space Coverage Path Planner - SSCPP

Input : set of waypoints W and viewpoints V, connection radius r, starting pose $start$, target coverage percentage and coverage tolerance t.
Output: coverage path $C.P(s|S)$, coverage percentage

W, $V \subset \mathbb{R}^3$
Generate the search space nodes $S \leftarrow W$, V
Generate graph network by connecting S nodes based on radius r
Find the closest search space node to $start$
Initialize Open List O and Closed List C
Add the start node to O
while $s \in S$ *&* *target coverage not achieved* **do**
  Pick s_{best} from O such that $f(s_{best}) \geq f(s)$, $\forall s \in O$
  Remove s_{best} from O and add to C
  Expand s_{best}: for all $n \in Star(s_{best})$ & $n \notin C$
  foreach n *in* $Star(s_{best})$ **do**
    compute the extra coverage at n_v
    compute the distance between s_w & n_w
    compute the angle difference between s_w & n_w
    compute $R(n)$
    if $n \in O$ **then**
      $p \leftarrow n$
      if $R(p) > R(n)$ **then**
        remove n from O
      end
      else
        remove p from O
      end
    end
    else
      add n to O
    end
  end
  $C.P(s|S) \leftarrow$ s
end

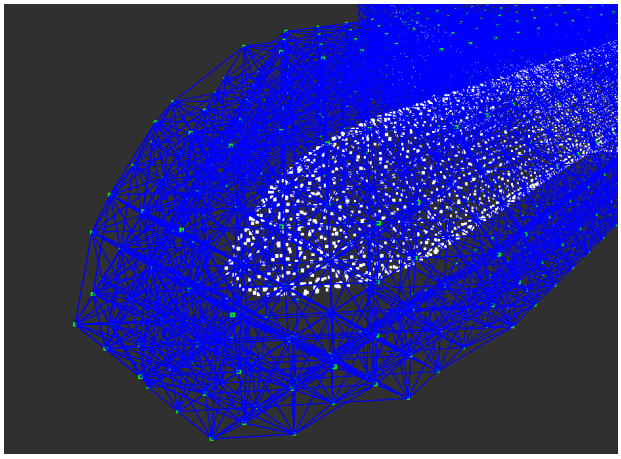


Fig. 4: Visualization of the connections generated from the search space samples applying a specific connection radius(3m)

C. Coverage Evaluation

The completeness of the coverage planning algorithm is assessed by quantifying the percentage of the covered volume of the structure compared to the predicted 3D structure volume across the generated path. The covered volume is the actual volume of the surface constructed by following the trajectory produced by the coverage path planning algorithm, and collecting data along the path. The predicted volume, however, is measured by performing frustum culling on the reference model at each trajectory waypoint, and accumulating the volume along the trajectory. Each of the covered and the original volumes of the structure is represented by a grid of voxels. These voxel grids are then used to calculate the coverage percentage as described in (2).

$$Coverage\ \% = \frac{Covered\ Volume_{\ Voxel\ Grid}}{Original\ Volume_{\ Voxel\ Grid}} \times 100 \quad (2)$$

This component was presented in literature as a critical part of the planning algorithm iterations, and as a significant performance criteria. The coverage completeness of the model-based algorithms can be computed directly based on the baseline used in the planning procedure as we preformed. A similar approach was adapted in [17], where the area coverage was quantified using a Monte-Carlo process by which the percentage is computed using the ratio of the number of sampled points that belongs to any of the sensed areas by the UAVs to the total number of the sampled points.

III. EXPERIMENTS AND RESULTS

A. Experimental Setup

In order to evaluate the proposed algorithm, a set of experiments were performed in a realistic robot simulator Gazebo [18]. A A340 aircraft mesh model was used to represent a complex aircraft structure that contains 32496 triangular faces. Software-in-the-loop (SITL) simulated experiments were conducted using an Iris quadrotor platform equipped with RGBD sensor. The main critical parameters of

http://www.imavs.org/

http://www.imavs.org/

the algorithm include the sensor FOV, grid resolution, sensor range, target coverage percentage, and tolerance to the target coverage percentage.

B. Experimental Results

An experiment was designed to check the predicted coverage percentage and compare it with the real coverage achieved. In this experiment, the quadrotor is assumed to carry a RGBD sensor mounted at $5.5°$ with a FOV of [58H,45V]° and a maximum depth range of $7m$. The 4×4 matrix presented in (3) is the sensor transformation matrix with respect to the quadrotor body frame used in the designed experiment. The proposed approach was used with a grid resolution of $1.5m$, a sensor distance of (2 to 4)m from the model, a connection radius of $3m$, a set of different target coverage percentages and a target tolerance of 1%.

$$\begin{bmatrix} 0.995401 & 0 & 0.0957973 & 0 \\ 0 & 1 & 0 & 0 \\ -0.0957973 & 0 & 0.995401 & -0.06 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

We repeated the same experiment with various target coverage percentages, and we evaluated the results based on metrics that include the path distance, the target coverage percentage and the end coverage percentage. The results of the experiments are summarized in Table I, and Figure 5 illustrates the path generated to achieve 90% coverage. Figure 6 illustrates the sensor FOV along some sampled viewpoints along the generated coverage path. As evident by our results, our method generated feasible path trajectories that consist of 656 waypoints to achieve the desired 90% coverage percentage. It's worth mentioning that due to the sensor mounting position, proximity of the aircraft to the ground, and the fact that only stable horizontal hovering of the quadrotor was considered, the maximum achievable coverage percentage is 92.1922% for this particular setup (measured by considering all the samples in the search space).

We also compared our method to the approach described in [11] using the same model, and the same experimental setup. This approach requires the sensor FOV, minimum and maximum distance from the model, sensor pitch mounting angle, incidence angle, and the number of iterations. These parameter were set to: sensor FOV = [120H,120V]° , minimum viewpoint distance of $1m$, maximum viewpoint distance of $20m$, camera sensor mounting angle of $5.5°$, incidence angle of $30°$ and 10 optimization iterations. The mesh model had to be simplified with this approach to include 10248 triangular faces instead of the original 32496. This is due to the fact that, this complex aircraft model contains a lot of occluded regions with a dense set of triangular faces, and that this approach depends on generating a viewpoint for each triangular face, making it difficult to generate a path passing through all these viewpoints.

This approach generated the path shown in Figure 7, consisting of 10248 viewpoints. As mentioned previously, an AGP is used in this approach to generate a set of viewpoints corresponding to each triangular face of the mesh based on the visibility of this face. This approach targets the visibility of all the faces of the mesh model through a comprehensive set of viewpoints.

For large complex geometrical structures such as the aircraft mesh, it is shown that even with the simplification performed on the model, significantly larger set of viewpoints was generated, making it hard to follow such a complex path to preform the inspection. The length of the generated path using this approach is $6758.06m$ which is considered a very long path compared to $1182.99m$ and 656 viewpoints generated using our proposed approach. Generating a short inspection path is of great significance since the UAVs have limited flight time.

TABLE I: Scenario 1 Summary of experiments results

Grid resolution = 1.5m, Effective sensor range = 2-4m, Connection radius = 3m, Tolerance to coverage percentage = 1%		
Target Coverage %	**Path Distance**	**End Coverage**
20%	96.8714m	19.023%
50%	386.462m	49.262%
70%	490.545m	69.218%
90%	1182.99m	89.026%

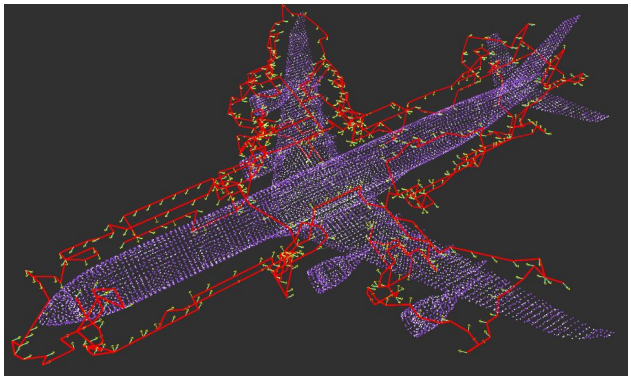
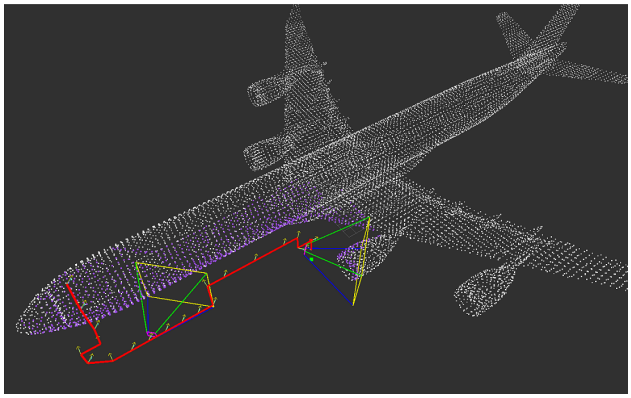


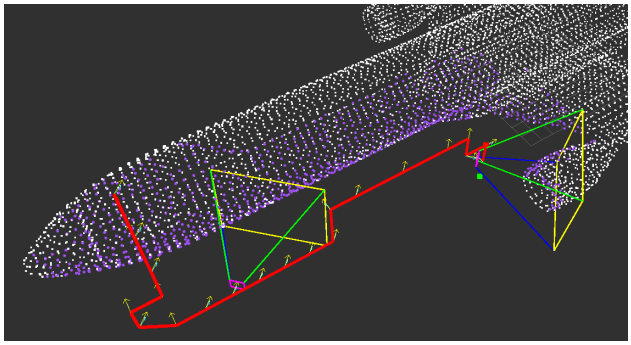
Fig. 5: Path generated to produce 90% coverage using our proposed method. The generated coverage path is shown in red, and the selected waypoints and their corresponding viewpoints in yellow. The original model pointcloud is shown in white and the covered part is shown in purple

Furthermore, the accuracy of the model generated by our approach was evaluated based on the RGBD sensor model. The accuracy of the 3D constructed model was evaluated by computing the standard deviation of error in depth Z at each point in the point cloud at each viewpoint in the generated coverage path following equation (4) presented in [19]. The values of $\frac{m}{f_b}$ and σ_d were computed in [19] as 2.85×10^{-5} and $\frac{1}{2}$ pixel by calibrating a Kinect RGBD sensor. The evaluation of accuracy helps in identifying and

http://www.imavs.org/



(a) Full view



(b) Closeup view

Fig. 6: Sample path generated to produce 10% coverage. The path is displayed in red and the selected waypoints and their corresponding viewpoints in yellow. The original model pointcloud is shown in white and the covered part is shown in purple. The FOV for one of these waypoints is shown, where the top plane is shown in green, the bottom plane in blue, the near plane in red, and the Far plane in yellow

improving the regions with the lowest accuracy in order to provide highly accurate 3D reconstruction. Figure 8 shows an illustration of the results of accuracy computations visualized as a yellow color gradient that ranges from the lightest (highest accuracy) to the darkest ranges (lowest accuracy). It provides an accuracy indication of the generated model taking into consideration the RGBD sensor noise model.

$$\sigma_z = (\frac{m}{f_b}) Z^2 \sigma_d \tag{4}$$

One of the generated coverage paths that targets a coverage of 90% was evaluated in simulation using Gazebo SITL. Figure 9 shows the reconstructed 3D model as a result of following that path. The 3D reconstruction was performed using Real Time Appearance Based Mapping (RTAB) [20] and Octomap [21]. The resulting model demonstrates that the generated path are traversable, and generate the desired coverage percentage.

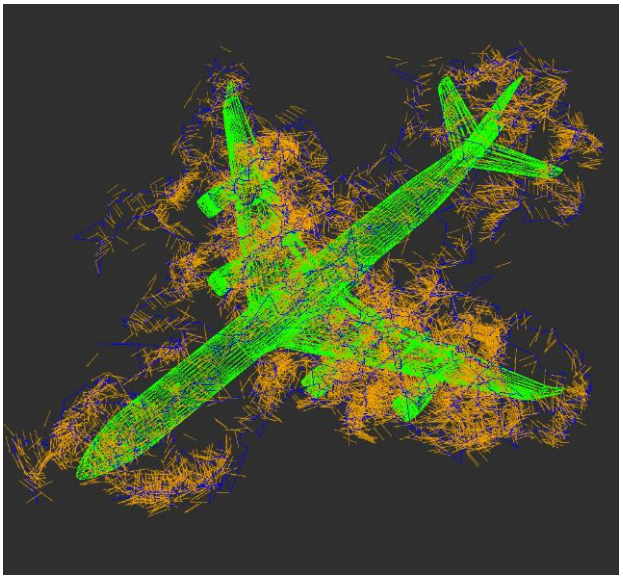


Fig. 7: Visualization of the coverage path generated using the approach of [11]. Orange arrows represent the viewpoints, and blue line segments represent the generated path.

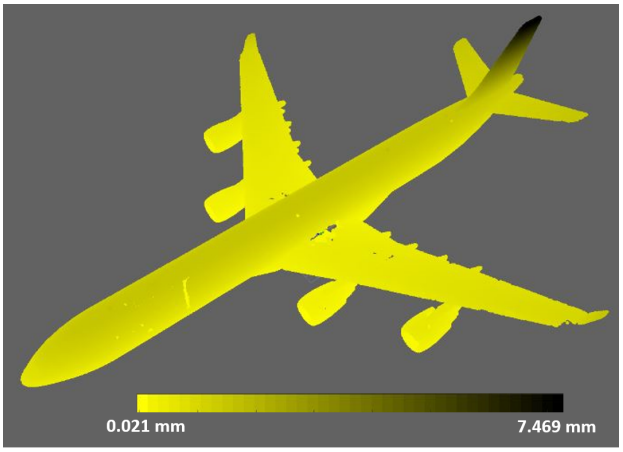
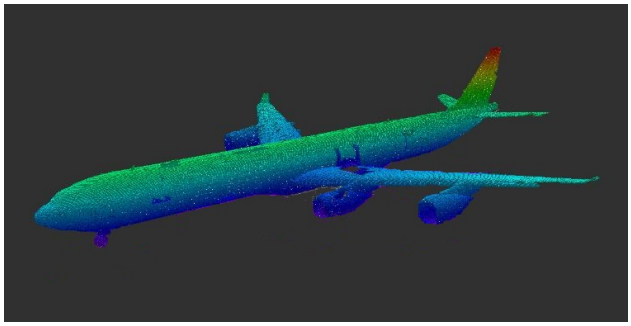


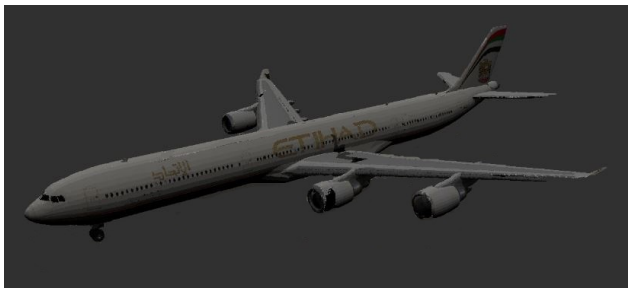
Fig. 8: Color gradient model representing the accuracy across the 90% path generated by our approach ranging from the lightest (highest accuracy) to the darkest (lowest accuracy). A colormap shows the lowest and highest standard deviation of depth and their corresponding color range

IV. CONCLUSION

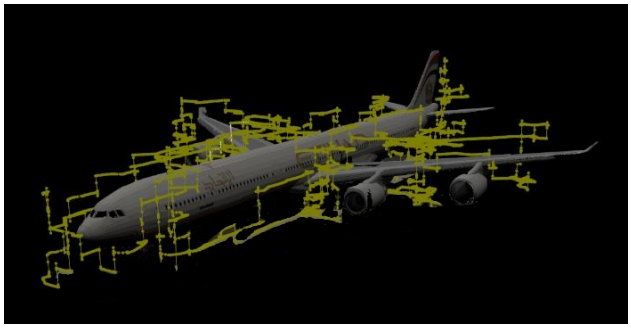
In this paper we introduced a coverage path planning approach that facilitates the inspection of large, geometrically complex structures using a quadrotor platform. We integrated sensor models to generate a coverage path offline and provide a prediction of the coverage percentage. The algorithm and the overall approach was verified using a realistic robot simulator where a quadrotor follows the generated coverage path and generates a 3D reconstructed model. Future work will focus on integrating the sensor accuracy directly in the planning heuristics to generate coverage paths that not



(a) 3D reconstruction using Octomap



(b) 3D reconstruction using RTAB



(c) The path followed by the quadrotor during the 3D reconstruction

Fig. 9: 3D reconstruction models generated by following the 90% coverage path generated in 1

only guarantee coverage to a certain percentage, but also guarantee that the model accuracy will be within a certain resolution.

REFERENCES

[1] W. R. Scott, "Model-based view planning," *Machine Vision and Applications*, vol. 20, no. 1, pp. 47–69, 2009.

[2] W. R. Scott, G. Roth, and J.-F. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM Computing Surveys*, vol. 35, no. 1, pp. 64–96, 2003.

[3] B. Englot and F. Hover, "Inspection planning for sensor coverage of 3D marine structures," *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pp. 4412–4417, 2010.

[4] B. Englot and F. S. Hover, "Sampling-based sweep planning to exploit local planarity in the inspection of complex 3D structures," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4456–4463, 2012.

[5] B. Englot and F. Hover, "Sampling-Based Coverage Path Planning for Inspection of Complex Structures." *Icaps*, pp. 29–37, 2012. [Online]. Available: http://www.aaai.org/ocs/index.php/icaps/icaps12/paper/download/4728/4711

[6] F. S. Hover, R. M. Eustice, a. Kim, B. J. Englot, H. Johannsson, M. Kaess, and J. J. Leonard, "Advanced Perception, Navigation and Planning for Autonomous In-Water Ship Hull Inspection," *Intl. J. of Robotics Research*, vol. 31, no. 12, pp. 1445–1464, 2012.

[7] N. Christofides, "Worst-Case Analysis of a New Heuristic Prepared for the Travelling Salesman Problem," no. February, 1976.

[8] K. Helsgaun, "Effective implementation of the Lin-Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[10] C. Dornhege, A. Kleiner, and A. Kolling, "Coverage search in 3D," *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2013*, 2013.

[11] A. Bircher, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart, "Structural Inspection Path Planning via Iterative Viewpoint Resampling with Application to Aerial Robotics," pp. 6423–6430, 2015.

[12] K. Alexis, C. Papachristos, R. Siegwart, and A. Tzes, "Uniform coverage structural inspection path-planning for micro aerial vehicles," *IEEE International Symposium on Intelligent Control - Proceedings*, vol. 2015-Octob, pp. 59–64, 2015.

[13] P. Janousek and J. Faigl, "Speeding up coverage queries in 3D multi-goal path planning," *Proceedings - IEEE International Conference on Robotics and Automation*, no. 1, pp. 5082–5087, 2013.

[14] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient Visual Exploration and Coverage with a Micro Aerial Vehicle in Unknown Environments," *in IEEE International Conference on Robotics and Automation*, 2015.

[15] (2014) Gurobi optimizer reference manual. [Online]. Available: http://www.gurobi.com/

[16] Y. Bestaoui Sebbane, *Planning and Decision Making for Aerial Robots*, 2014, vol. 71.

[17] A. Wallar, E. Plaku, and D. A. Sofge, "A Planner for Autonomous Risk-Sensitive Coverage (PARC OV) by a Team of Unmanned Aerial Vehicles," *in IEEE Symposium on Swarm Intelligence (SIS)*, 2014.

[18] Gazebo. [Online]. Available: http://gazebosim.org/

[19] K. Khoshelham and S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications," *Sensors*, vol. 12, no. 12, pp. 1437–1454, 2012. [Online]. Available: http://www.mdpi.com/1424-8220/12/2/1437/

[20] M. Labbe and F. Michaud, "Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 2661–2666.

[21] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

http://www.imavs.org/

An MAV Localization and Mapping System Based on Dual Realsense Cameras

Yingcai Bi*, Jiaxin Li, Hailong Qin, Menglu Lan, Mo Shan, Feng Lin and Ben M. Chen
National University of Singapore, Singapore

http://www.imavs.org/

ABSTRACT

Online localization and mapping in unknown environment is essential for Micro Aerial Vehicles (MAVs). Both accuracy and robustness are required in the realtime applications. In this paper, we present a dual camera system to estimate the pose of an MAV and generate an obstacle map for navigation. The recently released light-weight Intel Realsense depth cameras are utilized to build the system, one is forward-facing and the other is downward-facing. The downward-facing camera provides a high-frequency (60hz) velocity measurement while the forward-facing camera computes a low-frequency (10hz) position measurement. Experiments demonstrate the good performance of our proposed system.

1 INTRODUCTION

MAV is an efficient tool for multiple tasks, such as inspection and exploration, due to the small size and high maneuverability. The development of autonomous MAVs is a hot topic in both research community and commercial market. Currently, most of the MAVs rely on Global Positioning System (GPS) and Inertial Management Unit (IMU) to localize and fly autonomously in open space. Some MAVs are integrated with camera-based velocity estimation component (optical flow) to stabilize while GPS signal is not available. However, it is still very challenging for a more complete task like autonomous fly in GPS-denied environment. To achieve this, the MAV needs an intelligent navigation system with the capability of localization, mapping, obstacle avoidance and planning. The focus of this paper is to develop an efficient Simultaneous Localization and Mapping (SLAM) system for MAVs.

SLAM is a problem of building map and simultaneously keeping track of location for a robot in unknown environment. The challenges are mainly two-fold. On one hand, the hardware including processors and sensors should meet the requirement of size, weight, computation and power consumption. On the other hand, the software including software architecture and algorithms should meet the requirements of robustness and efficiency. Currently, the main sensors used for SLAM are laser scanners and cameras. There are already many robust and precise laser based SLAM solutions. However, integration of laser scanners on MAVs is still not efficient enough because of size, weight and cost limit. Comparing with laser based SLAM, visual SLAM is more promising. Cameras can provide much more data than laser scanner with smaller size and cheaper price. But, because of the difficulties in handling noise and image understanding, there are still few complete and robust solutions for visual SLAM.

In this paper, we will discuss the problem of how to deploy a complete visual SLAM system on MAVs. By investigating state-of-the-art methods from the perspectives of both hardware and software, we propose our solution to equip our MAV a realtime robust visual SLAM system. The investigated methods concern laser scanner, monocular camera, stereo camera and RGBD camera based methods, especially those representative methods which push the realtime robust performance to practical application phase in last several years. In our work, we choose Intel Realsense cameras (Fig. 1) as our visual sensors, Intel x86 portable PC as our processor, ORB-SLAM as our SLAM framework. The choice is discussed in detail and the implementation and experiments are covered in this paper.

The contributions of this work are as follows: We address the challenges and current achievements on visual SLAM which become practical to use in real world scenarios. Also we show the early attempt to deploy Intel Realsense depth camera on MAV to demonstrate a complete SLAM solution. Furthermore, we present the point cloud filtering and octomap building modules, which are the essential components for obstacle avoidance and path planning.



Figure 1: Intel Realsense camera

2 RELATED WORKS

According to different sensor type, the SLAM method varies. In this section, we will discuss current methods based on three categories. First is laser based SLAM, which has the longest history due to the high accuracy of range

*Email address: yingcaibi@u.nus.edu

measurements. Second one is visual SLAM, which is the hot topic nowadays. Third one is visual odometry, which can be considered as a subset of visual slam without loop closing. The visual odometry discussed here is a good way to estimate the short-term motion and benefit the visual SLAM method. Furthermore, we discuss about some existing applications on MAVs.

2.1 Laser SLAM

For the laser based methods, we discuss two representative ones. Kohlbrecher et al. present a 2D laser based method named Hector SLAM [1]. It is a robust localization and mapping method already verified on many practical applications. Hector SLAM calculates the relative position and orientation by aligning the incoming scan with the map maintained by all previous scans. The disadvantage of Hector SLAM is that it can only work under the assumption of 2D workspace. It needs to utilize more sensors to get a complete 6DOF estimation. Ji Zhang et al. propose an impressive 3D laser based localization and mapping method, which is called LOAM [2]. LOAM uses only one 3D laser scanner, either the rotating one-axis or non-rotating two-axis laser scanner. It solves the problem that range measurements are not synchronized and suffer severe distortion with external motion. It divides the complex problem of SLAM into two steps. One is distortion correction and fine registration of point cloud. The other is motion estimation at high frequency to estimate velocity of the laser scanner. The algorithm can provide accurate motion measurements and point cloud registration iteratively with both low-drift and low-computational complexity.

2.2 Visual SLAM

Klein and Murray propose Parallel Tracking And Mapping (PTAM) method for estimating the camera pose in a small-scale unknown scene [3]. This is the milestone of feature based realtime visual SLAM method with the first proposed idea of concurrent tracking and mapping. One thread is responsible for tracking the camera motion relative to the local map while another thread maintains a 3D map built from 3D points in keyframes by using bundle adjustment[4]. PTAM beated all the filter-based real-time methods at that time, including the famous EKF-SLAM and FastSLAM. However, there are still several shortages in PTAM including the less robustness of feature points, lack of efficient loop closing and lack of large scale environment handling.

Felix Endres et al. propose a fully functional SLAM method based on RGBD camera called RGBDSLAM [5]. They develop a typical feature based graph slam pipeline, which can detect the closing loop and optimize on the pose graph. Since they use frame to frame feature matching to find connection in pose graph, the computational complexity is high. Also, they don't handle the graph size efficiently, which will make the system slow down when the map size keeps growing.

Raúl Mur-Artal et al. recently propose ORB-SLAM, which is a comprehensive integration and innovation with the best ideas proposed in previous feature based methods [6, 7, 8]. The drawbacks of previous PTAM and RGBDSLAM are solved. It can work on both monocular and depth camera, showing a great potential for practical applications. Competing with the best feature based methods, Jakob Engel et al. propose a Large-Scale Direct Monocular SLAM (LSD-SLAM) method in ECCV 2014 [9]. It does not use corners or any other local features, but performs direct tracking by image-to-image alignment. Coarse-to-fine pyramid searching with a robust Huber loss is used to handle large motion and outliers. Extensions based on omnidirectional camera and stereo camera are also presented [10, 11] to solve the absolute scale and handle strong rotation.

2.3 Visual Odometry

Andreas Geiger et al. propose a realtime stereo vision based 3D reconstruction method called libviso2 [12]. They use simple blob and corner detector as features and calculate 3D coordinates of the features through triangulation in the previous image pair. The optimal motion is calculated by minimizing image reprojection error. Albert S. Huang et al. propose a visual odometry and mapping method using a RGB-D camera called libfovis [13]. It uses the FAST feature detector with an adaptive threshold on three Gaussian pyramid levels. They use similar method as Geiger to get the motion. Both algorithms use feature descriptors that are not invariant to rotation or scale changes. Therefore the running frequency of both algorithms need to be high to cope with movements.

Christian Forster et al. propose a Semi-Direct Visual Odometry (SVO) algorithm that is precise, robust, and fast. SVO detects feature points and find feature correspondence by direct motion estimation instead of feature matching. Thus, it achieves an increased speed by avoiding feature extraction in every frame, and increased accuracy through subpixel feature alignment. The problem of SVO is that it is designed and limited to work on global-shutter high-framerate downward facing camera.

2.4 Applications on MAVs

There are some real-time applications of SLAM methods on MAVs. Bry et al. presented impressive flight results for both fixed-wing and quadrotor using a 2D laser scanner and sensor fusion to get 3D state estimation. There are also some different designs from the literature [14, 15, 16]. Bachrach et al. proposed an estimation, planning and mapping solution using a RGBD camera [17]. Schmid et al. presented an autonomous navigation solution based on stereo vision [18]. They demonstrate that stereo vision works in both indoor and outdoor environments with a Core2Duo board and a FPGA card for the heavy computation.

http://www.imavs.org/

http://www.imavs.org/

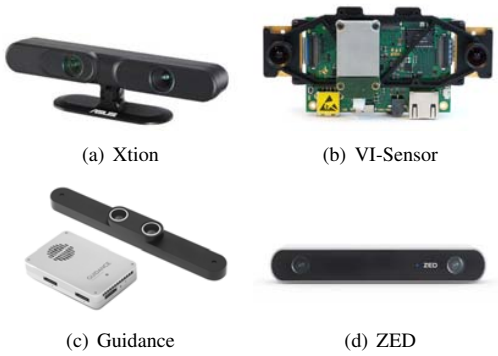
3 HARDWARE SELECTION

SLAM for MAV is a systematic problem. The core of algorithm development becomes mature, but the success still lies on a complete hardware-software solution. There is still not any commercial visual SLAM solution for MAV in the market while the recently released Dyson 360 Eye Robot Vacuum demonstrates a 360-degree camera based SLAM solution. In this section, we investigate the available visual sensors for MAV and discuss about the usability for a fully autonomous MAV.

Laser Scanner or Lidar is a classic accurate range measurement device based on the calculation of laser flight time. It can easily achieve centimeter accuracy working in both indoor and outdoor environment. However, to get a dense 3D scanning result, additional optical components and motors are needed. This make the laser scanner expensive and bulky. For example, the latest model of small-size 3D lidar from Velodyne weighs more than 800 grams [1]. On the other hand, camera obviously is a more compact solution as a 3D sensor. A typical camera weighs only 20 grams and can provide an image resolution up to 4K. Also, camera is cheap and accessible everywhere. However, the deployment of cameras on MAVs rely heavily on the hardware and software design to achieve enough accuracy.

Monocular camera is the most compact configuration, but monocular SLAM method is up to unknown scale and must include a metric initialization process to solve the scale problem. Stereo cameras are widely used in many industry applications for measuring the distance of objects by triangulation and demonstrated to be more robust. RGBD cameras based on structure light or Time of Flight (ToF) becomes very popular in recent years. RGBD cameras can provide depth information, thus share many properties with stereo cameras. The differences between them are the range and spatial density of depth data. Since RGBD cameras calculate depth with an active infrared structure light projector, they can estimate depth in areas with poor visual texture, but the range is limited and can only be used indoor. On the other hand, stereo cameras based on image matching are limited to rich texture environment but work well in outdoor environment. The stereo and RGBD camera based SLAM methods degrade to the monocular ones if depth is partially or totally unknown due to the low-texture or strong illumination change.

Kinect is the first commercial available RGB-D camera released in 2011 by Microsoft. It's demonstrated to be a huge success in the area of Human Computer Interface (HCI). Since the launch, it has been widely used on robots such as ground robots, service robots and MAVs. As the successor of Kinect, Asus Xtion(Fig. 2(a)) become popular because of the compact size and less power consumption. The VI-Sensor (Fig. 2(b)) is a light-weight, time-synchronized stereo camera for visual-inertial applications. It features a high-quality global shutter stereoscopic camera and an industrial-grade inertial measurement system. DJI Guidance (Fig. 2(c)) is another off-the-shelf stereo camera for MAVs. It uses ultrasonic sensors and cameras to gather realtime information about its surroundings. ZED stereo camera (Fig. 2(d)) developed by StereoLab is another popular options for MAVs with the capability of realtime depth calculation by GPU.



(a) Xtion (b) VI-Sensor

(c) Guidance (d) ZED

Figure 2: Camera options

Table 1: Camera comparison

Camera	Weight	Range	Environment	Depth
Xtion	Middle	Middle	Indoor	Yes
VI-Sensor	High	Middle	Indoor/Outdoor	No
Guidance	Middle	Middle	Indoor/Outdoor	Yes
ZED	Middle	Long	Indoor/Outdoor	Yes(GPU)
Realsense	Low	Middle	Indoor/Outdoor	Yes

Intel Realsense R200 camera has one camera providing RGB image and one stereo infrared camera producing depth. Table 3 shows a comparison of different cameras considering weight, range and operation environment. Realsense camera is the lightest one, and can work both indoor and outdoor with the help of a laser projector. Compared with standard visible light stereo camera like VI-sensor, Guidance and ZED, Realsense can provide short range depth information even in textureless environment. The indoor range is approximately 0.5-3.5 meters and outdoor range is up to 10 meters. In this paper, we try to push the performance of small size Realsense cameras to get a good localization and mapping result for MAVs.

4 CAMERA MODEL FOR REALSENSE R200 CAMERA

To fully configure and customize the algorithm, we use only the raw color image and depth image from Realsense camera for processing. The rgb image is aligned with the color camera, while the depth image is aligned with the left ir camera. Also, the resolution of them could be different according to configuration. Therefore, the color and depth

[1] http://velodynelidar.com/vlp-16.html

camera registration is needed to align the pixels from two different cameras.

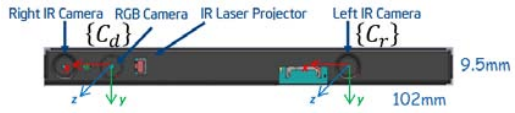


Figure 3: Realsense camera model, color camera coordinate system is based on RGB camera, depth coordinate system is based on left IR camera

As illustrated in Fig. 3, we define the corresponding coordinate systems. The depth coordinate system is denoted as C_d and color camera coordinate system is denoted as C_r. A 3D point coordinate in C_d is represented as P_id, where i is the index number. Similarly, A 3D point coordinate in C_r is represented as P_ir. The extrinsic matrix $[R_{rd}|t_{rd}]$ determines the transformation between P_id and P_ir.

$$P_ir = [R_{rd}|t_{rd}]P_id \tag{1}$$

With the 3D point coordinates in color camera frame, we can project these 3D points to image plane so that rgb pixels are aligned with their depth. The pinhole camera model is used to calculate the intrinsic matrix. The projection function is defined as,

$$\pi(P) = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \times P_ir \tag{2}$$

In our implementation, the extrinsic rotation matrix is an identity matrix and the translation only happens in the y direction. The image distortion is not included in the intrinsic matrix since the image is already rectified. We can get the related intrinsic and extrinsic calibration parameters from the factory calibration provided by the driver. This alignment procedure makes the system flexible to use diverse stream simultaneously.

5 LOCALIZATION AND MAPPING

5.1 *Downward-facing Velocity Estimation*

We estimate the 3D velocity by following the state-of-the-art feature based visual odometry method enhanced by the depth image [19]. The main advantage of this method is that color images and depth images are processed separately without the need of strict synchronization. This method matters because we observe that the color image and depth image from Realsense camera can not always arrive at the same time.

The algorithm structure is shown in Fig. 4. FAST corner features are detected from current image and tracked in the subsequent images by KLT optical flow method. The depth information of the visual features could be known and unknown. The corresponding known depth is searched in a k-d tree point cloud map which is built with all previous tracking poses and depth images. Those feature points with unknown depth can be tracked between multiple frames and triangulated to get the depth. The pose is calculated using Gauss-Newton optimization to minimize the projection error between consecutive two images. A point cloud map is maintained using the pose interpolation according to the timestamp.

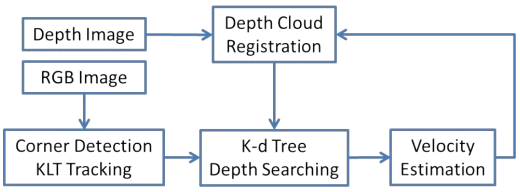


Figure 4: Velocity estimation algorithm structure

5.2 *Forward-facing SLAM*

As discussed in section 2, ORB-SLAM is currently one of the best SLAM method which benefits quite a lot from the feature based SLAM research in recent years. There are three concurrent threads in ORB-SLAM, namely tracking, mapping and loop closing. In the tracking thread, a constant velocity motion model is used to predict current camera pose. By searching the corresponding map points observed in the last frame, current camera pose is optimized. If tracking is lost due to small feature number, the frame will be converted into bag of words and queried in the recognition database to do global relocalization.

In the mapping thread, keyframes from tracking thread are inserted to the map as fast as possible followed by a mechanism to cull redundant keyframes. This make the tracking more robust to fast camera motion while keeping the efficiency of mapping. A covisibility graph is maintained to do local map optimization for both camera pose and map points.

In the loop closing thread, the image candidates sharing the similar view will be searched. The search is accomplished efficiently thanks to the inverse index in the vocabulary tree, which stores weights of the words in the images they appear. With the detected loop candidates, an essential graph is generated to optimize the global map with minimal pose.

5.3 *Octomap Generation*

Octomap is a compact representation of obstacles for MAV navigation. With the robust pose estimation, an octomap can be generated and used for path planning. Since the depth image from Realsense camera is noisy, we need to do filtering on it before inserting to octomap. In our implementation, we transform the depth image to point cloud and apply three different kinds of filters on point cloud. First, we use a voxel grid downsampling filter to reduce the data size according to the map resolution required by the planning

http://www.imavs.org/

http://www.imavs.org/

module. Second, we transform the point cloud from camera frame to the map frame and use a pass through filter to cover the specific height range. This filter can be tuned to remove the ground plane and further reduce data size. At last, an outlier filter is used to remove small patch obstacles, which highly possible to be sensor noise. The filters make sure we get a fine quality global map for planning use.

6 EXPERIMENTS

The indoor test images are captured in our lab, which is a typical clustered indoor environment. Two Realsense cameras are mounted on a rectangle box. one is facing forward and the other facing downward. The box is hand carried and moved in a rectangle path. We assume the two cameras are perpendicular to each other according to the installation. Fig. 5 shows the color image samples from both forward-facing camera and downward-facing camera.



(a) Forward image sample (b) Downward image sample

Figure 5: The indoor environment for experiments

Fig. 6 shows the octomap built from the depth cloud of forward camera. The designed filters can effectively remove noise and make the map clearer. The noise is significant because Realsense relies on a laser projector to generate infrared pattern, which is not very stable under indoor light condition. We can remove them assuming the depth continuity of the obstacle. This assumption is true because the depth is not negligible when the obstacle gets closer to the camera.



(a) Octomap without filtering



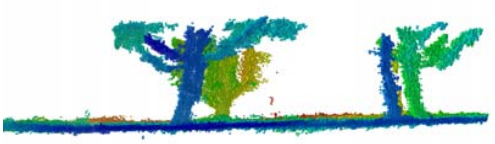
(b) Octomap after filtering

Figure 6: The octomap result

We also test the outdoor capability of our system in a typical canopy environment. The octomap is built using the same method as indoor, but the range in outdoor environment can be increased up to 10 meters. The outdoor environment and octomap result are shown in Fig. 7.



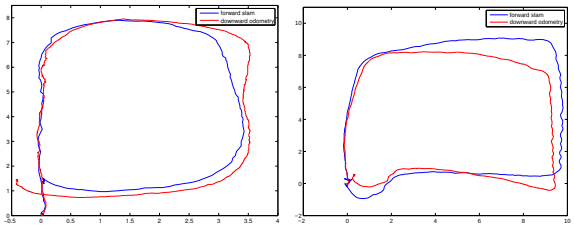
(a) Forward image sample (b) Downward image sample



(c) Octomap

Figure 7: The outdoor environment and octomap result

We compare the position outputs of two cameras excluding height. The result is shown in Fig. 8. The total length of the indoor camera movement is around 22m. The velocity output of downward facing camera is integrated as position. Through multiple tests, We observe that the total position error (the position difference of starting point on the loop) of ORB-SLAM is less than 20cm in average. The position error of downward facing camera accumulates and is a bit larger, but the velocity estimation is enough to stabilize the MAV. The outdoor test shows a similar result. We observe that Realsense camera can provide a better depth map in the good light condition of outdoor texture-rich environment. A fix exposure time performs better than auto exposure when the light condition varies in the canopy.



(a) Indoor result (b) Ourdoor result

Figure 8: The position output

7 CONCLUSIONS

In this paper, we present a localization and mapping system with dual Realsense cameras. The downward facing camera provides velocity estimation for stabilization of the

MAV. At the same time, the forward facing camera provides less-drift position estimation for navigation. We demonstrate the effectiveness of our system in both indoor and outdoor environment. In future work, we will test the system on our quadrotor platform and perform vision based autonomous flight.

REFERENCES

[1] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pages 155–160. IEEE, 2011.

[2] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems Conference (RSS)*, 2014.

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

[4] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmentła modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 1999.

[5] Felix Endres, Jurgen Hess, Jurgen Sturm, Daniel Cremers, and Wolfram Burgard. 3-d mapping with an rgb-d camera. *Robotics, IEEE Transactions on*, 30(1):177–187, 2014.

[6] Raúl Mur-Artal and Juan D Tardós. Fast relocalisation and loop closing in keyframe-based slam. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 846–853. IEEE, 2014.

[7] Raúl Mur-Artal, JMM Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *Robotics, IEEE Transactions on*, 31(5):1147–1163, 2015.

[8] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *Robotics, IEEE Transactions on*, 28(5):1188–1197, 2012.

[9] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[10] David Caruso, Jakob Engel, and Daniel Cremers. Large-scale direct slam for omnidirectional cameras. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 141–148. IEEE, 2015.

[11] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1935–1942. IEEE, 2015.

[12] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*, 2011.

[13] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *International Symposium on Robotics Research (ISRR)*, pages 1–16, 2011.

[14] Adam Bry, Charles Richter, Abraham Bachrach, and Nicholas Roy. Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments. *The International Journal of Robotics Research*, page 0278364914558129, 2015.

[15] Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Autonomous visual mapping and exploration with a micro aerial vehicle. *Journal of Field Robotics*, 31(4):654–675, 2014.

[16] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor mav. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4557–4564. IEEE, 2012.

[17] Abraham Bachrach, Samuel Prentice, Ruijie He, Peter Henry, Albert S Huang, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Estimation, planning, and mapping for autonomous flight using an rgb-d camera in gps-denied environments. *The International Journal of Robotics Research*, 31(11):1320–1343, 2012.

[18] Korbinian Schmid, Philipp Lutz, Teodor Tomić, Elmar Mair, and Heiko Hirschmüller. Autonomous vision-based micro air vehicle for indoor and outdoor navigation. *Journal of Field Robotics*, 31(4):537–570, 2014.

[19] Ji Zhang, Michael Kaess, and Sanjiv Singh. A real-time method for depth enhanced visual odometry. *Autonomous Robots*, pages 1–13, 2015.

http://www.imavs.org/

An Ultra-Wideband-based Multi-UAV Localization System in GPS-denied environments

Thien Minh Nguyen, Abdul Hanif Zaini, Kexin Guo and Lihua Xie

http://www.imavs.org/

ABSTRACT

This paper presents a technique for multi-UAV localization using ranging measurements from Two-Way Time-Of-Flight Ultra-Wideband (UWB) transceivers. In continuation with our previous work, the use of the Extended Kalman Filter (EKF) estimate is extended by fusing with other sensors to achieve a usable altitude estimate. Besides EKF, another method based on Non-linear Regression (NLR) is also developed to serve as an auxiliary localization to supplement the EKF. Experiments of autonomous flights are carried out to study the performance of these localization methods. Our success in supporting 4 UAVs demonstrates the capability of UWB for multi-UAV localization as a substitute for GPS in GPS-denied conditions. It can also serve as a cost-competitive alternative for very accurate yet expensive, highly centralized motion capture systems for indoor localization.

1 INTRODUCTION

Micro unmanned aerial vehicle (UAV) technology has plenty of potential and is seeing increasing application in recent years. Besides the well-reported military applications and the ubiquitous aerial photography consumer drones, UAVs are also being used for 3D mapping [1], aerial inspection [2], precision agriculture [3], as well as search and rescue [4]. Other noticeably interesting applications are also being explored such as deployment in indoor industrial environments [5], and even interception and capture of other UAVs [6].

Most of these applications rely heavily on the satellite-based localization system (Global Positioning System, GPS) which limits it to outdoor use. The system would thus fail to work indoors or in semi-cluttered urban environments and forests due to multipath and non-line-of-sight blockage. Hence, a satisfactory localization solution is desired to extend the use of UAV into complex environments.

The authors are with School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore. email: E150040@ntu.edu.sg (T.M. Nguyen); E150010@e.ntu.edu.sg (A.H.B. Zaini); GUOK0005@e.ntu.edu.sg (K. Guo); elhxie@ntu.edu.sg (L. Xie)

The research was partially supported by the ST Engineering - NTU Corporate Lab through the NRF corporate lab@university scheme.

For indoor localization, two infrastructure-dependent systems have attracted much attention from UAV research groups: a camera-based motion capture system, and a system that utilizes the existing wireless local area network. The camera-based motion capture system is commonly used by several research groups focusing on challenging autonomous aerial tasks such as formation, cooperative control and high speed maneuvers [7, 8]. Commercially-available motion capture systems are able to provide millimeter/degree accuracy and high update rates at upwards of 200Hz. However, the motion capture system is a highly centralized system utilizing many cameras connected to a single computer which then transmits the positions and orientations to the subjects within its view. It is also costly, has significant setup time, small sensing area, and can only be used indoors which limits use beyond research.

The second method utilizes the IEEE 802.11 standard usually referred to as WiFi. It has been used in commercial and residential buildings [9] and robot teams [10]. However, to date, no successful UAV flight using WiFi localization has been reported. This can be attributed to the unsatisfactory accuracy of WiFi-based localization methods for a demanding application such as UAV [11].

Recently, a third method for indoor localization has emerged which utilizes the pulse-based ultra-wide band (UWB) radio technology. The large bandwidth and pulse-based communication enables spectrum sharing and does not interfere with conventional signals. With the large bandwidth, this technology has the properties of strong multi-path resistance and, to some extent, wall-penetration, which enables accurate ranging via peer-to-peer two-way time of flight. With centimeter-level ranging accuracy, small size and light weight, low-power UWB modules can be applied for UAV localization [12]. UAV localization has also been achieved using time of arrival and time difference of arrival with one way UWB communications [13]. The work reported in this paper builds on the group's previous work in [12].

In this paper, PulsON 440 UWB modules from Time Domain [14] are installed on multiple quadcopters for localization in GPS-denied environments. Each UAV sends range requests to anchor nodes in a round-robin manner. Once a range measurement is retrieved, based on our proposed localization algorithm, position and velocity estimates will be updated and fed into the flight control unit of the quadcopters. A combination of Nonlinear Regression (NLR) and extended Kalman filter (EKF) localization methods is proposed where NLR can be used to initialize and reset the EKF estimates. Also, due

to the significance of the accuracy of anchors' coordinates for our localization estimation, a tedious procedure of measuring the anchors' coordinates must be conducted every time the operation sites are altered or the anchors are moved. Therefore, the proposed NLR method can also be used to assign the coordinates to the anchors. An autonomous formation flight test with a four-UAV team are also conducted and the data were studied to validate the performance of our proposed localization system,.

The content of this paper is organized as follows. In the first part we formulate the Extended Kalman Filter and Nonlinear Regression algorithms that use the distances to produce the locations of the agents. In the second part we describe our experiments where we fly 4 UAVs using the EKF estimates while running the other method simultaneously and summarize their performance. Finally, we conclude on the performance of the UWB-based localization system and its capability to support multiple UAVs as well as discuss issues that can be investigated for future works.

2 LOCALIZATION TECHNIQUES

2.1 Extended Kalman Filter

In this paper the Extended Kalman Filter is based on our previous work [12]. Let p be the aircraft's position, $v = \dot{p}$ be its velocity, $w(t)$ is a white noise process ($w(t) \in \mathbb{R}^3$). The UAV's motion is modelled as a random acceleration second-order system as follows:

$$X = \begin{bmatrix} p \\ v \end{bmatrix}; \quad \dot{X} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} X(t) + \begin{bmatrix} 0 \\ I \end{bmatrix} w(t)$$

The model above is discretized as:

$$X_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix}; \quad X_{k+1} = A_k X_k + w_k$$

where:

$A_k = \begin{bmatrix} I & \Delta t_k I \\ 0 & I \end{bmatrix}$ is a 6×6 discrete state transition matrix, Δt_k is the time difference between the last update and the current update. $\{w_k\}$ is a white noise sequence with its covariance derived in [12] as:

$$Q_k = \begin{bmatrix} \frac{1}{4}(\Delta t_k)^4 & \frac{1}{2}(\Delta t_k)^3 \\ \frac{1}{2}(\Delta t_k)^3 & (\Delta t_k)^2 \end{bmatrix} \otimes \mathrm{diag}(\sigma_x^2, \sigma_y^2, \sigma_z^2)$$

Q_k is a 6×6 covariance matrix for the random acceleration sequence. Here $\otimes$ is the Kronecker product and $\mathrm{diag}(\sigma_x^2, \sigma_y^2, \sigma_z^2)$ is a 3×3 diagonal matrix whose non-zero entries are the variance of the random acceleration on each direction.

We assume that the measurement is the distance from an anchor (whose position is denoted as p^a) to the UAV's position subject to a zero-mean Gaussian noise ν_k with variance R: $d_k = \|p_k - p^a\| + \nu_k$. The EKF algorithm will run through the following steps every time a new distance estimate is received:

$$\hat{X}^*_{k+1} = A_k \hat{X}_k \tag{1}$$

$$\hat{P}^*_{k+1} = A_k \hat{P}_k A_k^T + Q_k \tag{2}$$

$$\hat{d}_{k+1} = \left\| p^*_{k+1} - p^a \right\| \tag{3}$$

$$H_{k+1} = \left[\frac{1}{\hat{d}_{k+1}} (p^*_{k+1} - p^a)^T \middle| \mathbf{0}_{1\times 3} \right] \tag{4}$$

$$S_{k+1} = R + H_{k+1} \hat{P}^*_{k+1} H^T_{k+1} \tag{5}$$

$$K_{k+1} = \hat{P}^*_{k+1} H^T_{k+1} S^{-1}_{k+1} \tag{6}$$

$$\hat{X}_{k+1} = \hat{X}^*_{k+1} + K_{k+1} \left[d_{k+1} - \hat{d}_{k+1} \right] \tag{7}$$

$$\hat{P}_{k+1} = (I - K_{k+1} H_{k+1}) \hat{P}^*_{k+1} \tag{8}$$

The following Mahalanobis distance of each distance measurement is also calculated between step (5) and step (6):

$$D_M = \sqrt{(d_{k+1} - \hat{d}_{k+1})^T S^{-1}_{k+1} (d_{k+1} - \hat{d}_{k+1})}$$

If D_M is larger than a threshold then it is likely to be an outlier and we should skip fusing this measurement in step (7) and step (8).

In our previous work, the EKF estimate was used mainly as a 2D localization system. Therefore an independent laser-based range finding sensor was used for altitude estimate. Although the sensor is accurate, the UAV's altitude estimate is influenced by the terrain. In this work, we have combined the UWB altitude estimate with the low level sensors to achieve altitude control for UAV, independent from the variation of the terrain.

2.2 Substituting GPS in low level fusion

In our platforms, we utilize the high-level EKF estimate as the GPS input for the sensor fusion algorithm of the 3DR Pixhawk [15]. Our Pixhawk units run a customized version of the open-source PX4 [16] firmware and are set to use the 22-state EKF attitude and position estimator. In this 22-state EKF, the position, velocity, and attitude states are predicted by integration of acceleration and angular rate measurements from the IMU. In the correction step, GPS position measurement inputs are substituted and the relevant noise parameters are adjusted accordingly.

To ensure reliable and stable altitude estimation, the altitude data (received at a rate nearly 32Hz) are fused with the existing barometer-based altitude readings (sampled at 100Hz). The standard deviations of both measurements were found to be similar when held stationary at 18cm and 20cm respectively. However, the slower measurement rate of the noisy UWB-based altitude measurements does not provide the simultaneously smooth and responsive estimate required for altitude control. Nevertheless, this estimate still has the advantage that it is not subject to the variation of the ground level. As the barometer altitude reading is an absolute value

http://www.imavs.org/

measured from sea level, an offset is subtracted from the barometer measurements. This offset is obtained by calculating the difference between barometer measurement and the EKF altitude estimate every 10 seconds as shown in Figure 1. This arrangement simultaneously compensates for barometer drift and dampens sudden changes in UWB-based altitude measurement.

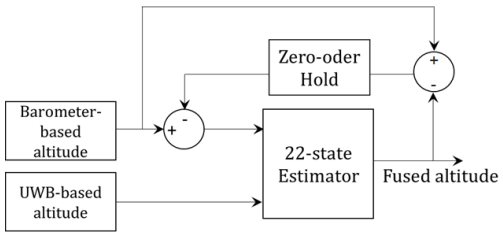


Figure 1: The method used for offsetting the barometer.

2.3 *Non-linear regression for coordinates assignment and localization*

Since our localization method requires setting up the anchors at the operation site, it usually would be very cumbersome, or sometimes impossible, for human to access the operation area to measure the coordinates of the anchors before the UAV team can be deployed. In this part we propose a method based on Non-linear regression to assign the coordinates to the anchors/UAVs on startup. This method can be implemented based on the communication capability of the UWB transceivers in the future.

Besides coordinates assignment, the algorithm can also be run in parallel with the EKF to serve as initializer and recovery value. A secondary localization technique is useful in this situation because there can be cases when the signal is lost for sometime (for example, 1 second), the step time Δt_k can be too large and cause the EKF to diverge. Also, since we assume a random acceleration model, when the UAV moves faster than the estimator can catch up, an over-conservative outlier rejection can cause it to stall completely. The NLR method does not require the past estimate unlike the EKF. Therefore, if enough distance measurements can be obtained in a short period, we can use the NLR to calculate the position and reset the EKF. In the Experiments section the performance of the NLR localization will be summarized and compared with the EKF.

Assume that we have a network of N agents, which can be either anchor or UAV, and each agent has its coordinates denoted as $p_i = \begin{bmatrix} x_i\ y_i\ z_i \end{bmatrix}^T$, $i = 1, 2...N$. An agent can have none, some or all of its coordinates known beforehand. In the context of a non-linear regression problem, the known coordinates will be in the set of error-free independent variables $\mathbf{x}$ and the unknown will be in the set of parameters β to be estimated. Regardless of their roles we will group these coordinates into one vector of length $3N$ as $\boldsymbol{p} = \begin{bmatrix} x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ \dots\ x_N\ y_N\ z_N \end{bmatrix}^T$

Let $d_{ij} = \|p_i - p_j\|$ be the distance between p_i and p_j, where $i, j = 1, 2, 3...N$ and $i \neq j$. Also since $d_{ij} = d_{ji}$ we only consider d_{ij} of $i < j$. The number of distances from all N agents in the network will be $K = N(N-1)/2$. Our non-linear model, denoted as $f(\mathbf{x}, \beta)$ or $f(\boldsymbol{p})$, will be a $K \times 1$ vector of all of the distances arranged in a particular way as follows:

$$\begin{bmatrix} d_{12} & d_{13}\ \dots\ d_{1N} & d_{23} & d_{24}\ \dots\ d_{2N}\ \dots & \dots\ d_{N-1,N} \end{bmatrix}^T$$

Our observed dependent variables y_{ij} are these distances subject to some error ϵ_{ij}, or $y_{ij} = d_{ij} + \epsilon_{ij}$.

To apply the Non-linear regression method, we start from some initial guess β_0 and seek to improve our estimate of β by the following recursive steps:

$$J_k = J_k(\mathbf{x}, \beta_k) = \left.\frac{\partial f(\mathbf{x}, \beta)}{\partial \beta}\right|_{\beta=\beta_k} \tag{9}$$

$$\Delta y = y - f(\mathbf{x}, \beta_k) \tag{10}$$

$$\Delta\beta = (J_k^T W J_k)^{-1} J_k^T W \Delta y \tag{11}$$

$$\beta_{k+1} = \beta_k + \Delta\beta \tag{12}$$

where $k = 0, 1, 2, 3...$ and W is a weight matrix to signify the comparative importance between the measurements.

During the initial phase when all agents are static W can be set as identity. In the second phase when some agents assume the role of anchors and some are the mobile nodes, the distances may not be acquired at the same time. The more recent a distance is, the more priority should be given to its residue Δy_i. Also, it should be noted that a time window should be applied to ignore the very old measurements. For example in a period from t_0 to t_m within our time window, we have measured the distances to m anchors at times t_1, $t_2, ...t_m$. W can be set as:

$$W = \frac{1}{\sum_{l=1}^{m} t_l - m t_0} \mathrm{diag}([\Delta t_1 \Delta t_2 \dots \Delta t_m])$$

where $\Delta t_l = t_l - t_0, l = 1, 2, ...m$.

To calculate the Jacobian J_k in equation (9) conveniently, we can employ a more expressive Jacobian $J_{\boldsymbol{p}} = \partial f(\mathbf{x}, \beta)/\partial \boldsymbol{p} = \partial f(\boldsymbol{p})/\partial \boldsymbol{p}$. $J_{\boldsymbol{p}}$ can be expressed as follows:

$$J_{\boldsymbol{p}} = \begin{bmatrix} \bar{J}_{12} & \bar{J}_{21} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \bar{J}_{13} & \mathbf{0} & \bar{J}_{31} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \bar{J}_{1N} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \bar{J}_{N1} \\ \mathbf{0} & \bar{J}_{23} & \bar{J}_{32} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \bar{J}_{24} & \mathbf{0} & \bar{J}_{42} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \bar{J}_{2N} & \mathbf{0} & \mathbf{0} & \dots & \bar{J}_{N2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \bar{J}_{N,N-1} \end{bmatrix} \tag{13}$$

http://www.imavs.org/

http://www.imavs.org/

where $\bar{J}_{ij} = \frac{(p_i - p_j)^T}{d_{ij}}$; $i, j \in \{1, 2, 3...N\}$ and $i \neq j$. $\mathbf{0}$ indicates a 1×3 block of zeros.

After we have substituted the values of $\mathbf{x}$ and β_k to (13), we can remove the columns corresponding to $\mathbf{x}$ to obtain J_k. For UAV localization, since only the UAVs are moving, all of the anchors' coordinates can be added to $\mathbf{x}$ so that the corresponding columns can be removed from $J_{\boldsymbol{p}}$. The same treatment can also be done to the rows corresponding to the distances between the anchors. Finally if the UAV only needs to localize itself then the final J_k matrix will be a $K^* \times 3$ matrix where K^* is the number of anchors.

Using NLR for coordinates assignment requires fixing some coordinates before estimating the others. The following lemma can provide several necessary conditions to assure the computational feasibility of the NLR algorithm.

Lemma 1. *Let l, m, n be the numbers of fixed x, y, z coordinates, respectively. The following are necessary conditions for J_k to have full rank:*

1. $(l + m + n) \geq 6$.
2. *The known coordinates must belong to at least three agents.*
3. $l \geq 1$, $m \geq 1$, $n \geq 1$.
4. *Any two in three numbers l, m, n must not be both equal to 1.*

Remark: Lemma 1 only specifies some conditions for the computational feasibility of the algorithm regarding the number of known coordinates. Some numerical instances of β_k may still result to loss of rank of J_k, especially when the agents are co-linear or coplanar. Therefore a check for infinity or NaN value in $\Delta\beta$ is still necessary. Also, since NLR is a gradient descent method, the solution may converge to a non-global optimum, therefore a good initial guess β_0 is also very important.

3 EXPERIMENTS AND RESULTS

3.1 Testbed

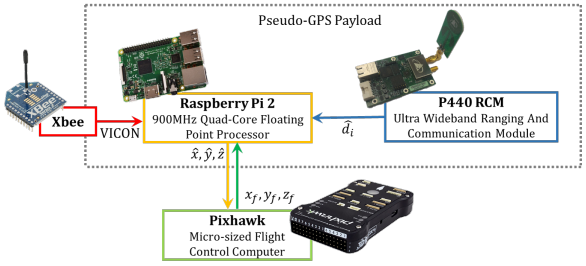


Figure 2: Main modules of the testbed.

In our experiment, the UWB transceivers in use are the P440 modules from Time Domain. There are four P440s acting as the anchor nodes and another four are mounted on the UAVs to range to the anchors. Each UAV has a high-level control board who hosts and directs the UWB modules to collect the distances to anchors.

The high level board then computes the location and sends the position estimates to the low-level board which fuses this data with its high-rate IMU and barometer sensors. The output is then sent back to the high level board at a lower rate for logging and off-line analysis.

3.2 Indoor autonomous flight

The indoor flight test was carried out using VICON as ground truth. 4 UAVs were loaded with predefined set points and set times. The set points were chosen to be on a $4m \times 4m$ square for the outer loop and an $2.5 \times 2.5m$ square for the inner loop. The anchors are positioned at the corners of a $6m \times 6m$ area with varying heights. After sending an RC signal to trigger the autonomous flight mode, all UAVs will register the trigger time. Each will then load the first set point from memory, fly to this point, stay there until the local clock exceeds the set time and then proceed to the next point. The ground truth was processed by a central computer and sent to each UAV to be stored locally. In Figure 5, all of the flight data were synchronized to the VICON system time by using time stamps and assuming that the clocks of P440, VICON, high-level and low-level controller boards have negligible drift during the experiment period.

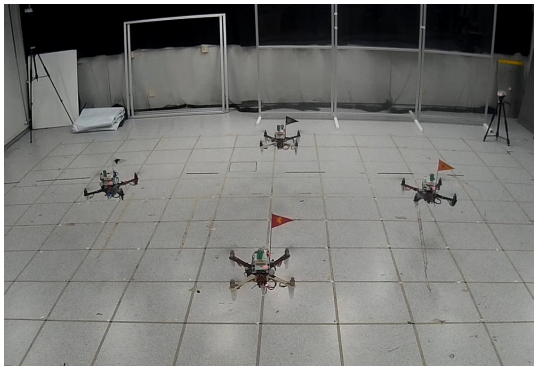


Figure 3: 4 UAVs at the closest set points (roughly $1.75m$ apart from each other) in the indoor test. Each square on the floor has an area of $0.5m \times 0.5m$

In Figure 4, it can be seen that x and y estimates from UWB-based localization methods have consistent accuracy across all UAVs, as the same level of error can be seen at the same area on the four plots. All of the RMS errors of the implemented methods are summarized in Table 1. For 2D localization analysis, which we only compare x and y estimates with VICON data, among all methods and UAVs, the maximum RMS error is 0.162m. The NLR method also notably

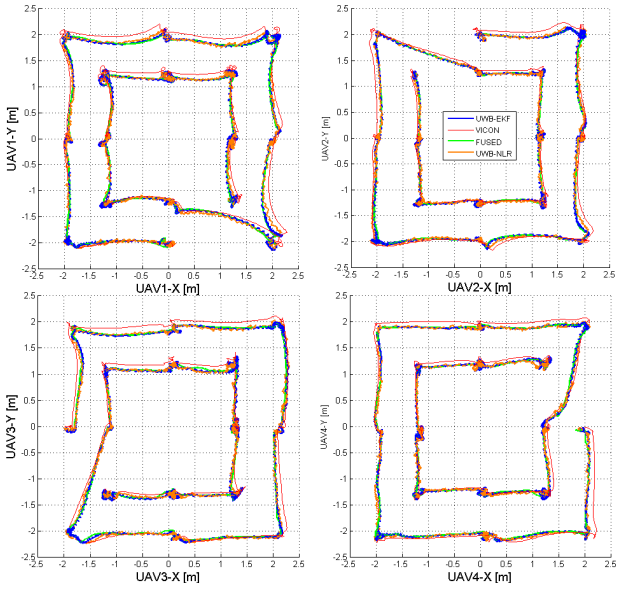


Figure 4: x and y estimates of 4 UAVs with VICON ground truth in red. EKF estimate is in blue, NLR orange and the EKF estimate fused with Pixhawk's sensors in green.

shows a slightly smaller error on all of the UAVs. The fused estimate appears to have a slightly larger error in comparison with the raw UWB-EKF estimate which can be explained because the data were down-sampled from 250Hz to 20Hz to ease the logging and analysis. The latency from transmitting the UWB-based estimate to px4 is also a factor of this larger error. This error is negligible for 2D movements of the UAVs.

For altitude estimate, the error is obviously more prominent than x and y estimates. The desired altitude is set at 1m for all the UAVs. The maximum RMS error is 0.354m from the fused estimate. It can be seen in Figure 5 that the error is location-wise consistent: any UAV will read the same altitude at the same place and the error level can act like an offset at that place. This effect thus does not forestall stabilization of altitude. On the other hand, it can be seen that the NLR altitude estimate has very large variation. It can suddenly "flip" at some points which can be explained as the algorithm converges to a value that is not globally optimal due to the imperfection of the distance as well as error in the declared position of the anchors.

Incompatible accuracy in the altitude estimate is a persistent issue in similar works [13]. For future work we may consider fusing external IMU data with the EKF/NLR estimates on the high level board to mitigate their respective shortcomings. If the application requires 2D localization only, x and y estimates of either method can be made into use. For 3D localization, the experiment shows that the EKF altitude estimate fused with Pixhawk's sensor is stable enough for flying. Though there is still significant altitude error, it is consistent among all UAVs and locations. This characteristics may offer a possibility for improvement in future works, for example the altitude error can be mapped out and compensated based on the 2D localization estimates.

UAV	EKF		FUSED		NLR	
	XY	**Z**	XY	**Z**	XY	**Z**
1	0.124	**0.353**	0.127	**0.354**	0.119	**0.346**
2	0.111	**0.343**	0.126	**0.349**	0.105	**0.327**
3	0.150	**0.335**	0.162	**0.338**	0.144	**0.336**
4	0.107	**0.348**	0.110	**0.347**	0.102	**0.341**

Table 1: RMS error of the indoor flight test for each UAV (all data are in m).

4 CONCLUSION

In this paper, the capability of the proposed technique for multi-UAV localization has been demonstrated by supporting up to 4 UAVs. A secondary localization method based on NLR was also investigated to work together with the EKF for initialization and reset. For 2D localization, both Extended Kalman Filter and Nonlinear Regression methods can provide sufficiently accurate estimate for UAVs. When altitude estimate is taken into account, the EKF estimate has been successfully tested for flying while the NLR method still has some problems with its altitude estimate. Nevertheless, the two methods are relatively independent and can both run in parallel to supplement each other.

REFERENCES

[1] F. Nex and F. Remondino, "Uav for 3d mapping applications: a review," *Applied Geomatics*, vol. 6, no. 1, pp. 1–15, 2014.

[2] "Application of robotics in onshore oil and gas industry—a review part i," *Robotics and Autonomous Systems*, vol. 75, Part B, pp. 490 – 507, 2016.

[3] C. Zhang and J. M. Kovacs, "The application of small unmanned aerial systems for precision agriculture: a review," *Precision Agriculture*, vol. 13, no. 6, pp. 693–712, 2012.

[4] J. Qi, D. Song, H. Shang, N. Wang, C. Hua, C. Wu, X. Qi, and J. Han, "Search and rescue rotary-wing uav and its application to the lushan ms 7.0 earthquake," *Journal of Field Robotics*, vol. 33, no. 3, pp. 290–321, 2016.

[5] Y. Khosiawan and I. Nielsen, "A system of uav application in indoor environment," *Production & Manufacturing Research*, vol. 4, no. 1, pp. 2–22, 2016.

[6] B. Zhu, J. Xu, A. H. B. Zaini, and L. Xie, "A three-dimensional integrated guidance law for rotary uav interception," in *2016 12th IEEE International Confer-*

http://www.imavs.org/

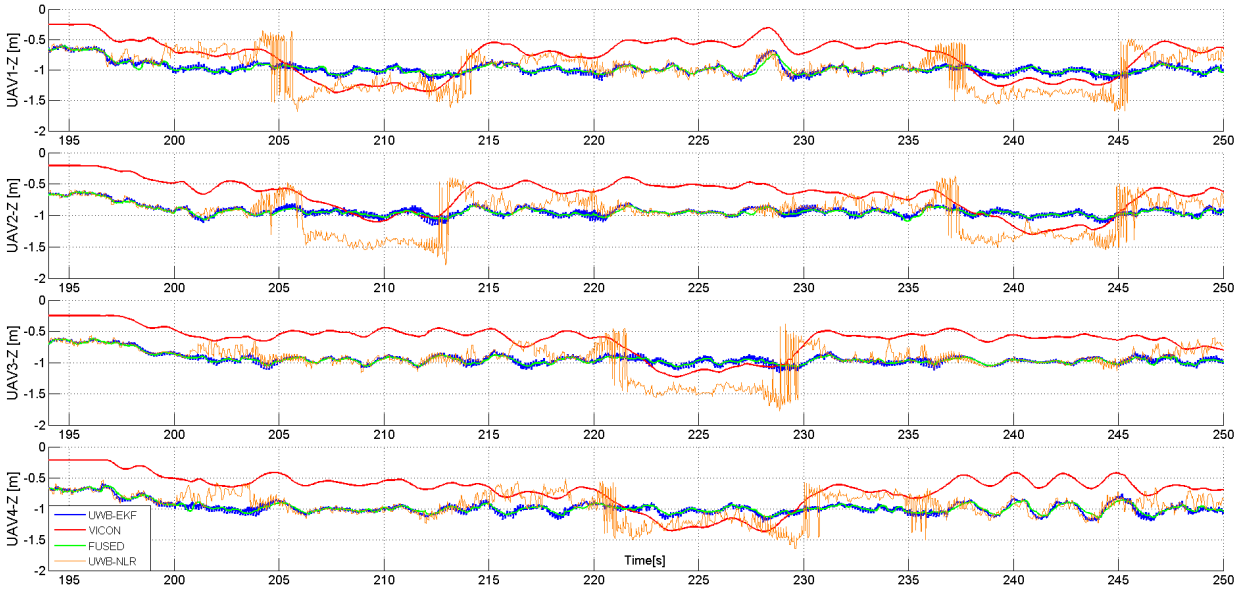


Figure 5: Concurrent z estimates of 4 UAVs for half of the flight test. The altitude is represented in NED convention, thus all values are negative. The ground-level measurement from UWB is 0.6m and the VICON height is about 0.25m. The UAVs take off at the 196 second and as the set value for altitude is -1m, the UWB-EKF and the fused estimates are maintained around this value by the control loop. For most of the time, the VICON altitude is lower than UWB-based estimate. However, for UAV 1, it can be seen that the error changes sign as it gets close to anchor 4 and anchor 2 at 206s to 214s and 238s to 245s respectively. These anchors are the ones with lower heights. The "flipping" effect of NLR estimates also occurs in these locations. The same effects are also observed on other UAVs.

http://www.imavs.org/

ence on Control and Automation (ICCA), June 2016, pp. 726–731.

[7] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[8] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, "A platform for aerial robotics research and demonstration: The flying machine arena," *Mechatronics*, 2014.

[9] B. Balaji, J. Xu, A. Nwokafor, R. Gupta, and Y. Agarwal, "Sentinel: occupancy based hvac actuation using existing wifi infrastructure within commercial buildings," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2013, p. 17.

[10] B. Benjamin, G. Erinc, and S. Carpin, "Real-time wifi localization of heterogeneous robot teams using an online random forest," *Autonomous Robots*, vol. 39, no. 2, pp. 155–167, 2015.

[11] Y. Sun, M. Liu, and M. Q. H. Meng, "Wifi signal strength-based robot indoor localization," in *Information and Automation (ICIA), 2014 IEEE International Conference on*, July 2014, pp. 250–256.

[12] K. Guo, Z. Qiu, C. Miao, A. H. Zaini, C.-L. Chen, W. Meng, and L. Xie, "Ultra-wideband-based localization for quadcopter navigation," *Unmanned Systems*, vol. 4, no. 01, pp. 23–34, 2016.

[13] A. Ledergerber, M. Hamer, and R. D'Andrea, "A robot self-localization system using one-way ultra-wideband communication," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 3131–3137.

[14] Time Domain PulsON 440. [Online]. Available: http://www.timedomain.com/products/pulson-440/

[15] 3DR Pixhawk. [Online]. Available: https://store.3dr.com/products/3dr-pixhawk

[16] PX4 Development Guide. [Online]. Available: http://dev.px4.io/

Applicability of Vector Field Histogram Star (VFH*) on Multicopters

R.J. van Breda* and W.J. Smit†
Department of Mechanical and Mechatronic Engineering
Stellenbosch University
Stellenbosch, South Africa

http://www.imavs.org/

ABSTRACT

This paper investigates the applicability of the Vector Field Histogram Star (VFH*) obstacle avoidance algorithm on multicopters using two-dimensional LIDAR systems. Modifications to the existing VFH* algorithm are made to account for position and yaw uncertainties of multicopters. The effects of tilt angles caused by pitch and roll manoeuvres are also addressed by modifying range measurements from the LIDAR system. The need for steering control and the effects of changes in the search depth of the VFH* algorithm are also investigated.

1 INTRODUCTION

The possible uses for multicopters are seemingly endless. From search and rescue operations, photography, surveying, emergency aid, and even drone racing it would seem their uses are limited only by our own imaginations. However, there are inherent dangers posed by multicopters. The loss of control over such a drone can easily result in injuries occurring from the falling mass or its spinning rotors. As safety concerns grow over the use of such devices the need for autonomous obstacle avoidance capabilities and autonomy have become essential.

Obstacle avoidance can either be done locally, globally or through a combination of the two. A local obstacle avoidance system uses real time sensor data to detect obstacles in the immediate vicinity of a robot and steers the robot clear of any obstacles encountered in the robot's path. On the other hand global obstacle avoidance system determines the most appropriate path for a robot to follow based on a map of the robot's known environment [1].

In 1989 Johann Borenstein and Yorem Koren suggested the Virtual Force Field (VFF) method which combined two existing concepts, that of certainty grids for obstacle representation and potential fields for navigation [2] . The method also made use of the Wall Following Method (WFM) to get out of minimum trap situations. For the WFM the robot follows the obstacles contour until it can resume its path to the target position [2] . In 1991 they suggested an improved algorithm called the Vector Field Histogram (VFH) method which uses a two-dimensional Cartesian histogram grid for a world model [3]. Onboard range sensors were used to update the world model and then a two-stage data reduction process carried out to compute the desired steering commands for the robot [3]. In the first stage the histogram grid is reduced to a one-dimensional polar histogram grid that is centered on the robot. In the second data reduction stage the area around the robot is divided into sectors each having its own polar obstacle density value and the most suitable sector is chosen for the robot to travel towards. Then, again in 1998 the VFH method was improved and appropriately named the VFH+ method. The VFH+ algorithm accounts for the robots width and approximates the available robot trajectories when selecting its optimal path [4]. This resulted in smoother robot trajectories and greater reliability. Then in 2000 the method was further enhanced and called the VFH* method [1]. The VFH* algorithm makes use of the A* search algorithm and appropriate cost and heuristic functions in order to verify whether candidate directions do in fact guide the robot around obstacles. This helps the robot to deal with situations that purely local obstacle avoidance algorithms such as the VFF, VFH, VFH+ do not take into consideration.

Our aim is to establish the applicability of the VFH* algorithm to multicopters since the algorithm was originally created with with mobile ground robots in mind. We also investigate what modifications can be made to better the algorithm for application on multicopters through simulations. We consider only the case where the multicopter would make use of a two-dimensional LIDAR system for range measurements.

In specific we investigate the effect of multicopter position and yaw uncertainties. The effects of pitch and roll movements on range measurements are also investigated and the effects of steering control and the VFH* algorithm's search depth in a multicopter capacity.

Section 2 provides a more detailed overview of the workings of the VFH* algorithm. Our suggested modifications are discussed in Section 3 with a conclusion on the applicability of the algorithm in Section 4.

2 THE VFH* ALGORTIHM

The VFH* algorithm makes use of a two-dimensional Cartesian histogram grid C for obstacle representation [3].

*Email address: 16546660@sun.ac.za
†Email address: wjsmit@sun.ac.za

http://www.imavs.org/

Each cell of the histogram grid C has its own certainty value $c(i, j)$. On-board range sensors are used to continuously and rapidly update the histogram grid in real time. Originally ultrasonic sensors were used but we propose the use of a two-dimensional LIDAR system for use on a multicopter because of its superior range capabilities compared to ultrasonic sensors . Each cells' certainty value $c(i, j)$ is updated as the robot moves. If an obstacle is detected the cell's certainty value is incremented otherwise its certainty value slowly decays away over time. Thus, if an obstacle is detected multiple times it will have a high certainty value while on the other hand random noise or misreadings will fade away with time [3].

VFH* makes use of a four stage data reduction process to reduce the data from the histogram grid C [4]. The aim of which is to obtain candidate directions in which the robot can move. Then the candidate directions are evaluated based on a cost function and look-ahead verification and the most suitable one is chosen as the robot's direction of motion [1].

2.1 First Stage - The Primary Polar Histogram

The first data-reduction stage maps the active region C_a from the histogram grid C onto the primary polar histogram, H^p [4]. The active region C_a is located around the robot's momentary location and has a radius of $r_{active-region}$. Each cell acts as an obstacle vector with a vector direction and magnitude [4]. The vector direction, $\beta_{i,j}$, is based on the direction of the active cell relative to the robot center point (RCP):

$$\beta_{i,j} = \arctan\left(\frac{y_o - y_i}{x_o - x_i}\right) \tag{1}$$

where:

x_o, y_o : Momentary coordinates of the RCP.

x_i, y_i : Coordinates of active cell $C_{i,j}$.

The vector magnitude, $m_{i,j}$ of an active cell $C_{i,j}$ is in turn calculated as follows:

$$m_{i,j} = c_{i,j}^2(a - bd_{i,j}^2) \tag{2}$$

where:

$c_{i,j}$: Certainty value of active cell $C_{i,j}$.

$d_{i,j}$: Distance from active cell $C_{i,j}$ to the RCP.

The parameters a and b are chosen according to:

$$a - b\left(\frac{r_{active-region} - 1}{2}\right)^2 = 1 \tag{3}$$

Note that the certainty value, $c_{i,j}$, is squared when the vector magnitude is calculated. This means that recurring range readings will result in high certainty values and thus we can be more confident that an obstacle resides in the obstacle cell $C_{i,j}$. In contrast noise or once-off range readings do not result in high certainty values.

The distance between the active cell and RCP, $d_{i,j}$, is also squared when calculating the vector magnitude. This means that occupied cells will have greater vector magnitudes the closer they get to the robot.

To prevent the robot from cutting corners we have to compensate for the robot's width [4]. This is done by enlarging occupied cells by the robot radius, r_{robot}. Additionally, a minimum allowed distance between the robot and obstacle, d_{safety}, can be added. The resulting radius r_{r+s}, is then defined as $r_{r+s} = r_{robot} + d_{safety}$. By enlarging occupied cells by r_{r+s} the robot can be treated as a point-like vehicle [4].

The primary polar histogram, H^p, is constructed by dividing the active window surrounding the robot into sectors [4]. An arbitrarily chosen angular resolution, α, is used so that the number of sectors is always an integer, i.e. $n = 360°/\alpha$. In our simulations we chose $\alpha = 5°$. Each angular sector corresponds to a discrete angle $\rho = k \cdot \alpha$.

Instead of updating only the sectors in which occupied cells fall all histogram sectors that are affected by the enlarged occupied cells are updated while building the primary polar histogram. The enlargement angle $\gamma_{i,j}$ is calculated as follows:

$$\gamma_{i,j} = \arcsin\left(\frac{r_{r+s}}{d_{i,j}}\right) \tag{4}$$

The polar obstacle density for each sector k is calculated as follows:

$$H_k^p = \sum_{i,j \epsilon C_a} m_{i,j} \cdot h'_{i,j} \tag{5}$$

with:

$$h'_{i,j} = 1 \qquad \text{if} \quad k \cdot \alpha \epsilon \left[\beta_{i,j} - \gamma_{i,j}, \beta_{i,j} + \gamma_{i,j}\right] \tag{6}$$

$$h'_{i,j} = 0 \qquad \text{otherwise} \tag{7}$$

The h' function acts as a low-pass filter and smooths the polar histogram [4]. The resulting polar histogram grid considers the robot width.

Figure 1 shows a robot surrounded by obstacles. The primary polar histogram is drawn around the robot indicating sectors with high polar density values or in other words sectors that contain obstacles. The effect that the enlargement angle $\gamma_{i,j}$ has on the primary polar histogram is also indicated. The sectors to the sides of each obstacle that fall within the enlargement angle are also deemed blocked and assigned a polar density value using Equation 5. The sectors that are blocked due to the enlargement angle are the filled sectors as shown in Figure 2.

2.2 Second Stage - The Binary Polar Histogram

When a robot moves in an environment with several narrow openings these openings can alternate between open and blocked between consecutive readings [4]. This causes the robot to keep changing its steering direction between alternative openings. To get rid of this indecisive behaviour the polar histogram is reduced to a binary polar histogram.

The binary polar histogram H^b uses a hysteresis based on two thresholds τ_{low} and τ_{high} [4]. By using these threshold values the primary polar histogram H^p's sectors are reduced from polar density values to either being free(0) or

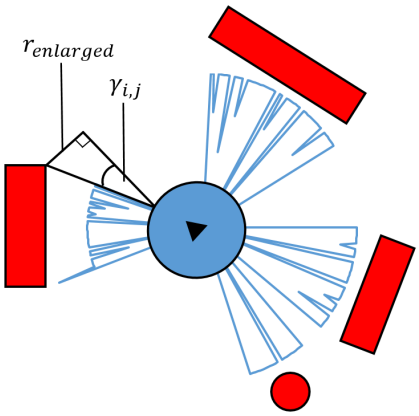


Figure 1: Primary polar histogram constructed around robot.

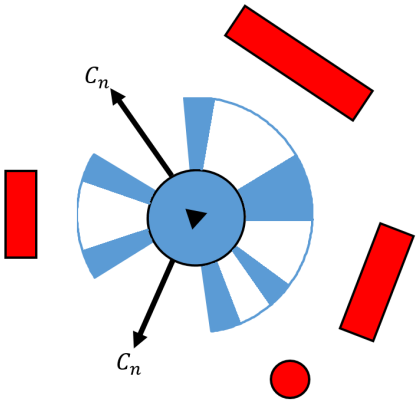


Figure 2: Binary polar histogram constructed around robot with two candidate directions.

blocked(1). The set of rules used to update the binary polar histogram H^b are shown:

$$H^b_{k,i} = 1 \qquad \text{if } H^p_{k,i} > \tau_{high} \tag{8}$$

$$H^b_{k,i} = 0 \qquad \text{if} \qquad H^p_{k,i} < \tau_{low} \tag{9}$$

$$H^b_{k,i} = H^b_{k,i-1} \qquad \text{otherwise} \tag{10}$$

This concept is illustrated in Figure 2 where the polar histogram from Figure 1 has been reduced to a binary polar histogram where the sectors are either open or closed.

2.3 *Third Stage - The Masked Polar Histogram*

VFH* approximates robot trajectories as circular arcs (constant curvature curves) to account for the fact that most mobile robots are unable to change their direction of motion instantaneously and are subject to turning circles [4]. The curvature of a curve is defined as $\kappa = 1/r$ where r is the steering radius. The values for minimum steering radii of a robot can be found experimentally and are functions of velocity. The steering radii to the right and left of the robot are defined as $r_r = 1/\kappa_r$ and $r_l = 1/\kappa_l$.

If the trajectory circle and an enlarged obstacle cell overlap all directions of motion are blocked to that side of the robot. The masked polar histogram shows which directions are accessible at the robot's current speed. If all sectors are blocked the robot would have to decrease its speed and reconstruct the masked polar histogram [4].

2.4 *Fourth Stage - Determining of Primary Candidate Directions*

By checking which sectors in the masked polar histogram are free and which are blocked primary candidate directions are identified [1, 4]. Then by evaluating each candidate direction at the hand of a cost function the new direction of motion is calculated [1].

The right and left borders, k_r and k_l, of all openings are determined to establish whether an opening is wide or narrow [4]. An opening is considered wide if the difference between its two borders is larger than s_{max} sectors ($s_{max} = 16$ in our simulations). In contrast if the difference between its two borders is less than s_{max} sectors the opening is considered narrow. Figure 2 shows a case where two narrow openings are present and two primary candidate directions exist.

A narrow opening has only one primary candidate direction which is the center of the opening:

$$c_n = \frac{k_r + k_l}{2} \qquad \text{centered direction} \tag{11}$$

A wide opening has at least two primary candidate directions, one to the right of its left border and one to the left of its right border [4]. When the opening presents a clear path to the target, that is when the target direction k_t lies between the two borders, the target direction k_t is also considered to be a primary candidate direction as shown:

$$c_r = k_r + \frac{s_{max}}{2} \qquad \text{towards the right} \tag{12}$$

$$c_l = k_l - \frac{s_{max}}{2} \qquad \text{towards the left} \tag{13}$$

$$c_t = k_t \qquad \text{if } k_t \,\epsilon\, [c_r, c_l] \tag{14}$$

The two primary candidate directions c_r and c_l make the robot follow the obstacle contour at a safe distance, while c_t leads the robot directly towards the target [4].

2.5 *Verification of Candidate Directions*

Look-ahead verification is used to verify which primary candidate direction is the most suitable direction of motion for the robot. VFH* does this by computing the new positions and orientations of the robot if it were to move a projected step distance d_s in each of the primary candidate directions [1] .

At every projected position a new primary polar histogram is constructed based on the histogram grid information available. Then the data reduction process as described

http://www.imavs.org/

previously is repeated. New candidate directions are found, called projected candidate directions. The process is repeated n_g times until finally we are left with a search tree of depth n_g, where the end nodes correspond to the total projected distance $d_t = n_g \cdot d_s$ [1] .

Every end node has a cost based on the path cost, which is the sum of the costs of the branches leading back to the start node, and a heuristic function that is implemented [1]. The primary candidate direction that leads to the end node with smallest total cost is then selected as the new direction of motion.

The larger d_t the greater the total look-ahead. If selected too high the algorithm is slowed down considerably while if its chosen to small the robot might not choose the best path going forward. Thus, selection of d_t is a trade-off between the speed and quality of the algorithm [1].

3 MODIFICATIONS TO VFH*

All multicopters are subject to inaccurate sensor data that cause uncertainties to arise with regards to their position and yaw estimations. The original VFH* algorithm does not currently take into consideration uncertainties in position and yaw estimation since the mobile ground robots that the algorithm was intended for have motor encoders and other mechanisms that can more accurately determine the robot's position and orientation (yaw). However, as will be shown, the VFH* algorithm can be modified to account for these uncertainties by updating the way in which the multicopter builds its histogram grid and primary polar histogram.

The VFH* algorithm is also modified to take into consideration a multicopter's pitch and roll angles. These parameters are important to consider when range measurements are taken since an obstacle may be closer than what is being measured due to the angle of measurement.

3.1 Additional Sensor Region

The current implementation of VFH* reacts to obstacles once they are within the multicopter's active region C_a. Though this makes sense for a mobile ground robot that makes use of short range ultrasonic sensors it is bothersome for multicopter system using a long range two-dimensional LIDAR system. The problem lies in choosing an appropriate value for the active region radius, $r_{active-region}$. If the region is too small the multicopter will not have sufficient time to react to obstacles. On the other hand if the region is too large the robot will start reacting to obstacles much earlier than what is necessary.

This could be addressed by tuning the thresholds when constructing the binary polar histogram as discussed in section 2.2. But this process would be a tedious task and may not always be reliable.

We present our solution in the form of an additional sensor region. The sensor region is used to update the histogram grid C and its size is chosen to accommodate the sensors range capabilities. In our simulations we chose a sensor region of radius $r_{sensor-region} = 20$m. This radius can be made much larger, especially for multicopters moving at high speeds. The size of the active region then determines when the multicopter will react to an obstacle. In our simulations we chose the active region to have a radius of $r_{active-region} = 5$m. Thus, the sizes of the two regions can vary as long as the following condition is satisfied:

$$r_{active-region} \leq r_{sensor-region} \tag{15}$$

This allows the detection of obstacles before the multicopter reacts to them. This characteristic is of utmost importance for the implementation of the suggested modifications in the following subsections. It will become clear why it is important to distinguish between these two regions as we proceed.

3.2 Adding Safety Distance to Range Data

The original VFH* algorithm made use of a safety distance d_{safety} when constructing the primary polar histogram as was shown in Section 2.1. Though this safety distance works well when the multicopter is moving around an object it has no effect on the initial distance between the multicopter and an obstacle. This is because the safety distance d_{safety} is used to calculate only the enlargement angle $\gamma_{i,j}$ in Equation 4 and is not accounted for in any way in terms of the direct distance between the multicopter and the obstacle. Using the original VFH* algorithm would mean that the multicopter would only react to an obstacle once the obstacle is within the multicopter's active region. This means that the safety distance is only considered as the multicopter moves around the obstacle. In general this works well but to ensure that the multicopter starts reacting at the appropriate distance from an obstacle we account for the safety distance d_{safety} when updating the histogram grid C as follows:

$$d_{enlarged} = d_{i,j} - d_{safety} \tag{16}$$

where:

$d_{i,j}$: Distance from active cell $C_{i,j}$ to the RCP.

By using $d_{enlarged}$ when updating the histogram grid instead of $d_{i,j}$ the multicopter will start reacting to an obstacle exactly a distance of d_{safety} sooner than it would when d_{safety} is only included in the enlargement angle $\gamma_{i,j}$. It is important to note that this improvement is only possible because we have a sensor region. This allows the histogram grid to be updated before an obstacle enters the multicopter's active region. The correction made in Equation 16 means that the multicopter will detect the obstacle a distance of d_{safety} sooner than normal. This however means that the effective distance between the multicopter's center point and the obstacle is:

$$d_{effective} = r_{active-region} + d_{safety} \tag{17}$$

3.3 Compensating for Multicopter Position Uncertainties

Most multicopters make use of GPS systems to determine their positions. Accurate GPS and vicon systems are often

http://www.imavs.org/

http://www.imavs.org/

very expensive and even then one can never be 100% certain of the multicpoter's position. This poses a problem for the VFH* algorithm since it constructs its primary polar histogram from the histogram grid C based on the multicopter's position on the histogram grid.

To account for the uncertainties in a multicopter's position we first acknowledge that the multicopter can be anywhere within a circular uncertainty region around its estimated position as shown in Figure 3. From the figure one can see that the multicopter would fit through the gap between the two obstacles if its position is 100% certain. But since that is unlikely one has to consider the uncertainty region around the multicopter which means that the multicopter might very well be on a collision course with one of the two obstacles. The radius of the uncertainty region may vary depending on the type and quality of sensor used as mentioned earlier. We define the radius of the position uncertainty region as $r_{position}$. The uncertainty region is accounted for in the VFH* algorithm by modifying the range readings from the two-dimensional LIDAR range sensor. This can be done by subtracting $r_{position}$ from the LIDAR range measurement when updating the histogram grid C:

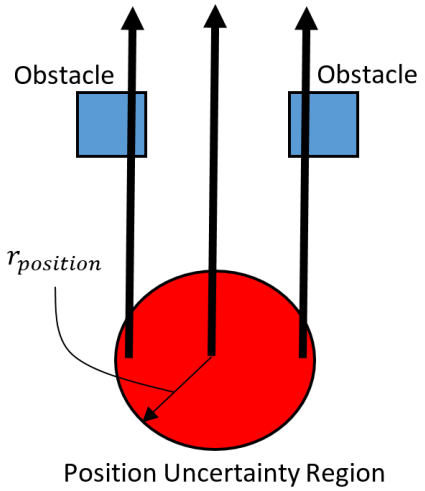


Figure 3: Multicopter position uncertainty.

$$d_{enlarged} = d_{i,j} - d_{safety} - r_{position} \tag{18}$$

where:

$d_{i,j}$: Distance from active cell $C_{i,j}$ to the RCP.

This enlarges the area an obstacle occupies in the histogram grid and allows the uncertainty region to be shifted from the multicopter to the obstacles as illustrated in Figure 4. This is however only half the solution as this does not enlarge the sides of the obstacle to account for the multicpoter's position uncertainty. To account for that we need to modify the enlargement angle $\gamma_{i,j}$ from Equation 4 used to construct the primary polar histogram:

$$r_{enlarged} = r_{multicopter} + d_{safety} + r_{position} \tag{19}$$

$$\gamma_{i,j} = \arcsin\left(\frac{r_{enlarged}}{d_{enlarged}}\right) \tag{20}$$

Thus, the multicopter will start reacting sooner to an obstacle to compensate for inaccuracies in its position estimation. The multicopter's position uncertainty is also compensated for when it wants to move around obstacles since we have included $r_{position}$ term in the enlargement angle $\gamma_{i,j}$ as shown in Figure 4. In so doing the multicopter's position uncertainty is taken into account when VFH* algorithm calculates candidate directions.

Figure 5 shows a simulation of the VFH* algorithm taking into consideration various position uncertainties. It can be observed that the multicopter reacts sooner when the uncertainty is more due to Equation 18 while the increase in the size of the arcs as the robot moves around the obstacle are the consequence of Equation 20. The effective distance between the multicopter and the obstacle where the multicopter starts reacting to the obstacle can be updated as follows:

$$d_{effective} = r_{active-region} + d_{safety} + r_{position} \tag{21}$$

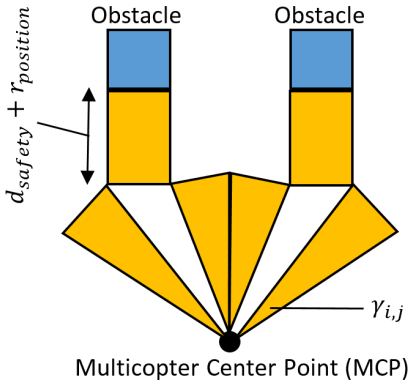


Figure 4: Enlarged obstacles with multicopter as point-like vehicle.

Figure 6 on the other hand shows the effect of three position uncertainties when the multicopter attempts to traverse terrain with clustered obstacles. The multicopter manages the shortest root when no uncertainty is present, $r_{position} = 0$m, while it is still able to move in between two obstacles which are slightly further apart when $r_{position} = 1$m. However when the uncertainty is increased even more to $r_{position} = 4$m the multicopter can no longer be certain that it won't collide with any of the obstacles and thus appropriately chooses to go around the cluster.

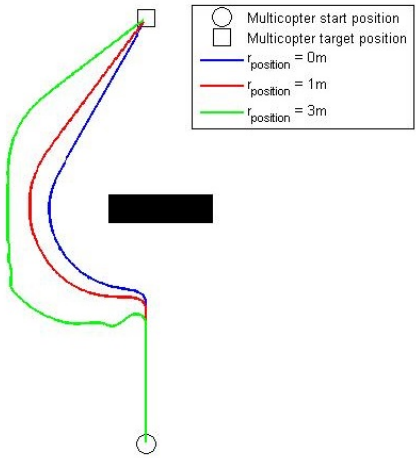


Figure 5: Effect of various position uncertainties on multicopter trajectory.

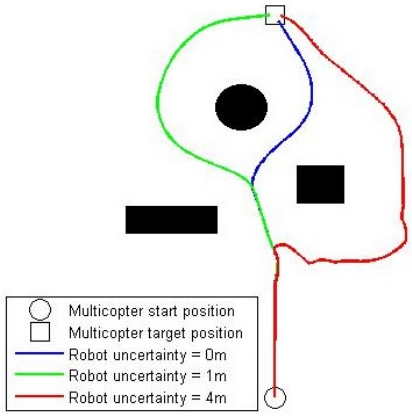


Figure 6: Effect of position uncertainty on multicopter trajectory in a cluttered environment.

3.4 *Compensating for Multicopter Tilt Angle*

One way in which multicopters are entirely different from mobile ground robots is that they use pitch and roll manoeuvres to move forward or to the sides. This also means that unlike mobile ground robots they do not have to change their orientations (yaw) to change direction as they can easily move in any direction through a combination of pitch and roll manoeuvres.

A potential problem that arises with implementing VFH* on a multicopter using a two-dimensional LIDAR range sensor is that if a gimbal is not used the tilt angle of the drone will affect the range measurements made by the sensor. Figure 7 shows a multicopter as seen from the side. If the multicopter is stationary while taking a range measurement the true distance to the object would be the distance R_1, ignoring sensor inaccuracies of a few centimetres. However, as the multicopter starts to move either towards the target, or away from it, a tilt angle θ_{tilt} is introduced and the sensor range measurement s would indicate that the object is further away from the multicopter than it actually is. This error is dependant on the obstacle distance to the multicopter and may even be insignificant if the multicopter travels at low speeds, causing small tilt angles (θ_{tilt}), or if the obstacle is very close to the multicopter, causing short range readings (s).

The effective range of an obstacle R_1 is easily calculated as:

$$R_1 = s \cdot \cos(\theta_{tilt}) \tag{22}$$

The concept is exactly the same for pitch and roll manoeuvres. Thus, the error caused by a multicopter's tilt angle during either pitch or roll manoeuvres can easily be corrected by calculating the effective range R_1 and using the effective range instead of the actual measured range s to update the histogram grid C. It is important to note that while the one end of the multicopter might be lifted by an angle θ_{tilt} the other end will be pointing down by an angle θ_{tilt} and every other range reading will be at an angle in the range $0° \leq \theta < \theta_{tilt}$. Figure 8 shows a simulation of the VFH* where the tilt angle

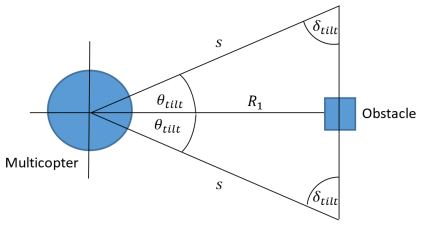


Figure 7: Effect of pitch angle on range readings.

was varied from $0°$ to $60°$ in increments of $15°$. It is important to note that the obstacle is shown in the position where the two-dimensional LIDAR sensor initially measures it to be before correcting the measured distance. Thus the obstacle would make up a region much closer to the multicopter on the histogram grid for each of the cases where the tilt angle is greater than zero. In all the cases the multicopter only reacts to the obstacle once it is detected in its active region. The effect of the tilt angle correction can clearly be seen as the multicopter starts reacting sooner on the figure as the tilt angle increases. As the tilt angle is increased the measured distance is corrected and the object is detected closer to the multicopter's starting position than what was initially measured.

3.5 *Compensating for Multicopter Yaw Uncertainties*

Uncertainties in a multicopter's estimated yaw angle effect the VFH* algorithm in a similar fashion as pitch and roll do. Figure 9 shows a top view of a multicopter with the yaw uncertainty, $\theta_{\Delta yaw}$, defined as the angle by by which the multicopter may differ from it's estimated yaw angle. It is impor-

http://www.imavs.org/

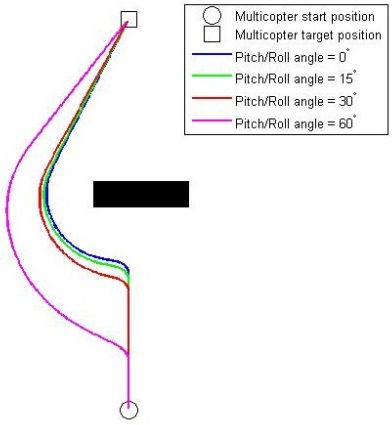


Figure 8: Effect of tilt angle on multicopter trajectory.

tant to note that we are referring to the uncertainty of the yaw angle and not the yaw angle itself. This is different from the roll and pitch angles in that errors in yaw angle estimation occur at random and have an influence on not only the perceived distance of an obstacle but also its perceived width.

To account for the effect yaw uncertainty has on the perceived distance of an obstacle to the multicopter we repeat the process followed in Section 3.4 as shown:

$$R_2 = s \cdot \cos(\theta_{\Delta yaw}) \tag{23}$$

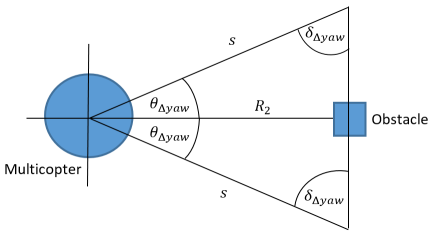


Figure 9: Effect of yaw angle on range readings and obstacle uncertainty.

R_2 is then the corrected distance from the multicopter to an obstacle. However if the multicopter has a tilt angle as shown in Figure 10 then we also need to account for θ_{tilt} when calculating the corrected distance, R_2:

$$R_2 = s \cdot \cos(\theta_{tilt}) \cdot \cos(\theta_{\Delta yaw}) \tag{24}$$

To account for the effect yaw uncertainty has on the width of an obstacle we include Δx, as shown in Figure 10, in the enlargement angle $\gamma_{i,j}$ as shown:

$$\Delta x = R_1 \cdot \sin(\theta_{\Delta yaw}) \tag{25}$$

then

$$\Delta x = s \cdot \cos(\theta_{tilt}) \cdot \sin(\theta_{\Delta yaw}) \tag{26}$$

$$r_{enlarged} = r_{multicopter} + d_{safety} + r_{position} + \Delta x \tag{27}$$

$$\gamma_{i,j} = \arcsin\left(\frac{r_{enlarged}}{d_{enlarged}}\right) \tag{28}$$

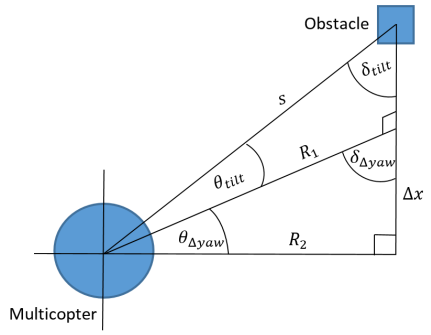


Figure 10: Combined effect of tilt and yaw angles on range readings and obstacle uncertainty.

3.6 *Effects of Steering Control and Search Depth*

Since multicopters do not have to change their orientation (yaw angle) to change their direction of motion the need for steering control in our simulations was questioned. Figure 11 shows the effect of different proportional steering control values on the VFH* algorithm. In this case having no steering control, $K_s = 0$, still produces the correct multicopter trajectory. However, this would only be a valid trajectory if the multicopter is travelling at low speeds. At higher speeds dynamic effects such as the multicopter's momentum would make it impossible to change its direction of motion instantaneously. Thus, if steering control is ignored while the multicopter is moving at moderate to high speeds the multicopter will behave differently than what the VFH* algorithm expects and can cause the multicopter to crash or hit an obstacle. In another observation it was found that the VFH* algorithm sometimes behaves in an unsatisfactory fashion when we neglect steering control from our simulations. The problem is illustrated in Figure 12 where the multicopter, without steering control, first moves to the right. As it moves to the right it does not immediately find a clear path to the target position and the path cost becomes increasingly expensive until the multicopter eventually turns around and moves left. In the end it does successfully move around the obstacle. The initial choice to go right is because we used a circular sensor and active region. This means that only a small part of the obstacle is initially detected. The detected part of the obstacle was inline with the multicopter and the target position as can be seen from the figure. Thus, the choice to go right was most likely made arbitrarily since not enough information was available.

http://www.imavs.org/

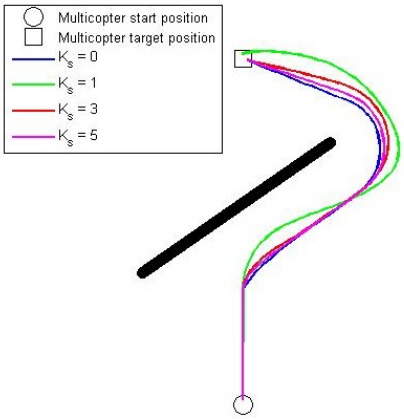


Figure 11: Effect of steering control proportional constant K_s on multicopter trajectory.

When steering control is included in the algorithm the multicopter cannot change its path so easily and commits to moving right until it can move around the obstacle. The reason for this is that if we do not consider steering control the multicopter can change its direction of motion instantaneously even though the cost functions used by the original VFH* does make it expensive for the multicopter to make large changes in its direction of motion. In the above described scenario the look-ahead verification search depth was chosen to be $n_g = 5$. If this is increased to $n_g = 10$ the VFH* algorithm determines that the least expensive path would in fact be to the left and makes the correct decision from the start as shown in Figure 13 since it has more information to make an informed decision.

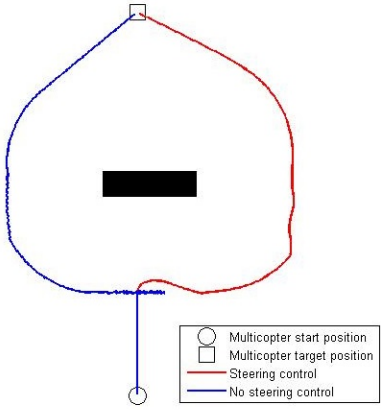


Figure 12: Effect of steering control on multicopter trajectory.

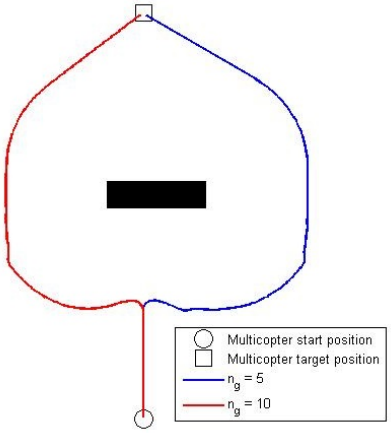


Figure 13: Effect of various search depths on multicopter trajectory.

4 CONCLUSION

This paper presented an overview of the VFH* algorithm and discussed the effects of position and yaw uncertainties. It also addressed the influence of tilt angles on the range measurement of obstacles when a two-dimensional LIDAR system is used. It was found that the VFH* algorithm is indeed applicable to multirotors and can be modified to account for position and yaw uncertainties. The algorithm can also be modified to consider the tilt angle of a multirotor as it builds its histogram grid with range measurements for a two-dimensional LIDAR system. It was also shown that the VFH* can still benefit from steering control and that the algorithm's search depth n_g is still an important parameter when the best path around obstacles needs to be determined.

REFERENCES

[1] I. Ulrich and J. Borenstein. VFH*: Local obstacle avoidance with look-ahead verification. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, 3:2505–2511, 2000.

[2] Johann Borenstein and Yorem Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man and Cybernetics*, 19(5):1179–1187, 1989.

[3] Johann Borenstein and Yoram Koren. The Vector Field Histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[4] Iwan Ulrich and Johann Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. pages 1572–1577, 1998.

http://www.imavs.org/

Autonomous Navigation in Partially Observable Environments Using Hierarchical Q-Learning

Y. Zhou *, E. van Kampen, and Q. P. Chu
Delft University of Technology, 2629HS Delft, The Netherlands

http://www.imavs.org/

ABSTRACT

A self-learning adaptive flight control design allows reliable and effective operation of flight vehicles in a complex environment. Reinforcement Learning provides a model-free, adaptive, and effective process for optimal control and navigation. This paper presents a new and systematic approach combining Q-learning and hierarchical reinforcement learning with additional connecting Q-value functions, which separate the effect of internal rewards from the external rewards. An online navigation algorithm with both Q-learning and Hierarchical decomposition is provided and applied to an illustrative task in a complex, partially observable environment. The environment is completely unknown at the beginning, and the agent learns the most efficient path online to avoid obstacles and to get to the target area. The present work compares the results using 'flat' Q-learning and hierarchical Q-learning. The results indicate that hierarchical Q-learning can help to accelerate learning, to solve 'curse of dimensionality' in complex navigation tasks, to naturally reduce the uncertainty or ambiguity at higher levels, and to transfer the learning results within tasks and across tasks efficiently. This proposed method can potentially design a near-optimal controller hierarchically for autonomous navigation without a-prior knowledge of the environment.

1 INTRODUCTION

In recent years, many studies have implemented Reinforcement Learning (RL) controllers to solve nonlinear, optimal control problems. Reinforcement learning is learning what actions to take to affect the state and to maximize the numerical reward signal by interacting with the environment, and to maximize expected future rewards ultimately. This method links bio-inspired artificial intelligence techniques to the field of control to overcome some of the limitations and problems in many control methods. Nevertheless, traditional RL methods solving the optimality problem is an off-line method, assuming that the state and environment is fully observable and that the observed states obey Markov processes. These methods, especially those applied to flying vehicles, are rendered intractable by 1) the nonlinearity of the unknown system, 2) the complexity of the task and environment, and 3) the partial observability of the system and environment. This paper focus on the last two problems induced by the complex and partially observable environment, which are common challenges in navigation.

Reinforcement learning often encounter the exponentially growing of states caused by the complexity of the environment and decision making, which is called the 'curse of dimensionality'. Many RL algorithms apply approximate dynamic programming (ADP), which uses a universal approximator with parameters to approximate the cost/value function, so that they can be used to solve the optimality problems with large or continuous state spaces online [1, 2, 3, 4]. However, with current 'flat' methods, the number of the parameters will still grow with the exponentially growing of the states and actions. Recent research attempts to deal with the 'curse of dimensionality' with hierarchical decomposition, which has always been a natural approach to problem solving [2, 5]. Hierarchical methods replace state-to-action mapping by a hierarchy of temporally abstract actions which operate over several time steps. Therefore, a complex problem may be solved by decomposing it into some smaller and simpler problems. Hierarchical decomposition speeds up learning for complex tasks more efficiently. Additionally, the experience and learning results gained during the learning can be generalized within not only a task and but also across tasks [6].

In the real world, the agent might not have perfect perception of the states or the environment [7]. The framework dealing with Partially Observable Markov Decision Process (POMDP) problems and deciding how to act in partially observable environments has been developed especially for these situations and remains an active area of research [8, 9, 10]. For instance, Nominal Belief-state Optimization (NBO) [9, 10] which combines application-specific approximations and techniques within the POMDP framework produced a practical design that coordinates the plants in the presence of occlusions. Another method, called the Posterior Belief Distribution (PBD) [11] , has been proposed to calculate the posterior distribution over beliefs after a sequence of actions. It is an online, forward-search algorithm to control a flying vehicle in a target monitoring task by evaluating the expected reward of a sequence of primitive actions, called 'macro-actions'. Other than that, Output-Feedback (OPFB) approximate dynamic programming algorithms [4, 12] have

*Email address(es): Y.Zhou-6@tudelft.nl

http://www.imavs.org/

been proposed, as opposed to full state feedback, to tackle problems without direct state observation. POMDP methods act based on the estimation of the hidden variables and are theoretically more powerful. However, most existing work studies 'flat' models and cannot completely circumvent the partially observability of the environment. Hierarchical methods allow hidden variables to be estimated at different levels, and naturally reduce the uncertainty at higher levels.

This paper begins with a brief introduction to Markov and semi-Markov decision processes. Then, we present a hierarchical reinforcement learning (HRL) approach for autonomous navigation with Q-learning, which can help to systematically accelerate learning and to solve 'curse of dimensionality' in complex navigation tasks online. In section 4, the proposed method is applied to an illustrative navigation task with a discrete state space, and the results of using 'flat' Q-learning and hierarchical Q-learning are compared and discussed, showing how much the hierarchical method can improve the performance. The last part concludes the advantages and disadvantages of using the hierarchical method in this paper, and addresses the challenges and possibilities of the future research.

2 MARKOV DECISION PROCESSES AND SEMI-MARKOV DECISION PROCESSES

Reinforcement Learning is defined not by characterizing learning methods, but by characterizing a learning problem which can be described as an optimal control problem of Markov Decision Processes (MDPs) [13, 14]. MDPs are used to provide a mathematical framework to model decision making in situations where their outcomes are partly random and partly under the control of a decision maker [15]. MDP with finite state and action spaces is called a finite MDP and most of the modern theory of RL is restricted to finite MDPs [13]. A particular finite MDP is defined by its state and action sets, $\mathcal{S}$ and $\mathcal{A}$, and by the one-step dynamics of the environment. With any state s and action a, the probability of each possible next state s' can be obtained by transition probabilities $\mathcal{P}^a_{ss'}$:

$$\mathcal{P}^a_{ss'} = P\{s_{t+1} = s' | s_t = s, a_t = a\}. \tag{1}$$

With any current state s, action a and any next state s', the expected value of the next reward can be given as follow:

$$\mathcal{R}^a_{ss'} = E\{r_t | s_t = s, a_t = a, s_{t+1} = s'\}, \tag{2}$$

where r_t denotes the immediate reward of the next state s' after taking a possible action a. These qualities completely specify the most important aspects of the dynamics of a finite MDP.

RL approach uses the concepts of a dynamical system's state and of a value function under certain policy π to define a functional equation, now often called the Bellman equation [13]. By modifying the current policy π and the state value function V or state-action value function Q, RL methods make them tend to the optimal policy π^* and the optimal value function, V^* or Q^*. For a specific problem, there is always one or more optimal policies which are better than or equal to all other policies. Because it is the optimal value function, its consistency can be written in a special form without reference to any specific policy. Thus, the Bellman optimality equation for state value function, V^*, is shown as follows:

$$\begin{aligned} V^*(s) &= \max \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s \right\} \\ &= \max_{a \in \mathcal{A}} E_{\pi^*} \left\{ r_t + \gamma V^*(s') | s_t = s, a_t = a \right\}, \end{aligned} \tag{3}$$

where $\gamma \in [0, 1]$ is a parameter called the *discounted rate*. This equation also provide the relationship between the value of the current state $V^*(s)$ and its possible successor states $V^*(s')$.

Similarly, the state-action value function, denoted $Q^\pi(s, a)$, is defined as the expected return starting from s, taking the action a, and thereafter following policy π. And the corresponding Bellman optimality equation, Q^*, is shown as follows:

$$\begin{aligned} Q^*(s,a) &= E\{r_t + \gamma \max_{a' \in \mathcal{A}_{s'}} Q^*(s', a') | s_t = s, a_t = a\} \\ &= \mathcal{R}^a_{ss'} + \gamma \sum_{s'} \mathcal{P}^a_{ss'} \max_{a' \in \mathcal{A}_{s'}} Q^*(s', a'). \end{aligned} \tag{4}$$

An MDP defines only the sequential decision process rather than the time between one decision and the next [5]. The semi-Markov decision process (SMDP) is a generalization of MDP. An SMDP defines the time τ as the remaining/waiting time in the current stage. The amount of time can be either real valued in continuous-time discrete-event systems or integer valued in discrete-time systems, which is used in this paper. When action a is executed in the current state s_t, the joint transition probabilities of the next state s_{t+1} occurring after τ is written as $P(s_{t+1}, \tau | s_t, a)$ [5]. The Bellman optimality equation for V^* is

$$V^*(s) = \max_{a \in \mathcal{A}} \left\{ \mathcal{R}^a_{ss'} + \sum_{s', \tau} \gamma^\tau P(s', \tau | s, a) V^*(s') \right\}, \tag{5}$$

and the Bellman optimality equation for Q^* is

$$\begin{aligned} Q^*(s,a) &= \mathcal{R}^a_{ss'} \\ &\quad + \sum_{s', \tau} \gamma^\tau P(s', \tau | s, a) \max_{a' \in \mathcal{A}_{s'}} Q^*(s', a'). \end{aligned} \tag{6}$$

3 HIERARCHICAL Q-LEARNING

Navigation for mobile robots in known or small-scaled environment has been well studied. However, in practice, mobile robots often need to explore an initially unknown environment with limited sensors in complex autonomous navigation tasks. With state space decomposition into smaller, lower level state spaces, the task can be divided into smaller and easier tasks, where 'flat' algorithms can be used.

http://www.imavs.org/

3.1 Q-learning

Q-learning is an off-policy temporary difference (TD) method. The learning of the state-action value function Q, which directly approximates the optimal state-action value function Q^* of the *estimation policy*, is independent of the *behaviour policy* [13]. This separation allows exploration with the behaviour policy and simplifies the analysis of the algorithm. In this paper, we use one-step 'flat' Q-learning algorithm to update state-action values in discrete form with immediate rewards r_t, as follows:

$$Q_{k+1}(s,a) = (1-\delta)Q_k(s,a) + \delta\left[r_t + \gamma \max_{a' \in \mathcal{A}_{s'}} Q_k(s',a')\right], \tag{7}$$

where δ is a *learning-rate parameter*.

Q-learning can be applied to SMDP with action a executed in the current state s and immediate rewards r_{t+i} during the waiting time τ. In the discrete-time system, the Q-learning updates the estimated state-action value $Q_t(s,a)$ of the optimal $Q^*(s,a)$ as follows [5]:

$$Q_{k+1}(s,a) = (1-\delta)Q_k(s,a) + \delta\left[R_{t,\tau} + \gamma^\tau \max_{a' \in \mathcal{A}_{s'}} Q_k(s',a')\right], \tag{8}$$

where $R_{t,\tau}$ is the accumulated reward during the waiting time:

$$R_{t,\tau} = r_t + \gamma r_{t+1} + \cdots + \gamma^{\tau-1} r_{t+\tau-1}. \tag{9}$$

SMDP Q-learning has 3 significant advantages [13, 5, 16]. First, it uses iterative data samples from the distributions obtained from the real world test or produced stochastic simulations, rather than access to the explicit knowledge of the expected rewards or the state-transition probabilities. This makes SMDP Q-learning can be used as a model-free RL method. Second, SMDP Q-learning uses state-action values. Thus, in discrete-event systems, finding optimal actions does not require one-step ahead search or accessing to the one-step action models, which is often difficult. Third, it is possible to store action-values for every state-action pairs for small-scale problems. And it is easy to extend this advantage to large-scale problems by using function approximation methods and/or hierarchical reinforcement learning methods.

3.2 Decomposition and Hierarchies

For large-scale control and navigation problems, decomposition of tasks and abstraction of actions allow systems to solve current sub-problems and to ignore irrelevant details at current level. Each higher level uses a partial description of the environment, which can partition the environment into *sub-environment* or *macro states*. This feature may naturally reduce the uncertainty or ambiguity induced by partial observability of the environment. The activities or decisions in higher levels are called *macro actions* or *behaviours*, which instruct the policy in lower level. The macro action spaces and values are dependent on the current macro state decomposed in their levels. The space of the original one-step actions for each state, which are called *primitive actions*, may stay the same or partially admissible depending on the higher level macro action.

Macro action b is defined over its input set, macro states m, and the current high level policy μ similarly to primitive actions, and additionally, a termination condition $\beta : \{0,1\}$. Correspondingly, the Q-learning algorithm updates the estimation of the lower level state-action value with lower level action a executed in the current state s, and following a higher level macro action b until b terminates after τ steps:

$$Q_{k+1}(s,a,b) = (1-\delta)Q_k(s,a,b) + \delta\left[R_{t,\tau} + r_{t+\tau,b} + \gamma^\tau \max_{a' \in \mathcal{A}_{s'}} Q_k(s',a',b')\right], \tag{10}$$

where $R_{t,\tau}$ is the accumulated lower level reward during the waiting time (see Eq.9), and $r_{t+\tau,b}$ is the higher level reward returned after the termination of b, which is executed at current macro state m. All the states visited before b terminates belongs to the current macro state m.

With decomposition of the environment, the primitive actions and macro actions can be evaluated and updated separately. In real applications, the state may be partially observable. With hierarchical algorithms, the higher level may collect some information and estimate other state features. With those similar features, the lower level states can be clustered together and constitute a macro state. In the one-step level, given a higher level action b, the Q-learning algorithm may keep its 'flat' form and assign state-action values with partially observable features:

$$Q^{i,b}_{k+1}(s,a) = (1-\delta)Q^{i,b}_k(s,a) + \delta\left[r_{s',t} + r_{b,t} + \gamma \max_{a' \in \mathcal{A}_{s'}} Q^{i,b}_k(s',a')\right], \tag{11}$$

where i denotes that this is ith level Q-value following a $(i-1)$th level instruction b, the immediate reward $r_{s',t}$ is partially observable features related, and $r_{b,t}$ is the reward of accomplishment of following macro action b. With this Q-value $Q^b_{k+1}(s,a)$, the system can greedily choose a primitive action following the higher level instruction: taking 'macro action' b.

In higher levels, the system choose a 'macro action' b in current macro state m and evaluates and updates the macro state-action values. If the system has a partial observability, macro states m can be estimated states, or *belief macro states*. The Q-learning algorithm only considers the high level macro states m:

$$Q^j_{k+1}(m,b) = (1-\delta)Q^j_k(m,b) + \delta\left[r_{m',T} + \gamma \max_{b' \in \mathcal{B}_{m'}} Q^j_k(m',b')\right], \tag{12}$$

http://www.imavs.org/

where j denotes that this is jth level Q-value considering high level rewards, $r_{m',T}$ is the immediate reward when the system reaches the next macro state m', and T denotes the time, $t+\tau$ in Eq.10, when b terminates. It only updates after the termination of the macro action b.

3.3 Strategy Connecting Hierarchies

With good assigned rewards, Eq.11 and 12 can already be used to form the hierarchical Q-learning algorithm. However, when the system changes or the formulation of hierarchy changes, the learned policy may not be reused. To make this algorithm more flexible, transferable and reasonable, the lower level Q-learning algorithm, Eq.11, can be further divided into a 'flat' Q-learning algorithm and a connecting Q-learning algorithm. This division separates the external reward $r_{s',t}$ and the internal reward $r_{b,t}$ into different state-action value functions. Additionally, the discount factors for the external rewards and for internal rewards can be assigned independently as well.

In 'flat' RL algorithm, external rewards are usually used. External rewards are positive or negative outcomes from the external reward system or other environmental sources. They are tangible, such as consuming energy or reaching a charger in a robot navigation task. On the other hand, HRL algorithms often require assigned internal rewards, which are intangible and come from the sense of performance itself, such as accomplishment of following a higher level instruction or achievement of the sub-goal. The internal reward can either be a reward observed immediately after taking action a in the current state s, or be a delayed reward returned after termination of the macro action b. Assigning suitable internal rewards is difficult. It can affect the evaluation of the actions in each state following an instruction.

In this paper, a delayed internal reward evaluating the accomplishment of following the higher level macro action b is used. The internal rewards $r_{b,t}$ in each episode are assigned a same value $r_{b,T}$. Thus, the 'flat' lower level Q-learning algorithm is

$$Q^i_{k+1}(s,a) = (1-\delta)Q^i_k(s,a) + \delta\left[r_{s',t} + \gamma_{E,i}\max_{a'\in\mathcal{A}_{s'}} Q^i_k(s',a')\right], \quad (13)$$

where i indicates that this is ith level 'flat' Q-value only considering the ith level external reward, and $\gamma_{E,i}$ is the discounted factor for the ith level external rewards. The connecting Q-learning algorithm is

$$Q^i_{k+1}(s,a,b) = (1-\delta)Q^i_k(s,a,b) + \delta\left[\gamma^{\tau}_{I,i}\cdot r_{b,T} + \gamma_{I,i}\max_{a'\in\mathcal{A}_{s'}} Q^i_k(s',a',b)\right], \quad (14)$$

where i indicates that this is the additional ith level Q-value taking internal reward into account when following $(i-1)$th level macro action b, $\gamma_{I,i}$ is the discounted factor for the ith level internal rewards, and τ is the waiting time with $\tau = T - t$. Noted that this lower level Q-learning algorithm in Eq.13 is in the same form as the higher level Q-learning algorithm in Eq.12. This makes the hierarchical Q-learning algorithm systematic and reusable.

To evaluate and choose an action in current state s following a higher macro action b, we can combine the 'flat' lower level Q-value $Q^i_{k+1}(s,a)$ and the connecting Q-value $Q^i_{k+1}(s,a,b)$, as follows:

$$Q^{i,b}_{k+1}(s,a) = Q^i_{k+1}(s,a) + \omega_i \cdot Q^i_{k+1}(s,a,b), \quad (15)$$

where ω_i adjusts the weight of the internal rewards indicating the ith level accomplishment. Note that the state-action value in Eq.11 and 15 can be different even with $\gamma_E = \gamma_I$ and $\omega_i = 1$, because the maximum state-action values in Eq.13 and 14 for the next state s' are calculated separately. Comparing to Eq.11, this new estimated Q-value $Q^{i,b}_{k+1}(s,a)$ in Eq.15 with an on-line adjustable weight ω_i makes the evaluation process more flexible.

An n level hierarchical Q-learning algorithm consists of a primitive level Q-learning algorithm (Eq.13), $(n-1)$ higher level Q-learning algorithms (Eq.12), and $(n-1)$ connecting Q-learning algorithms (Eq.14). This hierarchical algorithm updates each level state-action value function independently, and evaluates the action in a state following a higher level instruction with a connecting strategy. Additionally, this algorithm separates external and internal rewards in different state-action value functions, which makes adjusting the importance of the internal rewards online possible and more flexible. These features make this hierarchical Q-learning algorithm more systematic, expandable, and easy to apply to real applications.

4 APPLICATION

4.1 Problem Setup

This section will present an illustrative application of the hierarchical Q-learning algorithm: an indoor discretized navigation problem for a flying robot in a maze. Similar to Parr's maze problem [17], this maze has about 3600 *absolute primitive states* and a target area, as shown in Fig.1. However, this task is more complex and realistic with partial observability, different starting locations, and limited mobility. It actually has to perform two tasks at the same time: avoiding the obstacles, and reaching the target area as soon as possible.

As depicted in Fig.2, in this indoor task, the flying robot only has 3-step short-sight (obstacle in the 1, 2, 3-step away or nothing) in 3 directions (front, left, and right) and knows its heading (north, south, east, and west). Thus, this robot may only have 256 *observed primitive states*, which induces ambiguity in this complex task. This robot has limited mobility: turn left, turn right, and move forward. Taking any of the primitive action will consume the energy and result in a penalty '-1'. If the robot moves towards an obstacle when it

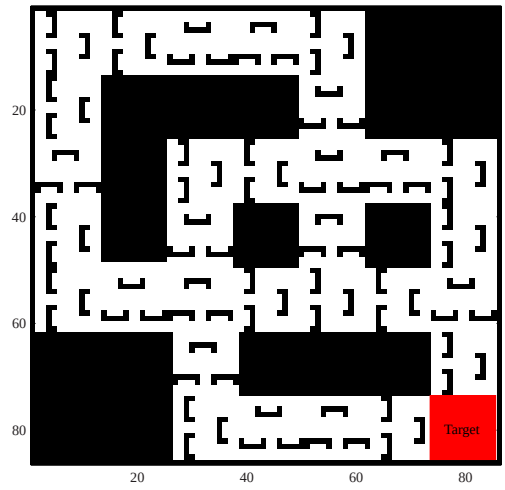


Figure 1: A POMDP maze problem with obstacles and a target.

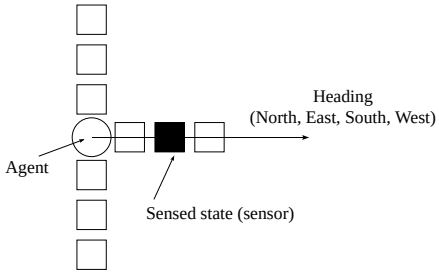


Figure 2: A primitive state of the flying robot.

is next to the obstacle, it will automatically turn around and result in a penalty '-3'. When it reaches the target area, the robot will land at this area and get a terminal reward '10'. This flying robot does not know the map of this maze in advance, but has an internal memory of its trajectory, which will be removed before another episode. This task focuses on the high-level autonomous navigation and online path planning rather than the system model or low-level control. We assume that the flying robot has the ability to perform the primitive actions.

4.2 Hierarchical Q-learning applied to Navigation

By using 'flat' Q-learning, the flying robot may only learn to avoid the obstacles, because it will not know its correct absolute position. To get to the target as soon as possible, it need to explore in the maze, to find its possible position, and to determine a proper direction from a higher level. This is how the hierarchical Q-learning method can be applied to this task.

This hierarchical method has 2 levels. The 1st level agent follows the ultimate goal of reaching the target area. It memorizes and draws the observed pattern of the local map, and estimates its possible position, which is called *belief macro state* $\widehat{m}$, by comparing the pattern with a memorized map. Each macro state is a decomposed environment and contains a subset of the primitive states. After each episode, the macro state-action value function can be updated with following algorithm:

$$Q^1_{k+1}(m,b) = (1-\delta)Q^1_k(m,b) + \delta\left[r_{m',T} + \gamma_{E,1}\max_{b'\in\mathcal{B}_{m'}} Q^1_k(m',b')\right], \quad (16)$$

where T denotes it is the Tth (m,b) macro state-action pair, and $r_{m',T}:\{0,10\}$ is the immediate reward when the system reaches the next macro state m'. With the belief state $\widehat{m}$ and the state-action value function $Q^1_{k+1}(\widehat{m},b)$, the agent in the 1st level will give an instruction, macro action $b:\{$ go north, south, east, west $\}$, for the 2nd level agent.

The 2nd level agent follows the higher level instruction b by using the connecting Q-value updated by

$$Q^2_{k+1}(s,a,b) = (1-\delta)Q^2_k(s,a,b) + \delta\left[\gamma^{\tau}_{I,2}\cdot r_{b,T} + \gamma_{I,2}\max_{a'\in\mathcal{A}_{s'}} Q^2_k(s',a',b)\right], \quad (17)$$

with a delayed internal reward $r_{b,T}:\{0.5,-0.5\}$, which assigns a reward for going the right direction and a penalty for going a wrong direction or not moving. In this equation, τ is the waiting time from current time t till the termination of the macro action b at time T: $\tau = T-t$. At the same time, it accomplishes the task in its own level, avoiding obstacles, using the primitive state-action value function updated by

$$Q^2_{k+1}(s,a) = (1-\delta)Q^2_k(s,a) + \delta\left[r_{s',t} + \gamma_{E,2}\max_{a'\in\mathcal{A}_{s'}} Q^2_k(s',a')\right], \quad (18)$$

where $r_{s',t}:\{-1,-3\}$ is the immediate reward when the system reaches the next state s'.

4.3 Result and Discussion

Two algorithms are applied to this navigation problem: a 'flat' Q-learning algorithm and the hierarchical Q-learning algorithm presented in this paper. Fig.3 compares the numbers of primitive actions taken in each iteration for the 'flat' Q-learning and hierarchical Q-learning algorithm. With 'flat' Q-learning, the agent learns how to prevent from collision in the first several iterations. However, the performance will not be improved significantly hereafter. There are two main reasons: first, the huge number of actions taken before reaching the target area makes the effect of the terminal reward propagate very slow to the experienced states; and second, the partial observability leads to the ambiguity in real absolute states, which prevents the convergence of observed state-action values. These features impair the agent's learning ability of performing the second task: reaching the target area as soon as possible.

http://www.imavs.org/

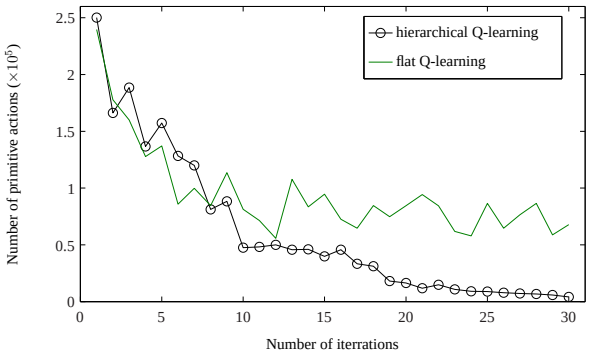


Figure 3: The number of primitive actions taken in each iteration with a 'flat' Q-learning and a hierarchical Q-learning algorithm (averaged of 5 runs).

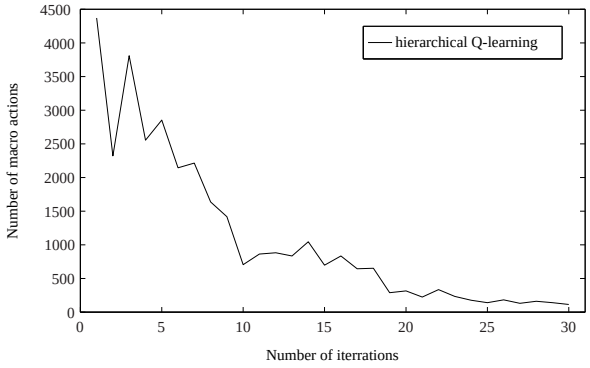


Figure 4: The number of macro actions taken in each iteration with the hierarchical Q-learning algorithm (averaged of 5 runs).

On the other hand, the hierarchical Q-learning algorithm improves the performance of both the collision avoidance task and searching for the target area task. The low level agent focus on avoiding obstacles and following high level's instruction. The high level agent focus its exploration of the state space and finds the optimal macro actions to get to the target area as soon as possible. In Fig.4, there is a clear trend of decreasing number of the macro actions taken during learning. This result indicates that the agent improves the performance also from a higher level. The hierarchical Q-learning algorithm appears to improve the performance considerably and much faster. Even with more iterations and longer learning time, the 'flat' Q-learning method can not improve the navigation performance as much as the hierarchical Q-learning method.

This application demonstrates the ability of the hierarchical Q-learning to perform a navigation problem with partial

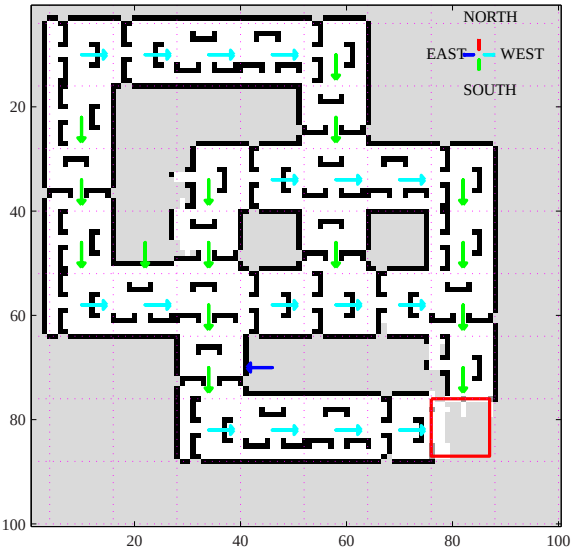


Figure 5: The greedy macro actions in the memorized map after 15 iterations with hierarchical Q-learning.

observability. Fig.5 presents the greedy macro actions learned by the high level hierarchical Q-learning in the memorized map. The grid with pink dotted line decomposes the environment into macro states. Since each macro state consists of a fixed size of area, the corridor area is not completely separated from the inaccessible area with walls with the environment decomposition. Some macro states can be partly corridor and partly inaccessible area with walls, which also have been experienced and have greedy macro actions. The agent created this map by stitching the constructed maps from the memory after each episode. The light gray states indicate that those areas haven't been experienced or observed. The dark gray states are experienced target states and are clustered with a red triangle.

These results show that the state-action values is converging with experiences. With more learning episodes, the agent will explore all possible actions and exploit those experience to learn to follow higher level instructions and to accomplish the task as good as possible. Thus, the memorized map will be more complete, and the state-action value functions will all converge. The agent has the ability to find the hierarchical optimal path online from any initial position.

Furthermore, in a new high level task, such as searching for a different target area, or a new environment, such as an expanded maze, the agent can still use the low level and connecting Q-learning results, and only need to learn a new higher level policy. The experience and learning results during the learning can be transferred across tasks. This feature also speeds up learning for complex tasks more efficiently, and makes this present hierarchical Q-learning a transfer learning method.

http://www.imavs.org/

http://www.imavs.org/

5 CONCLUSION

This paper proposes a systematic hierarchical Q-learning method and applies it to a complex navigation task in a partially observable environment. This method combines the advantages of both the 'flat' Q-learning method and the hierarchical structure. The results indicate that the present hierarchical Q-learning method can help to accelerate learning, to solve 'curse of dimensionality' in a complex autonomous navigation task, to naturally reduce the uncertainty or ambiguity at higher levels, and to transfer the experience and learning results within and across tasks efficiently.

This paper investigates the complexity and partial observability of the task and environment in navigation tasks, but not yet the nonlinear, unknown system of flying vehicles. In the future, we will apply this method to a more realistic micro aerial vehicle control problem by further decomposing the complex flight task into high-level navigation tasks and low-level control tasks, such as reference tracking. In low-level control, the problem of nonlinearity of the unknown system can be solved by using other 'flat' methods such as incremental approximate dynamic programming. Thus, three of the common problems in flight control mentioned at the beginning of this paper, which are 1) the nonlinearity of the unknown system, 2) the complexity of the task and environment, and 3) the partial observability of the system and environment, can be solved by using the hierarchical reinforcement learning combining different 'flat' algorithms in each level.

REFERENCES

[1] Said G Khan, Guido Herrmann, Frank L Lewis, Tony Pipe, and Chris Melhuish. Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annual Reviews in Control*, 36(1):42–59, 2012.

[2] Jennie Si. *Handbook of learning and approximate dynamic programming*, volume 2. John Wiley & Sons, 2004.

[3] Y. Zhou, E. van Kampen, and Qi Ping Chu. Incremental approximate dynamic programming for nonlinear flight control design. In *Proceedings of the EuroGNC 2015*, 2015.

[4] Y. Zhou, E. van Kampen, and Qi Ping Chu. Nonlinear adaptive flight control using incremental approximate dynamic programming and output feedback. In *AIAA Guidance, Navigation and Control Conference*, 2016.

[5] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(1-2):41–77, 2003.

[6] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, 10:1633–1685, 2009.

[7] Olivier Sigaud and Olivier Buffet. *Markov decision processes in artificial intelligence*. John Wiley & Sons, 2013.

[8] Alex Brooks, Alexei Makarenko, Stefan Williams, and Hugh Durrant-Whyte. Parametric POMDPs for planning in continuous state spaces. *Robotics and Autonomous Systems*, 54(11):887–897, 2006.

[9] Zachary A. Harris Scott A. Miller and Edwin K. P. Chong. A POMDP framework for coordinated guidance of autonomous uavs for multitarget tracking. *EURASIP Journal on Advances in Signal Processing*, 2009.

[10] Shankarachary Ragi and Edwin K. P. Chong. UAV path planning in a dynamic environment via partially observable markov decision process. *IEEE Transactions on Aerospace and Electronic Systems*, 49(4):2397–2412, 2013.

[11] Ruijie He, Emma Brunskill, and Nicholas Roy. Efficient planning under uncertainty with macro-actions. *Journal of Artificial Intelligence Research*, 40(1):523–570, 2011.

[12] Frank L Lewis and Kyriakos G Vamvoudakis. Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 41(1):14–25, 2011.

[13] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[14] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[15] Mohamed Rida, Hicham Mouncif, and Azedine Boulmakoul. Application of markov decision processes for modeling and optimization of decision-making within a container port. In *Soft Computing in Industrial Applications*, pages 349–358. Springer, 2011.

[16] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition, 1996.

[17] Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, pages 1043–1049, 1998.

Autonomous Positioning and Navigation for UAV in GPS-Denied Environments

Xiaolong Zhang*, Zhihong Peng, and Hongyang Liu
School of Automation, Beijing Institute of Technology, Beijing, 100081, China
State Key Laboratory of Intelligent Control and Decision of Complex System, Beijing, 100081, China
School of Automation, Beijing Institute of Technology, Beijing, 100081, China

http://www.imavs.org/

ABSTRACT

A platform integrating visual camera into Unmanned Aerial Vehicle (UAV) system to realize the function of autonomous positioning and navigation in GPS-denied environments is presented. Through motion tracking and depth perception, the visual camera can provide estimated information on the surrounding environments for UAV to navigate. A control method of pose estimation and navigation strategy is proposed to make this system more stable and efficient, and to enhance its robustness. Several experimental tests have been carried out, illustrating the validity of the proposed approach.

1 INTRODUCTION

Recent years have witnessed a wide range of autonomous applications of UAVs. UAVs have obtained lots of attention as low-cost, flexible, high-efficiency platforms covering vast regions and providing widespread field of view. UAVs always take field operations, including environment exploration, intelligence gathering, searching and rescuing victims and etc. All the jobs UAVs can do are mostly based on autonomous positioning and navigation technologies which rely on GPS and have reached its maturity. However, there are plenty of possibilities that UAVs are not able to access GPS in complex environments. In such cases, GPS signal is not available because of the poor weather condition or the block of obstacles.

Several studies focus on how to locate an UAV and guide it to desired waypoints in GPS-denied environments. Navigation strategy with RGB-D camera is applied in [1, 2]. [3] proposes a navigation research in GPS-denied environment based on monocular-SLAM technology. [4] proposes a visual navigation system for an UAV using optical flow in a GPS-denied environment. A framework for Google Map aided UAV navigation in GPS-denied environment is used by [5]. It applies geo-referenced navigation to provide drift-free localization. [6] presents a vision-aided inertial navigation system for small UAV and develops a delay-based approach for visual measurement in GPS-denied environments.

Consumer-grade technology seen in cameras and tablet has led to price-performance ratio falling dramatically over the last decade. Google, for example, is developing a customized tablet named Project Tango equipped with conventional sensors, like inertial measurement units (IMU), as well as depth cameras and it has significant computational capability. This paper constructs a programmable platform with this tablet as a visual sensor and chooses Pelican quad-rotors produced by Ascending Technologies as the experimental carrying body. Additionally, a personal computer (PC) plays the role of a ground station to make the system more efficient and flexible. Afterwards, this paper puts forward a method of its pose estimation and autonomous positioning and navigation in GPS-denied situation based on the visual camera.

The rest of this paper is organized as follows. Section 2 introduces a dynamic model for quad-rotors UAV. Section 3 shows how to construct the programmable platform in terms of hardware and software respectively. Section 4 proposes a control method of pose estimation and autonomous positioning and navigation strategy of UAV. Section 5 concludes the work of this paper.

2 MODELING

A quad-rotors is a system consists of four identical rotors and propellers located at the vertices of a square (see Figure 1).

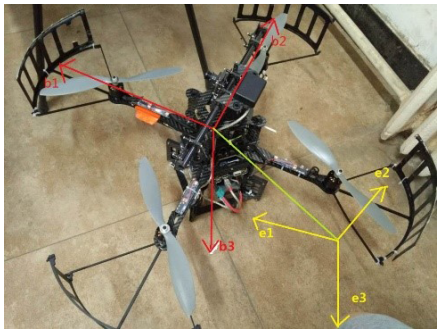


Figure 1: A quad-rotors.

Considering an inertial reference frame donated by $\{e_1, e_2, e_3\}$ and a body reference frame centered in the center of mass (COM) of the quad-rotors donated by $\{b_1, b_2, b_3\}$, a dynamic model can be expresses as

*Email address(es): zhangxiaolong23@sina.cn

$$\dot{x} = v$$
$$m\dot{v} = -R\tau e_3 + mge_3$$
$$\dot{R} = R\widehat{\Omega}$$
$$J\dot{\Omega} + \Omega \times J\Omega = M \quad (1)$$

where $x \in R^3$ is the Cartesian position of the quad-rotors related to inertial frame, $v \in R^3$ is the velocity of the quad-rotors in the inertial frame, $m \in R$ is the mass, $\Omega \in R^3$ is the angular velocity of the quad-rotors in the body frame, $J \in R^3$ is the inertia matrix with respect to the body frame, and $R \in SO(3)$ is the rotation matrix with respect to the body frame. $M \in R^3$ being the total moment along all axes of the body frame and the thrust $F \in R$ are control inputs of the plant. g is the standard gravitational acceleration with $e_3 = [0\ 0\ 1]^T$. The hat symbol $\widehat{*}$ means the skew-symmetry operator as $x\widehat{y} = x \times y, \forall x, y \in R^3$. Assuming that the force of each propeller is directly controlled [7] , the relationship among F, M and single-motor force $f_i, i = 1, 2, 3, 4$ can be expressed as

$$\tau = f_1 + f_2 + f_3 + f_4$$
$$M_1 = df_2 - df_4$$
$$M_2 = df_1 - df_3$$
$$M_3 = -cf_1 + cf_2 - cf_3 + cf_4 \quad (2)$$

where d is the distance from the center of each rotor to COM and c is a constant value.

3 SYSTEM STRUCTURE

3.1 *Hardware structure*

As previously mentioned, this paper chooses Project Tango tablet as a visual sensor, Pelican quad-rotors as a carrying body and PC as a ground station to construct a platform. Its structure is seen in Figure 2. These three parts are connected to each other through a wireless connection, which can make the system flexible, simple and easily debugged.

The Project Tango has such three core technologies as motion tracking, depth perception and area learning which we can take advantage of to allow the platform to know where it is and what its surroundings are, then to predict a way to move on.

The Pelican quad-rotors is equipped with an Autopilot board which consists of IMUs and two ARM7 processors named Low Level Processor (LLP) and High Level Processor (HLP) respectively. The LLP handles data fusion and other underlying operations. The HLP is programmable for our algorithm and it has serial communication interfaces to connect the ground station.

The Project Tango is attached to the Pelican physically.

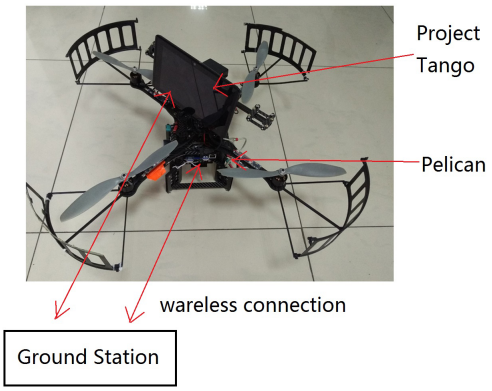


Figure 2: Hardware structure.

3.2 *Software structure*

The autonomous flight controlled system of the Pelican is designed to be a double loop controlled system, as shown in Figure 3. A proportional integral derivative (PID) controller in the inner control loop is fixed in the hardware of Pelican and receives information from the remote control (RC), while an integral separation PID controller is adopted in the external loop, where Project Tango acts as a visual sensor.

Being attached to the Pelican, the tablet can monitor its real-time status and compare it to the desired status. The external controller receives these error items as its own input while its output can be the given amounts of the internal controller instead of the control information from the RC. Finally, the autonomous control and navigation of the UAV platform can be enhanced by integrating the estimated information of surrounding environments.

As previously mentioned, three parts of this system are connected to each other over a wireless connection (see Figure 4). A android application routine is enabled on the tablet for pose and depth data streaming to the ground station using the user datagram protocol (UDP) which transfers data more efficiently and is more reliable.

As shown in Figure 5, the UDP packets are received by the ground station and are subsequently processed into usable commands which are eventually transferred to Pelican through universal asynchronous receiver/transmitter (UART). The Pelican receives commands from ground station to replace the control information from the RC to control the flight.

4 CONTROL METHOD

4.1 *Pose estimation*

The ability to estimate pose accurately and efficiently is very important to navigation in GPS-denied environments.

http://www.imavs.org/

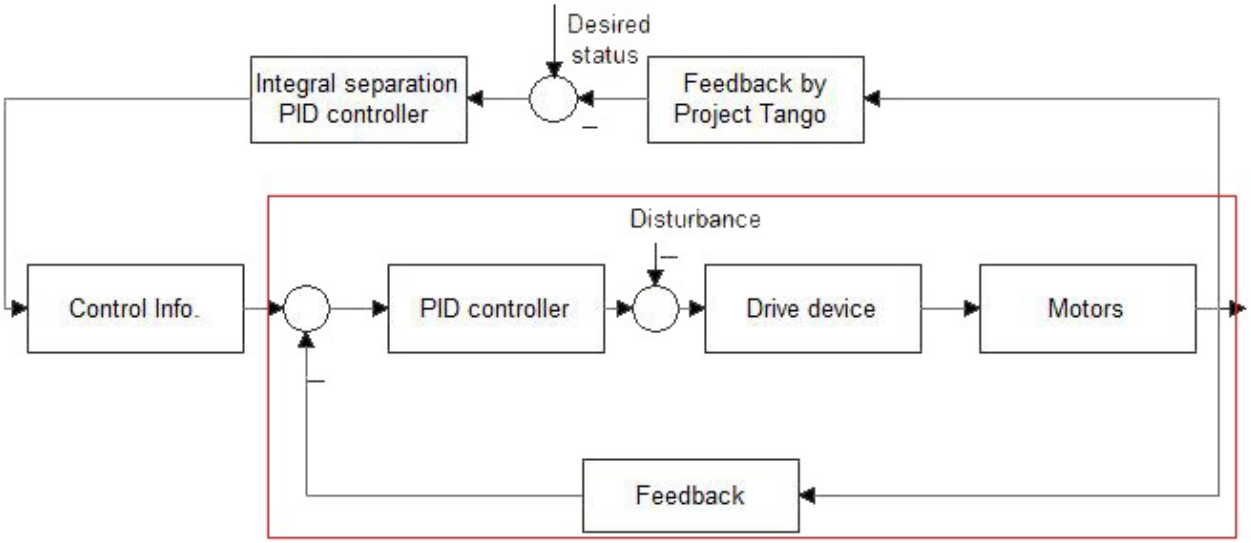


Figure 3: Autonomous flight control of the Pelican.

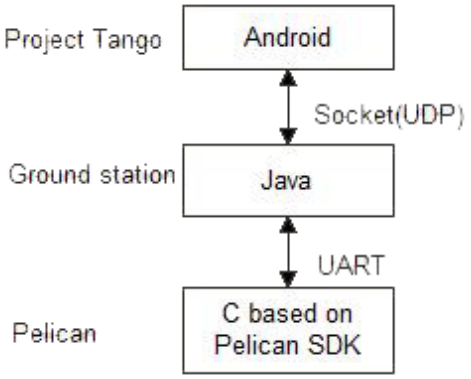


Figure 4: Software structure.

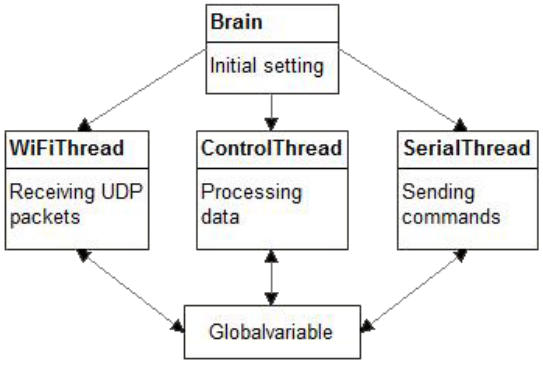


Figure 5: Ground station.

Pose data is accessible from Project Tango and so is Pelicans pose by processing these data since they are attached to each other and in the same coordinate system.

It is available to read position data with respect to the start point from Project Tango. By mean filter of continuous returned N data $p_i, i = 1, 2, , N$ to eliminate the influence of random disturbance [7] , the current position p_{now} of the Pelican with respect to the start point can be expressed as

$$p_{now} = \frac{1}{N} \sum_{i=0}^{N} p_i \tag{3}$$

Similarly, quaternion data can also be obtained from Project Tango which characterizes the attitude of Pelican. Doing the real-time math to solve the quaternion [7] , the attitude information will be obtained from

$$\begin{aligned} \phi &= \arctan \frac{2(q_0 q_1 + q_2 q_3)}{1 - 2(q_1^2 + q_2^2)} \\ \theta &= \arcsin(2(q_0 q_2 - q_1 q_3)) \\ \psi &= \arctan \frac{2(q_0 q_3 + q_1 q_2)}{1 - 2(q_2^2 + q_3^2)} \end{aligned} \tag{4}$$

where q_0 is the real part of the quaternion and q_1, q_2, q_3 are the imaginary parts, ϕ is the yaw angle, θ is the roll angle, and ψ is the pitch angle, respectively. These angles are relative angles between the inertial reference frame and the body reference frame.

http://www.imavs.org/

http://www.imavs.org/

4.2 *Navigation strategy*

This paper uses the depth perception technology of Project Tango to put forward a navigation strategy. A point cloud can be obtained by developing the Project Tango, like Figure 6.

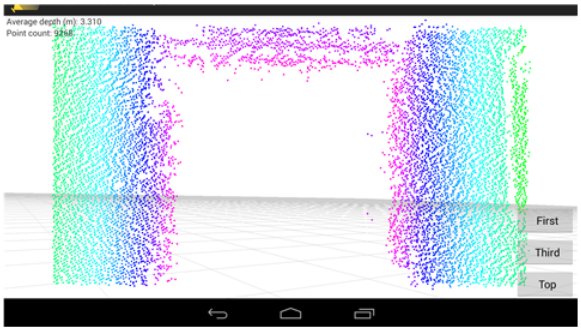


Figure 6: Point cloud.

The cloud point shown in Figure 6 means a three-dimensional (3-D) coordinate estimation which presents the relative positional relationship between the mapped point in real world and the Project Tango. According to this, the platform can realize the depth information of surroundings. Fixed by a recognition algorithm, it can learn which direction has enough space to go forward, additionally, detect and avoid obstacles.

4.3 *Control algorithm*

The platform can obtain its current state through pose estimation, and the future state it should achieve through navigation strategy. The next task is to consider how to control the conversion of state.

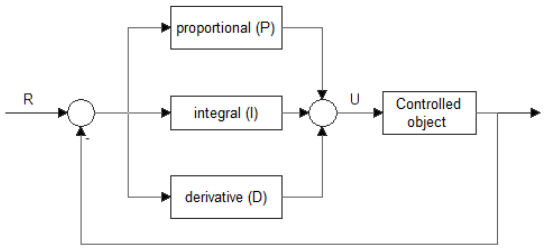


Figure 7: PID algorithm.

$$U(k) = k_p e(k) + k_i \sum_{j=0}^{k} e(j) + k_d(e(k) - e(k-1)) \quad (5)$$

What can be found is that the approach which is put forward above can always convert into some kinds of error control, which can be achieved by using the PID control algorithm (see Figure 7 and (5)) where $e(k)$ is the error, $U(k)$ is the output control amount, k_p is for present values of the error, k_i is for past values of the error, k_d is for possible future values of the error, respectively. However, there is one common problem resulting from the ideal PID implementations, which is the integral windup. Following a large change in set point, the integral term can accumulate an error much larger than the maximal value for the regulation variable (windup), thus the system overshoots and continues to increase until this accumulated error is unwound.

So, an integral separation PID control is adopted (see Figure 3), which is the improvement of the ideal PID and can be expressed as

$$U(k) = k_p e(k) + \beta k_i \sum_{j=0}^{k} e(j) + k_d(e(k) - e(k-1))$$
$$\beta = 1, e(k) \leq \varepsilon$$
$$\beta = 0, e(k) > \varepsilon \quad (6)$$

where ε is the separation limit, integral action will be performed only when $e(k) \leq \varepsilon$ to eliminate the integral windup.

4.4 *Experiment*

The experiment is based on the following assumptions:

(1)Choosing narrow space like corridor as experiment environment.

(2)Setting the initial position of Pelican artificially.

(3)The planning track for Pelican is flying in the horizontal midline while maintaining a certain height and returning if it reaches the end of the corridor.

Based on the above assumptions, this paper puts forward a specific recognition algorithm. What should to do first is to abstract Figure 6 to Figure 8 as shown below.

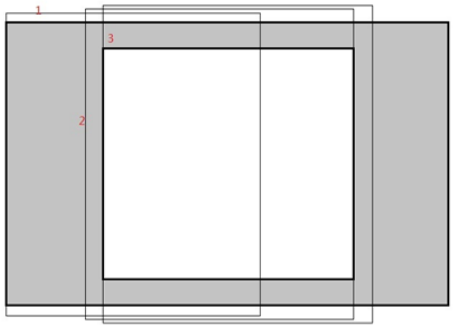


Figure 8: Abstract image.

The followings name the white rectangle unknown rectangle (UR) and the rectangle, which is filled with shadow and UR, known rectangle (KR). There exists UR because there is a limit of detection distance of the Project Tango so that it can not detect all the points in front of itself. Conversely, those points covered by the shadow are those that can be detected.

http://www.imavs.org/

Assuming that the head of Pelican is always parallel to the axis of the corridor, the problem that should be solved is the relative positional relationship between UR and KR. Doing this by following these steps.

(1)Recording 95% of the counts of those points that can not be detected in KR as Nc.

(2)Constructing a rectangle which is big enough to cover the UR, moving it from left to right with fixed step and recording the counts of those points that can not be detected in it as Ni.

(3)Recording the left boundary of the moving rectangle when Ni is bigger than Nc for the first time as L and the right boundary of the moving rectangle when Ni is bigger than Nc for the last time as R. $m = \frac{L+R}{2}$ will be the midline of UR. With the already known midline M of KR, we can obtain the relative positional relationship between UR and KR.

Additionally, to make sure that the head of Pelican is always parallel to the axis of the corridor, what should be done is to operate real-time monitor of the change of Pelicans yaw angle and make the change always smaller than a threshold.

In order to detect the end of the corridor, we can record the real-time average depth in the field of view. Actually, the average depth will hop when Pelican reaches the end of the corridor (see Figure 9).

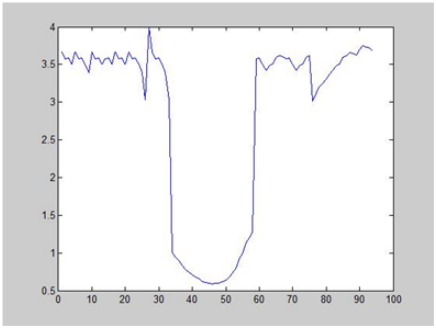


Figure 9: Average depth in the field of view.

As previously mentioned, the strategy which is put forward above can convert into error control, which can be achieved by using the integral separation PID algorithm.

In this experiment, the height of Pelican can stay in a relatively small range near a fixed value (see Figure 10). Additionally, the platform can find and reach the midline of the corridor, and the track of its horizontal position can coincide with the axis of the corridor in a small margin of error (see Figure 11).

5 CONCLUSION

This paper constructs a programmable UAV platform and demonstrates the hardware and software structure with the proposed control method to enable the autonomous positioning and navigation for the platform in simple GPS-denied environments. Serval experiments are carried out and some certain results are obtained.

Future work will take more complicated GPS-denied environments into account.

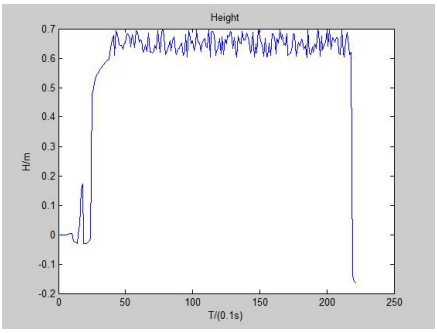


Figure 10: Height curve during the experiment.

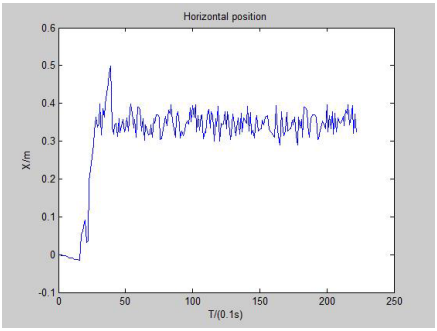


Figure 11: Horizontal position curve during the experiment.

REFERENCES

[1] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *International Symposium on Robotics Research (ISRR)*, volume 2, 2011.

[2] Giuseppe Loianno, Vincenzo Lippiello, and Bruno Siciliano. Fast localization and 3d mapping using an rgb-d sensor. In *Advanced Robotics (ICAR), 2013 16th International Conference on*, pages 1–6. IEEE, 2013.

[3] Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.

[4] Lan Huang, Jian Mei Song, Pu Hua Chen, and Gao Hua Cai. Visual navigation for uav using optical flow estimation. In *Control Conference (CCC), 2014 33rd Chinese*, pages 816–821. IEEE, 2014.

[5] Mo Shan, Fei Wang, Feng Lin, Zhi Gao, Ya Z Tang, and Ben M Chen. Google map aided visual navigation for uavs in gps-denied environment. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 114–119. IEEE, 2015.

[6] Tianmiao Wang, Chaolei Wang, Jianhong Liang, Yang Chen, and Yicheng Zhang. Vision-aided inertial navigation for small unmanned aerial vehicles in gps-denied environments. *International Journal of Advanced Robotic Systems*, 10, 2013.

[7] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The grasp multiple micro-uav testbed. *IEEE Robotics & Automation Magazine*, 17(3):56–65, 2010.

http://www.imavs.org/

Commercially Suitable Hybrid Multi-Visual, Inertial and Satellite Navigation Unit for Small Unmanned Aerial Vehicles

N.A. Klyuchnikov*, A.A. Uryasheva, N.S. Rodichenko
Skolkovo Institute of Science and Technology, Moscow

http://www.imavs.org/

ABSTRACT

Despite extensive research background and availability of components, navigation systems for small Unmanned Aerial Vehicles (UAVs) that incorporate all the three methods (vision-, inertia-, and satellite-based) in a single module are not available in the market. To solve this problem we develop a commercially suitable hybrid navigation module, incorporating two state-of-the-art visual navigation algorithms, namely: Fast Semi-Direct Monocular Visual Odometry and Large-Scale Direct Monocular SLAM, while a framework called Multi-Sensor-Fusion Extended Kalman Filter is used for fusing readings from the visual sensor with the other ones. The software is implemented on top of the Robot Operating System. After prototyping the system, we test it for robustness to different motion and environment types to ensure that it is viable for commercial applications.

1 INTRODUCTION

Over the past two decades there has been a lot of advances in research on visual and hybrid navigation for robots [1], however, most of the works were merely aimed at proving the concept, so there has been much engineering work left to be conducted. Nowadays, the standard navigation kit of small UAVs remains consisting only of Inertial Navigation System (INS), some secondary sensors (e.g. an atmosphere pressure sensor) and a Global Navigation Satellite System (GNSS) receiver [2]. Despite some attempts to endow small UAVs with visual sensing [3, 4, 5], there is still no ubiquitous vision-based GNSS backup system.

2 MATERIALS AND METHODS

Two state-of-the-art monocular visual navigation algorithms, which can be run online on modern microcomputers, were used:

1. Fast Semi-Direct Monocular Visual Odometry (SVO) [6] extracts feature-points from the image-frame only when the observed scene significantly differs from the reference one. For all other interim cases the algorithm applies a sparse direct method for matching patches around the highlighted features. To work correctly SVO requires a high frequency image input stream (approximately 50Hz and up).

2. Large-Scale Direct Monocular SLAM (LSD) [7] is a novel direct monocular visual navigation algorithm, which incorporates frame depth estimation errors into photometric error of their alignment. This approach enables correction of scale drifting by geometry optimization of the scale along with rigid body transformation.

Multi-Sensor-Fusion Extended Kalman Filter (MSF) [8] is an algorithm for fusing navigation data with automatic calibration of absolute and relative measurements.

Robot Operating System (ROS) [9] is an open source platform with a set of tools, libraries and drivers designed to ease the process of creating and sharing robotic software. ROS project constitutes a set of small program modules called nodes, which communicate with each other through publishing messages to specific topics and subscribing to them respectively.

Pixhawk [10] is a low-cost open hardware and software flight control module, which includes an autopilot, processing unit, sensors and input/output system.

PX4 Pro [11] is a running on Pixhawk flight control stack, which is a state-of-the-art software platform. PX4 Pro enables seamless integration of visual navigation data into position estimation by means of an inertial navigation filter INAV [12] or a Kalman filter.

Intel NUC i5 [13] is a mini, but complete computer with Intel Core i5 processor. Used in the work as an additional computational module.

PlayStation 3 Eye [14] is a low-cost digital usb camera which allows high-frequency frame capturing (up to 120 Hz).

To test the system robustness we used environments with different texture parameters. Two examples of test scenes are provided in figure 1. We also used different system motion types, which included rotations, horizontal and vertical translations.

*Email address: fmsnew@gmail.com



Figure 1: Examples of test environments.

3 RESULTS

We tested how the combination of the two visual navigation algorithms improves system robustness in terms of duration of critical moments (i.e. moments within which both algorithms fail to track the motion). Figure 2 represents the statistics of gain in robustness over 34 tests. The value of robustness_gain for each test was calculated as a ratio of the total duration of critical moments to the minimum total tracking fail durations of both algorithms. The mean value of the robustness_gain is approximately 0.46.

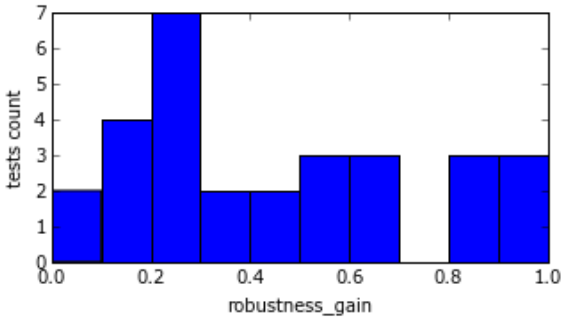


Figure 2: Tests statistics on gain in robustness when combination of visual navigation algorithms is used.

Duration of each single fail period is also an important indicator of robustness (see Figure 3), since the system has to rely only on an extremely unstable INS within critical moments. The mean LSD fail duration is 0.21s, whereas mean hybrid is 0.18. SVO has huge outliers, so it is better described by median, which is 0.39s.

4 SYSTEM DEVELOPMENT

4.1 System Architecture

The algorithms reviewed above are quite resource demanding, thus in order to run them online, an additional computationally powerful hardware is needed. On the other hand, to insure against failures of the computation module it is required to provide the autopilot with inertial and satellite data independently. Such considerations led us to a system design, where the autopilot receives these data first and then re-

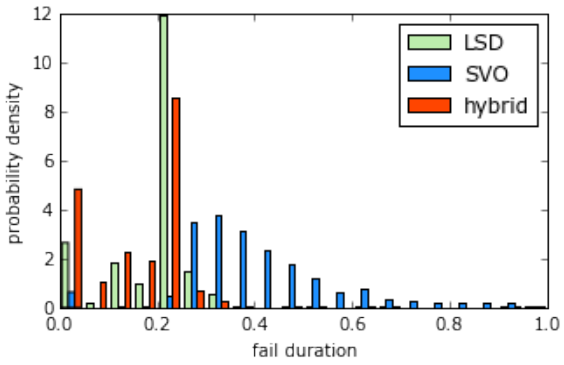


Figure 3: The duration statistics in seconds of each single fail period. More than 98% cases for LSD and hybrid, and 86% cases for SVO are under 1s. The rest cases for LSD and hybrid are under 2s, whereas SVO fail time is unlimited (for some tests this algorithm was even unable to initialize because of lack of features).

translates it to the computational module. A Pixhawk control module fits this design and it already includes INS and altimeter. Moreover, PX4 Flight Stack can also estimate states based on these data and fuse it with the one, calculated by the external module, in the role of the visual state estimation. Here it is worth being noticed that the computation module also fuses inertial and satellite data with the visual information, but passes the result in the figure of visual state estimation without any repercussions, just in order to fit the standard interface with the Pixhawk autopilot. ROS is helpful for organizing internal software modules, which run different algorithms (*Core Nodes*), as well as communications between the computation module and the autopilot (*mavros* interface) and camera (*usb_cam* interface). The described system design is represented in the Figure 4.

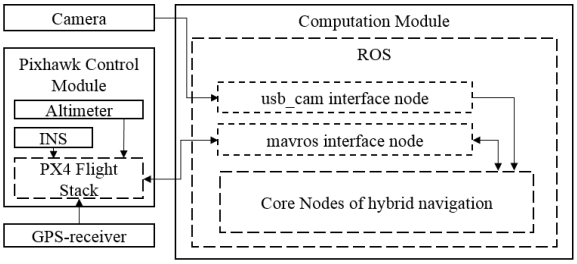


Figure 4: Software and hardware architecture of the hybrid navigation module.

4.2 Core Nodes Framework

Now let us look at the architecture of hybrid navigation core nodes in more detail (Figure 5). They have three input data flows: the first one is raw image shots by the

http://www.imavs.org/

http://www.imavs.org/

camera (*usb_cam/image_raw*), the second contains processed data from INS and altimeter (*mavros/imu*), and the third data flow represents the final fusion by PX4 Flight Stack of all state estimations incorporating the satellite data as well (*mavros/local_position*).

Visual navigation algorithms are run in the separate nodes and require monochromic undistorted images, which are rectified by *imag_proc* node. In case visual tracking is lost, the basic SVO and LSD implementation moves into the re-localization stage, trying to match incoming images with the explored scenes. The disadvantage of such an approach is that re-localization disables further tracking until the UAV returns in the previously visited area. We modified frameworks of these algorithms to force re-initialization instead of re-localization and notify MSF about it. However, such modification brought two new problems: firstly, the system resets the state estimation, so after re-initialization it works in a new coordinate system, and secondly, the system is losing information about motion during re-initialization, which could be critical if there are tiniest little changes in orientation or scale. We resolved these problems in the supervisor node. State estimations from SVO and LSD are fused with the inertial state estimation and atmosphere pressure in MSF, which is instantiated in two separate nodes (one for each visual navigation algorithm). The implementation of the MSF was also modified to do re-initialization once the corresponding notification is received.

The output data flow is formed by the Supervisor node in the figure of visual state estimation as it was mentioned in the previous section.

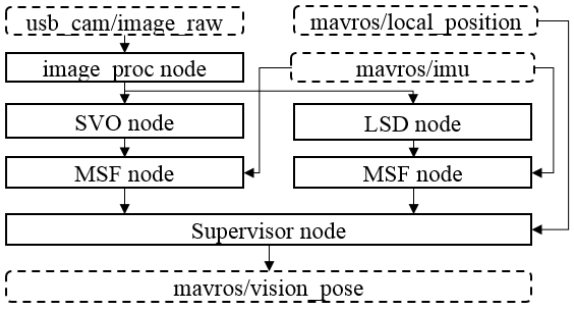


Figure 5: Hybrid navigation core nodes and their framework.

4.3 Supervisor Design

The supervisor design and framework are depicted in the Figure 6. Three callback functions (two of them process refined by MSF state estimations from SVO and LSD, the third one processes *local_position* via *mavros* interface) and the main thread are run in parallel. The callback functions receive messages from the corresponding nodes and write their info to the target buffers. The main thread firstly resolves the problems connected with re-initializations of visual navigation algorithms (Reinforced Navigation, elaborated in the next section). Here we emphasize that after this stage, state estimations originated from SVO and LSD are transformed in one consistent coordinate system. A wide class of algorithms now can be applied to fuse these states, from a simple states averaging smoothing through a Kalman filter to more advanced ones with, for example, applying machine learning techniques. The result of the fusion is advertised in *vision_pose* output data flow.

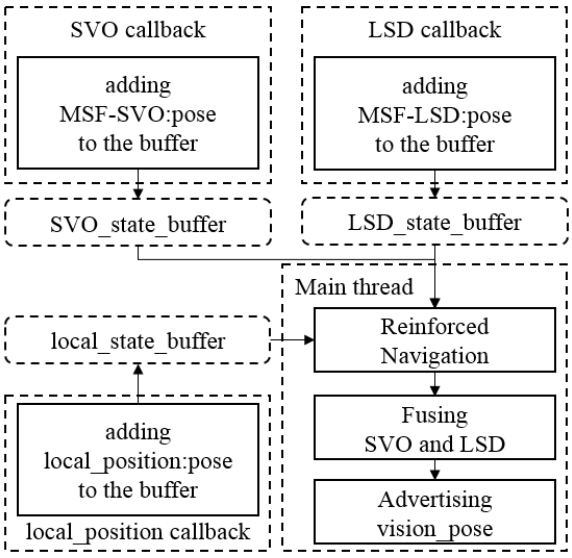


Figure 6: Supervisor node design and framework.

4.4 Reinforced Navigation

The Reinforced Navigation algorithm is based on the following idea: when one visual navigation node finishes re-initialization, state estimations from the other one, if it tracks motion correctly, can be used to supplement the lost information. In case both visual navigation algorithms are failed tracking simultaneously (assume that satellite data is not available too), the system can rely only on inertial and pressure sensors. However, it should be taken into account that commercial grade INS are reliable only for a few fractions of a second [15], thus we call such cases critical. Actually, the maximum duration of a critical moment depends on the scale of the mission, but generally, they shall be shorter than approximately 0.5–2 seconds.

5 DISCUSSION

The prototyped navigation unit has the total weight (approximately 500g) and components cost (approximately $600). These values can be considered as an upper bound estimation of the corresponding parameters of the off-the-shelf version, so here we imply that the system can be technically and economically viable for commercial small UAVs, which can carry payload from a few hundred grams and cost from a few thousand dollars.

We examined the system on its robustness by a number of tests with different environments and motion types and the demonstrated level is appropriate for large-scale outdoor applications. There are also several limitations connected with monocular implementation of visual navigation, the most important of them are: the observed scene should be almost static, sufficiently optically transparent and well-lit. However, despite these limits, the system is still suitable for the majority commercial applications, like agriculture, real estate, and infrastructure monitoring, since they use UAVs in good weather conditions during the daylight.

For the prototyping purposes we used only two visual navigation algorithms, however, the developed framework allows for adding other algorithms implemented as ROS node. Moreover the system robustness might be improved if different configurations of SVO or LSD are incorporated, which is also possible within the framework.

6 CONCLUSION

We developed a commercially suitable hybrid navigation unit, which includes three components; the first two are classical GNSS and INS, whereas the third one is vision-based, which is implemented via a monocular video camera. Visual information is processed in a separate powerful mini computer by two state-of-the-art algorithms for visual navigation, which are integrated and complemented to increase system robustness. Conducted tests showed an average two-fold increase in robustness in terms of visual tracking failure total durations. We also saw a slight gain in terms of average single failure duration time when the algorithms are combined. In addition, the developed framework allows using more algorithms limited only by the computational power.

REFERENCES

[1] Alberto Ortiz Francisco Bonin-Font and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of Intelligent and Robotic Systems*, 53:263–296, 2008.

[2] Farid Kendoul. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*, 29:315–378, 2012.

[3] Dji guidance. Website. Available at http://www.dji.com/product/guidance.

[4] Smart camera px4flow. Website. Available at https://pixhawk.org/modules/px4flow.

[5] Skybotix vi-sensor. Website. Available at http://www.ros.org/news/2014/05/skybotix-opens-vi-sensor-early-adopter-program.html.

[6] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[7] Jakob Engel, Thomas Sch ops, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision (ECCV)*, 2014.

[8] Stephan Weiss Margarita Chli Simon Lynen, Markus W. Achtelik and Roland Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[9] Robot operating system. Website. Available at http://www.ros.org.

[10] Pixhawk. Website. Available at https://pixhawk.org.

[11] Px4 pro stack. Website. Available at http://px4.io.

[12] Inertial position estimator inav. Website. Available at https://github.com/iNavFlight/inav/wiki/Inertial-position-estimator-(INAV).

[13] Intel nuc i5. Website. Available at http://www.intel.com/content/www/us/en/nuc/nuc-kit-nuc6i5syk.html.

[14] Playstation 3 eye. Website. Available at https://en.wikipedia.org/wiki/PlayStation_Eye.

[15] Ins and imu grades. Website. Available at http://www.vectornav.com/support/library/imu-and-ins.

http://www.imavs.org/

Control of a hybrid helicopter with wings

C. De Wagter* and E.J.J. Smeur†
Delft University of Technology, Kluyverweg 1, 2629HS Delft, the Netherlands

http://www.imavs.org/

ABSTRACT

This work investigates the design parameters and consequences in the control of a helicopter rotor combined with a pair of fixed wings. This hybrid vehicle has a light and aerodynamically efficient rotor that can achieve large pitch angles to allow forward flight. Because of the light stiff rotor and heavy wings, the hybrid vehicle exhibits couplings between the roll and pitch axes during hover flight. The rotor-wing interaction depends on a lot of parameters. In this paper, we utilize a simplified theoretic model and simulations in order to gain insight in the effect of these parameters on the vehicle dynamics. Finally, a controller is designed that compensates undesired coupling between pitch and roll.

1 INTRODUCTION

Rotorcraft dynamics have been well studied for many years[1, 2, 3, 4] with work ranging from rotorcraft modeling [5, 6], simulation [7], over matching measurement data with models [8], to the blade optimization in function of vibrations [9].

The design of controllers for conventional helicopters is well understood [10][11].

For less conventional designs like hinge-less low inertia rotors or high inertia fuselages, the design choices and control problems are more complex. Some studies into the fuselage-rotor interaction and fuselage ground interaction have identified resonance problems [12].

On the scale of small unmanned helicopters, models were used to identify which parameters affect the performance [13] and to reduce vibrations or noise [14]. Recent work presents a comprehensive non-linear model of a miniature unmanned helicopter [6].

1.1 Heli-Wing Hybrid

The approach of using a model to identify parameters important for the performance can also be applied to a vehicle such as a heli-wing hybrid. But when designing hybrid vehicles different parameters are important.

The subject of this paper is the design and control of a conventional cyclic and collective pitch controlled rotor on top of a fixed wing shaped heavy fuselage. Figure 1 shows the vehicle in semi transitioned attitude. The rotor allows the vehicle to hover while the wings and rotor design enable efficient fast forward flight.



Figure 1: Novel hybrid Unmanned Air Vehicle featuring a cyclic controlled rotor with wing.

The rotor blade design in the case of this vehicle is a compromise between efficient hover and efficient forward flight. This means the rotor is significantly different from rotors seen in conventional helicopters. It is relatively small, light, stiff and it has a high lift coefficient. During forward flight the *rpm* is reduced and pitch angle increased to generate propulsion efficiently. The wings which act as fuselage are large and systems are distributed over the entire wing, giving it a large moment of inertia.

These properties significantly affect the way the conventional helicopter rotor behaves during hover. On the other hand, during forward flight, the propulsion gyroscopic effect is much larger than usual, which also has an influence on the vehicle's dynamics.

In order to find an acceptable compromise between the dynamics in hover and forward flight, a mathematical model is created in Section 2. Design parameters are varied in Section 3. Section 4 investigates the interaction between rotor and fuselage and in Section 5 the model is used to gain insight in the flight test results of hovering flight.

*Email address: c.dewagter@tudelft.nl
†Email address: e.j.j.smeur@tudelft.nl

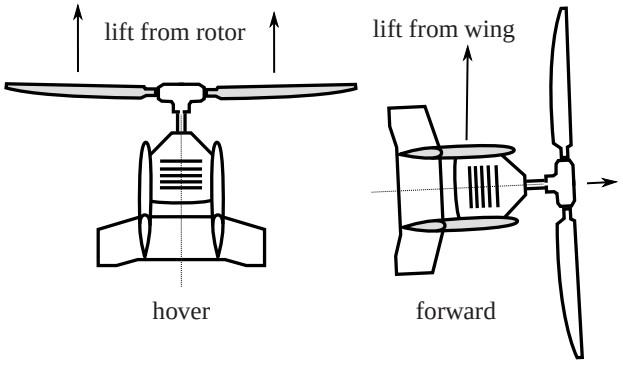


Figure 2: Hover and forward flight.

2 ROTOR MODEL

To explore the design options and their consequences, a helicopter model is derived mathematically. Figure 3 illustrates the basic rotor model [1]. The flapping angle β is measured around the spring hinge K. The rotor radius is R and it is spinning with a rate ω. The inputs are deflections of the swash-plate around the body X axis δ_x and Y axis δ_y. A positive input δ_x increases the pitch of the blade as it passes over the Y axis.

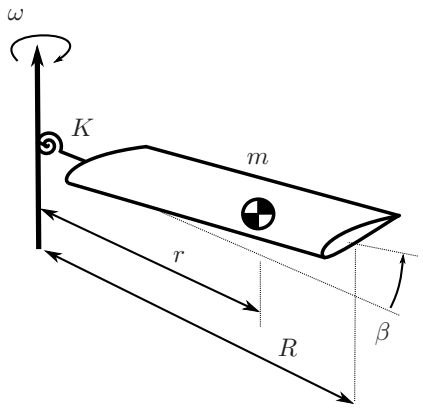


Figure 3: Simplified Rigid Rotor Model.

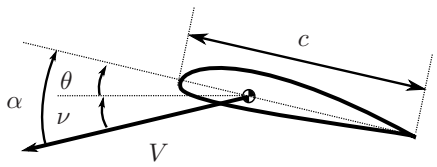


Figure 4: Angle of attack on a rotor section.

The forces acting on a blade element and the corresponding angles are illustrated in Figure 4. The model neglects the lagging angle of the rotor blade. The velocity V_x of the blade element then becomes ωr while the vertical speed of the blade element V_y is a function of the flapping rate $\dot{\beta}r$. The path angle ν is the arc tangent of V_x and V_y while θ is the feathering angle of the blade and is set with collective and cyclic pitch commands.

$$\bar{V} = \begin{pmatrix} V_x \\ V_y \end{pmatrix} = \begin{pmatrix} \omega r \\ \dot{\beta} r \end{pmatrix} \tag{1}$$

Since we are mainly interested in the lateral control properties, we will neglect collective pitch, meaning that there will be no induced velocity v_i and the angle of attack of a blade element is $\alpha = \theta + \nu$, and for small angles $\nu = \dot{\beta}/\omega$. The lift on a blade element becomes:

$$\delta L = \frac{\rho}{2}(\omega r)^2 c_{l_\alpha} \cdot c\delta r \tag{2}$$

in which c_{l_α} is the lift coefficient and ρ the air density. The *Lock Number* $\gamma = \rho c_{l_\alpha} c R^4 / I$ is substituted with I being the inertia of the rotor [1]. The radius r is made non-dimensionless as $x = r/R$ which yields:

$$\delta L = \gamma \frac{\omega^2 I}{R} \left(\theta - \frac{\dot{\beta}}{\omega} \right) \cdot \frac{1}{2} x^2 \cdot \delta x \tag{3}$$

The centrifugal force $\delta m \cdot r\cos(\beta) \cdot \omega^2$ and the aerodynamic force δL on a blade element together with the mass δm of the blade element yield the following moment equation at the hinge:

$$\int \delta L \cdot r - \int \delta m \cdot \omega^2 \cdot r\cos(\beta) \cdot r\sin(\beta) = \int r \cdot \delta m \tag{4}$$

δL is expressed in function of δr in Equation 3. δm can be expressed in function of a length unit δr using the rotor density ρ_{blade} and it's cross section $S_{cross\_section}$.

The hinge spring moment is given as $K \cdot \beta$. When linearizing Equation 4, filling the lift force from Equation 3 and integrating over a uniform rotor, the differential equation [1] for flapping around a fixed hinge is obtained:

$$\ddot{\beta} + \frac{\gamma}{8}\omega\dot{\beta} + (\omega^2 + \frac{K}{I})\beta = \frac{\gamma}{8}\omega^2(\theta) \tag{5}$$

Equation 5 relates the inertia of the rotor, the aerodynamic damping, the centrifugal and spring forces to the excitation. In simulation the rotor is defined as a finite number of sections and integral in Equation 4 is replaced by a summation over all segments.

The excitation using feathering angle θ is periodic and a function of the control inputs δ_x and δ_y

$$\theta = \delta_y \sin(\omega t) + \delta_x \cos(\omega t) \tag{6}$$

http://www.imavs.org/

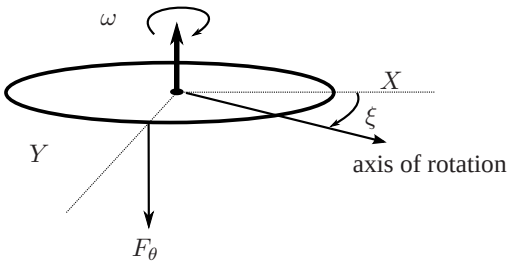


Figure 5: Applied force F_θ on a spinning rotor and corresponding axis of rotation at angle ξ. For non spinning bodies ξ is zero as a moment in the X axis will produce a rotation in the X axis. For pure gyroscopes ξ is 90 degrees.

3 DESIGN VARIATION

The derived model is analyzed in simulation. To gain insight into the design choices, several variables in the model that can be tuned in real life are varied over a range.

3.1 Hinged Rotor

Figure 6 shows the results for a rotor with $K = 0$. Parameters for the model can be found in Section Appendix A:. A deflection δ_y of 4 degrees is applied. After a few rotations[1], the rotor plane is tilted.

The deflection β has a phase lag of 90 degrees with the perturbation θ as is expected in gyroscopes. ξ (See Figure ref-figure:Advance) is equal to 90 degrees. The cyclic deflection in pitch decreases the feathering angle θ whenever the rotor blade is at the right position of the ω clockwise spinning rotor, to yield a rotor-plane inclination backward.

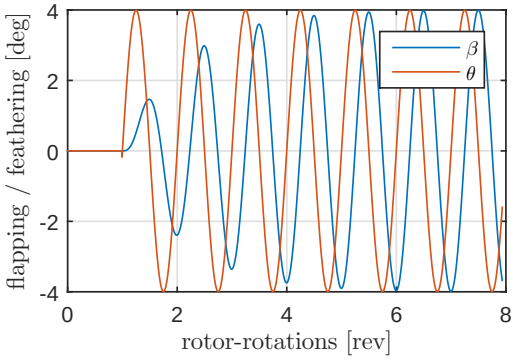


Figure 6: Blade dynamics based on cyclic deflection with zero spring stiffness K.

3.2 Stiff Rotor

When the stiffness of the spring K is increased, the dynamics of the rotor are significantly affected. Not only is the deflection β reduced, but also the direction flapping — in other words the rotor plane rotation — is affected.

Figure 7 shows the results for a rotor system with a stiff spring (high K). In this case the angle ξ, or in other words the phase difference between θ and β, is reduced from 90 degrees to below 30 degrees for a spring K which reduces the flapping angle β roughly by half.

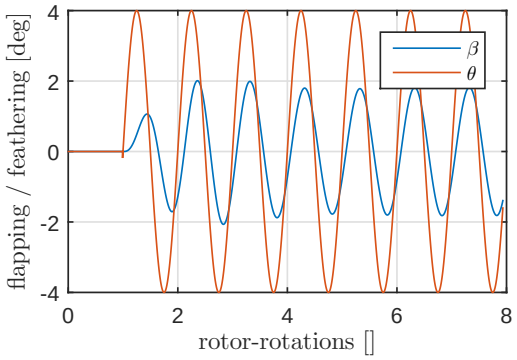


Figure 7: Blade dynamics based on cyclic deflection with significant spring stiffness K.

3.3 Design Parameters

From Equation 5 it can be found that the dynamics of the rotor are affected by its rotation rate ω, *Lock Number* γ and spring stiffness with respect to rotor inertia K/I.

For a given amount of lift, the *Lock Number* γ can only be altered by changing the rotor weight and will influence the speed of the response. High values of K can even change the direction of the actuator.

4 ROTOR HEAD AND FUSELAGE

4.1 Rotor Head Model

Since the previous sections have shown the importance of the hinge spring, the model of the actual rotor is investigated further. Figure 8 shows a cross section of the helicopter rotor head used in the platform from Figure 1. It consists of an aluminum head block on a main rotor shaft. The shaft holding both blades is only connected to the hub via rubber o-rings. A close up can be seen in Figure 13.

Modeling the bending of all parts of the rotor is complex. The exact bending of the feathering shaft, the blade grips and the blade itself contain a large number of parameters. Modeling how the force is transferred from the rotor system to the main rotor shaft is much easier as it only involves the rubber o-ring stiffness K_{o-ring} and their location l_{o-ring} as illustrated in Figure 9. For one blade the moment from the rotor on the rotor shaft becomes:

$$M_{rotor-shaft} = \Delta z_{o-ring} \cdot K_{o-ring} \cdot l_{o-ring} \qquad (7)$$

with

[1] The number of rotations depends on the *Lock Number* γ or in other words the damping to weight ratio.

http://www.imavs.org/

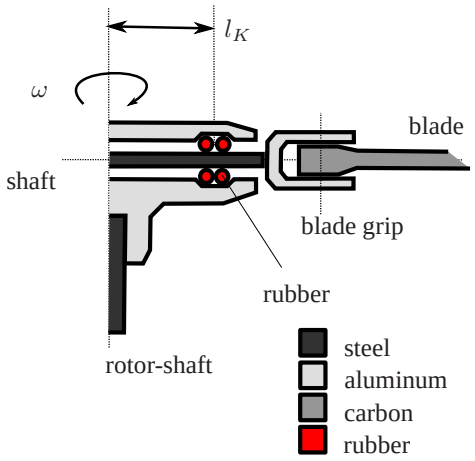


Figure 8: Actual Rotor Hub Simplified Schematics.

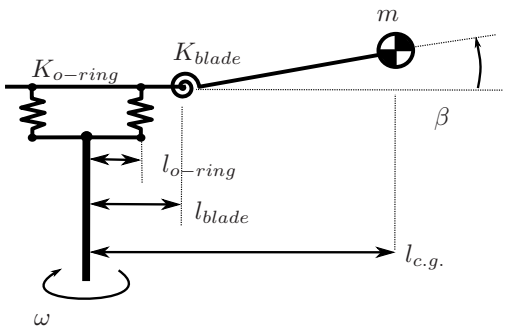


Figure 9: Simplified Rotor Hub.

$$\Delta z_{o-ring} = \frac{(M_{K_{blade}} + (\omega \cdot l_{c.g.})^2 m)}{l_{o-ring}} \tag{8}$$

4.2 *Constrained rotor-shaft motion*

When the rotor shaft is rotated in pitch or roll to simulate a fuselage change in attitude, the rotor plane is following the fuselage motion. Figure 10 depicts the simulation results of a 10 degree pitch up of the fuselage and main rotor shaft in the case of a pure hinged rotor with $K = 0$ and in the case of a bending rotor with K non-zero.

In the case of a fully hinged rotor, the rotor disc still tracks the motion of the rotor-shaft, as the feathering angle of the blade remains parallel to the rotor shaft through the swash-plate dynamics. When neglecting the blade grip push-rods and swash-plate forces, the moment from rotor to rotor shaft is zero.

On the other hand, whenever a moment can be transferred from the rotor to the rotor-shaft, it can be seen that the relative blade flapping angle β_{shaft} creates moments M_x and M_y with a frequency of 2ω, or two times the rotor frequency.

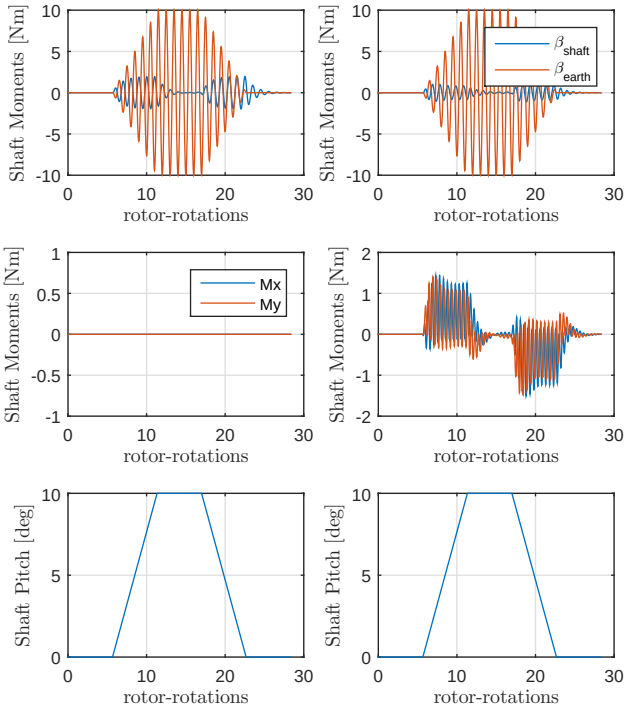


Figure 10: Rotor dynamics in function of fuselage imposed motion. Left $K = 0$, right $K \neq 0$.

It is interesting to note that a pure imposed pitch motion will generate moments in both M_x and M_y directions. The fuselage applies a moment to the rotor to make it pitch up, but the gyroscopic reaction of the rotor on that pitching motion is a rolling motion. This yields a roll moment from the rotor on the fuselage in return.

This is of particular importance in the case of a partially constrained fuselage. For instance, upon landing of the vehicle from Figure 1 with a roll angle, one tip will touch the ground first, yielding a constrained roll rate imposed on the rotor. As shown in Figure 10, this results in significant pitch moments imposed back from the rotor on the fuselage, which can cause undesired pitch motions caused by imposed roll motions.

Similarly, in forward flight, a pitching moment from the fixed wing and its elevons will cause an undesired yawing moment from the rotor back on the fuselage.

4.3 *Free fuselage dynamics*

To simulate the free fuselage dynamics, the fuselage is modeled as four point loads of a quarter of the total mass. The fuselage is symmetric around the X and Y axis but the dimensions are not equal.

The distance from the real center of gravity to each point load in the x direction is l_x and in the y direction l_y.

http://www.imavs.org/

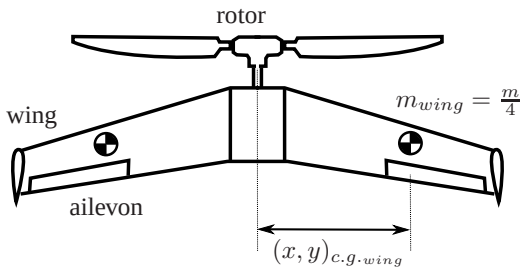


Figure 11: Body Model.

$$I_{xx} = \frac{m}{4} \cdot x_{c.g._{wing}} \tag{9}$$

$$I_{yy} = \frac{m}{4} \cdot y_{c.g._{wing}} \tag{10}$$

Assuming no aerodynamic forces on the wing during hover, no yaw rate and small angles, the fuselage rates are obtained through integration of the rotor shaft moments.

$$p_{fuselage} = \int_0^t \frac{M_x}{I_{xx}} dt \tag{11}$$

$$q_{fuselage} = \int_0^t \frac{M_y}{I_{yy}} dt \tag{12}$$

5 CONTROL CONSIDERATIONS

The derived models were applied to the rotor design of the hybrid rotor-wing vehicle depicted in Figure 1. Its light carbon rotor, mounted on a stiff conventional rotor head with high lift coefficient rotor blade airfoils — parameters given in Section Appendix A: — is not behaving like a conventional helicopter anymore.

To validate the model, real flight tests were performed. A governor was programmed to yield a constant rpm, to remove extra variables from the problem. Figure 14 shows the rpm is kept constant as soon as the vehicle takes off. The flight controller used is the paparazzi autopilot project [15, 16].

The vehicle can be flown with a simple proportional controller that controls the vehicle's angular rates. The output of this controller are cyclic commands, which are mapped to the three servos that control the swash-plate. The feedback was initially done with a 90 degree offset, such that roll feedback was applied to δ_x and pitch feedback to δ_y. δ_x produces a moment in the pitch axis and δ_y produces a moment in the roll axis. Due to gyroscopic precession this then produces an angular rate in the correct axis. This method of control is very common for helicopters [1].

For the vehicle in question, this controller did not provide control without couplings. Inputs of the pilot resulted in rates in a different axis than intended and on top of that a transient 'wobble', where the vehicle oscillates in both pitch

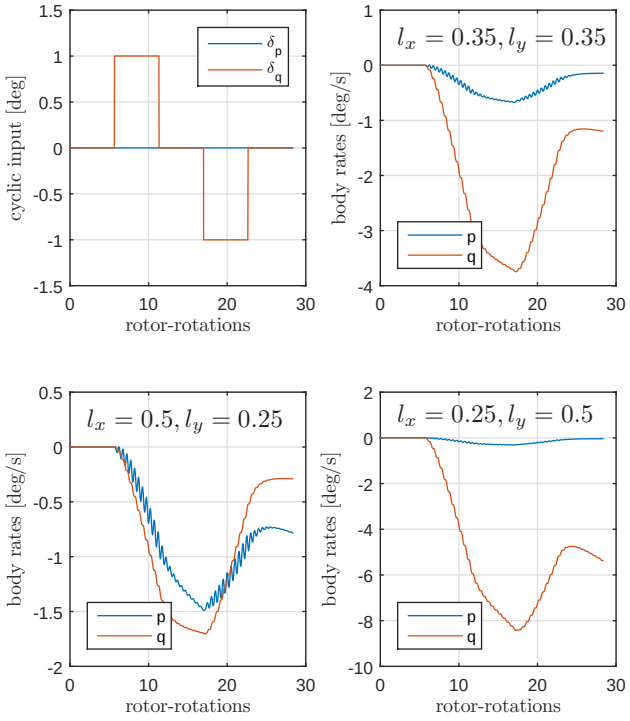


Figure 12: The influence of fuselage inertia on a free body. Three simulations of an identical rotor with identical non-zero stiffness and different inertia distribution of the fuselage.

and roll with a 90 degree phase difference. Using an onboard logging on an SD card, all turn rates and control deflections were logged. This data was used to model the angular acceleration in pitch and roll using the inputs δ_x and δ_y, the rates in roll and pitch and an offset $O = 1$. The model is shown in Equations 13 and 14, where f_p and f_q are linear functions of the parameters.

$$\dot{p} = f_p(O, \delta_x, \delta_y, p, q) \tag{13}$$

$$\dot{q} = f_q(O, \delta_x, \delta_y, p, q) \tag{14}$$

From the first data-set, it was discovered that the control mapping was incorrect, as δ_x and δ_y both caused a roll as well as a pitch acceleration. This was expected to cause some of the problems, which is why a second test flight was conducted using the correct control mapping.

Figure 15 shows the angular acceleration in roll and pitch along with the best fit of $f_p()$ and $f_q()$ for this second flight [17]. All signals were filtered with a second order filter with a cutoff frequency of 15 rad/s. The plot is taken during a 20 degree step in roll. From the figure it can be seen that the model fit is very accurate for this part of the flight. However, it is also clear from the figure that the transient wobble was still present. This is attributed to the effect of the rate on

http://www.imavs.org/

http://www.imavs.org/

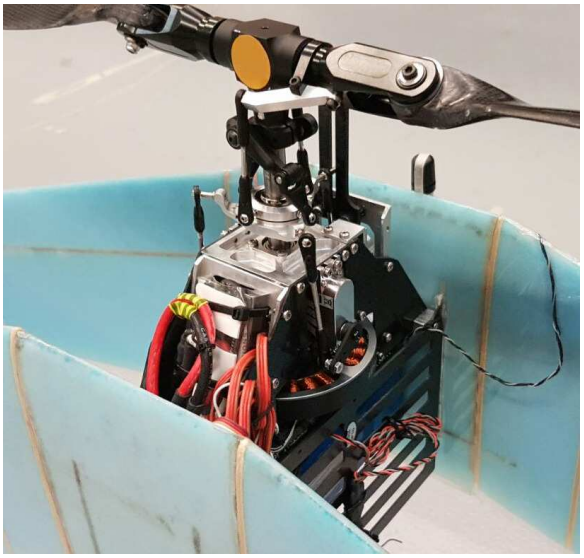


Figure 13: Close up of the rotor head of the hybrid winged helicopter vehicle. The swash-plate has 3 servos at 120 degrees from each other. Collective pitch has much greater reach than conventional helicopters and the blades have more twist to allow efficient forward flight.

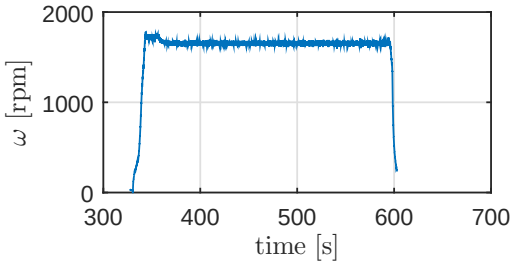


Figure 14: Test Flight RPM.

angular acceleration in the other axis. The coefficients that were found using the data shown in Figure 15 are shown in Table 1.

Note the coefficients for C_p and C_q. They state that a roll rate causes a pitch acceleration and vice verse. This might well explain the observed wobble. Therefore, in order to remove the wobble, the angular acceleration due to the rates needs to be canceled by a control input. The linear controller is revised to:

$$\begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = G^{-1} \begin{bmatrix} K_p p_{err} + q C_{q_{\dot{p}}} K_c \\ K_q q_{err} + p C_{p_{\dot{q}}} K_c \end{bmatrix} \tag{15}$$

Where p_err and q_err are the difference between the pilot command and the actual rates of the vehicle, and

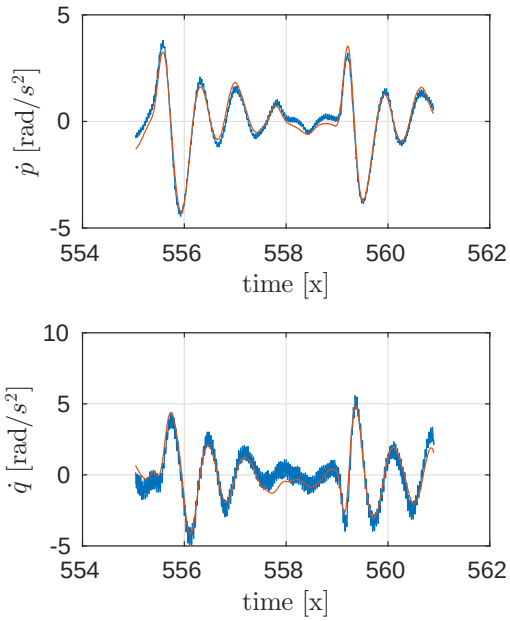


Figure 15: Fitting control inputs and body rates to body accelerations. In blue the filtered angular acceleration and in red the best model fit.

Coefficient	f_p	f_q
C_O	-2.4661	-2.8847
C_{δ_x}	0.0032	-0.0044
C_{δ_y}	0.0011	0.0073
C_p	-0.5703	7.4479
C_q	-3.4308	-3.4487

Table 1: Identified parameters.

$$G = \begin{bmatrix} C_{\delta_{x_{\dot{p}}}} & C_{\delta_{y_{\dot{p}}}} \\ C_{\delta_{x_{\dot{q}}}} & C_{\delta_{y_{\dot{q}}}} \end{bmatrix} \tag{16}$$

And K_c is a value between 0 and 1. K_c was introduced in order to gradually enable the compensation of angular acceleration due to rates. Test flights showed that a value of 0.5 gives better results than a value of 1. This may be caused by actuator dynamics, as a control moment can not be instantly generated when a rate is measured. More research is necessary to better explain why $K_c = 1$ still gives a wobble.

Figure 17 shows the measured angular rates of the vehicle during some pitch maneuvers in the first part of the flight and some roll maneuvers in the second part of the flight. The rates were filtered with a second order filter with a cutoff frequency of 25 rad/s. From this figure it can be seen that no wobble is present, and the motion in roll and pitch is uncoupled. Compare this to Figure 16, where $K_c = 0$ results in clear coupling between roll and pitch.

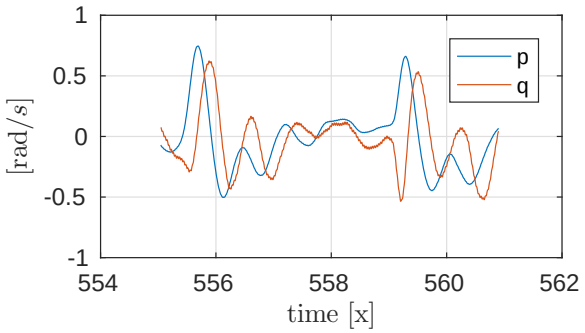


Figure 16: Flight with $K_c = 0$.

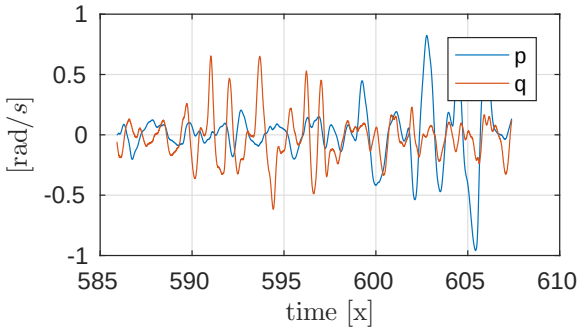


Figure 17: Flight with $K_c = 0.5$.

6 CONCLUSION

When designing hybrids between conventional cyclic controlled helicopters and fixed wings, it is crucial to understand the interactions between rotor and wing in order to optimize the design.

Lock Number and rotor hinge spring stiffness where shown to influence the speed and even direction of the control effectiveness.

Non-homogeneous inertia of the fuselage and fuselage-rotor interactions add non-symmetrical coupling between the pitch and roll axes. Compensation for gyroscopic effects was needed in the controller to remove this coupling.

ACKNOWLEDGMENTS

We thank the sponsors of the TU-Delft Outback Medical Challenge Entry to have made this research possible, and the designers of this airframe to have made this research necessary.

REFERENCES

[1] Anthony RS Bramwell, David Balmford, and George Done. *Bramwell's helicopter dynamics*. Butterworth-Heinemann, 2001.

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

[3] J Wagtendonk. Principles of helicopter flight by w. 1996.

[4] Wieslaw Zenon Stepniewski and CN Keys. *Rotary-wing aerodynamics*. Courier Corporation, 1979.

[5] Wayne Johnson. A comprehensive analytical model of rotorcraft aerodynamics and dynamics. part 1. analysis development. Technical report, DTIC Document, 1980.

[6] T. H. Lee G. Cai, B. M. Chen and K.-Y. Lum. Comprehensive nonlinear modeling of a miniature unmanned helicopter. *Journal of the American Helicopter Society*, 57(1):1–13, january 2012.

[7] Gareth D Padfield. *Helicopter flight dynamics*. John Wiley & Sons, 2008.

[8] Gloria K Yamauchi, Ruth M Heffernan, and Michel Gaubert. Correlation of sa349/2 helicopter flight test data with a comprehensive rotorcraft model. *Journal of the American Helicopter Society*, 33(2):31–42, 1988.

[9] David A Peters, Timothy Ko, Alfred Korn, and Mark P Rossow. Design of helicopter rotor blades for desired placement of natural frequencies. 1984.

[10] H Shim, Tak John Koo, Frank Hoffmann, and Shankar Sastry. A comprehensive study of control design for an autonomous helicopter. In *In: Proc. 37th IEEE Conf. on Decision and Control (CDC98*. Citeseer, 1998.

[11] Vladislav Gavrilets. Dynamic model for a miniature aerobatic helicopter. In *Handbook of Unmanned Aerial Vehicles*, pages 279–306. Springer, 2015.

[12] Robert A Ormiston. Rotor-fuselage dynamics of helicopter air and ground resonance. *Journal of the American Helicopter Society*, 36(2):3–20, 1991.

[13] FX Caradonna and Chee Tung. Experimental and analytical studies of a model helicopter rotor in hover. 1981.

[14] G Bernardini, E Piccione, A Anobile, J Serafini, and M Gennaretti. Optimal design and acoustic assessment of low-vibration rotor blades. *International Journal of Rotating Machinery*, 2016, 2016.

[15] 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.

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

http://www.imavs.org/

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

APPENDIX A: DATA

Parameters of the model are supplied below:

Variable	Value	Unit
R	51	cm
$r_{c.g.}$	30	cm
m	55	gram
c_{tip}	3.0	cm
c_{root}	5.7	cm
c_{l_α}	2π	cm
K	88	Nm/rad

Table 2: Blade parameters.

Variable	Value	Unit
$x_{c.g._{wing}}$	.8	m
$y_{c.g._{wing}}$	.12	m
m_{wing}	0.95	kg
m_{total}	3.9	kg

Table 3: Fuselage parameters.

To asses the stiffness of the central rotor block rubbers, a setup was created using a dummy weight to measure the increase in deflection as shown in Figure 18 with parameters from Table 4.

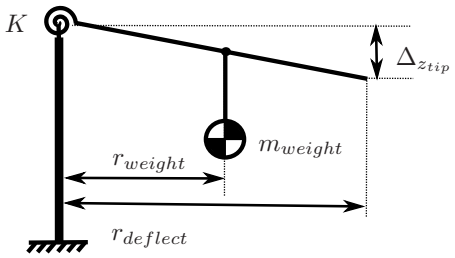


Figure 18: Stiffness measuring.

Variable	Value	Unit
m_{weight}	730	gram
r_{weight}	41	cm
$r_{deflect}$	41	cm
$\Delta_{z_{tip}}$	17	mm

Table 4: Central Rotor Block Spring Mounts.

http://www.imavs.org/

Design and Implementation of a Configurable Multi-rotor UAV

Yuying Qian[1]*, Kan Liu[1], Lijun Zhang[1], Junzhe Lin[1], Zixuan Lu[1], and Swee King Phang[2]
[1]Hwa Chong Institution, Singapore
[2]National University of Singapore, Singapore

http://www.imavs.org/

ABSTRACT

This manuscript details the design process and implementation considerations of a configurable multi-rotor unmanned aerial vehicle (UAV), which is switchable between quadcopter mode and hexacopter mode in easy plug-and-play action. The design includes customized detachable rotor arms, complemented with an altered arrangement of motors to enable the simple switching feature between the two types of platforms. An innovative flight control architecture is formulated to handle both types of platforms using a single control algorithm. Mathematical modelling of the platforms is crucial to the proposed flight control algorithm and it will be briefly discussed in this manuscript. The configurable platform is implemented and actual flying tests have been conducted in both configurations to verify the feasibility of the design.

1 INTRODUCTION

With the advancement of robotic technologies, unmanned aerial vehicles (UAVs) or drones today are recognized to have huge potential in upgrading future transportation network, especially in aerial delivery system [1, 2, 3]. Specifically, the electrical multi-copters characterized by vertical taking-off and landing (VTOL) appear to be the most promising choice for logistics industry [4]. The commercial outlook of UAV delivery has attracted many technology companies to incorporate this fashionable element into their services [5]. One of the most notable products of recent time would be Prime Air, a delivery system designed by Amazon, which promises to safely bring the package to customers' doorstep within 30 minutes [6]. The high efficiency of this solution also projects the possibility of goods delivery in emergency cases, e.g. drug shipment and registered mail consignments [7]. On the other hand, the National Aeronautics and Space Administration (NASA) of America is also looking into the prospect of the drone-aided disaster management. With the speculation drones hovering above the disaster zone and transmitting back the real-time data and visual images of the constant-changing situations, authorities will be able to give faster responses and optimize their disaster relief planning [8].

*Email address(es): qyy324@gmail.com

However, most UAVs in today's market, with a fixed range of payload capacity and a fixed number of rotor arms, are highly specialized in task execution. In the scenario, for example, where thousands of UAVs are employed for delivery tasks, since requirements on the loading capacity, delivery speed, flight duration and flight distance would vary case by case, the primary demand would be how to customize and deploy them to suit various needs of each delivery. Therefore, the problem with task specialization surfaces: dozens of models designed for different purposes need to be developed and mass produced to create a diverse pool of UAVs.

There are many existing researches seeking to improve the efficiency and flexibility of drone delivery system. On the hardware side, hybrid powering system is developed to meet varying power demands of the UAV. Managing the power from two energy sources into a single output is made possible with the smart *Power Manager* [9]. However, energy is not the only consideration to sustain a highly efficient delivery network. Hardware resources and its resilience are also important concerns for mass delivery. During peak periods, when UAV-network capacity is loading up to a saturation point (i.e. not enough vehicles to meet the service requests), even the optimal allocation of carriers in shipping process is not able to utilize them beyond the physical restrictions of the machines (the number of vehicles and their capacities). Hence, configurable drones will be at advantage to adjust themselves to suit the need of each and every task [10].

With this problem as our main motivation, a solution to enhance the adaptability of multi-rotor UAVs is proposed. Our solution makes use of detachable rotor arms of multi-rotor UAVs to support multi tasks with specific payload or flight duration. This manuscript provides detailed design methodology of an integrated multi-rotor UAV which is reconfigurable between two types of platforms. The paper is divided as follows: Section 1 given as introduction to this work; Section 2 shows a system overview of the solution; Section 3 details the hardware design and construction of the platform; a proposed universal flight controller will be proposed in Section 4; flight results will be shown in Section 5 and concluding remarks will be made in the last Section.

2 SYSTEM OVERVIEW

The proposed integrated system of multiple configurations provides several advantages over conventional multi-

http://www.imavs.org/

rotors:

1. *Versatility* - High power output from fully-loaded rotor arms enables larger loading capacity, faster speed and higher agility. Lower power output from partially-loaded rotor arms enables a longer flight duration.

2. *Usage* - High adaptability maximizes the potential of UAVs in all aspects of real-life applications.

3. *Portability* - The detachable mechanical parts can be disassembled and packed into a small package, saving space and reducing the possible damages to the delicate structure when being carried around.

4. *Cost-effectiveness* - Various multi-rotor models are integrated into one to achieve multiple purposes.

To start with the idea, a prototype is developed by adapting the most conventional configurations, quad-copter and hexa-copter, into one integrated system. This lays the foundation for integrating more configurations in the future. To achieve this, two major parts are considered:

1. *Physical structure* - The structure of the multi-rotor UAV needs to support multiple configurations, with specially important easy mount-and-remove characteristic on each of the rotor arms.

2. *Flight controller* - Conventionally, flight controller design for different types of platform will be slightly different. In this work, an all-in-one flight controller for different multi-rotor configurations is proposed.

3 HARDWARE DESIGN

As both quad-copter and hexa-copter are well developed and designed, the focus of this paper is on the integration of both platforms into one single UAV. The emphasis is thus on the connecting mechanism, which allows user to mount the rotor arms to the UAV body with simple plug-and-play fashion. Following this, an alternate configuration of quad-copter and hexa-copter rotors placement is then proposed.

3.1 Connection Mechanism

The adaptability feature of the designed UAV requires easy restructuring. In design, connection mechanism of the rotor on the UAV body should fulfil two criteria: detachability and stability. In other words, it needs to be easily reconfigured while minimizing vibration during the flight. However, detachability and stability appear to be conflicting goals in conventional mechanical designs. To solve the dilemma of being either locked up or weakly secured, after research and experiments, a new form of connection mechanism is developed.

Shock-absorbing rubber tube clamps are used to minimize vibration during the flight. They provide strong friction to fix the rotor arm position while acting as the cushion on the main platform. Rubber tube clamps also allow the users to freely detach or plug in the rotor arms.



Figure 1: Shock-absorbing rubber tube clamp

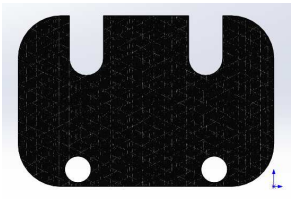


Figure 2: SolidWorks impression of the stabilizer

Since both the rotor arm and rubber tube clamps are circular-shaped, tube rotation is very likely to cause the rotor arm to deviate from its upright position. A stabilizer is specially designed to minimize the rotational shifting of circular-shaped rotor arms. One side of the stabilizer is fixed on top of the tube clamp, while the other side is plugged into the main platform to secure the relative position of the arm and UAV body.

The two additional mechanical designs, rubber tube clamps and stabilizers, take good care of the detachability and stability of rotor arms, allowing convenient installation and restructuring without undermining flying performance.

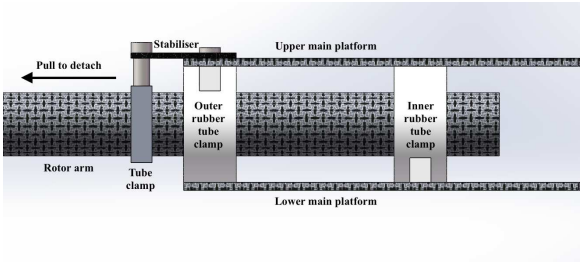


Figure 3: SolidWorks impression of the connection mechanism

3.2 Multi-rotor Configuration

This section describes the special configurations of the UAV in detail. To enhance the adaptability of the reconfigurable UAV, it is designed in such a way that, when users decide to restructure the UAV into a hexa-copter from a quad-copter, they do not have to change positions of any of the 4

http://www.imavs.org/

rotor arms of the quad-copter. Therefore the 4 rotors on the quad-copter are organized in a non-perpendicular X shape to align with the rotor positions of hexa-copter configuration. To be more precise, the 1st and 2nd rotor arms are 60 degrees from each other while the 1st and 4th are 120 degrees from each other, just as Fig. 4 shows. The 5th and 6th rotor arms can be simply inserted so that the six rotor arms are all 60 degrees apart.

Similar to normal quad-copters, the UAV has two of its rotors rotating clockwise while the other two rotating anti-clockwise in quad-copter mode.

To make the platforms easily interchangeable, the 3rd and 4th rotor arms of the hexa-copter is swapped (from conventional hexa-copter design) so that their rotating directions are consistent of that of the quad-copter platform. It is proved to be dynamically viable in Section 4 of the paper.

Overall, three main modifications are made in hardware design to realize the adaptability of both quad-copter and hexa-copter configurations: connection mechanism, non-perpendicular X configuration of quad-copter and rotor re-arrangement of hexa-copter. The platform is first designed in SolidWorks 3D simulation (Fig. 5 to Fig. 6) then implemented to actual platform.

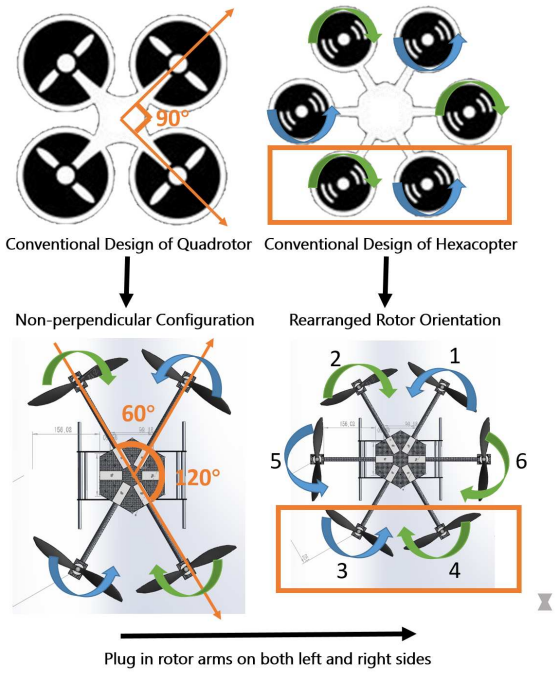


Figure 4: Graphical illustration of hardware modifications

4 2-IN-1 CONTROLLER DESIGN

In literature, mathematical modellings and flight controller designs for both quad-copter and hexa-copter are well



Figure 5: SolidWorks impression of quad-copter configuration



Figure 6: SolidWorks impression of quad-copter configuration

derived and documented [11, 12, 13]. The two models are, however, distinct and individual flight controllers were specifically designed for both models. In our integrated system of convertible quad-copter and hexa-copter, a novel idea of controlling both the platforms with a single controller is proposed in this manuscript. It will be discussed extensively in this section.

4.1 Overview

The overview of a typical flight control structure for multi-rotor systems can be visualized in Fig. 7 [14]. The proposed 2-in-1 controller uses the same structure as in Fig. 7, with 2 different Inner Multi-rotor Dynamics blocks: one for quad-copter, and one for hexa-copter. These two sets of inner dynamics will be automatically switched when a different configuration of the multi-rotor is detected.

Translational/rotational kinetics and outer multi-rotor dynamics of all multi-rotor UAVs are similar, and it is governed by the following equations [12]:

$$\begin{aligned} m\dot{V}_b + \omega \times (mV_b) &= F, \\ J\dot{\omega} + \omega \times (J\omega) &= M, \\ \dot{V}_b &= a_b, \\ \ddot{P}_n &= a_n. \end{aligned} \tag{1}$$

Here, m is the mass of the UAV which can be easily measured. J is the moment of inertia of the UAV which can be

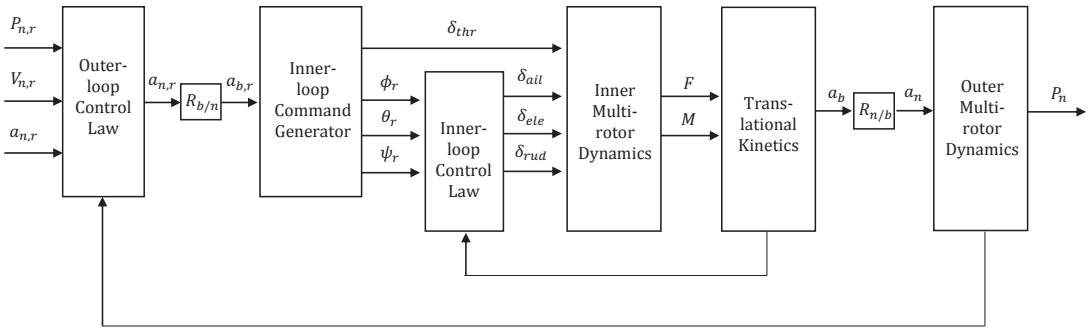


Figure 7: Flight control block diagram

determined from the trifilar pendulum experiment [12]. Both the m and J for quad-copter and hexa-copter configurations will be recorded and used in the unified flight controller.

For inner-loop command generator, it relates the body-frame acceleration reference $a_{b,r}$ to the Euler angle references $\{\phi_r, \theta_r\}$ and the normalized input to the throttle channel δ_{thr}, as

$$\begin{bmatrix} \delta_{thr} \\ \phi_r \\ \theta_r \end{bmatrix} = \begin{bmatrix} 0 & 0 & K_{thr} \\ 0 & 1/g & 0 \\ -1/g & 0 & 0 \end{bmatrix} a_{n,r}, \quad (2)$$

where K_{thr} is a constant gain depending on the thrust coefficient of the motors.

4.2 Inner Multi-rotor Dynamics

As the platform is inter-transformable between a quad-copter and a hexa-copter, the inner multi-rotor dynamics which generates the forces and moments acting on the UAV's body can be divided into 2 portions. The dynamical models of the two configurations will be similar as long as the inputs and outputs to this block remain consistent, namely the four normalized inputs $\{\delta_{thr}, \delta_{ail}, \delta_{ele}, \delta_{rud}\}$ and the forces and moments $\{F, M\}$ acting on the UAV.

By making both the inputs and outputs of this block consistent, it allows us to design a single controller to control both the systems, with just a simple switching of inner multi-rotor dynamics block and the parameters m and J as mentioned above.

By the assumption of fast motor rise time, we can approximate the relationship between the inputs and the outputs of this block to be static. In the other words, only the steady-state is concerned.

According to [14], the relationship between thrust produced by the rotating rotor and the rotating speed is

$$T = C_T \rho D^4 \frac{\Omega^2}{4\pi^2}, \quad (3)$$

where C_T is the aerodynamic thrust coefficient of the rotor, ρ is the density of air, and D is the diameter of the rotor. As for small fixed-pitch propellers, C_T is constant. We thus can lump the constants together as

$$T = k_T \Omega^2. \quad (4)$$

Similarly for torque of the rotor, we have

$$Q = k_Q \Omega^2. \quad (5)$$

4.2.1 Quad-copter

The relationship between the rotating speed of four rotors and the total forces and moments generated by the quad-copter, according to the quad-copter configuration shown in Fig. 4 can be easily derived as

$$\begin{bmatrix} f_z \\ m_x \\ m_y \\ m_z \end{bmatrix} = \mathbf{K_Q} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}, \quad (6)$$

where

$$\mathbf{K_Q} = \begin{bmatrix} -k_T & -k_T & -k_T & -k_T \\ -\frac{lk_T}{2} & \frac{lk_T}{2} & \frac{lk_T}{2} & -\frac{lk_T}{2} \\ \frac{lk_T\sqrt{3}}{2} & \frac{lk_T\sqrt{3}}{2} & -\frac{lk_T\sqrt{3}}{2} & -\frac{lk_T\sqrt{3}}{2} \\ k_Q & -k_Q & k_Q & -k_Q \end{bmatrix}. \quad (7)$$

Each motor of the UAV can be represented by

$$\Omega(s) = \frac{1}{\tau_m s + 1}(C_R \delta + \Omega_{trim}), \quad (8)$$

where τ_m is the motor time constant, which is generally small enough to be ignored, and C_R is the steady-state gain of the motor rotating speed given the input δ.

http://www.imavs.org/

Finally, a mapping matrix to map the block's inputs $(\delta_{thr}, \delta_{ail}, \delta_{ele}, \delta_{rud})$ to the motors' input $(\delta_1, \delta_2, \delta_3, \delta_4)$ as follows:

$$\begin{bmatrix} \delta_{thr} \\ \delta_{ail} \\ \delta_{ele} \\ \delta_{rud} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \end{bmatrix}. \quad (9)$$

Note that in order to obtain a unique solution of the inputs $(\delta_{thr}, \delta_{ail}, \delta_{ele}, \delta_{rud})$ given any desired forces and moments (F, M), $\mathbf{K_Q}$ must be invertible.

4.2.2 Hexa-copter

Similar to that of the quad-copter counterpart, the relationship between the rotating speed of six rotors and the total forces and moments generated by the hexa-copter, according to the hexa-copter configuration shown in Fig. 4, can be derived as

$$\begin{bmatrix} f_z \\ m_x \\ m_y \\ m_z \end{bmatrix} = \mathbf{K_H} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \\ \Omega_5^2 \\ \Omega_6^2 \end{bmatrix}, \quad (10)$$

where

$$\mathbf{K_H} = \begin{bmatrix} -k_T & -k_T & -k_T & -k_T & -k_T & -k_T \\ -\frac{lk_T}{2} & \frac{lk_T}{2} & \frac{lk_T}{2} & -\frac{lk_T}{2} & lk_T & -lk_T \\ \frac{lk_T\sqrt{3}}{2} & \frac{lk_T\sqrt{3}}{2} & -\frac{lk_T\sqrt{3}}{2} & -\frac{lk_T\sqrt{3}}{2} & 0 & 0 \\ k_Q & -k_Q & k_Q & -k_Q & k_Q & -k_Q \end{bmatrix}.$$

Note that the motor dynamics and mapping matrix are identical to the quad-copter's part.

In order to obtain a unique solution of this hexa-copter system, we need to find an inverse of $\mathbf{K_H}$. As $\mathbf{K_H}$ is rank 4, one solution to the system would be taking the right inverse of $\mathbf{K_H}$, i.e.

$$\mathbf{\Omega} = \mathbf{K_H^T}(\mathbf{K_H K_H^T})^{-1} \begin{bmatrix} f_z \\ m_x \\ m_y \\ m_z \end{bmatrix}. \quad (11)$$

4.3 Control

Once the control structure of the system is fixed as in Fig. 7, controllers can be designed accordingly. There are two separated controllers in the system, the inner-loop controller and the outer-loop controller.

Inner-loop controller helps stabilize the UAV and orientate the platform to any desired Euler angle reference. As the inner-loop controller runs at a much faster rate as compared to the outer-loop, a position controller can be designed by assuming any desired Euler angle can be achieved within a single control loop on the outer-loop. This control strategy is well established and widely used in UAV control and thus it will not be further elaborated here. Interested reader can refer to [12, 14].

5 PROTOTYPING



Figure 8: Assembled hexa-copter configuration



Figure 9: Assembled quad-copter configuration

A UAV with the transforming capability was constructed according to the SolidWorks impression shown in Fig. 4. Main parts of the UAV's body are made of carbon fiber, while a Pixhawk flight controller is included to implement the proposed 2-in-1 controller. Specification of the UAV of both configurations matched with a 4 cells 5200 mAh battery is shown in Table 1. The design specification is met with actual flying experiment of the UAV in its two different modes.

	Quad-copter	Hexa-copter
Weight	1532 g	1796 g
Payload	100 g	600 g
Flight duration	23 mins	15 mins

Table 1: Specification of the platform

http://www.imavs.org/

6 CONCLUSION

The paper elaborates in detail the design and modelling of a configurable multi-rotor UAV. Rotor arms are arranged into the formation adaptable to both quad-copter and hexacopter flights. Connection mechanism enables easy detachment or installment of arms. Mathematical modelling proves the viability of modified configuration in flight kinematics and forces and moments generation. The configurable multirotor UAV performs well in actual flight tests with different payloads. In future, more configurations can be integrated into this UAV platform to further enhance its versatility. Autonomous control law can also be implemented such that the UAV would fly automatically with the assistant from GPS. With this UAV prototype as a foundation, future UAV platforms are able to operate with adjustable loading capacity, flying speed and flight duration over a wider range. When in service among large UAV fleets, each is more energy efficient and adaptable in catering to its specific task; when in private use, configurable UAV is flexible enough to handle personal requests of all sorts.

REFERENCES

[1] B. Gupta. Aerial delivery systems and technologies. *Defence Science Journal*, 60(2):124–136, 2010.

[2] R.A. Tiaden O.A. Yakimenko, N.J. Slegers. Development and testing of the miniature aerial delivery system snowflake. In *20th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*, 2009.

[3] R. Benney G. Brown. Precision aerial delivery systems in a tactical environment. In *18th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*, 2005.

[4] Matthias Heutger. *Unmanned aerial vehicles in logistics*. DHL Customer Solutions & Innovation, Troisdorf, Germany, 2014.

[5] Bill Canis. *Unmanned aircraft systems: commercial outlook for a new industry*. CRS Report, 2015.

[6] Amazon prime air. https://www.amazon.com/b?node=8037720011. Accessed: 2016-06-01.

[7] G. Paravati A. Sanna c. Demartini A. Lisanti G. Venezia V. Gatteschi, F. Lamberti. New frontiers of delivery services using drones: a prototype system exploiting a quadcopter for autonomous drug shipments. In *2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSAC)*, 2015.

[8] V. Ambrosia J. Brass R. S. Dann S. Wegener, D. Sullivan. Development and implementation of real-time information delivery systems for emergency management. In *International Global Disaster Network (GDIN) Information Technology*, 2000.

[9] Emerging technologies. https://www.dso.org.sg/ cmsresource/20121116474501748085.pdf. Accessed: 2016-06-01.

[10] R. Kusumo A. K. Sinha. A system framework to automate design decision support of re-configurable uavs. In *23rd International Congress of Aeronautical Sciences*, 2002.

[11] F. Wang M. Lan B. M. Chen S. K. Phang, S. Lai. Systems design and implementation with jerk-optimized trajectory generation for uav calligraphy. *Mechatronics*, 30:65–75, 2015.

[12] K. H. Yu B. M. Chen T. H. Lee S. K. Phang, K. Li. Systematic design and implementation of a micro unmanned quadrotor system. *Unmanned Systems*, 2(2):121–141, 2014.

[13] C. Milazzo A. Ricciardello L. Trefiletti A. Alaimo, V. Artale. Mathematical modeling and control of a hexacopter. In *2013 International Conference on Umanned Aircraft Systems (ICUAS)*, pages 1043–1050, 2013.

[14] T. H. Lee G. Cai, B. M. Chen. *Unmanned rotorcraft systems*. Springer, London/New York, 2011.

http://www.imavs.org/

Development of a Quadrotor with an original shape in order to do rescue applications

Eugénie Soerensen, Rapahël Schruoffeneger, Nestor-alonso Santos Ortiz, Renaud Kiefer, Marc Vedrines, Stéphane Vujasinovic, Loïc Imperiale, Martin Meyer, Thomas Pavot, Guillaume Simonet

INSA Strasbourg, Strasbourg, France

http://www.imavs.org/

Abstract

This project is included in Mechatronics engineering education at the National Institute of Applied Sciences (INSA) of Strasbourg. It consists in designing, building and controlling a Micro Aerial Vehicle with quadrotor structure in composite materials. In this article, the design, manufacturing and control approach is detailed.

The drone is completely homemade. The choice of an innovative shape and the research of a reliable material will be developed. The MAV is able to drop a parcel or carrying and fulfilling a water container and contains image sensors for precision landing and cartography applications.

This article will also talk about embedded electronics and control, control done with a LabVIEW interface we designed.

The link between the work of the team and the competition is also made. Indeed, the specification of the Quadcopter project at school coincides with the one of the outside competition. A closer look will be taken at the Bambi-Bucket, a system imagined by the team to collect water for the "Sample water around the oil rig and then release the water into a specified water container" mission.

1 Introduction

This project have been developed in the INSA Strasbourg's school, a French engineering school based in Strasbourg. It began five years ago with the aim of designing and creating an innovative quadrotor. The first team was composed by ten members who worked on this project as a part of the mechatronic cursus (four hours per week). A new team of six members will take over the project as an internship this summer in order to attend the IMAV competition. The following pages will present the work done the first team and the former ones.

First the innovative and optimized body will be presented. Then will come the electronical and programming part to describe how the MAV is from a far fully controlled. Finally, according to the competition specifications, the Quadrotor adapted for rescue missions part will be developed.

http://www.imavs.org/

2 An innovative and optimized body

The entire structure of the MAV has been thought out to be as light as possible to carry the maximum amount of rescue objects. It is well known that MAV structures should be as light as possible to be able to add lots of sensors, cameras, different kind of payloads and high capacity batteries [1]. Our quadcopter has a specific design and is completely manufactured in composite structures which provide excellent mechanical properties for such application.

1) Global design of the Quadcopter

The plates have been designed in a drop form. Indeed, they are not circled to be less fragile to eventual impacts. They have been thought out to be the more ergonomic as possible.

The layout of the plates is noteworthy compared to drones made with carbon tubes. Indeed, the electrical parts are protected between these plates [2].
Vibrations are also better absorbed with this structure.

Below is a screenshot of the latest quadcopter CAD version:



http://www.imavs.org/

Figure 1: CAD of the Drone

2) Arms design and manufacturing

The structure of the arms is composed of two rotational symmetric elements, made of a three-layer sandwich composite material, based on carbon-kevlar tissue and airex, in an epoxy matrix [4]. This composite provides excellent mechanical properties for such applications. Its special shape allows arms work in traction and compression, in conjunction with traditional bending, and demonstrated great strength, a very good vibration absorption and a very weak distortion at the end of the arms. Moreover, the structure embraces and protects the embedded electronics while allowing free access to components and a modularity of the elements that provides easy and fast intervention. The minimalist and uncluttered characteristic of the structure gives us a very good compromise between protection and weight. The two main parts of the structure, strictly identical, were produced by vacuum lamination with a mold realized on a Computer Numerical Control machine. The two elements of the structure are ready for use and provide an easy assembly thanks to the direct integration of fastening systems

Below is a CAD view of an arm assembly:

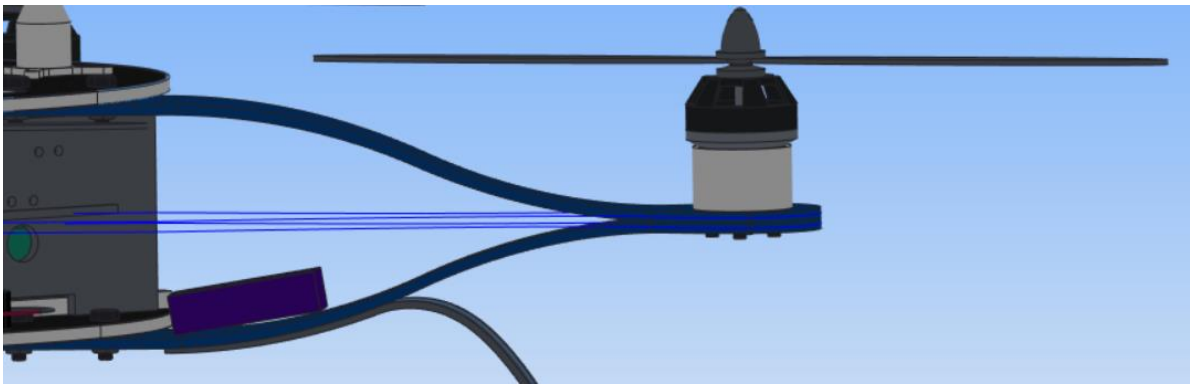


Figure 2: CAD of one arm assembly fixed to the MAV

The arms have been designed to be screwed on the shell made by the plates described above.

The machining and fabrication of the MAV's own arms shape is done at the INSA. It is designed with composite materials such as Kevlar, carbon and fiberglass. All molds are self-made, designed and tooled in the school.

Arms are made of a sandwich composite material composed of a symmetrical structure made of a layer of aramid fibers textile (trademark Kevlar) between two layers of cross linked polymer foam (Airex C71). This structure is placed between two layers of Carbon fibers textile, as described in the figure below.

http://www.imavs.org/



Figure 3: Sandwich composite structure of the arms

This kind of composite structure is widely used in aerospace design [3]. This sandwich advantage is due to the strength and stiffness properties of Airex combined to the high rigidity of carbon fibers. The central layer of aramid limits the risk of shredding of arm in case of shocks. The other advantage of the Airex is its high vibration absorption possibilities. It also helps to limit the weight of the sandwich and absorbs low frequencies vibrations.

Sandwiches we use for our tests and for our quadrotors are composed of two layers of carbon fiber (65g/m² Torayca® T300J) and one layer of carbon aramid (210 g/m² twill wave) between two layer of 1.2mm of Airex C71 and an Epoxy matrix (Resin L from R&G composite)

Arms molds have been modeled using a CAD CAM 3D software (CREO). This 3D model has then been used with a CAM software in order to machine the mold on a CNC Milling Machine. The material of the mold is a high density tooling board (Figure 5).

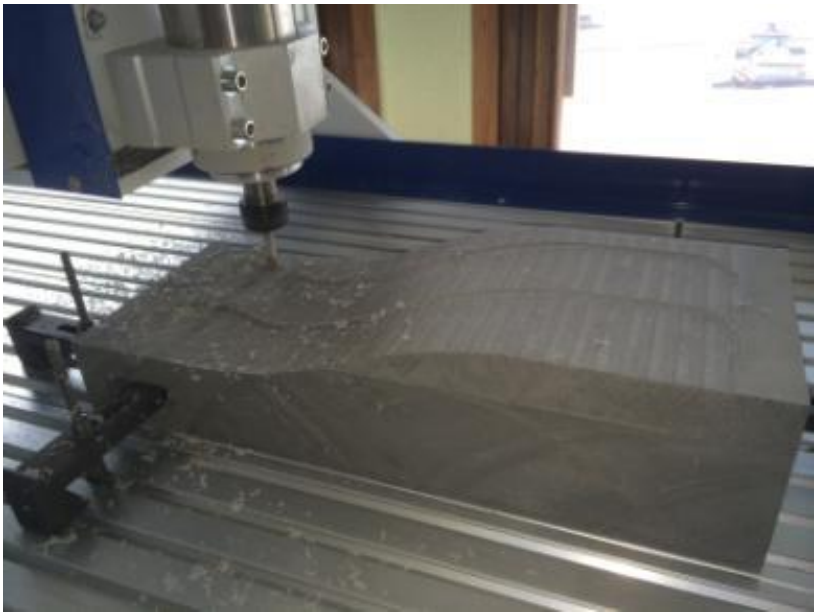


Figure 4: Arms molds are tooled in high density tooling board

http://www.imavs.org/

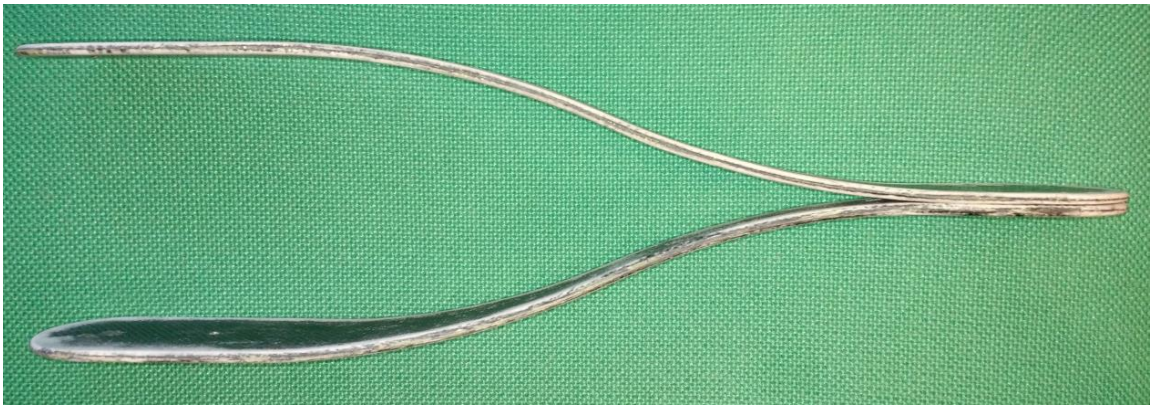


Figure 5: Asembly of one arm, each part molded at the INSA

Several vibration, resonance measurements and simulations have been made to determine the behavior of the arms through different conditions and to avoid any vibrations or resonances due to motors and propellers rotations.

Specimen have been manufactured and tested using a tensile test machine according to norm ASTM D 3039 (Tensile Properties of Polymer Matrix Composite Materials). Tensing tests have been done on (Zwick/Roell tensile machine) with two kind of test specimens with carbon fiber oriented at 90° and 45°.

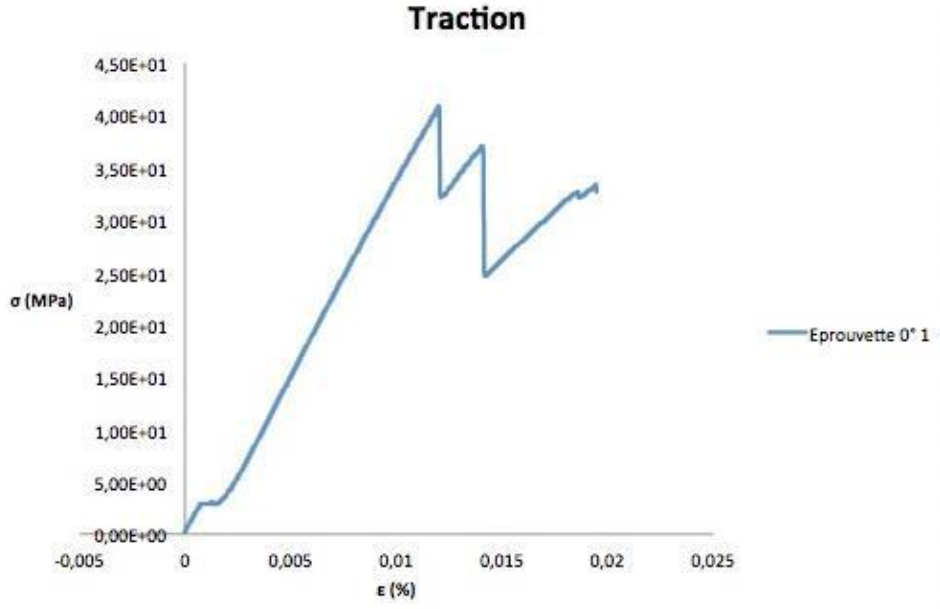


Figure 6: Example of tensile results on 90° carbon composite sandwich. The mean Young modulus obtained on 3 samples is E= 3832 Mpa

Finally after multiple measurement we obtain an average Young's modulus of E= 3832 Mpa with 90° oriented carbon fibers (longitudinal direction of arms). The density of the specimen is 0,4971g/cm3. This composite specimen is a stratified material and has an

http://www.imavs.org/

orthotropic behavior: the Young modulus in 1 and 2 direction (in plane modulus) are E1=E2 = 3832 MPa.

The tensile tests below (Figure 5) is performed on two different types of plates we found efficient. The test have been made on four samples of 2C-A-A-2C and on four samples of 2C-A-C-A-2C, at 0° and 45°. Details about each curve can be found in the table below. Then, the two Young's moduli have been extracted in the plane of the plate (E1 and E2) to calculate the stress.

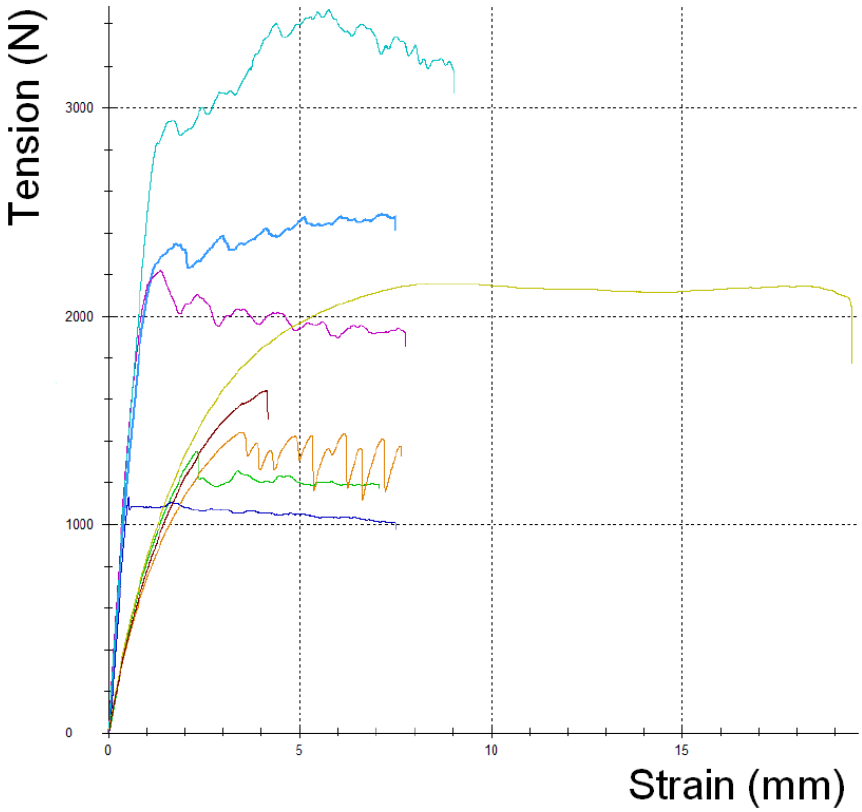


Figure 7: Tensile tests made on the arms

	0°	45°	Young's moduli
2C-A-A-2C	Blue, Dark blue	Green, Orange	E1=E2=3879.5±348.5 MPa
2C-A-C-A-2C	Violet, Red	Clear blue, Yellow	E1=E2=4353.5 ±83.5 MPa

The test below was performed over a frequency range from 0 Hz to 200 Hz on the material we decided to use. After the solicitation, we observe two resonance modes, one at 14Hz and one at 180Hz.

http://www.imavs.org/

A resonance of 95dB is observed at 14Hz and 85dB at 180Hz. Knowing that traditional propeller frequencies propeller are between 70Hz and 120Hz, we can assume that there is no resonance problem with this structure.

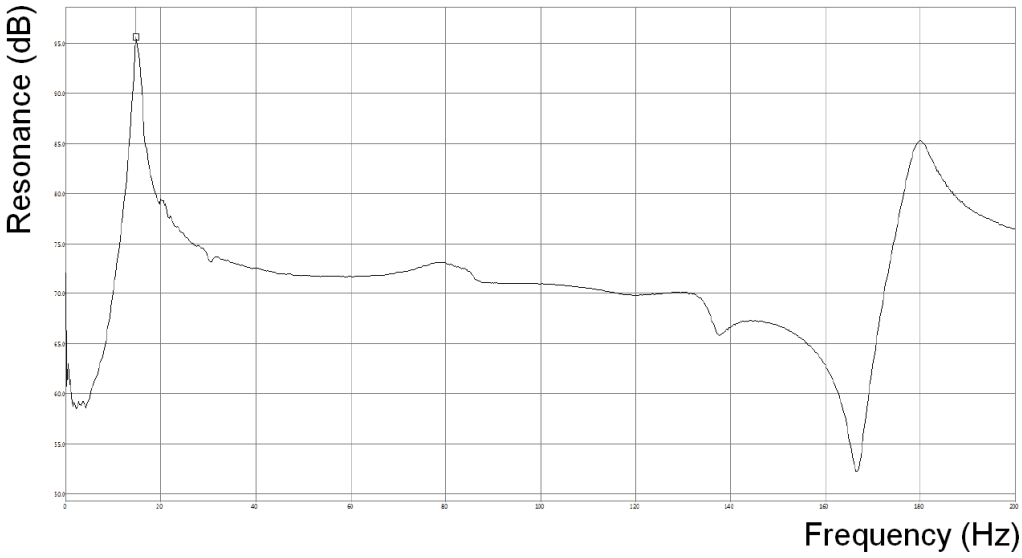


Figure 8: Resonance test made on an arm

Static analysis have also been carried out, with a load corresponding to the resultant of the motor weight and thrust (10 N, which is the worth case) to determine the arms structure behavior.

More details concerning arms design and characterization are described on this paper: [4].



Figure 9: Photography of the MAV we are continually improving

http://www.imavs.org/

3 A MAV from a far fully controlled

1) Global electronic design

The control board is an APM 2.8. It controls the stabilization of the MAV in flight and the auto piloting through the APM:Copter (formerly called the ArduCopter). Basic parameters are configured via the software Mission Planner. However, we are trying to completely integrate configuration in a stand-alone program which we will explain later. For the autopilot, as usual we use GPS, barometer and IMU. These sensors are completed by a network of ultrasonic sensors and an Odroid board for image processing. An XBee module pair is used to communicate the orders from the ground control station (GCS) to the drone, and also the feedback in the opposite direction.

The embedded camera, a Mobius 1080p, is mounted on a 2-axis gimbal stabilizer allowing it to counterbalance roll and pitch rotations and therefore having a better image. The gimbal is also able to rotate towards specific angles by putting offsets to the commands. Images from the embedded camera are transmit by specific analog emitter to the GCS. A newly developed ability concerning the drone's vision is the target tracking. The drone will be able to follow a mark on the ground and land on it. The image processing will be done thanks to an on-board ODROID, which will simply return the relative movements of the drone to follow the target.

Several electronic boards are currently designed to add some functions such as the Bambi-bucket controlling. The function of this board simply consists in commanding the servomotor that drives the pulley in rotation. When the APM sends a certain signal to this board, the bambi-bucket will have to go down until a fixed distance or go up until the cable is totally rolled up. Similarly, we consider to develop a motorized hook to automatically drop a life belt, which also comes with its own controller board.

Finally, to make the drone able to see its environment, a sensor board will be developed to gather distance measures (such as ultrasound or infrared sensors) and send it to the APM. A measure of the distance to the ground will also enable a better precision of the altitude servoing.

Here is a recap of all the electronic systems interacting with each other:

http://www.imavs.org/

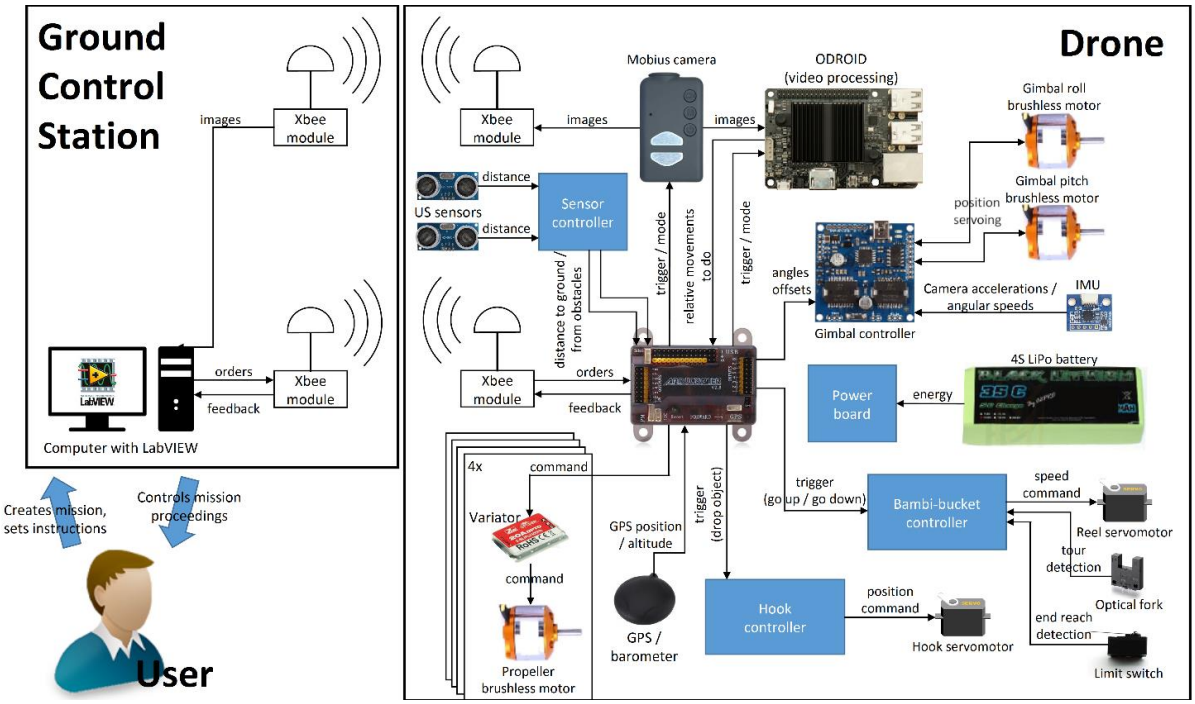


Figure 10: Overview of the electronic of the MAV

2) Description of the programming part

The goal of the main control program is to allow a user to create and send a mission to a drone equipped with an APM control board, so it can fly autonomously. While there are already some open source interfaces or software such as Mission Planner to communicate with the APM, their capabilities are sometimes limited in order to deal with the kind of mission that our drone will have to face during the competition. Thus, the development of a new interface gives us the possibility to integrate all capabilities needed. A LabVIEW interface has been set up since this platform offers an easy environment of development for engineering applications while keeping quality and robustness

In order to communicate with the APM, a set of LabVIEW functions allowing to send and receive different type of messages has been developed based on the standard communication protocol known as MAVLink. These modules enable for example to send a mission item to the memory of the APM. Once the drone starts the mission, it executes each command saved on its ROM and performs an autonomous flight. The basic commands that can be saved in a mission are commands to take off over a specific altitude, navigate to a specific GPS location and land. In addition, the communication functions allow the interface to read and decode messages sent by the drone in order to display a feedback of its status and location on the screen. Likewise, there are commands to set PWM and relay

http://www.imavs.org/

outputs of the APM, which can be used, for example, to control the angle of the camera gimbal or to close and open the water tank (Bambi-Bucket). As a result, these functions can then be easily integrated into the main LabVIEW interface.

3) Mission creation and transmission strategy

In order to create a mission, the graphical user interface has a list where the user can add new instructions for the drone. Thus, each instruction is ordered and described in the list. It will be possible to pick instructions from a toolbox, edit it and then place it wherever we want in the program.

To make the mission creation more intuitive, the user is able to click on a map generated from a public API (GMap) to automatically add new waypoints to the list of instructions.

Moreover, a mission can be saved and loaded from the computer. As all mission instructions are stored in objects, an XML structure is used to convert them into text data.

In order to send instructions to the drone, two different strategies are considered. The first, the simplest, acts like Mission Planner. That is to say, it sends all instructions at the beginning to the drone, so the latter can store them in its memory and then execute them one after the other. However the instructions are limited to the simple mission items provided by the MAVLink protocol. Impossible then to make the drone perform complex decisions during the flight. This mission mode is currently the first priority in development.

Then comes the other mission mode, which enables the interface to send new instructions during the mission execution. That way, the drone can send its situation to the GCS and the latter can make the complex decisions or treatments and send new orders according to the results. This includes for example QR-code reading during a mission.

http://www.imavs.org/

4 A Quadrotor adapted for rescue missions

1) **Take-off (and landing) from a moving and rocking platform.**

The take-off and landing is aimed to be done fully automatically through the LabVIEW interface described upper.

2) **Do a fast mapping of the accident area to evaluate the whole situation and then find designated targets, like the location of the people in the water**

The mapping and the finding of the targets will also be made through the LabVIEW interface using the Google Maps module, as described earlier. The image processing will be made coupled with the APM Card.

3) **Deliver lifebuoy to the person's vicinity**

The lifebuoy delivery will be made thanks to a hook designed for the IMAV2015 parcel delivery. This hook will be fitted to the specifications of this test once more information will be released.

4) **Sample water around the oil rig and then release the water into a specified water container.**

For the « Sample water around the oil rig and then release the water into a specified water container » mission, a system of water recovery inspired from the fire-fighting planes ones, namely the Bambi-bucket has been imagined.

Below is a picture of the MAV taken in flight during a water sampling test:



Figure 11: Bambi-Bucket attached to the MAV

http://www.imavs.org/

The system chosen is reputed simple (one servomotor and no pump used) and lightweight. This system is set in the shell of the two plates.

The principal rope, controlled by a pulley/servomotor system stretches to lift the background of the bucket. The purpose of the other small rope in the inside of the bucket is to open the bottom for the water sampled to be relieved. Through gravity the water pushes the bottom to fill the bucket, there is no need for an actuator in this part.

Currently, the winding system is regulated through a temporization. This system is not optimal and the team is working on building a standalone electronic-card for the Bambi-Bucket, matched with ultrasound sensors for the control to be the more autonomous as feasible. These sensors are aimed to measure the altitude of the MAV above the water. This system should be ready by the end of the summer.

5 Conclusion

In this paper, we have tried to synthesize the work done by the teams presented in the previous pages. The main goal of the UAV is the innovative shape, the Bambi-Bucket and the LabVIEW interface. The optimal design of such structures in order to fit given specifications is not easy as mechanical as well as electronical parts have to be optimized together, and this remains a topic of research [5].

After all the tests and the reasoning, the team UAVs are now ready to fly and will still be improved until October.

6 References

1. *Rotary Wing Micro Air Vehicle Endurance*. **Neitzke, Klaus-Peter**. Toulouse : s.n., 2013.
2. *Development of a quadrotor based on a innovative composite Shape with carbon and kevlard structure.* **Baptiste Delannoy, Benoit Wach, Renaud Kiefer, Achraf Haouet, Maxime collette and Vincent Person.** Toulouse : s.n., 2013. IMAV 2013, technical session.
3. *Future aircraft strusctures: from metal to composite structures*. **A.Beukers, H.Bersee, Otto Bergsma, S.Koussios and Michel van Tooren** s.l. : Springer Verlag london, 2010. DOI: 10.1002/9780470686652.eae194.
4. *Proposition of design method and optimization of multirotor composite structures*, **Renaud Kiefer, Marc Vedrines**, Aachen Germany, 2015
5. *Systematic design metohodology and construction of UAV helicopters.* **Guowei Cai, Lin feng, Ben M.Chen, Tong H.Lee.** 18, s.l. : Elesevier, 2008, Mechatronics, pp. 545-558.

http://www.imavs.org/

Deviation Correction and Stability Augmentation Control for UAV Taxiing

Li Zhen, Guo Jie, Chen Tianyue, Liu Zhenchang
Beijing Institute of Technology, Zhongguancun sorth street No.5, Beijing

http://www.imavs.org/

ABSTRACT

For the taxiing deviation correction control of medium and small sized UAV and the taxiing rollover problem which is likely to happen during deviation correction, this paper analyzed the force acting on UAV during taxiing, established the six degrees of freedom mathematical model of UAV taxiing, including the landing gear absorber model and the tire model. The Fuzzy-PID (proportion integration differentiation) control system based on ACO (ant colony algorithm) was designed to optimize PID parameters offline and to adjust PID parameters online. The cause of UAV rollover was analyzed, the fuzzy stability augmentation control system was presented to enhance the taxiing stability of UAV by synergetic controlling nose wheel, elevator and aileron. The simulation results show that the Fuzzy-PID control system based on ACO has obvious effect on deviation correction, and the control system can enhance the taxiing stability of UAV and keep UAV from rolling over effectively. the feasibility and validity of the control system was verified.

1 INTRODUCTION

In recent years, the flight performance of UAV is taken seriously in the development of UAV[1]. Although taking off and landing only take 2%-3% of the whole mission time, there are a lot of accidents happened during this time[2]. As for medium and small sized UAV, whose mass is light and tires are small, usually, the runways they used are simple, interference factors such as gust and unfairness of runway will make strong disturbance to UAV during taxiing, the disturbance will change the yaw angle of UAV and make lateral deviation[3]. If the deviation couldn't be corrected effectively, the UAV may taxi out of runway. If the deviation is corrected sharply, the UAV is easy to roll over[4]. So it can be concluded that the deviation correction and stability augmentation control is an important mission during taking off and landing.

In terms of UAV taxiing modeling, GU Hong-bin[5] and ZHANG Ming[6] established the taxiing model of UAV which used the rigid tricycle landing gear. F. Holzapfel[7] presented the accurate landing gear model. Kapseong Ro[8]、W.A.Ragsdale[9]and Xiaohui Wei[10] established taxiing model of UAV with stretch landing gear. Phil Evans[11] investigated the UAV taxiing performance while the landing gear break down. But some of them used simplified tire theory to solve the lateral force of tire, when the sideslip angle of tire was large, that will make the large error. And few of them considered the stretch of landing gear will change the attitude of UAV, which will influence the taxiing stability of UAV.

In terms of UAV taxiing control, Song Lei[3] investigated the instability problem which was observed during high-speed taxi tests of an experimental flying-wing aircraft. Am Cho[12] designed the deviation correction control system with linear quadratic regulator. WANG Yong[13] presented the deviation correction laws based on genetic algorithm. WU Cheng-fu[14]and YUAN Zhao-hui[15] used fuzzy logic to correct the taxiing deviation. JI Li-li[16] designed the deviation correction control laws with traditional PID control. But for the taxiing control of medium and small sized UAV, the control system above considered the deviation correction performance only, few of them considers the stability of UAV during taxiing, in fact, the UAV may roll over when the deviation is corrected sharply.

To solve the taxiing rollover problem which is likely to happen during deviation correction, this paper presents a deviation correction and stability augmentation control system. The UAV taxiing mathematical model is established, which considered the landing gear absorber model and the accurate tire model. The cause of rollover was analyzed. The fuzzy controller was designed to enhance the stability of UAV by synergetic controlling nose tire, elevator and aileron. The Fuzzy-PID control system based on ant colony algorithm (ACO) was designed to optimize parameters offline and adjust parameters online, it enhanced the performance of deviation correction. Finally, through simulation and taxiing experiments, the feasibility and validity of the control system is verified.

2 UAV TAXIING MODEL

A small UAV[17] is investigated in this paper, the UAV is adopted with tandem wing configuration, tricycle landing gear and steerable nose wheel, the control surface are elevator and aileron. When the UAV taxis with three wheels on the ground, there are gravity, thrust, aerodynamic force and ground reaction force acting on UAV. In this paper, the ground reaction force on landing gear is selective analyzed, the air oil absorber model and tire model are built.

2.1 Landing gear absorber model

The air oil absorber is researched in this paper, in the axial direction of landing gear, the air spring force F_{elastic} and the oil damping force F_{damp} are considered only. The axial force of landing gear which acts on UAV can be expressed

by

$$P = F_{\text{elastic}} + F_{\text{damp}} \tag{1}$$

where

$$F_{\text{elastic}} = \frac{P_0 V_0 S_g}{V_0 - S_g \delta} \tag{2}$$

$$F_{\text{damp}} = \frac{K_o \rho_o S_o^{\ 3} \delta'^2}{2 S_d^{\ 2}} \tag{3}$$

In the equation, P_0 and V_0 are initial pressure and initial volume of the air cavity respectively. δ is the compression stroke of the absorber. S_g, S_o and S_d are piston area of air cavity, piston area of oil cavity and damping hole area respectively. K_o and ρ_o are oil damping coefficient and oil density. Compression stroke δ and compression velocity δ' can be got from UAV attitude and attitude rate:

$$\begin{cases} \delta_a = L_a - a\sin\vartheta - h \\ \delta_{bl} = L_{bl} + b\sin\vartheta - b_w \sin\gamma - h \\ \delta_{br} = L_{br} + b\sin\vartheta + b_w \sin\gamma - h \end{cases} \tag{4}$$

$$\begin{cases} \delta'_a = V_z - a\omega_z \\ \delta'_{bl} = V_z + b\omega_z - b_w\omega_x \\ \delta'_{br} = V_z + b\omega_z + b_w\omega_x \end{cases} \tag{5}$$

In the equation, δ_a, δ_{bl} and δ_{br} are compression stroke of front landing gear, left rear (LR) landing gear and right rear (RR) landing gear respectively. δ'_a, δ'_{bl} and δ'_{br} are compression velocity of front landing gear, LR landing gear and RR landing gear respectively. L_a, L_{bl} and L_{br} are uncompressed length of front landing gear, LR landing gear and RR landing gear respectively. a is lengthways projection distance between center of gravity and front landing gear. b is lengthways projection distance between center of gravity and rear landing gear. b_w is lateral projection distance between center of gravity and rear landing gear. ϑ, γ and ψ are pitch angle, roll angle and yaw angle of UAV. ω_x and ω_z are pitch rate and roll rate of UAV. h is height between ground and central of gravity. V_z is the velocity component on z axes in ground coordinate system.

2.2 Tire model

As is shown in figure 1, there are three force acting on the tire, they are support force P, rolling friction force of tire N and lateral force of tire S. The support force P can be solved through analyzing axial force of landing gear. The rolling friction coefficient between ground and tire is μ_r, so the rolling friction force of three tires can be expressed by:

$$\begin{cases} N_a = \mu_r P_a \\ N_{bl} = \mu_r P_{bl} \\ N_{br} = \mu_r P_{br} \end{cases} \tag{6}$$

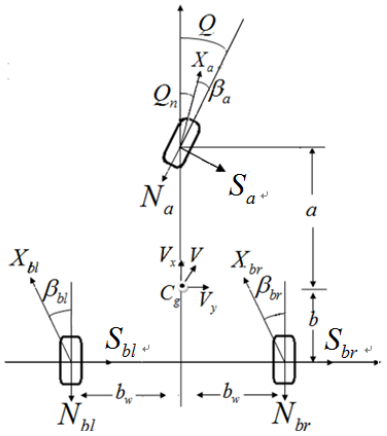


Figure 1: Ground force acting on tires

Then the lateral force of tire is analyzed emphatically: UAV needs to turn the front wheel to control the yaw angle, and there is friction force between ground and tire, which is perpendicular to the rotation surface of tire, this force is lateral force of tire. The angle between rotation force and velocity direction of tire is slip angle of tire. There are two stages while calculating the lateral force of tire: the elastic deformation stage and the slip stage. The critical slip angle is defined to distinguish the two stages.

The critical slip angle β_c is determined by support force P, tire cornering stiffness k_β and the rolling friction coefficient μ_r. The critical slip angle can be expressed by:

$$\beta_c = \frac{3\mu_s |P|}{k_\beta} \tag{7}$$

If the slip angle of tire is equal or less than the critical slip angle ($|\beta| \le \beta_c$), the lateral force of tire can be expressed by:

$$S = -\mu_s |P|(1 - B^3) sign(B) \tag{8}$$

$$B = 1 - \frac{\beta_c |\tan(\beta)|}{3\mu_s |P|} \tag{9}$$

If the slip angle of tire is equal or less than the critical slip angle ($|\beta| > \beta_c$), the lateral force of tire can be expressed by:

$$S = -\mu_s |P| \text{sign}(\beta) \tag{10}$$

Where, β can be solved from figure 1:

$$\begin{cases} \beta_a = Q - \arctan\left(\dfrac{V_y + ra}{V_x \cos\theta}\right) \\ \beta_{bl} = \arctan\left(\dfrac{rb - V_y}{rb_w + V_x \cos\theta}\right) \\ \beta_{br} = \arctan\left(\dfrac{rb - V_y}{V_x \cos\theta - rb_w}\right) \end{cases} \tag{11}$$

http://www.imavs.org/

In the equation, β_a, β_{bl} and β_{br} are slip angle of front tire, LR tire and RR tire. V_x and V_y are velocity component on X axes and Y axes in body coordinate system.

The force and moment acting on UAV from ground can be solved from equation (12)and (13). In the equation, $\boldsymbol{T}$ is transfer matrix from stable coordinate system to body coordinate system, Q is deflection angle of front wheel. Combine ground force and aerodynamic force together, the taxiing model of UAV can be got.

$$\boldsymbol{F_{be}} = \begin{bmatrix} F_{bex} \\ F_{bey} \\ F_{bez} \end{bmatrix} = \boldsymbol{T} \begin{bmatrix} -(S_a \sin Q + N_a \cos Q + N_{bl} + N_{br}) \\ S_a \cos Q - N_a \sin Q + S_{bl} + S_{br} \\ -(P_a + P_{bl} + P_{br}) \end{bmatrix} \tag{12}$$

$$\boldsymbol{M_{be}} = \begin{bmatrix} M_{bex} \\ M_{bey} \\ M_{bez} \end{bmatrix} = \boldsymbol{T} \begin{bmatrix} (P_{bl} + P_{br})b_w - \\ P_a a - (P_{bl} + P_{br})b - \\ (S_a \cos Q - N_a \sin Q)a - \\ (S_a \cos Q - N_a \sin Q + S_{bl} + S_{br})h \\ (S_a \sin Q + N_a \cos Q + N_{bl} + N_{br})h \\ (S_{bl} + S_{br})b \end{bmatrix} \tag{13}$$

3 DEVIATION CORRECTION AND STABILITY AUGMENTATION CONTROL

3.1 Instability analysis of UAV taxiing

From figure1, the force acting on UAV can be got, when the UAV turns right, these force will generate the negative pitching moment and the left rolling moment (figure 2), there will be a roll-over tendency for UAV. Because the absorber could be compressed, the UAV will present the negative pitching and left rolling attitude. In this attitude, the aerodynamic lift and engine thrust will increase the roll-over tendency of UAV. So if turning angle of front wheel is too much, the UAV is easy to roll over.

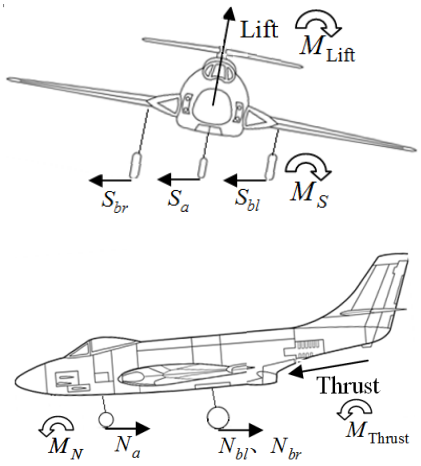


Figure 2: Analysis of rollover force

The rollover accident is simulated in this paper, when the compression stroke of absorber decreases to zero, it can be considered that there is no support force between ground and tire, the rollover accident happened. In simulation, the turning angle of front wheel increases 1° starting from 0°, until the UAV roll over, the critical turning angle $Q_{critical}$ can be gotten. The simulation above is done in different taxiing velocity, and then the critical rollover condition can be got, it's shown in figure 3. It can be found the lower the gravity center is, the higher the stability is. It also can be found if the taxiing velocity $V > 10m/s$, the critical turning angle of front wheel is small, the UAV is easy to roll over due to the large turning angle of front wheel, so it can be considered the stability of this UAV is poor.

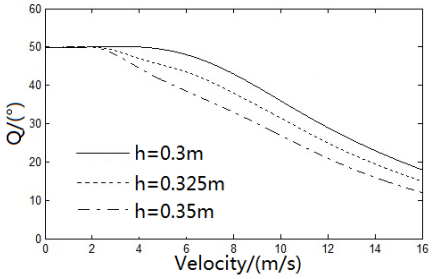


Figure 3: Critical condition of taxiing rollover

3.2 Stability Augmentation Control

In order to enhance the UAV taxiing stability and avoid roll-over accident, the fuzzy stability augmentation control system is designed. Through deflecting the elevator and aileron, the aerodynamic moment is generated to balance the rollover moment, the UAV attitude could keep steady. The fuzzy logic rules are used to control the turning of front wheel and the deflection of elevator and aileron, the rules are shown below:

1) When the taxiing velocity of UAV is low, and the turning angle of front wheel is small, there is no rollover risk, the elevator positive deflects through large angle, the negative pitching moment is generated to increase the pressure between the ground and the front tire, that can improve the turning efficiency of UAV. The aileron deflects through small angle, that can decrease the air drag, the UAV could reach the taking-off velocity as soon as possible.

2) When the taxiing velocity of UAV is low, and the turning angle of front wheel is large, there is small rollover risk, the elevator negative deflects through small angle, the aileron deflects through small angle.

3) When the taxiing velocity of UAV is high, and the turning angle of front wheel is small, there is small rollover risk, the elevator positive deflects through small angle, the aileron deflects through large angle, the large drag is generated to decrease the lateral velocity of UAV, the large rolling moment is generated to balance the rollover moment.

4) When the taxiing velocity of UAV is high, and the turning angle of front wheel is large, there is large rollover risk, the elevator negative deflects through large angle, the aileron deflects through large angle, the resultant moment is

http://www.imavs.org/

http://www.imavs.org/

generated to prevent the UAV from rolling over.

The fuzzy logic rules of elevator deflection and aileron deflection are shown in table 1. The fuzzy language is NB (Negative big), NM (negative medial), NS (negative small), ZO (zero), PS (positive small), PM (positive medial), PB (positive big). Membership function is triangle and normal type, the position and coverage area of fuzzy subset are adjusted to get the membership function with good convergence property and rapid respond. the membership function of elevator deflection is shown as figure 4.

V	Q						
	NB	NM	NS	ZO	PS	PM	PB
ZO	PS/NS	PM/NS	PM/ZO	PB/ZO	PM/ZO	PM/PS	PS/PS
PS	NS/NM	NS/NS	PS/NS	PB/ZO	PS/PS	NS/PS	NS/PM
PM	NB/NB	NM/NM	ZO/NS	PS/ZO	ZO/PS	NM/PM	NB/PB
PB	NB/NB	NB/NB	ZO/NM	PS/ZO	ZO/PM	NB/PB	NB/PB

Table 1: Fuzzy control rules of elevator/aileron

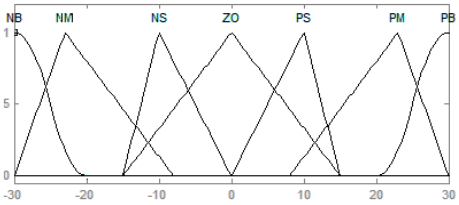


Figure 4: Membership function of elevator deflection

According the fuzzy rules in table 1, the centroid method is used to solve ambiguity, the fuzzy logic surface of elevator and aileron are shown as figure 5 and figure 6.

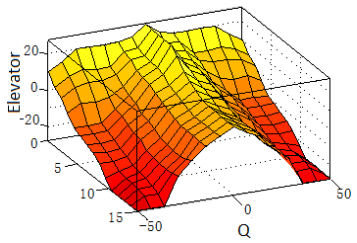


Figure 5: Fuzzy control rules of elevator

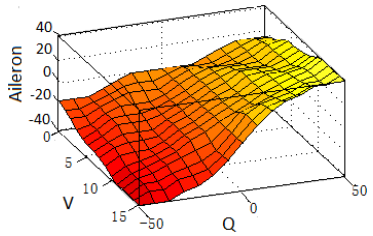


Figure. 6: Fuzzy control rules of aileron

3.3 *Deviation correction control*

The UAV correct deviation through controlling the turning of front wheel. The Fuzzy-PID control system based on ant colony algorithm (ACO) is designed to control the deviation correction during taxiing, the control structure is shown as figure 7, the ACO is used to optimize the PID parameters offline, and the fuzzy logic control is used to adjust PID parameters online.

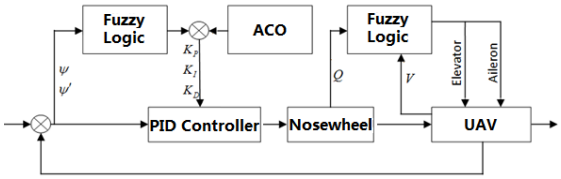


Figure. 7: Deviation correction and stability augmentation control system

In the control system, ACO is used to optimize the K_P, K_I, K_D three gain parameters, the objective function is:

$$J_{\mathrm{PID}} = \int_0^{\infty} \left(k_1 \left| e(t) \right| + k_2 u(t) \right) dt + k_3 t_d + k_4 \sigma_p \qquad (14)$$

In the equation, $e(t), u(t), t_d, \sigma_p$ are system error, control quantity, rise time and overshot respectively, k_1, k_2, k_3, k_4 are weight.

It assumes there are m ants, K_P , K_I , K_D are three sets, the ants need to pass three nodes in K_P , K_I and K_D in order, and according to the transition probability, the ants random select the next point. The transition probability can be solved by:

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{\left[\tau_{ij}(t)\right]^{\varsigma}\left[\eta_{ij}(t)\right]^{\tau}}{\sum_{s \in allowed(t)^k}\left[\tau_{is}(t)\right]^{\varsigma}\left[\eta_{is}(t)\right]^{\tau}} & , \ j \in allowed(t)^k \\ 0 & , \ j \notin allowed(t)^k \end{cases} \qquad (15)$$

In the equation, $P_{ij}^{k}(t)$ is the transition probability of nth ant at t time; $\tau_{ij}(t)$ and $\eta_{ij}(t)$ are the pheromone content and visibility between two neighbor nodes; ς and τ are the relative importance of the route and the visibility; $allowed(t)^k$ is the node set that can be selected by nth ant at the t time. At the t time, the visibility of nodes can be expressed by:

$$\eta_{ij}(t) = 1 / d_j(t) \qquad (16)$$

$d_j(t)$ is the distance between the current node and the optimal node in last round. When all of m ants finish once circulation, the pheromone between two neighbor nodes will be updated, the update rules are shown as below:

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \qquad (17)$$

$$\Delta\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \qquad (18)$$

$$\Delta\tau_{ij}^{\,k}(t)=\begin{cases}\dfrac{Q_{\text{ant}}}{J_{\text{PID}}} & ,\ if\ the\ ants\ pass\ the\ node\\ 0 & ,\ else\end{cases} \tag{19}$$

In the equation, Q_{ant} is the pheromone intensity of the ant. After N_c times iteration, the optimal solution of objective function and the optimal parameters of PID can be gotten. In this control system, when the initial velocity is $5m/s$ and $10m/s$, the initial yaw angle error is 5.7° and 11.5°, the parameters optimization results is shown as table2.

Parameters	$\psi=5.7^\circ$ $V=5m/s$	$\psi=5.7^\circ$ $V=10m/s$	$\psi=11.5^\circ$ $V=5m/s$	$\psi=11.5^\circ$ $V=10m/s$
K_P	3.00	3.00	3.00	3.00
K_I	0.12	0.13	0.02	0.10
K_D	0.61	0.75	0.72	0.84

Table 2: Parameter optimization results with ACO

The Fuzzy-PID control system is consist of fuzzy logic controller and PID controller. In the fuzzy logic controller, the yaw angle error $|\psi|$ and error rate $|\psi'|$ are input into controller, the output are the increment ΔK_P, ΔK_I and ΔK_D, as is shown in formula (20), through adjusting the parameters $K_{P\text{fuzzy}}$, $K_{I\text{fuzzy}}$ and $K_{D\text{fuzzy}}$ online, the PID parameters fuzzy self-tuning system is generated.

$$\begin{cases}K_{P\text{fuzzy}}=K_P+\Delta K_P\\ K_{I\text{fuzzy}}=K_I+\Delta K_I\\ K_{D\text{fuzzy}}=K_D+\Delta K_D\end{cases} \tag{20}$$

PID parameters fuzzy logic rules are shown below:

1) When $|\psi|$ is large, in order to speed up the response of system, the large K_P and small K_D, K_I should be used.

2) When the $|\psi|$ is medium, in order to decrease the overshoot of system and keep the rapid response speed, K_P should be decreased and the medium K_D and K_I should be used.

3) When the $|\psi|$ is small, in order to decrease the state error of system, the large K_P and K_I should be used. Considering the anti-disturbance performance of system, if the $|\psi'|$ is large, the small K_D should be used, else the K_D should be medium.

The fuzzy logic rules of PID parameters are shown in table 3, according the fuzzy logic rules, the PID parameters fuzzy self-tuning controller is designed.

4 SIMULATION

In order to verify the deviation correction property and the stability augmentation performance of the control system, the UAV taxiing is simulated under the initial condition $V=10m/s$ and $\psi=11.5^\circ$. The simulation results are shown in figure8-figure11.

The roll angle of UAV is recorded during the process of correction deviation, the result is shown in figure.8, it could be found that after using the fuzzy stability augmentation control, the roll angle change of UAV is smaller, and the attitude of UAV is steadier; The critical rollover condition of UAV which use the fuzzy stability augmentation control is analyzed. From figure 9 it could be found that after using the fuzzy stability augmentation control, the critical turning angle of front wheel becomes larger, that means the UAV could correct deviation with a larger range; The taxiing track and yaw angle of UAV are recorded during the process of correction deviation, from figure.10 and figure.11, it could be found that compared with the UAV which uses fixed parameters PID controller, the UAV which uses the ACO-Fuzzy control system has better effect on the deviation correction response, the overshoot, and the steady time, the deviation distance is smaller.

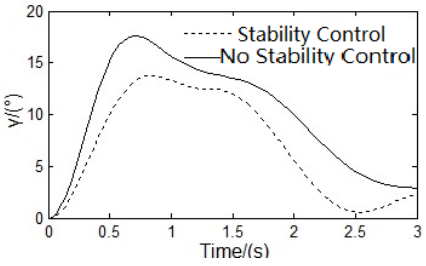


Figure. 8: Change of roll angle during deviation correction

Table 3 Fuzzy logic rules of $\Delta K_P/\Delta K_I/\Delta K_D$

ψ	ψ'						
	NB	NM	NS	ZO	PS	PM	PB
NB	PB/NB/PS	PB/NB/NS	PM/NM/NB	PM/NM/NB	PS/NS/NB	ZO/ZO/NM	ZO/ZO/PS
NM	PB/NB/PS	PB/NB/NS	PM/NM/NB	PS/NM/NM	PS/NS/NM	ZO/ZO/NS	NS/ZO/ZO
NS	PM/NB/ZO	PM/NM/NS	PM/NS/NM	PS/NS/NM	ZO/ZO/NS	NS/PS/NS	NS/PS/ZO
ZO	PM/NM/ZO	PM/NM/NS	PS/NS/NS	ZO/ZO/NS	NS/PS/NS	NM/PM/NS	NM/PM/ZO
PS	PS/NM/PB	PS/NS/ZO	ZO/ZO/ZO	NS/PS/ZO	NS/PS/ZO	NM/PM/ZO	NM/PB/ZO
PM	PS/ZO/PB	ZO/ZO/NS	NS/PS/PS	NM/PS/PS	NM/PM/PS	NM/PB/PS	NB/PB/PB
PB	ZO/ZO/PB	ZO/ZO/PM	NM/PS/PM	NM/PM/PM	NM/PM/PS	NB/PB/PS	NB/PB/PB

http://www.imavs.org/

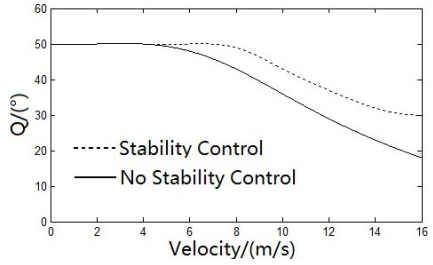


Figure 9: Critical condition of taxiing rollover

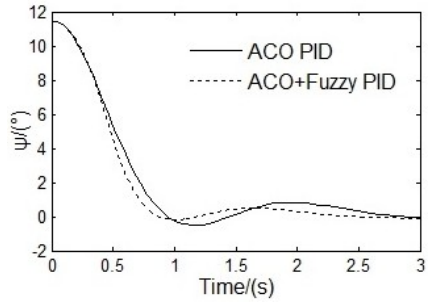


Figure 10: Change of yaw angle during deviation correction

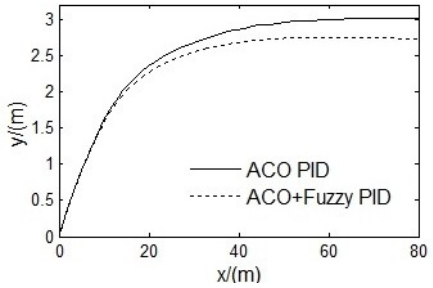


Figure 11: Taxiing track during deviation correction

5 CONCLUSION

Through establishing the mathematical model of UAV taxiing, the process of UAV rollover is simulated, the critical rollover condition is gotten, the results shows the stability of UAV is poor during deviation correction. The fuzzy stability augmentation controller and the fuzzy-PID controller based on ACO is designed, the simulation results show that the Fuzzy-PID control system based on ACO has obvious effect on deviation correction, and the control system can enhance the taxiing stability of UAV and keep UAV from rolling over effectively. The simulation results validate the feasibility and validity of control system.

REFERENCES

[1] Vivek A, Roy J. H. Optimization of UAV Designs for Aerodynamic Performance using Genetic Algorithms 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference. Orlando: AIAA, 2010: 12-15.

[2] HUO Z Q. Key Issue Research on Runway Safety Management of China Civil Aviation Based on Historical Data. Nanjing: Nanjing University of Aeronautics and Astronautics, 2012.

[3] SONG L, YANG H, YAN X F et al. A study of instability in a miniature flying-wing aircraft in high-speed taxi. Chinese Journal of Aeronautics, 2015, 28(3): 749-756.

[4] ZHOU Y Z, SHU P. Analysis on Prevention of Runway Overrun /Excursion Accident during Takeoff. China Safety Science Journal, 2009, 19(1): 38-44.

[5] GU H B. Dynamic Model of Aircraft Ground Handing. Acta Aeronautica et Astro nautica Sinica, 2001, 22(2): 163-167.

[6] ZHANG M, NIE H. Dynamics Analysis of Aircraft Ground Steering and Breaking Responses[J]. Acta Aeronautica et Astro nautica Sinica, 2008, 29(3): 617-621.

[7] F Holzapfel. High Fidelity Landing Gear Modeling for Real-Time Simulation. AIAA Modeling and Simulation Technologies Conference and Exhibit. Keystone: AIAA, 2006: 21-24.

[8] Kapseong R. A Descriptive Modeling and Simulation of Aircraft-Runway Dynamics. AIAA Modeling and Simulation Technologies Conference and Exhibit. Norfolk: AIAA, 2003: 7-10.

[9] W A Ragsdale. A Generic Landing Gear Dynamics Model for Lasrs++. AIAA Modeling and Simulation Technologies Conference and Exhibit. Denver: AIAA, 2000: 14-17.

[10] WEI X H, LIU C L, HONG N et al. Improved Model of Landing-Gear Drop Dynamic. Journal of Aircraft, 2014, 5(22): 695-700.

[11] Phil E, Mario G. P, Steven M et al. Modeling and Simulation of a Tricycle Landing Gear at Normal and Abnormal Conditions .AIAA Modeling and Simulation Technologies Conference. Toronto: AIAA, 2010: 2-5.

[12] Am C, Jihoon K, Sanghyo L et al. Fully Automatic Taxiing, Takeoff and Landing of a UAV only with a Single-Antenna GPS Receiver. AIAA 2007 Conference and Exhibit. Rohnert Park: AIAA, 2007: 7-10.

[13] WANG Y, WANG Y X. Lateral Deviation Correction Control for UAV Taxiing. Acta Aeronautica et Astro nautica Sinica, 2008, 29(Sup): 142-149.

[14] YUAN Z H, WANG Y, YANG F. Application of New Fuzzy Control in Airplane's Integrated Ground Direction Control System. Computer Simulation, 2011, 28(2): 76-79.

[15] WU C F, YAN B, SHAO P Y. Exploring Taxiing Take-off Control for Unmanned Aerial Vehi-

http://www.imavs.org/

cle(UAV)Based on Fuzzy Control. Journal of Northwestern Polytechnical University, 2015, 33(1): 33-39.

[16] JI L L. Research on Control System Design of Auto Landing for Unmanned Aerial Vehicle. Nanjing: Nanjing University of Aeronautics and Astronautics, 2012.

[17] TANG S J, LIU Z C, ZHOU Y Q. Investigation on Aerodynamic Performance of Tandem Wing UAV with Rare Wing Dihedral [J]. Transactions of Beijing Institute of Technology, 2015, 35(12): 1-7.

http://www.imavs.org/

Distance and velocity estimation using optical flow from a monocular camera

H.W. Ho*, G.C.H.E. de Croon, and Q.P. Chu
Delft University of Technology, The Netherlands

http://www.imavs.org/

ABSTRACT

Monocular vision is increasingly used in Micro Air Vehicles for navigation. In particular, optical flow, inspired by flying insects, is used to perceive vehicles' movement with respect to the surroundings or sense changes in the environment. However, optical flow does not directly provide us the distance to an object or velocity, but the ratio of them. Thus, using optical flow in control involves nonlinearity problems which add complexity to the controller. To deal with that, we propose an algorithm to use an extended Kalman filter to estimate the distance and velocity of the vehicles from optical flow while approaching a surface, and then use these estimates for control. We implement and test our algorithm in computer simulation and on-board a Parrot AR.Drone 2.0 to demonstrate its feasibility for MAVs landings. Both results show that the algorithm is able to estimate height and velocity of the MAV accurately.

1 INTRODUCTION

Micro air vehicles (MAVs) have been widely used due to its small size and enormous capabilities. These two advantages are also two of the biggest challenges to the MAVs as the amount of payloads can be carried on-board and the computing power are limited [1]. Therefore, light-weight sensors which can provide rich information about the surrounding of the MAVs are highly desired. In addition, effective and efficient algorithms are needed to estimate useful inputs to the MAVs from the sensory information [2].

In fact, flying insects have the capabilities to perform complex tasks by only using their bare eyes and a tiny brain. For instance, honeybees heavily use optical flow to perceive the environment and avoid dangerous objects while flying [3, 4]. Optical flow refers to visual relative motion of the observer's eyes or a camera with respect to the objects in the scene [5]. This information tells us not only how fast the camera moves, but also how close it is relatives to the things it sees. However, optical flow does not give us the absolute distance to a surface or velocity of the camera directly.

For a particular case when an MAV is approaching the ground during landing, a downward-facing camera on-board the MAV perceives divergent pattern of the optical flow, or so-called flow divergence. Flow divergence is the ratio of velocity to distance to a surface in the same direction and it is the reciprocal to time-to-contact. By only keeping the flow divergence constant, the MAV is able to perform smooth landing using a monocular camera [6, 7, 8, 9]. In fact, using directly the flow divergence in control involves the problem of nonlinearity in the controller which is complex, and is sensitive when the MAV is close to the ground [10, 11].

The common state estimation techniques using a visual-inertial system in MAVs combined the advantages of using an Inertial Measurement Unit (IMU) and a camera sensor for real-time computation [12, 13, 14]. Although these sensors are usually installed on-board the MAVs, it is definitely an added advantage for MAVs if only a monocular camera is used for navigation. Some recent approaches used 'efference copies' or available control input information fused with flow divergence, to estimate distance in a constant flow divergence control [15, 16]. In [15], they used the flow divergence and the control input that moved a camera along a track toward an image scene to first estimate the initial distance. Since the constant flow divergence control will result in exponential decay of distance, they simplified the estimation problem to exponential propagation of distance after obtaining the initial value. In [16], a stability approach that detects self-induced oscillation caused by high controller gains and uses these gains to estimate distance was introduced.

The main contribution of this paper is to estimate height (distance to ground) and vertical velocity of an MAV using a monocular camera while approaching a surface, e.g. in the landing phase. The difference from the above-mentioned approaches is that we propose a straightforward algorithm that uses an extended Kalman filter (EKF) to 'split' the height and vertical velocity of an MAV from flow divergence. The significant advantage of this method is that we simplify the nonlinear control of the constant flow divergence to a linear control that uses height and velocity estimates. Thus, this provides the possibility of using some optimization techniques with vision output from a monocular camera to improve the performance of the control.

The remainder of the paper is set up as follows: In Section 2, we provide some knowledge about flow divergence and constant flow divergence guidance and control strategy. Section 3 describes the proposed EKF-based height and velocity estimation using flow divergence and vision algorithms used to estimate flow divergence with a monocular camera.

*Email address(es): h.w.ho@tudelft.nl

http://www.imavs.org/

Then, Section 4 shows the results of estimation in computer simulation while in Section 5 we also show the estimation results and landing using the estimates in flight. Finally, a conclusion with future works is drawn in Section 6.

2 BACKGROUND

2.1 Flow Divergence

For vertical landings of an MAV, flow divergence (D) or visual looming as shown in Fig. 1 can be computed as the ratio of its vertical velocity V_Z to height from the ground Z:

$$D = \frac{V_Z}{Z}. \tag{1}$$

When the MAV is approaching the ground, we measure positive Z, negative V_Z, and thus $D < 0$ according to the coordinate systems shown in Fig. 2. Camera coordinate system is assumed to be coincided with the body coordinate system.

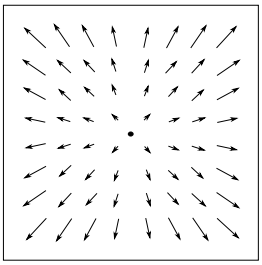


Figure 1: Divergence of optical flow vectors (flow divergence) when the observer is approaching a surface.

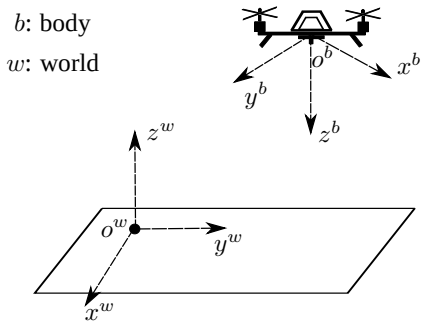


Figure 2: MAV body (x^b, y^b, z^b) and world (x^w, y^w, z^w) coordinate systems.

2.2 Constant flow divergence Guidance and Control

Constant flow divergence approach has been used for vertical landing of the MAVs. This approach controls the vertical dynamics of the MAVs by tracking a constant flow divergence. When D equals to a constant c, we can derive from Eq. 1 the height $Z = Z_0 e^{-ct}$, and velocity $V_Z = -cZ_0 e^{-ct}$, where Z_0 is the initial height. This shows that both height and velocity will decrease exponentially and eventually become zeros when touching the ground.

A proportional feedback controller is used to track the desired flow divergence D^* or c as shown in Eq. (2):

$$\mu = K_p(D^* - D), \tag{2}$$

where K_p is the gain of the proportional controller.

A double integrator system is used to model the dynamics of an MAV towards the ground in one dimensional space. The continuous state space model can be written as:

$$\begin{aligned}\dot{\mathbf{r}}(t) = f\big(\mathbf{r}(t), \mu(t)\big) &= \mathbf{A}\cdot\mathbf{r}(t) + \mathbf{B}\cdot\mu(t) \\ &= \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\mathbf{r}(t) + \begin{bmatrix}0\\ 1\end{bmatrix}\mu(t),\end{aligned} \tag{3}$$

$$y(t) = h\big(\mathbf{r}(t)\big) = [r_2(t)/r_1(t)] = D, \tag{4}$$

where $\mathbf{r} = [r_1, r_2]^T = [Z, V_Z]^T$ and μ is the control input.

It is clear that the model dynamics in Eq. (3) are linear but the observation in Eq. (4) is nonlinear. Fig. 3 shows a time response of the system when tracking a constant flow divergence. In this figure, we can see that the MAV accelerates in the first 2.5 s and then decelerates to zero velocity to touch the ground. Both height and velocity of the MAV descrease exponentially to zero in the end.

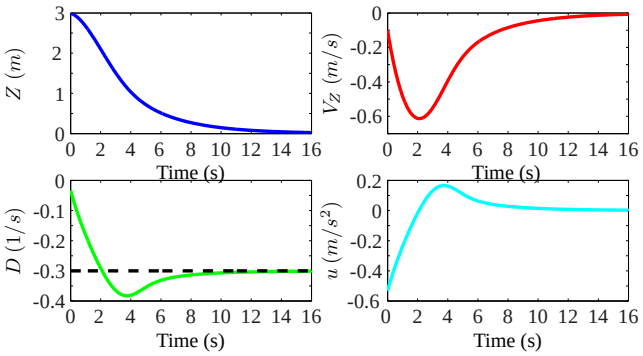


Figure 3: Height Z, velocity V_Z, and flow divergence D of an MAV during landing using constant flow divergence strategy ($D^* = -0.3\ s^{-1}$) with control input μ.

3 EKF-BASED HEIGHT AND VELOCITY ESTIMATION AND CONTROL USING FLOW DIVERGENCE

An overview of the methodology is presented in Fig. 4. Images captured from a camera are served as the inputs to the software while the control input commands the actuator to control the MAV. In this section, we describe: (1) the extended Kalman filter (EKF) algorithm using flow divergence and efference copies to estimate height and velocity of an MAV for landing control, (2) the vision method to compute flow divergence, and (3) the controller.

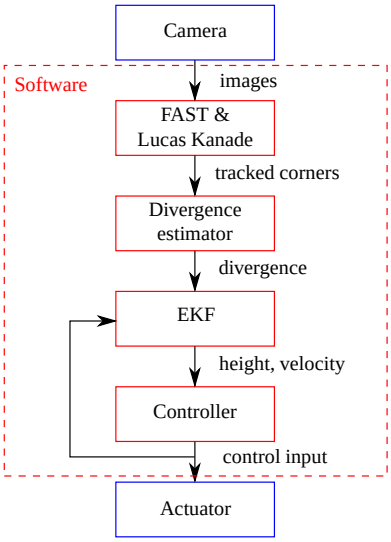


Figure 4: An overview of the methodology.

3.1 Extended Kalman Filter

In practice, the flow divergence is computed using an on-board processor. Therefore, we used a discrete-time extended Kalman filter to estimate the height and vertical velocity of the MAV. The system model and observation model are shown in Eq. 5 and 6, respectively.

$$\dot{\mathbf{x}}(t) = f\big(\mathbf{x}(t), \mu(t)\big) + \mathbf{w}(t). \tag{5}$$

$$\mathbf{z}_k = h\big(\mathbf{x}_k\big) + \mathbf{v}_k. \tag{6}$$

where $f(\cdot)$ and $h(\cdot)$ are the system matrix and the observation matrix which are obtained from Eq. 3 and 4. $\mathbf{w}(t)$ and $\mathbf{v}_k$ are the system noise and observation noise which are both assumed to be zero mean multivariate Gaussian process with covariance $\mathbf{Q}$ and $\mathbf{R}$, respectively.

Several computational steps taken in EKF when the update of flow divergence is obtained are shown below:
a) One-step ahead prediction:

$$\hat{\mathbf{x}}_{k|k-1} = \hat{\mathbf{x}}_{k-1|k-1} + \int_{t_{k-1}}^{t_k} f\big(\mathbf{x}(\tau), \mu(\tau)\big)\mathrm{d}\tau \tag{7}$$

b) Covariance matrix of the state prediction error vector:

$$\mathbf{P}_{k|k-1} = \mathbf{\Phi}_k \mathbf{P}_{k-1|k-1} \mathbf{\Phi}_k^T + \mathbf{\Gamma}_k Q \mathbf{\Gamma}_k^T \tag{8}$$

where $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ are discretized matrices of $\mathbf{A}$ and $\mathbf{B}$, and can be computed as below:

$$\mathbf{\Phi}_k = \begin{bmatrix} 1 & t_k - t_{k-1} \\ 0 & 1 \end{bmatrix}, \quad \mathbf{\Gamma}_k = \begin{bmatrix} (t_k - t_{k-1})^2/2 \\ t_k - t_{k-1} \end{bmatrix} \tag{9}$$

c) Kalman gain:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^\top \big(\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + R\big)^{-1} \tag{10}$$

where

$$\mathbf{H}_k = \frac{\partial h}{\partial \mathbf{x}}\bigg|_{\hat{\mathbf{x}}_{k|k-1}} = \big[-\frac{\hat{x}_{2_{k|k-1}}}{\hat{x}^2_{1_{k|k-1}}}, \frac{1}{\hat{x}_{1_{k|k-1}}} \big]^T \tag{11}$$

d) Measurement update:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\big(\mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1})\big) \tag{12}$$

where $\mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1})$ is called the innovation of EKF.
e) Covariance matrix of state estimation error vector:

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k R \mathbf{K}_k^T \tag{13}$$

3.2 Features-based flow divergence estimation

In computer simulation, we used Eq. 1 to compute flow divergence, while in flight tests, we estimate flow divergence using Eq. 14 (Proof can be found in [11].). For each image captured from an on-board camera, corners were detected using FAST algorithm [17, 18], and tracked in the next image using Lucas-Kanade tracker [19]. Then, the image distances between every two corners at one image, $d_{(t-\Delta t),i}$ and at the next image, $d_{t,i}$ were computed. By further computing the ratio of $d_{(t-\Delta t),i} - d_{t,i}$ to $d_{(t-\Delta t),i}$, we can measure the expansion and contraction of the flow. We take the arrange of these ratios and with known time interval between these two images Δt, we estimated divergence using Eq. 14:

$$\widehat{D} = \frac{1}{n} \cdot \frac{1}{\Delta t} \sum_{i=1}^{n} \big[\frac{d_{(t-\Delta t),i} - d_{t,i}}{d_{(t-\Delta t),i}}\big], \tag{14}$$

where n is the total number of tracked corners. In practice, the vision output is often noisy. Therefore, we used a low pass filter to reduce the noise in the estimation.

3.3 Controller

Consider the common way using flow divergence to control the vertical dynamics of an MAV as shown in Eq. 2, it involves nonlinearity in the observation ($D = V_Z/Z$). In MAVs landings, the closer the MAV to the ground, the more sensitive the flow divergence is. Therefore, a nonlinear controller or a gain-scheduling method with the knowledge of initial height is needed to deal with the problem.

By using the proposed EKF algorithm, we are able to 'splits' the flow divergence into height and velocity. These two states can then be used to land MAVs using a linear PI controller as shown in Eq. 15.

$$\mu_k = K_{Z_1}(\hat{x}_{1,t_k} - x^*_{1,t_k}) + K_{Z_2} \sum_{i=1}^{k} (\hat{x}_{1,t_i} - x^*_{1,t_i}) \cdot \Delta t. \tag{15}$$

where K_{Z_1} and K_{Z_2} are the PI controller gains for tracking the desired height x^*_1. A desired landing profile can be chosen by assigning the values of x^*_1. Note that after obtaining

http://www.imavs.org/

http://www.imavs.org/

both height and velocity, it is possible to improve the control performance using gain optimization or a linear robust controller.

4 COMPUTER SIMULATION

Before performing flight tests, we simulated the proposed algorithm presented in Subsection 3.1 in MATLAB to show the feasibility of the algorithm.

4.1 Landing simulation with simulated control inputs

In the simulation, we generated 3000 data (height and velocity) with a timestamp of $0.05\ s$ using a set of control inputs, μ as shown in Fig. 5e. These data were served as ground truth for validation. The flow divergence measurement was generated using Eq. 1 with measurement noise standard deviation of $0.001\ s^{-1}$ as illustrated in Fig. 5d.

By using the control input, μ and the flow divergence measurement, D, we estimated the height, Z and the velocity, V_Z with the proposed EKF algorithm. Fig. 5 shows estimated states (Z and V_Z) and their ground truth, innovation of the EKF, flow divergence measurement, and the control inputs. In this simulation, we can observe that the estimated height and velocity converge and follow the ground truth after a few seconds, even with different initial conditions (Z_0 and V_{Z_0}) than the actual values. In addition, the innovation of EKF has zero mean meaning that the filter is working correctly. Simulation results show that the proposed algorithm is able to estimate the distance to the ground and velocity accurately.

5 EXPERIMENTS AND RESULTS

We implemented the EKF algorithm and vision algorithms presented in Section 3 in Paparazzi Autopilot, an open source autopilot software [20]. A Parrot AR.Drone 2.0 equipped with a downward-looking camera was used as a testing platform and all algorithms were running on-board the MAV. We used an *OptiTrack* system to track the position of the MAV in order to provide the ground truth of its height and velocity *only for validation purposes* and these measurements were not used in the estimation. In this section, we show the feasibility of the algorithm by performing two different control strategies for MAVs landings.

5.1 Landing using constant flow divergence

The first strategy we used in the flight tests is the constant flow divergence. A flow divergence setpoint was tracked for the whole landing. Instead of using P controller as described in Section 2, we used a PI controller to compensate the steady state error of the tracking. During the landing, the EKF algorithm was running in real-time to estimate the height and velocity of the MAV.

Fig. 6 shows the estimation results of the landing with a constant divergence of $-0.3\ s^{-1}$. The initial value of the height for EKF was set to be $\hat{Z}_0 = 3m$ which was different than the true initial height, i.e. $Z_0 \approx 2m$. In the figure, we can see that the height estimate converges to the true height,

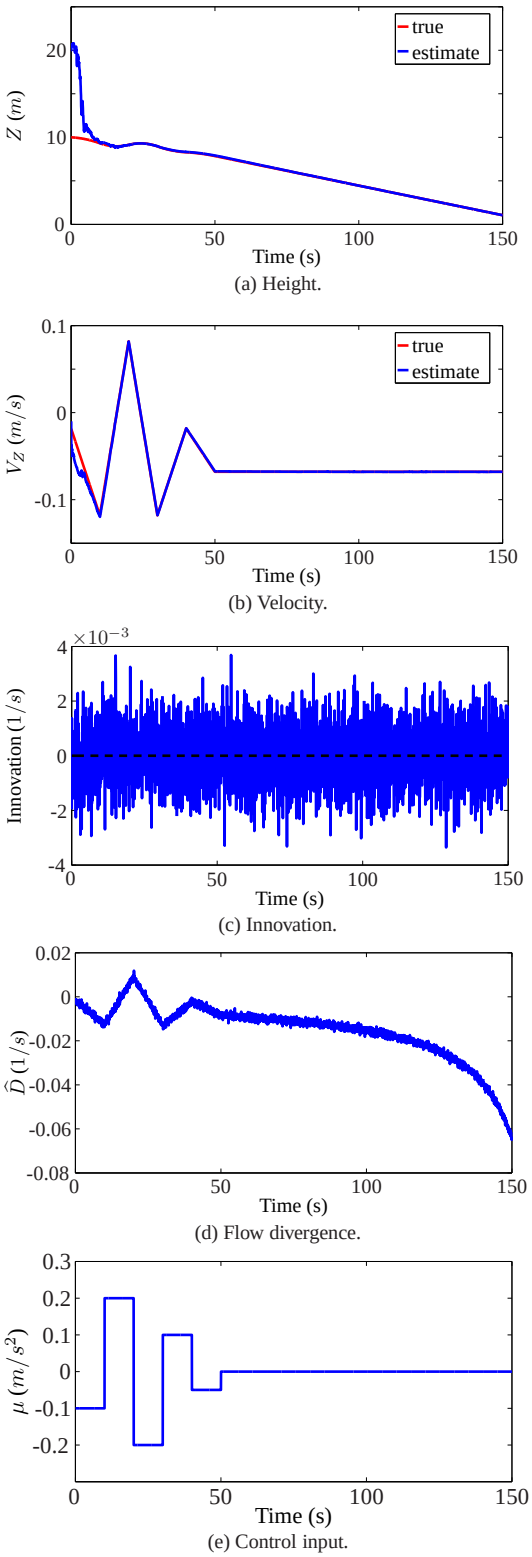


Figure 5: EKF-based height and vertical velocity estimation from flow divergence for landing control.

http://www.imavs.org/

even though the initial value is different. In addition, the velocity estimate also matches well with the true velocity.

From the landing results, we can observe that this strategy exponentially decreases both height and velocity to nearly zero by only tracking a constant flow divergence. In fact, the drawback of vision algorithms is that when the image is too dark, the algorithms will not work properly. For instance, when the camera is too close to the ground, the flow divergence estimate becomes incorrect. To deal with that, we constantly decrease the trim throttle when the MAV is very close to the ground ($Z < 0.2m$) to allow the MAV for touchdown. Another fact is that the flow divergence ($D = V_Z/Z$) will become sensitive when Z is small. A fine-tuned control gain or a gain-scheduling method adapted with height is then needed to solve the sensitivity issue which might cause instability to the MAV. This issue can be seen from the measured flow divergence in Fig. 6d when the MAV is close to the ground.

To show the reliability of the proposed algorithm, we performed multiple landings with different flow divergence setpoints ($D^* = -0.1\ s^{-1}, -0.2\ s^{-1}, -0.3\ s^{-1}$) at different initial heights ($Z_0 \approx 2m, 3m$). The same initial guess of the height ($3m$) was used in the EKF algorithm for these flight tests. Fig. 7 shows the height and velocity estimates using EKF algorithm in different landings. In this figure, we can see that both height and velocity estimates match well with the true values. With a larger setpoint, the MAV landed faster with the exponential decrease in both height and velocity. Besides, these results also show that a good guess of the initial height has a faster convergence of the estimates.

The height and velocity estimated from EKF can then be adapted to tune the controller gain in real-time for the constant divergence landing. By doing that, oscillations due to improper chosen gain can be reduced, and thus a smooth landing can be achieved by only using a monocular camera.

5.2 *Landing using height and velocity from EKF*

The second strategy we used for landing is tracking a height profile using the estimates from EKF. To this end, the landing begins by (1) giving an initial excitation to the MAV, or (2) using constant divergence strategy, for the first few seconds. This can ensure that the EKF converges and the controller tracks the right height estimate.

This first initialization was performed in an indoor flight test by giving a block input as the excitation as shown in Fig. 8e to the MAV for the first $0.5\ s$. This allowed the MAV to go down or move in order to 'observe' flow divergence and initialize the EKF algorithm. After that, the controller used the estimates from the EKF to track a desired profile or setpoints for landing.

Fig. 8 shows the height and velocity estimates from EKF compared with their ground truth, innovation of EKF, flow divergence from vision algorithms and from the ground truth, and the control input. After giving the initial excitation, the MAV started to track a height profile represented by the

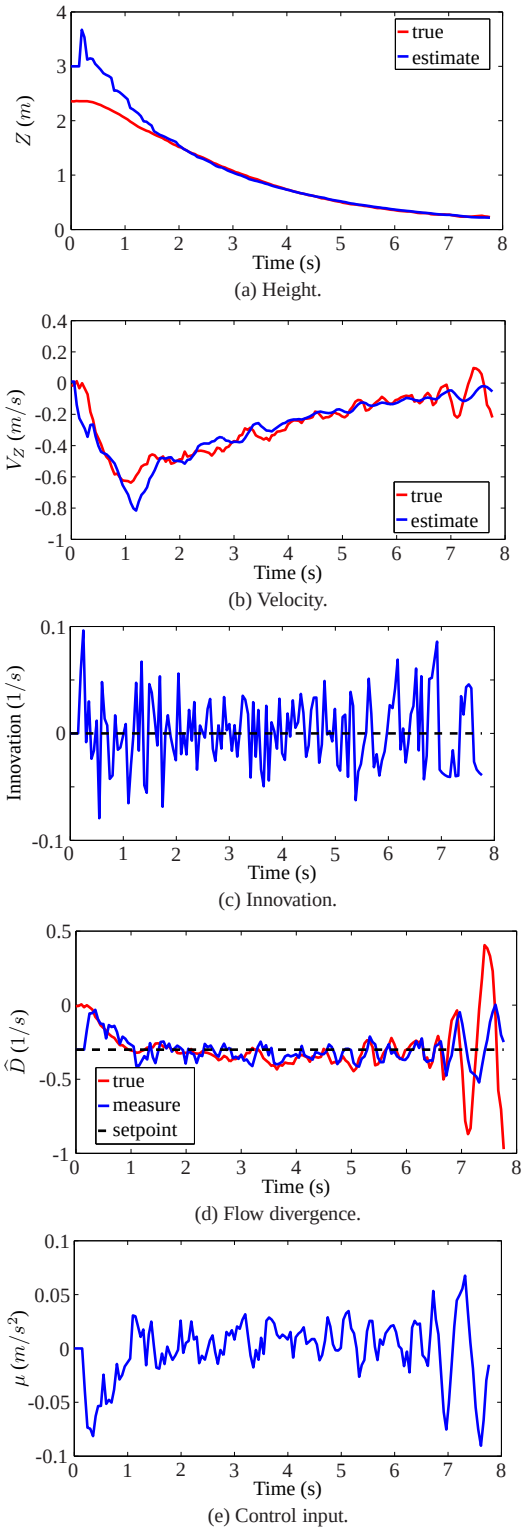


Figure 6: Landing with constant flow divergence ($D^* = -0.3\ s^{-1}$) with EKF-based height and velocity estimation.

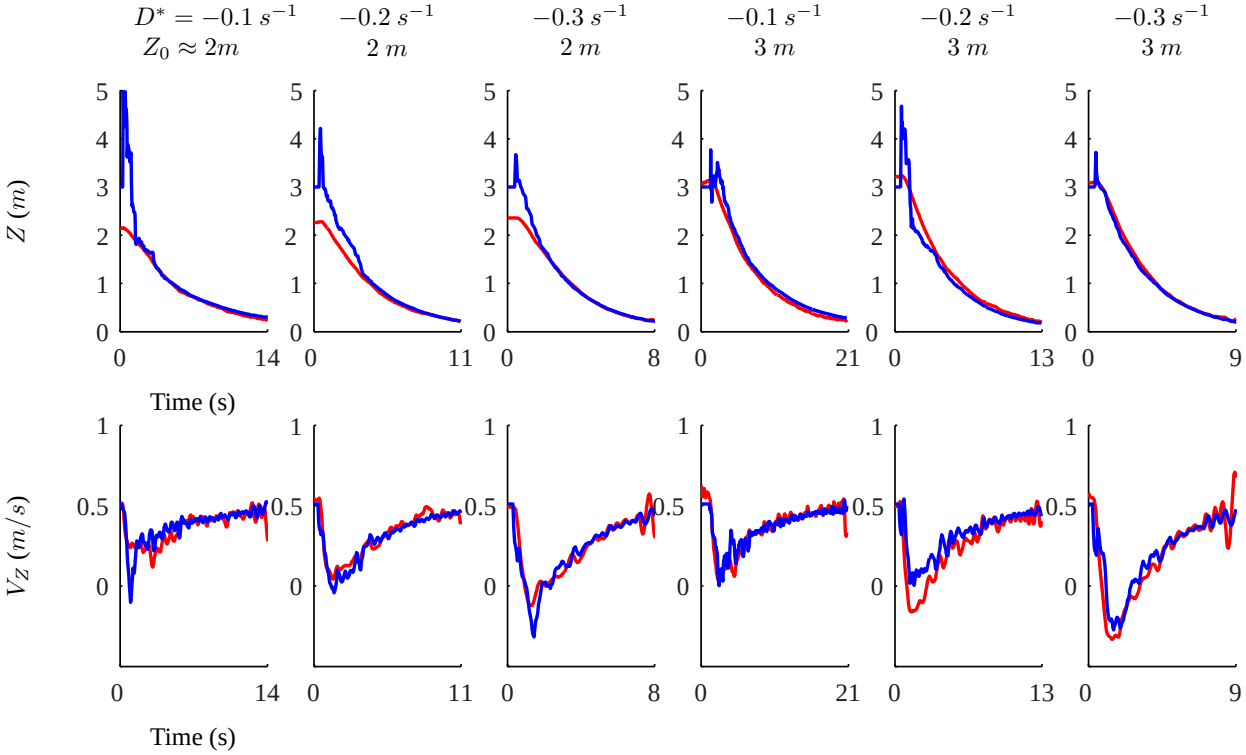


Figure 7: Multiple landings with different flow divergence setpoints ($D^* = -0.1\ s^{-1}, -0.2\ s^{-1}, -0.3\ s^{-1}$) at different initial heights ($Z_0 = 2m, 3m$). The estimates are shown with blue lines while their true values are represented by red lines.

black dash-dot line. This profile was generated using $x^*_{1,k} = x^*_{1,k-1} + x^*_2 \cdot \Delta t$ with $x^*_{1,0} = \hat{Z}_0$ and $x^*_2 = -0.2ms^{-1}$. The results show that the height and velocity can be estimated accurately using the proposed EKF algorithm and further used in the controller for landing. The zero mean innovation of the EKF also tells us that the filter was working properly.

The second initialization was conducted in an outdoor flight by tracking a desired divergence for the first few seconds as shown in Fig. 9. Since *OptiTrack* system is not available for outdoor flight, we can only use the onboard sensor, such as Sonar, to provide the true height and velocity (dZ/dt) for accuracy comparison.

The landing started at $5\ m$ and the desired divergence of $-0.2\ s^{-1}$ was tracked. After $2\ s$, the controller was switched to track a height profile as presented in Fig. 9a which was generated based on the velocity estimate at the instance when the controller was switched ($\approx -0.8\ ms^{-1}$). In fact, the height profile can also be created based on a desired velocity (see Fig. 8). The results show that both height and velocity estimates are accurately estimated, compared to the measurements from Sonar. In addition, the height estimate can also be used in landing control. It can also be observed from Fig. 9 that the estimates converge soon after the landing starts, thus the time interval for initialization can also be shortened. Note that the previous indoor flights were performed to validate the proposed algorithm using the highly accurate *OptiTrack* system while outdoor flight was conducted to show the robustness of the algorithm to the challenging outdoor environment. Some videos of the flight tests are available online [1].

6 CONCLUSION

In conclusion, we proposed an algorithm to use an extended Kalman filter (EKF) to estimate height and vertical velocity of an MAV from flow divergence while approaching a surface, and use these estimates for landing control. This algorithm was tested in computer simulation as well as in flight tests. The results show that the proposed EKF approach managed to estimate accurately the height and velocity of the MAV compared to the ground truth provided by the external cameras. In addition, the estimates were used in the controller to land the MAV. The proposed approach avoids the complexity of having nonlinearity in the constant flow divergence based landing by 'splitting' the flow divergence into height and velocity which allows the use of the linear controllers. For future works, adaptive Kalman filter can be used to improve the estimation when vision data is hardly observable and strong external disturbances are involved. Also, the algorithm opens up the possibilities to use gain optimization techniques to improve the performance of the control.

[1] Experiment videos: `https://goo.gl/KiJurr`

http://www.imavs.org/

http://www.imavs.org/

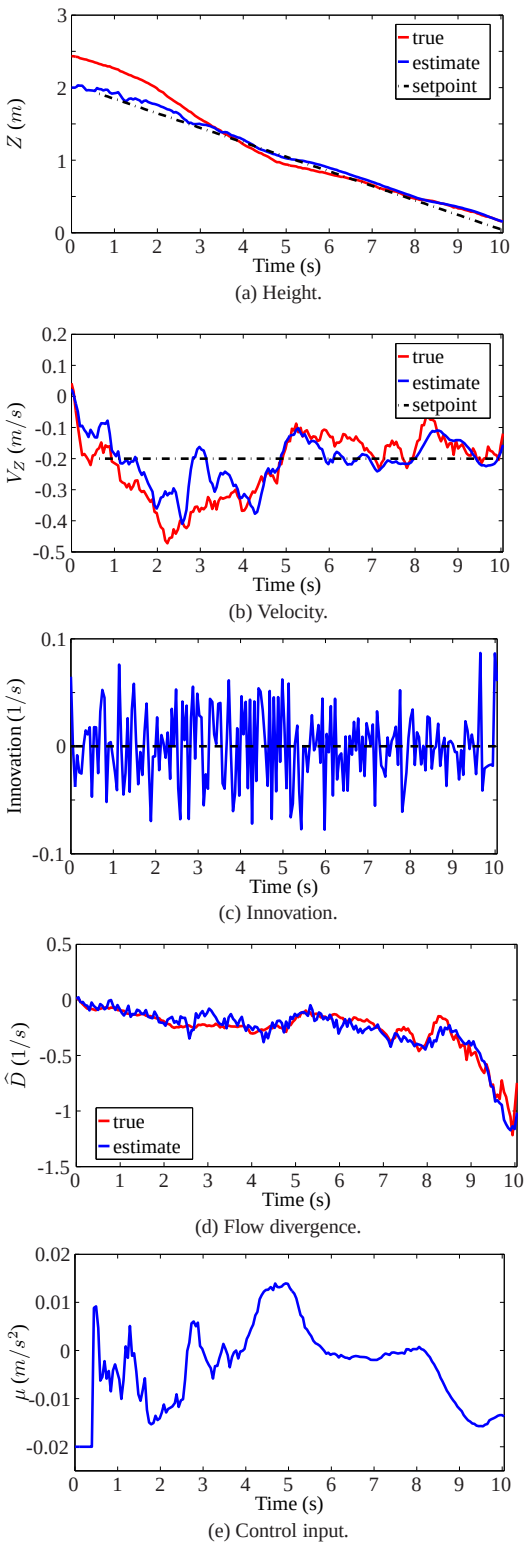


Figure 8: Landing with a linear controller using estimates from EKF (Indoor Flight).

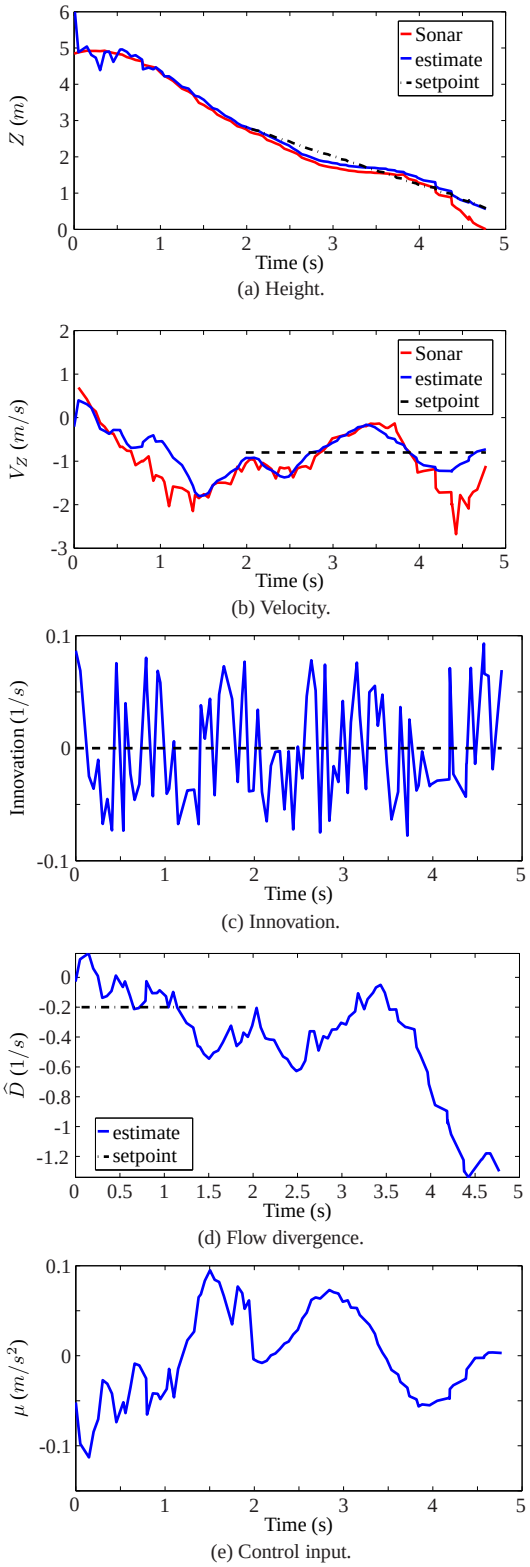


Figure 9: Landing with a linear controller using estimates from EKF (Outdoor Flight).

http://www.imavs.org/

REFERENCES

[1] C. de Wagter, S. Tijmons, B. D. W. Remes, and G. C. H. E. de Croon. Autonomous flight of a 20-gram flapping wing MAV with a 4-gram onboard stereo vision system. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4982–4987. IEEE, 2014.

[2] G. C. H. E. de Croon, H. W. Ho, C. de Wagter, E. van Kampen, B. D. W. Remes, and Q. P. Chu. Optic-flow based slope estimation for autonomous landing. *International Journal of Micro Air Vehicles*, 5(4):287–298, 2013.

[3] M. Srinivasan, S. Zhang, M. Lehrer, and T. Collett. Honeybee navigation en route to the goal: visual flight control and odometry. *The Journal of Experimental Biology*, 199(1):237–244, 1996.

[4] L. F. Tammero and M. H. Dickinson. The influence of visual landscape on the free flight behavior of the fruit fly drosophila melanogaster. *Journal of Experimental Biology*, 205(3):327–343, 2002.

[5] W. E. Green and P. Y. Oh. Optic-flow-based collision avoidance. *Robotics & Automation Magazine, IEEE*, 15(1):96–103, 2008.

[6] C. McCarthy, N. Barnes, and R. Mahony. A robust docking strategy for a mobile robot using flow field divergence. *Robotics, IEEE Transactions on*, 24(4):832–842, 2008.

[7] B. Herisse, F.-X. Russotto, T. Hamel, and R. Mahony. Hovering flight and vertical landing control of a VTOL unmanned aerial vehicle using optical flow. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 801–806. IEEE, 2008.

[8] F. Ruffier and N. Franceschini. Aerial robot piloted in steep relief by optic flow sensors. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1266–1273. IEEE, 2008.

[9] M. T Alkowatly, V. M Becerra, and W. Holderbaum. Bioinspired autonomous visual vertical control of a quadrotor unmanned aerial vehicle. *Journal of Guidance, Control, and Dynamics*, pages 1–14, 2014.

[10] H. W. Ho and G. C. H. E. de Croon. Characterization of flow field divergence for MAVs vertical control landing. In *AIAA Guidance, Navigation, and Control Conference*, page 0106, 2016.

[11] H. W. Ho, G. C. H. E. de Croon, E. van Kampen, Q. P. Chu, and M. Mulder. Adaptive control strategy for constant optical flow divergence landing. *ArXiv preprint arXiv:1609.06767*, September 2016.

[12] P. Bristeau, F. Callou, D. Vissiere, and N. Petit. The navigation and control technology inside the AR.Drone micro UAV. *IFAC Proceedings Volumes*, 44(1):1477–1484, 2011.

[13] J. Kelly and G. S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research*, 30(1):56–79, 2011.

[14] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 957–964. IEEE, 2012.

[15] F. van Breugel, K. Morgansen, and M. H. Dickinson. Monocular distance estimation from optic flow during active landing maneuvers. *Bioinspiration & biomimetics*, 9(2):025002, 2014.

[16] G. C. H. E. de Croon. Monocular distance estimation with optical flow maneuvers and efference copies: A stability-based strategy. *Bioinspiration & biomimetics*, 11(1):016004, 2016.

[17] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006*, pages 430–443. Springer, 2006.

[18] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1508–1515. IEEE, 2005.

[19] J. Bouquet. Pyramidal implementation of the Lucas Kanade feature tracker. *Intel Corporation, Microprocessor Research Labs, http://www. intel. com/research/mrl/research/opencv*, 2000.

[20] B. D. W. Remes, D. Hensen, F. van Tienen, C. de Wagter, E. van der Horst, and G. C. H. E. de Croon. Paparazzi: How to make a swarm of Parrot AR Drones fly autonomously based on GPS. In *International Micro Air Vehicle Conference and Flight Competition (IMAV2013)*, pages 17–20, 2013.

Electric Propulsion System Characterization through Experiments

Gautier Hattenberger*, Antoine Drouin* and Murat Bronz*
University of Toulouse ; ENAC; F-31077 Toulouse, France

http://www.imavs.org/

ABSTRACT

Electrical propulsion system characteristics are very important in UAV design, operation and control. This article presents the characterization of electric propulsion sets through experiments. A motor test bench have been build based on previous experience in order to improve the quality of the measurements. Moreover, the bench fits in a wind tunnel, allowing to perform a complete characterization over the full airspeed range of the considered mini and micro-UAVs. After recalling the general theoretical model of an electric motor, results from various combinations are presented.

1 INTRODUCTION AND MOTIVATIONS

Electrical propulsion system characteristics are very important in Unmanned Air Vehicle (UAV) design, operation and control. They have a strong impact on the consumption and therefore on the overall performances of the system in terms of range and endurance. Several models can be found in the literature [1] for continuous current motors or brushless motors. The second type is the most commonly used on mini and micro UAVs for their superior efficiency. Manufacturers are usually providing sufficient information to estimate the main characteristics of their products, however these data are sometimes optimistic regarding the real performances of the motors. Likewise, basic information provided for propellers are only the diameter and pitch values which are not sufficient for estimating the real performance values. Some other elements can effect the overall propulsive and electrical efficiency like the choice of propeller's material and electronic speed controllers (ESC).

For the *SkyScanner* project[1] [2], which deploys a fleet of mini-UAVs in order to study the evolution of clouds, several tasks require a good knowledge of the aircraft performances. It is especially the case of the trajectory planning algorithms and the wind estimation algorithms. This paper presents the experimental setup and results of a propulsion test bench which has been developed in order to build an accurate database for several motors and propellers. These information can be used for the design process [3, 4], for model-based control methods or for improving the estimation of other parameters like the wind or the aircraft performances [5]. The measurements of the propulsion system parameters need to be done at both zero speed for the static characteristics, and also over the complete airspeed range. Then, the propulsion test bench have been designed to be used inside a wind tunnel. Due to the available wind tunnel test section size, propellers only smaller than 12 inches can be tested, hence the force sensors that are used for thrust and torque measurements are selected accordingly.

The first sections will present the propulsion model and the experimental setup designed in-house. Finally, the experimental results are presented and discussed.

2 PROPULSION MODEL

2.1 Electric Motor

Basically, electric motors are electromechanical machines that converts electrical input power into mechanical output power. The general power supply used in the UAVs is DC (Direct Current) so DC motors will be investigated in this paper. Most common types are brushed and brushless motors. Brushed motors use mechanical and brushless motors use electronic commutation in order to change the direction of electric current and generate a pulling magnetic force between the stator and the magnets. Brushless motors have numerous advantages such as having a higher efficiency than brushed motors, longer lifetime, generating less noise, having higher power to weight ratio. Therefore they are more reliable for the UAV applications.

The important task is to choose the suitable motor for the specified mission requirements. In order to be able to select the correct motor, the characterization is a must to have.

Therefore, a first order simplified model using three motor constants, and experimentally obtained characteristics of DC motors will be explained in this section. Figure 2 shows an equivalent circuit model of an electric motor.

As described in [1], the resistance R of the motor is assumed to be constant and the motor shaft torque Q_m is proportional to the current i according to motor torque constant K_Q. The friction based losses can be represented by the no load current i_0 as a subtraction.

$$Q_m(i) = (i - i_0)/K_Q \tag{1}$$

Internal voltage v_m is assumed to be proportional to the rotation rate Ω according to the speed constant K_v of the mo-

*firstname.lastname@enac.fr
[1] https://www.laas.fr/projects/skyscanner/

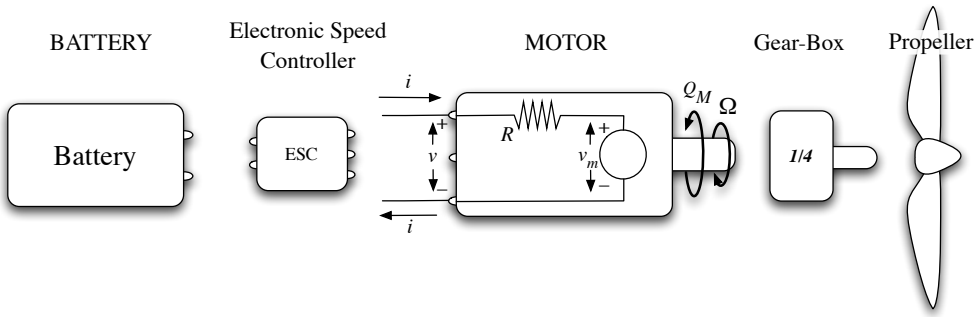


Fig. 1: Elements of a generic electric propulsion system.

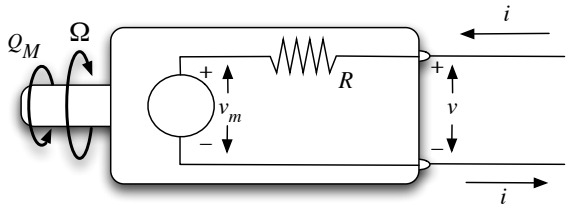


Fig. 2: Equivalent circuit for a DC electric motor[1].

tor.

$$v_m(\Omega) = \Omega/K_v \tag{2}$$

Then the motor terminal voltage can be obtained by adding the internal voltage and the resistive voltage drop.

$$v(i,\Omega) = v_m(\Omega) + iR = \Omega/K_v + iR \tag{3}$$

The above model equations can be rewritten in order to give power, torque, current and efficiency as a function of terminal voltage and rotation rate of the motor. Firstly, the current function is obtained from equation 3.

$$i(\Omega, v) = \left(v - \frac{\Omega}{K_v}\right)\frac{1}{R} \tag{4}$$

Then the others follow ;

$$Q_m(\Omega, v) = \left[i(\Omega, v) - i_0\right]\frac{1}{K_Q} = \left[\left(v - \frac{\Omega}{K_v}\right)\frac{1}{R} - i_0\right]\frac{1}{K_Q} \tag{5}$$

$$P_{shaft}(\Omega, v) = Q_m\Omega \tag{6}$$

$$\eta_m(\Omega, v) = \frac{P_{shaft}}{iv} = \left(1 - \frac{i_0}{i}\right)\frac{K_v}{K_Q}\frac{1}{1 + iRK_v/\Omega} \tag{7}$$

where P_{shaft} is the mechanical power on the shaft, K_v is usually given in RPM/Volt in motor specifications, however here it is taken as rad/s/Volt and K_Q is taken in Amp/Nm. It should be also noted that $K_Q \approx K_v$.

By knowing the first order motor constants (K_v, K_Q, i_0, R) of any off the shelf motor, the theoretical characteristic plots can be obtained by using above equations.

2.2 Propeller

The propeller is a rotating wing which utilizes the mechanical power input in order to accelerate the air particles to generate thrust. The basics of characterization of the propeller is included here, however a deeper explanation can be found in [6] and [7]. The thrust, power and torque coefficients are used to characterize a propeller, which depend on the advance ratio J (which can be found noted λ, eventually with a factor π compared to J), the average blade Reynolds number Re, and the geometry of the propeller. They are respectively defined as:

$$C_T = C_T(J, Re, geometry) \tag{8}$$

$$C_P = C_P(J, Re, geometry) \tag{9}$$

$$C_Q = C_Q(J, Re, geometry) \tag{10}$$

Reynolds number (R_e) of the propeller is defined as:

$$R_e = \frac{\rho\,\Omega R_b\, c_{ave}}{\mu} \tag{11}$$

where c_{ave} is the average chord length, ρ is the fluid density, μ is the dynamic viscosity, and R_b is the propeller blade radius. With all this parameters being constant, R_e becomes a function of the rotation speed Ω.

Advance ratio depends on the flight velocity V, and the propeller rotation speed

$$J(\Omega, V) = \frac{V}{nD} = \frac{V\pi}{\Omega R_b} \tag{12}$$

where $n = 2\pi\ \Omega$ is the number of revolutions per second and D is the propeller diameter.

Once the thrust C_T, power C_P and torque C_Q coefficients of the propeller are specified, the thrust and torque of the propeller can be calculated for any rotation rate and velocity as in [7],

$$T(J, \Omega) = T(\Omega, V) = \rho n^2 D^4 C_T(J, R_e) \tag{13}$$

$$P(J, \Omega) = P(\Omega, V) = \rho n^3 D^5 C_P(J, R_e) \tag{14}$$

http://www.imavs.org/

$$Q(J,\Omega) = Q(\Omega,V) = \rho n^2 D^5 C_Q(J,R_e) \quad (15)$$

Finally, the efficiency of the propeller at any condition can be found as,

$$\eta_{propeller}(\Omega,V) = \frac{T\,V}{P} = \frac{T\,V}{Q\,\Omega} = \frac{J\,C_T}{2\pi\,C_P} \quad (16)$$

3 EXPERIMENTAL SETUP

3.1 Previous design

In order to study the propulsion system performances, a preliminary test bench has been designed [5], resulting in promising results, but also with several flaws. This bench, shown in figure 3, was constructed with pre-formed aluminum bars and assembled 3D printed plastic (ABS) pieces. Two force sensors were used in order to measure the propulsion force and torque. A custom electronic board was measuring the electrical current and voltage, and the motor RPM as well. In addition, a Pitot tube placed in front was giving the wind tunnel speed. The complete testing procedure was controlled by a *MyRIO* data acquisition board from *National Instruments* and a *LabView* graphical interface.



Fig. 3: First test bench version.

This first design gave promising results as shown on figure 4, where the thrust is displayed as a function of the motor RPM for different airspeed. These tests have nevertheless showed some limitations. In particular, some combinations of motors and propellers were producing strong vibrations leading to false measurements.

The main reason was the rather fragile mechanical mounting of the bench. An other source of error was the position of the sensors. Their sensitivity and size were imposing to have them far from the tested motor, at the tip of the bottom aluminum bar for the thrust and using a 10 cm carbon rod at the back of bench for the torque. With longer connection, the vibrations were greatly increased at some airspeed and RPM combinations. The electric sensors (voltage, current, RPM) were integrated on custom electronic boards using rather long cables, which adds some measurement noise.

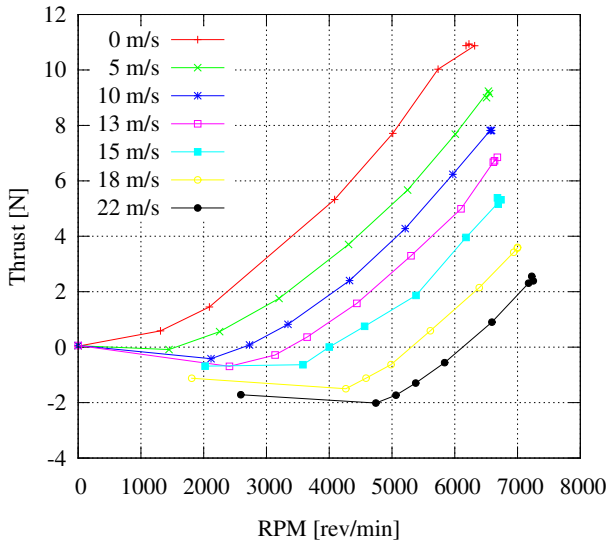


Fig. 4: Thrust versus RPM at different airspeed from wind tunnel experiment.

3.2 Current Propulsion Test Bench

Keeping in mind the previous design flaws, the new test bench has been designed accordingly while giving more attention to the below mentioned items:

- the structure should be stiffer
- the force and torque sensors should be placed closer to the motor and propeller
- the electric sensors should have a better integration
- optionally, the amount of sensors used has to be reduced.

Figure 5 shows the new version of the propulsion test bench inside the wind tunnel of ENAC (French Civil Aviation University) that is build based on these considerations. The wind tunnel has a 0.5m by 0.5m open test section with 1 meter long. It is mainly used for instructional purpose for the aviation students, however it is still suitable to be used for research purposes. Figure 6 shows the detailed CAD drawing where the sensors are placed directly to the main shaft holding the motor and propeller couple to reduce the additional structural vibrations. The two force sensors that are used for thrust and torque measurements are respectively SMD2207-050 and SMD2207-020 from *strain measurement devices*.[2] The main shaft is free to move forward and backward and also rotates freely in order to transfer force and moments directly to the sensors. The linear and rotational motion is obtained by using thin wall bearings FMT-012 from *PCB*.[3] The main

[2] www.smdsensors.com
[3] www.pcblinear.com

http://www.imavs.org/

http://www.imavs.org/

shaft is free to rotate and translate, however it is held by the sensors measuring the thrust and torque. Hence the moment of inertia of the main shaft has no impact on the measurements. The clean integration of the sensors and the power cables to the main test bench mast minimize their additional effects on the measurements.

The speed controller used for the new test bench is ESC32v3[4]. The main reason is that it can directly give the motor voltage, the current draw, and the RPM via a serial link, so that reducing the number of required sensors on the system.

Finally the bench is fixed on a wooden plate placed in the wind tunnel test section, where the length can vary in order to fit into bigger wind tunnel test sections.

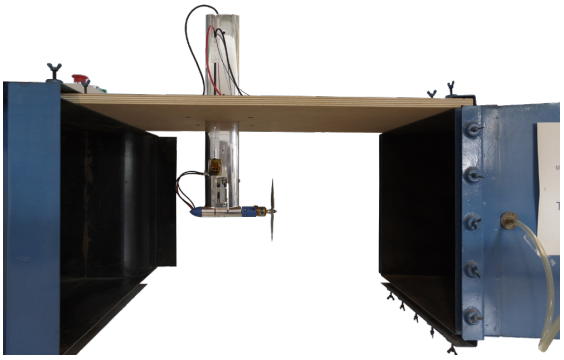


Fig. 5: Improved new test bench version inside the wind tunnel.

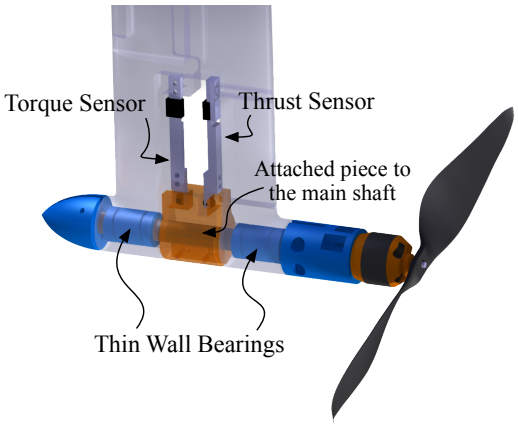


Fig. 6: The detailed CAD view of the sensor integration used in the new version of propulsion test bench.

3.3 Calibration Process

Calibration of the sensors is one of the most important issues for the correct measurements. Usually this process is difficult to make and done only once in a while. However, it has been observed that after the replacement of propeller or motor, the torque sensor is getting effected and changing its calibration values. In order to overcome this problem, and make sure that the test bench is well calibrated at each experiment, easy to attach and remove clip-on calibration parts have been designed. The attached pieces are shown in Figure 7.

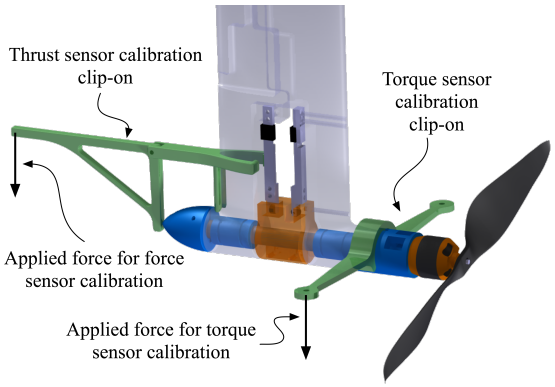


Fig. 7: Clip-on calibration parts.

The objective while designing the clip-on parts was to make the calibration process as easy and fast as possible. On the graphical user interface, the calibration window consists of several predefined loading values. After placing the clip-on to its designated location, the user starts applying the predefined load from the locations that are shown in Figure 7. A dynamic point, moving in real-time on the graph, shows the location of the load value versus raw values. The user can go over all the predefined loads and recalibrate the sensor, or just use the graph in order to check the fit of calibration curves for any given loading.

One of the calibration steps consists of subtracting the motor and shaft drag, when there is no propeller, from the measurements. Thanks to the design of the test bench this correction remains very low.

A particular problem that has been encountered was the static friction of the thin wall bearings. During the calibration, the small loadings would not overcome the static friction of the system and became *invisible* to the sensors. This has been solved by adding a small vibration to the system. The motor is rotated really slowly, so slowly that it does almost not generate any thrust, however the phase changes of the magnets on the motor vibrates the shaft. This effect naturally exists during all the real tests and was sufficient to overcome the static friction so that even the smallest loads such as 0.05 N can be used during the calibration process.

[4] www.autoquad.org/esc32/

http://www.imavs.org/

4 RESULTS

4.1 Thrust, Power and Torque Coefficients

The C_T, C_P and C_Q coefficients can be generated from the measurements using equations 13, 14 and 15. In addition, we assume that in the useful range of flight speed (around 10 to 20 m/s), these coefficients can be linearized as follow:

$$C_T(J,\Omega) = \frac{T}{\rho n^2 D^4} = C_{T_0} + C_{T_J} J + C_{T_{RPM}} RPM \quad (17)$$

$$C_P(J,\Omega) = \frac{P}{\rho n^3 D^3} = C_{P_0} + C_{P_J} J + C_{P_{RPM}} RPM \quad (18)$$

$$C_Q(J,\Omega) = \frac{Q}{\rho n^2 D^5} = C_{Q_0} + C_{Q_J} J + C_{Q_{RPM}} RPM \quad (19)$$

where $C_{\{T,P,Q\}_{\{0,J,RPM\}}}$ are constant coefficients. This assumption have been verified experimentally as shown on Figure 8 that plots the thrust coefficient as a function of J and RPM, and its linear regression plane corresponding to the above model.

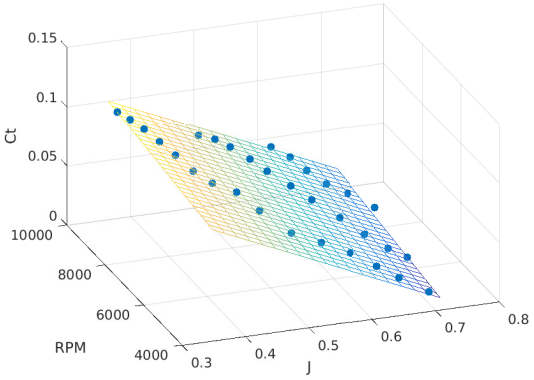


Fig. 8: Thrust coefficient (C_T) as a function of advance ratio (J) and RPM. The plane is showing a multiple linear regression of C_T over these two coefficients.

The tested configurations are all using the same motor (*AXI-2212/26*) with different propellers (glass reinforced *HQprop* propellers, see Figure 9) identified by the number of blades, the diameter (in inches) and the pitch (in inches):

- 2 blades, 8x5
- 2 blades, 6x4.5
- 2 blades, 5.5x4.5
- 3 blades, 5x4.5
- 3 blades, 4x4.5



Fig. 9: Propellers evaluated during the experiments.

prop	C_{T0}	C_{TJ}	C_{TRPM}	R^2
2_8x5	1.565e-01	-2.320e-01	3.260e-06	0.9907
2_6x4.5	1.856e-01	-2.139e-01	1.524e-06	0.9882
2_5.5x4.5	2.032e-01	-2.320e-01	-1.271e-07	0.9896
3_5x4.5	2.749e-01	-2.617e-01	-6.336e-07	0.9362
3_4x4.5	2.942e-01	-2.580e-01	1.846e-06	0.8700

Tab. 1: Results for C_T.

prop	C_{P0}	C_{PJ}	C_{PRPM}	R^2
2_8x5	9.045e-02	-1.097e-01	2.453e-06	0.9782
2_6x4.5	1.542e-01	-1.363e-01	1.497e-06	0.9760
2_5.5x4.5	1.306e-01	-1.247e-01	1.686e-06	0.9675
3_5x4.5	2.484e-01	-2.108e-01	2.381e-06	0.9864
3_4x4.5	3.358e-01	-2.559e-01	7.290e-06	0.9657

Tab. 2: Results for C_P.

prop	C_{Q0}	C_{QJ}	C_{QRPM}	R^2
2_8x5	1.440e-02	-1.746e-02	3.904e-07	0.9782
2_6x4.5	2.455e-02	-2.170e-02	2.383e-07	0.9760
2_5.5x4.5	2.078e-02	-1.985e-02	2.683e-07	0.9675
3_5x4.5	3.953e-02	-3.356e-02	3.789e-07	0.9864
3_4x4.5	5.344e-02	-4.073e-02	1.160e-06	0.9657

Tab. 3: Results for C_Q.

The results are presented in Tables 1, 2 and 3, with the coefficient of determination R^2 for each of the linear regressions. These results are matching over data found in the literature [8], validating the methodology.

It is important to note that typical values for the advance ratio J are between 0.3 and 1, while the RPM is between 3000 and 10000 rotations per seconds. So, the magnitude difference between the two is around 10^5, which is also the

http://www.imavs.org/

magnitude difference between the $C_{\_J}$ and $C_{\_RPM}$ coefficients. The conclusion is that, in most cases, the effect of *RPM* is as great as J, while the literature usually only consider the relation with J. The main reason is probably that for larger motors, propellers and flight speeds, the effect of the Reynolds (and hence *RPM* assuming over parameters mostly constant) is less important.

Complete raw value plots of the *HQprop 8x5* propeller are presented in Appendix A.

4.2 *Relation between Aerodynamic Power and Electrical Power*

In addition to the equations presented in section 2, let's define the propulsive power as the product of the thrust and the flight speed,

$$P_{aero} = V\,T \tag{20}$$

and recall the electrical power,

$$P_{elec} = iv \tag{21}$$

The total energy of the system is then the ratio between this two values, and is mainly a function of the speed:

$$\eta_{tot}(V) = \frac{P_{aero}}{P_{elec}} \tag{22}$$

Data analysis have shown that P_{aero} and P_{elec} have a linear relation relatively independent of the speed V in the usual flight speed range of the considered UAVs. This can be seen in Figure 10.

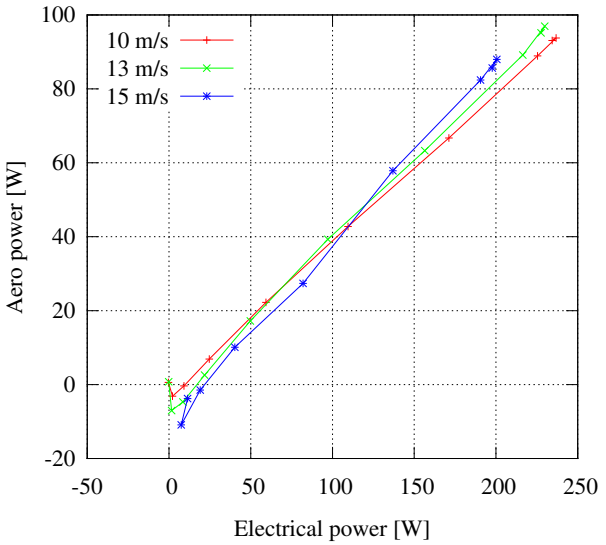


Fig. 10: Propulsive power as a function of input electrical power.

This relation is very useful since the electrical power can be easily measured by on-board voltage and current sensors. Thus, an estimation of the aerodynamic propulsive power can be used in control or other estimation algorithms involving the aircraft model. The linear model is defined as follow:

$$P_{aero} = a\,P_{elec} + b \tag{23}$$

Results are presented in table 4.

prop	a	b	R^2
2_8x5	0.4780	-9.554	0.9527
2_6x4.5	0.4287	-7.355	0.9584
2_5.5x4.5	0.4630	-7.514	0.8826
3_5x4.5	0.4303	-7.683	0.8604
3_4x4.5	0.3718	-6.876	0.7468

Tab. 4: Results for P_{aero} as a function of P_{elec}.

By looking at the data, it can be seen that the b coefficient is rather small compared to the magnitude of P_{aero} and P_{elec}. By neglecting it, the coefficient a, corresponding to the slope of the curve, is then equal to total efficiency η_{tot}. As expected, larger propellers have a better efficiency. In the case of smaller ones, the correlation factor is decreasing, which means that the assumption of linear relation being independent of the speed may not be valid anymore, or only valid for a smaller speed range.

5 CONCLUSION

This article have presented a test bench that is build in order to characterize the propulsion sets of small UAVs. The design is an improvement of an earlier version in order to get better and more stable results.

The presented results are covering thrust, power and torque coefficients, as well as propulsion sets efficiency and some useful relations. They can be used by researchers and engineers both during the aircraft design phase and for UAV control.

Only five propellers with the same motor have been studied so far, but the calibration and measurement procedure have been validated, and further experiments will be done.

ACKNOWLEDGMENTS

The *SkyScanner* project is founded by the RTRA-STAE foundation.

REFERENCES

[1] Mark Drela. First-order dc electric motor model. Technical report, MIT, Aero and Astro, February 2007.

[2] S. Lacroix, G. Roberts, E. Benard, M. Bronz, F. Burnet, E. Bouhoubeiny, J.-P. Condomines, C. Doll, G. Hattenberger, F. Lamraoui, A. Renzaglia, and C. Reymann. Fleets of enduring drones to probe atmospheric phenomena with clouds. In *European Geosciences Union General Assembly 2016*, 2016.

[3] Murat Bronz, Gautier Hattenberger, and Jean-Marc Moschetta. Development of a long endurance mini-uav: Eternity. *International Journal of Micro Air Vehicles*, 5(4):261–272, 2013.

[4] Murat Bronz, Jean-Marc Moschetta, and Gautier Hattenberger. Multi-Point Optimisation of a Propulsion Set as Applied to a Multi-Tasking MAV. In *IMAV 2012, International Micro Aerial Vehicle Conference and Competition*, Braunschweig, Germany, July 2012.

[5] Jean-Philippe Condomines, Murat Bronz, Gautier Hattenberger, and Jean-François Erdelyi. Experimental wind field estimation and aircraft identification. In *IMAV 2015: International Micro Air Vehicles Conference and Flight Competition*, Aachen, September 2015.

[6] Mark Drela. Dc motor and propeller matching , lab 5 lecture notes. Technical report, MIT, March 2005.

[7] McCormick B.W. *Aerodynamics, Aeronautics & Flight Mechanics*. John Wiley & Sons, Inc., 1979.

[8] Robert W Deters, Gavin K Ananda, and Michael S Selig. Reynolds number effects on the performance of small-scale propellers. In *Procedings of 32nd AIAA Applied Aerodynamics Conference, American*, 2014.

APPENDIX A HQPROP 8X5 MEASUREMENTS

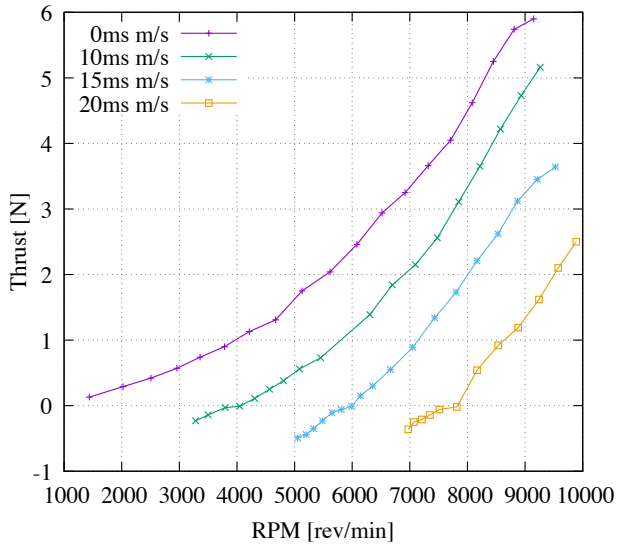


Fig. 11: Thrust vs RPM

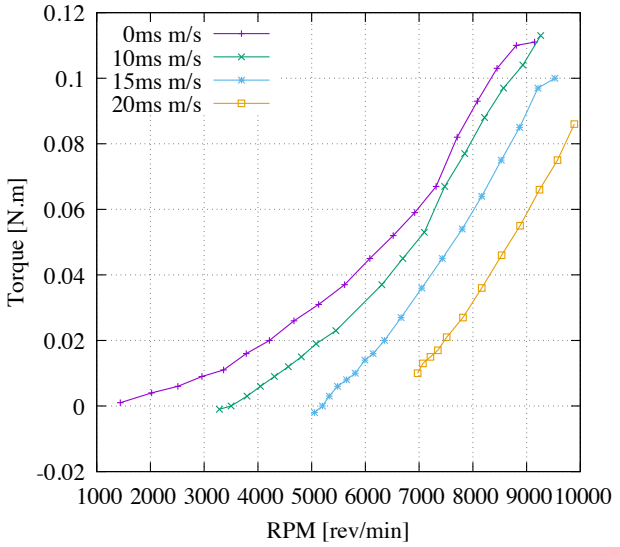


Fig. 12: Torque versus RPM

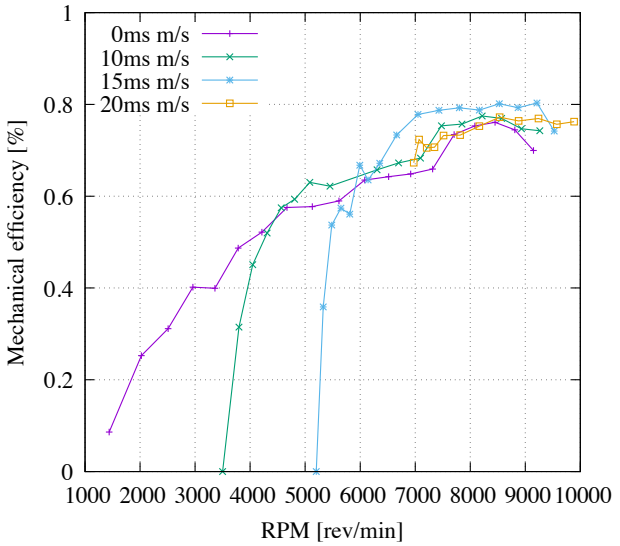


Fig. 13: Mechanical efficiency versus RPM

http://www.imavs.org/

http://www.imavs.org/

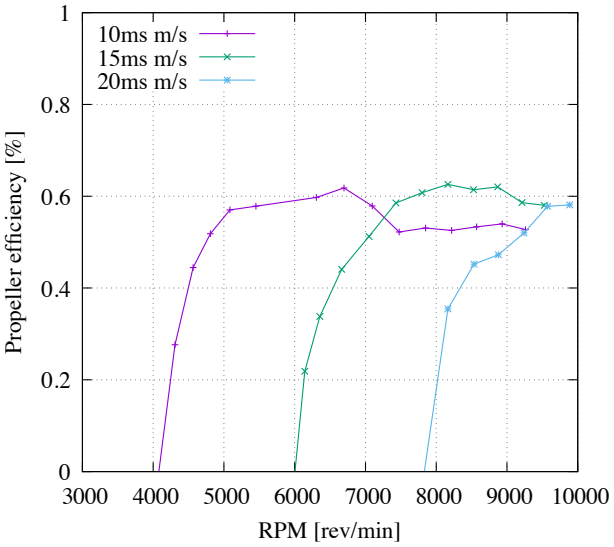


Fig. 14: Propeller efficiency versus RPM

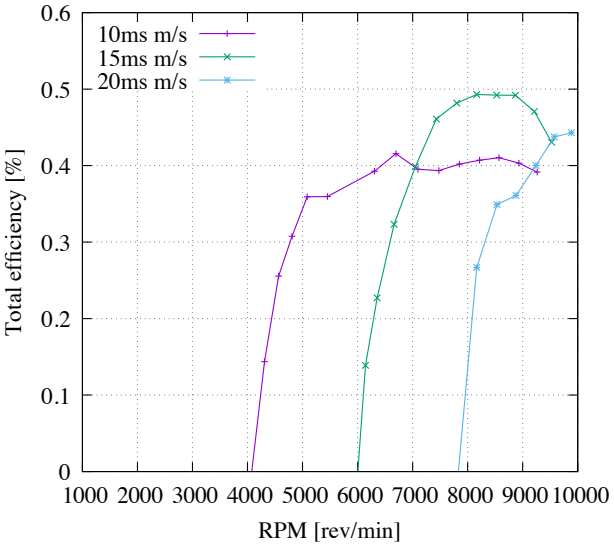


Fig. 15: Total system efficiency versus RPM

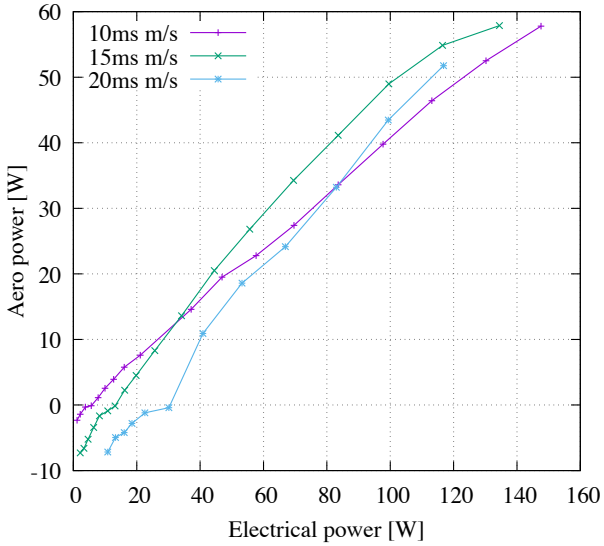


Fig. 16: Propulsive power versus electrical input power

Experimental and Computational Investigation on Interaction between Nano Rotor and Aerodynamic Rudder

D. YANG, Z. LIU*, C. BU
State Key Laboratory for Strength and Vibration of Mechanical Structures, school of Aerospace Xi'an Jiaotong University, Xi'an, China

http://www.imavs.org/

ABSTRACT

The interference effect between the nano rotor and aerodynamic rudder was studied experimentally and computationally. Propulsive performance of nano rotor and aerodynamic performance of aerodynamic rudder were achieved experimentally. The disturbed flow field of nano rotor was also analyzed computationally to disclose the flow mechanics of the interaction. Results showed that the nano rotor has a great effect on the aerodynamic performance of aerodynamic rudder. The moment of aerodynamic rudder fluctuated with the rotor-to-rudder spacing and achieved the smallest value at the spacing of 0.5 R. And the moment of aerodynamic rudder varied with deflection angle linearly. Aerodynamic rudder influenced the propulsion performance of the nanor rotor slightly. The thrust coefficient and torque coefficient increased a little with spacing but changed slightly with the deflection angle. Numerical simulation showed that aerodynamic rudder blocked the flow field of the nano rotor and the counter-clockwise rotation of the rotor drives the flow in the downstream rotating in a counterclockwise direction resulting in the different angle of attack between left and right rudder surface.

1 INTRODUCTION

Aerodynamic rudder is one of the useful methods to control the attitude of rotary-wing nano air vehicle (NAV) due to the limitation of its size and weight [1][2]. The interaction between nano rotor and aerodynamic rudder induces complex flow phenomenon and influences the performance of both nano rotor and rudder. Therefore, it is necessary to study the interference effect and disclose the inherit flow mechanics so as to design nano rotor and rudder with high performance.

Nano rotor is characterized by small size and low rotational velocity causing that its operational Reynolds number is very low which is usually below 2.0×10^4 or less. At low Reynolds number, it is laminar flow and unstable. Due to the weak inertia force in the boundary layer, strong adverse pressure gradient appears at the maximum suction point of the leading edge. As a result, the flow separates from blade surface and wake vortices generate. The vortices interact with rudder resulting in the change of control efficiency of rudder. On the contrary, the rudder in the wake flow blocks the wake flow of the nano rotor and influences the propulsion performance of nano rotor.

The interference effect between propeller and wing is well studied . Fratello et al. [3] investigated the mutual interference of large propeller and wing using wind tunnel and lifting surface method. Results showed that the drag coefficient of the wing increased while the lift coefficient of the wing decreased. On the other hand, both the thrust coefficient and power coefficient of the propeller increased. Moens et al. [4] used the actuator disk model to replace the propeller. And the interference effect between the slipstream of propeller and wing was studied using Navier-Stokes solver. Gamble [5] tested the interaction between a micro propeller and a wing in the wind tunnel and it was found that 12% to 18% thrust of the propeller transformed to the drag of the wing. Hu et al. [6] studied the influence of the rudder to the wake flow of propeller. Potential flow method was used to establish the integrated formulation of the strength of doublet at the surface of propeller blade. As the induced flow velocity was solved, the influence of the rudder to the propeller was detected. Wang et al. [7] employed Reynolds average N-S equations to establish the interference model of propeller to rudder. Results showed that the propulsion performance of the front propeller was not sensitive to the horizontal and vertical position of the rudder. Yang et al. [8] used the lifting surface method to study the rudder performance in the wake flow of propeller. Duan et al. [9] analyzed the influence of propeller slipstream to the aerodynamic performance of the wing at different angles of attack using the sliding mesh technique. And results showed that the thrust of propeller changed as the result of wing and the lift of wing increased at high angle of attack because of the influence of propeller.

In summary, the experimental and numerical methods are mainly employed to study the mutual interference effect between the rotor or propeller and the wing. These researches provide the study of interference of nano rotor and aerody-

*Email address: liuz@mail.xjtu.edu.cn

http://www.imavs.org/

namic rudder with some references. However, the interference effect between rotor and rudder is scarcely studied because most of research focuses on the aerodynamic interaction between propeller and wing. As the aerodynamic rudder operates at different deflection angles, it has a high impact on the propulsion performance of rotor, especially at the high deflection angle. Therefore, it is necessary to investigate the mutual interference between rotor and rudder. Furthermore, most of the researches above studied the interaction at high Reynolds number. It shall be pointed out that the rotary-wing NAV operates at an ultra-low Reynolds number. At ultra-low Reynolds number, the phenomenon, that is laminar separation and laminar separation bubble, appear due to the adverse pressure gradient in the boundary layer, which results in the unsteady non-linear aerodynamic characteristics for flight vehicles [10]. In this paper, the mutual interference effects between nano rotor and aerodynamic rudder at ultra-low Reynolds were therefore studied experimentally and numerically. The influence of rudder deflection angle and rotor-to-rudder spacing to nano rotor propulsion performance and aerodynamic rudder aerodynamic performance was investigated. And the interfered flow field of nano rotor was studied to find out the radical reason for the variation of rotor performance. It is believed that the research will provide the design of rotary-wing NAV with a reference.

2 EXPERIMENTAL SETUP

2.1 Nano rotor and Aerodynamic rudder

Conventional helicopter rotor typically has uniform chord. It controls attitude and fight direction through collective and cyclic pitch control. Because small MAVs are characterized by extremely small size and weight, conventional controls are not feasible. Therefore, rotor blades with twisted and non uniform chord are particularly attractive for propulsion optimization. Thus a nano rotor with diameters of 7.5 cm was designed based on low Reynolds number aerodynamics and minimum induced loss theory to minimize energy loss. The blade airfoil has 2% thickness with 5% curvature circular arc. The rotor has a mean chord of 0.33R and mean twist angle of 17.21 as shown in Figure 1. The geometry of nano rotor is shown in Figure 2. After obtaining chord length and twist angle distribution, the nano rotor was fabricated with carbon laminar and epoxy resin.

To control the attitude of rotary-wing NAV, an aerodynamic rudder system is installed in the wake flow of nano rotor. The aerodynamic rudder system includes two pieces of rudder as shown in Figure 3. Each piece of rudder is a rectangle flat plate whose the long edge is 0.8 R long and the short edge is 0.53 R long. Here R represents the radius of nano rotor. To avoid the influence of motor and increase the control efficiency of aerodynamic rudder, a horizontal spacing of 0.4 R exists between two pieces of rudder. The rudder was fabricated with carbon laminar and epoxy resin as well with a thickness of 0.3mm.

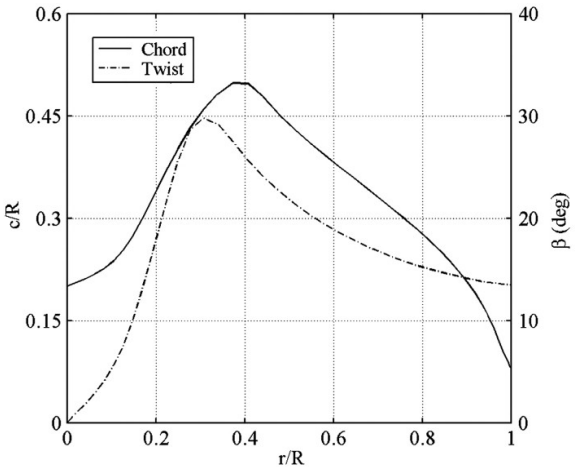


Figure 1: Blades chord and twist distribution of nano rotor.

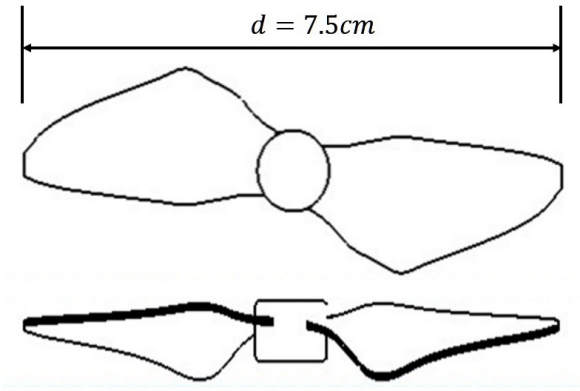


Figure 2: Schematic of of nano rotor.

2.2 Interference Effect Test Bench

An interference effect test bench was designed to measure the thrust and torque of nano rotor and the control moment and force of rudder due to interference effect as shown in Figure 4. The test bench was composed of aerodynamic rudder test part, nano rotor test part, energy supply system and the data acquisition system.

The rudder system consists of the aerodynamic rudder, a small servo, a control panel, a 6-DOF balance band a support beam. The aerodynamic rudder was assembled horizontally with a support beam under the wake flow of nano rotor. The support beam was fixed on the small servo by which the rudder can change its deflection angle accurately. The control command was generated by the computer. Then it was transferred to the servo by virtual of control panel typed Arduino Mega,ATmega1280. The 6-DOF balance is a ATI nano 17 (Ti) balance with a maximum force capacity of 14.1 N and a maximum moment capacity of 50 N·mm. The aerodynamic

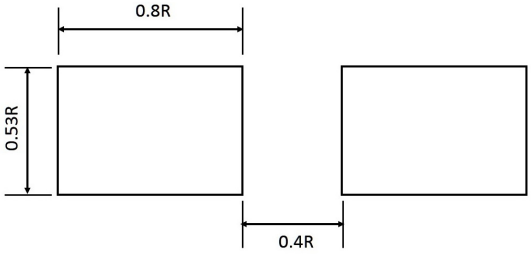


Figure 3: Schematic of aerodynamic rudder.

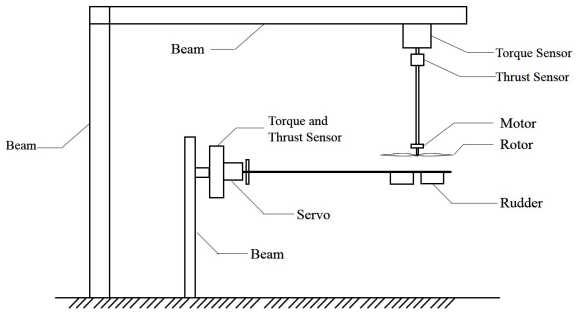


Figure 4: Schematic of interference effect test bench.

force and moment can then be measured with the balance.

The nano rotor test part consists of the nano rotor, speed controller, a torque sensor, a load cell, a support beam. In order to establish the pure torque of the rotor, an ultra-low-capacity static torque sensor DH15 by the SCAIME company with a capacity of 0.005 N·m and an accuracy class of 0.1% was used to measure the torque. In addition, the load cell for measuring the thrust was a load cell Honeywell Model 31 with a capacity of 2.0 N. The torque cell was installed on the load cell directly. And an extended supporting beam was installed vertically supporting the motor and rotor to avoid the effect of the torque. The controller was YGE4-BL for brushless motors. The speed control command generated by computer was transferred to speed controller with which the rotational speed of nano rotor can be adjusted.

The energy supply system is a regulated DC power supply, which can adjust the voltage and stabilize it at a certain value to provide micro motor and servo with current.

The data acquisition system was composed of a speedometer, an amperemeter, a voltmeter, a MDR data recording system and an USB analog-to-digital data acquisition (DAQ) of NI USB-6251 BNC. The speedometer was used to measure the speed of nano rotor. The electric parameters measurement system consists of an amperemeter and a voltmeter to measure the input current and voltage. The MDR acquisition system and the USB-6251 can acquire the torque and force parameters of nano rotor and aerodynamic rudder. A control software was developed to generate control command and save the acquisition data.

3 COMPUTATION METHODOLOGY

To capture the detailed flow field characteristics of nano rotor, Computational Fluid Dynamics (CFD) method was used. Because of the special flight condition of nano rotor, the operating Reynolds number of nano rotors is typically lower than 20,000 and the blade tip Mach number of nano rotors is less than 0.1 Ma. For low-Mach and low-Re flow, conventional NS equations might fail to converge to a correct solution. Since the rotor rotates at high speed, the rotor motion can be regarded as a quasi-steady state during numerical simulation when the rotation is stable. Therefore, 3D incompressible Navier-Stokes (INS) solver was used to compute the aerodynamic performance of nano rotor at ultra-low Re. The simulation was performed based on NS equations in a coordinate system which rotates around the y-axis with an angular velocity Ω. The steady incompressible Navier-Stokes equations in integral form for an arbitrary control volume are written in three dimensions as follows.

$$\frac{\partial}{\partial t}\iiint_V \vec{Q}dV + \oiint_{\partial V}(\vec{F}-\vec{F}_v)\cdot\vec{n}dS = \iiint_V \vec{W}dV \qquad (1)$$

where $\vec{Q} =$

$$\begin{bmatrix} p \\ u \\ v \\ w \end{bmatrix}$$

$\vec{F} =$

$$\begin{bmatrix} \beta(\vec{U}-\vec{U}_g) \\ u(\vec{U}-\vec{U}_g)+p\vec{i} \\ v(\vec{U}-\vec{U}_g)+p\vec{j} \\ w(\vec{U}-\vec{U}_g)+p\vec{k} \end{bmatrix}$$

$\vec{F}_v =$

$$\begin{bmatrix} 0 \\ \tau_{xx}\vec{i}+\tau_{xy}\vec{j}+\tau_{xz}\vec{k} \\ \tau_{yx}\vec{i}+\tau_{yy}\vec{j}+\tau_{yz}\vec{k} \\ \tau_{zx}\vec{i}+\tau_{zy}\vec{j}+\tau_{zz}\vec{k} \end{bmatrix}$$

$\vec{W} =$

$$\begin{bmatrix} 0 \\ \Omega w \\ 0 \\ -\Omega u \end{bmatrix}$$

Here, $\vec{Q}$ is vector of primitive flow variables,$\vec{F}$ termed vector of convective fluxes are related to the convective transport of quantities in the fluid.$\vec{F}_v$ termed vector of viscous

http://www.imavs.org/

http://www.imavs.org/

fluxes contain the viscous stresses τ_{ij} , In the formula,$\vec{U}$ and $\vec{U}_g$ are the velocity component and moving grid velocity component which can be expressed as

$$\vec{U} = u\vec{i} + v\vec{j} + w\vec{k} \tag{2}$$

$$\vec{U}_g = u_g\vec{i} + v_g\vec{j} + w_g\vec{k} \tag{3}$$

The equations were solved with finite volume method and Roes flux scheme was employed.

Since the rotation of rotor can be taken as a quasi-steady case, multi reference frame(MRF) was applied to describe the rotor rotation. The MRF model is a steady-state approximation in which individual cell zones use different frame equations to solve N-S equations. Zones containing moving components can then be solved using moving reference frame equations, whereas stationary zones can be solved with stationary frame equations. It is convenient to use MRF so that the steady-state solutions are possible with relatively short computing time and relatively high precision. As shown in Figure 5, thus the flow field is divided into three zones in which two small zones are dedicated to the nano rotor and the aerodynamic rudder, respectively. The external mesh boundary is formed by a 25.33R high cylinder with the top and bottom radii of 12R and 16R, respectively. Every small block has a height of 2/3R and a diameter of 1.5R. The upper small block is dedicated to the nano rotor which rotates around the y-axis with an angular velocity Ω as shown in Figure 6. And the lower small block is a static fluid zone for aerodynamic rudder. The fine O-topology was employed and the distance of between the first layer of mesh and the solid surface is 5.0×10^{-4}. The number of grid cells for the upper block is 1.8 million and that for lower block is 1.9 million. The total number of grid cells for the whole flow field is about 6.0 million.

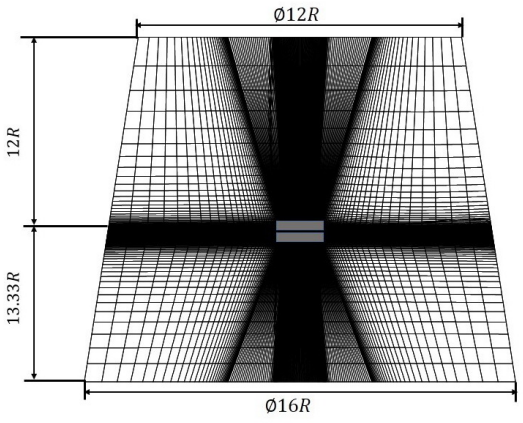


Figure 5: Global view of the mesh.

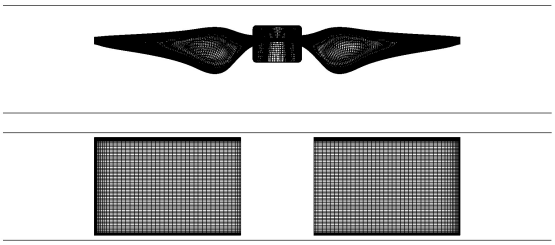


Figure 6: Surface mesh of nano rotor and aerodynamics rudder.

4 EXPERIMENT DESIGN AND COMPUTATION DESIGN

To investigate the interference effect, two parameters that are rotor-to-rudder spacing d and rudder deflection α shall be studied. Experiments were carried out with spacing varying from 0.3 R to 1.5 R while rudder deflection angle changes from 0 degree to 50 degree with an increment of 10 degree. The detailed parameters were listed in Table 1. Therefore, sixty groups of experiments were performed with different rotor-to-rudder spacing and rudder deflection. During the experiments, the nano rotor was driven by motor MICRO at a voltage of 3.6 V. At this voltage, the rotational velocity could be adjusted by the controller with PWM signals. But all the experiments were performed with the rotor rotational velocity of 6500 RPM. Aerodynamic rudder was driven by a small servo to adjust rudder deflection angle. The moment of aerodynamic rudder was measured with the 6-DOF balance. And the thrust and the torque of nano rotor were measured with load cell and torque sensor, respectively.

Parameter	Values
Spacing d/R	0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.2 1.5
Deflection angle α /°	0 10 20 30 40 50

Table 1: List of spacing and deflection angle.

The experimental data collected by the MDR acquisition system belong to temporal signal which often superimposed by various signals, such as high-frequency signals and low-frequency signals. In addition, the signals generated by the rotor are mainly low-frequency signals due to the low weight of nano rotor. The traditional Fourier transform is difficult to get the desired results. Therefore, all the signals were processed with EMD analysis tools during the experiment to eliminate the influence of external disturbance. Prior to the experiments, calibrations of test benches were performed to eliminate nonlinearity of load cell and uncertainty factors. The calibration was carried out before the experiments. Results showed that the relative fit errors of of load cell are very small and that of the torque sensor is higher.

In the experiment, two principal sources of uncertainty

http://www.imavs.org/

are contained. One is the bias errors inherent in the measurement devices with regard to offset and drift, and the other is the precision error. The precision error of measurement was calculated using the Kline-McClintock method for error propagation. All of the results have a confidence of 95% in experiments.

The computation was performed to observe the detailed flow field of nano rotors to find inherent reason of the variation of rotor performance and rudder performance induced by the mutual interference between the nano rotor and the aerodynamic rudder. Simulations were carried out at the rotational velocity of 6500 RPM for the nano rotor. And the rotor-to-rudder spacing is 0.5R while the deflection angle of aerodynamic rudder is 30 degree. The flow field of nano rotor without aerodynamic rudder was also simulated as a reference. A short cylinder with diameter of 10 mm was added at the center of rotor to take into account the influence of motor. The computations were performed on HP station with 40 CPU and 64GB memory for one weeks.

5 RESULTS AND DISCUSSION

5.1 *Aerodynamic performance of aerodynamic rudder under interference*

Figure 7 shows the variation of the moment with spacing between rotor and rudder. Generally, the moment of aerodynamic rudder fluctuates with the spacing for all deflection angles. Besides, the amplitude of fluctuation augments with the deflection angle of aerodynamic rudder. And the moment reaches 0.37 N·mm at high deflection angle. The least moment is achieved at a rotor spacing of 0.5 R whereas the highest moment is achieved at a rotor spacing of 1.5 R. When the spacing is lower than 0.5 R, it is found that the moment drops with the increase of spacing. In addition, the moment augments with the increase of spacing from 0.5 R to 1.0 R except for 0.7R. However, it varies little with spacing when the spacing is larger than R. Since the slipstream of nano rotor influences the generation of moment, the distribution of induced velocity of nano rotor determined the moment.

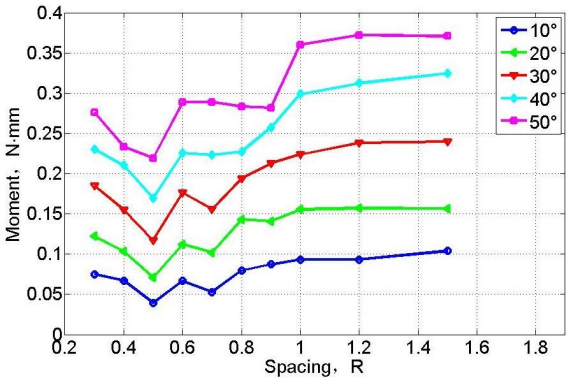


Figure 7: Rudder moment varying with spacing.

Figure 8 shows the variation of rudder moment with defection angle. It is found that the moment of aerodynamic rudder augments with deflection angle nearly linearly. However, the slope of the curve varies with the spacing. The lowest slope was found at the spacing of 0.5 R and the highest slope was found at the spacing of 1.5 R. It shall be pointed that the slope does not change a lot with the spacing.

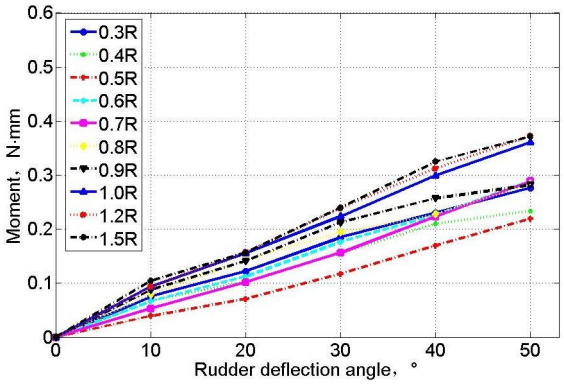


Figure 8: Rudder moment varying with deflection angle.

5.2 *Propulsion performance of nano rotor under interference*

To analyze the propulsion performance of nano rotor under interference clearly, the experimental results obtained at fixing rudder deflection angle or at fixing spacing was presented. The figure 9 show the thrust coefficient of nano rotor varying with spacing at fixing rudder deflection angle of 30 degree. It is found that the thrust coefficient fluctuates with the spacing. In general, the thrust coefficient decreases slightly with spacing. It is illustrated that the aerodynamic rudder will influence the rotor thrust coefficient a little. Figure 10 shows the torque coefficient of nano rotor varying with spacing at fixing rudder deflection angle of 30 degree. Results show that the torque coefficient drops slightly with spacing, which states that the interference effect becomes weak as the rudder leaves away from the nano rotor. In summary, the thrust coefficient and torque coefficient increases slightly with spacing. Figure 11 shows the thrust coefficient of nano rotor varying with rudder deflection angle at fixing spacing of 0.5R. Results show that the thrust coefficient of nano rotor increases slightly with the increases of deflection angle. Since the higher deflection angle enhances the interference effect, the thrust of nano rotor increase as a result. And the torque coefficient of nano rotor varies little with the deflection angle as shown in Figure 12. In summary, the aerodynamic rudder influences the propulsive performance of nano rotor slightly.

5.3 *Flow Field analysis*

Figure 13 and figure14 show the pressure coefficient contour on the press surface and suction surface of rotor with and

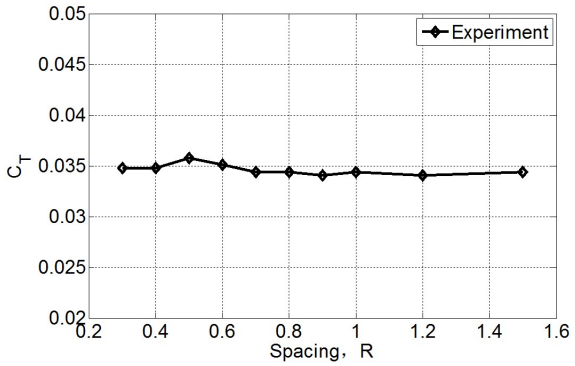


Figure 9: Thrust coefficient of nano rotor varying with spacing at fixing rudder deflection angle of 30 degree.

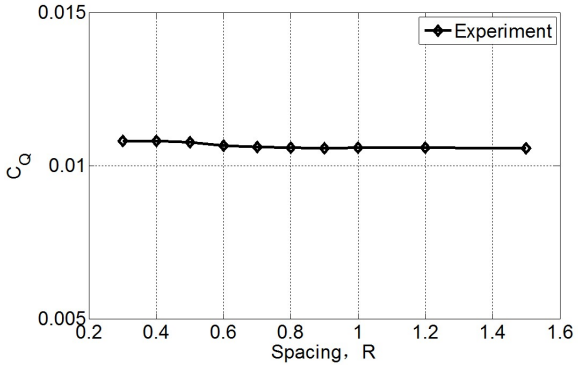


Figure 10: Torque coefficient of nano rotor varying with spacing at fixing rudder deflection angle of 30 degree.

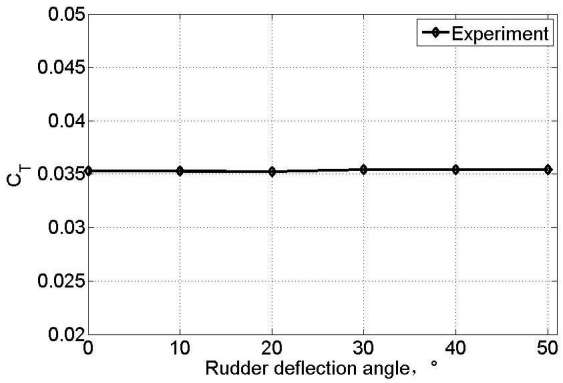


Figure 11: Thrust coefficient of nano rotor varying with rudder deflection angle at fixing spacing of 0.5R.

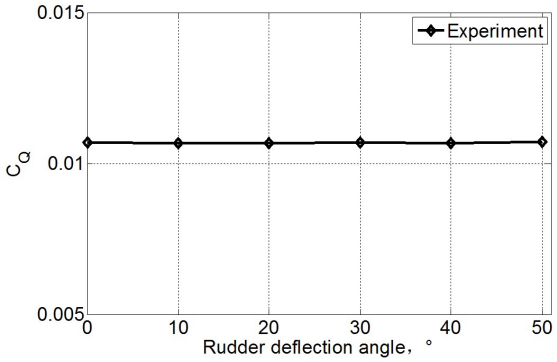
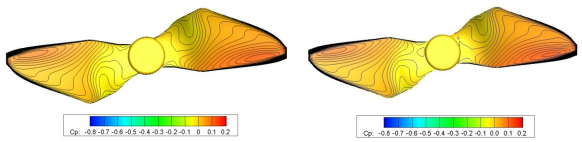


Figure 12: Torque coefficient of nano rotor varying with rudder deflection angle at fixing spacing of 0.5R.

without rudder. The distance between the propeller and rudder is 0.53R and the rudder deflection angle is 30 degrees. Results show that the distribution of pressure coefficient matches well each other for both cases. Figure 15 show the dimensionless velocity magnitude of the flow field with and without rudder. It can be found that the flow field was not disturbed for single rotor and induced velocity increases gradually with the distance from nano rotor at the beginning. Then it will decreases with the distance away from the nano rotor. For the flow field with aerodynamic rudder, it is found that the rudder blocks the flow and the right flow field is different from the left flow field despite of the fact that both rudders deflect to the same direction. It is analyzed that the swirl wake flow of nano rotor causes the phenomena. The variation can also be found from the pressure coefficient contour of aerodynamic rudder as shown in Figure 16. It is clearly found that there are obvious discrepancies in pressure distribution of two rudder surface. The surface of the right rudder presents a larger pressure gradient. It is illustrated that the counter-clockwise rotation of the rotor drives the flow in the downstream rotating in a counterclockwise direction resulting in the different angle of attack between left and right rudder surface. Therefore, the angle of attack of the right rudder surface is higher than the left one. In summary, the aerodynamic rudder has little effect on the pressure coefficient of nano rotor, but nano rotor influences the aerodynamic performance of aerodynamic rudder greatly causing the variation on different rudders.



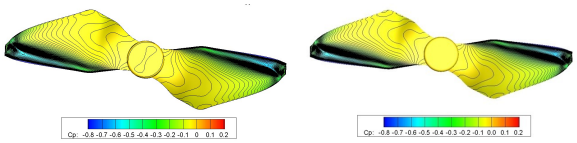
(a) (b)

Figure 13: Contour of pressure coefficient of press surface (a) single rotor, (b) rotor with rudder.

5.4 Comparison of experimental and numerical results

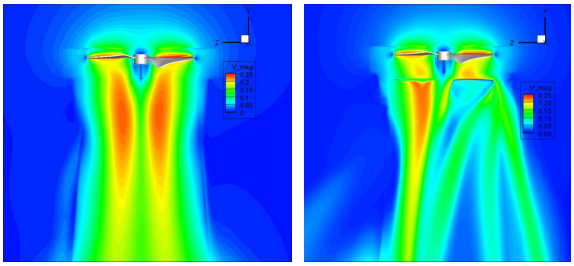
Figure 17 show the comparison of experimental and computational rudder moment at different spacing and rudder deflection angle. It can be concluded that the experimental and

http://www.imavs.org/



(a) (b)

Figure 14: Contour of pressure coefficient of suction surface (a) single rotor, (b) rotor with rudder.



(a) (b)

Figure 15: Dimensionless velocity magnitude contour: (a) single rotor, (b) rotor with rudder.

computational results of rudder moment agree well at different rudder deflection angle. However, at different spacing, the CFD results show that the rudder moment firstly increases with spacing and then become independent with spacing. The experimental results show that the rudder moment firstly decrease with spacing. When the spacing is larger than 0.53R, the rudder moment increases with spacing. The possible reason for this discrepancy is that experiment was disturbed by the stick connecting the two rudders.

6 CONCLUSION

In this paper, the interference effect between the nano rotor and aerodynamic rudder was studied experimentally and computationally. An experimental platform was established. And propulsive performance of nano rotor and aerodynamic performance of aerodynamic rudder were achieved experimentally at different rudder-to-rotor spacing and deflection

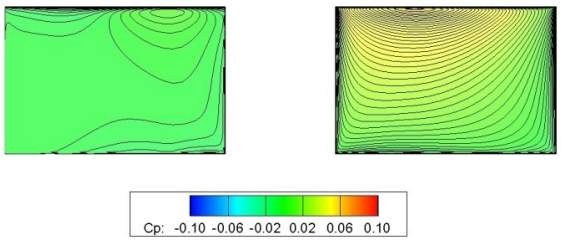
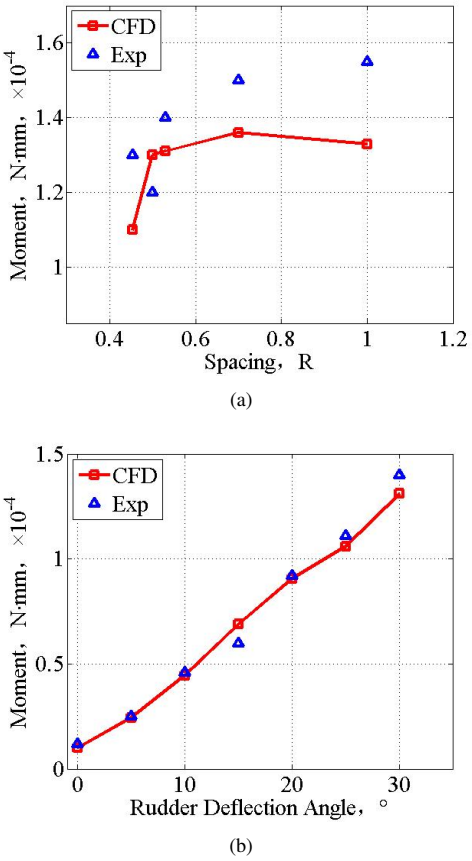


Figure 16: Pressure coefficient contour of aerodynamic rudder.



(a)

(b)

Figure 17: Comparison of experimental and computational rudder moment at (a)different spacing and (b)different rudder deflection angle.

angle of rudder. The disturbed flow field of nano rotor was also analyzed computationally to disclose the flow mechanics of the interaction. It is found that the nano rotor has a great effect on the aerodynamic performance of aerodynamic rudder. The moment of aerodynamic rudder fluctuates with the rotor-to-rudder spacing. And the moment drops with the increase of spacing when the spacing is lower than 0.5 R but it augments with the spacing from 0.5 R to 1.0 R. Analysis reveals that the distribution of induced velocity results in the fluctuation. And the moment of aerodynamic rudder varies with deflection angle linearly. The influence of spacing and rudder deflection angle to the propulsion performance of nano rotor was also studied. It is found that the thrust coefficient and torque coefficient increase slightly with spacing but change little with the deflection angle. Numerical simulation of interference between rotor and aerodynamic rudder showed that aerodynamic rudder blocks the flow field of the nano rotor and the counter-clockwise rotation of the rotor drives the flow in the downstream rotating in a counterclockwise direction resulting

http://www.imavs.org/

in the different angle of attack between left and right rudder surface. In summary, the aerodynamic performance of aerodynamic rudder is influenced greatly by the nano rotor. The moment of aerodynamic rudder varies with spacing and deflection angle. The propulsion performance of nano rotor is influenced slightly by the aerodynamic rudder. In the further work, the different rotational speeds of nano rotor shall be studied. And the visual experiment using PIV will be carried out.

ACKNOWLEDGEMENTS

The authors would like to thank the support of National Natural Science Foundation of China (Grant No. 11302164), Specialized Research Fund for the Doctoral Program of Higher Education of China (20130201120031), and the Fundamental Research Funds for the Central Universities (Grant No. xjj2013031).

REFERENCES

[1] Ronald A. Hess. Handling qualities and flight safety implications of rudder control strategies and systems in transport aircraft. In *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, Providence, Rhode Island, 8 2004. the American Institute of Aeronautics and Astronautics, Inc.

[2] Wenfeng Li Zeren Yang, Xuanshi Meng and Jinsheng Cai. Vortex flow control over a delta wing using leading-edge rudder. In *Aerospace Sciences Meeting*, National Harbor, Maryland, 1 2014. the American Institute of Aeronautics and Astronautics, Inc.

[3] Fratello G, Favier D, and Maresca C. Experimental and numerical study of the propeller/fixed wing interaction. *Journal of Aircraft*, 28(2001-2404), 6 1991.

[4] Moens F and Gardarein P. Numerical simulation of the propeller/wing interactions for transport aircraft. In *Applied Aerodynamics Conference*, pages 2001–2404, Anaheim, 2001. AIAA E-library.

[5] Gamble BJ and Reeder MF. Experimental analysis of propeller-wing interactions for a micro air vehicle. *Journal of Aircraft*, 46(65-73), 1 2009.

[6] Hu Jian, Ma Cheng, and Huang Sheng. Propeller wake flow field numerical analysis of the propeller wake field of numerical analysis. *Journal of Huazhong University of Science and Technology*, 29(20-22), 12 2009.

[7] Wang ZhanZhi and Xiong Ying. Rudder decorate effect on propeller hydrodynamic performance. *Journal of Huazhong University of Science and Technology*, 12(1-26), 12 2008.

[8] Yang Jianmin. Prediction of performance of the rudder with plates behind a propeller. *Journal of SHANGHAI JIAOTONG UNIVERSITY*, 31(137-140), 11 1997.

[9] Duan ZhongZhe and Ma Rong. A certain type of propeller slipstream numerical studies of effects on airfoil aerodynamic performance. *Journal of Aerospace Power*, 26(1776-1781), 8 2011.

[10] Darryll J. Pines and Felipe Bohorquez. Challenges facing future micro-air-vehicle development. *Journal of Aircraft*, 43(290-305), 3 2006.

http://www.imavs.org/

Experimental Investigation of Rotational Control of a Constrained Quadrotor Using Backstepping Method

N.Parhizkar*, A.Naghash†, and M.Naghshineh‡
Amirkabir University of technology

http://www.imavs.org/

ABSTRACT

Considering nonlinear and simple dynamics of quadrotors, it is feasible to implement different types of control methods, so it is an appropriate subject for experimental research. In recent years, many researchers have investigated different aspects of quadrotors. In this study, at first, the utilized quad rotor is introduced and then its dynamics is modeled. Next step is to measure physical parameters of the quadrotor that are needed for simulation. Then, PID and Backstepping controllers are implemented in Simulink and after fixing translational motion of the quadrotor, both controllers are implemented on real model using Labview. Due to having 3DOF and 4 control inputs, the system is over-actuated, so an optimization is done to make the total thrust minimum. Results show that Euler angles are controlled by both PID and Backstepping controllers. Backstepping method has had better results.

1 INTRODUCTION

Many methods have been used for controlling quadrotors. Lyapunov theory has been used in [1],[2] for making sure to have asymptotic stability of system. PD and PID controllers have been used in many projects, for example in [3]. The advantage of PD and PID controllers is their easy implementation. PD and PID help us control a plant adequately without having full information of the plant characteristics or transfer functions. Adaptive techniques are used in [4] due to its good performance in cases of unmodeled dynamic and uncertainty. Classic LQR and SDRE (State-dependent Ricatti Equation) control are applied due to their advantage in presenting input signal from the feedback of the variables,[5],[6]. But the problem with this method is the difficulty of solution of the Ricatti equation. Backstepping Controller is another method that guarantees convergence of the inner variables but its calculations are complicated [7],[8]. Sliding mode controller is applied in much research and it is similar to backstepping method. This controller is robust to uncertainty in parameters and perturbations. The problem with this method is vibrations and chattering [9]. Other methods like dynamic feedback [10],visual feedback [11],Neural Networks [12] are used too but not as frequently as previously mentioned methods.

*M.S. graduate, parhizkar@aut.ac.ir
†Associate Professor, naghash@aut.ac.ir
‡PhD candidate, m.naghshineh@aut.ac.ir

2 AR. DRONE

The used quadrotor in this project is AR. Drone (Figure 1). AR. Drone is equipped with an inertial measurement unit (IMU) that measures pitch, roll, yaw and accelerations along all axes. The vehicle is controlled by sending commands over a Wi-Fi connection[13]. The AR. Drone carries an internal computer with a 468MHz ARM9-processor and 128MB of RAM, running a custom Linux operating system[13].



Figure 1: AR. Drone.

2.1 Gear Ratio

The gear ratio is defined as the input speed relative to the output speed. To obtain the gear ratio for the motor, it is enough to devide the number of teeth of the gear wheel for the motor to the gear wheel connected to the propeller. Two gear wheels have been shown in Figure 2. The gear ratio shown by G is as follows:

$$G = \frac{\text{motor gear wheel's teeth}}{\text{propeller gear wheel's teeth}} = \frac{69}{8} = 8.6 \quad (1)$$



Figure 2: The motor's and the propeller's gear wheels.

http://www.imavs.org/

2.2 *Command to the motors*

AR.Drone has the capability to command its motors one by one directly using a parameter that it is shown by "n". This parameter can be changed from 0 to 500. It should be at least 5 to keep the motor running and the rotor speed will saturate for n=500. For simulation, it is necessary to know the relation between n and fm (the frequency of the motor signal). Different values of n are given to one of the motors and the motor voltage is observed in an oscilloscope. Then frequencies for different values of n are obtained. Figure3 indicates the voltage signal for n=10.

$$\omega_P = \frac{2\pi f_m}{G} \tag{2}$$

f_m is the frequency of the signal of the motor voltage and G is the gear ratio. Results have been shown in Figure 4.

$$(5 \leq n \leq 500) \qquad \omega_P = 0.724n + 125.9 \tag{3}$$

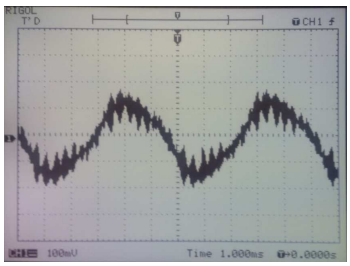


Figure 3: The signal of the motor voltage in n=10.

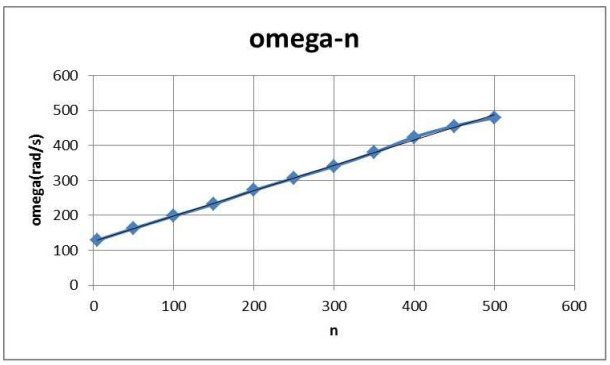


Figure 4: omegap versus different values of n.

3 ROTATIONAL DYNAMIC MODEL OF THE QUADROTOR

Dynamic model of the quadrotor is derived in [14] , [15] and it is just briefly explained. To develop dynamic model, it is assumed:

a. The structure and propellers are supposed to be rigid.

b. The structure is symmetrical and the body fixed frame origin is located on CG. The center of mass and the body fixed frame origin are assumed to coincide.

c. Propeller's thrust and drag are proportional to the square of its rotational speed.

d. Body drag due to motion and rotation is neglected due to low movement speed.

e. Ground effects are neglected and the earth is flat and fixed. Let us consider an inertial frame and a body fixed frame whose origin is in the center of mass of the quadrotor as shown in Figure 5.

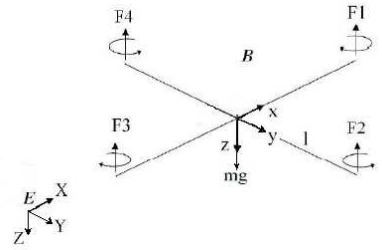


Figure 5: Body and Inertial Frames.

The orientation of the quadrotor is given by 3 Euler angles, namely yaw angle ψ, pitch angle θ and roll angle φ that together form the vector $[\varphi\ \theta\ \psi]^T$. To derive the equations of the rotational motion, Newton-Euler's formula is defined as following:

$$\vec{M} = \left.\frac{d(\vec{H})}{dt}\right)_{\substack{inertial\\frame}} = \left.\frac{d(\vec{H})}{dt}\right)_{\substack{Body\\frame}} + \vec{\omega} \times \vec{H} \tag{4}$$

$\vec{M}$ is the total external torque acting on the vehicle and $\vec{H}$ is the angular momentum made of one generated by body rotation and the other generated by the propeller's angular velocity:

$$\begin{aligned} \vec{H}_{body} &= \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} . \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad , \\ \vec{H}_{blade} &= \begin{bmatrix} 0 \\ 0 \\ J_r \Omega_{blade} \end{bmatrix} \end{aligned} \tag{5}$$

Ω_{blade} is the difference between clockwise and counterclockwise propeller's angular velocity:

$$\Omega_{blade} = \Omega_1 + \Omega_3 - \Omega_2 - \Omega_4 \tag{6}$$

Ω_i , $i = 1, 2, 3, 4$, is the angular velocity of the each propeller. I_x, I_y and I_z are the moments of inertia and because of body symmetry, the products of inertia become zero. The

following expression is used to obtain equation in inertial coordinate system:

$$\vec{M} = \begin{bmatrix} I_x\dot{\omega}_x + \omega_y\omega_z(I_z - I_y) + J_r\Omega\omega_y \\ I_y\dot{\omega}_y + \omega_x\omega_z(I_x - I_z) - J_r\Omega\omega_x \\ I_z\dot{\omega}_z + \omega_x\omega_y(I_y - I_x) \end{bmatrix} \quad (7)$$

$\vec{M}$ is moment's differences caused by angular velocity of the propeller. τ_i is the torque generated in the opposite direction of propeller rotation and is exerted on the propeller by air. For τ_i and T_i (thrust of motors), $i = 1, 2, 3, 4$, we have:

$$\tau_i = d\Omega_i^2 \quad , \quad T_i = b\Omega_i^2 \quad (8)$$

So we have:

$$\vec{M} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} -(T_2 - T_4).L \\ (T_1 - T_3).L \\ (-\tau_1 - \tau_3 + \tau_2 + \tau_4) \end{bmatrix} = \begin{bmatrix} -(T_2 - T_4).L \\ (T_1 - T_3).L \\ \frac{d}{b}(T_2 + T_4 - T_1 - T_3) \end{bmatrix} \quad (9)$$

In the above equation, τ_x, τ_y and τ_z are roll, pitch and yaw torques respectively. L is the distance between the center of body mass and the axis of propeller rotation. Using equations (7) and (9) and rewriting the obtained equation in terms of the derivatives of angular velocities, we have:

$$\begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} \frac{\tau_x}{I_x} - \omega_y\omega_z\frac{(I_z - I_y)}{I_x} - \frac{J_r\Omega\omega_y}{I_x} \\ \frac{\tau_y}{I_y} - \omega_x\omega_z\frac{(I_x - I_z)}{I_y} + \frac{J_r\Omega\omega_x}{I_y} \\ \frac{\tau_z}{I_z} - \omega_x\omega_y\frac{(I_y - I_x)}{I_z} \end{bmatrix} \quad (10)$$

Now it is obligatory to obtain a relation between angular velocities about body axes and rates of Euler angles. To achieve this, the rates are transferred to the body coordinate system and by assuming small Euler angles finally we will have:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi} - \sin(\theta)\dot{\psi} \\ \cos(\phi)\dot{\theta} + \sin(\phi)\cos(\theta)\dot{\psi} \\ -\sin(\phi)\dot{\theta} + \cos(\phi)\cos(\theta)\dot{\psi} \end{bmatrix} \quad (11)$$

Using above equation and equation (10), and neglecting terms consisting 3 and 4 terms products and presenting in terms of Euler angles we will have:

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} a_1.(\dot{\theta}\dot{\psi}) - a_4.(\dot{\theta} + \phi\dot{\psi}).\Omega + b_3\tau_z\theta + b_1\tau_x \\ a_2.(\dot{\phi}\dot{\psi}) + a_5.(\dot{\phi} - \theta\dot{\psi}).\Omega - b_3\tau_z\phi + b_2\tau_y \\ a_3.(\phi\dot{\theta}) + (a_5\Omega\dot{\phi} + b_2\tau_y).\phi + b_3\tau_z \end{bmatrix} \quad (12)$$

Where:

$$\begin{aligned} a_1 &= \left[\tfrac{(I_y - I_z)}{I_x} + 1\right] \quad , \quad a_2 = \left[\tfrac{(I_z - I_x)}{I_y} - 1\right] \\ a_3 &= \left[\tfrac{(I_x - I_y)}{I_z} + 1\right] \quad , \quad a_4 = \tfrac{J_r}{I_x} \\ a_5 &= \tfrac{J_r}{I_y} \quad , \quad b_1 = \tfrac{1}{I_x} \quad , \quad b_2 = \tfrac{1}{I_y} \quad , \quad b_3 = \tfrac{1}{I_z} \end{aligned} \quad (13)$$

Finally to study the system we consider:

$$\begin{cases} x_1(t) = \phi(t) \\ x_2(t) = \dot{\phi}(t) \\ x_3(t) = \theta(t) \\ x_4(t) = \dot{\theta}(t) \\ x_5(t) = \psi(t) \\ x_6(t) = \dot{\psi}(t) \end{cases} , \quad \begin{cases} U_2(t) = \tau_x \\ U_3(t) = \tau_y \\ U_4(t) = \tau_z \\ \delta = \Omega \end{cases} \quad (14)$$

So rewriting equations (12) and (13) using state space notation, we have:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = a_1.(x_4.x_6) - a_4.(x_4 + x_1.x_6).\Omega + \\ \quad + b_3.u_4.x_3 + b_1.u_2 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = a_2.(x_2.x_6) + a_5.(x_2 - x_3.x_6).\Omega - \\ \quad - b_3.u_4.x_1 + b_2.u_3 \\ \dot{x}_5 = x_6 \\ \dot{x}_6 = a_3.(x_2.x_4) + a_5.(x_1.x_2).\Omega + \\ \quad + b_2.u_3.x_1 + b_3.u_4 \end{cases} \quad (15)$$

4 PHYSICAL PARAMETERS OF THE QUADROTOR

Table 1 presents the physical parameters of AR. Drone needed for simulation that have been measured.

Symbol	Definition	Value	Unit
m	Quadrotor mass	335	gr
L	Arm length	18	Cm
I_z	Moment of inertia about z axis	4.7×10^{-3}	Kg.m^2
I_x	Moment of inertia about x axis	1.8×10^{-3}	Kg.m^2
I_y	Moment of inertia about z axis	1.8×10^{-3}	Kg.m^2
b	Propeller Thrust factor	5.7231×10^{-6}	N.s^2
d	Propeller Drag factor	1.7169×10^{-7}	N.m.s^2
J_r	Rotor inertia	1.85×10^{-5}	Kg.m^2

Table 1: Physical parameters of the quadrotor.

Moment of Inertia: There is a common method to measure moment of inertia. This method is to make body oscillate. In this method, by measuring the period of the oscillating body, the moment of inertia about different axis is obtained.

Propeller thrust factor: Thrust is measured for different values of *n* and thrust in terms of *n* is estimated as follows:

$$T = a_1 n^2 + a_2 n + a_3 \quad (5 \leq n \leq 500)$$
$$a_1 = 3 \times 10^{-6} \, , \, a_2 = 0.001 \, , \, a_3 = 0.0907 \quad (16)$$

http://www.imavs.org/

http://www.imavs.org/

Combining (3) and (16) we have:

$$T = 5.7231 \times 10^{-6}\omega_p^2 \tag{17}$$

Then we have:

$$b = 5.7231 \times 10^{-6} \tag{18}$$

Propeller Drag factor: Produced moment is measured for different values of *n* and similar to the previous section, we will have:

$$d = 1.7169 \times 10^{-7} \tag{19}$$

5 ROTATIONAL CONTROL OF THE CONSTRAINED QUADROTOR

Quadrotor is a 6 DOF system and so it is an underactuated system but our aim is to control quadrotor rotations. The system is 3DOF and so it is an overactuated system. To study an overactuated system, it is possible to optimize a cost function which is U_1in this study, U_1 is total thrust:

$$U_1 = T_1 + T_2 + T_3 + T_4 \tag{20}$$

The optimal value for U_1 is the least possible value to keep motors running. Using equations (9), (14) and (20) we have:

$$T_1 = \tfrac{1}{4}U_1 + \tfrac{1}{2l}U_3 - \tfrac{b}{4d}U_4 \quad , \quad T_2 = \tfrac{1}{4}U_1 - \tfrac{1}{2l}U_2 + \tfrac{b}{4d}U_4$$

$$T_3 = \tfrac{1}{4}U_1 - \tfrac{1}{2l}U_3 - \tfrac{b}{4d}U_4 \quad , \quad T_4 = \tfrac{1}{4}U_1 + \tfrac{1}{2l}U_2 + \tfrac{b}{4d}U_4$$

$$\Rightarrow \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -l & 0 & l \\ l & 0 & -l & 0 \\ -\frac{d}{b} & \frac{d}{b} & -\frac{d}{b} & \frac{d}{b} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{21}$$

In the following section, total thrust will be optimized using equations in (21).

5.1 Optimization of total thrust

Let's look at equations in (21). As it is observed each of T_1, T_2, T_3 and T_4 consists of 3 terms, a common $^1/_4U_1$ and two other terms, which are named $\bar{U}_i$ $(i = 1,\ 2,\ 3,\ 4)$:

$$\begin{aligned} \bar{U}_1 &= +\tfrac{1}{2l}U_3 - \tfrac{b}{4d}U_4 \quad , \quad \bar{U}_2 = -\tfrac{1}{2l}U_2 + \tfrac{b}{4d}U_4 \\ \bar{U}_3 &= -\tfrac{1}{2l}U_3 - \tfrac{b}{4d}U_4 \quad , \quad \bar{U}_4 = +\tfrac{1}{2l}U_2 + \tfrac{b}{4d}U_4 \end{aligned} \tag{22}$$

Now, one can prove that all values of $\bar{U}_i$ can not be positive values simultaneously. Let's assume all values for $\bar{U}_i$ are positive: $\bar{U}_i > 0$ Then simply we will have:

$$\begin{aligned} \bar{U}_1 > 0 \ , \ \bar{U}_2 > 0 \quad &\Rightarrow \ \mathrm{U}_2 < U_3 \\ \bar{U}_3 > 0 \ , \ \bar{U}_4 > 0 \quad &\Rightarrow \ \mathrm{U}_3 < U_2 \end{aligned}$$

So it is clear that at least for one $\bar{U}_i$ we have: $\bar{U}_i < 0$ Now, considering the mentioned point we can determine min(U_1) as follows:

$$U_1 = 4\left|\min \bar{U}_i\right| + T_{\min} \tag{23}$$

$T_{\min}$is the minimum thrust that can be obtained by replacing *n=5* into equation (16):

$$T_{\min} = 0.096N$$

5.2 Propeller-motor model

Combining equations (3) and (17), we will have:

$$n = 1.3812\sqrt{\frac{T}{5.7231 \times 10^{-6}}} - 173.895 \tag{24}$$

This equation gives required value of *n* to control the quadrotor.

5.3 PID Controller

The output of a PID controller, is the control input to the system and in the time-domain it is as follows:

$$U = K_P e + K_d\frac{de}{dt} + K_i\int edt \tag{25}$$

Where e is the error signal. Then for Euler angles we will have:

$$\begin{aligned} U_2(t) &= k_{p\phi}(\phi_d - \phi) + k_{d\phi}(\dot{\phi}_d - \dot{\phi}) + k_{i\phi}\textstyle\int(\phi_d - \phi)dt \\ U_3(t) &= k_{p\theta}(\theta_d - \theta) + k_{d\theta}(\dot{\theta}_d - \dot{\theta}) + k_{i\theta}\textstyle\int(\theta_d - \theta)dt \\ U_4(t) &= k_{p\psi}(\psi_d - \psi) + k_{d\psi}(\dot{\psi}_d - \dot{\psi}) + k_{i\psi}\textstyle\int(\psi_d - \psi)dt \end{aligned} \tag{26}$$

PID controller coefficients are given in Table 2. Both theorical and experimental coefficients have been obtained by try and error.

	Coefficient	Theorical	Experimental
Roll Control	$k_{p\phi}$	0.08	0.1
	$k_{d\phi}$	0.02	0.05
	$k_{i\phi}$	0.09	0.06
Pitch Control	$k_{p\theta}$	0.08	0.1
	$k_{d\theta}$	0.03	0.0.5
	$k_{i\theta}$	0.07	0.06
Yaw Control	$k_{p\psi}$	0.07	0.09
	$k_{d\psi}$	0.03	0.05
	$k_{i\psi}$	0.09	0.05

Table 2: PID Coefficients.

5.4 Backstepping Controller

Backstepping method is presented in [16] completely and we briefly explain it then we will use it for the quadrotor. We consider the special case of integrator backstepping. Consider the system:

$$\begin{aligned} \dot{\eta} &= f(\eta) + G(\eta)\zeta \\ \dot{\zeta} &= f_a(\eta,\zeta) + G_a(\eta,\zeta)u \end{aligned} \tag{27}$$

Where $f_a$and $G_a$are smooth. The state-space equation (15) is considered and vector of variables and input are defined as following:

$$\eta = \begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix} \ , \ \zeta = \begin{bmatrix} x_2 \\ x_4 \\ x_6 \end{bmatrix} \ , u = \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{28}$$

http://www.imavs.org/

Backstepping controller design is a two-step method. In first step, it is assumed that ζ is the input of the first equation of equations (27). The first equation is stabilized using $\zeta = \Phi(\eta)$ in a way that it will be possible to find a Lyapunov function for the first equation. $\Phi(\eta)$ is considered as following:

$$\Phi(\eta) = \begin{bmatrix} -k_1 & 0 & 0 \\ 0 & -k_3 & 0 \\ 0 & 0 & -k_5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_5 \end{bmatrix} = \begin{bmatrix} -k_1 x_1 \\ -k_3 x_3 \\ -k_5 x_5 \end{bmatrix}$$
$$k_1 \ , k_3 \ , \ k_5 \ > \ 0 \tag{29}$$

In this step, Lyapunov function $V(\eta)$ is considered as following:

$$V(\eta) = \frac{1}{2}\eta^T \eta = \frac{1}{2}(x_1^2 + x_3^2 + x_5^2) \tag{30}$$

In second step, for Lyapunov function, V_a, we consider:

$$V_a = V(\eta) + \frac{1}{2}\left[\zeta - \Phi(\eta)\right]^T \left[\zeta - \Phi(\eta)\right] \tag{31}$$

Now *u* is considered such that the derivative of the Lyapunov function to be negative and to stabilize the second equation. Considering *u* as following:

$$u = G_a^{-1}\left[\frac{\partial \Phi}{\partial \eta}(f + G\zeta) - (\frac{\partial V}{\partial \eta}G)^T - f_a - K(\zeta - \Phi)\right] \tag{32}$$

Where:

$$K = \begin{bmatrix} k_2 & 0 & 0 \\ 0 & k_4 & 0 \\ 0 & 0 & k_6 \end{bmatrix}$$
$$k_2 \ , k_4 \ , k_6 > \ 0 \tag{33}$$

And substituting related functions in (32) we have:

$$u = \begin{bmatrix} b_1 & 0 & b_3 x_3 \\ 0 & b_2 & -b_3 x_1 \\ 0 & b_2 x_1 & b_3 \end{bmatrix}^{-1} \begin{bmatrix} (-k_1 x_2 - k_2 x_2 - k_1 k_2 x_1 - \\ x_1 - a_1 x_4 x_6 + \\ a_4.(x_4 + x_1 x_6).\Omega_r) \\ \\ (-k_3 x_4 - k_4 x_4 - k_3 k_4 x_3 - \\ x_3 - a_2 x_2 x_6 - \\ a_5.(x_2 - x_3 x_6).\Omega_r) \\ \\ (-k_5 x_6 - k_6 x_6 - k_5 k_6 x_5 - \\ x_5 - a_3 x_2 x_4 \\ -a_5 x_1 x_2.\Omega_r) \end{bmatrix} \tag{34}$$

To implement Backstepping controller on real model, let's assume that ω_x, ω_y and ω_z are equal to $\dot{\varphi}$, $\dot{\psi}$ and $\dot{\theta}$. If Euler angles are small, this assumption will be valid. So, equation (15) will change to the following equations in state-space form:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = a_1.(x_4.x_6) - a_4.x_4.\Omega + b_1.U_2 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = a_2.(x_2.x_6) + a_5.x_2.\Omega + b_2.U_3 \\ \dot{x}_5 = x_6 \\ \dot{x}_6 = a_3.(x_2.x_4) + b_3.U_4 \end{cases} \tag{35}$$

Where:

$$\begin{array}{lll} a_1 = \frac{(I_y - I_z)}{I_x} & , & a_2 = \frac{(I_z - I_x)}{I_y} \\ a_3 = \frac{(I_x - I_y)}{I_z} & , & a_4 = \frac{J_r}{I_x} \\ a_5 = \frac{J_r}{I_y} & & \end{array} \tag{36}$$

Similar to the previous state:

$$u = \begin{bmatrix} \frac{1}{b_1} & 0 & 0 \\ 0 & \frac{1}{b_2} & 0 \\ 0 & 0 & \frac{1}{b_3} \end{bmatrix} \begin{bmatrix} (-k_1 x_2 - k_2 x_2 - k_1 k_2 x_1 - \\ x_1 - a_1 x_4 x_6 + a_4.x_4.\Omega_r) \\ \\ (-k_3 x_4 - k_4 x_4 - k_3 k_4 x_3 - \\ x_3 - a_2 x_2 x_6 - a_5.x_2.\Omega_r) \\ \\ (-k_5 x_6 - k_6 x_6 - k_5 k_6 x_5 - \\ x_5 - a_3 x_2 x_4) \end{bmatrix} \tag{37}$$

Equation (34) is used in simulation and equation (37) is used in the experimental method. values of *ki , i=1 to 6*, are obtained using try and error in both simulation and experimental method. These coefficients are given in Table 3.

Parameters	Simulation	Experimental
$\mathbf{k}_1$	14	10
$\mathbf{k}_2$	5	3
$\mathbf{k}_3$	10	10
$\mathbf{k}_4$	5	3
$\mathbf{k}_5$	10	6
$\mathbf{k}_6$	4	2

Table 3: Backstepping Coefficients.

6 SIMULATION RESULTS

In the following figures, Euler angles controlled by PID and backstepping controllers are shown.

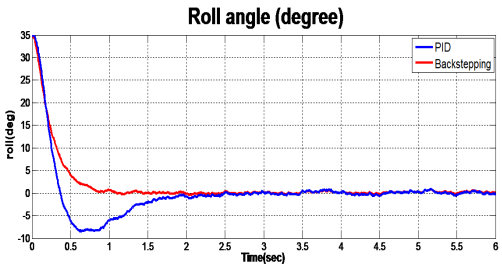


Figure 6: Roll controlled by PID and Backstepping

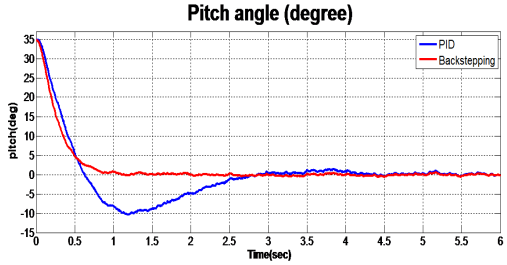


Figure 7: Pitch controlled by PID and Backstepping

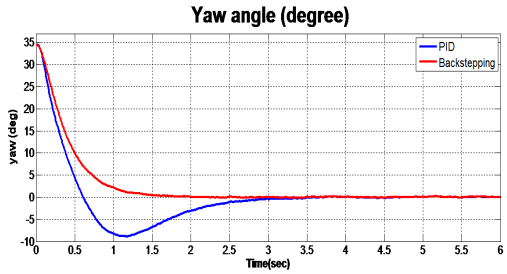


Figure 8: Yaw controlled by PID and Backstepping

As it is seen in the above figures backstepping controller has regulated angles to zero about 1.5 to 2 seconds faster than PID controller. Following figures indicate the input signals to the motors and as it is seen maximum values for n in PID is more than in backstepping.

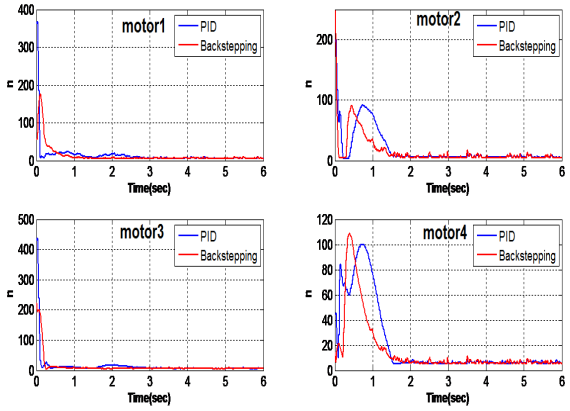


Figure 9: The input signals to the motors in PID and Backstepping Controllers

7 EXPERIMENTAL RESULTS FOR PID AND BACKSTEPPING

Following figures indicate experimental results for PID and Backstepping controllers. In each figure, one channel has been excited by tapping. As it is seen in experimental results, by exciting one channel the other channels are excited too.

PID:

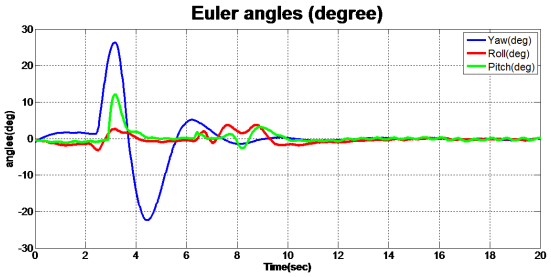


Figure 10: The Euler angles controlled by PID when Yaw is excited

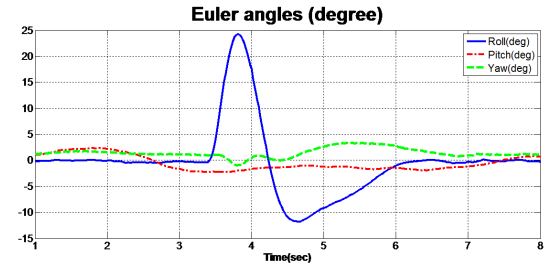


Figure 11: The Euler angles controlled by PID when Roll is excited

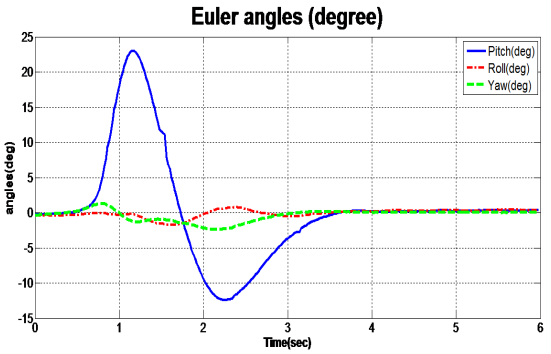


Figure 12: The Euler angles controlled by PID when Pitch is excited

Backstepping:

http://www.imavs.org/

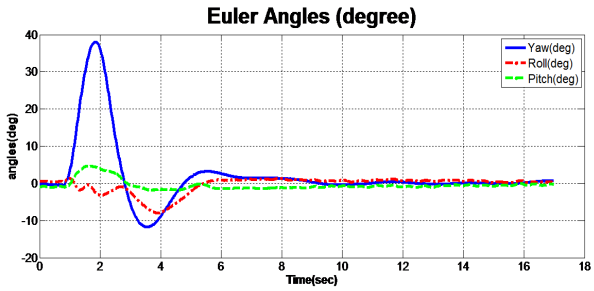


Figure 13: The Euler angles controlled by Backstepping when Yaw is excited

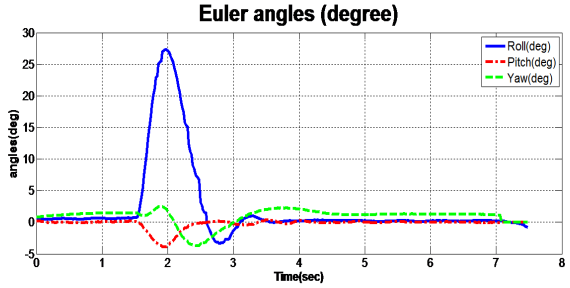


Figure 14: The Euler angles controlled by Backstepping when Roll is excited

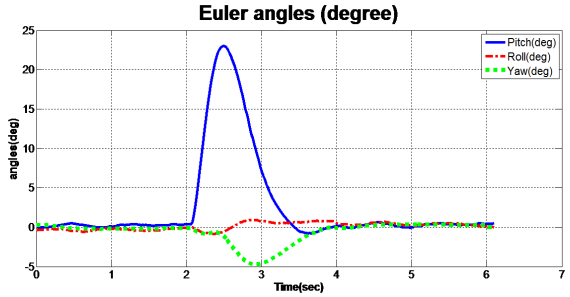


Figure 15: The Euler angles controlled by Backstepping when Pitch is excited

In Figures 12 and 15, Pitch is excited 23 degrees, as it is seen it has taken about 2 seconds in backstepping to regulate but it is about 2.7 seconds in PID. In two other angles Backstepping has controlled bigger disturbances in shorter time and this shows Backstepping acted faster than PID.

8 CONCLUSION

Rotational model of the quadrotor was simulated and some physical parameters are measured by methods that are explained briefly. Then backstepping controller was implemented on simulated model and the results were compared with those of PID. Then both control methods were implemented on real model. Results have shown that Backstepping acted faster than PID. However PID was easier to implement to control without having full information of the plant.

REFERENCES

[1] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and Control of an Indoor Micro Quadrotor," *IEEE Int. Conf. Robot. Autom.* 2004. Proceedings. ICRA '04. 2004, vol. 5, April, 2004.

[2] P. Castillo, a Dzul, and R. Lozano, "Real-time Stabilisation and Tracking of a Four Rotor Mini-rotorcraft" *IEEE Trans. Control Syst. Technol.*, 2004.

[3] A. Tayebi and S. McGilvray, "Attitude Stabilization of a VTOL Quadrotor Aircraft," *IEEE Trans. Control Syst. Technol.*, vol. 14, no. 3, 2006.

[4] Y. Morel and A. Leonessa, "Direct Adaptive Tracking Control of Quadrotor Aerial Vehicles," *Dyn. Syst. Control. Parts A B*, 2006.

[5] A. O. Kivrak, "Design of Control Systems for a Quadrotor," A Master's Thesis in Mechatronics Engineering, Atilim University, December, 2006.

[6] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, M. Candidate, J. S. Jang, and C. J. Tomlin, "The Stanford Testbed of Autonomous Rotorcraft For Multi Agent Control System Hardware Full System,", Stanford University, 2004.

[7] T. Madani and A. Benallegue, "Backstepping Control for a Quadrotor Helicopter," *2006 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2006.

[8] A. Soumelidis, P. Gspr, G. Regula, and B. Lantos, "Control of an Experimental Mini Quad-rotor UAV," *2008 Mediterr. Conf. Control Autom. - Conf. Proceedings, MED'08*, 2008.

[9] M. Guisser and H. Medromi, "A High Gain Observer and Sliding Mode Controller for an Autonomous Quadrotor Helicopter," *Int. J. Intell. Control Syst.*, vol. 14, 2009.

[10] A. Mokhtari and A. Benallegue, "Dynamic Feedback Controller of Euler Angles and Wind Parameters Estimation for a Quadrotor Unmanned Aerial Vehicle," *IEEE Int. Conf. Robot. Autom. 2004. Proceedings. ICRA '04*. 2004, vol. 3, 2004.

[11] N. Guenard, T. Hamel, and R. Mahony, "A Practical Visual Servo Control for an Unmanned Aerial Vehicle," vol. 24, no. 2, 2008.

[12] H. Voos, "Nonlinear and Neural Network-based Control of a Small Four-Rotor Aerial Robot," *Int. Conf. Adv. Intell. mechatronics*, 2007.

[13] N. Dijkshoorn, "Simultaneous Localization and Mapping with the AR . Drone," , *Master's Thesis in Artifial Intelligence, Univ. Van Amsterdam*, 2012.

[14] A. Naghash, M. Naghshineh, and A. Honari, "Minimum Time Trajectory Optimization for Flying a Quadrotor in an 8 - shaped Path," , *International Micro Air Vehicle Conference*, September, 2013.

[15] S. N. Ghazbi, A. L. I. Akbar, and M. Reza, "Quadrotor: Full Dynamic Modeling , Nonlinear Simulation and Control of Attitudes Degrees of Freedom and the Movement," vol. 1,

http://www.imavs.org/

no. 2, 2014.
[16] H. Khalil, "Nonlinear Systems.", Prentice Hall, 2002.

http://www.imavs.org/

Experimental Investigation of Wing Flexibility on Force Generation of a Hovering Flapping Wing Micro Air Vehicle with Double Wing Clap-and-Fling Effects

Q.V. Nguyen*, W.L. Chan, and M. Debiasi
National University of Singapore, 5A Engineering Drive 1, Singapore 117411, Singapore

http://www.imavs.org/

ABSTRACT

Experimental investigation of wing flexibility on vertical thrust generation and power consumption of a hovering flapping wing micro air vehicle (FW-MAV), namely FlowerFly, weighing 14.5 g with a 3 g onboard battery and having four wings with double wing clap-and-fling effects in hovering condition, was conducted for several wing configurations with the same shape, area, and weight. A data acquisition system was setup to simultaneously record forces, power consumption, and wing motions at various flapping frequencies. The forces and power consumption were measured with a loadcell and a custom-made shunt circuit, respectively, and the wing motion was captured by a high-speed camera. The results show a phase delay of the wing tip displacement observed for wings with high flexible leading edge at high frequency, resulting in less vertical thrust produced when compared with the wings with less leading edge flexibility at the same frequency. Positive wing camber was observed during wing flapping by arranging the wing supporting ribs. Comparison of thrust-to-power ratio between the wing configurations was undertaken to figure out a wing configuration for high vertical thrust production but less power consumption.

1 INTRODUCTION

Vertical Takeoff and Landing (VTOL), agile maneuvering, and hovering flight are extremely desirable features of flying insects in development of Flapping-Wing Micro Air Vehicles (FW-MAVs), which enable an insect-inspired FW-MAV to retake off and fly in confined spaces for indoor surveillance. Moreover, FW-MAVs have been proven to be more efficient at low Reynolds number regime (typically from 10^3 to 10^5 [1]) under unsteady aerodynamics when their size becomes less than 15 cm. This is because FW-MAVs benefit from unsteady aerodynamic mechanisms at low Reynolds number: wing leading vortex and delayed stall [2], wing clap-and-fling [3], and wing rotation and wake capture [4, 5]. Operation-wise, FW-MAVs flap their wings at relatively low frequencies and fly at low forward speed or hovering, resulting in producing less noise, and less dangerous than their counter parts of fixed wings flying at high forward speed and rotary wings operating at high motor or propeller speed. However, the payload ability of FW-MAVs is very limited in sub-10 g due to their nature of small size and light weight. Therefore, there are a number of experimental reports looking into changing the wing designs of various FW-MAVs, such as changing materials of wing membrane, wing aspect ratio, wing supporting veins for aerodynamic force enhancement [6, 7, 8]. However, and the wing mass was not kept the same for all cases of wing configurations.



Figure 1: The FlowerFly with onboard camera and control system for pitch-, yaw-, and roll-control controlled by three servos.

As a continuing effort to increase the payload ability for our FlowerFly [9], see Figure 1, for more system integration on the next version, we have improved the aerodynamic force generation ability of the FlowerFly by experimentally inves-

*Email address(es): tslnqv@nus.edu.sg

http://www.imavs.org/

tigating various wing configurations with different flexibility in both spanwise and chordwise. The aim is to find a set of wings that can generate high vertical thrust, but consumes low power. In order to avoid effect of wing shape, wing aspect ratio, and wing inertia difference between wing configurations, the wing shape, area, and wing mass are kept the same for all cases. Experimental investigation was conducted for five different wing configurations to acquire vertical thrusts, power consumptions, and flapping wing motions. All acquired data was synchronized with a trigger. Wing efficiency was characterized by thrust-to-power ratio, which is defined as the ratio of the vertical thrust to the input power or power consumption. In addition, the design of gearbox using one motor to simultaneously drive four wings and create double wing clap-and-fling is also described.

2 MATERIALS AND METHODS

2.1 *Flapping mechanism and gearbox*

Flapping mechanism is one of the challenging parts in the design. It converts the rotary motion of the motor or crank into flapping motion of wings. In this design, we used a combination of crank-slider mechanism and linkage mechanism, see Figure 2. The rotary motion of the crank is transformed into the linear motion of the slider engaging in the slot through the connecting link, and then the linear motion of the slider is transformed into the flapping motion of the coupler to where the wings are attached.

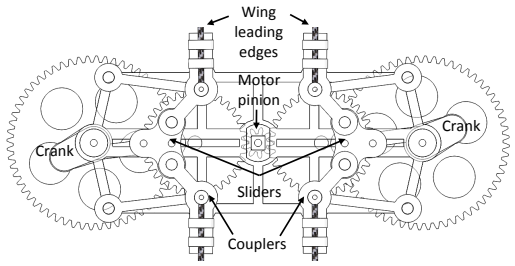


Figure 2: Gearbox with double clap-and-fling mechanisms.

Figure 3 shows a comparison between a sinusoidal function and wing flapping angle produced by the designed gearbox for one flapping cycle. The flapping angle is close to the sinusoidal function; that help to avoid high angular acceleration or wing inertia during flapping. We have invented a gearbox with a gear ratio of 1 to 20 [9] to synchronously drive four wings and create double wing clap-and-fling effects during one flapping cycle. The gearbox combines two modules driven by a brushless motor (AP02, Kv = 7000, hobbyking.com) controlled by an electronic speed controller (Mi-3A ESC module, hobbyking.com), see Figure 2. Each module consists of a crank-slider and linkage (two couplers and two output links). There are wing holders installed on the coupler for wing attachment. The flapping angle of each coupler or wing is designed to be 90°, correspondingly, the four

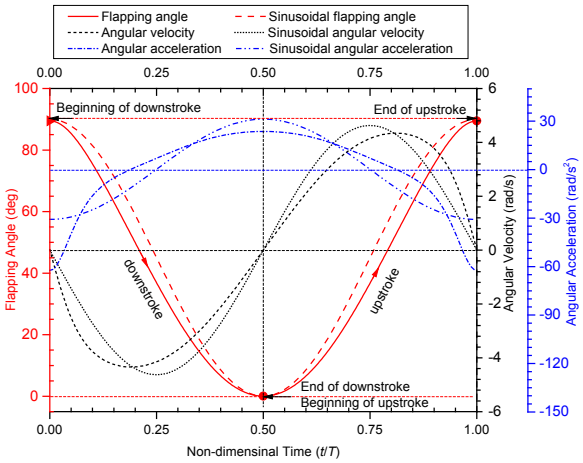


Figure 3: Flapping angles, angular velocities, and angular accelerations of the flapping mechanism (red line for flapping angle, black line for angular velocity, and blue line for angular acceleration. The sinusoidal functions are only for reference.)

wings sweep a flapping angle of 360° for a flapping cycle. The double wing clap-and-fling effects, see Figure 4, can be created at the end of each half flapping stroke: downstroke and upstroke. The large flapping angle and wing clap-and-fling effects are expected to produce high thrust.

2.2 *Wings*

We use lightweight materials such as thin carbon rod (0.6 mm rod and 0.7 mm tube for leading edge and 0.3 - 0.5 mm rods for wing ribs or supporting veins) and thin Mylar film (5 μm) for wing fabrication. The detail fabrication process can be found in reference [10]. In order to investigate the effect of wing flexibility on vertical thrust and power consumption while avoiding wing inertia difference between the wings, various wings with the same shape, area, wing membrane thickness (5 μm Mylar film), and weight, but different wing leading edge diameters (0.6 mm carbon rod, and 0.7 mm carbon tube) and supporting veins (0.5 mm and 0.3 mm carbon rods) were fabricated and used, see Table 1. The wing *AR* of an individual wing is 1.68; the wing length (root to tip) was 110 mm, with maximum and mean chord lengths of 66.7 mm and 65.3 mm, respectively, and the surface area of each wing is 7184 mm^2. Weight difference between these wings is only 3.6%, thus the effect of wing inertia difference between the wings can be assumed the same for all wings and neglected in comparison between these wings even though the wing mass distribution along the wing span is slightly different. Figure 5 shows the wing design for five cases. The wings can be divided into two wing groups; Group#1 includes Wing#1 and Wing#2 with the same leading edge stiffness (0.6 mm carbon rod) and slight difference in chordwise stiffness, see Figure 5, and Group#2 includes Wing#3, Wing#4, and Wing#5 with

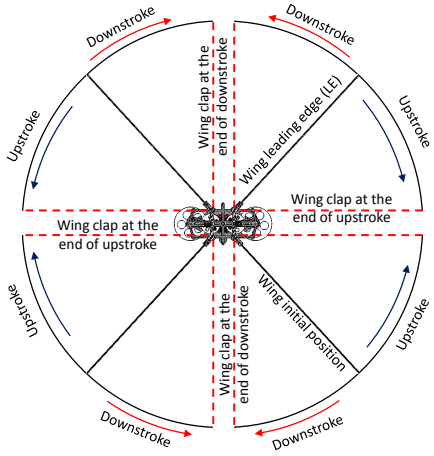


Figure 4: Flapping strokes of the gearbox and double wing clap-and-fling at the end of downstroke and upstroke.

the same leading edge stiffness (0.7 mm carbon tube) and different chordwise stiffness.

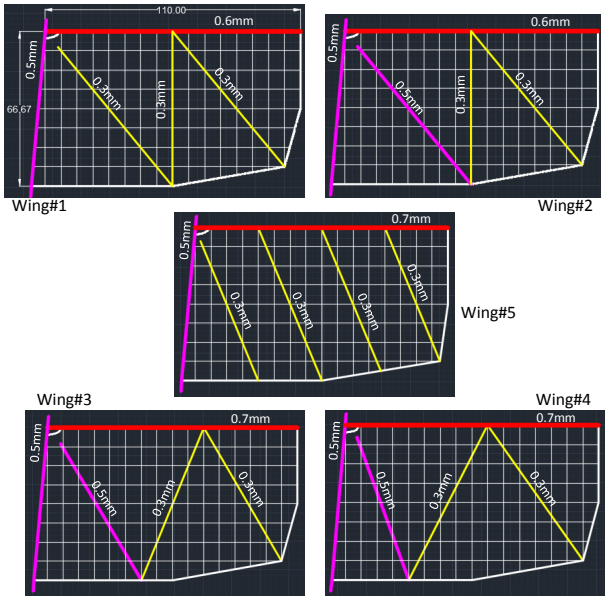


Figure 5: Fabricated wings made of Mylar film and carbon rods.

2.3 *Experimental apparatus*

The experimental apparatus consists of the FW-MAV, FlowerFly, without tail equipped with a Neodymium magnet placed in the gearbox, a radio receiver (DTRx31d, 2.4 GHz, Deltang, UK), and controlled by a transmitter (DX7, 2.4 GHz DSM2, Spektrum, USA), a test jig for mounting the FW-MAV, a 3-axis force/torque loadcell (Nano 17 Titanium, ATI Industrial Automation, USA) for force measurement, a

#	Leading edge (mm)	N. of veins	Span (mm)	Area (mm^2)	Mass (g) (difference)
1	0.6	3	110	7184	0.28*
2	0.6	3	110	7184	0.29 (3.6%)
3	0.7	3	110	7184	0.29 (3.6%)
4	0.7	3	110	7184	0.29 (3.6%)
5	0.7	4	110	7184	0.28*

Table 1: Parameters of wings, *mass reference.

custom-made shunt circuit for current and voltage measurement and a tachometer built in house with Hall Effect sensor for frequency readings, a high speed camera (Phantom Miro M320S, LaVision, Germany) for flapping wing motion capture, a power supply (Tenma 72-8350, China), a data acquisition card (DAQ card BNC-2110, National Instrument, USA) for acquiring data, a PC for data reading and recording, and an in-house built trigger for data synchronization. The experimental setup was shown in Figure 6.

2.4 *Measurement of force, power, and wing motion*

All the flapping tests were conducted in still air in hovering condition from 9 Hz onward the maximum frequency that the FW-MAV can operate at full throttle level. Lower frequencies than 9 Hz were refrained from experiment due to low force expected. Data acquisition for each wing configuration was repeated five times for each flapping frequency. Since the two pairs of wings flap symmetrically in opposite phase, see Figure 4, the forces in horizontal plane or horizontal forces acting on each wing are canceled by each other. Thus, only force in vertical direction or vertical thrust can be acquired by the loadcell.

The FW-MAV was directly mounted to the loadcell, and then firmly installed into the test jig in vertical configuration; this is similar to the free hovering flight condition of the FlowerFly. The power supply maintained the applied voltage of 3.7 V (equivalent to a single cell LiPo battery) to the motor. The transmitter generated the PWM (Pulse Width Modulation) command and transmitted it to the receiver for controlling the motor speed or flapping frequency of the FW-MAV through the ESC. The flapping frequency acquired by the Hall Effect sensor and tachometer was sent to the PC for readings. Data of force, power in terms of voltage and current, and high speed images of wing motion were acquired at the same time with the trigger. When the trigger was turned on the Hall Effect sensor picked up the magnetic signal from the Neodymium magnet and transmitted an electrical pulse to simultaneously activate the high speed camera, the loadcell, and the shunt circuit for data recording. Thus, the images recorded from the camera were synchronized with the data of

http://www.imavs.org/

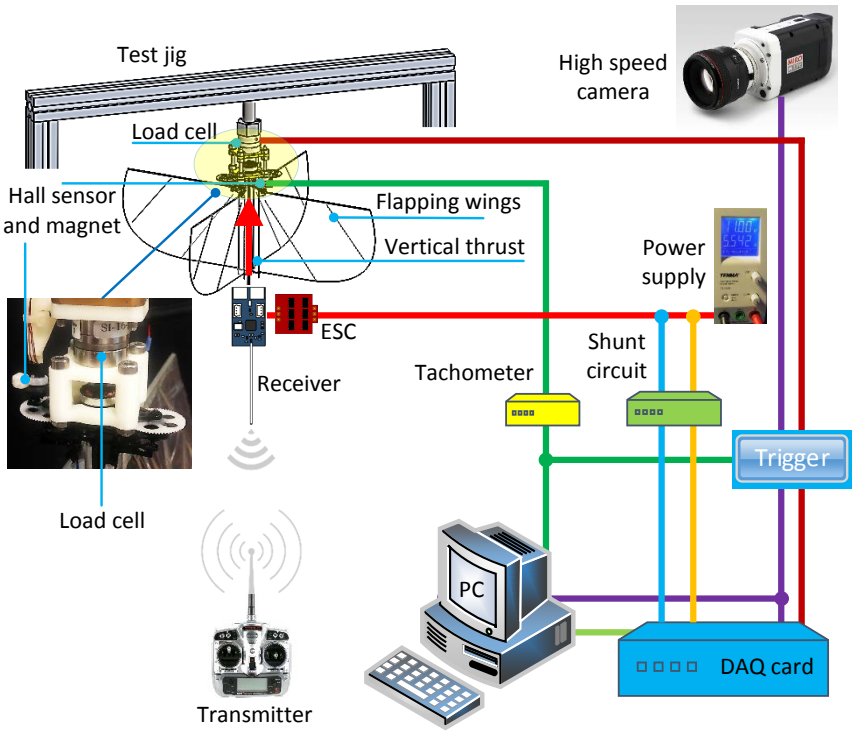


Figure 6: Experimental apparatus and set-up for measurement and synchronization of forces, power, and wing motion.

force and power consumption. The sampling rate of force, power, and wing motion was set at the same frequency of 5 kHz to avoid time shift of the recorded data.

2.5 *Data process*

Contaminated noise due the vibration of the flapping-wing system and structural vibration of the test jig was filtered out by a low pass filter with a cut-off frequency of 3 times greater than the flapping frequency as recommended by the reference [11]. 20 flapping cycles was taken from each measurement (out of five), and then averaged to represent the cycle-averaged value of each measurement. Afterward, we calculated the statistical mean value of five cycle-averaged values to represent the cycle-averaged force and power consumption of the FW-MAV. The high speed images of wing motion were used to examine the wing deformation in terms of spanwise and chordwise deformation as well as interpret the vertical force and power consumption histories.

3 RESULT AND DISCUSSIONS

3.1 *Time-dependent vertical thrust*

The typical instantaneous vertical thrust, power consumption, and flapping angle synchronized with each other are plotted in Figure 7 with respect to (w.r.t) non-dimensional time for one flapping cycle. The instantaneous vertical thrust displays two peaks with different magnitude and two troughs with almost the same magnitude of nearly zero. From the high speed camera images, the first peak and second peak are found to occur at the end of wing fling during downstroke and upstroke, respectively, while the effects of wing clap are not clearly seen. This phenomenon is due to the passive wing rotation mechanism used in the FW-MAV [9], the wings cannot actively rotate to push the air down when the two wing leading edge approach each other. The two troughs are almost zero and found to occur at the beginning of downstroke and upstroke, respectively, where the wings are almost aligned in vertical direction, i.e., angle of attack of the flapping wing is almost 90^0. Similarly, the power consumption history also displays two peaks and two troughs in one flapping cycle with slightly phase shift compared to the vertical thrust history at the first peak and second peak, respectively. It also can be seen that the FW-MAV produces asymmetrical vertical thrust peaks during downstroke and upstroke. The thrust peak at the end of wing fling during upstroke is higher than that at the end of wing fling during downstroke. This phenomenon is due to the wing root gap difference in the gearbox design and slightly tolerance of the gearbox assembly, resulting in wing clap-and-fling more fully at the end of upstroke than downstroke.

Figures 8 - 12 show the vertical thrust history synchronized with power consumption for one flapping cycle. The positive values dominate the force profiles in both half flap-

http://www.imavs.org/

http://www.imavs.org/

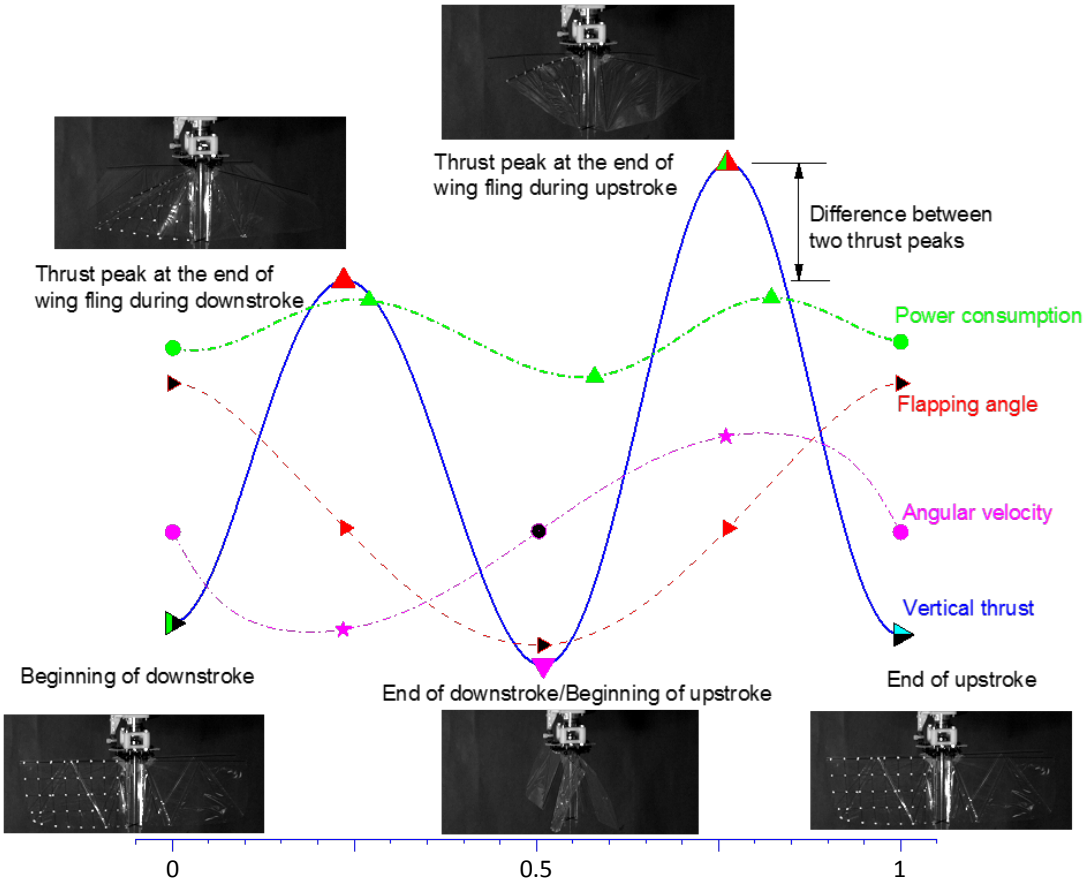


Figure 7: Typical vertical thrust and power consumption histories of the FW-MAV for one flapping cycle; flapping angle and angular velocity are for reference.

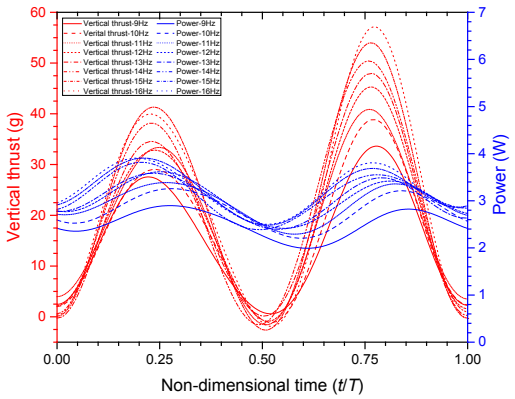


Figure 8: Time-dependent vertical thrust of Wing#1 for various frequencies .

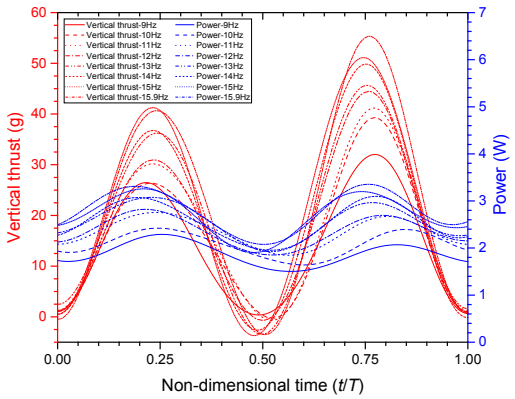


Figure 9: Time-dependent vertical thrust of Wing#2 for various frequencies.

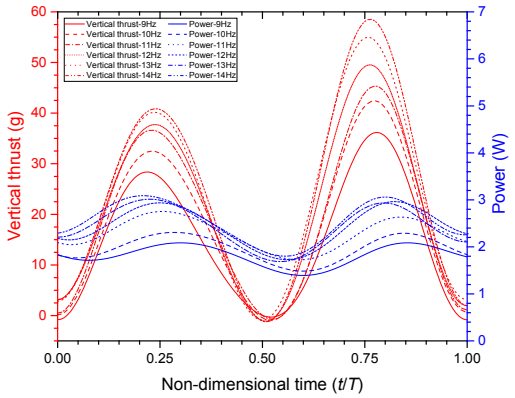


Figure 10: Time-dependent vertical thrust of Wing#3 for various frequencies.

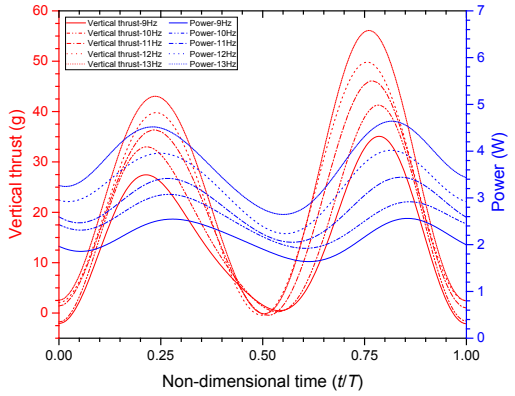


Figure 11: Time-dependent vertical thrust of Wing#4 for various frequencies.

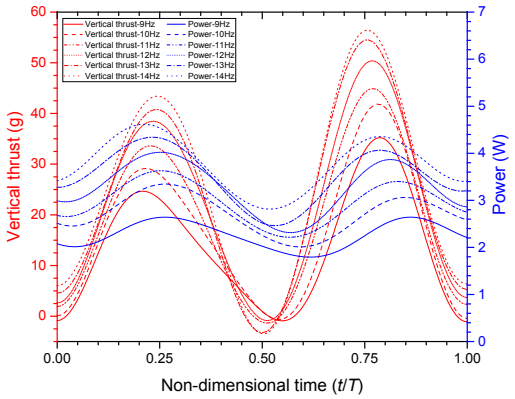


Figure 12: Time-dependent vertical thrust of Wing#5 for various frequencies.

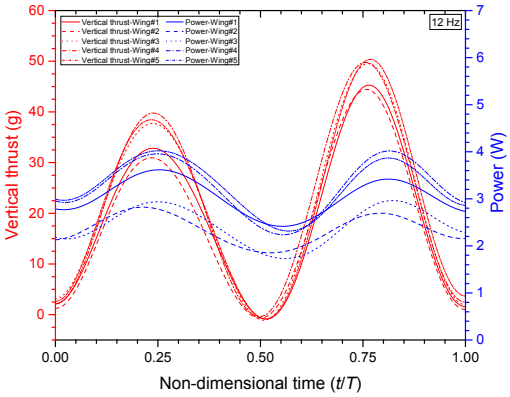


Figure 13: Vertical thrust and power consumption of five wing configurations at flapping frequency of 12 Hz for one flapping cycle.

ping strokes: downstroke and upstroke, resulting in positive cycle-averaged vertical thrust. The results indicate that the flapping frequency significantly influences the vertical force and power consumption. Higher flapping frequency yields a considerable increase in vertical thrust history at the region around the two peaks while the two troughs are almost not affected, and remain at nearly zero. Power consumption is shifted up in similar patterns as flapping frequency increases.

Figure 13 shows comparisons of vertical thrusts and power consumption of five wing configurations at flapping frequency of 12 Hz for one flapping cycle. It can be seen that the wings with more rigid leading edge (Group#1, 0.7 mm carbon tube leading edge) significantly increase the two vertical thrust peaks when compared with the wings with less rigid leading edge (Group#2, 0.6 mm carbon rod leading edge). The two troughs in the vertical thrust are not affected by the wing spanwise flexibility in both Group#1 and #2, and remain the same value at nearly zero. In general, the wings with the same wing leading edge stiffness (0.7 mm carbon tube, 0.6 mm carbon rod) produce similar vertical thrust history; only slightly difference was observed, see Figure 13. It is suggested that the wing leading edge stiffness plays more important role in vertical thrust production than the wing chordwise flexibility.

3.2 Maximum flapping frequency and vertical thrust

Given the same motor and gearbox, the maximum flapping frequency of the FW-MAV is depended on not only wing mass or inertia but also wing frontal area which is determined by the wing area projected on the plane perpendicular to the wing motion direction. Larger frontal wing area is subjected to more drag, and vice versa. When the wing configurations are changed, the wing deformation as well as wing rotation angle are passively changed. The wing deformation in spanwise and chordwise as well as wing rotation change the frontal area of the flapping wing, resulting in change of

http://www.imavs.org/

drag acting on the wing. Thus, different wing configuration has different flapping frequency at a given throttle level. Figure 14 shows maximum flapping frequency and vertical thrust of five wing configurations at maximum throttle level. The results indicate that the wings in each group (Group#1 and Group#2) show similar maximum frequency and vertical thrust. Moreover, it was observed from the high speed images of wing motion that the wings in Group#1 (Wing#1 and Wing#2) have larger wing deformation in spanwise direction than the wings in Group#2 (Wing#3, Wing#4, and Wing#5). Phase delay of wing tip displacement, i.e., the wing tip and wing root move in opposite direction was observed only in the wings in Group#1. Again, based on the results shown in Figure 14 it can be concluded that maximum flapping frequency and vertical thrust mostly depend on wing leading edge stiffness in spanwise direction, slightly change in chordwise stiffness plays marginal role in contribution to the maximum frequency and vertical thrust.

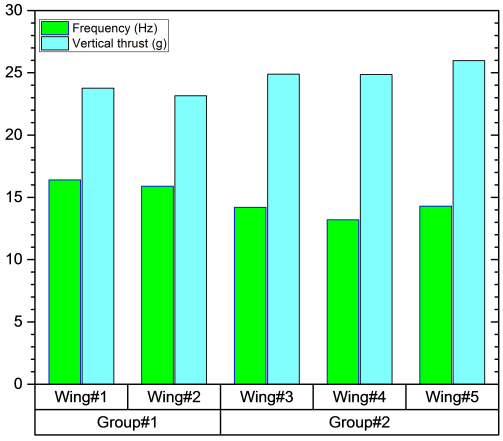


Figure 14: Maximum flapping frequency and vertical thrust of various wing configurations at maximum throttle level.

3.3 *Cycle-averaged vertical thrust and power consumption*

The results shown in Figure 15 indicate almost a nearly linear relationship between cycle-averaged vertical thrust and flapping frequency for all wing configurations. Theoretically, the vertical thrust or lift is proportional to the velocity or frequency square. Therefore, the nearly linear relationship obtained from the experiment can be explained by the induced velocity which reduces the effective angle of attack on the flapping wing. The induced velocity tends to be increased as the flapping frequency increases, thus, resulting in reducing effective angle of attack on the flapping wing or resulting in reducing vertical thrust on the wing. Again, it can seen the wings with the same leading edge stiffness produce similar cycle-averaged vertical thrust at the same flapping frequency. And the wings with more rigid leading edge generate more vertical thrust than the wing with less rigid leading edge.

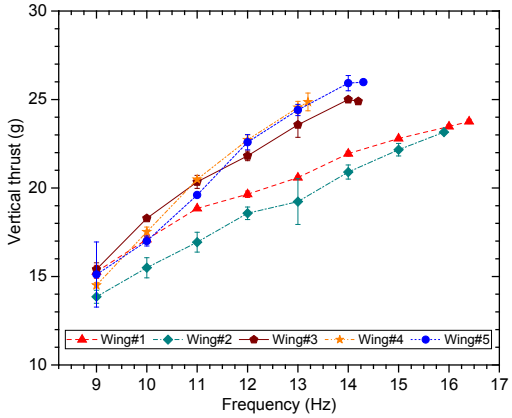


Figure 15: Vertical thrust w.r.t flapping frequency for various wing configurations.

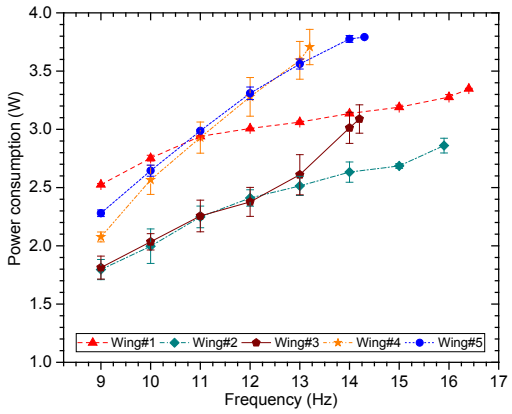


Figure 16: Power consumption w.r.t flapping frequency for the various wing configurations.

The power consumption also increases almost linearly with the flapping frequency. The wings with the same leading edge stiffness show the similar trend in power consumption, see Figure 16. Figure 17 indicates the thrust to power ratio for each wing configuration. This curve is relative flat with respect to frequency; there is about one unit difference from 9 Hz to the maximum flapping frequency of each wing. Wing#3 shows similar cycle-averaged vertical thrust to the Wing#4 and Wing#5 while consuming less power, resulting in the highest thrust to power ratio among the five wings. This is due to the more uniform wing rotation from wing root to wing tip, and positive wing camber created during flapping compared with the other wings, which was observed by the high speed images of wing motion. Wing#1 produces comparable vertical thrust, but consumes more power at frequencies lower than 14 Hz, resulting in inefficiency of thrust-to-power ratio. Overall, it is clear that spanwise wing flexibility influence both the vertical thrust history and power consump-

http://www.imavs.org/

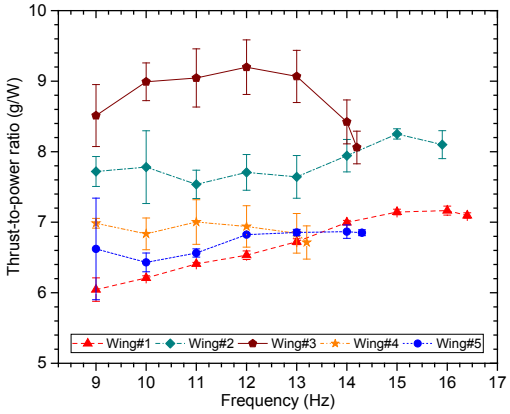


Figure 17: Thrust-to-power ratio w.r.t flapping frequency for the various wing configurations.

tion, while the chordwise wing flexibility plays marginal effect. The wings with the same leading edge stiffness produce similar patterns of both vertical thrust and power consumption histories, and also similar trend in cycle-averaged vertical thrust and power consumption.

4 CONCLUSION

Experiments have been conducted to investigate the wing flexibility on vertical thrust production and power consumption of a FW-MAV with double wing clap-and-fling effects, namely FlowerFly, which has two pairs of wings and a weight of 14.5 g. In summary, the following conclusions can be derived from this study: 1) Two vertical thrust peaks with different magnitude and two vertical thrust trough with almost zero magnitude were observed within one flapping cycle. Due to the passive wing rotation mechanism used in the FW-MAV, the vertical thrust peaks at about 25% and 75% flapping cycle, respectively, are created by the wing fling rather than the wing clap. The lower and higher peaks occur at the end of wing fling during downstroke and upstroke, respectively. 2) At flapping frequency of 9 Hz, the FlowerFly is about to produce enough vertical thrust for lift-off. Higher flapping frequency produces larger vertical thrust peaks while thrust troughs remain unchanged at almost zero value. 3) Wings with the same leading edge stiffness produce similar vertical thrust history and cycle-averaged vertical thrust. At the same flapping frequency, wings with stiffer leading edge produce higher thrust peaks, resulting in larger averaged thrust compared with the wing with less leading stiffness. Wing chordwise stiffness plays marginal role in both vertical thrust history and cycle-averaged thrust.

REFERENCES

[1] T. Nakata, H. Liu, Y. Tanaka, N. Nishihashi, X. Wang, and A. Satoi. Aerodynamics of a bioinspired flexible flapping-wing micro air vehicle. *Bioinspiration& Biomimetics*, 6(4):1–11, 2011.

[2] C.P. Ellington, C. Van Den Berg, A.P. Willmott, and A.L.R. Thomasi. Leading-edge vortices in insect flight. *Nature*, 384:626–630, 1996.

[3] T. Weis-Fogh. Unusual mechanisms for the generation of lift in flying animals. *Scientific American*, 233:80–87, 1975.

[4] M.H. Dickinson, F.O. Lehmann, and S.P. Sane. Wing rotation and the aerodynamic basis of insect flight. *Science*, 284:1954–1960, 1999.

[5] S.P. Sane. The aerodynamics of insect flight. *The Journal of Experimental Biology*, 206:4191–4208, 2003.

[6] L. Zhao, Q. Huang, X. Deng, and S.P. Sane. Aerodynamic effects of flexibility in flapping wings. *Journal of The Royal Society Interface*, 7:485–497, 2010.

[7] H. Hua, A.G. Kumar, G. Abate, and R. Albertanic. An experimental investigation on the aerodynamic performances of flexible membrane wings in flapping flight. *Aerospace Science and Technology*, pages 575–586, 2010.

[8] S. Deng, M. Percin, B. van Oudheusden, B. Remes, and H. Bijl. An experimental investigation on the aerodynamic performances of flexible membrane wings in flapping flight. *International Journal of Micro Air Vehicles*, 6(2):105–115, 2014.

[9] Q.V. Nguyen, W.L. Chan, and M. Debiasi. Performance tests of a hovering flapping wing micro air vehicle with double wing clap-and-fling mechanism. In *The International Micro Air Vehicle Conference and Competition*, Aachen, Germany, 2015.

[10] Q.V. Nguyen, W.L. Chan, and M. Debiasi. Hybrid design and performance tests of a hovering insect-inspired flapping-wing micro aerial vehicle. *Journal of Bionic Engineering*, 13:235–248, 2016.

[11] J.V. Caetano, M. Percin, B.W. van Oudheusden, B. Remes, C. de Wagter, G.C.H.E. de Croon, and C.C. de Visser. Error analysis and assessment of unsteady forces acting on a flapping wing micro air vehicle: free flight versus wind-tunnel experimental methods. *Bioinspiration& Biomimetics*, 10:1–21, 2015.

http://www.imavs.org/

First-Principles Modeling of A Miniature Tilt-Rotor Convertiplane in Low-Speed Operation

Guowei Cai*, Adnan Saeed, Fatima AlKhoori†, Ahmad Bani Younes, Tarek Taha, Jorge Dias, and Lakmal Seneviratne
Khalifa University, Abu Dhabi, UAE

ABSTRACT

In this paper, we present a comprehensive first-principles model that is developed for a miniature tilt-rotor convertiplane with Y-6 configuration. First, a flight dynamics model architecture is proposed based on the physical principle of the tilt-rotor convertiplane. Next, the parameter identification is conducted to determine the numerical values of all necessary parameters. The model fidelity is finally evaluated using practical data collected in flight experiments

http://www.imavs.org/

1 INTRODUCTION

Miniature Unmanned Aerial Vehicles (UAVs) have recently gained significant popularities in both defence and civil applications. In general, miniature UAVs are dominated by two types: fixed-wing and rotorcraft, and each of which, however, has inherent limitations. A fixed-wing UAV generally advances in payload capacity, flight range, and endurance but requires runways or special equipments for take-off and landing. On the other hand, a rotorcraft UAV features the unique hovering capability and versatility but its speed and endurance limit significantly trunks its applications in missions requiring wide coverage or long endurance. As such, a new trend of miniature UAV design, which integrates the advantages of both and contributes to a much broader range of applications, has been recently explored. As a result, the Hybrid UAV, or fixed-wing Vertical Take-Off and Landing (VTOL) UAV, is born.

The miniature hybrid UAVs available in the market are categorized into two types: convertiplane UAVs and tail-sitter UAVs. A convertiplane maintains its airframe orientation in all flight modes, and certain transition mechanisms such as tilting rotors or wings are required to achieve mode traverse. On the other hand, a tail-sitter is an aircraft that takes off and lands vertically on its tail, and the the entire airframe needs to tilt to accomplish cruise flight. Recently, the convertiplane shows a much enhanced popularity over the tail-sitter, which is partially due to the non-airframe-transition, equivalent maneuverability, more robustness to environmental disturbances, and sufficient payload. Inspired by its popularity, we select a miniature tilt-rotor convertiplane with Y-6 rotor configuration (i.e., tri-copter with coaxial motor setup), named FireFLY6 and illustrated in Fig. 1, is selected as our research platform.



Figure 1: FireFLY6 miniature convertiplane.

In the procedure of a UAV development, establishing a reliable model that is able to capture the flight dynamics accurately over the flight envelope of interest is critically important. For the case of hybrid UAV, the modeling work is much more challenging, given its complicated mechanical design and aerodynamic features. A number of modeling work (see, e.g., [1] [2] [3] [4]) have been documented in the literature. Our survey provides two observations: 1) no systematic modeling work for the hover or low-speed operation, which is an essential flight condition but challenging to be modeled, and 2) very rare work provide model fidelity validation based on flight-test data. Inspired by such niche, in this paper we intend to present a first-principles modeling work for our tilt-rotor convertiplane at low-speed range, and further prove the model fidelity via a comparison between the actual flight data and simulation model responses.

The remaining content of this paper is organized as follows: In Section 2, a complete first-principles modeling architecture is proposed. Section 3 presents the procedure on determining the physical parameters involved in the proposed model. In Section 4, the fidelity of the model is comprehensively examined. Finally, Section 5 draw some key conclusions.

2 FIRST-PRINCIPLES FLIGHT DYNAMICS MODEL

In this section, a flight dynamics model architecture is proposed. The flight dynamics of a tilt-rotor convertiplane

*Corresponding author
†Research and Development Engineer at Emirates Technology and Innovation Center (ETIC), Abu Dhabi, UAE

over the low-speed envelope consists of five key elements: 1) kinematics, 2) six degree-of-freedom (6-DOF) rigid-body dynamics, 3) forces and moments generated by the rotors, 4) aerodynamic forces and moments of the fuselage, and 5) on-board stabilizer dynamics. To simplify the model, it is assumed that the aircraft operates in a rather calm environment. Consequently, wind effect can be omitted.

2.1 Kinematics

The kinematics component addresses the relative translation and rotation between a multi-rotor vehicle and the local environment. Two coordinate systems are involved: the body frame and the local North-East-Down (NED) coordinate frame. The kinematics can then be expressed by

$$\begin{cases} \dot{\mathbf{P}}_{\mathrm{n}} &= \mathbf{R}_{\mathrm{n/b}}\,\mathbf{V}_{\mathrm{b}} \\ \dot{\mathbf{\Phi}} &= \mathbf{S}_{\mathrm{n/b}}\,\boldsymbol{\omega}_{\mathrm{b}} \end{cases} \tag{1}$$

where $\mathbf{P}_{\mathrm{n}} = [\,p_{\mathrm{x}} \quad p_{\mathrm{y}} \quad p_{\mathrm{z}}\,]'$ is the position vector expressed in the local NED frame, $\mathbf{\Phi} = [\,\phi \quad \theta \quad \psi\,]'$ is the Euler angle vector, $\mathbf{V}_{\mathrm{b}} = [\,u \quad v \quad w\,]'$ and $\boldsymbol{\omega}_{\mathrm{b}} = [\,p \quad q \quad r\,]'$ are the body-axis velocities and angular rates. $\mathbf{R}_{\mathrm{n/b}}$ and $\mathbf{S}_{\mathrm{n/b}}$ are the corresponding rotation matrices and their expressions can be found in many textbooks such as [5].

2.2 6DOF Rigid-Body Dynamics

The 6-DOF rigid-body equation of motion representing a vehicle's translational and rotational movements in its body frame are defined by

$$\begin{cases} \dot{\mathbf{V}}_{\mathrm{b}} &= -\boldsymbol{\omega}_{\mathrm{b}} \times \mathbf{V}_{\mathrm{b}} + \dfrac{\mathbf{F}_{\mathrm{b}}}{m} + \dfrac{\mathbf{F}_{\mathrm{g}}}{m} \\ \dot{\boldsymbol{\omega}}_{\mathrm{b}} &= \mathbf{J}^{-1}\left[\mathbf{M}_{\mathrm{b}} - \boldsymbol{\omega}_{\mathrm{b}} \times (\mathbf{J}\,\boldsymbol{\omega}_{\mathrm{b}})\right] \end{cases} \tag{2}$$

where m is the vehicle's mass, $\mathbf{J} = diag\{J_{\mathrm{xx}}, J_{\mathrm{yy}}, J_{\mathrm{zz}}\}$ is the moment of inertia matrix, $\mathbf{F}_{\mathrm{g}}$ is the gravity force projected onto the body frame, and $\mathbf{F}_{\mathrm{b}}$ and $\mathbf{M}_{\mathrm{b}}$ are the aerodynamic forces and moments. Their expressions are particularly essential to fidelity of the derived first-principles modeling. For the case of the tilt-rotor convertiplane, they are mainly generated by the multiple rotors $\mathbf{F}_{\mathrm{rt}}$ and the fuselage $\mathbf{F}_{\mathrm{f}}$. Their detailed expression will be addressed below.

$$\begin{cases} \mathbf{F}_{\mathrm{b}} &= \mathbf{F}_{\mathrm{rt}} + \mathbf{F}_{\mathrm{f}} \\ \mathbf{M}_{\mathrm{b}} &= \mathbf{M}_{\mathrm{rt}} + \mathbf{M}_{\mathrm{f}} \end{cases} \tag{3}$$

2.3 Forces and Moments Generated by Rotors

Forces and moments determination related to multiple rotors is important in establishing the dynamics model. Particularly, for the convertiplane with Y-6 rotor configuration, the performance interference between the upper and lower rotors needs to be considered carefully. Before going to the detailed analysis, three reasonable assumptions are made as follows:

1. The coaxial rotors require the same rotation speeds for torque balance.
2. The performance of the upper rotor is not influenced by the lower rotor.
3. Only half of the area of the lower rotor operates in an effective climb velocity induced by the upper rotor.

Combining the analysis based on the Momentum Theory [6] and force/torque expression introduced in [7], we obtain

$$\begin{cases} T_{j,U} &= C_{\mathrm{T,j}}\,\rho\,\Omega_{j,U}^2\,d_{\mathrm{rt}}^4 \\ Q_{j,U} &= C_{\mathrm{P,j}}\,\rho\,\Omega_{j,U}^2\,d_{\mathrm{rt}}^5/(2\pi) \\ \Omega_{j,U} &= K_{\mathrm{rt}}\,r_{j,U} + \Omega_{\mathrm{hov}} \end{cases} \tag{4}$$

$$\begin{cases} T_{j,D} &= k_{\mathrm{intf}} C_{\mathrm{T,j}}\,\rho\,\Omega_{j,D}^2\,d_{\mathrm{rt}}^4 \\ Q_{j,D} &= C_{\mathrm{P,j}}\,\rho\,\Omega_{j,D}^2\,d_{\mathrm{rt}}^5/(2\pi) \\ \Omega_{j,D} &= K_{\mathrm{rt}}\,r_{j,D} + \Omega_{\mathrm{hov}} \end{cases} \tag{5}$$

where the subscript U(D) denotes the upper(lower) rotor belong to the same coaxial rotor set, the subscript j stands for j^{th} rotor set marked in the previous figure, C_{T} and C_{P} are dimensionless thrust and power coefficients, ρ is the air density, Ω is the motor rotation speed, d_{rt} is the rotor diameter, r is the normalized PWM-based motor driving input with the range of [-1, 1], k_{intf} is the thrust factor from interference, K_{rt} is the scaling factor of rotation speed change to normalized joystick input deflection, and Ω_{hov} is the rotation speed at hover. Note that

1. C_{T} and C_{P} can be determined either via ground experiments or based on the wind-tunnel test data which is available online (see, for example, [8]).
2. Due to the motor's extremely fast response, a simple proportional relationship (using hover condition as the trim) is utilized in the first-principles model.
3. The constant value of k_{intf} is based on the Momentum Theory, given the case of balanced torques [6].
4. Given the assumption 1 and the Y-6 configuration, all the six rotors share an identical RPM, that is, Ω_{hov}, at ideal hover.

Both the translational and rotational movements of the vehicle leads to slight flapping motion of the rotor blade. To account for this effect, a quasi-steady rotor flapping expression, proposed in [9], is adopted in our modeling process, and defined by:

$$\begin{cases} \beta_{\mathrm{lc},j} &= (K_{\mathrm{ab1}}q + K_{\mathrm{ab2}}p - K_{\mathrm{ab3}}u - K_{\mathrm{ab4}}v)/K_{\mathrm{ab5}} \\ \beta_{\mathrm{ls},j} &= (-K_{\mathrm{ab1}}p + K_{\mathrm{ab2}}q - K_{\mathrm{ab3}}v + K_{\mathrm{ab4}}u)/K_{\mathrm{ab5}} \end{cases} \tag{6}$$

where $\beta_{\mathrm{lc},j}$, and $\beta_{\mathrm{ls},j}$ are the longitudinal and lateral flapping angles, K_{ab1} to K_{ab5} are the lumped derivatives whose detailed definitions can be found in [9]. Is should be highly noted that due to the above definition, the upper and lower

http://www.imavs.org/

rotors of the same coaxial rotor set share the same flapping angles. Therefore, no subscript U and D are used in the above equation. With the flapping angles defined, the force and moment vectors of the upper and low rotors are calculated by:

$$\begin{cases} \mathbf{F}_{\mathrm{rt},j,U} &= [\,T_{j,U}\,\mathbf{s}(\beta_{\mathrm{lc}}) \quad -T_{j,U}\,\mathbf{s}(\beta_{\mathrm{ls}}) \quad -T_{j,U}\,\mathbf{c}(\beta_{\mathrm{lc}})\,\mathbf{c}(\beta_{\mathrm{ls}})\,]' \\ \mathbf{F}_{\mathrm{rt},j,D} &= [\,T_{j,D}\,\mathbf{s}(\beta_{\mathrm{lc}}) \quad -T_{j,D}\,\mathbf{s}(\beta_{\mathrm{ls}}) \quad -T_{j,D}\,\mathbf{c}(\beta_{\mathrm{lc}})\,\mathbf{c}(\beta_{\mathrm{ls}})\,]' \end{cases} \tag{7}$$

$$\begin{cases} \mathbf{M}_{\mathrm{rt},j,U} &= [\,0 \quad 0 \quad \pm Q_{j,U}\,]' \\ \mathbf{M}_{\mathrm{rt},j,D} &= [\,0 \quad 0 \quad \pm Q_{j,D}\,]' \quad (+\text{ for CW}) \end{cases} \tag{8}$$

The aerodynamic drag caused by the fuselage of multi-rotor vehicles is generally small and can thus be ignored. Therefore, the combined aerodynamic force $\mathbf{F}_{\mathrm{b}}$ and moment $\mathbf{M}_{\mathrm{b}}$ can be reduced to:

$$\begin{cases} \mathbf{F}_{\mathrm{rt}} &= \sum_{j=1}^{n} (\mathbf{F}_{\mathrm{rt},j,U} + \mathbf{F}_{\mathrm{rt},j,D}) \\ \mathbf{M}_{\mathrm{rt}} &= \sum_{j=1}^{n} (-\mathbf{M}_{\mathrm{rt},j,U} - \mathbf{M}_{\mathrm{rt},j,D} + l_{j,U} \times \mathbf{F}_{\mathrm{rt},j,U} + l_{j,D} \times \mathbf{F}_{\mathrm{rt},j,D}) \end{cases} \tag{9}$$

where $l_{j,U/D}$ is the body-frame position vector of the jth upper (lower) rotor with respect to the center of gravity (CG).

2.4 Forces and Moments of Fuselage

A unique feature of the low-speed operation of a tilt-rotor convertiplane is the aerodynamic forces and moments generated by its fuselage. From the perspective of aerodynamic effect, there are generally three force components: lift, drag, and side forces. Each of the three force components is further associated with a moment generation. Furthermore, in low-speed operation, the side force and moments are sufficiently small and thus can be omitted.

As for the lift and drag analysis, stripe theory is adopted in our work to achieve sufficient accuracy. More specifically, the entire fuselage was divided into multiple sections along the starboard with sufficiently small span value. For each section, we have

$$\begin{cases} \Delta L &= \frac{1}{2}\rho \Delta A V_{\mathrm{local}}^2 C_{\mathrm{l,b}} \\ \Delta D &= \frac{1}{2}\rho \Delta A V_{\mathrm{local}}^2 C_{\mathrm{d,b}} \end{cases} \tag{10}$$

where ΔL and ΔD are the sectional lift and drag forces, ΔA is the effective fuselage area for lift and drag generation, V_{local} is the local total velocity at the current section, $C_{\mathrm{l,b}}$ and $C_{\mathrm{d,b}}$ are the lift and drag coefficients for the current section

Once the sectional lift and drag forces are determined, the decomposition to the body frame needs to be conducted as follows

$$\begin{cases} \Delta F_{\mathrm{f,x}} &= -L\,\mathbf{s}(\alpha_{\mathrm{in}}) - D\,\mathbf{c}(\alpha_{\mathrm{in}}) \\ \Delta F_{\mathrm{f,z}} &= -L\,\mathbf{c}(\alpha_{\mathrm{in}}) + D\,\mathbf{s}(\alpha_{\mathrm{in}}) \end{cases} \tag{11}$$

where α_{in} is the incidence angle.

The corresponding moment is calculated by

$$\Delta M_{\mathrm{f}} = \Delta F_{\mathrm{f}} \times l_{\mathrm{ae}} \tag{12}$$

where $l_{\mathrm{ae}} = [\,x_{\mathrm{ae}} \quad y_{\mathrm{ae}} \quad z_{\mathrm{ae}}\,]'$ is the aerodynamic center's location with respect to the CG.

The overall aerodynamic force and moment generated by the fuselage is achieved by integrating the sectional forces and moments along the starboard.

2.5 On-board Stabilizer Dynamics

The last dynamics component of the first-principles modeling is related to the on-board stabilizer module, which is widely used in multi-rotor vehicles to dampen the rotor response. In our work, we employ an open-source stabilizer module, named Pixhawk, with configurable PID control structure and control gains. Its stabilization mechanism is shown in Fig. 2. More specifically,

1. For the rolling and pitching dynamics, both Euler angle (ϕ or θ) and angular rate (p or q) are fed back to the stabilizer. Two successive control loops are included: PID control structure is adopted in the inner loop for angular rate stabilization, whereas P control is adopted in the outer loop for Euler angle stabilization.

2. For the heave dynamics, vertical acceleration (a_z) is fed back for stabilization, which is achieved by a single PI control loop.

3. For the yawing dynamics, both heading angle Ψ and yaw rate r are fed back to the stabilizer. Two successive control loops are adopted: PI control structure is adopted in the inner loop for yaw rate stabilization, whereas P control is adopted in the outer loop for heading angle stabilization.

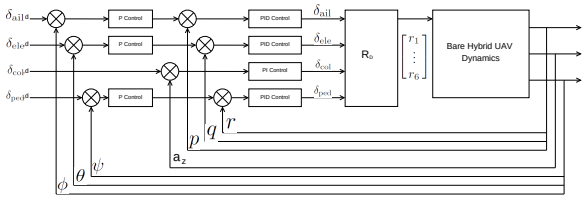


Figure 2: Onboard stabilizer dynamics.

Finally, the stabilizer module also acts as a signal distributor, converting the driving commands to motor-recognized control signals for n motors (i.e., r_1 to r_6), realized by

$$\begin{bmatrix} r_1 \\ \vdots \\ r_6 \end{bmatrix} = R_{\mathrm{D}} \begin{bmatrix} \delta_{\mathrm{ail}} \\ \delta_{\mathrm{ele}} \\ \delta_{\mathrm{col}} \\ \delta_{\mathrm{ped}} \end{bmatrix} \tag{13}$$

http://www.imavs.org/

where R_{D} is 6×4 signal distribution matrix given by

$$R_{\mathrm{D}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ 0 & -1 & 1 & 1 \\ 0 & -1 & 1 & -1 \end{bmatrix} \tag{14}$$

3 PARAMETER IDENTIFICATION

To complete the first-principles modeling, a set of physical parameters are required to be determined. These parameters are categorized into three types, including: 1) parameters that can be measured directly, 2) parameters that can be determined via computer-aided-design (CAD), and 3) parameters that need to be determined experimentally. Note that

1. In Table 1, all the parameters that can be measured directly are summarized in the first category.

2. CAD software such as SolidWorks can be adopted as an efficient tool to determine the physical properties of the fuselage and rotor blades. The virtual hybrid UAV built in the CAD environment is required to maximally match the actual one, in terms of physical dimension and material property. Fig. 3 illustrate the virtual and real FireFLY6 platforms, and the related parameters are summarized in the second category of Table 1.



Figure 3: Virtual and actual FireFLY6 hybrid UAVs.

3. The aerodynamic coefficients for the propellers are determined either via wind-tunnel or ground experiment, depending on the availability of the wind-tunnel test data. Recently, thorough research and wind-tunnel experiments have been carried out for hobby-based propellers and representative airfoils, and the associated results are accessible in the literature or online databases (see, e.g., [8]). However, if such data is not available, particularly to certain newly developed propellers, dynamometer-based experiment should be conducted, in which the thrust and torque values at different rotation speeds should be recorded for determining the coefficients via data curve-fitting.

4. The lift, drag, and moment coefficients for the sectional fuselage can be either determined via wind-tunnel or using simulation toolkit. The latter is preferable in our work given its availability and efficiency. A popular software analysis toolkit named XFLR5 is adopted to generate these coefficients over the complete range of angle of attack [-π, π], assuming a uniform airfoil is applied to the entire flying-wing fuselage. The resulting coefficients are obtained in the lookup table format, and plotted in Figs. 4 to 6 for convenient reference.

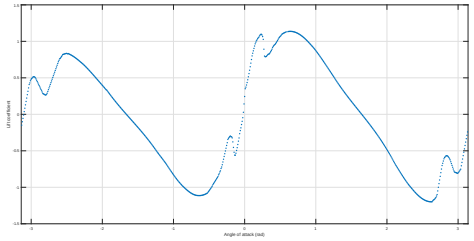


Figure 4: Lift coefficient of the selected airfoil at ultra-low Reynolds number.

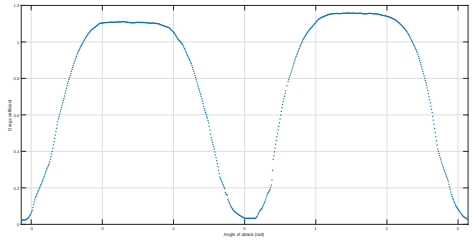


Figure 5: Drag coefficient of the selected airfoil at ultra-low Reynolds number.

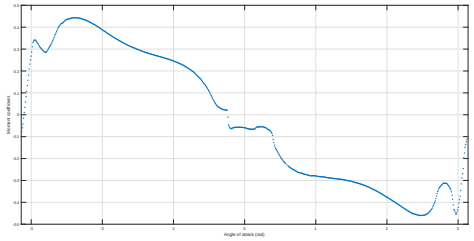


Figure 6: Moment coefficient of the selected airfoil at ultra-low Reynolds number.

5. As for the on-board stabilizer, if the control structure and control gains are configurable, their values can be

http://www.imavs.org/

http://www.imavs.org/

obtained directly. For the case of FireFLY6, these parameters are summarized in the third category of Table 1. If these information are not available, two types of ground experiments are required to initially estimate the parameters related to the on-board stabilizer. The former is for determining the scaling factors, in which the hybrid UAV is required to be fixed onto a testing-rig with only one free degree of freedom (i.e., roll, pitch, heave, or yaw), and step input is injected into one of the four input channels. Both the input and the corresponding output should be recorded for scaling factor determination. The second experiment is conducted for estimating the PI/PID control gains and the signal distribution matrix. The control gains and distribution matrix can be determined via input-output data curve-fitting.

Table 1: Parameters required for FireFLY6 first-principles modeling.

Category	Symbol	Value	Unit
1	c	0.027	m
1	d_{rt}	0.255	m
1	m	2.71	kg
1	ρ	1.16	kg m^{-3}
2	J_{xx}	0.0945	kg m^2
2	J_{yy}	0.0816	kg m^2
2	J_{zz}	0.1691	kg m^2
2	I_β	3.72 e-9	kg m^2
2	k_β	0.251	N m rad^{-1}
3	K_ϕ	5	NA
3	$K_{P,p}$	0.12	NA
3	$K_{I,p}$	0.002	NA
3	$K_{D,p}$	0.003	NA
3	K_θ	5.5	NA
3	$K_{P,q}$	0.12	NA
3	$K_{I,q}$	0.002	NA
3	$K_{D,q}$	0.003	NA
3	K_ψ	2.8	NA
3	$K_{P,r}$	0.22	NA
3	$K_{I,r}$	0.02	NA
3	$K_{D,r}$	0.002	NA
3	$K_{P,az}$	2	NA
3	$K_{I,az}$	0.0015	NA

4 MODEL VALIDATION

Model validation has been be conducted based on multiple flight experimental data. Frequency sweep, which refers to a class of control inputs that has a quasi-sinusoidal shape of increasing frequency [10], is adopted as the input signal because it is relatively easy to issue and can persistently perturb the frequency band of interest. Each individual perturbation starts with the hovering condition. The frequency sweep, which is issued by an human pilot, is then sequentially injected into one of the four input channels to excite the flight dynamics. Meanwhile, the pilot maintains the rough trim condition by issuing minimum and uncorrelated control signals in the remaining input channels. The same procedure is repeated for the remaining joystick inputs. As a result, four types of perturbations are recorded.

The validation results are depicted in Figs. 7 to 12, which all belong to the elevator channel perturbation. Note that for each output response, a fairly good matching has been achieved. Mismatch can be still observed, which is mainly caused by the omittance of certain flight dynamics mode, the simplified model architecture of several flight dynamics modes included, and the environmental disturbances that are not modeled in the current work.

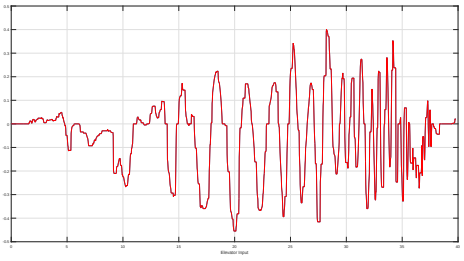


Figure 7: Elevator input.

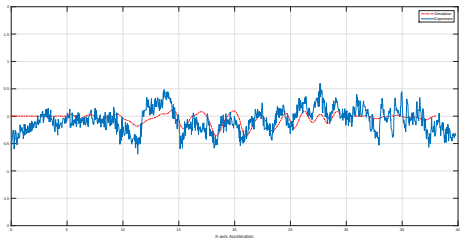


Figure 8: X-axis acceleration.

5 CONCLUSION

In this paper, we have proposed a feasible first-principles model for a Y-6 hybrid UAV. Parameter identification is discussed and flight test data collected in the low-speed range are adopted for model fidelity validation. The future work will concentrate on the following key issues: 1) enhanced first-principles architecture, 2) enhanced parameter identification procedure with higher accuracy, 3) more comprehensive evaluation with necessary analysis, and 4) an in-depth analysis on

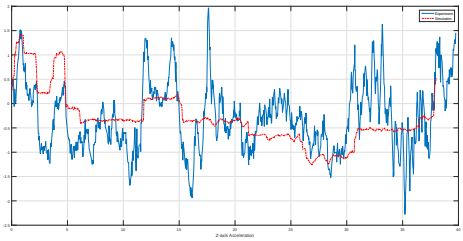


Figure 9: Z-axis acceleration.

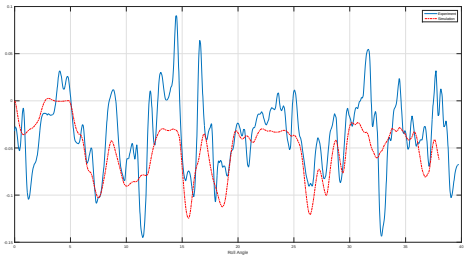


Figure 10: Roll angle.

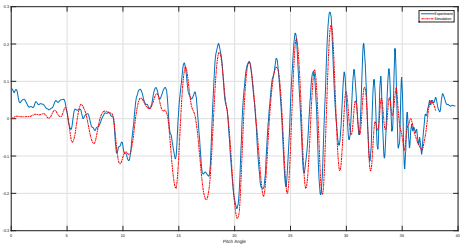


Figure 11: Pitch angle.

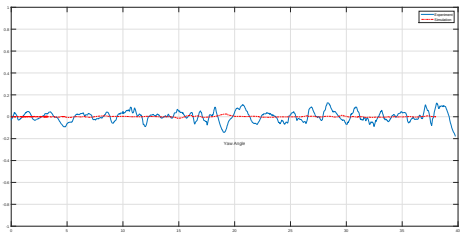


Figure 12: Heading angle.

the necessity and contribution of the dynamic components involved in structuring the first-principles model.

REFERENCES

[1] Kaan Taha Öner, Ertuğrul Çetinsoy, Mustafa Ünel, Mahmut Faruk Akşit, Ilyas Kandemir, and Kayhan Gülez. Dynamic model and control of a new quadrotor unmanned aerial vehicle with tilt-wing mechanism. 2008.

[2] Kaan Taha Öner, Ertuğrul Çetinsoy, EFE SIRIMOĞLU, Cevdet Hançer, Mustafa Ünel, Mahmut Faruk Akşit, Kayhan Gülez, and Ilyas Kandemir. Mathematical modeling and vertical flight control of a tilt-wing uav. *Turkish Journal of Electrical Engineering & Computer Sciences*, 20(1):149–157, 2012.

[3] Pierre-Richard Bilodeau and Franklin Wong. Modeling and control of a hovering mini tail-sitter. *International Journal of Micro Air Vehicles*, 2(4):211–220, 2010.

[4] J Escareno, S Salazar, and R Lozano. Modeling and control of a convertible vtol aircraft. In *45th IEEE Conference on Decision and Control, San Diego, California*, pages 13–15. Citeseer, 2006.

[5] Guowei Cai, Ben M Chen, and Tong Heng Lee. *Unmanned rotorcraft systems*. Springer Science & Business Media, 2011.

[6] J Gordon Leishman. *Principles of Helicopter Aerodynamics with CD Extra*. Cambridge university press, 2006.

[7] John B Brandt and Michael S Selig. Propeller performance data at low reynolds numbers. In *49th AIAA Aerospace Sciences Meeting*, pages 2011–1255, 2011.

[8] Gavin Ananda et al. Uiuc propeller database. *URL: http://www. ae. illinois. edu/m-selig/props/data/apce_9x4. 5_jb1000_6917. txt,[cited 15 Feb 2011]*, 2008.

[9] Pierre-Jean Bristeau, Philippe Martin, Erwan Salaün, Nicolas Petit, et al. The role of propeller aerodynamics in the model of a quadrotor uav. In *European control conference*, volume 2009, 2009.

[10] Mark B Tischler and Robert K Remple. *Aircraft and rotorcraft system identification: engineering methods with flight test examples*. Reston: American Institute of Aeronautics and Astronautics, 2006.

http://www.imavs.org/

Improved Efficiency Electronic Speed Controller Development for 3-Phase Brushless DC Motor in Unmanned Aerial Systems

Kun Li*, Ao Chen, Kun Zhang and Ben M. Chen
Department of Electrical and Computer Engineering, NUS, Singapore, 117576

http://www.imavs.org/

ABSTRACT

This paper presents the methodology and technique of implementing vector control method on brushless DC (BLDC) motors. Traditional scalar control method and the advanced field oriented control (FOC) are compared with the aspects of efficiency, control performance and reference speed range. Theoretical analysis of both methods are explained with their decomposed function components in detail. To obtain rotor position and its angular rate, a nonlinear observer based on the motor mathematical model is adopted to estimate these unknown states, for feedback to the control process. Customized hardware kit is designed and developed for implementation of the FOC algorithm. Several experiments are conducted to test the convergence speed, efficiency and reference speed range for both methods. The results show the superiority of FOC scheme with a properly tuned gain and parameter set.

1 INTRODUCTION

Brushless direct current (BLDC) motor has become one of the most popular motor types used in small-scaled electric vehicles, especially in the area of unmanned aerial systems (UAS). From 2003 to 2010, the BLDC motor market grew from $300 million to over $1.3 billion. Compared with brushed direct current (BDC) motors, BLDC motor shows a distinct advantage for its high control accuracy, long lifespan, high reliability, high power-to-volume ratio and low noise. When operated in rated conditions, the life expectancy is over 10,000 hours [1]. As a tradeoff, high control accuracy and reliability rely on the complexity of electrical speed controller (ESC) for BLDC motor.

The normal motor control techniques depend on the knowledge of accurate rotor position and some techniques require the feedback of rotational speeds. These techniques fall into two major categories: sensor feedback control and sensorless control [2, 3]. Sensor feedback control take in the sensor-measured angle and angular rate data as control feedbacks. These sensors providing the angle measurement include hall sensors, resolvers and encoders. For sensorless control, the normal method is to detect the back eletromagnetic force (BEMF) [4]. However, the BEMF signal is hard to detect, due to the electrical noise from the rotating machine, especially at low speed. The most common algorithm implemented in the commercial-off-the-shelf (COTS) ESC for BLDC motor is the scalar control method, also named six-step commutation scheme [5, 6]. However, according to the working principle, this method suffers from low efficiency caused by the copper loss. At low speed, the scalar control method is inaccurate due to its discrete nature and noise in BEMF detection [7]. Several research works proposed novel methods to detect the BEMF and improve the accuracy [8, 9], while efficiency is still a problem for scalar control method.

The field oriented control (FOC) is a vector control technique which was first conceptualized in 1929 by Robert H.Park [10]. FOC technique [11, 12] maximizes the quadrature current component and minimizes direct current component, in order to produce constant torque effectively. It is originally designed for permanent magnetic synchronous motors (PMSMs). Some researchers [13] even extended this advanced control technique for BLDC (with trapezoidal BEMF compared to sinusoidal BEMF for PMSMs). However, due to the difference between 3-phase AC machine and BLDC, the BEMF method cannot be directly implemented for FOC technique, and rotor position needs to be observed for sensorless control method.

This research work mainly focuses on the development of a FOC control hardware kit and the implementation of the vector control scheme based on the algorithm workflow. Section 2 gives the general mathematical model of a 3-phase BLDC. Section 3 gives the comparison between the traditional scalar control method and vector control method FOC. Section 4 gives the algorithm implementation details and the hardware overview. Followed by the experimental setup in Section 5 and results are discussed in Section 6.

2 GENERAL BLDC MOTOR MODEL

Fig. 1 gives a general synchronous motor model, namely the BLDC type or 3-phase AC type, controlled by a logically designed invertor network. Each phase is controlled by two electrical switches, i.e. power metaloxidesemiconductor field-effect transistors (MOSFETs) shown as T_i in Fig. 1, one terminal of which connects to power E and the other terminal connects to power ground G.

The mathematical model can be described as a differential equation with constant coefficients with applying the Lenz-

*Email address: elelik@nus.edu.sg

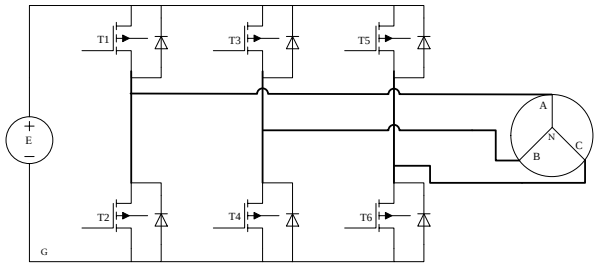


Figure 1: Closed-loop controller architecture of a BLDC ESC module.

Faraday model as follows:

$$\begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} = R_s \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} L_s & M & M \\ M & L_s & M \\ M & M & L_s \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \Phi_{\text{sf}} \cos(\theta) \\ \Phi_{\text{sf}} \cos(\theta - \frac{2\pi}{3}) \\ \Phi_{\text{sf}} \cos(\theta + \frac{2\pi}{3}) \end{bmatrix} \quad (1)$$

where R_s and L_s are the stator resistance and inductance, M is the mutual inductance between two stators, $[v_a \ v_b \ v_c]^T$ and $[i_a \ i_b \ i_c]^T$ are stator voltage and current, θ is the rotor position and for a multi-pole BLDC, the rotor position θ is calculated from the electrical angle θ_e with $\theta = 2\theta_e/N_p$, where N_p is the number of pole pairs. Φ_{sf} is the max flux value created by the permanent magnet through the stators.

3 SCALAR CONTROL METHOD

COTS ESCs are implemented with the most direct method to control the rotor motion, scalar control [14, 15]. Each conducting interval is 120° by electrical angle, which is exactly 120° by mechanical rotating angle for a single pole-pair permanent magnet rotor.

Fig. 2 shows the control waveform of the six-step commutation and the sequence is as AB-AC-BC-BA-CA-CB to make sure that magnetic field generated by the winding keeps changing 60° each commutation step, yielding a constant rotating direction. The core technology inside is the knowledge of the rotor position, i.e. the exact sector the rotor currently lies in. The technique normaly uses the third floating phase to detect the BEMF [16, 17]. This is the voltage generated in BLDC motors when there is relative motion between the armature of the motor and the magnetic field from the motor's field magnets, or windings. Further, the voltage is proportional to the magnetic field, length of wire in the armature, and the speed of the motor according to Faraday's law.

As it is difficult to obtain the neutral voltage of a three-phase BLDC motor, which is often built in the mechanical structure. In the practical case, a virtual neutral point resistance network is built to emulate the motor neutral point for zero-crossing BEMF detection [14]. As such, the timing of commutation sequence is obtained via measuring the zero-crossing BEMF of the floating terminal.

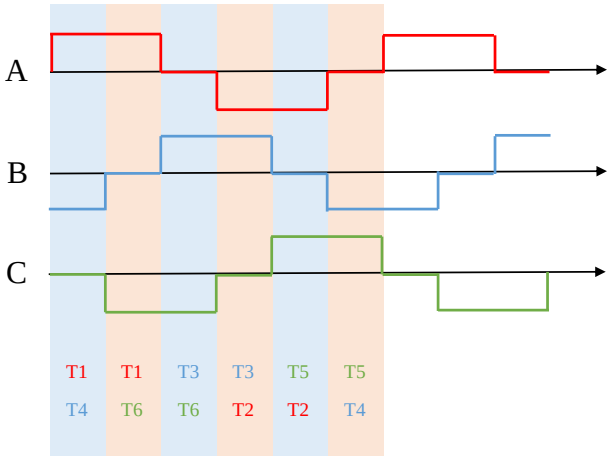


Figure 2: Six-step commutation control waveforms.

4 VECTOR CONTROL METHOD

FOC is commonly used for PMSM motor, and has not been commercially applied on BLDC motor. According to the model provided by Eq. 1, the rotational motion is created by the force exerted by the magnetic fields from stator and rotor. Since the rotor's magnetic field is fixed, the way of exerting stator magnetic field determines the performance and efficiency of the rotation. By properly aligning the rotor and stator flux, the most optimal torque production of the rotor is generated by vector control method [18].

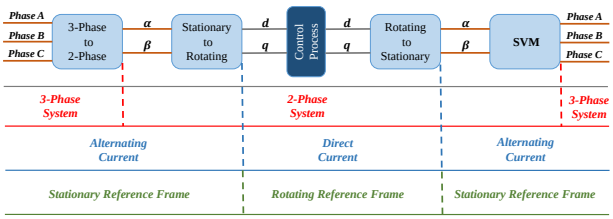


Figure 3: Process of FOC.

As shown in Fig. 3, the process of FOC can be modularized into five blocks. The first block is conversion of stator winding current or voltage from 3-phase rotating coordinate (a, b, c) to 2-phase rotating coordinate (α, β), also known as Clarke transformation, in order to reduce the control complexity. The projection into the 2-phase coordinate system depending on time and rotational speed:

$$\begin{aligned} i_{S\alpha} &= i_a \\ i_{S\beta} &= \frac{1}{\sqrt{3}} i_a + \frac{2}{\sqrt{3}} i_b \end{aligned} \quad (2)$$

http://www.imavs.org/

http://www.imavs.org/

where the model in Eq. 1 can be rewritten in the 2-phase rotating frame $(\alpha,\ \beta)$ as,

$$l_s \dot{i}_{\alpha\beta} = -r_s i_{\alpha\beta} + \omega \Phi_{\text{sf}} \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} + v_{\alpha\beta} \tag{3}$$

where $l_s = 3/2L_s$ and $r_s = 3/2R_s$.

The second block is a transformation from time-variant system $(\alpha,\ \beta)$ to a time-independent system direct-quadrature $(d,\ q)$, also known as Park transformation. The d-axis is aligned with the rotor flux and the q-axis is orthogonal to the rotor flux. The core of the Park transformation is the feedback of the rotor position θ, with which the current can be transformed into flux and torque components with the following equations:

$$\begin{aligned} i_{Sd} &= i_{S\alpha}\cos\theta + i_{S\beta}\sin\theta \\ i_{Sq} &= -i_{S\alpha}\sin\theta + i_{S\beta}\cos\theta \end{aligned} \tag{4}$$

The third block is the control process component regarding with the decoupled time-invariant system. In this paper, a simple PI controller was implemented for both the speed and torque component. The forth block is formulation of the control outputs. Control output generated from the speed and torque controller are then fed to the reverse Park transformation as:

$$\begin{aligned} v_{ref\alpha} &= v_{refd}\cos\theta - v_{refq}\sin\theta \\ v_{ref\beta} &= v_{refd}\sin\theta + v_{refq}\cos\theta \end{aligned} \tag{5}$$

Ideally the current vector should always be in the quadrature direction of the rotor and rotate together with the rotor. In this case, constant torque can be produced efficiently [7, 19].

The last block is to generate actuator signal to the 3-phase invertors, namely space vector modulation (SVM). SVM is a method to generate pulse width modulation (PWM) to control the timing of the commutation. There will be two considerations of the MOSFET switches as: a) three of the switches must be ON and the other three must be OFF; b) the high and the low side switches of the same phase must be driven with two complementary pulsed signals. Under these constraints, there will be eight combinations which forms six sectors of the whole loop.

As state above, knowledge of the rotor position is the core of BLDC motor control, especially in vector control method, where a slight error in rotor position will bring large deviation in d-q axis estimation. However, due to limitations of the applications in UAS on the aspects of weight and cost budgets, sensorless control of the BLDC is considered instead of using resolvers or encoders. Research works in [20, 21] provide us a solution to estimate rotor position and rotational speed with a nonlinear observer. Consider the model in $(\alpha,\ \beta)$ frame, a new state variable is defined as,

$$x = l_s i_{\alpha\beta} + \Phi_{\text{sf}} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \tag{6}$$

$$y = -r_s i_{\alpha\beta} + v_{\alpha\beta} \tag{7}$$

where we have simply as $\dot{x} = y$ according to the model in Eq. 3. A vector function is defined to construct the nonlinear observer,

$$\eta(x) = x - l_s i_{\alpha\beta} \tag{8}$$

and we have the Euclidean norm of the vector function as $||\eta(x)||^2 = \Phi_{\text{sf}}^2$. The nonlinear observer is constructed as:

$$\dot{\hat{x}} = y + \frac{\kappa}{2}\eta(\hat{x})\left[\Phi_{\text{sf}}^2 - ||\eta(\hat{x})||^2\right] \tag{9}$$

where $\hat{x}$ is the observer state variable and $\kappa > 0$ is the observer gain. According to the state expression in Eq. 6, we have,

$$\begin{bmatrix} \cos\hat{\theta} \\ \sin\hat{\theta} \end{bmatrix} = \frac{1}{\Phi_{\text{sf}}}(\hat{x} - l_s i_{\alpha\beta}) \tag{10}$$

and we can obtain the estimate of the rotor position as,

$$\hat{\theta} = \tan^{-1}\left(\frac{\hat{x}_2 - l_s i_\beta}{\hat{x}_1 - l_s i_\alpha}\right) \tag{11}$$

and thus an observer error can be further obtained as,

$$\begin{aligned} \dot{\tilde{x}} &= -\kappa\alpha(\tilde{x},t)\left\{\tilde{x} + \Phi_{\text{sf}}\begin{bmatrix}\cos\theta(t) \\ \sin\theta(t)\end{bmatrix}\right\} \\ \alpha(\tilde{x},t) &= \frac{1}{2}||\tilde{x}||^2 + \Phi_{\text{sf}}[\tilde{x}_1\cos\theta(t) + \tilde{x}_2\sin\theta(t)] \end{aligned} \tag{12}$$

where $\alpha(\tilde{x},t)$ is the cost function to evaluate the stability properties. The stability proof and construction of the speed observer can be referred in [20, 21].

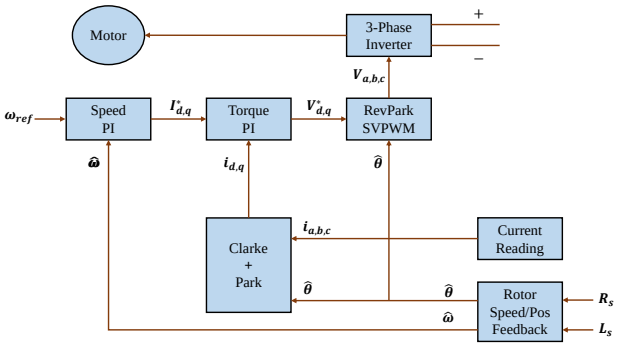


Figure 4: Overview of the FOC control logics.

The whole workflow of the FOC process is explained in Fig. 4. The startup procedure depends on an alignment or pre-position phase to check if the angle and rotational speed estimator are able to converge. For BLDC motor, the torque produced is not relevant with direct component, hence the reference i_{refd} can be set to zero for optimal performance. Several parameters are recursively tuned to achieve the best performance, such as startup duration, etc. These parameters significantly affect the dynamic performance of rotor.

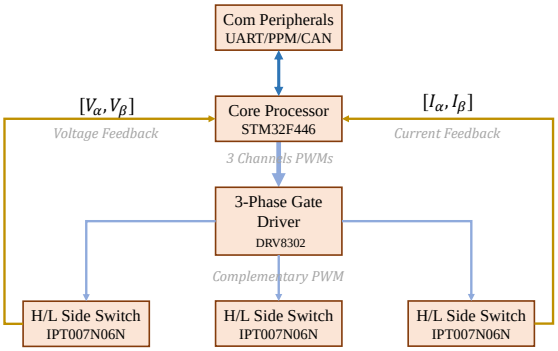


Figure 5: FOC hardware development: function blocks incorporated.

5 HARDWARE AND EXPERIMENTS

According to the process components analyzed in Fig. 3, function blocks need to be incorporated are listed in Fig. 5 with the signal flows. Communication peripherals are reserved for the upper-level command from the UAS avionic system, with protocols of serial or controller area network (CAN) bus or PWM signals. The processor used is STM32F446 with ARM Cortex-M3 core with a real-time operating system (RTOS)[1] running on. An device DRV8302 is used for 3-phase gate driving and process the measured phase current. The layout of the board can be seen in Fig. 6.

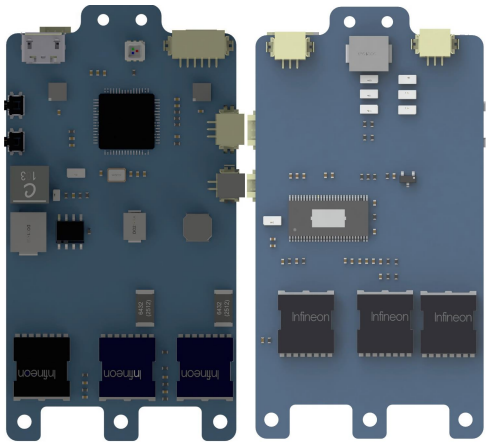


Figure 6: FOC hardware development: printed circuit board (PCB) overview.

An experiment is designed to verify the advantages in control performance and efficiency for BLDC compared with the traditional ESCs (here we use a T-motor ESC as a comparison). BLDC motors (hexTronik outrunner 1700 kV and BR 2804 − 1800 kV) with 6030 propeller load is mounted on ATI load cell Mini40 for force and torque measurement. Eagle-tree elogger v4 testing suite is used for the recording of rotational speed, overall voltage, current assumption and temperature information. Efficiency η can be calculated as equation below, where τ is the torque produced by the propeller,

$$\eta = \frac{2\pi\omega\tau}{60 \times vi} \tag{13}$$

[1]ChibiOS: please refer http://www.chibios.org/dokuwiki/doku.php for more information.

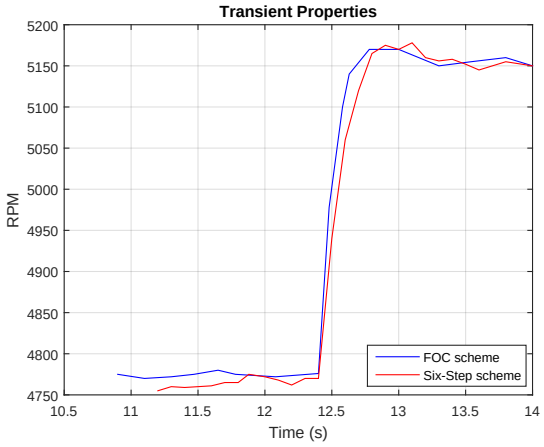


Figure 8: Dynamic response of FOC/SixStep with changing reference speed for rotor combo of hexTronik outrunner 1700 kV and 6030 propeller.

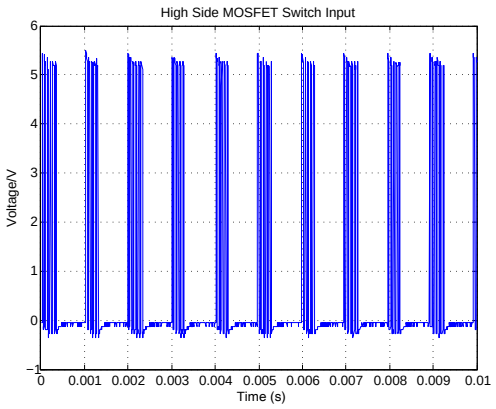


Figure 7: High side switching input with reference speed of 1500 RPM for rotor combo of hexTronik outrunner 1700 kV and 6030 propeller.

6 RESULTS AND DISCUSSIONS

The same reference speed from 4750 RPM to 5150 RPM was fed to the traditional ESC set as well as the FOC board with the PID parameters recursively tuned. The FOC scheme

http://www.imavs.org/

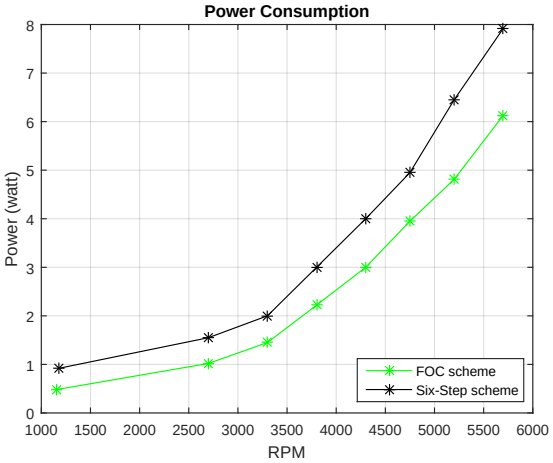


Figure 9: Power consumed by the motor under different speed for rotor combo of hexTronik outrunner 1700 kV and 6030 propeller.

shows a faster convergent period around 0.4 s compared to the six-step method. During the startup phase, the controller ramps up from zero to reference speed with a pre-positioning algorithm to achieve alignment. This is used to make sure the initial rotor position and check the reliability of rotor position and speed estimator.

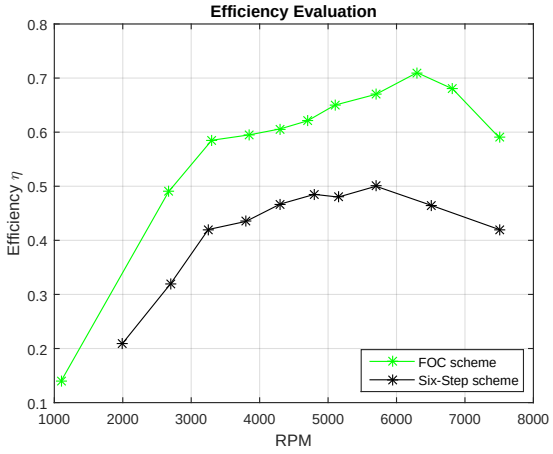


Figure 10: Efficiency evaluated at different rotating speed for rotor combo of BR $2804-1800$ kV and 6030 propeller.

Fig. 7 gives the signal generated from the module SVPWM to drive for the 3-phase invertor. As shown from the graph, the duty cycle of the PWM signal is updated each loop to adjust the vector during different sectors of SVM. High efficiency is the core advantage of FOC among other motor control techniques, which is shown in Fig. 9 and Fig. 10 respectively. Fig. 9 gives the data plot in power consumption

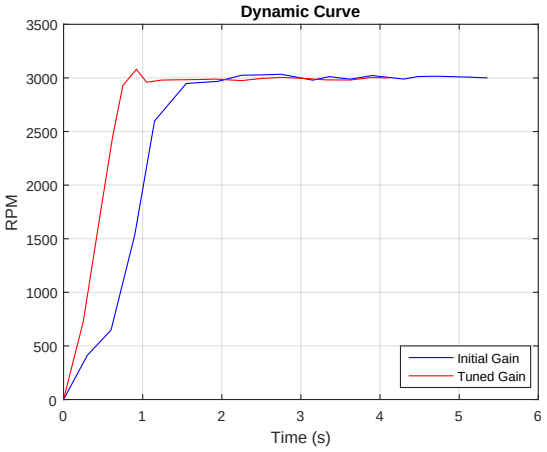


Figure 11: Start up process with a reference speed of 3000 RPM with hexTronik outrunner 1700 kV and 6030 propeller.

form for the rotor set hexTronik outrunner 1700 kV. Fig. 10 gives calculated efficiency with the same rotor set under FOC and six-step algorithm separately. Under different reference speed, the efficiency of FOC controller is about 15% higher than commercial six-step controller.

As we can see, another significant advantage of FOC controller among other commercial scalar control ESCs is the ability to maintain stable control at low speed reference. The experiment in Fig. 11 reveals that the FOC controller is able to maintain both motors at low reference speed and deliver significant torque at the same time. On the other hand, commercial controller cannot drive both BLDC motor to speed below 2000 RPM, compared to the performance of FOC scheme, which is indicated in the ramp-up process in Fig. 11. This is because the BEMF zero crossing detection method used in six-step algorithm is not accurate due to the electrical noise, especially at low commutation speed.

As current hardware setup does not support real-time measurement log file to record current and voltage, observation of measurements is from oscilloscope and external devices for rotational speed and torque. Thus it is impractical to identify an accurate dynamic model for model-based control. PI gains are tuned for both speed and torque loop based on the performance we have recorded, i.e., the overshoot and settling time.

7 CONCLUSION

This paper presented the development of a vector control method for BLDC motor and implementation of FOC control algorithm. Each core component of FOC algorithm is detailed with specific explanation of the rotor position and speed estimation with a nonlinear observer. The hardware system to implement the FOC algorithm is built from scratch. From the experiment results, the vector control method out-

http://www.imavs.org/

weighs the traditional scalar control method in the aspects of dynamic performance, transient properties, efficiency in power consumption and low reference speed control. In the next phase, we will identify an accurate dynamic model of the rotor system for implementation of a model-based control. More experiments will be done to verify the robustness and efficiency of the FOC method with long endurance tests and different rotor loads before integration into the UAS in the future.

REFERENCES

[1] P. Yedamale. Brushless DC (BLDC) motor fundamentals. *Microchip Technology Inc*, 20:3–15, 2003.

[2] Z. Chen, M. Tomita, S. Doki, and S. Okuma. An extended electromotive force model for sensorless control of interior permanent-magnet synchronous motors. *IEEE transactions on Industrial Electronics*, 50(2):288–295, 2003.

[3] T. H. Kim and M. Ehsani. Sensorless control of the bldc motors from near-zero to high speeds. *IEEE Transactions on Power Electronics*, 19(6):1635–1645, 2004.

[4] F. Genduso, R. Miceli, C. Rando, and G. R. Galluzzo. Back emf sensorless-control algorithm for high-dynamic performance pmsm. *IEEE Transactions on Industrial Electronics*, 57(6):2092–2100, 2010.

[5] J. Holtz, W. Lotzkat, and A. M. Khambadkone. On continuous control of PWM inverters in the overmodulation range including the six-step mode. *IEEE Transactions on Power Electronics*, 8(4):546–553, 1993.

[6] M. A. Abbas, R. Christen, and T. M. Jahns. Six-phase voltage source inverter driven induction motor. *IEEE Transactions on industry applications*, (5):1251–1259, 1984.

[7] N. Matsui and M. Shigyo. Brushless DC motor control without position and speed sensors. In *Industry Applications Society Annual Meeting, 1990., Conference Record of the 1990 IEEE*, pages 448–453. IEEE, 1990.

[8] J. Shao, D. Nolan, and T. Hopkins. A novel direct back EMF detection for sensorless brushless DC (BLDC) motor drives. In *Applied Power Electronics Conference and Exposition, 2002. APEC 2002. Seventeenth Annual IEEE*, volume 1, pages 33–37. IEEE, 2002.

[9] W. Zhao, M. Cheng, W. Hua, H. Jia, and R. Cao. Back-EMF harmonic analysis and fault-tolerant control of flux-switching permanent-magnet machine with redundancy. *IEEE Transactions on Industrial Electronics*, 58(5):1926–1935, 2011.

[10] R. H. Park. Two-reaction theory of synchronous machines generalized method of analysis-part i. *Transactions of the American Institute of Electrical Engineers*, 48(3):716–727, 1929.

[11] I. Takahashi and T. Noguchi. A new quick-response and high-efficiency control strategy of an induction motor. *IEEE Transactions on Industry applications*, (5):820–827, 1986.

[12] R. Gabriel, W. Leonhard, and C. J. Nordby. Field-oriented control of a standard AC motor using microprocessors. *IEEE transactions on industry applications*, (2):186–192, 1980.

[13] J. P. John, S. S. Kumar, and B. Jaya. Space vector modulation based field oriented control scheme for brushless DC motors. In *Emerging Trends in Electrical and Computer Technology (ICETECT), 2011 International Conference on*, pages 346–351. IEEE, 2011.

[14] J. Shao. *Direct back EMF detection method for sensorless brushless DC (BLDC) motor drives*. PhD thesis, Virginia Polytechnic Institute, 2003.

[15] J. Shao, D. Nolan, M. Teissier, and D. Swanson. A novel microcontroller-based sensorless brushless DC (BLDC) motor drive for automotive fuel pumps. *Industry Applications, IEEE Transactions on*, 39(6):1734–1740, 2003.

[16] D. M. Erdman. Control system, method of operating an electronically commutated motor, and laundering apparatus, March 31 1987. US Patent 4,654,566.

[17] K. Iizuka, H. Uzuhashi, M. Kano, T. Endo, and K. Mohri. Microcomputer control for sensorless brushless motor. *Industry Applications, IEEE Transactions on*, (3):595–601, 1985.

[18] G. N. Mahesh, Y. Kiran, and D. Parthasarthy. Modelling of buck DC-DC converter using simulink. *International Journal of Innovative Research in Science Engineering and Technology*, 3(7):14965–14975, 2014.

[19] L. Hoang. Comparison of field-oriented control and direct torque control for induction motor drives. In *Industry Applications Conference, 1999. Thirty-Fourth IAS Annual Meeting. Conference Record of the 1999 IEEE*, volume 2, pages 1245–1252. IEEE, 1999.

[20] R. Ortega, L. Praly, A. Astolfi, J. Lee, and K. Nam. Estimation of rotor position and speed of permanent magnet synchronous motors with guaranteed stability. *IEEE Transactions on Control Systems Technology*, 19(3):601–614, 2011.

[21] J. Lee, J. Hong, K. Nam, R. Ortega, L. Praly, and A. Astolfi. Sensorless control of surface-mount permanent-magnet synchronous motors based on a nonlinear observer. *IEEE Transactions on power electronics*, 25(2):290–297, 2010.

http://www.imavs.org/

Incremental Model Based Heuristic Dynamic Programming for Nonlinear Adaptive Flight Control

Y. Zhou *, E. van Kampen, and Q. P. Chu
Delft University of Technology, 2629HS Delft, The Netherlands

http://www.imavs.org/

ABSTRACT

This paper presents a new and effective approach, incremental model based heuristic dynamic programming, to design an adaptive near-optimal controller without a-prior knowledge of the dynamic model. Both traditional heuristic dynamic programming algorithm and incremental model based heuristic dynamic programming algorithm are provided and applied to an illustrative on-line learning task. The system dynamics are completely unknown at the beginning, and the agent learns the local system models and the control policies on-line to follow a reference signal. It was found that using incremental models in heuristic dynamic programming can avoid off-line learning of the system model and help to accelerate the on-line learning. This proposed method can potentially design a near-optimal controller for autonomous flight of unmanned aerial vehicles without a-prior knowledge of the system dynamics.

1 INTRODUCTION

Control of a complex, nonlinear flying vehicle without sufficient knowledge of the system dynamics is a challenging problem to maintain functionality and safety in aviation. Until recent decades, adaptive control methods allow certain levels of robustness and fault-tolerance to be achieved. These methods in some form or another rely on off-line or/and on-line identification of air vehicles' dynamics and adaptation of control laws when necessary. However, on-line identification of unknown dynamical systems is not a trivial task especially when the system is complex and highly nonlinear.

In recent years, Adaptive/Approximate Dynamic Programming (ADP), which obtains approximately optimal solutions of the Hamilton-Jacobi-Bellman (HJB) equations, has been actively researched to solve nonlinear, optimal, fault-tolerant control problems [1, 2, 3, 4, 5]. Different from traditional Reinforcement Learning (RL) methods, ADP applies a function approximator with parameters to approximate the value/cost function to solve optimality problems with large or continuous state spaces on-line and to tackle the 'curse of dimensionality', which traditional RL methods often confront with [6, 7]. Adaptive Critic Designs (ACDs), which are also known as actor-critic designs, constitute a class of ADP methods that separate evaluation and improvement using parametric structures [2].

The most basic form and widely used structure of ACD is Heuristic Dynamic Programming (HDP). An action independent heuristic dynamic programming controller consists of an actor, a critic and an approximated plant structure connected between the actor and the critic [2, 6, 7]. An alternative approach is Action Dependent Heuristic Dynamic Programming (ADHDP), which does not need plant approximation, but has a direct connection from the output of the actor network to the input of the critic network. However, from the theoretical perspective, the actor output is not necessarily an input to the critic for estimating the optimal value function. From the practical point of view, extra input will increase the complexity of the critic network. Furthermore, some research has investigated the difference between HDP and ADHDP, and found that HDP controller with the approximated plant dynamics can operate in a wider range of flight conditions and has a higher success learning ratio in controlling an F-16 model [8]. Therefore, in this paper, only HDP, which refers to action independent heuristic dynamic programming, is considered.

Neural networks are most widely used as function approximators to approximate plants. However, this method has two main drawbacks which may lead to failure when applied in practice. First, on-line identification of the plant using neural networks needs certain time to approximate feasible model, which may even need an off-line identification beforehand. Second, neural networks may add two sources of errors. One is lacking adequate computing power when neural networks are used to perform the least-square approximation of the desired cost-to-go function. Another is that the function approximator is trained from a simulation model which might not be correct due to the unknown system [9].

Incremental methods are able to deal with system nonlinearity. These methods compute the required control increment instead of the total control input. However, some parts of the system model are still required in order to complete the design process [10, 11, 12]. Incremental Approximate Dynamic Programming (iADP) was developed for the first time to control nonlinear unknown systems without using models. This control strategy uses a quadratic function to approximate the value function [13, 14].

In this paper, an action independent Heuristic Dynamic

*Email address(es): Y.Zhou-6@tudelft.nl

http://www.imavs.org/

Programming controller using incremental models, which is named *Incremental model based Heuristic Dynamic Programming* (IHDP), is developed as a model-free adaptive control approach for nonlinear unknown systems. This is called a model-free approach, because it does not need any a priori model information at the beginning of the algorithm nor online identification of nonlinear systems, but only the on-line identified linear incremental model. The incremental form of a nonlinear dynamic system is actually a linear time-varying approximation of the original system assuming sufficiently high sample rate for discretization. As the plant to be controlled in this paper is nonlinear, the IHDP is therefore developed based on the linearized incremental model of the original nonlinear system. This algorithm can be seen as an extension to the algorithm developed in [13, 14] with more general value function approximaters.

The rest of the paper is structured as follows. An HDP algorithm with a widely used neural network plant approximator is introduced and designed in section 2. An IHDP algorithm using incremental approach is first presented in section 3. Then, in section 4, the two algorithms are applied to an illustrative application, and the results are compared and discussed, showing how much the IHDP method can improve the performance. The last part concludes the advantages and disadvantages of using the incremental approach with HDP, and addresses the challenges and possibilities of the future research.

2 HEURISTIC DYNAMIC PROGRAMMING

Similar to other ADP methods, action independent Heuristic Dynamic Programming (HDP) algorithms operate by alternating between two steps: policy evaluation, implemented by the critic, and policy improvement, implemented by the actor [14, 9]. Fig.1 is a schematic diagram of an HDP controller, which uses 3 Neural Networks to approximate actor, critic, and system dynamics with weights w_a, w_c, and w_m, respectively.

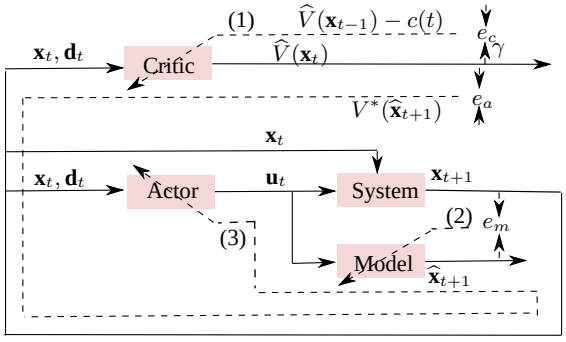


Figure 1: Architecture of HDP using Neural Networks to approximate system model

2.1 HDP using Neural Networks to approximate system model

2.1.1 Critic

The critic network is used to approximate the state-value function $V(x_t)$, which is the cumulative future rewards from any initial state $\mathbf{x}_t$:

$$V^{\mu}(\mathbf{x}_t) = \sum_{i=t}^{\infty} \gamma^{i-t} c_t, \tag{1}$$

where μ is the *current policy* for this algorithm, γ is called *discount factor* or *forgetting factor*, which is a scalar with $0 < \gamma < 1$, and c_t is the *one-step cost function*. The discount factor ensures that the cost for any state is finite and provides a reasonable evaluation and approximation to infinite-horizon problems as well as problems involving a finite but very large number of stages. By adjusting γ, it is able to control the extent to which the short-term cost or long-term cost is concerned [9].

To minimize the cost of the system approaching its goal, the one-step cost function is defined quadratically as a function of the difference between the current state and the desired state, as follows:

$$c_t = c(\mathbf{x}_t, \mathbf{d}_t) = (\mathbf{x}_t - \mathbf{d}_t)^T Q (\mathbf{x}_t - \mathbf{d}_t), \tag{2}$$

where $\mathbf{d}_t$ is the reference track, Q is a positive definite matrix. To normalize the effect of each state, we usually use normalization factors in the Q matrix. Thus, we let Q be a diagonal matrix, and Eq. 2 can be rewritten as follows:

$$c_t = \sum_{i=1}^{n} (\zeta_i)^2 \cdot \left(\frac{x_{t,i} - d_{t,i}}{x_{max,i}} \right)^2, \tag{3}$$

where ζ_i is a given weight to indicate the importance of the cost for the i-th state approaching the desired track.

Actor-Critic methods are on-policy Temporal Difference (TD) methods, which continually estimate the cost-to-go for the current policy by updating the critic, and change the policy towards greediness by updating the actor at the same time [15]. The evaluation of the critic is the TD error:

$$e_c(t) = c_{t-1} + \gamma \widehat{V}(\mathbf{x}_t) - \widehat{V}(\mathbf{x}_{t-1}), \tag{4}$$

where $\widehat{V}(\mathbf{x}_t)$ is the approximated cost-to-go from state $\mathbf{x}_t$ under current policy. Note that $\widehat{V}$ is a function of $\mathbf{x}_t$ and $\mathbf{w}_c(t)$ with a static neural network structure. The target for the critic update is $c_{t-1} + \gamma \widehat{V}(\mathbf{x}_t)$.

The critic network tries to minimize the defined error function:

$$E_c(t) = \frac{1}{2} e_c^2(t). \tag{5}$$

Therefore, the weights of the critic network are updated according to a gradient-descent algorithm with a learning rate η_c:

$$\mathbf{w}_c(t+1) = \mathbf{w}_c(t) + \Delta \mathbf{w}_c(t), \tag{6}$$

where

$$\begin{aligned}\Delta \mathbf{w}_c(t) &= -\eta_c \cdot \frac{\partial E_c(t)}{\partial \mathbf{w}_c(t)} \\ &= -\eta_c \cdot \frac{\partial E_c(t)}{\partial \widehat{V}(\mathbf{x}_t)} \cdot \frac{\partial \widehat{V}(\mathbf{x}_t)}{\partial \mathbf{w}_c(t)}.\end{aligned} \tag{7}$$

With a fixed policy and converged critic, the neural network parameters $\mathbf{w}_c(t)$ will be constant.

2.1.2 Actor

The actor is used to find the policy which minimizes the difference between the defined cost-to-go function $\widehat{V}(\mathbf{x}_t)$ and the goal $V^*(t)$:

$$E_a(t) = \frac{1}{2} e_a^2(t), \tag{8}$$

$$e_a(t) = \widehat{V}(\mathbf{x}_t) - V^*(t), \tag{9}$$

where the goal $V^*(t)$ is set to 0.

The policy is determined by the weights of the actor network. However, updating the actor network is more complicated, since it involves the critic network and the model network. Fig. 1 shows that, through the 3rd back-propagation direction, the actor weights affect cost-to-go function $V(\mathbf{x}_{t+1})$ through affecting $\mathbf{x}_{t+1}$ and $\mathbf{u}_t$. Thus, the actor network weights can be updated according to the gradient-descent algorithm with a learning rate η_a:

$$\mathbf{w}_a(t+1) = \mathbf{w}_a(t) + \Delta \mathbf{w}_a(t), \tag{10}$$

where

$$\begin{aligned}\Delta \mathbf{w}_a(t) &= -\eta_a \cdot \frac{\partial E_a(t+1)}{\partial \mathbf{w}_a(t)} \\ &= -\eta_a \cdot \frac{\partial E_a(t+1)}{\partial \widehat{V}(\mathbf{x}_{t+1})} \frac{\partial \widehat{V}(\mathbf{x}_{t+1})}{\partial \mathbf{x}_{t+1}} \frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t} \frac{\partial \mathbf{u}_t}{\partial \mathbf{w}_a(t)}.\end{aligned} \tag{11}$$

The approximated system model can be used to estimate the next state $\mathbf{x}_{t+1}$ with an input $\mathbf{u}_t$. This helps to get the useful term $\frac{\partial \widehat{\mathbf{x}}(t+1)}{\partial \mathbf{u}_t}$ approximating $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t}$ in updating the actor network [8]. Therefore, Eq. 11 can be rewritten as follows:

$$\Delta \mathbf{w}_a(t) = -\eta_a \frac{\partial E_a(t+1)}{\partial \widehat{V}(\widehat{\mathbf{x}}_{t+1})} \frac{\partial \widehat{V}(\widehat{\mathbf{x}}_{t+1})}{\partial \widehat{\mathbf{x}}_{t+1}} \frac{\partial \widehat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}(t)} \frac{\partial \mathbf{u}(t)}{\partial \mathbf{w}_a(t)}. \tag{12}$$

2.1.3 Model

The model network approximates the system dynamics and gives the estimated next state $\widehat{\mathbf{x}}_{t+1}$ as output. The next state is a function of input $\mathbf{u}_t$ and the network parameters $\mathbf{w}_m(t)$ with a fixed neural network structure: $\widehat{\mathbf{x}}_{t+1}(\mathbf{u}_t, \mathbf{w}_m(t))$. The update of the model network is by minimizing the difference between the measured state $\mathbf{x}_t$ and the estimated state $\widehat{\mathbf{x}}_t$:

$$E_m(t) = \frac{1}{2} e_m^2(t), \tag{13}$$

where

$$e_m(t) = \mathbf{x}_t - \widehat{\mathbf{x}}_t. \tag{14}$$

The model network weights are updated according to the gradient-descent algorithm with a learning rate η_m:

$$\mathbf{w}_m(t+1) = \mathbf{w}_m(t) + \Delta \mathbf{w}_m(t), \tag{15}$$

where

$$\begin{aligned}\Delta \mathbf{w}_m(t) &= -\eta_m \cdot \frac{\partial E_m(t+1)}{\partial \mathbf{w}_m(t)} \\ &= -\eta_m \cdot \frac{\partial E_m(t+1)}{\partial \widehat{\mathbf{x}}_{t+1}} \frac{\partial \widehat{\mathbf{x}}_{t+1}}{\partial \mathbf{w}_m(t)}.\end{aligned} \tag{16}$$

2.2 HDP training by back-propagation

The actor, critic and model neural networks in this paper are all Multilayer Perceptrons (MLP), which consist of multiple, fully connected, and feedforward layers of nodes. Each neural network has an input layer, a hidden layer and an output layer. Each node in hidden layers is a neuron with a continuous, nonlinear hyperbolic tangent activation function σ:

$$\sigma(y) = \frac{1 - e^{-y}}{1 + e^{-y}}. \tag{17}$$

At each point, it has a positive derivative:

$$\frac{\partial \sigma(y)}{\partial y} = \frac{1}{2}(1 - \sigma(y)^2). \tag{18}$$

In fully connected multilayer neural networks, the input of the $(n+1)$-th layer consists of the outputs of the n-th layer and sometimes also a bias term b_n. When the neural network has I inputs, J hidden neurons, and K outputs, the neural network weight from i-th input layer neuron to j-th hidden layer neuron is $w_{ji}(i = 1, ..., I+1, j = 1, ..., J)$, and the weight from j-th hidden layer neuron to k-th output layer neuron is $w_{kj}(j = 1, ..., J+1, k = 1, ..., K)$. Thus, the feedforward neural networks from i-th input layer (noted as superscript in) neuron to j-th neuron of the hidden layer (noted as superscript hi) can be described as follows:

$$\sigma_j^{hi}(t) = \frac{1 - e^{-y_j^{hi}(t)}}{1 + e^{-y_j^{hi}(t)}}, \tag{19}$$

$$y_j^{hi}(t) = \sum_{i=1}^{I+1} w_{ji}^{hi}(t) x_i^{in}(t), \tag{20}$$

where, $\sigma_j^{hi}(t)$ is the output of the j-th hidden layer neuron, $y_j^{hi}(t)$ is the network input of the j-th hidden layer neuron, $w_{ji}^{hi}(t)$ is the weight from the i-th input neuron to j-th hidden layer neuron at time t, and $x_i^{in}(t)$ is the i-th input of the hidden layer, which consists I inputs of the system and the bias term b^{in}. The feedforward neural networks from j-th hidden

http://www.imavs.org/

http://www.imavs.org/

layer neuron to k-th output layer (noted as superscript out) neuron can be described as follows:

$$y_k^{out}(t) = \sum_{j=1}^{J+1} w_{kj}^{out}(t) x_j^{hi}(t), \tag{21}$$

where, $y_k^{out}(t)$ is the output of the k-th output layer neuron, $w_{kj}^{out}(t)$ is the weight from the j-th hidden layer neuron to k-th output layer neuron at time t, and $x_j^{hi}(t)$ is the jth input of output layer, which consists J outputs of the hidden layer neurons and a bias term b^{hi}.

Because the output of a hyperbolic tangent function is bounded with $(-1, 1)$, and the outputs of the neural network is a summation with parameters, the neural network with bias items can approximate any value theoretically. Thus, the output of the neural networks is written as $\mathbf{O}(t)$:

$$\begin{aligned} \mathbf{O}(t) &= \mathbf{y}^{out}(t) \\ &= [y_1^{out}(t), y_2^{out}(t), ..., y_K^{out}(t)]^T. \end{aligned} \tag{22}$$

2.2.1 Critic and Model

To update the critic and model network weights (through the 1st and 2nd back-propagation directions in Fig. 1) according to Eq. 7 and Eq. 16, the partial derivative of the network output with respect to the network weights is needed:

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial \mathbf{O}(t)} \frac{\partial \mathbf{O}(t)}{\partial w(t)}. \tag{23}$$

To be more specific, the partial derivative of each network output $y_k^{out}(t)$ with respect to the network weights from the hidden layer to the output layer $w_{kj}^{out}(t)$ and the weights from the input layer to the hidden layer $w_{ji}^{hi}(t)$ is shown below, respectively:

$$\frac{\partial y_k^{out}(t)}{\partial w_{kj}^{out}(t)} = x_j^{hi}(t), \tag{24}$$

$$\begin{aligned} \frac{\partial y_k^{out}(t)}{\partial w_{ji}^{hi}(t)} &= \frac{\partial y_k^{out}(t)}{\partial \sigma_j^{hi}(t)} \cdot \frac{\partial \sigma_j^{hi}(t)}{\partial y_j^{hi}(t)} \cdot \frac{\partial y_j^{hi}(t)}{\partial w_{ji}^{hi}(t)} \\ &= w_{kj}^{out}(t) \cdot \frac{1}{2}(1 - \sigma_j^{hi}(t)^2) \cdot x_i^{in}(t). \end{aligned} \tag{25}$$

2.2.2 Actor

To update the actor network weights, the network error E_a go through the critic network, model network, and finally the actor network along the 3rd back-propagation directions in Fig. 1. Thus, the partial derivative of the network output with respect to the network input $x_i^{in}(i = 1, ..., I)$ is also needed for the term $\frac{\partial \widehat{V}(\widehat{\mathbf{x}}_{t+1})}{\partial \widehat{\mathbf{x}}_{t+1}}$ and $\frac{\partial \widehat{\mathbf{x}}_{t+1}}{\partial \mathbf{u}(t)}$ in Eq. 12 to update the weights:

$$\frac{\partial \mathbf{O}(t)}{\partial \mathbf{x}^{in}} = \frac{\partial \mathbf{O}(t)}{\mathbf{y}^{out}(t)} \frac{\mathbf{y}^{out}(t)}{\partial \mathbf{x}^{in}}. \tag{26}$$

The partial derivative of the k-th output with respect to the i-th input can be calculated as follows:

$$\begin{aligned} \frac{\partial y_k^{out}(t)}{\partial x_i^{in}} &= \sum_{j=1}^{J} \left[\frac{\partial y_k^{out}(t)}{\partial \sigma_j^{hi}(t)} \cdot \frac{\partial \sigma_j^{hi}(t)}{\partial x_i^{in}} \right] \\ &= \sum_{j=1}^{J} \left[w_{kj}^{out}(t) \cdot \frac{1}{2}(1 - \sigma_j^{hi}(t)^2) w_{ji}^{hi}(t) \right]. \end{aligned} \tag{27}$$

3 INCREMENTAL HEURISTIC DYNAMIC PROGRAMMING

Fig.2 is the diagram for implementation of an Incremental model based Heuristic Dynamic Programming (IHDP) controller. It uses 2 Neural Networks to approximate the actor and critic with weights w_a and w_c, and an incremental model to find the system dynamics at a certain moment. The updating of the critic weights is the same as updating critic in HDP algorithm, which will not be reiterated in this section. This section focus on the new idea of using the incremental approach to approximate $\frac{\partial \mathbf{x}(t+1)}{\partial u(t)}$ in Eq. 11 as part of updating actor.

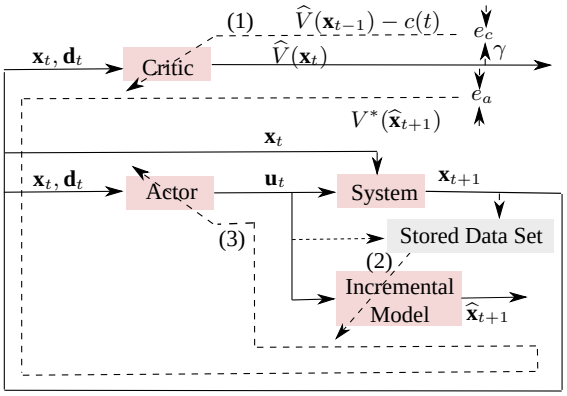


Figure 2: Architecture of HDP using incremental approach to approximate system model

3.1 HDP using incremental approach

3.1.1 Incremental Model

Many physical system, such as aircraft, are highly non-linear and can be generally given as follows:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t)], \tag{28}$$

$$\mathbf{y}(t) = h[\mathbf{x}(t)], \tag{29}$$

where Eq. 28 is the *kinematic state equation*, in which $f[\mathbf{x}(t), \mathbf{u}(t)] \in \mathcal{R}^n$ provides the physical evaluation of the state vector over time, Eq. 29 is the *output (observation) equation*, which can be measured using sensors, $h[\mathbf{x}(t)] \in \mathcal{R}^p$ is a vector denoting the measured output.

http://www.imavs.org/

The system dynamics around the condition of the system at time t_0 can be linearized approximately by using the first-order Taylor series expansion:

$$\begin{aligned}\dot{\mathbf{x}}(t) &\simeq f[\mathbf{x}(t_0),\mathbf{u}(t_0)] \\ &+ \frac{\partial f[\mathbf{x}(t),\mathbf{u}(t)]}{\partial \mathbf{x}(t)}|_{\mathbf{x}(t_0),\mathbf{u}(t_0)}[\mathbf{x}(t)-\mathbf{x}(t_0)] \\ &+ \frac{\partial f[\mathbf{x}(t),\mathbf{u}(t)]}{\partial \mathbf{u}(t)}|_{\mathbf{x}(t_0),\mathbf{u}(t_0)}[\mathbf{u}(t)-\mathbf{u}(t_0)] \\ &= \dot{\mathbf{x}}(t_0) + F[\mathbf{x}(t_0),\mathbf{u}(t_0)][\mathbf{x}(t)-\mathbf{x}(t_0)] \\ &+ G[\mathbf{x}(t_0),\mathbf{u}(t_0)][\mathbf{u}(t)-\mathbf{u}(t_0)],\end{aligned} \tag{30}$$

where $F[\mathbf{x}(t),\mathbf{u}(t)] = \frac{\partial f[\mathbf{x}(t),\mathbf{u}(t)]}{\partial \mathbf{x}(t)} \in \mathcal{R}^{n\times n}$ is the *system matrix* at time t, and $G[\mathbf{x}(t),\mathbf{u}(t)] = \frac{\partial f[\mathbf{x}(t),\mathbf{u}(t)]}{\partial \mathbf{u}(t)} \in \mathcal{R}^{n\times m}$ is the *control effectiveness matrix* at time t.

We assume that the states and state derivatives of the system are measurable, which means $\Delta\dot{\mathbf{x}}(t), \Delta\mathbf{x}(t), \Delta\mathbf{u}(t)$ are measurable. Under this assumption, the model around time t_0 can be written in the incremental form:

$$\begin{aligned}\Delta\dot{\mathbf{x}}(t) \simeq F[\mathbf{x}(t_0),\mathbf{u}(t_0)]\Delta\mathbf{x}(t) \\ + G[\mathbf{x}(t_0),\mathbf{u}(t_0)]\Delta\mathbf{u}(t).\end{aligned} \tag{31}$$

This current incremental model can be identified using least squares (LS) techniques and can be used to obtain an approximated value of $\frac{\partial \mathbf{x}(t+1)}{\partial u(t)}$ without using model networks in previous section.

The physical systems are generally continuous, but the data we collect are discrete samples. We assume that the control system has a constant, sufficiently high sampling frequency. With this constant data sampling rate, the non-linear system can be written in a discrete form:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t,\mathbf{u}_t), \tag{32}$$

$$\mathbf{y}_t = h(\mathbf{x}_t), \tag{33}$$

where $f(\mathbf{x}_t,\mathbf{u}_t) \in \mathcal{R}^n$ provides the *system dynamics*, and $h(\mathbf{x}_t) \in \mathcal{R}^p$ is a vector denoting the measuring system.

By taking the Taylor expansion, we can get the system dynamics linearized around x_0:

$$\begin{aligned}\mathbf{x}_{t+1} &= f(\mathbf{x}_t,\mathbf{u}_t) \\ &\simeq f(\mathbf{x}_0,\mathbf{u}_0) + \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}|_{\mathbf{x}_0,\mathbf{u}_0}(\mathbf{x}_t-\mathbf{x}_0) \\ &+ \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}|_{\mathbf{x}_0,\mathbf{u}_0}(\mathbf{u}_t-\mathbf{u}_0).\end{aligned} \tag{34}$$

When Δt is very small, $\mathbf{x}_{t-1}$ approximates $\mathbf{x}_t$. Thus, $\mathbf{x}_0, \mathbf{u}_0$ in Eq. 34 can be replaced by $\mathbf{x}_0 = \mathbf{x}_{t-1}$ and $\mathbf{u}_0 = \mathbf{u}_{t-1}$, and we obtain the discrete incremental form of this non-linear system:

$$\begin{aligned}\mathbf{x}_{t+1} - \mathbf{x}_t \simeq F(\mathbf{x}_{t-1},\mathbf{u}_{t-1})(\mathbf{x}_t-\mathbf{x}_{t-1}) \\ + G(\mathbf{x}_{t-1},\mathbf{u}_{t-1})(\mathbf{u}_t-\mathbf{u}_{t-1}),\end{aligned} \tag{35}$$

$$\Delta\mathbf{x}_{t+1} \simeq F(\mathbf{x}_{t-1},\mathbf{u}_{t-1})\Delta\mathbf{x}_t + G(\mathbf{x}_{t-1},\mathbf{u}_{t-1})\Delta\mathbf{u}_t, \tag{36}$$

where $F(\mathbf{x}_{t-1},\mathbf{u}_{t-1}) = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathcal{R}^{n\times n}$ is the *system transition matrix*, and $G(\mathbf{x}_{t-1},\mathbf{u}_{t-1}) = \frac{\partial f(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}|_{\mathbf{x}_{t-1},\mathbf{u}_{t-1}} \in \mathcal{R}^{n\times m}$ is the *control effectiveness matrix* at time step $t-1$. Because of the high frequency sample data and the relatively slow-variant system assumption, the current linearized model can be identified by using the measured data in previous M steps.

3.1.2 Actor

The structure of the actor is the same as the one in HDP controller. It is used to minimize the difference between the cost-to-go function $\widehat{V}(\mathbf{x}_t)$ and the goal $V^*(t)$. However, updating this actor network is easier and faster than the one in HDP controller, since it involves a critic network and an incremental model.

Through the 3rd back propagation direction in Fig. 2, the actor weights affect cost-to-go function $V(\mathbf{x}_{t+1})$ also through affecting $\mathbf{x}_{t+1}$ and $\mathbf{u}_t$. The actor network weights can be updated according to the gradient-descent algorithm as shown in Eq. 10 and Eq. 11. The incremental model of the system can be used to approximate the derivative of the next state with respect to the input, $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t}$.

3.2 IHDP training by back-propagation and incremental model identification

3.2.1 Incremental Model

Since $\Delta\mathbf{x}(t), \Delta\mathbf{u}(t)$ are measurable as assumed, F_{t-1}, G_{t-1} are identifiable by using the simple equation error method:

$$\begin{aligned}\Delta x_{i,t-k+1} &= \mathbf{f}_i\Delta\mathbf{x}_{t-k} + \mathbf{g}_i\Delta\mathbf{u}_{t-k} \\ &= \begin{bmatrix}\Delta\mathbf{x}_{t-k}^T & \Delta\mathbf{u}_{t-k}^T\end{bmatrix}\begin{bmatrix}\mathbf{f}_i^T \\ \mathbf{g}_i^T\end{bmatrix},\end{aligned} \tag{37}$$

where $\Delta x_{i,t-k+1} = x_{i,t-k+1} - x_{i,t-k}$ is the increment of ith state element, $\mathbf{f}_i$ and $\mathbf{g}_i$ are the elements of ith row vector of F_{t-1}, G_{t-1}, and $k = 1, 2...M$ denotes at which time the historic information is available. Because there are $n+m$ parameters in the ith row, M needs to satisfy $M \geq (n+m)$. By using the Ordinary Least Squares (OLS) method, the linearized system dynamics (ith row) can be identified from M different data points:

$$\begin{bmatrix}\mathbf{f}_i^T \\ \mathbf{g}_i^T\end{bmatrix} = (\mathbf{A}_t^T\mathbf{A}_t)^{-1}\mathbf{A}_t^T\mathbf{y}_t, \tag{38}$$

where

$$\mathbf{A}_t = \begin{bmatrix}\Delta\mathbf{x}_{t-1}^T & \Delta\mathbf{u}_{t-1}^T \\ \vdots & \vdots \\ \Delta\mathbf{x}_{t-M}^T & \Delta\mathbf{u}_{t-M}^T\end{bmatrix},\ \mathbf{y}_t = \begin{bmatrix}\Delta x_{i,t} \\ \vdots \\ \Delta x_{i,t-M+1}\end{bmatrix}. \tag{39}$$

Choosing a suitable number of data M is also important. The identified incremental model will be used and is only capable of describing the system behaviour within a small time

range. When M is very large, the identified model may not represent the local linearized model of the nonlinear system. However, when M is too small, the linear system might be ill-conditioned, especially at which point the excitation is not sufficient. Thus, choosing M depends not only on the sampling frequency and non-linearity, but also the intensity of the excitation. In this paper, M is chosen to be $2 \cdot (n+m)$.

3.2.2 Actor

To update the actor network weights, the network error E_a go through the critic network, incremental model, and the actor network consequently along the 3rd back-propagation directions in Fig. 2. The partial derivative of the network output with respect to the network weights (Eq. 23 to 25) and the partial derivative of the network output with respect to the network input (Eq. 26, 27) are needed for the term $\frac{\partial \widehat{V}(\mathbf{x}_{t+1})}{\partial \mathbf{x}_{t+1}}$ and term $\frac{\partial \mathbf{u}_t}{\partial \mathbf{w}_a(t)}$ in Eq. 11, respectively.

With the linearized incremental model, the term $\frac{\partial \mathbf{x}(t+1)}{\partial u(t)}$ can be easily approximated:

$$\begin{aligned}\frac{\partial \mathbf{x}(t+1)}{\partial u(t)} &\simeq \partial[\mathbf{x}_t + F(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})(\mathbf{x}_t - \mathbf{x}_{t-1}) \\ &\quad + G(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})(\mathbf{u}_t - \mathbf{u}_{t-1})]/\partial u(t) \\ &= G(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}).\end{aligned} \tag{40}$$

This method simplified the approach of updating actor network weights and accelerate the learning with direct on-line identification of the incremental model.

4 APPLICATION

This section will present an illustrative application of both the HDP and the IHDP algorithms on a simulation model for validation. The flight control task is to track a changing reference, when there is input disturbances, which is a most basic and important control task for air vehicles.

4.1 Air vehicle model

A nonlinear air vehicle simulation model will be used in this section. Air vehicle models are highly nonlinear and can be generally given as follows:

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t), \mathbf{u}(t) + \mathbf{w}(t)], \tag{41}$$

$$\mathbf{y}(t) = h[\mathbf{x}(t)], \tag{42}$$

where Eq. 41 is the kinematic state equation, $\mathbf{w}(t)$ is the external disturbace, which is set to be caused only by the input noise, and Eq. 42 is the output equation.

As an application for these control algorithms, only elevator deflection will be regulated as pitch control to stabilize the air vehicles. Thus, we are interested in two longitudinal states, *angle of attack* α and *pitch rate* q (i.e. the system variables are $\mathbf{x} = [\alpha \;\; q]$), and one control input, *elevator deflection angle* δ_e.

The nonlinear model in the pitch plane is simulated around a steady wings-level flight condition:

$$\dot{\alpha} = q + \frac{\bar{q}S}{m_a V_T} C_z(\alpha, q, M_a, \delta_e), \tag{43}$$

$$\dot{q} = \frac{\bar{q}Sd}{I_{yy}} C_m(\alpha, q, M_a, \delta_e), \tag{44}$$

where $\bar{q}$ is dynamic pressure, S is reference area, m_a is mass, V_T is speed, d is reference length, I_{yy} is pitching moment of inertia, C_z is the aerodynamic force coefficient, and C_m is the aerodynamic moment coefficient. C_z and C_m are highly nonlinear functions of angle of attack α, pitch rate q, Mach number M_a and elevator deflection δ_e.

As a preliminary test, an air vehicle model (parameter data) is taken in the pitch plane for $-10^o < \alpha < 10^o$ [16, 17]:

$$\begin{aligned}
&C_z(\alpha, q, M_a, \delta_e) = C_{z1}(\alpha, M_a) + B_z \delta_e, \\
&C_m(\alpha, q, M_a, \delta_e) = C_{m1}(\alpha, M_a) + B_m \delta_e, \\
&B_z = b_1 M_a + b_2, \\
&B_m = b_3 M_a + b_4, \\
&C_{z1}(\alpha, M_a) = \phi_{z1}(\alpha) + \phi_{z2} M_a, \\
&C_{z2}(\alpha, M_a) = \phi_{m1}(\alpha) + \phi_{m2} M_a, \\
&\phi_{z1}(\alpha) = h_1 \alpha^3 + h_2 \alpha|\alpha| + h_3 \alpha, \\
&\phi_{m1}(\alpha) = h_4 \alpha^3 + h_5 \alpha|\alpha| + h_6 \alpha, \\
&\phi_{z2} = h_7 \alpha|\alpha| + h_8 \alpha, \\
&\phi_{m2} = h_9 \alpha|\alpha| + h_{10} \alpha,
\end{aligned} \tag{45}$$

where $b_1, ..., b_4$, $h_1, ..., h_{10}$ are validated constant coefficients in the flight envelop [17].

To accomplish the reference tracking task, an adaptive controller with the actor need to be found out by minimizing the cost-to-go function $V(\mathbf{x}_t)$ with a feasible critic and model.

4.2 Results and Discussions

Two algorithms are applied to this problem: traditional HDP uses a neural network to approximate the plant model and IHDP uses the incremental approach. The identified models are used 1) to predict the next states, which is used to estimate the cost-to-go of the next state and its difference from the minimal cost, and 2) to estimate the control effective matrix, which is used to update the actor during the error back-propagation.

Fig.3 shows the one-step prediction of α and q, when there is a sine input excitation, using the on-line identified neural network model and the incremental model. As illustrated in Fig.3 (a), the one-step state predictions using both methods are feasible. However, the prediction using the neural network needs more time to learn at the beginning. When having a close look at the prediction errors with Fig.3 (b) and (c), the prediction using incremental approach has significantly higher precision. Fig.4 presents the identification

http://www.imavs.org/

http://www.imavs.org/

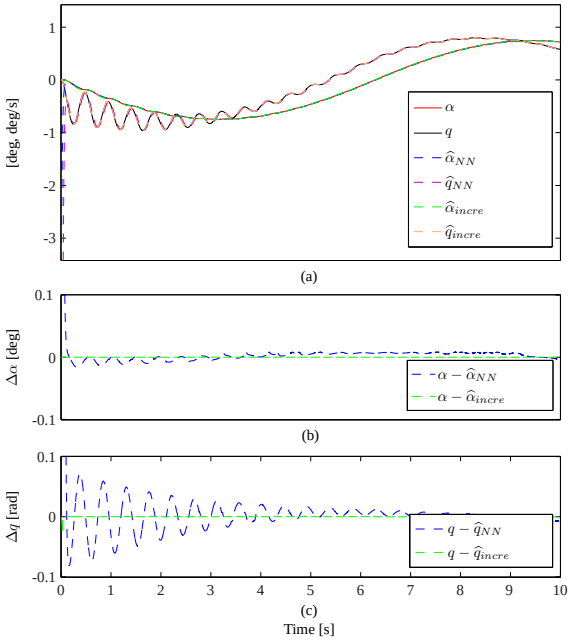


Figure 3: One-step prediction with on-line identified model using neural networks and incremental approach.

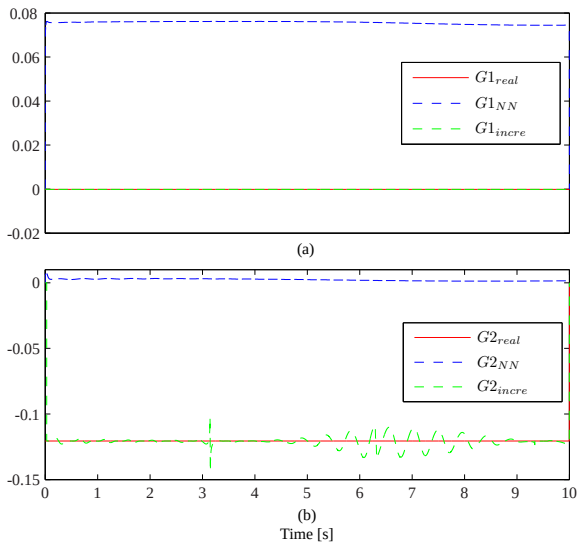


Figure 4: On-line identified control effective matrix using neural networks and incremental approach.

results of the control effective matrix, $G = [G1, G2]'$, using the two methods. It is apparent that the incremental method has a substantially improved identified control effective matrix, which is to approximate an important term $\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{u}_t}$ in updating Actors.

Fig.5 illustrates the performance of the traditional HDP method and IHDP method when applied to an on-line tracking problem. Comparing to traditional HDP method, IHDP method can identify the local model and reject the disturbance much quicker at the beginning, and follow the reference signal more precisely. A slow on-line training of model neural network of traditional HDP may lead to a large overshoot and lost control at the initial stage. In realistic cases, traditional HDP needs an off-line training of the model before on-line training of the controller to prevent failures. On the other hand, IHDP does not need off-line learning because of a quick linearized local model identification at the beginning, which is very fast and accurate.

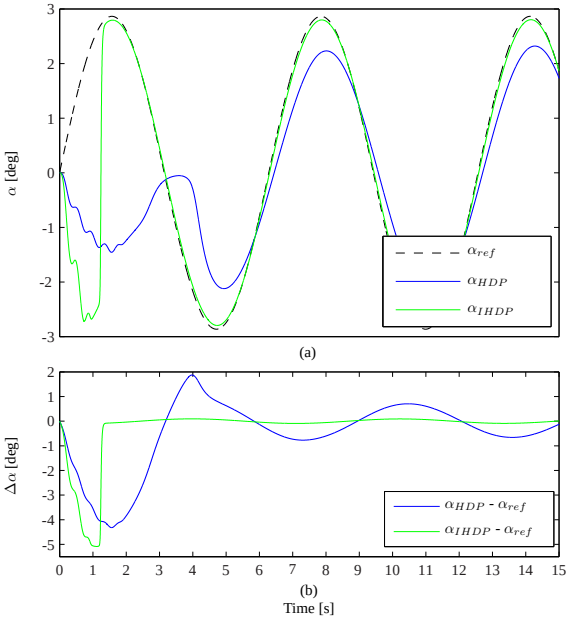


Figure 5: On-line tracking problem using neural networks and incremental approach.

5 CONCLUSION

This paper proposed a new approach, incremental model based heuristic dynamic programming, to design an adaptive flight control method without sufficient a-prior knowledge of the system dynamics. This method combines the advantages of HDP methods, which are adaptive and use a more general function approximator, and of incremental approaches, which do not need off-line learning and accelerate the on-line learning efficiently. The HDP method using a neural network to approximate the system dynamics and the IHDP method using incremental models are applied to a simple and illustrative application. By comparing the results, it is apparent that the presented IHDP method speeds up the on-line learning at the beginning and has a significantly higher precision than traditional HDP methods. To accelerate the on-line learning when a-prior knowledge is unknown or the system dynamics are changed suddenly is of great practical value.

As an extension of iADP method using a quadratic cost-to-go function, the IHDP method presented in this paper uses neural network functions with greater approximation ability and separates the policy evaluation and improvement with two approximators. This study generalized the use and applications of the iADP methods. Further investigation into different type of approximator and experimentation into more complex and realistic applications are strongly recommended.

REFERENCES

[1] Russell Enns and Jennie Si. Helicopter trimming and tracking control using direct neural dynamic programming. *Neural Networks, IEEE Transactions on*, 14(4):929–939, 2003.

[2] Silvia Ferrari and Robert F Stengel. Online adaptive critic flight control. *Journal of Guidance, Control, and Dynamics*, 27(5):777–786, 2004.

[3] Thomas Hanselmann, Lyle Noakes, and Anthony Zaknich. Continuous-time adaptive critics. *Neural Networks, IEEE Transactions on*, 18(3):631–647, 2007.

[4] Vivek Yadav, Radhakant Padhi, and SN Balakrishnan. Robust/optimal temperature profile control of a high-speed aerospace vehicle using neural networks. *Neural Networks, IEEE Transactions on*, 18(4):1115–1128, 2007.

[5] Fei-Yue Wang, Huaguang Zhang, and Derong Liu. Adaptive dynamic programming: an introduction. *Computational Intelligence Magazine, IEEE*, 4(2):39–47, 2009.

[6] Said G Khan, Guido Herrmann, Frank L Lewis, Tony Pipe, and Chris Melhuish. Reinforcement learning and optimal adaptive control: An overview and implementation examples. *Annual Reviews in Control*, 36(1):42–59, 2012.

[7] Jennie Si. *Handbook of learning and approximate dynamic programming*, volume 2. John Wiley & Sons, 2004.

[8] E. van Kampen, Q. P. Chu, and J. A. Mulder. Continuous adaptive critic flight control aided with approximated plant dynamics. In *Proc AIAA Guidance Navig Control Conf*, volume 5, pages 2989–3016, 2006.

[9] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall International, Inc., 1999.

[10] P. Simplício, M. D. Pavel, E. van Kampen, and Q. P. Chu. An acceleration measurements-based approach for helicopter nonlinear flight control using incremental nonlinear dynamic inversion. *Control Engineering Practice*, 21(8):1065–1077, 2013.

[11] S. Sieberling, Q. P. Chu, and J. A. Mulder. Robust flight control using incremental nonlinear dynamic inversion and angular acceleration prediction. *Journal of guidance, control, and dynamics*, 33(6):1732–1742, 2010.

[12] P. J. ACQUATELLA, E. van Kampen, and Qi Ping Chu. Incremental backstepping for robust nonlienar flight control. In *Proceedings of the EuroGNC 2013*, 2013.

[13] Y. Zhou, E. van Kampen, and Qi Ping Chu. Incremental approximate dynamic programming for nonlinear flight control design. In *Proceedings of the EuroGNC 2015*, 2015.

[14] Y. Zhou, E. van Kampen, and Qi Ping Chu. Nonlinear adaptive flight control using incremental approximate dynamic programming and output feedback. In *AIAA Guidance, Navigation and Control Conference*, 2016.

[15] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[16] Lars Sonneveldt. *Adaptive backstepping flight control for modern fighter aircraft*. TU Delft, Delft University of Technology, 2010.

[17] Seung-Hwan Kim, Yoon-Sik Kim, and Chanho Song. A robust adaptive nonlinear control approach to missile autopilot design. *Control engineering practice*, 12(2):149–154, 2004.

http://www.imavs.org/

Comparison between Experimental, Analytical, and Numerical Methods in Natural Frequencies of the Longitudinal Flight Modes for Small UAVs

M. El-Salamony*, S. Serokhvostov†, A.Epikhin‡, and K.Zaripov§
Moscow Institute of Physics and Technology
Department of Aeromechanics and Flight Engineering
140180, Gagarina street, 16, Zhukovsky, Russia

http://www.imavs.org/

ABSTRACT

In order to design the good controller for an unmanned aerial vehicle (UAV), an accurate mathematical model of aircraft must be constructed first. Geometry and mass of an aircraft are important factors in flight mechanics and in the calculations of stability and natural frequencies of its flight modes, which are of great importance in controller designing process. In this paper, comparison between the formulae of the large aircrafts applied on small UAVs scale, XFLR5 as a numerical commercial vortex lattice method (VLM) program and experimental data obtained from flight tests is made to investigate their accuracy.

NOMENCLATURE

c	aircraft chord
C_L	airplane lift coefficient in steady state condition
C_{L_α}	variation of airplane lift coefficient with angle of attack
C_{m_α}	variation of airplane pitching moment coefficient with angle of attack
C_{m_q}	variation of airplane pitching moment coefficient with pitch rate
g	gravitational acceleration
I_{xx}	moment of inertia around fuselage axis
I_{yy}	moment of inertia around wing axis
I_{zz}	moment of inertia around normal to X and Y axes
m	aircraft mass
M_α	Pitch angular acceleration per unit angle of attack
$M_{\dot{\alpha}}$	Pitch angular acceleration per unit rate of change of angle of attack
M_q	Pitch angular acceleration per unit pitch rate
M_{T_α}	Pitch angular acceleration per unit angle of attack due to thrust
M_u	Pitch angular acceleration per unit change in speed
$\bar{r_z^2}$	dimensionless MAV radius of inertia along the wing
S	reference area
$_{T_u}$	Pitch angular acceleration per unit change in speed due to thrust
U_1	cruise velocity
X_α	forward acceleration per unit angle of attack
X_u	forward acceleration per unit change in speed
Z_α	vertical acceleration per unit angle of attack
$Z_{\dot{\alpha}}$	vertical acceleration per unit rate of change of angle of attack
Z_q	vertical acceleration per unit pitch rate
Z_{T_u}	forward acceleration per unit change in speed due to thrust
Z_u	vertical acceleration per unit change in speed
θ	pitch angle
μ	dimensionless aircraft density
ρ	air density
ϕ	roll angle
ψ	yaw angle
$\omega_{n,lp}$	natural frequency long mode
$\omega_{n,sp}$	natural frequency short mode

*Email address: elsalamony.mostafa@phystech.edu
†Email address: serokhvostov@phystech.edu
‡Email address: ead_94@mail.ru
§Email address: kamil@zaripov.net

http://www.imavs.org/

1 INTRODUCTION

Since the UAVs (including mini and micro) are designed according to the procedure used for the large aircrafts, the stability calculations usually also follow the same formulae derived for the large aircrafts, but the geometrical scale is much smaller. So, the forces which affect the UAV change their order of power nonlinearly and some assumptions for the manned aircrafts cannot be valid in our case, and we are to introduce new assumptions which are not valid for the manned ones. The attempts of preliminary investigations were conducted previously [1, 2] and now it is being investigated in more detail.

It is well known from flight mechanics that if any disturbance occurred to the aircraft (as gust wind or control surface deflection) the stable aircraft try to damp this disturbance and return to its initial state in a way similar to mass-damper-spring system; oscillations are to be damped to zero. The aircraft has two longitudinal independent oscillations; short mode and long mode. Short mode oscillation is due to variation of angle of attack and the long mode oscillation is due to speed variation. So we can model these motions as two different simple mass-damper-spring systems. In case of damped vibrations, there are 2 definitions for the frequency; the damped frequency and the undamped / natural frequency. In this paper, the natural frequencies are considered.

In order to design a control system for an aircraft, the main step is to make a mathematical model of flight mechanics for the aircraft. The controller's accuracy depends on the accuracy of the mathematical model with respect to the physical model. Even a simple PID control could be used to make the required response if the mathematical model was accurate enough. The goal of this research is to understand how accurate the calculations based on the traditional formulae with respect to the experimental values are.

In this research, we investigate the formulae and some assumptions of calculating the natural frequencies of the longitudinal short and long modes mentioned by J. Roskam[3], D. Hull[4], and I. Ostoslavsky[5] and compare their results with the numerical VLM calculations from XFLR5 which is mainly designed for small UAVs[6] then these results are compared to real measurements of the natural frequencies obtained from UAV "Sonic 185" at flight.

2 PROCEDURE

For an aircraft, we can model its longitudinal motion as fourth order equation which - under some conditions - can be divided into two second order equations, short and long period modes. Roskam and Ostoslavsky considered this technique but with different procedure. Roskam's procedure[3] -and same for Hull[4] - starts from the aerodynamic coefficients, then calculating the forces on the aircraft then cast them together to create the coefficients of the main characteristic equation. By deep understanding of the physics behind the aircraft forces and their order of magnitude and vibrations, Hull and Roskam have made an approximate solutions for obtaining the natural frequency directly instead of solving the main fourth order equation by linking the forces directly to the natural frequency and damping ratio.

Ostoslavsky has derived the characteristic equation by a different method. Instead of calculating the forces, he used directly the non-dimensional aerodynamic coefficients and cast them together to get the coefficients of the characteristic equation. By deep understanding of the mathematics behind the characteristic equation, he decomposed the fourth order equation into two second order ones, each one describe a single flight mode under the fact that the natural frequency of the short mode is much bigger than the long mode.

In numerical methods as VLM as used in XFLR5 - the aircraft is divided into small panels. For each panel, a normal vector is set to be perpendicular to the camber. Also a combination of source, doublet, and vortex are added in one quarter of the panel, and a control point is added after three quarters of the panel to achieve the no-penetration condition[7]. By solving N equations obtained from the N panels, the total vortex strength is determined then the normal and tangential forces acting on the aircraft are obtained then converting them into non-dimensional coefficients. The next step is to import these values which are depend on the angle of attack - into the state space matrix and obtain the eigen values of the matrix which are a combination of natural frequency and damping ratio and they can be separated easily. XFLR5 is used because it is open source and used widely for UAV design process and also has the ability to obtain the natural frequencies and damping ratios.

Experimental data are obtained from real flight measurements using two different devices; the gyroscope of ArduPilot[8] with low sampling frequency and SmartAP[9] which has higher sampling frequency. Two methods are used in measuring the pitch angle; Euler angles from the gyroscope directly and quaternions to investigate the error between them. More details are in section 4.

2.1 The aircraft

The UAV used in this research is Sonic 185 of DYNAM[10]. Aircraft parameters and geometry are measured and listed in tables 1 and 2.

2.2 Airfoil

There is no data about the airfoils used in the wing and empennage of the airplane investigated. So, it was decided to find nearly similar profile from the known ones. Estimation of the airfoil is based on measurements of the thickness to chord ratio and the position of maximum camber and searching for a similar airfoil. Given that thickness to chord ratio is 10.7% at 39%, this leads to choose the airfoil E231 of Eppler series which shows good convergence as its thickness to chord ratio is 12.3% at 39.4%. For the empennage, NACA 0006 is used.

Property	value
$Mass\ [kg]$	1.183
$I_{xx}\ [kg.m^2]$	0.108
$I_{yy}\ [kg.m^2]$	0.065
$I_{zz}\ [kg.m^2]$	0.122
$Cruise\ velocity\ [m/s]$	8
$Aspect\ ratio$	10.295
$Span\ [m]$	1.85
$Wing\ area\ [m^2]$	0.33
$Center\ of mass\ from\ leading\ edge\ of\ root\ section\ [m]$	0.07

Table 1: Aircraft parameters.

Property	Wing	Horizontal Tail	Vertical Tail
$Aspectratio$	10.295	4.92	2.03
$Rootchord\ [m]$	0.205	0.125	0.2
$Tipchord\ [m]$	0.06	0.02	0.115
$Mean\ chord\ [m]$	0.189	0.1	0.16
$Span\ [m]$	1.85	0.48	0.16
$Area\ [m^2]$	0.33	0.046	0.03
$Sweep\ angle\ from\ Leading\ edge\ [degree]$	6.71	18.17	23.25

Table 2: Sonic 185 geometry.

3 CALCULATIONS

3.1 Aerodynamic coefficients

The aerodynamic and stability coefficients are calculated according to Roskam procedure[11] and the results are shown in table 3. It is noticed that C_{mq} in Roskam formulae is twice the one in Ostoslavsky formulae by definition.

Derivative	Lift coeff.	Drag coeff.	Moment coeff.
α	5.253	0.377	-0.47
q	7.312	0	Roskam:-8.59 Ostoslavsky: -4.295
$\dot{\alpha}$	4.655	0	-4.1337

Table 3: Aerodynamic derivatives

3.2 Roskam and Hull procedure

In this section we will consider the procedure of Roskam (exact and approximate) and Hull together because they have quite similar formulae and results as a sequence.
The characteristic equation of Roskam[3] is

$$Ax^4 + Bx^3 + Cx^2 + Dx + E = 0 \tag{1}$$

where;

$$A = U_1 - Z_{\dot{\alpha}} \tag{2}$$

$$B = (-(U_1 - Z_{\dot{\alpha}})(X_u + X_{Tu} + M_q) - Z_\alpha M_{\dot{\alpha}}(U_1 + Z_q) \tag{3}$$

$$C = (X_u + X_{Tu})[M_q * (U_1 - Z_{\dot{\alpha}}) + Z_\alpha + M_{\dot{\alpha}}(U_1 + Z_q)] + M_q Z_\alpha - Z_u X_\alpha + M_{\dot{\alpha}} g \sin\theta_1 - (M_\alpha + M_{T\alpha})(U_1 + Z_q) \tag{4}$$

$$D = g\sin\theta_1[M_\alpha + M_{T\alpha} - M_{\dot{\alpha}}(X_u + X_{Tu})] + g\cos\theta_1[Z_u M_{\dot{\alpha}} + (Mu + M_{Tu})(U_1 - Z_{\dot{\alpha}})] + (M_u + M_{Tu})[-X_{\dot{\alpha}}(U_1 + Zq)] + Z_u X_\alpha M_q + (X_u + X_{Tu})[(M_\alpha + M_{T\alpha})(U_1 + Z_q) - M_q * Z_\alpha] \tag{5}$$

$$E = g\cos\theta_1[(M_\alpha + M_{T\alpha})Z_u - Z_\alpha(M_u + M_{Tu})] + g\sin\theta_1[(M_u + M_{Tu})X_\alpha - (X_u + X_{Tu})(M_\alpha + M_{T\alpha})] \tag{6}$$

After analysis, the results showed that ω_n=1.0066 Hz for the short mode, and 0.178 Hz for the long mode. It is important to mention that the damping ratio of the short mode is 0.96.
The approximate solution for the natural frequency of the short and long modes according to Roskam are:

$$\omega_{n,sp} = \sqrt{\frac{Z_\alpha * M_q}{U_1} - M_\alpha} \tag{7}$$

$$\omega_{n,lp} = \sqrt{\frac{-Z_u * g}{U_1}} \tag{8}$$

The results are: ω_n=1.0501 Hz for the short mode, and 0.2835 Hz for the long mode.
According to Hull method[4], the approximate formulae for short mode and long mode are

$$\omega_{n,sp} = \sqrt{\frac{Z_\alpha * M_q - M_\alpha * (U_1 + Z_q)}{U_1 + Z_{\dot{\alpha}}}} \tag{9}$$

$$\omega_{n,lp} = \sqrt{\frac{-Z_u * g}{U_1 + Z_q}} \tag{10}$$

The results are quite similar to the approximate Roskam formulae as shown: for short mode,ω_n=0.9888 Hz and for long mode ω_n=0.302 Hz.

3.3 Ostoslavsky procedure

Ostoslavsky method [5] is good method for direct determination of what parameters affect the frequencies. He introduced the characteristic equation of the system as fourth order function in its eigen values as:

$$F = \lambda^4 + a_1\lambda^3 + a_2\lambda^2 + a_3\lambda + a_4 = 0 \tag{11}$$

http://www.imavs.org/

http://www.imavs.org/

where the coefficients of this equations under the steady state condition are:

$$a_1 = C_{L\alpha} - \frac{C_{m\dot{\alpha}} + C_{mq}}{\bar{r_z^2}} \tag{12}$$

$$a_2 = \frac{C_{m\alpha} * \mu + C_{L\alpha} * C_{mq}}{\bar{r_z^2}} \tag{13}$$

$$a_3 = \frac{-2C_L[(C_L - C_{D\alpha}) * C_{mq} + C_L * C_{m\dot{\alpha}}]}{\bar{r_z^2}} \tag{14}$$

$$a_4 = \frac{-2C_L^2 * \mu * C_{m\alpha}}{\bar{r_z^2}} \tag{15}$$

and

$$\bar{r_z^2} = \frac{I_{yy}}{m * c^2} \tag{16}$$

$$\mu = \frac{2m}{\rho * S * c} \tag{17}$$

By solving equation (11), we obtain the eigen values of the system which indicate their natural frequencies and damping ratios.
In order to simplify the decomposition of the characteristic equation, Ostoslavsky supposed that the eigen values of the long mode is negligible with respect to the short mode, and finally the natural frequencies of the short and long modes can respectively obtained from:

$$\lambda^2 + a_1\lambda + a_2 = 0 \tag{18}$$

$$\lambda^2 + \frac{a_2a_3 - a_1a_4}{a_2^2}\lambda + \frac{a_4}{a_2} = 0 \tag{19}$$

The results for the exact equation are ω_n = 0.1787 Hz for the long mode but for the short mode the eigen values were real which means that the short mode vibration is over damped. The results of the simplified equations are ω_n=0.1758 Hz for the long mode and the short mode also is over damped.

Another approach is used to identify the natural frequency in this case. It is well known from the definition of the eigen value that:

$$\lambda = -\omega_n * \zeta \pm \sqrt{(\omega_n * \zeta)^2 - \omega_n^2} \tag{20}$$

so the natural frequency can be calculated from the analysis of the left and right parts individually and comparing them with the formula of the roots of the second order equation. Using this 'equation-similarity' approach, the natural frequency is calcuated and the result is 1.0605 Hz. This is close to the result obtained by Roskam procedure.

3.4 XFLR5 Calculations

The UAV is drawn on XFLR5 using the measurements from tables 1 and 2 as shown in figure 1 . It should be noted that there is no fuselage in this model.
Calculation done shows that ω_n=1.129 Hz for the short mode, and 0.178 Hz for the long mode.

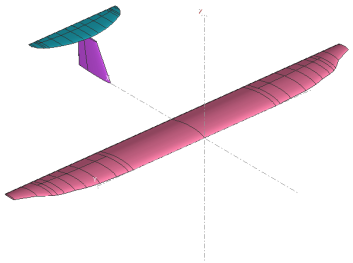


Figure 1: Sonic 185 drawing in XFLR5

3.5 Experiment and Experimental Data

The experiment was conducted for the steady cruise flight of the aircraft with two measuring devices. To prevent the additional disturbances, the plane was controlled in manual mode without autopilot stabilization. During the flight the aircraft was balanced so as it moves straight at the constant altitude and with the constant velocity. Flight parameters were controlled form the ground station of Ardupilot by means of telemetry link.
Two sets of sensors were used independently for more accuracy and doubling the amount of data and also to notice the difference between results for different sampling frequency and different ways to measure flight angles. First set is the data obtained from the gyroscope of the autopilot of type ArduPilot Mega which measure pitch, roll, and yaw angles of the aircraft and has sampling frequency of 3.7 Hz. The second set of data is from SmartAP that can measure with sampling frequency up to 250 Hz and use quaternions which provide a convenient mathematical notation for representing rotations and orientations of objects in three dimensions. It has some benefits compared to Euler angles; they are simpler to compose and avoid the problem of gimbal lock. Compared to rotation matrices they are more numerically stable and efficient[12].
In order to convert quaternions to pitch, roll, and yaw, transformation of equation (21) [13] is used. Quaternions and their equivalent Euler angles are shown in figures 2 and 3 respectively.

$$\begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} = \begin{pmatrix} \tan^{-1} \frac{2(q_0q_1+q_2q_3)}{1-2(q_1^2+q_2^2)} \\ \sin^{-1} 2(q_0q_2 - q_1q_3) \\ \tan^{-1} \frac{2(q_0q_3+q_1q_2)}{1-2(q_1^2+q_2^2)} \end{pmatrix} \tag{21}$$

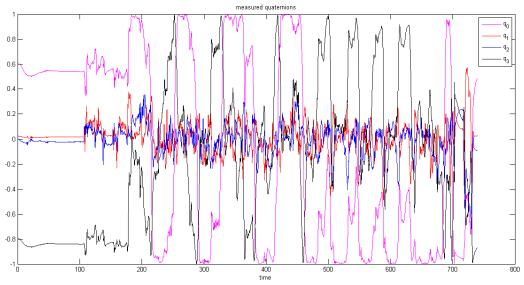


Figure 2: Quaternions estimated during flight.

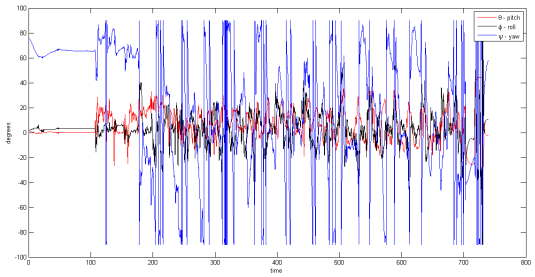


Figure 3: Euler angles calculated from quaternions.

Both data sets retrieved are processed by Fast Fourier Transform[14] once without filter and another time with filter. While examining the signal without filter, it is taken into account the signal first and last points have the same values to prevent aliasing as illustrated in figure 4.

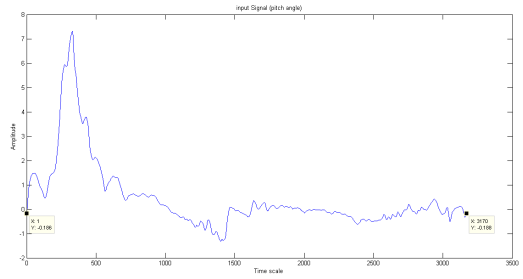


Figure 4: Sample for a signal which first and last points have the same values to prevent aliasing.

Data are filtered by Hanning filter[15, 16] to prevent leakage in the transform[17]. Such filter is chosen for this case because:

1. we are investigating a random signal with unknown frequency.
2. the vibrations are within narrow band.
3. the exact amplitude is not as important as the value of the frequency itself.
4. overcome the noise and get the mean value of the frequency.

For these four reasons, the most suitable filter is Hanning filter[17]. Data signal of pitch angle are shown in figure 5.

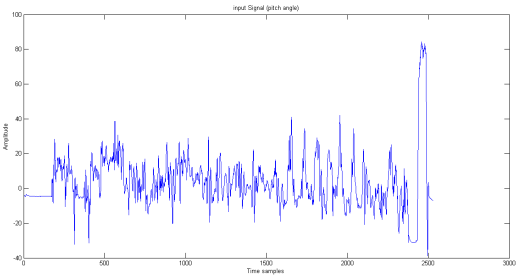


Figure 5: Pitch angle during flight.

Fast Fourier Transform -or simply FFT- is used to convert discrete time samples from time to frequency domain. After processing and filtering, some frequencies appeared close to each other with comparable amplitudes. After some averaging, the results show a freq of 0.18±0.01 Hz. Vibrations due to motor are not captured because its frequency is rather high but it act at noise all over the signal. Figures 6 to 8 show samples of the obtained results at different periods of time.

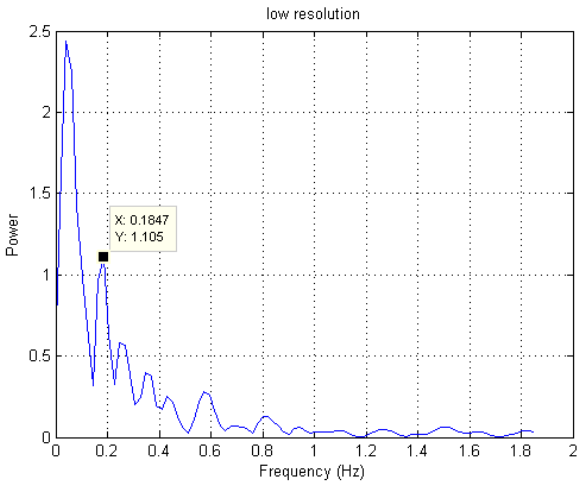


Figure 6: Sample 1 of the processed signal .

http://www.imavs.org/

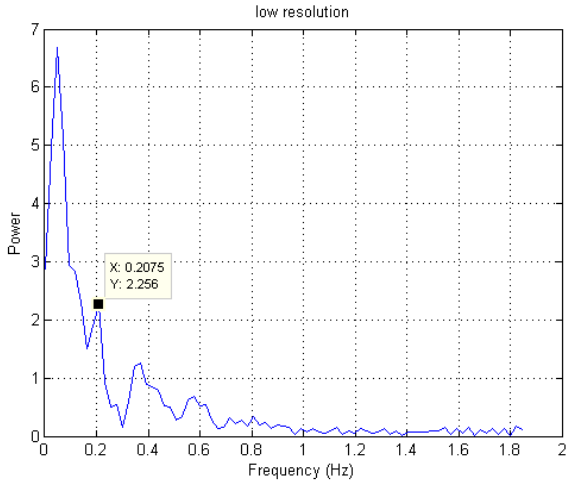


Figure 7: Sample 2 of the processed signal .

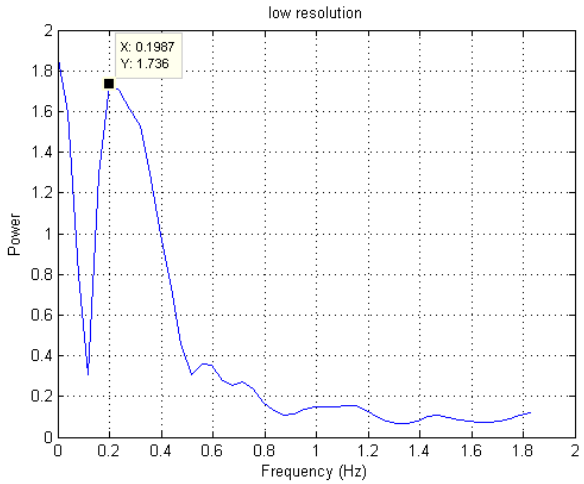


Figure 8: Sample 3 of the processed signal .

4 ANALYSIS AND DISCUSSION

Concerning the mismatch of C_{mq} definition between Roskam and Ostoslavsky, although this conflict may be a problem but the C_{mq} used in the formula of Roskam for the natural frequency is divided by factor of 2, so the result is the same as Ostoslavsky. This shows that every procedure must follow its own definition for the aerodynamic and stability coefficients.

It is noticed that approximate methods of Roskam and Hull overestimate the long mode frequency. This is because they neglected some terms under the assumption that μ -which indicates the mass of the aircraft - is rather big compared to $C_{L\alpha} * C_{mq}$ but this condition is not applicable in the UAVs.

Considering the devices used in the experimental measurement, the low-frequency device could not measure frequencies more than 2 Hz because it need more sampling frequency. The signal of the pitch angle from the gyroscope and the transformed one are quite similar as shown in figure (9) except in the peak points since the high frequency device can capture it better.

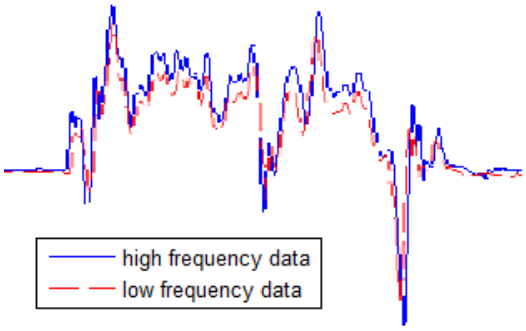


Figure 9: Sample from both signals together.

By comparing the calculations with the experimental results (table 4), it is obvious that the long mode frequencies obtained from Roskam exact formula, Ostoslavsky formulae and XFLR5 are within the allowable region of the experimental result. For the short mode frequency, Ostoslavsky method shows that this motion has no oscillations, Roskam methods also shows that the damping ratio is 0.96, so even if it exists, it will be rather difficult to be observed since the damping ratio decrease the amplitude in the frequency domain. In order to clarify this point, this problem will be considered again with additional calculations and measurements.

Method	Short	Long
Roskam - exact	1.0066	0.178
Roskam - approx	1.0501	0.2835
Hull (approx)	0.9888	0.302
Ostoslavsky - exact	–	0.1787
Ostoslavsky - approx	1.0605[1]	0.1758
XFLR5	1.12	0.178
experiment	–	0.18 ± 0.01

Table 4: Results of natural frequencies of all methods.

. Regarding the assumption of that the long mode frequency is much lower than the short mode frequency, these two frequencies obtained from the results are compared

[1] using the 'equation-similarity' approach in subsection 3.3

http://www.imavs.org/

to each other. In Roskam case, the long mode frequency is 17.7% of the short mode. In XFLR5, long mode frequency is 15.9% of the short mode so we can conclude that the assumption of short mode is much bigger than the long mode is valid for small UAVs.

5 CONCLUSION

This research compares the natural frequencies of longitudinal motion calculated by the usual methods of estimation for the manned aircrafts found in the references of Roskam, Hull, and Ostoslavsky, and the numerical VLM program XFLR5 with experimental values of real flight measured by two different devices with different sampling frequencies and methods of measuring the angles. It is found that there is no dependence of the experimental data on the method of measuring the flight angles or on sampling frequency. The assumption of long mode can be neglected with respect to the short mode is valid. For the long mode, exact Roskam, Ostoslavsky, and XFLR5 estimate the frequency within range of the experimental results while the methods of Roskam approximation and Hull are not correct in case of small UAVs. As for the short mode, we encountered the situation that our data are near the critical point and a set of additional analysis and experiments must be provided. It is noted that results from analytical methods are valid only for the aerodynamic coefficients defined in the same procedure.

ACKNOWLEDGEMENT

The first author acknowledge the scholarship for masters degree from Moscow Institute of Physics and Technology (MIPT) for international students.

REFERENCES

[1] S.V. Serokhvostov, A.A.Dementyev, and T.E. Churkina. Analytical investigation of mav dynamical characteristics and its influence on design. In *proc of International Micro Air Vehicle Conference (IMAV), Braunschweig, Germany*, 2010.

[2] S.V. Serokhvostov, N. Pushchin, and K. Shilov. Mav unsteady characteristics in-flight measurement with the help of smartap autopilot. In *proc of International Micro Air Vehicle Conference (IMAV), Toulouse, France*, 2013.

[3] Jan Roskam. *Airplane flight dynamics and automatic flight controls*. DARcorporation, 1995.

[4] David G Hull. *Fundamentals of airplane flight mechanics*. Springer, 2007.

[5] I.V. Ostoslavsky. Aerodynamics of aircraft. *Oborongiz, Moscow*, 1957.

[6] André Deperrois. XFLR5 analysis of foils and wings operating at low Reynolds numbers. *Guidelines for XFLR5*, 2009.

[7] Edward Lewis Houghton and Peter William Carpenter. *Aerodynamics for engineering students*. Butterworth-Heinemann, 2003.

[8] ardupilot dev Team. Archived:apm 2.5 and 2.6 overview. `http://ardupilot.org/copter/docs/common-apm25-and-26-overview.html`, 2015. last accessed april,2016.

[9] Smart AutoPilot. smartap autopilot manual. `http://sky-drones.com/content/10-manuals`, 2015. last accessed april,2016.

[10] Universal Power Enterprizes Ltd. *Sonic 185: Instruction manual.* Universal Power Enterprizes Ltd., Kowloon, Hong Kong, 2008.

[11] Jan Roskam. *Methods for estimating stability and control derivatives of conventional subsonic airplanes*. DARcorporation, 1971.

[12] William Rowan Hamilton. Ii. on quaternions; or on a new system of imaginaries in algebra. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 25(163):10–13, 1844.

[13] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006.

[14] Richard S Figliola and Donald Beasley. *Theory and design for mechanical measurements*. John Wiley & Sons, 2015.

[15] Tim Wescott. Sampling: What nyquist didnt say, and what to do about it. *Wescott Design Services, Oregon City, OR*, 2010.

[16] L Dactron. Understanding fft windows. *Application Note. lds group*, 2003.

[17] Pierre Wickramarachi. Effects of windowing on the spectral content of a signal. *Sound and Vibration*, 37(1):10–13, 2003.

http://www.imavs.org/

Mixed differential game in Target-Attacker-Defender problem

Li Liang[1,2]*, Zhihong Peng[1,2], Benmei Chen[3], Xinxing Li[1,2]
[1]School of Automation, Beijing Institute of Technology, Beijing, 100081, China
[2]State Key Laboratory of Intelligent Control and Decision of Complex System, Beijing, 100081, China
[3]National University of Singapore, Singapore

http://www.imavs.org/

ABSTRACT

This paper addresses a mixed differential game with three players: Target, Attacker and Defender, where the Attacker aims to capture the Target whilst avoiding being captured by the Defender. At this point, there are two pursuit-evasion problems in this game, and two focuses should be considered: one is the cooperation between the Target and the Defender; the other one is the role changes between pursuer and evader of the Attacker. This paper discusses the mixed differential game analytically, obtains the optimal strategies of the three players, and provides numerical solutions for different initial states.

1 INTRODUCTION

Multi-agent pursuit-evasion games has become a research hotpot in the fields of aerospace, control, and robotics. In this paper, we consider a particular differential game, called Target-Attacker-Defender game, where the Attacker chases the Target while avoids being captured by the Defender. Oppositely, the Target should cooperate with the Defender in order to escape from the Attacker whilst acting a bait to help the Defender capture the Attacker. Thus, there are two pursuit-evasion problems in this game: Attacker-Target and Defender-Attacker, so we call it as a mixed differential game.

The literature contains a large number of work on pursuit-evasion games, including the problem formulations and solution techniques of single pursuer-single evader problem [1–3] , and the determination of the role assignments of pursuer and evader in [4, 5].

Pursuit-evasion games with two players cooperating to antagonize to the other player are also covered in the literature [6–8]. For example, a so-called "fishing game" with point capture is proposed in [6]. The authors construct a barrier by using the method of explicit policy in the game of kind and solve the players' optimal strategies in terms of the game of degree. Cooperation between two agents with the goal of avoiding captured by a single pursuer is addressed in [7] where the agents' strategies should trade off evasion and herding. In [8], inspired by hunting and foraging behaviors of various fish species, the authors present a novel multi-phase cooperative strategy in which the pursuers move in specific formations and confine the evader to a bounded region.

Recently, the Target-Attacker-Defender problem has been concerned by Gutman [9, 10], Shima [11, 12], David and Pachter [13–16]. The problem without cooperation between the Target and the Defender is addressed in [9, 10] where linearization-based Attacker maneuvers are presented to evade the Defender and continue pursuing the Target. Different types of the cooperation between the Target and the Defender are considered in [11–18], where the Target represents an aircraft trying to evade an Attacker and the Defender is a pursuer in order to intercept the Attacker. These articles only consider on target set that the interception point between the Defender and the Attacker is closer to the Target's position. In other words, the Attacker only aims to capture the Target but not regard the threat from the Defender. And the game terminates only when the Target is captured by the Attacker.

In this paper, we consider two focuses on the Target-Attacker-Defender problem: one is the cooperation between the Target and the Defender; the other one is the role changes between pursuer and evader of the Attacker. Specifically, as a pursuer, the Attacker attempts to capture the Target; while as a evader, the Attacker should avoid being captured by the Defender. We obtain the analytical solutions of this mixed differential game, and derive out the optimal strategies of the three players. In addition, we provide the numerical solution of this game for different initial states.

This paper is organized as follows. Section II describes the engagement scenario. In section III optimal control strategies are achieved. In section IV, numerical solutions of the differential game are provided and the examples are given. Finally concluding remarks are made in section V.

2 THE PROBLEM STATEMENT

The Target-Attacker-Defender differential game is shown in Figure 1. The speeds of the Target, Attacker, and Defender are denoted by V_T, V_A, V_D, respectively, which are assumed to be constant. The dynamics of the Target-Attacker-Defender in the realistic game space are given as follows:

$$\dot{x}_T = V_T \cos\hat{\phi}, \dot{y}_T = V_T \sin\hat{\phi} \tag{1}$$

$$\dot{x}_A = V_A \cos\hat{\chi}, \dot{y}_A = V_A \sin\hat{\chi} \tag{2}$$

*Email address(es): liangli@bit.edu.cn

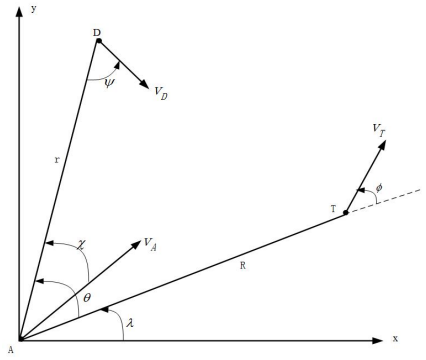


Figure 1: Reduced state space

$$\dot{x}_D = V_D \cos\hat{\psi}, \dot{y}_D = V_D \sin\hat{\psi} \tag{3}$$

where the $\hat{\phi} = \phi + \lambda, \hat{\chi} = \lambda + \theta - \chi$, and $\hat{\psi} = \psi + \theta + \lambda - \pi$.

The speed ratio parameters are defined as $\alpha = \frac{V_T}{V_A}$ and $\beta = \frac{V_D}{V_A}$. In this paper, we consider that the speed of the Attacker is faster than the Target and the Defender, so that $\alpha, \beta < 1$. The parameter R_{AT} represents the distance between the Target and the Attacker. The parameter r_{DA} represents the distance between the Attacker and Defender. The positive constant R_c represents the capture radius of the Attacker and the positive constant r_c represents the capture radius of the Defender. This game contains two pursuit-evasion games: the Attacker pursues the Target and the Defender pursues the Attacker. As a result, the Attacker plays two roles, purser for the Target and evader for the Defender. Naturally, in order to win the game as far as possible, the Target and the Defender should cooperate as a team.

3 THE DIFFERENTIAL GAME

In this section, we describe the corresponding dynamics of the three-player engagement using a reduced state space formed by the ranges R_{AT} and r_{DA}, and the angle between them , denoted by θ. The objective of the Attacker is to choose an optimal heading angle, denoted by χ, such that the distance R_{AT} is minimum and the distance r_{DA} is maximum. On the contrary, the objectives of the Target and the Defender are to adopt the optimal heading angles, denoted by ϕ and ψ respectively, such that R_{AT} is maximum and r_{DA} is minimum. When the distance $R_{AT}(t_1) = R_c$ or the distance $r_{DA}(t_2) = r_c$, the differential game terminates. The t_f, denotes the game terminal time, is equal to $\min\{t_1, t_2\}$.

We transform the relative heading angles to the heading angles in the reduced state space with respect to the fixed coordinate axis x using the line of sight angle from the Attacker to the Target, denoted by λ. For the convenience of notation, we denote R_{AT} as R and denote r_{DA} as r. The dynamics in the reduced state space are as follows:

$$\dot{R} = \alpha\cos\phi - \cos(\theta - \chi) \tag{4}$$

$$\dot{r} = -\cos\chi - \beta\cos\psi \tag{5}$$

$$\dot{\theta} = -\frac{\alpha}{R}\sin\phi + \frac{1}{R}sin(\theta - \chi) - \frac{\beta}{r}sin\psi + \frac{1}{r}sin\chi \tag{6}$$

$$\begin{aligned} R(t_0) &= R_0 \\ r(t_0) &= r_0 \\ \theta(t_0) &= \theta_0 \end{aligned} \tag{7}$$

The objective of the Attacker is to minimize the distance R maximize the distance r at the terminal time t_f, where t_f is free. The objective of the Target and the Defender is to maximize the R and minimize the r. Thus, the payoff function can be indicated as

$$\max_{\phi,\psi}\min_{\chi} J = \max_{\phi,\psi}\min_{\chi}\int_{t_0}^{t_f} w_1\dot{R} - w_2\dot{r}dt \tag{8}$$

where the parameters w_1 and w_2 are the weighting coefficients.

Then from the differential game theory [1] [2], the Hamiltonian function is given by

$$\begin{aligned} H(\lambda,\phi,\psi,\chi,t) &= w_1\dot{R} - w_2\dot{r} + \lambda_R\dot{R} + \lambda_\theta\dot{\theta} \\ &= w_1(\alpha\cos\phi - \cos(\theta-\chi)) - w_2(-\cos\chi - \beta\cos\psi) \\ &+\lambda_R(\alpha\cos\phi - \cos(\theta-\chi)) + \lambda_r(-\cos\chi - \beta\cos\psi) \\ &+\lambda_\theta(-\frac{\alpha}{R}\sin\phi + \frac{1}{R}\sin(\theta-\chi) - \frac{\beta}{r}\sin\psi + \frac{1}{r}\sin\chi) \end{aligned} \tag{9}$$

The co-state dynamics are given by

$$\dot{\lambda}_R = \frac{\lambda_\theta}{R^2}(sin(\theta-\chi) - \alpha sin\phi) \tag{10}$$

$$\dot{\lambda}_r = \frac{\lambda_\theta}{r^2}(sin\chi - \beta sin\psi) \tag{11}$$

$$\dot{\lambda}_\theta = -w_1\sin(\theta-\chi) - \lambda_R\sin(\theta-\chi) - \frac{\lambda_\theta}{R}\cos(\theta-\chi) \tag{12}$$

The game ends in two cases. In one case, we define the team of the Defender and the Target wins the game and the Attacker is captured. The terminal state $r(t_f)$ is fixed and equal to r_c. The terminal states $R(t_f)$ and $\theta(t_f)$ are free, we define $\lambda_R(t_f) = 0, \lambda_\theta(t_f) = 0$. The best solution for this problem requires that the terminal condition $H(x^*(t_f), u^*(t_f), \lambda^*(t_f), t_f) = 0$. In summary, the terminal conditions are:

$$\begin{aligned} r(t_f) &= r_c \\ \lambda_R(t_f) &= 0 \\ \lambda_\theta(t_f) &= 0 \\ w_1\alpha + \beta(w_2 - \lambda_r(t_f)) & \\ -\sqrt{w_1^2 + (w_2 - \lambda_r(t_f))^2 - 2w_1(w_2 - \lambda_r(t_f))cos\theta(t_f)} &= 0 \end{aligned} \tag{13}$$

http://www.imavs.org/

In the other case, the Attacker wins the game, captures the Target but not be captured by the Defender, the terminal conditions are:

$$R(t_f) = R_c$$
$$\lambda_r(t_f) = 0$$
$$\lambda_\theta(t_f) = 0$$
$$(w_1 + \lambda_R(t_f))\alpha + w_2\beta$$
$$-\sqrt{(w_1 + \lambda_R(t_f))^2 + {w_2}^2 - 2w_2(w_1 + \lambda_R(t_f))cos\theta(t_f)} = 0 \quad (14)$$

Theorem 1: In the mixed differential game, the Target and Defender optimal control angles are given by

$$\sin\phi^* = -\frac{\lambda_\theta/R}{\sqrt{(\lambda_\theta/R)^2 + (w_1 + \lambda_R)^2}}$$
$$\cos\phi^* = \frac{w_1 + \lambda_R}{\sqrt{(\lambda_\theta/R)^2 + (w_1 + \lambda_R)^2}} \quad (15)$$
$$\sin\psi^* = -\frac{\lambda_\theta/r}{\sqrt{(\lambda_\theta/r)^2 + (w_2 - \lambda_r)^2}}$$
$$\cos\psi^* = \frac{w_2 - \lambda_r}{\sqrt{(\lambda_\theta/r)^2 + (w_2 - \lambda_r)^2}} \quad (16)$$

The optimal control angle of the Attacker is given by Eqs(17)(shown on the top of the next page).

Proof: The analytic solution of differential game can be obtained by:

(1) Differentiating the Hamiltonian function in ϕ and setting the derivative to zero:

$$\frac{\partial H}{\partial\phi} = \alpha[-\frac{\lambda_\theta}{R}cos\phi - (w_1 + \lambda_R)sin\phi] = 0 \quad (18)$$

Using the trigonometric identity

$$\sin^2\phi + \cos^2\phi = 1$$

we can conclude Eqs(15).

The second partial derivative of the Hamiltonian function in ϕ :

$$\frac{\partial^2 H}{\partial\phi^2} = \alpha[\frac{\lambda_\theta}{R}\sin\phi - (w_1 + \lambda_R)cos\phi] =$$
$$-\alpha[\frac{\lambda_\theta^2 + R^2(w_1 + \lambda_R)^2}{R^2\sqrt{(\lambda_\theta/R)^2 + (w_1 + \lambda_R)^2}}] < 0 \quad (19)$$

which means that ϕ^* maximizes the payoff J; That is to say, it maximizes the final distance $R(t_f)$ and minimizes the distance $r(t_f)$.

(2) With regard to ψ ,we can obtain the optimal heading of the Defender in a similar way.

$$\frac{\partial H}{\partial\psi} = \beta[-\frac{\lambda_\theta}{r}cos\psi - (w_2 - \lambda_r)sin\psi] = 0 \quad (20)$$

and conclude Eqs(16).

The second partial derivative of the Hamiltonian function in ψ :

$$\frac{\partial^2 H}{\partial\psi^2} = \beta[\frac{\lambda_\theta}{r}\sin\psi - (w_2 - \lambda_r)cos\psi] =$$
$$-\beta[\frac{\lambda_\theta^2 + r^2(w_2 - \lambda_r)^2}{r^2\sqrt{(\lambda_\theta/r)^2 + (w_2 - \lambda_r)^2}}] < 0 \quad (21)$$

which means that ψ^* maximizes the payoff J; that is to say, it maximizes the final distance $R(t_f)$ and minimizes the distance $r(t_f)$.

(3) The optimal heading χ^* of the Attacker can also be solved by differentiating the Hamiltonian function in χ and setting the derivative to zero:

$$\frac{\partial H}{\partial\chi} = \left((w_1 + \lambda_R)cos\theta - (w_2 - \lambda_r) - \frac{\lambda_\theta}{R}\sin\theta\right)\sin\chi$$
$$-\left((w_1 + \lambda_R)sin\theta + \frac{\lambda_\theta}{R}\cos\theta - \frac{\lambda_\theta}{r}\right)\cos\chi = 0 \quad (22)$$

Letting

$$a = (w_1 + \lambda_R)sin\theta + \frac{\lambda_\theta}{R}\cos\theta - \frac{\lambda_\theta}{r}$$
$$b = (w_1 + \lambda_R)cos\theta - (w_2 - \lambda_r) - \frac{\lambda_\theta}{R}\sin\theta$$

Then, we can obtain

$$\sin\chi^* = \frac{a}{\sqrt{a^2 + b^2}}$$
$$\cos\chi^* = \frac{b}{\sqrt{a^2 + b^2}} \quad (23)$$

Furthermore, we have

$$\frac{\partial^2 H}{\partial\chi^2} = b\cos\chi + a\sin\chi = \frac{b^2}{\sqrt{a^2 + b^2}} + \frac{a^2}{\sqrt{a^2 + b^2}} > 0 \quad (24)$$

It shows that the optimal heading of the Attacker minimizes the payoff J, which is equivalent to minimizing the distance $R(t_f)$ and maximizing the distance $r(t_f)$.

http://www.imavs.org/

$$\sin\chi^* = \frac{(w_1+\lambda_R)sin\theta + \frac{\lambda_\theta}{R}\cos\theta - \frac{\lambda_\theta}{r}}{\sqrt{\left((w_1+\lambda_R)sin\theta + \frac{\lambda_\theta}{R}\cos\theta - \frac{\lambda_\theta}{r}\right)^2 + \left((w_1+\lambda_R)cos\theta - (w_2-\lambda_r) - \frac{\lambda_\theta}{R}\sin\theta\right)^2}}$$

$$\cos\chi^* = \frac{(w_1+\lambda_R)cos\theta - (w_2-\lambda_r) - \frac{\lambda_\theta}{R}\sin\theta}{\sqrt{\left((w_1+\lambda_R)sin\theta + \frac{\lambda_\theta}{R}\cos\theta - \frac{\lambda_\theta}{r}\right)^2 + \left((w_1+\lambda_R)cos\theta - (w_2-\lambda_r) - \frac{\lambda_\theta}{R}\sin\theta\right)^2}} \quad (17)$$

4 NUMERICAL SOLUTIONS OF THE OPTIMAL CONTROL PROBLEM

Eqs(4)-(14) represent a two point boundary value problem (TPBVP). In many cases, it is difficult to solve analytically. At present, there are some methods to solve this problem numerically [19], such as shooting method, difference method and finite element method and so on. Regardless of the methods, it is necessary to estimate the initial values of the TPBVP. If the initial values are uncertain, obtaining an ideal solution will be very difficult. That is the main challenge in solving TPBVP [20]. Since the co-state variable has no obvious physical meaning, in this paper, we indirectly estimate the initial value of the co-state variables by setting the control variables. Considering the cooperation of the Target and the Defender, we assume that at the initial time the Target moves toward the Defender and the Defender moves toward the Target. The Attacker, aimed to capture the Target, moves toward the Target initially.

Example1: Let the initial conditions be given by $R_0 = 5, r_0 = 5$, and $\theta_0 = 1rad$ .The initial line-of-sight angle is given by $\lambda_0 = 1rad$. The Attacker initially located at the coordinates $(0,0)$. The capture radiuses are given by $R_c = 1$, and $r_c = 1$. The speed ratios are $\alpha = 0.6$, and$\beta = 0.8$. We obtain the initial value of the co-state variables using the initial value of the control angles: $\lambda_R(0) = 0.2235, \lambda_r(0) = -0.2235$, and $\lambda_\theta(0) = -6.6215$ .The optimal control strategies of three players are shown in Figure 2 and the optimal states trajectories of three players are shown in the Figure 3. The optimal motion trajectories of three players are shown in Figure 4. The 'o' and '*' respectively represent the initial point and the terminal point of three players. We noted that the Defender captured the Attacker at 4.6s. That is to say, the Defender and the Target win the game.

Example2: We give the initial conditions $r_0 = 3, \theta_0 = 2.9rad, \lambda_0 = 0.5rad$, and the other parameters are same as the parameters in example 1. Then we have $\lambda_R(0) = 34.7858, \lambda_r(0) = -37.1870$, and $\lambda_\theta(0) = -17.2051$. The optimal control strategies and the optimal states trajectories are shown in Figure 5 and Figure 6, respectively. The optimal motion trajectories of three players are shown in Figure 7. We noted that the Attacker captured the Target at 10.5s. That is to say, the Attacker win the game.

Example3: Based on the initial conditions of example1,we change the initial conditions $r_0 = 3, \theta_0 = 1.5rad$, and $\lambda_0 = 0.5rad$. The results of game are shown in Figure 8, Figure 9 and Figure 10. We noted that the game continues at all times and no one can win the game.

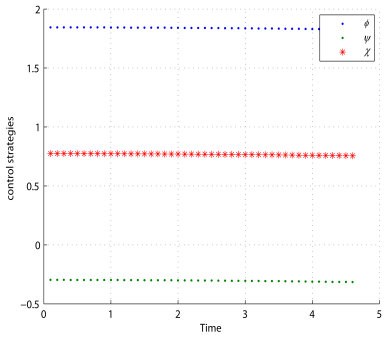


Figure 2: The optimal heading angles of three players in Example1

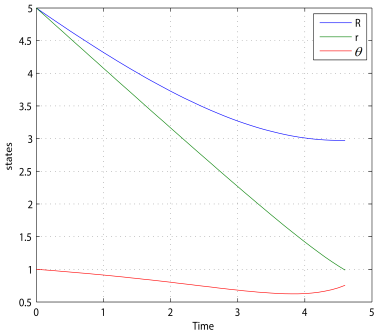


Figure 3: The optimal state trajectories of three players in Example1

5 CONCLUSIONS

In this paper, we formulate an Attacker-Target-Defender mixed differential game and solve the optimal solutions. In this game, two pursuit-evasion games are described and two target sets are determined. The Attacker plays two roles, and the Target and the Defender cooperate with each other. We provide the optimal strategy for each player. By estimating

http://www.imavs.org/

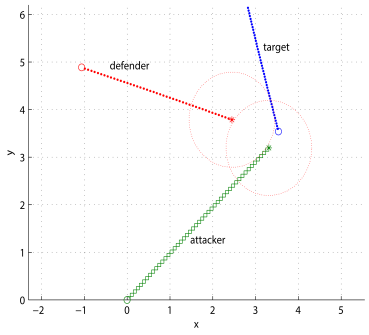


Figure 4: The optimal motion trajectories of three players in Example1

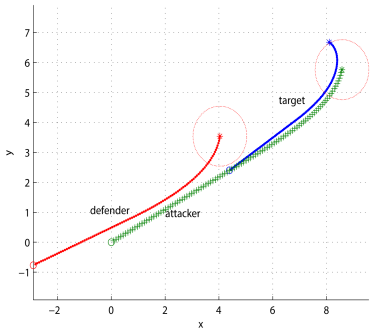


Figure 7: The optimal motion trajectories of three players in Example2

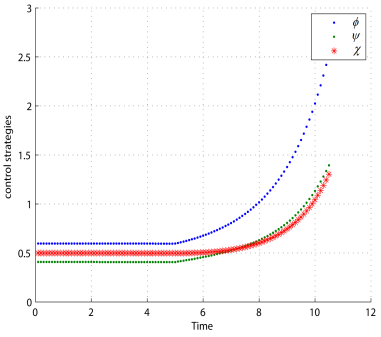


Figure 5: The optimal heading angles of three players in Example2

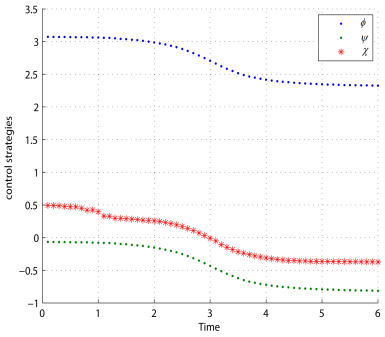


Figure 8: The optimal heading angles of three players in Example3

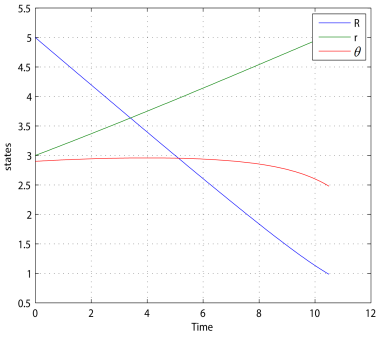


Figure 6: The optimal state trajectories of three players in Example2

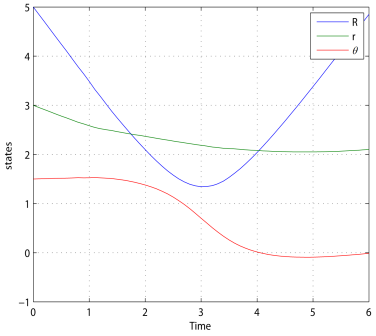


Figure 9: The optimal state trajectories of three players in Example3

the initial value of the control variables, we obtain the numerical solutions of this problem and illustrate the simulation results. In different initial conditions, three players adopt this optimal strategies may cause different outcome: the Attacker or the team of Target and Defender wins the game, even more no one wins the game. In the further, we will analyze the winning conditions for each player involving the initial position,speed ratio, capture radius and the weighting coefficients. Also, we will construct such a barrier to delineate the winning region for each player and obtain a complete solution for this game.

http://www.imavs.org/

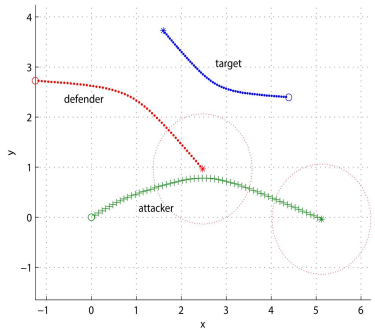


Figure 10: The optimal motion trajectories of three players in Example3

REFERENCES

[1] R. Isaacs. *Differential games.* New York: John Wiley and Sons., 1965.

[2] Tamer Basar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory, 2nd Edition*. Society for Industrial and Applied Mathematics., 1999.

[3] Pierre T. Kabamba and Anouck R. Girard. *Fundamentals of aerospace navigation and guidance*. Cambridge University Press., 2014.

[4] W.M. Getz and M. Pachter. Two-target pursuit-evasion differential games in the plane. *Journal of Optimization Theory and Applications*, 3(34):383–403, 1981.

[5] A.W.Merz. To pursue or to evade-that is the question. *Journal of Guidance Control Dynamics*, 8(2):161–166, 1985.

[6] W. Zha, J. Chen, Z. Peng, and D. Gu. Construction of barrier in a fishing game with point capture. *IEEE Trans Cybernrtics*, 2016.

[7] William Scott and Naomi Ehrich Leonard. Pursuit, herding and evasion: A three-agent model of caribou predation. *Proceedings of the American Control Conference*, pages 2978–2983, 2013.

[8] Shaunak D. Bopardikar, Francesco Bullo, and Joao P. Hespanha. A cooperative homicidal chauffeur game. *AUTOMATICA*, 45(7):1771–1777, 2009.

[9] Sergey Gutman and Rubinsky Shaul. Three body guaranteed pursuit and evasion. *AIAA Guidance, Navigation and Control Conference*, pages 1–24, 2012.

[10] Sergey Rubinsky and Shaul Gutman. Three-player pursuit and evasion conflict. *Journal of Guidance, Control, and Dynamics*, 37(1):98–110, 2014.

[11] Vitaly Shaferman and Tal Shima. Cooperative multiple-model adaptive guidance for an aircraft defending missile. *Journal of Guidance, Control, and Dynamics*, 33(6):1801–1813, 2010.

[12] Tal Shima. Optimal cooperative pursuit and evasion strategies against a homing missile. *Journal of Guidance, Control, and Dynamics*, 34(2):414–425, 2011.

[13] David W. Casbeer, Eloy Garcia, and Zachariah E. Cooperative target defense differential game with a constrained-maneuverable defender. 2015.

[14] Eloy Garcia, David W. Casbeer, and Meir Pachter. Active target defense differential game with a fast defender. *Proceedings of the American Control Conference*, pages 3752–3757, 2015.

[15] Meir Pachter, Eloy Garcia, and David W. Casbeer. Active target defense differential game. *Proceedings of the American Control Conference*, pages 3752–3757, 2015.

[16] Eloy Garcia, David W. Casbeer, and Meir Pachter. Cooperative strategies for optimal aircraft defense from an attacking missile. *Journal of Guidance, Control, and Dynamics*, 38(8):1510–1520, 2015.

[17] Oleg Prokopov and Tal Shima. Linear quadratic optimal cooperative strategies for active aircraft protection. *Journal of Guidance, Control, and Dynamics*, 36(3):753–764, 2013.

[18] Perelman Andrey, Shima Tal, and Rusnak Ilan. Cooperative differential games strategies for active aircraft protection from a homing missile. *journal of guidance Control and Dynamics*, 34(3):761–773, 2011.

[19] Zhang Hongwen, Zhang Kenan, and Chen Wanchun. Indirect method for trajectory optimization of hypersonic vehicle with static parameters. *Journal of Beijing University of Aeronautics and Astronautics*, (02):141–147, 2014.

[20] Emran Tohidi and Hassan Saberi Nik. A bessel collocation method for solving fractional optimal control problems. *Applied Mathematical Modelling*, 39(2):455–465, 2015.

http://www.imavs.org/

Numerical Investigation of a Proof-of-Concept Rotor in Martian Atmosphere

P. Lv[1]*, J.M. Fu[2], Z.F. Xia[1], F. Mohd-Zawawi[3], E. Benard[4], and J.M. Moschetta[4]
[1]China Academy of Aerospace Aerodynamics, Beijing 100074, China
[2]Shanghai Academy of Spaceflight Technology, Shanghai 201109, China
[3]Universiti Teknologi Malaysia, Johor 81310, Malaysia
[4]Institut Supérieur de l'Aéronautique et de l'Espace, Toulouse 31400, France

http://www.imavs.org/

ABSTRACT

This paper presents 2-D Blade Element Momentum Theory predictions and 3-D Reynolds-averaged Navier-Stokes calculations for a proof-of-concept rotor in Martian atmosphere. The rotor that operates on Mars will experience a unique low Reynolds number and high Mach number environment, thus it presents a new challenge. The sectional aerodynamic characteristics of Eppler 387 airfoil are obtained from a 2-D Reynolds-Averaged Navier-Stokes solver. Based on the sectional aerodynamic characteristics, the thrust coefficient and Figure of Merit of rotor are initially predicted by Blade Element Momentum Theory. Then, the rotor with a collective pitch of 10° is investigated using a 3-D Reynolds-Averaged Navier Stokes solver. The results show a clear discrepancy between 2-D predictions and 3-D calculations due to 3-D rotation effect, such as stall modification and root vortices. In addition, the 3-D calculations are not sensitive to the usage of turbulent model or transition model when compared to 2-D predictions.

1 INTRODUCTION

During the past fifty years, planetary exploration has progressed greatly, especially on Mars. The rovers have been launched successfully and provided invaluable information of Mars [1]. Recently, rotary wing vehicle has attracted the attention of the Mars exploration community [2–5]. In contrast to rover, rotary wing vehicle has greater range, speed and field of view. Moreover, it will not be restricted by the unprepared Martian terrain. The concept and feasibility of Martian rotary wing vehicle have been demonstrated by the Army/NASA Rotorcraft Division at NASA Ames Research Center. A NASA Mars rotor prototype was initially proposed based on a coaxial configuration [6]. The study mainly described numerical methods for use in Mars rotor design.

*Email address: peng.lv@outlook.com

Martian atmosphere proposes unique design problems for rotary wing vehicle, as seen in Table 1. On Mars, the gravity is 0.4 times that of the gravity on Earth while the pressure is 0.008 times that of the pressure on Earth. According to Martian density and viscosity, the Reynolds numbers on Mars is 0.02 times what is on Earth. Besides, the speed of sound on Mars is 0.7 times that of the speed of sound on Earth. Therefore, producing the required lift will generate a very high tip Mach number in Martian atmosphere. Finally. the rotor that operates on Mars will suffer a unique low Reynolds number and high Mach number environment, which proposes unconventional design requirement.

Table 1: Comparison of Mean Atmospheric Properties.

	Earth	Mars
Gravity, $\mathrm{m/s^2}$	9.81	3.66
Pressure, Pa	1.0135e5	790.53
Temperature, K	288.16	210.56
Density, $\mathrm{kg/m^3}$	1.225	0.0167
Viscosity, $\mathrm{kg/(ms)}$	1.789e-5	1.289e-5
Gas constant, $\mathrm{J/kg \cdot K}$	287	192
Adiabatic coefficient	1.4	1.29
Speed of sound, $\mathrm{m/s}$	320	230
Atmospheric Gases	$78\%N_2$	$95\%CO_2$
	$21\%O_2$	2.7% N_2

The first Chinese Mars obiter is Yinghuo 1, as shown in Figure 1. The scientific objectives of Yinghuo 1 were:



Figure 1: Yinghuo 1 Mars obiter.

http://www.imavs.org/

- To conduct investigation of plasma environment and magnetic field;
- To study ion escape processes and mechanisms;
- To perform ionosphere occultation measurement;
- To observe sandstorms on the Martian surface.

In 2011, Yinghuo 1 obiter and Fobos-Grunt spacecraft were launched from Baikonur Cosmodrome, Russia. However, the subsequent rocket burns failed and Yinghuo 1 could not enter Mars orbit. In 2016, the head of China National Space Administration (CNSA) announced that China will launch a new Mars probe in 2020 and start Mars exploration. Under this circumstance, Martian rotary wing vehicle is proposed for the potential exploration mission. Taking NASA rotor prototype as a reference, a small-scale proof-of-concept rotor is proposed based on the stowed dimension of Yinghuo 1 (length 0.75m, width 0.75, and height 0.6m). Table 2 lists the specifications of the proof-of-concept rotor. A conventional low Reynolds number airfoil Eppler 387 is selected for the proof of concept rotor. Notice that 40% root cut-out is to simulate blade telescoping required for transport and storage. The typical helicopters operating on Earth usually have tip Mach number of 0.8, but parts of the flow can accelerate to Mach 1. In Martian atmosphere, tip Mach number 0.65 is chosen to keep the blade far from sonic conditions.

Table 2: Proof-of-concept rotor description.

Number of blades	4
Rotor radius	0.25m
Blade root cut-out	40%
Airfoil	Eppler 387
Mach number at R	0.65
Reynolds number at R	5,900
Mach number at 0.75R	0.48
Reynolds number at 0.75R	4,400
Blade chord	0.03
Linear twist rate	+2.4° to –2.4°

In present study, a Blade Element Momentum Theory (BEMT) model is employed to analyze the proof-of-concept rotor. The sectional aerodynamic characteristics used for BEMT is calculated from a 2-D Reynolds-Averaged Navier-Stokes (RANS) solver. Then, a 3-D Reynolds-Averaged Navier Stokes solver is used to investigate the rotor with a collective pitch of 10°. This paper aims at exploring the numerical methods for future Mars rotor design and assessment.

2 2-D BEMT PREDICTIONS

The BEMT is a method that combines the blade element theory and one dimensional momentum theory. BEMT allows the inflow distribution along the blade to be estimated with the assumption that the elements in spanwise direction have no mutual effects on each other. Twist distribution, chord distribution, airfoil characteristics and operation condition are needed for a BEMT analysis. The rotor disk area can be discretized into successive annuli with an area $dA = 2\pi y dy$ [7]. The axial climb velocity V_c can be generalized by free-stream velocity V_∞, say $V_c = V_\infty$. Based on one dimensional momentum theory, the incremental thrust on the rotor annulus can be calculated as the product of the mass flow rate through the annulus and twice induced velocity at the section. The incremental thrust on the annulus is:

$$dT = 2\rho(V_\infty + v_i)v_i dA = 4\pi\rho(V_\infty + v_i)v_i y dy, \quad (1)$$

where ρ is the density of the fluid and v_i is the induced velocity. Using non-dimensional quantities, Equation 1 becomes:

$$dC_T = \frac{2\rho(V_\infty + v_i)v_i dA}{\rho(\pi R^2)(\Omega R)^2} = 4(\frac{V_\infty + v_i}{\Omega R})(\frac{v_i}{\Omega R})(\frac{y}{R})d(\frac{y}{R}), \quad (2)$$

where R is the blade radius and Ω is the rotational speed. The incremental thrust coefficient can be simply written as:

$$dC_T = 4\lambda\lambda_i r dr = 4\lambda(\lambda - \lambda_\infty)r dr, \quad (3)$$

where λ is the total inflow ratio, λ_i is the induced inflow ratio, λ_∞ is the tip speed ratio, r is the non-dimensional radial location and dr is the non-dimensional length of each element.

Tip vortex results in an increased inflow at blade tip, reducing the effective angles of attack there. Therefore, the lift produced at the outer sections of each blade gradually decreases with the radius increasing. Blade tip effect can be considered in BEMT model using the Prandtl tip loss function. The correction factor F is expressed by:

$$F = \frac{2}{\pi}\text{acos}(e^{-f}), \quad (4)$$

and f is defined by:

$$f = \frac{N_b}{2}(\frac{1-r}{r\phi}), \quad (5)$$

$$\phi = \frac{\lambda(r)}{r}, \quad (6)$$

where N_b is the number of blades and ϕ is the inflow angle. The correction factor F can also be included in the one dimensional momentum model to change the flow velocity through the annulus. Hence, Equation 3 becomes:

$$dC_T = 4F\lambda(\lambda - \lambda_\infty)r dr. \quad (7)$$

Using blade element theory, the incremental thrust coefficient on the annulus is:

$$dC_T = \frac{1}{2}\sigma C_l r^2 dr = \frac{\sigma C_{l\alpha}}{2}(\theta r^2 - \lambda r)dr, \quad (8)$$

where σ is the local solidity ($\sigma = \frac{N_b c}{\pi R}$, c is the local chord of blade), C_l is the lift coefficient, $C_{l\alpha}$ is the 2-D lift-curve-slope of airfoil and θ is the local pitch angle. The local pitch angle θ is defined by:

$$\theta = \theta_0 + \theta_{tw}, \quad (9)$$

http://www.imavs.org/

where θ_0 is the collective pitch angle and θ_{tw} is the twist angle. Equating the incremental thrust coefficients from momentum theory (Equation 7) and blade element theory (Equation 8) gives:

$$4F\lambda(\lambda - \lambda_\infty)rdr = \frac{\sigma C_{l\alpha}}{2}(\theta r^2 - \lambda r)dr. \quad (10)$$

Rearranging Equation. 10 in terms of λ, the canonic form is obtained:

$$\lambda^2 + (\frac{\sigma C_{l\alpha}}{8F} - \lambda_\infty)\lambda - \frac{\sigma C_{l\alpha}}{8F}\theta r = 0. \quad (11)$$

This quadratic equation in λ has the positive solution:

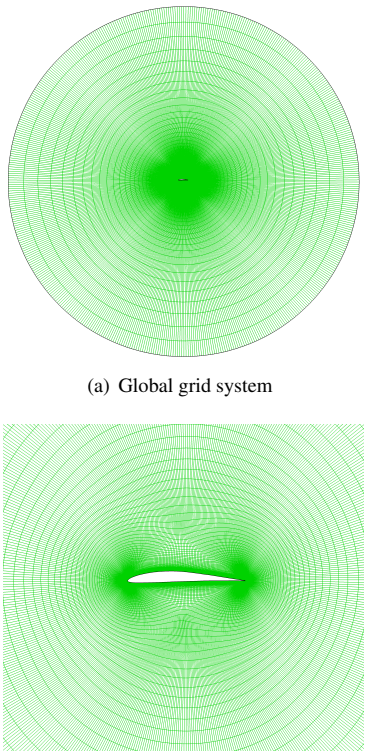
$$\lambda(r, \lambda_\infty) = \sqrt{(\frac{\sigma C_{l\alpha}}{16F} - \frac{\lambda_\infty}{2})^2 + \frac{\sigma C_{l\alpha}}{8F}\theta r} - (\frac{\sigma C_{l\alpha}}{16F} - \frac{\lambda_\infty}{2}). \quad (12)$$

Equation 12 can be solved numerically by discretizing the blades into a series of small spanwise elements. It is solved iteratively by first assuming $F = 1$ and then finding F from Equation 4 and recalculating λ from Equation 12, up to convergence within a specified percentage. Usually, a few iterations are required to achieve convergence and obtain the inflow solution. BEMT results can provide total thrust coefficient C_T and total power coefficient C_P, which allow the calculation of Figure of Merit (FM) by:

$$\text{FM} = \frac{C_T^{\frac{3}{2}}/\sqrt{2}}{C_P}. \quad (13)$$

In the BEMT model, the airfoil characteristics are included by the 2-D lift-curve-slope of airfoil $C_{l\alpha}$. A constant $C_{l\alpha}$ was originally used to analyze the full-scale rotors. However, at low Reynolds number, the lift coefficient curves can be highly nonlinear. For low Reynolds number aerodynamics, under an adverse pressure gradient of sufficient magnitude, the laminar fluid flow tends to separate before becoming turbulent. After separation, the flow structure becomes increasingly irregular. Beyond a certain threshold, it undergoes transition from laminar to turbulent. The turbulent mixing process brings high momentum fluid from the free stream to the near wall region, which can overcome the adverse pressure gradient, causing the flow reattachment and Laminar Separation Bubble (LSB). LSB can increase drag and produce nonlinear lift and moment for the airfoil [8,9]. To increase prediction accuracy, a table look-up was implemented into BEMT model. The lift, drag and moment coefficients for a given airfoil are tabulated as function of local angle of attack and Reynolds number.

The sectional aerodynamic characteristics of Eppler 387 airfoil were obtained from a 2-D Reynolds-averaged Navier-Stokes solver. Standard $k - \epsilon$ turbulent model and $\gamma - \text{Re}_\theta$ transitioned model were employed to simulate fully turbulent flow and transitioned flow, respectively. The calculations were run at the representative Mach number 0.48 and Reynolds number 4,400, for a series of angles of attack. The structured grids (497x112 points) were used to accommodate Eppler 387 airfoil, as shown in Figure 2. The distance from boundary to airfoil was selected as $15c$.



(a) Global grid system

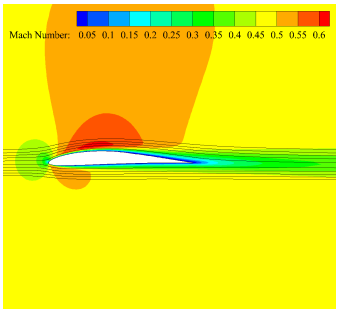
(b) Close-up of grids near airfoil

Figure 2: 2-D grids on Eppler 387 airfoil.

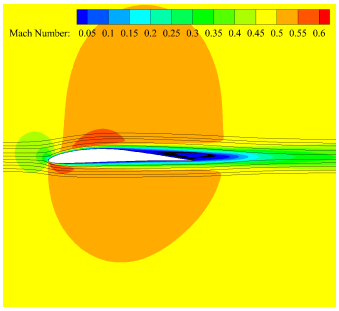
The Mach number contours from 2-D RANS calculations on the Eppler 387 airfoil are shown in Figure 3 and 4. At angle of attack $\alpha = 0°$, the flow field obtained from $\gamma - \text{Re}_\theta$ transitioned model exhibits a separation vortex which is being shed off the trailing edge of the airfoil. The flow phenomenon fairly agrees with the observations by Mayda et al. in wind turbine airfoil study [10]. They found that, at low angle of attack, the vortex shedding on the Eppler 387 airfoil is generally induced by LSB. However, the LSB is not seen in present study. By contrast, the separation vortex does not appear in the flow field obtained from $k - \epsilon$ turbulent model. At angle of attack $\alpha = 10°$, the separation vortices are clearly observed from the two cases. In contrast to turbulent case, the transitioned case shows a larger vortex structure and earlier separation.

Figure 5 shows the aerodynamic characteristics of Eppler 387 airfoil calculated by 2-D RANS solver. The lift coefficients and lift-to-drag ratio from $\gamma - \text{Re}_\theta$ transitioned model are lower than those from $k - \epsilon$ turbulent model. This is mainly caused by the earlier separation.

http://www.imavs.org/

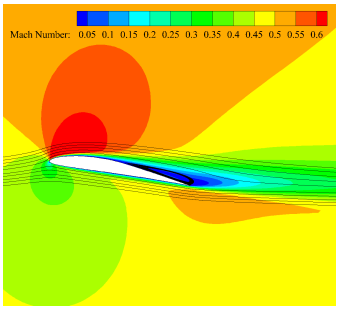


(a) $k-\epsilon$ turbulent model

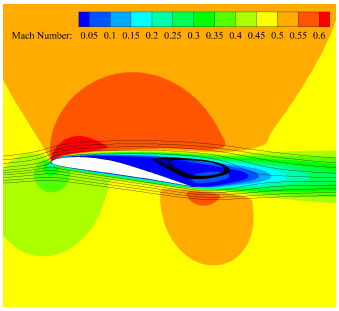


(b) $\gamma - \mathrm{Re}_\theta$ transition model

Figure 3: Mach number contours at angle of attack $\alpha = 0°$ (Re=4,400, Ma=0.48).

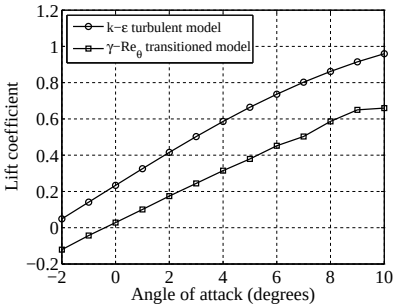


(a) $k-\epsilon$ turbulent model

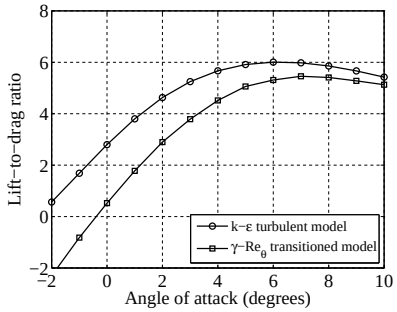


(b) $\gamma - \mathrm{Re}_\theta$ transitioned model

Figure 4: Mach number contours at angle of attack $\alpha = 10°$ (Re=4,400, Ma=0.48).

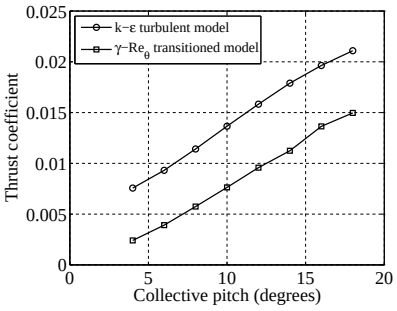


(a) Lift coefficient

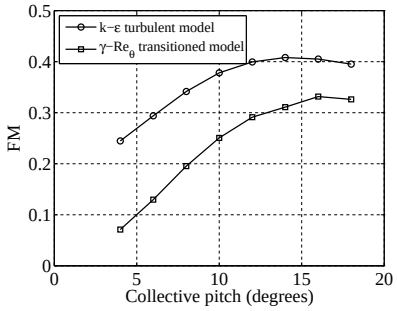


(b) Lift-to-drag ratio

Figure 5: Aerodynamic characteristics of Eppler 387 airfoil (Re=4,400, Ma=0.48).



(a) Thrust coefficient



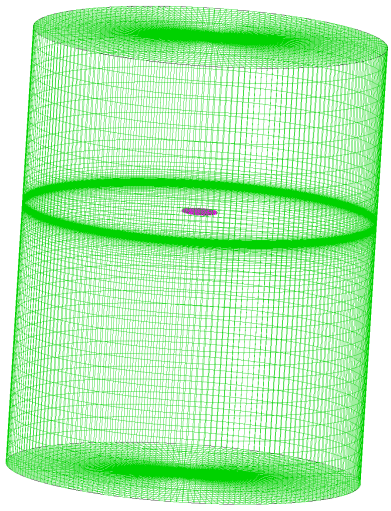
(b) Figure of Merit

Figure 6: Performance predictions of the rotor.

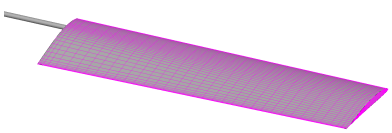
The representative aerodynamic characteristics were then used for BEMT predictions. Figure 6 shows the performance predictions of the proof-of-concept rotor in hover for a series of collective pitch angles. The thrust coefficients and FM values of transitioned case are lower than the counterparts of turbulent case. The maximum value of FM is as low as 0.4, which results from the combination of low Reynolds number and high Mach number. For both cases, the stall behavior of rotor seems to start from the collective pitch of 10°.

3 3-D RANS CALCULATIONS

The proof-of-concept rotor calculations were computed through a 3-D RANS solver, including $k-\epsilon$ turbulent case and $\gamma-\mathrm{Re}_\theta$ transitioned case. The dual-time stepping scheme was used for the unsteady simulations. The time step was chosen as 0.0001s while the number of sub-iteration was fixed by 20. The calculations were performed for the proof-of-concept rotor with a collective pitch of 10°. The 3-D structured grids on proof-of-concept rotor consisted of near-body part (2,500,000 volumes) and off-body part (2,000,000 volumes), as seen in Figure 7. The radius of the flow domain was set to $12R$. Besides, the distance from inlet to rotor was chosen as $12R$ while the distance from rotor to outlet was increased to $18R$.

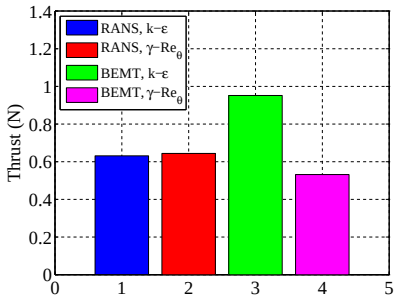


(a) Global grid system

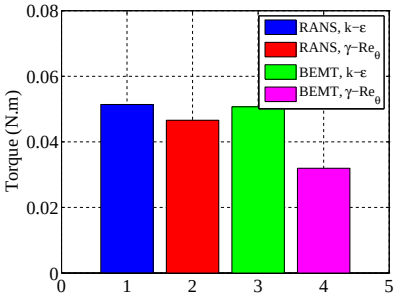


(b) Close-up of grids on blade surface

Figure 7: 3-D grids on the rotor.

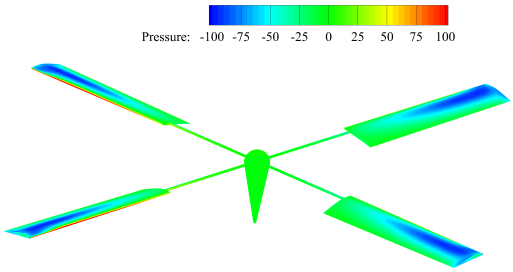


(a) Thrust

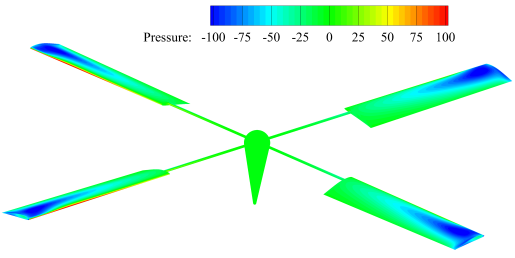


(b) Torque

Figure 8: Results comparisons between 2-D BEMT and 3-D RANS.



(a) $k-\epsilon$ turbulent model

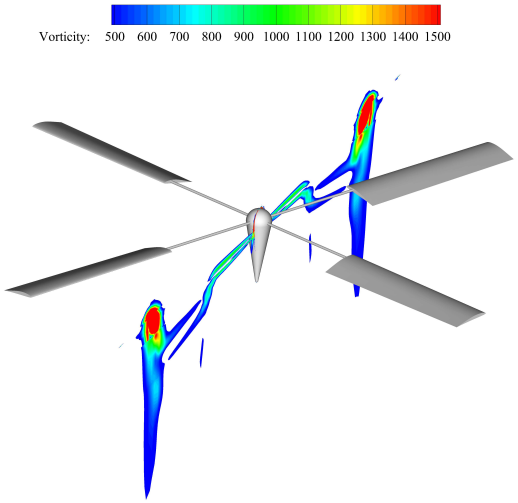


(b) $\gamma-\mathrm{Re}_\theta$ transitioned model

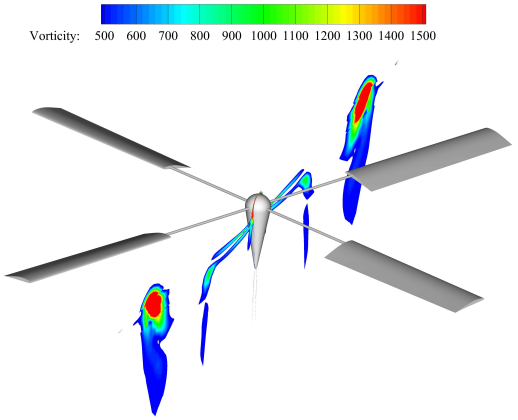
Figure 9: Pressure distributions on the upper surface of rotor blades (units: Pa).

http://www.imavs.org/

The results of thrust and torque are compared between 2-D BEMT predictions and 3-D RANS calculations, as shown in Figure 8. The results show a clear discrepancy between 2-D predictions and 3-D calculations due to 3-D rotation effect, such as stall modification and root vortices. In addition, the 3-D calculations are not sensitive to the usage of turbulent model or transition model when compared to 2-D predictions. This is further examined by the insight into the physics of pressure distributions. The similar pressure distributions on upper surface of the rotor blades are observed for the two cases, as demonstrated in Figure 9.



(a) $k-\epsilon$ turbulent model



(b) $\gamma - \mathrm{Re}_\theta$ transitioned model

Figure 10: Vorticity magnitudes on a slice located 60° behind rotor blades (units: s^{-1}).

For the two cases, the vorticity magnitudes on a slice located 60° behind the rotor blades are shown in Figure 10 shows. The tip vortices are captured successfully by the 3-D RANS calculations. The root vortices are apparent due to the 40% root cut-out, which can not be modeled in present BEMT method.

4 CONCLUSION

This paper presents 2-D BEMT predictions and 3-D RANS calculations for a proof-of-concept rotor that operates in Martian atmosphere. The proof-of-concept rotor experiences a unique low Reynolds number and high Mach number environment. A BEMT model is implemented to analyze the proof-of-concept rotor. The sectional aerodynamic characteristics used for BEMT is calculated from a 2-D RANS solver. Then, a 3-D RANS solver is employed to investigate the rotor with a collective pitch of 10°. The results show a clear discrepancy between 2-D predictions and 3-D calculations due to 3-D rotation effect, such as stall modification and root vortices. Besides, the 3-D calculations are not sensitive to the selection of turbulent model or transition model when compared to 2-D predictions. The tip vortices are captured successfully by the 3-D RANS calculations. The root vortices occur due to the 40% root cut-out, which can not be modeled in present BEMT method. In the future, the BEMT method is expected to be improved in terms of aerodynamic characteristics correction and root vortex loss. Combining BEMT predictions and RANS calculations, the potential Mars rotor will be designed and assessed.

ACKNOWLEDGEMENTS

The authors would like to thank Guo Xiaoyan, project manager at CNSA for helpful discussion on this work.

REFERENCES

[1] N.G. Barlow. *Mars: an introduction to its interior, surface and atmosphere*. Cambridge University Press, Cambridge, UK, 2014.

[2] L.A. Young, R.T. Chen, E.W. Aiken, and G.A. Briggs. Design opportunities and challenges in the development of vertical lift planetary aerial vehicles. In *American Helicopter Society International Vertical Lift Aircraft Design Specialist's Meeting, San Francisco, CA*, 2000.

[3] L.A. Young and E.W. Aiken. Vertical lift planetary aerial vehicles: three planetary bodies and four conceptual design cases. In *27th European Rotorcraft Forum, Moscow, Russia*, 2001.

[4] A. Datta, B. Roget, D. Griffiths, and G. Pugliese. Design of a martian autonomous rotary-wing vehicle. *Journal of Aircraft*, 40(3):461–472, 2003.

[5] J. Fuentes and R.P. Kaluarachchi. Martian rhover feasibility study. In *53rd AIAA Aerospace Sciences Meeting, Kissimmee, Florida*, AIAA 2015-0001, 2015.

http://www.imavs.org/

[6] K.J. Corfeld, R.C. Strawn, and L.N. Long. Computational analysis of a prototype martian rotorcraft experiment. In *20th AIAA Applied Aerodynamics Conference, St. Louis, Missouri*, AIAA 2002-2815, 2002.

[7] J.G. Leishman. *Principles of helicopter aerodynamics*. Cambridge University Press, New York, USA, 2006.

[8] W. Shyy, Y. Lian, J. Tang, D. Viieru, and H. Liu. *Aerodynamics of low Reynolds number flyers*. Cambridge University Press, New York, USA, 2008.

[9] E.V. Laitone. Wind tunnel tests of wings at reynolds numbers below 70 000. *Experiments in Fluids*, 23(5):405–409, 1997.

[10] E.A. Mayda, C.P. van Dam, and E.P. Duque. Bubble induced unsteadiness on wind turbine airfoils. In *AIAA Wind Energy Conference, Reno, NV*, AIAA 2002-0033, 2002.

http://www.imavs.org/

Optimal Distributed Cooperative Control for Multiple Quadrotors: Heading Consensus Experiment with Paparazzi Autopilot and AR. Drone 2

Alireza Mohamadifard [*], Abolghasem Naghash [†], Ali Jamei [‡]

http://www.imavs.org/

Abstract

An LQR control strategy for distributed coordination of multi-vehicle systems with linear dynamics is studied in this article. Using partitioning method, linear consensus algorithms for multi-vehicle systems with single-integrator dynamics is extended for high-order integral dynamics. Then the proposed high-order consensus protocol, is modelled as an optimal linear quadratic regulator (LQR) problem. Then, consensus protocol law is modified to achieve consensus merely on state variables while applying an optimal internal controller on higher states derivatives. A theorem is proposed to formulate the consensus strategy for this control problem and the stability analysis of multi-vehicle system is discussed using Lyapunov stability criterion. To illustrate the direct application of the proposed control theory and demonstrate the effectiveness of the proposed optimal high-order consensus protocol, linear second order heading dynamics of quadrotor is utilized to implement the consensus law on a real high-order dynamic MAV. Paparazzi open-source autopilot and the Parrot AR.Drone II commercial quadrotors are providing the software and hardware test bed for this experiment.

Keywords: Optimal Consensus, linear quadratic regulator (LQR), coordinated formation.

1. Introduction

Multi-Agent Systems are groups of vehicles capable of imitating human behavior as a team in the sense of coordination, cooperation and consensus. Agents in this venue are mobile robots in general operating on air, on the surface and below water. With emphasis to aerospace application, these agents are flying vehicles such as UAVs in formation, MAVs in cooperation and satellites in constellation. Intelligence of the vehicle in this regard can be defined as the ability of observing, decision making and directing its activities towards achieving an exclusive or inclusive goals.

There are tasks that sole vehicle cannot afford due to limitations in sensors, performance and cost. Therefore advantages such as scalability, flexibility, robustness, fault tolerance, easy maintenance, and less expensive schemes, can be achieved by having a group of vehicles work in collaboration with each other. As an important approach for these cooperative missions, cooperative control have received extensive attention [1] in recent control research areas. Formation control approaches

[*] PhD student, Dept of Aerospace Engineering, Amirkabir University of Technology, Iran. Email: alireza.mfard@aut.ac.ir
[†] Associate Professor, Dept of Aerospace Engineering, Amirkabir University of Technology, Iran. Email: naghash@aut.ac.ir
[‡] M.S. student, Dept of Aerospace Engineering, Amirkabir University of Technology, Iran.

http://www.imavs.org/

like Swarming [2], distributed optimization [3], [4] and [5] potential field [6], Spatiotemporal planning like rendezvous [7], reconfiguration [8], coverage [9], cooperative tasking like assignment protocols [10] and consensus are among the famous categories in this field. The latter, i.e. consensus algorithms is a prevalent strategy in decentralized multi-vehicle cooperative control which have been studied widely in the literature. Consensus is based on exchanging the local neighbor-to-neighbor information between the vehicles, and it can be defined as: state-convergence of vehicles on a common value by negotiating with their neighbors or more conceptually, a distributed decision making process to agree on a value of interest. On the other hand, formation control can be proposed as maintaining desired spacing between the agents' position, velocity and higher order of derivatives relative to each other or relative to a reference. The combination of these two approaches lead to consensus formation control definition that can be proposed as: state-agreement of vehicles on particular values to provide a desired shape between vehicles states such as polygon.

Consensus problem is subject of research in both analysis and synthesis approaches. Typical problem specification in multi agent systems consensus algorithms include systems with time-delay communication links [11] and [12], time-dependent communication links [13], measurement noise [14], stochastic topology [15], analysis of graph Laplacian matrix theory [16], finite time convergence [17] and average seeking problems [18]. Regarding consensus protocol synthesis with respect to dynamics of agents, consensus laws are proposed for linear dynamics, including single, double and high-order systems; and nonlinear dynamics. In [19], consensus algorithms for single integrator kinematics was studied in different settings. Consensus for vehicles with double-integrator dynamics was studied in [20] wherein inspects the effect of relative state derivative availability to all agents or a subset of them. High-order integral dynamics was investigated in [21] which also presents the idea of high-order consensus with leader and the concept of model reference consensus problem. Regarding Consensus problem in nonlinear model context, the work in [22] is focused on Euler-Lagrangian nonlinear system for distributed coordination problem with directed topology consumption. Non-linear protocols have been proposed in [23] for optimal distributed consensus. Consensus reaching via distributed algorithms for general functions can be found in [24]. Optimal linear consensus algorithms for vehicles with single-integrator kinematics in a continuous-time setting from an LQR perspective have been studied in [19]. [25] has worked on optimal consensus protocol with the purpose of reaching minimum time convergence by deriving an optimal weighting matrix. Optimal average consensus problem was solved in [26], based on the optimal interaction graph approach. [27] designed distributed consensus algorithms which minimizes a collective objective function. In [28], the Linear Matrix Inequality (LMI) approach is addressed for optimization of consensus algorithm problems. Regarding optimal LQR control for input affine linear systems, [29] proposed a structured performance index to address the topology of the graph using an inverse optimality together with partial stability approach. [30] Proposes a framework for cooperative tracking control, considering a leader node as command generator. It also shows the duality of cooperative controller design and cooperative observer design in network systems. Taking external disturbance into account, in [31] a robust controller is proposed for a directed network of multiple agents with high-order integral dynamics, via defining an appropriate controlled output and a model transformation, which transfers the consensus performance into a normal $H\infty$ problem.

With emerging different control methodologies, optimality is one of the vital issues which is considered in the control science literature. Optimal considerations improve performance characteristics of control algorithms. In the case of linear differential dynamic systems with quadratic cost function, LQR theory is one of the most fundamental optimal control methods. Although, numerous linear-consensus algorithms investigate the optimality issues, mentioned above, optimal consensus for high-order dynamic systems is ignored. While in reality, a wide group of vehicles require second or higher order dynamic model. Dealing with high-order derivatives of states for LQR-based consensus formulation, makes the present work different from [19]. On the other hand, [32] deals with the same network system as of this paper, considering also the higher order terms in its proposed consensus protocol, but lacks in proposing an optimal controller formulation. Also unlike [29] that takes the inverse optimal approach where Laplacian matrix is known, we employ the direct approach in which Laplacian matrix in unknown and has to be defined via solution of the Riccati equation. In this sense, we comply with the prevalence in LQR problem

venue in which Q and R matrices are design parameters and Laplacian matrix, being the optimal state/output feedback controller gain, is derived based on the solution of Algebraic Riccati equation (ARE).

Overally, the optimal LQR-based consensus algorithm contribution in the current research is twice. First, the work of Yongcan Cao and Wei Ren [19] is extended, in three aspects: 1) the high-order linear differential dynamics of vehicles are proposed in new partitioned form, 2) the optimal Laplacian matrix determination problem for high-order dynamics is modeled based on LQR theory and 3) the optimal LQR based consensus protocol for high-order dynamics is derived. Second, with the aim of preventing the consensus-reached value of states to vary, there should be consensus only on states as well as regulation on higher states derivatives, which leads to a novel optimal consensus with internal controller protocol for high-order dynamics. For this optimal control problem, a theorem with its stability proof based on Lyapunov function is formulated in this regard.

This paper is structured as follows: In Section 2, a brief definition of graph theory is presented. In Section 3, preliminaries of consensus problem for high-order dynamic system is described. Section 4 presents the optimal consensus and optimal internal controller formulation for the multi vehicle system via a theorem with stability analysis. Linear dynamic representation for quadrotor with focus on heading channel along with the special form of the proposed protocol for this channel are addressed in section 5. Finally, experimental results are given in Sections 6 while Section 7 contains conclusion and future work.

2. Graph theory

In general, the interaction graph between a group of n agents can be modeled by directed/undirected graph. Let $G(v, \varepsilon, \mathcal{A})$ be a directed graph, where $v = \{1, \dots, n\}$ is set of nodes, $\varepsilon \in v^2$ is an edge set of regular pairs of nodes and $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{n\times n}$ is nonnegative adjacency matrix.

In a directed graph, edge $(i, j) \in \varepsilon$ notes that agent i can send its information to agent j, but it does not imply $(j, i) \in \varepsilon$. Whereas in undirected graph edge $(i, j) \in \varepsilon$ yields to $(j, i) \in \varepsilon$. The adjacency matrix represents the interconnection of adjacent nodes, that can be defined as $a_{ii} = 0, a_{ij} > 0$ if $(j, i) \in v$ and $i \neq j$. The nonsymmetrical Laplacian matrix $\mathcal{L} = [\ell_{ij}] \in \mathbb{R}^{n\times n}$ is defined as $\ell_{ii} = \sum_{j\neq i} a_{ij}$ and $\ell_{ij} = -a_{ij}$, where $i \neq j$. The Laplacian matrix of undirected graph is symmetric positive semi-definite. In contrast, in directed graph $\mathcal{L}$ does not have this property. Let 1 be defined as $n \times 1$ column vector of all ones. With respect to the definition of $\mathcal{L}$, the Laplacian matrix has zero eigenvalue with an associated eigenvector 1. For undirected information exchange graph, $\mathcal{L}$ has simple zero eigenvalue if and only if the graph is connected. In case of directed graph, $\mathcal{L}$ has simple zero eigenvalue if and only if the directed information exchange graph has a directed spanning tree.

3. Dynamic system preliminaries

3.1 L^{th}-order dynamic system modeling

Assume n agents with L^{th}-order-integrator dynamics (n-by-L dynamics) is presented by:

$$\xi_i^{(L)}(t) = u_i(t), \qquad i \in \{1, \dots, n\}, \tag{1}$$

Where $\xi_i(t) \in \mathbb{R}$ is state parameter, $\xi_i^{(l)}(t)$ denotes L^{th} derivative of state, $u_i(t) \in \mathbb{R}$ is the control input and i^{th} presents the system number.

With respect to the existence of n agents that each of them is an Lth-order-integrator dynamic system, the matrix form statement of all agent dynamics prompts better conception. General matrix form of the system dynamics can be written by matrix partitioning technique. Consider

$$x^{(l)}(t) = \left[\xi_1^{(l)}(t), \dots, \xi_n^{(l)}(t)\right]^T, \qquad l \in \{1, \dots, L\}, \tag{2}$$

Therefore the matrix form is expressed as

$$\dot{\mathrm{X}}(t) = A\mathrm{X}(t) + BU(t), \tag{3}$$

With

$$\mathrm{X}(t) = \left[x(t), \dots, x^{(L-1)}(t)\right]^T, \quad U(t) = [u_1(t), \dots, u_n(t)]^T \tag{4}$$

And also

$$A = I_n \otimes \begin{bmatrix} 0_{(L-1)\times 1} & I_{(L-1)\times(L-1)} \\ 0_{1\times 1} & 0_{1\times(L-1)} \end{bmatrix} \tag{5}$$

http://www.imavs.org/

$$= \begin{bmatrix} 0_{(\{L-1\}*n)\times n} & I_{(\{L-1\}*n)\times(\{L-1\}*n)} \\ 0_{n\times n} & 0_{n\times(\{L-1\}*n)} \end{bmatrix},$$

And

$$B = I_n \otimes \begin{bmatrix} 0_{(L-1)\times 1} \\ I_{1\times 1} \end{bmatrix} = \begin{bmatrix} 0_{(\{L-1\}*n)\times n} \\ I_{n\times n} \end{bmatrix}, \quad (6)$$

Where $x^{(l)}(t) \in \mathbb{R}^{n\times 1}$, $\mathrm{X}(t) \in \mathbb{R}^{(L*n)\times 1}$, $\mathrm{U}(t) \in \mathbb{R}^{n\times 1}$ is vector of control inputs, $\mathrm{A} \in \mathbb{R}^{(L*n)\times(L*n)}$ and $\mathrm{B} \in \mathbb{R}^{(L*n)\times n}$ are constant coefficient matrices. I is identity matrix with dimensions according to the A and B matrices definition.

3.2 L^{th}-order dynamics consensus algorithms

To address the consensus formulation, we define the error state as:

$$\tilde{\mathrm{X}}(\mathrm{t}) = \mathrm{X}(\mathrm{t}) - \mathrm{X}_\mathrm{c} \quad (7)$$

Where $\tilde{\mathrm{X}}$ is the state error and X_c is the final consensus value of each agents' state and higher derivatives. According to equation (4) we have:

$$\tilde{\mathrm{X}}(\mathrm{t}) = [\tilde{\mathrm{x}}, \dots, \tilde{\mathrm{x}}^{(\mathrm{L}-1)}] \quad (8)$$

And according to equation (2) we have:

$$\tilde{\mathrm{x}}^{(\mathrm{l})} = \left[\left(\xi_1^{(\mathrm{l})}(\mathrm{t}) - \xi_{1\mathrm{c}}^{(\mathrm{l})}\right), \dots, \left(\xi_\mathrm{n}^{(\mathrm{l})}(\mathrm{t}) - \xi_{\mathrm{nc}}^{(\mathrm{l})}\right)\right]^\mathrm{T}, \mathrm{l} \in \{0,1,\dots,\mathrm{L}\} \quad (9)$$

Where $\xi_{\mathrm{ic}}^{(\mathrm{l})}, \mathrm{i} \in \{1,\dots,\mathrm{n}\}, \mathrm{l} \in \{0,1,\dots,\mathrm{L}\}$ are the final consensus values for each agents' state i of order l and is reached when $\left\| \xi_\mathrm{i}^{(\mathrm{l})}(\mathrm{t}) - \xi_\mathrm{j}^{(\mathrm{l})}(\mathrm{t}) \right\|_{\mathrm{i}\neq\mathrm{j}} \to 0$.

Based on the above descriptions, the error dynamics or disagreement dynamics is:

$$\dot{\tilde{\mathrm{X}}}(\mathrm{t}) = \mathrm{A}\tilde{\mathrm{X}}(\mathrm{t}) + \mathrm{BU}(\mathrm{t}) \quad (10)$$

The consensus algorithm of the Lth-order dynamic systems is expressed as [18]:

$$u_i(t) = -\sum_{j=1}^{n} a_{ij}\left[\sum_{k=0}^{L-1}\left(\gamma_{ij}\right)_k \left(\xi_i^{(k)}(t) - \xi_j^{(k)}(t)\right)\right], \quad (11)$$

Where $\left(\gamma_{ij}\right)_0 = 1$, $\left(\gamma_{ij}\right)_k > 0$, $k = 0,\dots,L-1$, $i,j = 1,\dots,n$, $\xi_i^{(k)}(t)$ presents k^{th} derivative of $\xi_i(t)$ that $\xi_i^{(0)}(t) = \xi_i(t)$, a_{ij} is (j,i) th entry of weighted adjacency matrix associated with interaction graph between the agents with $a_{ii} \triangleq 0$.

It can be noted that by applying protocol (11), each agent just needs its local neighbors' information. In case of n agents, the objective of (11) is declared, $\xi_i^{(k)}(t) \to \xi_j^{(k)}(t), k = \{0,\dots,L-1\}, \forall i \neq j$ as $t \to \infty$, which is the aim of consensus algorithms.

Interaction graph between a group of n agents is modeled by unidirectional graph $(\nu, \varepsilon, \mathcal{A})$. From the definition of $\mathcal{L}$ (Laplacian matrix) and also by substituting equation (11) in equation (3), general dynamics matrix form of n agents with L^{th}-order-integrator dynamic that are controlled by consensus algorithm can be defined as

$$\dot{\tilde{\mathrm{X}}}(\mathrm{t}) = -\left(\begin{bmatrix} 0_{(\{\mathrm{L}-1\}*\mathrm{n})\times\mathrm{n}} & \mathrm{I}_{(\{\mathrm{L}-1\}*\mathrm{n})\times(\{\mathrm{L}-1\}*\mathrm{n})} \\ \mathcal{L}_{\mathrm{n}\times\mathrm{n}} & \Lambda_{\mathrm{n}\times(\{\mathrm{L}-1\}*\mathrm{n})} \end{bmatrix}\right)\tilde{\mathrm{X}}(\mathrm{t}), \quad (12)$$

With

$$\Lambda = [\,\eta_1\mathcal{L}, \dots, \eta_{L-1}\mathcal{L}\,], \quad (13)$$

It is also worthy noticing that the pair $(\mathrm{a_{ii}}, \gamma_{\mathrm{ij}})$ in (11), plays the same role as the pair $(\mathcal{L}, \eta)$ in (16), while regarding to the definition of Laplacian matrix $\mathcal{L}$ in section 2, as the Laplacian matrix $\mathcal{L}$ is associated with $\mathcal{A}$, the η is associated with Γ, Where $\Gamma = [\gamma_{\mathrm{ij}}] \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$. It is clear that equation (16) is a linear differential equation. Considering the definition of $\mathrm{X}(\mathrm{t})$ in equation (4), equation (16) can be expressed as follows: The last $n-th$ entry of the $\dot{\tilde{\mathrm{X}}}(\mathrm{t})$ matrix, corresponds to the control inputs as are defined in equation (4) and can be written as following matrix form equation:

http://www.imavs.org/

$$U(t) = -\mathcal{K}\,\mathrm{X}(t)\,,$$
where
$$\mathcal{K} = [\mathcal{L} \quad \eta_1\mathcal{L} \quad \dots \quad \eta_{L-1}\mathcal{L}]\,, \tag{14}$$

Therefore, the consensus protocol of n agents with L^{th}-order dynamics based on Laplacian matrix of interaction graph between the agents i.e. equation (14), can be presented in algebraic form by:

$$U(t) = -\sum_{k=0}^{L-1} \eta_k \mathcal{L} x^{(k)}(t), \tag{15}$$

Where $\boldsymbol{\eta_0 = I_{n\times n}}$, $\boldsymbol{\eta_k} \in \mathbb{R}^{\boldsymbol{n\times n}}$ is unknown constant coefficients matrix that are associated with $\boldsymbol{k^{th}}$ derivative of $\boldsymbol{x(t)}$ and $\boldsymbol{k = 0, \dots, L-1}$.

4. Optimal cooperative controllers

4.1 LQR-consensus controller

From equation (3), it is clear that general matrix form of n agents with L^{th}-order-integrator dynamics is modeled as a system with linear differential equation therefore similar to the optimal-control problems, the consensus cost function can be proposed as

$$J = \int_0^\infty \{\sum_{i=1}^{n}\sum_{j=1}^{n-1}\left(\sum_{k=0}^{L-1}\sum_{k=0}^{k-1}(q_{ij})_{k\acute{k}}\left(\xi_i^{(k)}(t) - \xi_j^{(k)}(t)\right)^2\right) + \sum_{i=1}^{n} r_i\, u_i^2(t)\}\, dt, \tag{16}$$

Where $(q_{\mathrm{ij}})_{k\acute{k}} \geq 0 \in \mathbb{R}$ and $\mathrm{r_i} > 0 \in \mathbb{R}$ can be chosen freely; therefore, J in equation (16), can be called as an interaction-free cost function. Therefore the optimization problem can be proposed as

$$min_{a_{ij},\gamma_k}\ J\ , \text{ subject to (3) and (15)}, \tag{17}$$

According to protocol (15), deriving the optimal consensus algorithms is equivalent to deriving optimal Laplacian matrix $\mathcal{L}$ and optimal constant coefficients matrix η_k. to derive an optimal solution, continuous-time setting of linear quadratic regulator (LQR) algorithm is hereby utilized. Optimal control problem (17) subject to linear system which is defined as equation (3) and interaction related cost function (16) can be written as

$$min_{a_{ij},\gamma_k}\ J = \int_0^\infty [\mathrm{X}^T(t)\ Q\ \mathrm{X}(t) + U^T(t)\ R\ U(t)]\ dt, \tag{18}$$
$$\text{Subject to } \dot{\mathrm{X}}(t) = A\mathrm{X}(t) + BU(t),$$
$$U(t) = -\sum_{k=0}^{L-1} \eta_k \mathcal{L} x^{(k)}(t),$$
$$\text{Where, } \mathrm{X}(t) = \left[x(t), \dots, x^{(L-1)}(t)\right]^T,$$

Where $\mathrm{X}(t), U(t), A, B$ are defined in (3), $x^{(k)}$ is defined in (2), $Q = [q_{ij}] \in \mathbb{R}^{(L*n)\times(L*n)}$ is a symmetric positive semi-definite (PSD) design matrix associated with the multi agent system. Each diagonal element $q_{ii} \in \mathbb{R}^{n\times n}$ represents designer's desired weighted interaction on the agents' states and its higher derivatives of the same type, and the non-diagonal terms accounts for the cross-type weighting between states and derivatives i.e. q_{ij}, $i \neq j$ are transversal weighting design matrices. In its simplest form, we can ignore cross-type weighting and also assign a sole design matrix q_{11} on the states and expand it for higher derivatives. . q_{11} is chosen as a symmetric Laplacian matrix associated with a connected graph. In this case, Q can be defined in the Kronecker form of equation (21). $\mathrm{R} \in \mathbb{R}^{n\times n}$ is a positive-definite diagonal matrix which r_i is ith diagonal entry and $\mathcal{L}, \eta_k$ are optimal Laplacian and optimal constant coefficients matrix that is defined in (15) and are going to be derived. This problem's solution can be explained by following steps.

Step 1: With the aim of deriving the general equation of optimal Laplacian $\mathcal{L}$ and constant coefficients matrix η_k for n agents with L^{th}-order-integrator dynamic, constant coefficients matrix in equation (18) should be written in partitioned matrix form. With respect to equations (5) and (6), it can be proposed as

$$A = \begin{bmatrix} 0 & I & 0 & 0 & \dots & 0 \\ 0 & 0 & I & 0 & \dots & 0 \\ 0 & 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & I \\ 0 & 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \tag{19}$$

http://www.imavs.org/

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ I \end{bmatrix}, \tag{20}$$

$$Q = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1L} \\ q_{21} & q_{22} & \cdots & q_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ q_{L1} & q_{L2} & \ldots & q_{LL} \end{bmatrix} = (I_L \otimes q_{11}), \tag{21}$$

Where A, B are defined in equations (5) and (6), respectively and Q is defined in equation (18). Each of the matrices entries denotes an n-by-n matrix (i.e., a matrix with n rows and n columns). Whereas, R is defined in equation (18) as an n-by-n matrix, does not need partitioning. 0 and I donate null and identity matrices, respectively.

Step 2: In this step, the optimal problem (18) is alternated to standard LQR problem by considering simple form of controller input and then analytical solution for n-by-L dynamics (3), is derived. The simplified problem can be expressed as

$$\begin{aligned} &min_{U(t)}\ J \\ &\text{Subject to: } \dot{\mathrm{X}}(t) = A\mathrm{X}(t) + BU(t), \\ &\qquad\qquad U(t) = -\mathcal{K}\mathrm{X}(t), \\ &\text{Where, } \mathcal{K} = R^{-1}B^T P \end{aligned} \tag{22}$$

Where J is defined in equation (18), $\mathcal{K} \in \mathbb{R}^{n\times(L*n)}$ is optimal control command's coefficient matrix and $P = \left[p_{ij}\right] \in \mathbb{R}^{(L*n)\times(L*n)}$ satisfies the continuous-time algebraic Riccati equation

$$A^T P + PA - PBR^{-1}B^T P + Q = 0_{(L*n)\times(L*n)} \tag{23}$$

Where each of the matrix entries p_{ij}, denote an n-by-n matrix. For solving the simplified optimal problem (22), first, by applying equations (19), (20), (21) and (24) with respect to R is defined in (18), equation (23) can be written in matrix form

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1L} \\ p_{21} & p_{22} & \cdots & p_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ p_{L1} & p_{L2} & \ldots & p_{LL} \end{bmatrix}, \tag{24}$$

$$\begin{bmatrix} \Pi_{1_{n\times n}} & \Pi_{2_{n\times(\{L-1\}*n)}} \\ \Pi_{3_{(\{L-1\}*n)\times n}} & \Pi_{4_{(\{L-1\}*n)\times(\{L-1\}*n)}} \end{bmatrix} = 0_{(L*n)\times(L*n)}, \tag{25}$$

Where, $\Pi_i\ i = \{1, \ldots, 4\}$ is proposed in equation (22). For n agents with L^{th}-order-integrator dynamic with specific interaction graph among agents G, L and Q are known, and R is noted weighted control coefficient matrix, which is supposed to be known for all direct optimal problem. Therefore, by solving matrix system of equation (25), $P = \left[p_{ij}\right]$ can be computed by (26).

On the other hand, applying equations (20) and (24) with assuming R as presented by (18), simplified control command matrix equation (22), can be given by following partitioned matrix form equation

$$\begin{aligned} \Pi_1 &= -R^{-1}p_{1L}p_{L1} + q_{11}, \\ \Pi_2 &= [p_{11} - R^{-1}p_{1L}p_{L2} + q_{12}, \ldots, p_{(L-1)1} - R^{-1}p_{1L}p_{LL} + q_{1L}], \\ \Pi_3 &= [p_{11} - R^{-1}p_{2L}p_{L1} + q_{21}, \ldots, p_{(L-1)1} - R^{-1}p_{LL}p_{L1} + q_{L1}], \\ \Pi_4 &= \begin{bmatrix} p_{12} + p_{12} - R^{-1}p_{2L}p_{L2} + q_{22} & \cdots & p_{1L} + p_{(L-1)2} - R^{-1}p_{2L}p_{LL} + q_{2L} \\ \vdots & \ddots & \vdots \\ p_{1L} + p_{(L-1)2} - R^{-1}p_{LL}p_{L2} + q_{L2} & \cdots & p_{(L-1)L} + p_{(L-1)L} - R^{-1}p_{LL}p_{LL}\, q_{LL} \end{bmatrix}, \end{aligned} \tag{26}$$

http://www.imavs.org/

$$U(t) = -[R^{-1}\wp_{L1}, \dots, R^{-1}\wp_{LL}]\,\mathrm{X} \tag{27}$$

This equation represents the optimal LQR-based control command which is the foundation of general form of optimal consensus control protocol.

4.2 LQR- state Consensus with internal control on derivatives

One popular approach in consensus problems is to reach an agreement only on the state such as position and orientation, forcing and keeping higher orders of that state zero[31-32]. For example in the case that $\xi_i(t)$ denotes the position of i th agent, $\xi_i^{(1)}(t), \xi_i^{(2)}(t), \dots, \xi_i^{(L)}(t)$, denote velocity, acceleration and L^{th} derivative of position state of ith agent respectively. If the condition is to hold the consensus-reached position, thus the velocities and other derivatives of position must be driven to zero. It means that, applying consensus protocol on states and exerting an internal controller on higher order derivatives. Therefore, the equation (11) should be rewritten as

$$\begin{aligned} u_i'(t) = &-\sum_{j=1}^{n} a_{ij}\{\xi_i(t) - \xi_j(t)\} \\ &-\sum_{j=1}^{n} a_{ij}\{\sum_{k=1}^{L-1}(\gamma_{ij})_k\,\xi_i^{(k)}(t)\}, \\ &i \in \{1, \dots, n\} \end{aligned} \tag{28}$$

With regard to the relation between the adjacency matrix and degree matrix arguments for undirected graph of n agents, i.e. $\sum_{j=1}^{n} a_{ij} = d_i, i \in \{1, \dots, n\}$ [33], where d_i denotes the degree of ith agent, and also, as $\sum_{j=1}^{n} a_{ij}$ is independent of the counter of the second sigma, it follows that

$$\begin{aligned} u_i'(t) = &-\sum_{j=1}^{n} a_{ij}\{\xi_i(t) - \xi_j(t)\} \\ &-\sum_{k=1}^{L-1} d_i.(\gamma_{ij})_k\,\xi_i^{(k)}(t), \\ &i \in \{1, \dots, n\} \end{aligned} \tag{29}$$

Rewriting this equation in the matrix form, like what has been done for equation (15) yields to

$$U'(t) = -\mathcal{L}\tilde{x}(t) - \sum_{k=1}^{L-1} \eth_k D\, x^{(k)}(t), \tag{30}$$

Where $D = diag\{d_i\} \in \mathbb{R}^{n\times n}$ is the degree matrix of interaction graph between agents. Considering $\mathcal{L} = D - \mathcal{A}$ [33], the equation can be expressed as

$$U'(t) = \mathcal{A}x(t) - \sum_{k=1}^{L-1} \delta_k D\tilde{x}^{(k)}(t), \tag{31}$$

$\delta_k \in \mathbb{R}^{n\times n}$ where $\delta_k D$ is equivalent but not equal to $\eta_k\mathcal{L}$ for $k = \{1, \dots, L-1\}$ and $\Delta = [\,\delta_1 D, \dots, \delta_{L-1} D\,]$ is equivalent to Λ in (13). By defining $W \in \mathbb{R}^{n\times(L*n)}$ where, $w_{ij} = a_{ij}$ for $i\ and\ j \in \{1, \dots, n\}$ and $w_{ij} = 0$ for $j > n$, the equation (31) is proposed as

$$U'(t) = W\tilde{X}(t) - \sum_{k=1}^{L-1} \delta_k D\tilde{x}^{(k)}(t), \tag{32}$$

As of step 2 presentation in the part 4.1, in order to reach the optimal consensus protocol with optimal internal controller, one should equalize the consensus protocol (15) and optimal control command (22). Hence, in the same stream, the above equation will be as follows

$$\begin{aligned} U'(t) &= W\tilde{\mathrm{X}}(t) - \mathcal{K}\tilde{X}(t) \\ \mathcal{K} &= [D \quad \delta_1 D \quad \dots \quad \delta_{L-1} D], \end{aligned} \tag{33}$$

At last, according to the definition of optimal control command in equation (22), the optimal consensus with internal controller protocol illustrated as

$$U'(t) = W\tilde{\mathrm{X}}(t) - U(t), \tag{34}$$

Where U is optimal internal control command for state derivatives while the consensus attributes lay in $W\mathrm{X}(t)$ term. While just control command changes, the new optimal control problem subject to the linear system (3) is derived by substituting (33) in equation (18), as

$$\begin{aligned} min_{a_{ij},\gamma_k}\ J = &\int_0^{\infty} \left[\mathrm{X}^T(t)\,Q\tilde{\mathrm{X}}(t) + U'^T(t)R\,U'(t)\right] dt, \\ &\text{Subject to: } \dot{\tilde{\mathrm{X}}}(t) = A\tilde{\mathrm{X}}(t) + BU'(t), \\ &U'(t) = W\tilde{\mathrm{X}}(t) - \mathcal{K}\tilde{\mathrm{X}}(t), \\ &\text{Where, } \tilde{\mathrm{X}}(t) = \left[\tilde{x}(t), \dots, \tilde{x}^{(L-1)}(t)\right]^T, \end{aligned} \tag{35}$$

http://www.imavs.org/

In order to write the new optimal problem in a comparable form with equation (18), i.e. (defining the new problem with use of parameters of equation (18)), the equation (34) should be substitute in equation (35)

$$J = \int_0^\infty [\tilde{X}^T(t)Q\,\tilde{X}(t) + (W\tilde{X}(t) - U(t))^T(t)R\,(W\tilde{X}(t) - U(t))(t)]\,dt, \quad (36)$$

$$J = \int_0^\infty [\tilde{X}^T(t)\,Q\,\tilde{X}(t) + X^T(t)\,W^T RW\,\tilde{X}(t) + U^T(t)RW\,\tilde{X}(t) + \tilde{X}^T(t)\,W^T R\,U(t) + U^T(t)R\,U(t)]\,dt,$$

By defining $Q' = Q + W^T RW$ and $\mathcal{K}' = R^{-1}(B^T P - R_1)$, where, $R_1 = R\,W$, and assuming that $U^T(t)RW\,X(t)$ is a symmetric matrix, the new optimal problem can expressed as

$$min_{a_{ij},\gamma_k}\ J = \int_0^\infty [\tilde{X}^T(t)\,Q'\,\tilde{X}(t) + 2\tilde{X}^T(t)\,R_1\,U(t) + U^T(t)R\,U(t)]\,dt, \quad (37)$$

Subject to: $\dot{\tilde{X}}(t) = A\tilde{X}(t) + BU(t)$,

$U(t) = -\mathcal{K}'\tilde{X}(t)$,

Where, $\tilde{X}(t) = [\tilde{x}(t), \dots, \tilde{x}^{(L-1)}(t)]^T$,

$Q' = Q + W^T RW$,

$\mathcal{K}' = R^{-1}(B^T P - R_1)$,

Where, $R_1 = R\,W$

Before moving on, the following lemma succeeded by a theorem are introduced to prove asymptotic stability of the proposed optimal control command.

Lemma 1. Consider the dynamical system

$$\dot{\tilde{X}}(t) = f\left(\tilde{X}(t), U(t)\right), \quad \tilde{X}(0) = \tilde{X}_0, \quad t \geq 0, \quad (38)$$

With $f(0,0) = 0$ and a cost function proposed by

$$J(\tilde{X}_0, U(0)) = \int_0^\infty T\left(\tilde{X}(t), U(t)\right)\,dt, \quad (39)$$

Where the state feedback controller $U(t) = \phi\left(\tilde{X}(t)\right)$ minimizes the cost function and the $U(0)$ is an admissible control which can be defined as

$$U(0) = \phi\left(\tilde{X}(0)\right), \quad (40)$$

Let $D \subseteq \mathbb{R}^n$ be an open set and $\Omega \subseteq \mathbb{R}^L$, assume $0 \in D$, let $f: D \to \mathbb{R}^n$, $\phi: D \to \Omega$, and T: $D \to \mathbb{R}$. Assume there is a continuously differentiable function $V: D \to \mathbb{R}$ such that

$$V(0) = 0, \quad (41)$$

$$V(\tilde{X}) > 0, \quad \tilde{X} \in D, \quad \tilde{X} \neq 0, \quad (42)$$

$$V'(\tilde{X}) f\left(X(t), \phi\left(\tilde{X}(t)\right)\right) < 0, \quad \tilde{X} \in D, \tilde{X} \neq 0, \quad (43)$$

$$T\left(\tilde{X}(t), \phi(X(t))\right) + V'(\tilde{X}) f\left(\tilde{X}(t), \phi(\tilde{X}(t))\right) = 0, \tilde{X} \in D, \quad (44)$$

Then the zero solution $\tilde{X}(t) \equiv 0$ of closed-loop system

$$\dot{\tilde{X}}(t) = f\left(\tilde{X}(t), \phi(\tilde{X}(t))\right), \quad \tilde{X}(0) = \tilde{X}_0, \quad t \geq 0 \quad (45)$$

Is locally asymptotically stable, and

$$J\left(\tilde{X}_0, U(0)\right) = V(\tilde{X}_0), \quad \tilde{X}_0 \in D, \quad (46)$$

Finally, if $D = \mathbb{R}^n$, $\Omega = \mathbb{R}^L$, and

$$V(\tilde{X}) \to \infty \quad as \quad \|\tilde{X}\| \to \infty, \quad (47)$$

Then the zero solution $X(t) \equiv 0$ of closed-loop system is globally asymptotically stable.

Proof: omitted. Refer to [34].

According to the multi-agent system dynamics representation of equations (1) to (6), and also definitions in (37), this study is confined to a specific form of system (22), where $U(t) = -\mathcal{K}'\tilde{X}(t)$ and $\mathcal{K}' = R^{-1}(B^T P - R_1)$ and the formal optimal regulator equation $\mathcal{K} = R^{-1}B^T P$ in which by substituting the B as defined in equation (6), is $\mathcal{K} = R^{-1}[\mathcal{P}_{L1}, \dots, \mathcal{P}_{LL}]$, thus we have $\mathcal{K}' = R^{-1}[(\mathcal{P}_{L1} - R_1), \mathcal{P}_{L2}, \dots, \mathcal{P}_{LL}]$.

Step 3: As the last step, the optimal Laplacian and optimal constant coefficients matrix should be derived. It can be done by according the extracted control law matrix $\mathcal{K}'$ mentioned above, to the control command

http://www.imavs.org/

matrix form of equation (33). First, by considering $\mathcal{K}'$ the command law U' can be rewritten

$$U'(t) = -\{(R^{-1}\mathcal{P}_{L1} - \mathcal{A})\tilde{x}(t) + R^{-1}\mathcal{P}_{L2}\dot{\tilde{x}}(t) \dots + R^{-1}\mathcal{P}_{LL}\tilde{x}^{(L-1)}(t)\}, \quad (48)$$

And expansion form of equation (31) is given by

$$U'(t) = -\{\delta_0 D\tilde{x}(t) + \dots + \delta_{(L-1)} D\tilde{x}^{(L-1)}(t)\}, \quad (49)$$

By matching equations (48) and (49), the optimal Laplacian matrix, $\mathcal{L}$ and optimal constant coefficients matrix, $\eta_{(k)}$, where $k = 1, \dots, L-1$ are derived as

$$D = R^{-1}\mathcal{P}_{L1},$$

And,

$$\delta_0 = \mathcal{L}D^{-1}$$
$$\delta_{(k)} = R^{-1}\mathcal{P}_{L(k+1)}D^{-1}, \quad k \in \{1, \dots, L-1\}, \quad (50)$$

Theorem 1. The matrix $\mathcal{L}$ in Equation (50) is a Laplacian matrix and is equal to $\sqrt{\mathrm{R}^{-1}\, q_{11}}$.

Proof: According to the first partition of Equation (26), we have $-\mathrm{R}^{-1}\mathcal{P}_{1\mathrm{L}}\mathcal{P}_{\mathrm{L}1} + q_{11} = 0$, on the other side, with regard to LQR problem setting, we know that P is symmetric matrix, meaning that $\mathcal{P}_{1\mathrm{L}} = \mathcal{P}_{\mathrm{L}1}$. Therefore, it is followed that $\mathcal{P}_{\mathrm{L}1} = \sqrt{\mathrm{R}\, q_{11}}$.

The proof is finalized by substituting $\mathcal{P}_{\mathrm{L}1}$ into Equation (50), which is yield to $\mathcal{L} = \mathrm{R}^{-1}\mathcal{P}_{\mathrm{L}1} = \sqrt{\mathrm{R}^{-1}\, q_{11}}$. So by Assigning q_{11} as a symmetric Laplacian matrix with a simple zero eigenvalue associated with a connected graph, $\mathrm{R}^{-1}\, q_{11}$ is a (non-symmetric) Laplacian matrix with a simple zero eigenvalue, since R is a positive definite diagonal matrix.

Therefore, the partitioned matrix form of optimal consensus protocol for n agents with L^{th} -order-integrator dynamic is derived as

$$U(t) = -\sum_{k=0}^{L-1} \delta_k D\tilde{x}^{(k)}(t), \quad (51)$$

$U(t)$ and $P = [\mathcal{P}_{ij}]$ are defined in equations (4) and (20) , respectively. D and δ_k are n-by-n matrices.

With the aim of deriving the optimal consensus algorithm of n agents with L^{th} -order-integrator dynamic, the optimal consensus protocol (2) should be rewritten in basic consensus algorithm form (11). According to the presented procedure of deriving equations (12-15) by use of equations (3) and (11), the optimal consensus algorithm is given by

$$u_i(t) = -\sum_{j=1}^{n} a_{ij}\{\xi_i(t) - \xi_j(t)\} - \sum_{k=1}^{L-1} d_i.(\gamma_{ij})_k \xi_i^{(k)}(t), \quad i \in \{1, \dots, n\},$$

Where,

$$\begin{cases} \Gamma_k = [\gamma_{ij}]_k \\ \Gamma_0 = 1_{n\times n} \quad k = 0, \\ \Gamma_k = (R^{-1}\mathcal{P}_{L(k+1)})\mathcal{A}^{-1} \quad k \neq 0 \end{cases} \quad (52)$$

$$\mathcal{A} = [a_{ij}], \quad where \begin{cases} a_{ij} = -\ell_{ij} & i \neq j \\ a_{ij} = 0 & i = j \end{cases},$$

$$D = [d_i], \quad i \in \{1, \dots, n\},$$

Note that from the equality of (48) and (49), we have $R^{-1}\mathcal{P}_{L(k+1)} = \delta_k D$, and we already know that D is a diagonal matrix, thus $R^{-1}\mathcal{P}_{L(k+1)}, k = \{0, \dots, L-1\}$ are all degree matrices.

Based on the above lemma and the afterward explanations, we propose the following theorem complying the aforementioned control protocols for linear system and quadratic performance index.

Theorem 1. Consider the linear dynamical system

$$\dot{\tilde{\mathrm{X}}}(t) = A\tilde{\mathrm{X}}(t) + BU(t), \quad \tilde{\mathrm{X}}(0) = \tilde{\mathrm{X}}_0, \quad t \geq 0,$$
$$y(t) = -\mathcal{K}'\left(\tilde{\mathrm{X}}(t)\right), \quad (53)$$

With Quadratic performance function

$$J\left(\tilde{\mathrm{X}}_0, U(0)\right) = \int_0^\infty \left(\tilde{\mathrm{X}}^T(t)\, Q'\, \tilde{\mathrm{X}}(t) + 2\tilde{\mathrm{X}}^T(t)\, R_1\, U(t) + U^T(t) R\, U(t)\right)dt, \quad (54)$$

Where $U(t) = -\mathcal{K}'\,\tilde{\mathrm{X}}(t)$ is the state feedback controller, which the optimal regulator is $\mathcal{K}' = R^{-1}(B^T P - R_1)$, and $U(0)$ is an admissible control.

Then, $V = \tilde{\mathrm{X}}^T P\tilde{\mathrm{X}}$ can be proposed as a candidate Lyapunov function, and the zero solution of closed-loop system

$$\dot{\tilde{\mathrm{X}}}(t) = (A - B\mathcal{K}')\tilde{\mathrm{X}}(t), \quad \tilde{\mathrm{X}}(0) = \tilde{\mathrm{X}}_0, \quad t \geq 0, \quad (55)$$

Is globally asymptotically stable.

http://www.imavs.org/

Proof: By using lemma 1, if the candidate Lyapunov function $V = \tilde{\mathrm{X}}^T P\tilde{\mathrm{X}}$ can satisfies all required conditions, hence, for all $\tilde{\mathrm{X}} \in D$ this theorem is applicable .

The time derivative of candidate Lyapunov function yields

$$\dot{V}(\mathrm{X(t)}) = \dot{\mathrm{X}}(t)^T P\,\mathrm{X(t)} + \mathrm{X(t)}^T P\,\dot{\mathrm{X}}(t), \tag{56}$$

By substituting the closed-loop system equation (55) and optimal regulator $\mathcal{K}' = R^{-1}(B^T P - R_1)$

$$\begin{aligned}\dot{V}(\mathrm{X(t)}) = {} & [(A - B\mathcal{K}')\mathrm{X(t)}]^T P\,\mathrm{X(t)} \\ & + \mathrm{X(t)}^T P\,[(A - B\mathcal{K}')\mathrm{X(t)}], \\ = \mathrm{X(t)}^T & [(A - B\{R^{-1}(B^T P - R_1)\})^T P \\ & + P(A - B\{R^{-1}(B^T P - R_1)\})]\mathrm{X(t)}, \\ = \mathrm{X(t)}^T & [PA + A^T P - 2PBR^{-1}B^T P \\ & + R_1 R^{-1}B^T P + PBR^{-1}R_1\,]\mathrm{X(t)},\end{aligned} \tag{57}$$

Due to the algebraic Riccati equation which is correspond to quadratic performance function (54) [35], there exist the following equality

$$\begin{aligned}0 = {} & PA + A^T P - PBR^{-1}B^T P - R_1 R^{-1}R_1 \\ & + R_1 R^{-1}B^T P + PBR^{-1}R_1 + Q',\end{aligned} \tag{58}$$

The time derivative of candidate Lyapunov function is henceforth represented as

$$\begin{aligned}\dot{V}\left(\tilde{\mathrm{X}}(\mathrm{t})\right) = \tilde{\mathrm{X}}(\mathrm{t})^T[& -Q' + R_1 R^{-1}R_1 \\ & - PBR^{-1}B^T P\,]\tilde{\mathrm{X}}(\mathrm{t}),\end{aligned} \tag{59}$$

Recalling the definitions $Q' = Q + W^T RW$ and $R_1 = R\,W$, one has

$$\dot{V}\big(\mathrm{X(t)}\big) = \tilde{\mathrm{X}}(\mathrm{t})^T[-Q - PBR^{-1}B^T P\,]\tilde{\mathrm{X}}(\mathrm{t}), \tag{60}$$

Reaffirmed that, P is positive-definite matrix which is defined in equation (22), hence for $R > 0$ and $R^{-1} > 0$, $PBR^{-1}B^T P$ is positive-definite matrix too, i.e. $PBR^{-1}B^T P > 0$. Also Q is positive-semi-definite matrix, i.e. $Q \geq 0$ which is defined in equation (18), hence, $PBR^{-1}B^T P + Q > 0$ and $\dot{V} < 0$. Furthermore, as the Frechet derivative $(\acute{\cdots})$ is mentioned by equation (44), hence, $\acute{V}\big(\tilde{\mathrm{X}}(\mathrm{t})\big)\, f\big(\tilde{\mathrm{X}}(t), U(t)\big) < 0$ for all $\tilde{\mathrm{X}} \in D$.

On the other hand, substituting the state feedback controller function in Quadratic performance function (54) yields

$$\begin{aligned}J\big(\mathrm{X}_0, U(0)\big) = {} & \int_0^\infty \big(\tilde{\mathrm{X}}^T(t)\,Q'\,\tilde{\mathrm{X}}(t) + 2\tilde{\mathrm{X}}^T(t)\,R_1\,U(t) \\ & + U^T(t)R\,U(t)\big)dt,\end{aligned} \tag{61}$$

$$\begin{aligned}J\left(\tilde{\mathrm{X}}_0, U(0)\right) = {} & \int_0^\infty \Big(\tilde{\mathrm{X}}(\mathrm{t})^T Q'\,\tilde{\mathrm{X}}(\mathrm{t}) + \\ & 2\tilde{\mathrm{X}}^T(t)\,R_1\left(-R^{-1}(B^T P - R_1)\tilde{\mathrm{X}}(\mathrm{t})\right) + \\ & \left(-R^{-1}(B^T P - R_1)\tilde{\mathrm{X}}(\mathrm{t})\right)^T R\left(-R^{-1}(B^T P - \right. \\ & \left.R_1)\tilde{\mathrm{X}}(\mathrm{t})\right)\Big)\,dt, \\ = {} & \int_0^\infty \tilde{\mathrm{X}}(\mathrm{t})^T[Q' - R_1 R^{-1}R_1 \\ & + PBR^{-1}B^T P\,]\tilde{\mathrm{X}}(\mathrm{t})\,dt,\end{aligned}$$

And hence, by (59)

$$\begin{aligned}J\left(\tilde{\mathrm{X}}_0, U(0)\right) = {} & \int_0^\infty \left(-\dot{V}\left(\tilde{\mathrm{X}}(\mathrm{t})\right)\right)dt \\ = {} & \underbrace{-V\left(\tilde{\mathrm{X}}(\mathrm{t} \to \infty)\right)}_{\to 0} + V\big(\tilde{\mathrm{X}}_0\big) \\ = {} & V\big(\tilde{\mathrm{X}}_0\big), \qquad for\ all\ \tilde{\mathrm{X}}_0 \in D,\end{aligned} \tag{62}$$

Also, by (62), (39) and (61) it can be concluded that

$$\begin{aligned}\mathrm{T}\left(\tilde{\mathrm{X}}(t), \phi\big(\tilde{\mathrm{X}}(\mathrm{t})\big)\right) = {} & -\dot{V}\left(\tilde{\mathrm{X}}(\mathrm{t})\right) \\ = {} & \acute{V}\big(\tilde{\mathrm{X}}\big) f\left(\tilde{\mathrm{X}}(t), \phi\big(\tilde{\mathrm{X}}(\mathrm{t})\big)\right), \qquad \tilde{\mathrm{X}} \in \\ & D,\end{aligned} \tag{63}$$

Then, the last necessary condition is satisfied too, i.e. $\mathrm{T}\big(\tilde{\mathrm{X}}(t), \phi(\tilde{\mathrm{X}}(\mathrm{t}))\big) + \acute{V}\big(\tilde{\mathrm{X}}\big) f\big(\tilde{\mathrm{X}}(t), \phi(\tilde{\mathrm{X}}(\mathrm{t}))\big) = 0,$ for all $\tilde{\mathrm{X}} \in D$. Therefore, the zero solution of closed-loop system (55) is globally asymptotically stable.

The next session discusses the formation strategy but before moving on, one must notice that for a multi agent system with a MIMO-dynamical system for each agent, by defining $\xi|_\mu$ and $u|_\mu$, $\mu \in \{1, \dots, m\}$, the proposed control law (52) can be rewritten as:

http://www.imavs.org/

$$u_i(t)|_\mu = -\sum_{j=1}^{n} a_{ij}\left[\sum_{k=0}^{L-1}(\gamma_{ij})_k\left(\xi_i^{(k)}(t)\big|_\mu - \xi_j^{(k)}(t)\big|_\mu\right)\right], \quad i \in \{1,\dots,n\}, \mu \in \{1,\dots,m\} \tag{64}$$

Where $u_i(t)|_\mu$ and $\xi_i(t)|_\mu$ are the $\mu-$th entries of state and input vectors ξ and u, respectively.

5. Quadrotor Linearized Heading Dynamics

Proposing a simplified model for control design where the aerodynamic, gyroscopic and Coriolis forces and moments are negligible, and assuming small changes for the Euler angles, meaning that $[\varphi, \theta, \psi] \simeq [p, q, r]$ and having linearized the equations of motion around $\varphi_0, \theta_0, \psi_0, u_{1_0}$, we obtain:

$$\begin{cases} \ddot{x} = \theta \\ \ddot{y} = \varphi \\ \ddot{z} = u_1 \\ \ddot{\varphi} = u_2 \\ \ddot{\theta} = u_3 \\ \ddot{\psi} = u_4 \end{cases} \tag{65}$$

We define the state vector $\xi = [x, y, z, \varphi, \theta, \psi]$ and input vector $u = [u_1, u_2, u_3, u_4]$. In the above equations, the input force and torque signals are normalized per mass and moment of inertia for quadrotor, respectively.

Considering the second order integral dynamics for heading (ψ), the optimal consensus algorithm for heading state and its first derivative is

$$u_{4,i} = -\sum_{j=1}^{n} a_{ij}\left[(\psi_i - \psi_j) + \gamma_{ij}\,\dot{\psi}_i\right], i \in \{1,\dots,n\} \tag{66}$$

Since all simulation results, not only for heading state but also for three dimensional position consensus formation is delivered in [36], the reader is referred to that paper for simulations details, thus in this paper we focus on the experimental aspects of this consensus algorithm.

6. Experiment

Paparazzi is an open-source autopilot software that have attracted a lot of attention from academics in the field of drone programming all over the world. Parrot AR.Drone II, on the other side is a ubiquitous commercial quadrotor with open access micro-processor. Paparazzi software can be setup on the AR.Drone II's computer [37]. We employ this combination of hardware-software to implement the consensus protocol (66) in a real-world experiment. WiFi connection between the drones can provide instantaneous data exchange viz. heading angles of the drones amongst each other. Each drone calculates the u_4 command separate on its processer but alternatively command generating for all agents can be performed on a ground computer and uplinked to the relevant drone. Thus each drone must downlink its data to the ground computer and receive the u_4 command back from it. In this sense, the consensus algorithm still has its distributed nature, but the data manipulation and command calculation is implemented in a centralized manner.

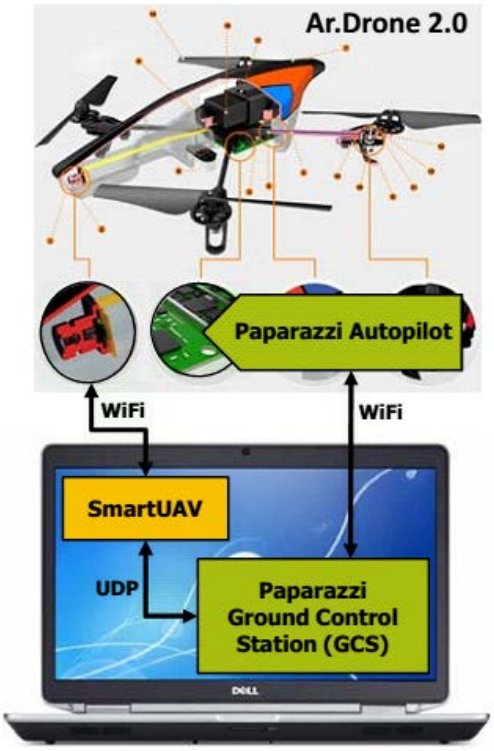


Figure 1. Test bed system architecture [38]

Now, consider a network of 3 quadrotors which are stabilized in hover mode while their heading channels are prone to follow consensus algorithm described by equation (66). Optimal designing parameters Q and R are proposed as follows

http://www.imavs.org/

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} I_L \otimes r_1,$$

$$Q = I_L \otimes q_{11},$$

Where, (67)

$$r_1 = 1,$$

$$q_{11} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix},$$

Where Q is considered as a simplest form which is defined in section 4.1.

It then follows from theorem 1 that the optimal Laplacian matrix $\mathcal{L}$ is given by (68):

$$\mathcal{L} = \begin{bmatrix} 1.1547 & -0.5774 & -0.5774 \\ -0.5774 & 1.1547 & -0.5774 \\ -0.5774 & -0.5774 & 1.1547 \end{bmatrix}, \quad (68)$$

Subsequently, the solution of algebraic Riccati equation (26) for yaw angle (ψ) of equation (65), with $L = 2$ (derivation order), is given by (69)

$$P = \begin{bmatrix} 2.9 & -1.4 & -1.4 & 1.1 & -0.5 & -0.5 \\ -1.4 & 2.9 & -1.4 & -0.5 & 1.1 & -0.5 \\ -1.4 & -1.4 & 2.9 & -0.5 & -0.5 & 1.1 \\ 1.1 & -0.5 & -0.5 & 1.6 & -0.8 & -0.8 \\ -0.5 & 1.1 & -0.5 & -0.8 & 1.6 & -0.8 \\ -0.5 & -0.5 & 1.1 & -0.8 & -0.8 & 1.6 \end{bmatrix}, \quad (69)$$

The following figures show the consensus protocols implemented on a formation 3 quadrotors for their heading states while other independent states viz. 3 position states are regulated for a stable hover via PID controller.

Figure 2 shows initial arbitrary heading values of 64, 110 and 22 degrees for 1st, 2nd and 3rd quadrotors respectively as written near each drone in the flight plan screen.

Figure 3 shows consensus reached value of around 67 degrees for 3 quadrotors with a variation of 1 degree due to magnetometer sensor imperfections, communication delays and other sources of errors. The diagram below the flight plan screen shows the time history of heading trajectories that leads to the consensus value. The planar trajectories of quadrotors in X-Y plane is also depicted in the flight plan screen.

7. Conclusion And Future Works

In this paper, LQR-based optimal consensus algorithms for multiagent systems with L^{th} -order integrator dynamics in contineous-time setting was studied. The basic LQR consensus algorithm was then developed by introducing intenal controller on states derivatives to keep the agreed state values invarient. Stability of the optimal consensus protocol was investigated via Lyapunov function stability criterion. To validate the effectiveness of the expressed consensus protocol, a seccond order consensus algorithm was implemented for heading angles of a set of 3 AR Drone 2 quadrotors using Paparazzi autopilot software.

Future work will be dedicated to include position states in the consensus algorithm experiment.

http://www.imavs.org/

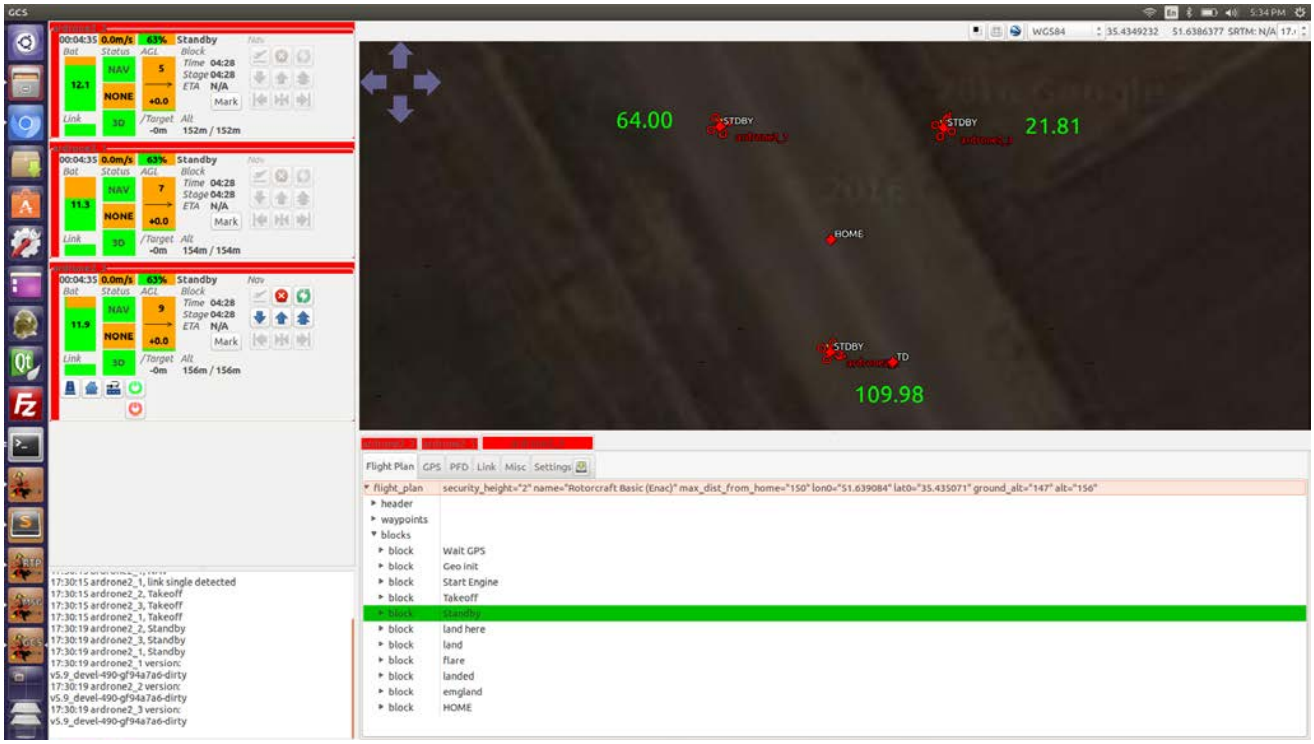


Figure 2. Initial heading values for 3 quadrotors

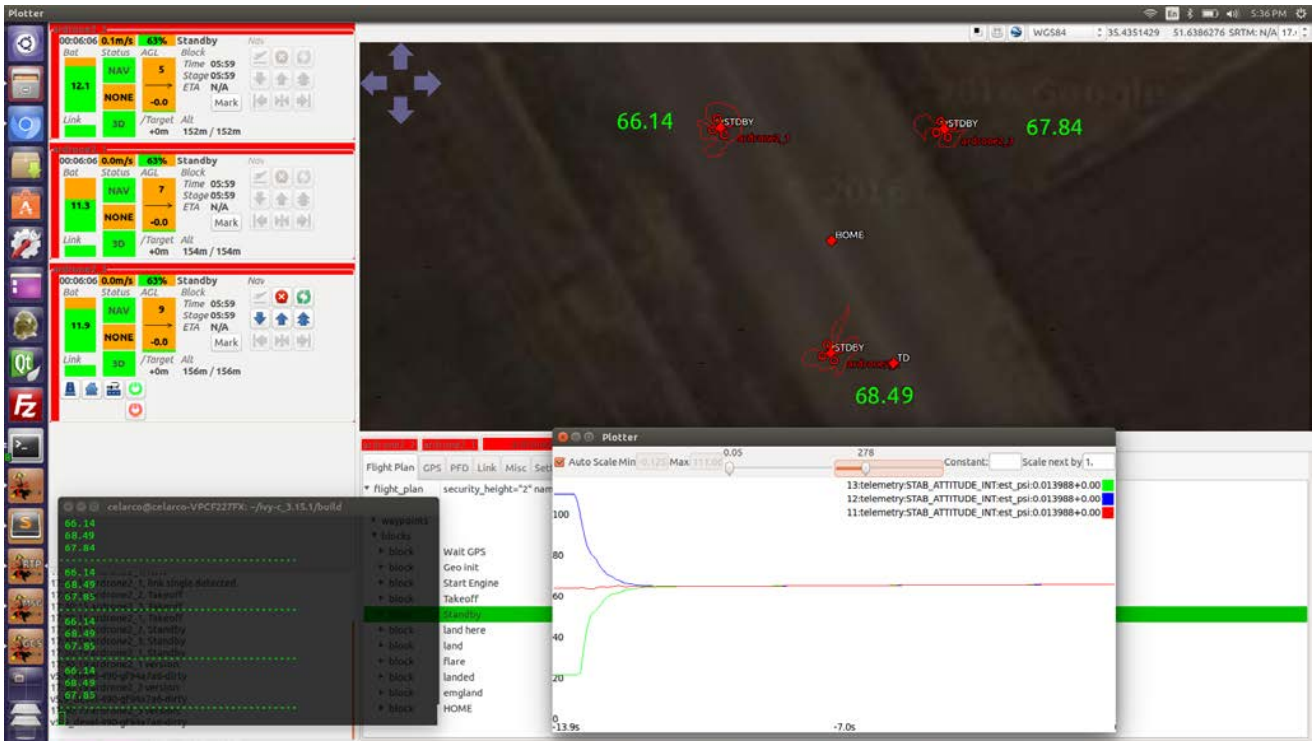


Figure 3. Final consensus reached heading values for 3 quadrotors along with their planar trajectories

http://www.imavs.org/

http://www.imavs.org/

References

1 Murray, R.M.: 'Recent Research in Cooperative Control of Multivehicle Systems'*J. Dyn. Syst. Meas. Control*, 2007, **129**, (5), p. 571.

2 Jadbabaie, A., Jie Lin, Morse, A.S.: 'Coordination of groups of mobile autonomous agents using nearest neighbor rules'*IEEE Trans. Automat. Contr.*, 2003, **48**, (6), pp. 988–1001.

3 Dunbar, W., Murray, R.: 'Distributed receding horizon control for multi-vehicle formation stabilization'*Automatica*, 2006.

4 Riehl, J., Collins, G.: 'Cooperative search by UAV teams: A model predictive approach using dynamic graphs'*IEEE Trans.*, 2011.

5 Gu, G., Chandler, P., Schumacher, C.: 'Optimal cooperative sensing using a team of UAVs'*Aerosp.* …, 2006.

6 Ogren, P., Fiorelli, E., Leonard, N.E.: 'Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environment'*IEEE Trans. Automat. Contr.*, 2004, **49**, (8), pp. 1292–1302.

7 Bhattacharya, R., Tiwari, A., Fung, J., Murray, R.M.: 'Cone invariance and rendezvous of multiple agents'*Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.*, 2009, **223**, (6), pp. 779–789.

8 Kim, Y., Mesbahi, M., Hadaegh, F.Y.: 'Multiple-Spacecraft Reconfiguration Through Collision Avoidance, Bouncing, and Stalemate'*J. Optim. Theory Appl.*, 2004, **122**, (2), pp. 323–343.

9 Cortes, J., Martinez, S., Karatas, T., Bullo, F.: 'Coverage control for mobile sensing networks', in 'Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)' (IEEE, no date), pp. 1327–1332

10 Klavins, E.: 'A Formal Model of a Multi-Robot Control and Communication Task'no date.

11 Olfati-Saber, R., Murray, R.M.: 'Consensus Problems in Networks of Agents With Switching Topology and Time-Delays'*IEEE Trans. Automat. Contr.*, 2004, **49**, (9), pp. 1520–1533.

12 Jiang, F., Wang, L., Xie, G.: 'Consensus of high-order dynamic multi-agent systems with switching topology and time-varying delays'*J. Control Theory Appl.*, 2010, **8**, (1), pp. 52–60.

13 Moreau, L.: 'Stability of multiagent systems with time-dependent communication links'*IEEE Trans. Automat. Contr.*, 2005, **50**, (2), pp. 169–182.

14 Li, T., Zhang, J.-F.: 'Mean square average-consensus under measurement noises and fixed topologies: Necessary and sufficient conditions' (2009)

15 Porfiri, M., Stilwell, D.J.: 'Consensus Seeking Over Random Weighted Directed Graphs'*IEEE Trans. Automat. Contr.*, 2007, **52**, (9), pp. 1767–1773.

16 Lewis, F.L., Zhang, H., Hengster-Movric, K., Das, A.: 'Cooperative Control of Multi-Agent Systems' (Springer London, 2014)

17 Li, S., Du, H., Lin, X.: 'Finite-time consensus algorithm for multi-agent systems with double-integrator dynamics' (2011)

18 Rezaee, H., Abdollahi, F.: 'Average Consensus Over High-Order Multiagent Systems'*IEEE Trans. Automat. Contr.*, 2015, **60**, (11), pp. 3047–3052.

19 Yongcan Cao, Y., Wei Ren, W.: 'Optimal Linear-Consensus Algorithms: An LQR Perspective'*IEEE Trans. Syst. Man, Cybern. Part B*, 2010, **40**, (3), pp. 819–830.

20 Ren, W.: 'Consensus Algorithms for Double-integrator Dynamics', in (Springer London, 2008), pp. 77–104

21 Ren, W., Moore, K.L., Chen, Y.: 'High-Order and Model Reference Consensus Algorithms in Cooperative Control of MultiVehicle Systems'*J. Dyn. Syst. Meas. Control*, 2007, **129**, (5), p. 678.

22 Mei, J., Ren, W., Ma, G.: 'Distributed containment control for Lagrangian networks with parametric uncertainties under a directed graph' (2012)

23 Bauso, D., Giarré, L., Pesenti, R.: 'Non-linear protocols for optimal distributed consensus in networks of dynamic agents'*Syst. Control Lett.*, 2006, **55**, (11), pp. 918–928.

24 Cortés, J.: 'Distributed algorithms for reaching consensus on general functions'*Automatica*, 2008, **44**, (3), pp. 726–737.

25 Xiao, L., Boyd, S.: 'Fast linear iterations for distributed averaging'*Syst. Control Lett.*, 2004, **53**, (1), pp. 65–78.

26 Delvenne, J.-C., Carli, R., Zampieri, S.: 'Optimal strategies in the average consensus problem'*Syst. Control Lett.*, 2009, **58**, (10), pp. 759–765.

27 Bauso, D., Giarre, L., Pesenti, R.: 'Distributed Consensus in Networks of Dynamic Agents', in 'Proceedings of the 44th IEEE Conference on Decision and Control' (IEEE, 2005), pp. 7054–7059

28 Semsar-Kazerooni, E., Khorasani, K.: 'An LMI approach to optimal consensus seeking in multi-agent systems', in '2009 American Control Conference' (IEEE, 2009), pp. 4519–4524

29 Movric, K.H., Lewis, F.L.: 'Cooperative Optimal Control for Multi-Agent Systems on Directed Graph Topologies'*IEEE Trans. Automat. Contr.*, 2014, **59**, (3), pp. 769–774.

30 Hongwei Zhang, H., Lewis, F.L., Das, A.: 'Optimal Design for Synchronization of Cooperative Systems: State Feedback, Observer and Output Feedback'*IEEE Trans. Automat. Contr.*, 2011, **56**, (8), pp. 1948–1952.

31 Liu, Y., Jia, Y.: 'Consensus problem of high-order multi-agent systems with external disturbances: An H∞ analysis approach'*Int. J. Robust Nonlinear Control*, 2010, **20**, (14), pp. 1579–1593.

32 He, W., Cao, J.: 'Consensus control for high-order multi-agent systems'*IET Control Theory Appl.*, 2011, **5**, (1), p. 231.

33 Xie, G., Wang, L.: 'Consensus control for a class of networks of dynamic agents'*Int. J. Robust Nonlinear Control*, 2007, **17**, (10-11), pp. 941–959.

34 Bernstein, D.S.: 'Nonquadratic cost and nonlinear feedback control'*Int. J. Robust Nonlinear Control*, 1993, **3**, (3), pp. 211–229.

35 Haddad, V.M., Chellabonia, V.: 'Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach' (Princeton University Press, 2008)

36 Mohamadifard, A., Radmanesh, A., & Naghash, A.: 'Optimal Linear Quadratic Consensus for High-Order Integral Dynamics: Quadrotors Formation Control Application' in ' Proceedings of the International Micro Air Vehicle Conference and Flight Competition' (IMAV, 2015),

37 Remes, B. D. W., Hensen, D., Van Tienen, F., De Wagter, C., Van der Horst, E., & De Croon, G. C. H. E.: 'Paparazzi: how to make a swarm of parrot ar drones fly autonomously based on gps' in ' Proceedings of the International Micro Air Vehicle Conference and Flight Competition' (IMAV, 2013),

38 De Croon, G. C. H. E., De Wagter, C., Remes, B. D. W., & Ruijsink, R.: 'Sky Segmentation Approach to obstacle avoidance' in 'Proceedings of the IEEE Aerospace Conference' (IEEE, 2011), pp. 1-16

http://www.imavs.org/

http://www.imavs.org/

Parameter selection method and performance assessment for the preliminary design of electrically powered transitioning VTOL UAVs

B. Theys* and J. De Schutter
KU Leuven, Leuven, Belgium

http://www.imavs.org/

ABSTRACT

This paper describes a parameter selection method and performance assessment for the preliminary design of Vertical Take-Off and Landing Unmanned Aerial Vehicles (VTOL UAVs) that use a combination of wings and a set of electrically powered propellers for providing lift and thrust during cruise flight. This method allows us to quickly evaluate the possibilities of current technology for a given set of user and mission-specific requirements, and to create a preliminary design to meet these requirements. To this end, the acceptable range for the variable design parameters is predetermined and the parameters to optimize are identified. Mass and power models are presented for the components of the UAV and a novel model for a propeller in oblique flow conditions is applied. The models are used in a design algorithm that calculates all combinations of components. All feasible solutions are selected and displayed to the user after which the optimal solution can be chosen. A design case is presented and a sensitivity analysis shows the influence of different design parameters on this case.

1 INTRODUCTION

Currently, UAVs are increasingly deployed in various applications such as surveillance, mapping, inspection or transportation. VTOL UAVs such as multi-rotors prove to be advantageous over conventional fixed-wing UAVs due to their ability to land and take-off vertically or hover. The disadvantage is their limited flight speed and range. Therefore, VTOL UAVs are being developed that can transition from hover to efficient forward flight. Figure 1 presents two of such UAVs. A broad overview of hybrid UAV designs, their modelling and used control strategies, is presented by Saeed et. al. [1].

With improved performance, new applications or missions become possible, if a UAV can be designed that meets the mission-specific requirements. In many cases the user has only vague or conflicting requirements, since it is unknown whether these requirements are feasible or not. Therefore, a tool is required that can guide the user in defining the requirements by providing an overview of what is possible with current technology. A preliminary design that meets the requirements of the user is selected and presented as a set of design parameters with the associated performance.



Figure 1: Left: VertiKUL 1, designed to make a transition of $90°$ with a large wing to provide all the lift. Right: VertiKUL 2, designed to make a transition of $45°$ with smaller wings to improve wind gust resistance.

The design of a UAV consists of various systems and components with strong interactions. Multiple disciplines including propulsion, aerodynamics and materials are combined. The challenge for such integrated design is to capture interactions between the various systems and components. In recent years, multiple integrated design environments have been developed by industrial, governmental, and academic research groups [2]. A good example of an integrated design environment is the Boeing Integrated Vehicle Development System (BIVDS) [3] for the development of various types of aircraft. For multi-rotor UAVs, the optimal selection of propulsion components is addressed in [4]. In this work, payload, number of rotors and flight duration is fixed. Other studies on automated design methods for multi-rotor UAVs are described by Bouabdallah [5] and Lundstrom [6]. Lundstrom goes a step further than preliminary design; the CAD files are subsequently evaluated with a panel code for detailed aerodynamic evaluation. Aksugur [7] presents a design methodology, specifically for tailsitter UAV concepts with a hybrid propulsion system based on analytical calculations and experimental data.

From a general engineering point of view, the design process of a new product is split into several phases [8]. The present paper focusses on the first two phases: feasibility

*Email address: bart.theys@kuleuven.be

http://www.imavs.org/

study and preliminary design. The goal of the parameter selection method in the presented paper is to have absolute freedom in specifying which of the design parameters are varied to come to an optimal solution and to have a flexible tool to evaluate the possibilities of current technology. Feasible preliminary designs are generated within a weight class and type or family of UAV that use a similar design approach. The incentive for the development of this parameter selection method originated from the design of a VTOL UAV for autonomous parcel delivery, presented in [9]. In this prior work, a more dedicated design method is presented for a case in which many of the design parameters had already been fixed [10], reducing the size of the parameter space.

Section 2 models the systems and components that are used for an electrically powered multi-rotor VTOL UAV. Next, section 3 describes the algorithm that evaluates all possible designs and presents the best solution, based on a set of user requirements. Finally, the design process is demonstrated for a specific case and the impact of technology on the performance of the design is discussed in section 4.

2 COMPONENT MODELLING

An electrically powered multi-rotor VTOL UAV consists of many components. Components that are not decided in advance can be modelled using three different methods. A statistical method models a component based on data from existing components. It therefore requires a dataset that is both dense enough to provide a high accuracy and large enough to cover a broad range of designs, since extrapolating these models can lead to erroneous results. An analytical or quasi-analytical method models components based on their underlying physical properties and behaviour. This method allows us to extrapolate and requires only little data of existing components to validate the model. However, this data needs to be representative for the design problem and the accuracy might be low. Finally, a design-based method is an accurate method based on component-specific parameters that determine directly the properties of the component. However, not many UAV components can be modelled with this method.

All three methods are used for the component modelling described in the present paper. This section provides general models for the different components that contribute to the mass and performance of the UAV: body, wing, propellers, motor, electronic speed controller (ESC) and battery. The accuracy of the design algorithm is only as accurate as the models of these components. Individual models can be easily updated if new technology becomes available.

2.1 Body and wing

The aerodynamic performance of the design consists of the drag of the UAV without wings, D_{body}, and the lift and drag of the wings, L_{wing} and D_{wing}, that can be added to the UAV. As it is common practice in helicopter performance identification [11], the drag of the UAV without lifting surfaces is modelled by an equivalent frontal surface A_{eq} with $C_D = 1$:

$$D_{body} = \frac{1}{2} V^2 \rho \, A_{eq}, \tag{1}$$

with V the speed and ρ the air density. The drag of the wing is assumed to have a parabolic relationship with the lift coefficient and is calculated as:

$$D_{wing} = \frac{1}{2} C_{D_{wing}} V^2 \rho \, A_{wing}. \tag{2}$$

With

$$C_{D_{wing}} = C_{D0} + \frac{C_L^2}{\pi \, AR \, e}. \tag{3}$$

Herein, C_{D0} represents the drag coefficient of the airfoil at zero-lift. The aspect ratio of the wing AR is defined as $AR = \frac{b^2}{A_{wing}}$ with b the span of the wing and A_{wing} the wing surface area. The span efficiency factor e can vary between 0.65 and 0.93 [12]. The lift coefficient of the finite wing, C_L, is calculated based on the zero-lift angle, α_0, and the lift slope of the finite wing a_{wing}, which is smaller than the lift slope of the airfoil, a_0.

$$C_L = a_{wing}(\alpha_{wing} - \alpha_0), \tag{4}$$

with the lift slope for a finite wing calculated as:

$$a_{wing} = \frac{a_0}{1 + \frac{a_0}{\pi \, AR \, e}}. \tag{5}$$

The mass of the body m_{body} depends on many factors such as the size, total mass m_{tot} and thrust-to-weight ratio of the UAV, the materials used and quality of the construction method. In this paper the mass of the body is modelled with a quasi-analytical method as a fraction k_{body} of the total mass:

$$m_{body} = k_{body} \, m_{tot}. \tag{6}$$

Since wings can be added to the body, their mass can be modelled separately. The mass of the wing consists of surface material that covers the wing, $m_{surface}$, core material to shape the wing, m_{core}, and a central beam with mass m_{beam} to increase stiffness and strength.

$$m_{wing} = m_{surface} + m_{core} + m_{beam} \tag{7}$$

Figure 2 illustrates these different components for a lightweight UAV wing. An accurate design-based method is used for the surface and core material:

$$m_{surface} = 2 \, k_{surface} \, A_{wing} \tag{8}$$

$$m_{core} = k_{core} \, V_{wing} \tag{9}$$

with V_{wing} the volume of the wing, $k_{surface}$ the area density of the surface material and k_{core} the volumetric mass density of the core material.

The mass of the central beam is modelled with an analytical method that is based on the use of a hollow, thin-walled

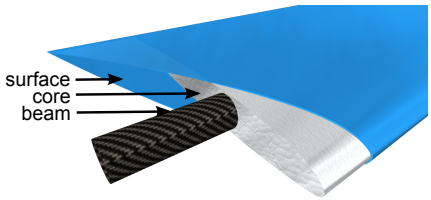


Figure 2: Lightweight construction of a UAV wing.

cylindrical tube. A maximum allowed deflection of the tip of the beam in the lateral direction, x, relative to the span of the wing is imposed to guarantee the stiffness of the wing $x_{rel} = \frac{x}{b/2}$. The relative deflection of the beam with one end clamped and the load $q = \frac{m_{tot}\, G\, g}{b}$ assumed to be equally distributed, is calculated as:

$$x_{rel} = \frac{q\,(b/2)^3}{8\,I\,E} \tag{10}$$

Herein is G the load factor and E the Young's modulus of the material of the beam. The area moment of inertia is calculated as $I = \pi\,(d/2)^3\,t_{beam}$, with d the thickness of the wing thus maximum diameter of the beam and t_{beam} the wall thickness of the beam. The thickness of the wing is calculated as $d = c\,t_{af}$ with c the chord of the wing and t_{af} the maximum relative thickness of the selected airfoil. We can now calculate the required thickness of the beam as:

$$t_{beam} = \frac{m_{tot}\,G\,g\,b^2}{8\,\pi\,E\,c^3\,t_{af}^3\,x_{rel}} \tag{11}$$

The mass of the beam m_{beam} is calculated as:

$$m_{beam} = \frac{m_{tot}\,G\,g\,b^3}{8\,E\,c^2\,t_{af}^2\,x_{rel}} = k_{beam}\,\frac{m_{tot}\,G\,g\,b^3}{c^2}. \tag{12}$$

The constant k_{beam} can be experimentally determined or calculated as

$$k_{beam} = (8\,E\,t_{af}^2\,x_{rel})^{-1} \tag{13}$$

Comparable models are presented by Noth [13], Tennekes [14] and Rizzo & Frediani [15].

2.2 Propellers

The propeller model presented in this paper goes one step further than most propeller models. Usually for multi-rotor UAVs, the propeller is modelled in near-hover conditions. For a transitioning UAV, however, the propeller will travel through the air at high speeds and angles α_{plr} ranging from 0° to 90° [16] as illustrated on figure 3 with F_{plr} the force vector produced by the propeller and α_{plr} defined as the angle of the propeller with respect to the free-stream airspeed. For a propeller with a specified diameter, momentum theory provides an expression for the consumed 'ideal' power. Glauert [17] generalized the axial momentum theory from Rankine - Froude [18] for a propeller operating in oblique flow:

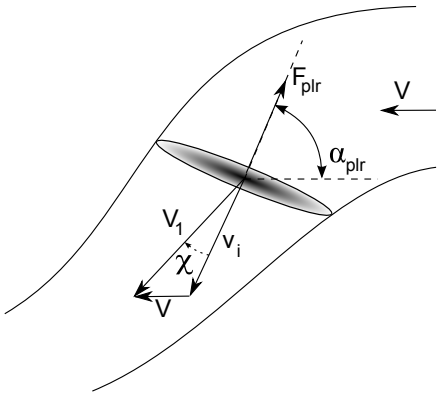


Figure 3: A propeller in oblique flow.

$$V_1 = \sqrt{(V\sin\alpha_{plr})^2 + (V\cos\alpha_{plr} + v_i)^2} \tag{14}$$

$$F_{plr} = 2\rho A_{plr} v_i V_1 \tag{15}$$

$$P_i = F_{plr}(v_i + V\cos\alpha_{plr}) \tag{16}$$

$$P_{mech} = \frac{P_i}{\eta_{plr}} \tag{17}$$

Herein is V_1 the airspeed vector at the propeller, v_i the propeller induced velocity and A_{plr} the propeller disk area. Equations (14) to (17) allow to calculate the required mechanical power P_{mech} if the propeller efficiency η_{plr} is known. From wind tunnel tests of a propeller in oblique flow conditions [16], it was found that the propeller efficiency decreases for an increasing skewing angle χ, that is calculated as:

$$\chi = \tan^{-1}\left(\frac{V\sin(\alpha_{plr})}{(v_i + V\sin(\alpha_{plr}))}\right) \tag{18}$$

A correlation between the propeller efficiency in hover $\eta_{plr_{hover}}$ and skewing angle was found [19]:

$$\eta_{plr} = \frac{\eta_{plr_{hover}}}{4.1\cdot 10^{-6}\chi^3 - 0.00028\chi^2 + 0.006\chi + 1} \tag{19}$$

With χ expressed in degrees. This correlation takes into account the loss in efficiency due to the off-design working regime of the propeller.

The mass of the propeller m_{plr} is modelled as recommended by the 'General Dynamics propeller weight method' described by Roskam [20] for plastic or composite propellers:

$$m_{plr} = k_{plr}\,n_{blades}^{k_{blades}}\,(D_{plr}\,P_{mech_{max}})^{e_{plr}} \tag{20}$$

With n_{blades} the number of blades, D_{plr} the diameter of the propeller, $P_{mech_{max}}$ the maximum mechanical power that the motor delivers to the propeller and k_{plr}, k_{blades} and e_{plr} model-specific values.

http://www.imavs.org/

http://www.imavs.org/

2.3 *Battery - motor - ESC*

In this paper, the battery is characterized solely by its mass m_{bat} and energy E_{bat}. Although the volume of the battery can also be a limiting factor in some designs, for UAVs usually the mass is more important. For the study in this paper we therefore propose following design-based model [21]:

$$E_{bat} = k_{bat}\, m_{bat}\, f_{usable} \tag{21}$$

Herein is f_{usable} (< 1) the fraction of the total energy of the battery that will be consumed each cycle in order to extend lifetime of the battery. Many chemistries of battery are possible to use for electrically powered UAVs. Therefore, the battery mass constant k_{bat} has to be adapted for the selected type of battery.

The motor is modelled with a statistical method as a component with a mass m_{mot} and a maximum rated power, $P_{mot_{max}}$.

$$m_{mot} = k_{mot} P_{mot_{max}}, \tag{22}$$

with the motor mass constant k_{mot} determined in a statistical way from a large data set of existing motors [21]. The efficiency of the conversion to mechanical energy from electrical energy of the motor is determined by:

$$P_{mech} = \eta_{mot}\, P_{ele} \tag{23}$$

The electronic speed controller (ESC) is modelled similarly as the motor with k_{ESC} the mass constant and η_{ESC} the efficiency.

3 DESIGN ALGORITHM

3.1 *Parameter classification*

A preliminary design is defined by a large set of parameters that have mutual interactions and cannot all be changed independently of each other. A limited set with the most important parameters is selected to be used in the design algorithm to keep the calculation time as low as possible. As input parameters for one design, a minimum set of parameters that define one design is selected so that remaining output parameters can be determined by straightforward calculations that require only a very small computational time. The input parameters are divided into fixed parameters and iteration parameters. The fixed parameters, for example gravity g or mass of the avionics m_{av}, are kept constant throughout one run of the algorithm. The iteration parameters are the parameters for which the user does not yet know what value to select in order to find a design that meets the requirements such as for example total mass m_{tot}, battery mass m_{bat}, propeller diameter D_{plr}, ... Output parameters are calculated based on the component models and their mutual relations. The user applies the desired boundaries to these parameters, such as minimum required speed, maximum allowed span, minimum required range, ... Figure 4 presents a schematic overview of the classification of the set of parameters. In this figure, the parameters in the shaded region of the input parameter space are iterated and on the parameters in the shaded region of the output parameter space, boundaries are applied. Input parameters outside the shaded region are decided in advance or cannot be changed by the user. Output parameters outside the shaded region are parameters that are calculated, but that are not of any interest to the user.

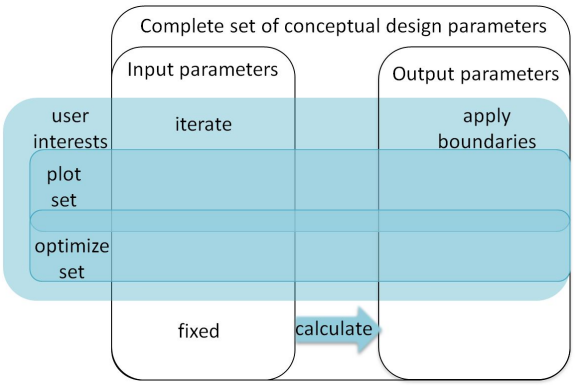


Figure 4: Schematic presentation of the parameter classification.

Table 1 presents the set of input parameters with their valid range that are used to calculate remaining output parameters.

3.2 *Calculation method*

Based on the selected set of input parameters, the remaining output parameters are calculated. First the aerodynamic properties of the UAV are calculated following equation (1) to (3). Next, the cruise flight speed is calculated. Based on the assumption that the UAV flies at a constant speed and altitude with the geometry determined by the input parameters, only one flight speed results in a force equilibrium, illustrated in figure 5:

$$F_{prop} = F_{plr} n_{plr} \tag{24}$$

Herein is n_{prop} the number of propellers.

$$L_{prop} = F_{prop} \sin \alpha_{prop} \tag{25}$$

$$T_{prop} = F_{prop} \cos \alpha_{plr} \tag{26}$$

$$T_{prop} = D = \frac{1}{2} \rho V_{cruise}^2 (A_{eq} + A_{wing} C_{D_{wing}}) \tag{27}$$

$$L_{prop} = T_{prop} \tan \alpha_{plr} \tag{28}$$

$$L_{wing} = \frac{1}{2} \rho V_{cruise}^2 A_{wing} C_{L_{wing}} \tag{29}$$

$$L_{prop} + L_{wing} = m_{tot}\, g \tag{30}$$

$$V_{cruise}^2 = \frac{2\, m_{tot}\, g}{\rho \Big(\tan \alpha_{plr} (A_{eq} + A_{wing} C_{D_{wing}}) + A_{wing} C_{L_{wing}} \Big)} \tag{31}$$

http://www.imavs.org/

Symbol	explanation	value	unit
D_{plr}	propeller diameter	$0.1 - 0.6$	$[m]$
$C_{L_{wing}}$	wing lift coefficient	$0.1 - 1.2$	$[-]$
m_{tot}	total mass	$1 - 10$	$[kg]$
α_{plr}	propeller angle	$0 - 90$	$[°]$
n_{prop}	number of propellers	$4 - 8$	$[-]$
A_{wing}	wing surface	$0 - 1$	$[m^2]$
mf_{bat}	battery fraction	$0.1 - 0.8$	$[-]$
dim_{dia}	max. dimension	$0.4 - 2$	$[m]$
m_{av}	avionics mass	0.12	$[kg]$
P_{av}	avionics power	1	$[W]$
P_{pld}	payload power	0	$[W]$
$t_{rel_{hover}}$	rel. time in hover	0.05	$[-]$
TW_{ratio}	thrust to weight	1.5	$[-]$
ρ	air density	1.22	$[kg/m^3]$
g	gravitation	9.81	$[m/s^2]$
k_{bat}	bat. energy density	$120 - 360$	$[Wh/kg]$
η_{esc}	ESC efficiency	0.95	$[-]$
η_{mot}	motor efficiency	0.75	$[-]$
k_{esc}	ESC mass cnst.	$1/10000$	$[kg/W]$
k_{mot}	motor mass cnst.	$1/3400$	$[kg/W]$
n_{blades}	number of plr blades	2	$[-]$
k_{plr}	propeller mass cnst.	$4.4 \cdot 10^{-4}$	$[-]$
k_{blades}	plr blade mass cnst.	0.391	$[-]$
e_{plr}	plr mass model exp.	0.782	$[-]$
$\eta_{plr_{hover}}$	plr hover efficiency	0.7	$[m]$
k_{drag}	drag cnst.	$0.1 - 0.25$	$[-]$
AR	Aspect Ratio	$1 - 10$	$[-]$
e	Oswald factor	0.85	$[-]$
C_{d0}	Zero-lift drag	0.01	$[-]$
a_0	airfoil lift slope	2π	$[-/rad]$
k_{body}	body structure quality	$0.1 - 0.2$	$[-]$
k_{beam}	wing structure quality	0.014	$[m^2/s^2]$
$k_{surface}$	surface mass cnst.	$0.05 - 0.35$	$[kg/m^2]$
k_{core}	wing core mass cnst.	$10 - 40$	$[kg/m^3]$
f_{usable}	usable bat. capacity	0.8	$[-]$

Table 1: Used set of preliminary design input parameters.

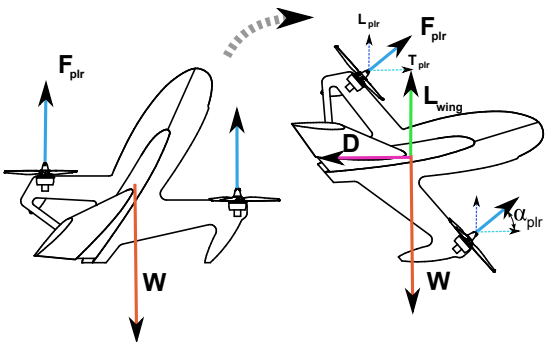


Figure 5: Overview of the forces acting upon a transitioning UAV in hover and in cruise flight.

With the speed known, the required mechanical power for the propulsion system in cruise flight is calculated with equations (14) to (17). Next, the mechanical power is also calculated for hover and maximum throttle regime and the electrical power for one propulsion system P_{ele} is calculated with equation (23). The mass of the propulsion system results from the maximum required power and is calculated with the models, presented in section 2. The total electrical power consumption P_{tot} is found by:

$$P_{tot} = P_{prop} + P_{av} + P_{pld} \tag{32}$$

In which the power of the avionics P_{av} and the power of the payload P_{pld} is known. The power of the propulsion system P_{prop} is calculated as:

$$P_{prop} = n_{prop}\, P_{ele}. \tag{33}$$

The mass of the structure and wings is calculated with equations (6) to (12) and the remaining mass available for payload is found by:

$$m_{pld} = m_{tot} - m_{av} - m_{body} - m_{prop} - m_{wing} - m_{bat} \tag{34}$$

Note that this formula can result in a negative payload mass for a set of input parameters; these solutions are filtered out by the boundary condition of a minimum payload mass that is above $0\ kg$. The total flight time is:

$$t_{flight_{total}} = \frac{E_{bat}}{((1 - t_{rel_{hover}})P_{tot_{cruise}} + t_{rel_{hover}} P_{tot_{hover}}}. \tag{35}$$

Herein is $t_{rel_{hover}}$ the relative amount of mission time spend in the hover phase and $P_{tot_{cruise}}$ and $P_{tot_{hover}}$ are calculated with equation (33) for cruise and hover phase respectively. The flight time in cruise is

$$t_{flight_{cruise}} = t_{flight_{total}}(1 - t_{rel_{hover}}). \tag{36}$$

Therefore, the range of the UAV is

$$d = t_{flight_{cruise}}(V_{cruise} - V_{wind}), \tag{37}$$

with V_{wind} the headwind during flight and V_{cruise} as in equation (31).

3.3 *Algorithm work flow*

The whole parameter space that is defined by the iteration input parameters, is evaluated in order to find the global optimum that is specified by the mission requirements. For each combination of the iteration input parameters in combination with the fixed input parameters, the remaining parameters of the preliminary design are calculated. Afterwards, the designs are filtered to only keep solutions that lie within the parameter space, defined by boundaries set by the user. The parameters of interest from the set of solutions are then plotted on 2D or 3D graphs to give the user an overview of the

trade-offs between these parameters. This is useful in case the user wants to optimize more than one parameter like for instance both speed and range. Instead of choosing the design that meets the user requirements and performs best in terms of parameters to optimize, weights can be applied to the set of parameters to optimize and the best solution can be computed from the valid solution set. This best solution, based on weights applied to parameters to optimize, is highlighted in the plots that present all solutions. The design algorithm is schematically presented in figure 6.

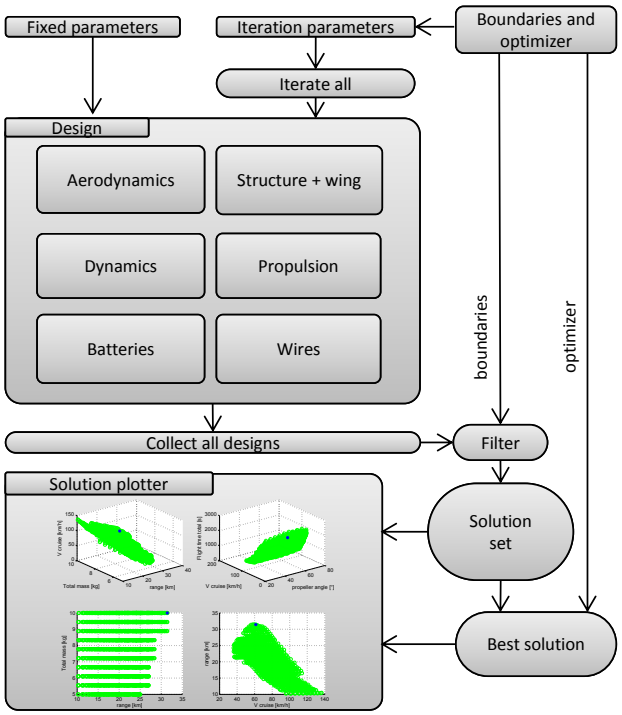


Figure 6: Overview of the design algorithm.

The set of design parameters is stored in a spreadsheet and forms the basis for a parametric CAD model.

4 DESIGN CASE

As a demonstration of the algorithm, this section presents a case for which a UAV has to transport a package of $1kg$ as far and as fast as possible. The total mass should not exceed $5kg$ and the span should not exceed $0.8m$. The input parameters are chosen according to table 1, except for those specified in particular for this mission and presented in table 2.

The algorithm now calculates the performance for a set of 15000 combinations of input parameters. On figure 7, all dots present possible solutions. The best solutions in terms of speed and range lie on the right top edge of the solution cloud on the right graph. From this graph, the user selects the preferred performance. The user can observe that there is no point in flying slower than $14m/s$, since both range and speed will decrease. Above this speed, the user needs to make a trade-off between speed and range. In this case we select a UAV design with a range of $33.4km$ and a cruise speed of $16.3m/s$, marked with the star on the graphs. Figure 8 shows the mass distribution of the selected design with a total mass of $5kg$. Battery and payload represent almost two third of the total mass.

Parameter	type	value	unit
D_{plr}	iterate 3	$0.229 - 0.279$	$[m]$
$C_{L_{wing}}$	iterate 5	$0.1 - 1.2$	$[-]$
A_{wing}	iterate 5	$0 - 0.2$	$[m^2]$
mf_{bat}	iterate 10	$0.3 - 0.6$	$[-]$
α_{plr}	iterate 20	$0 - 80$	$[^\circ]$
m_{tot}	fixed	5	$[kg]$
n_{prop}	fixed	4	$[-]$
dim_{dia}	fixed	0.8	$[m]$
k_{bat}	fixed	220	$[Wh/kg]$
k_{drag}	fixed	0.1	$[-]$
AR	fixed	2	$[-]$
k_{body}	fixed	0.15	$[-]$
$k_{surface}$	fixed	0.05	$[kg/m^2]$
k_{core}	fixed	10	$[kg/m^3]$
f_{usable}	fixed	0.8	$[-]$
$t_{flight_{tot}}$	boundary	$0 - \infty$	$[s]$
m_{pld}	boundary	$1 - \infty$	$[kg]$
V_{cruise}	boundary & optimize	$0 - \infty$	$[m/s]$
$range$	boundary & optimize	$0 - \infty$	$[m]$

Table 2: Mission specific parameters for the design case.

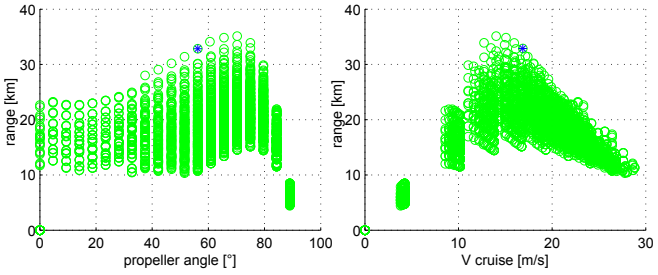


Figure 7: Set of 2895 feasible solutions resulting from one run of the algorithm, with the chosen solution highlighted.

4.1 *Off-Design performance*

Now that the preliminary design is fixed, the performance of the UAV in other flight regimes can be evaluated. To this end, the transition angle is varied from hover up to the point for which the propulsion system delivers its maximum rated power; this is the top speed. Lift and drag of the UAV are calculated for each transition angle with equations (2) and (4) and the speed and associated power are calculated according to equations (31) and (32). The simulated performance of the

http://www.imavs.org/

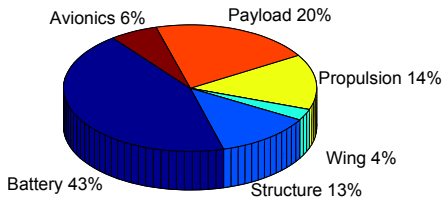


Figure 8: Mass distribution of the selected design.

selected design is plotted in figure 9. The graphs in this figure show that the required power drops significantly from hover up to $10m/s$, which is due to two reasons. The first reason is the more energy efficient working regime of the propellers due to the speed of the incoming air stream. The second reason is the beneficial effect of the added wing, that becomes more pronounced at higher speeds and angles of attack that approach the design point. The highest range is achieved at about $13.5m/s$, this is lower than the selected design speed of $16.3m/s$, since a trade-off was made between speed and range. The longest flight time is achieved at an even slower speed of only $10m/s$, the point at which power consumption is the lowest.

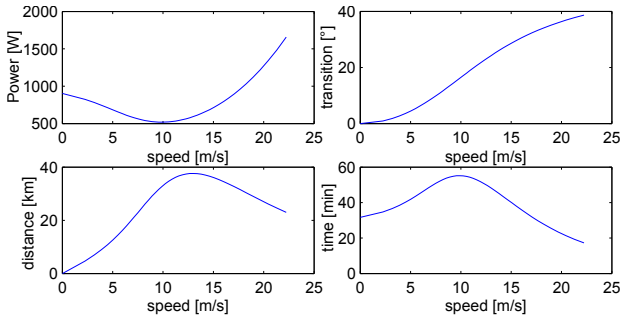


Figure 9: Simulated performance of the chosen design from hover to maximum speed.

4.2 *Parametric sensitivity analysis*

On the one hand, the parameter selection method allows to quickly evaluate the impact of technology improvements on the performance of electrically powered multi-rotor VTOL UAVs. An improvement in technology translates into an updated model for the different components, described in section 2. On the other hand, the parameter selection method is flexible in a way that the user can choose to broaden or narrow the parameter space, to evaluate the influence of these boundaries on the best performing design. As an example, the impact of technology and the influence of the user selected boundaries on the performance of the selected design in terms of speed and range are graphically presented in figure 10 for a select set of design parameters. The bars on this chart present the relative impact of a 10% change of these parameters on respectively the possible speed and range of the presented design.

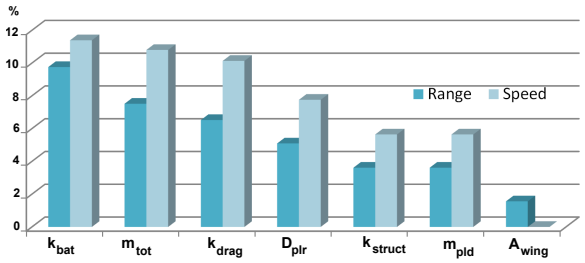


Figure 10: Relative influence of different design parameters on the best performing solution in terms of range and speed.

This analysis learns that for the design case in this paper, the energy density of the battery has the largest impact on the range and speed of the design. It also becomes clear that increasing the allowed total mass m_{tot} of $5kg$ or increasing the maximum allowed propeller diameter, has a large impact on the performance of the design. After the battery energy density and the total mass, decreasing the drag of the body of the UAV is most important in achieving higher range or speed. Increasing the allowed wing surface for this design did not have any influence on increasing the speed, but did have a small influence on increasing the range. It should be noted that this sensitivity analysis is only valid for the design case presented in this paper.

5 CONCLUSION

This paper presented a parameter selection method for the preliminary design and performance assessment of electrically powered multi-rotor VTOL UAVs. The method makes use of simplified models for the different components of the UAV. A novel propeller model is introduced, which takes into account the beneficial effect of an incoming air velocity at oblique flow conditions and efficiency losses due to the off-design working conditions. The performance of all possible design combinations of a user defined input parameter space is calculated and evaluated. The advantages of this approach are that the global optimum of the input set is always found and that the tool is flexible in a way that the user can choose the range of the parameters, the iteration resolution and the parameters to optimize. The disadvantage is the vast increase in computing time when the input parameter space is enlarged by increasing the resolution or increasing the range of the parameters. A solution to reduce computing time, is evaluating a broad range of input parameters with a low resolution and zooming in around the best solution during a second run of the algorithm to increase the resolution. The design algorithm provides a tool to evaluate the possibilities with the current technology and serves as initial sizing and component selection for all types of missions for VTOL UAVs. To improve the accuracy of the parameter selection method, more detail can be added to the individual models or more statistical models

http://www.imavs.org/

can be used. Therefore, a large and detailed database of different components is required. Flight tests on existing UAVs and prototypes can be used to validate the method and refine the models for the aerodynamics of wing, body and propeller, since these can only be measured in flight or in wind tunnel experiments. After creating a preliminary design with this method, a parametric CAD model can be generated and high-fidelity modelling such as finite element modelling can be used for the detailed design of the UAV.

REFERENCES

[1] Adnan S Saeed, Ahmad Bani Younes, Shafiqul Islam, Jorge Dias, Lakmal Seneviratne, and Guowei Cai. A review on the platform design, dynamic modeling and control of hybrid uavs. In *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, pages 806–815. IEEE, 2015.

[2] Kevin G Bowcutt. A perspective on the future of aerospace vehicle design. *AIAA Paper*, 6957:2003, 2003.

[3] Joel E Hirsh, Joseph B Wilkerson, and Robert P Narducci. An integrated approach to rotorcraft conceptual design. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, pages 8–11, 2007.

[4] Christos Ampatis and Evangelos Papadopoulos. Parametric design and optimization of multi-rotor aerial vehicles. In *Applications of Mathematics and Informatics in Science and Engineering*, pages 1–25. Springer, 2014.

[5] S. Bouabdallah. *Design and control of quadrotors with application to autonomous flying*. Ph.D. dissertation, 2007.

[6] David Lundström, Kristian Amadori, and Petter Krus. Automation of design and prototyping of micro aerial vehicle. In *47th AIAA Aerospace Sciences Meeting and Exhibit, Orlando, FL, USA*, 2009.

[7] Mirac Aksugur and Gokhan Inalhan. Design methodology of a hybrid propulsion driven electric powered miniature tailsitter unmanned aerial vehicle. *Journal of Intelligent and Robotic Systems*, 57(1-4):505–529, 2010.

[8] Morris Asimow. *Introduction to design*, volume 394. Prentice-Hall Englewood Cliffs, NJ, 1962.

[9] Menno Hochstenbach, Cyriel Notteboom, Bart Theys, and Joris De Schutter. Design and control of an unmanned aerial vehicle for autonomous parcel delivery with transition from vertical take-off to forward flight-vertikul, a quadcopter tailsitter. *International Journal of Micro Air Vehicles*, 7(4):395–406, 2015.

[10] Menno Hochstenbach and Cyriel Notteboom. *Ontwerp en bouw van een onbemand vliegtuig voor autonoom pakkettransport met gecontroleerde transitie van verticaal opstijgen naar voorwaartse vlucht*. PhD thesis, Master Thesis, KU Leuven, 2014.

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

[12] Jan Roskam and Chuan-Tau Edward Lan. *Airplane aerodynamics and performance*. DARcorporation, 1997.

[13] Andre Noth, Roland Siegwart, and Walter Engel. *Design of solar powered airplanes for continuous flight*. PhD thesis, ETH, 2008.

[14] Hendrik Tennekes. *The simple science of flight: from insects to jumbo jets*. MIT press, 2009.

[15] E Rizzo and A Frediani. A model for solar powered aircraft preliminary design. *Aeronautical Journal*, 112(1128):57–78, 2008.

[16] Bart Theys, Gregorius Dimitriadis, Thomas Andrianne, Patrick Hendrick, and Joris De Schutter. Wind tunnel testing of a vtol mav propeller in tilted operating mode. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 1064–1072. IEEE, 2014.

[17] Hermann Glauert. Airplane propellers. In *Aerodynamic theory*, pages 169–360. Springer, 1935.

[18] W Froude. On the elementary relation between pitch, slip, and propulsive efficiency. 1920.

[19] B. Theys, G. Dimitriadis, P. Hendrick, and J. De Schutter. Experimental and numerical study of mini-uav propeller performance in oblique flow. *To appear in AIAA Journal of Aircraft*, 2016.

[20] J Roskam. Airplane design, part v: component weight estimation, 1985. *Roskam Aviation Corporation, Ottawa, Kansas*, page 85.

[21] Jay Gundlach. *Designing unmanned aircraft systems: A comprehensive approach*. American Institute of Aeronautics and Astronautics, 2012.

http://www.imavs.org/

Precision Position Control of the DelFly II Flapping-wing Micro Air Vehicle in a Wind-tunnel

T. Cunis*, M. Karásek, and G.C.H.E. de Croon
Delft University of Technology, Kluyverweg 1, Delft, The Netherlands

http://www.imavs.org/

ABSTRACT

Flapping-wing MAVs represent an attractive alternative to conventional designs with rotary wings, since they promise a much higher efficiency in forward flight. However, further insight into the flapping-wing aerodynamics is still needed to get closer to the flight performance observed in natural fliers. Here we present the first step necessary to perform a flow visualization study of the air around the flapping wings of a DelFly II MAV *in-flight*: a precision position control of flight in a wind-tunnel. We propose a hierarchical control scheme implemented in the open-source Paparazzi UAV autopilot software. Using a decoupling, combined feed-forward and feed-back control approach as a core, we were able to achieve a precision of $\pm 2.5\,\mathrm{cm}$ for several seconds, which is much beyond the requirements for a time resolved stereo PIV technique.

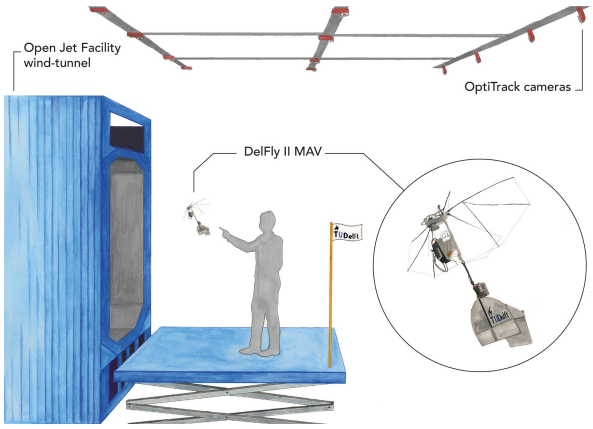


Figure 1: The DelFly II preforming a precision flight in the wind-tunnel. The achieved precision of $\pm 2.5\,\mathrm{cm}$ will allow unprecedented *in-flight* aerodynamic measurements in future. *Commissioned illustration by Sarah Gluschitz, licensed under CC BY-ND.*

1 INTRODUCTION

The recent boom of Micro Air Vehicles (MAVs) has induced a deep interest in flapping flight. At the scales of insects and small birds, flapping-wing MAVs are expected to challenge conventional rotorcraft designs by combining precision and agility at hover with energy efficient cruising flight. However, their development is still at the beginning and even the performance of state-of-the-art designs [1–3] remains far behind their biological counterparts. One of the reasons of this gap is our limited understanding of the complex and unsteady aerodynamic mechanisms occurring during flapping.

Numerical modelling of flapping wings can provide some insight; nevertheless, it remains a challenge on its own. Despite an intense research in recent years [4, 5], the existing models still struggle to match empirical data consistently. Thus, experimental techniques remain the most suitable tool for studying flapping flight.

So far, most experiments have been carried out in a tethered configuration where the aerodynamic forces and moments can be measured directly, in natural fliers [6] as well as in MAVs [7, 8]. However, it remains debatable whether these measurements are representative of free-flight conditions, especially in animals that behave abnormally when their body motion is restrained. A possible contamination of tethered measurements by structural vibrations was also reported in MAVs when compared to free flight estimates [9].

Another experimental tool used for studying flapping flight are flow visualization techniques. Particle Image Velocimetry (PIV) was used to capture the flow patterns around and behind man-made flapping wings in a tethered configuration [10–13] and also in various natural fliers flying in a wind-tunnel [14–16]. Nevertheless, such measurements always include a random factor due to unpredictable behaviour of even a well trained animal. Free flight PIV measurements with man made flappers have not yet been tried, but could bring much more consistent data as their manoeuvres can be pre-programmed.

Here, we present a first necessary step towards a free-flight PIV measurement on a flapping-wing MAV: a precision autonomous flight in a wind-tunnel (Fig. 1), something that has not yet been tried in flapping-wing MAVs apart from the DelFly platform itself [17]. Even for other platforms (fixed-wings, multi-rotors), only a few examples of autonomous wind-tunnel flying can be found in the literature [18, 19].

So far, most research on flapping flight control focused on tail-less designs that require active attitude stabilization to

*T. Cunis was MSc candidate at RWTH Aachen University. Authors' email addresses: torbjoern.cunis@wtnet.de; m.karasek@tudelft.nl; g.c.h.e.decroon@tudelft.nl

http://www.imavs.org/

stay in the air, e.g. [20–22]; this is not needed in tailed designs, like the DelFly, that benefit from passive stability. The higher level (guidance) control has deserved much less attention, e.g. in GPS-based flying [23]; however, very good trajectory tracking performance was achieved with a tethered, but unconstrained, insect-sized tail-less RoboBee [24]. Despite the lack of an accurate robot model, a Root Mean Square (RMS) error of $1\,\mathrm{cm}$ ($\approx 30\,\%$ of its wingspan) was achieved in hovering and near-hover trajectory tracking flights lasting several seconds with a single loop adaptive flight control. Including the Iterative Learning Control algorithm, which learns the robot dynamics uncaptured by the model, allowed to track also more aggressive manoeuvres [25].

Nevertheless, the DelFly differs significantly from the RoboBee: it is a tailed, passively stable platform designed primarily for (slow) forward flight. It has no (direct) roll control authority and a significant coupling exists between the pitch angle and vertical dynamics due to gliding effects. Also, no aggressive manoeuvres are needed for the intended tests. Thus, we adopted a rather classical approach with a cascaded structure, enhanced with a speed-thrust control block to deal with the longitudinal coupling effects and an adaptation phase to compensate for the model errors. Compared to the previous tests [17], we further improved the position sensing by using a commercial external motion tracking system.

In this paper, we first introduce the experimental setup in Section 2. The main contribution of this work is the control approach described in Section 3. The test results are presented and discussed in Section 4, which is followed by Section 5 with the conclusion.

2 EXPERIMENTAL SETUP

2.1 DelFly II MAV

The flapping-wing MAV used in the experiments described here is the DelFly II (Fig. 2, further called the DelFly), which is a well studied platform that has been developed by the TU Delft for over ten years [3, 26]. The DelFly has two flexible wings flapping in cross configuration; its characteristic wing motion with lift enhancing *clap-and-peel* mechanism is displayed in Fig 3.

The MAV is stabilized and controlled by a conventional tail with horizontal and vertical stabilizers and two actuated control surfaces; the elevator deflection results in a pure pitch moment, the rudder generates a combined roll and yaw moment.

The DelFly is equipped with a Lisa/S autopilot board [27], which includes a 6 DOF MEMS IMU for attitude estimation and an MCU running the Paparazzi UAV software[1]. Further on-board components include a DelTang Rx31 receiver and an ESP8266 Esp-09 WiFi module for data uplink and telemetry downlink. Four active, infra-red LED markers are placed at the nose, fin, at the tailplane (2x), forming a rotational asymmetric tetrahedron; thus, attitude and position of its geometrical centre can be uniquely recognized by the tracking system. In the current implementation, the controller uses the position and heading information from the tracking system and estimates roll and pitch angle from the on-board IMU.

[1] https://wiki.paparazziuav.org

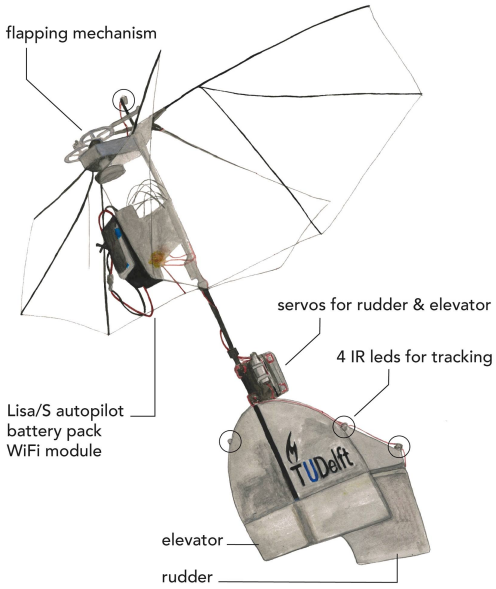
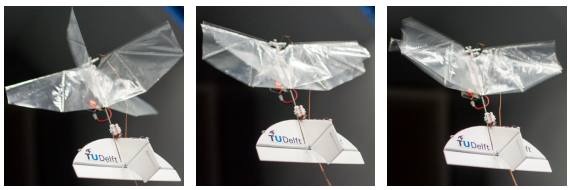


Figure 2: The DelFly II flapping-wing MAV used in this study. *Commissioned illustration by Sarah Gluschitz, licensed under CC BY-ND.*



(a) (b) (c)

Figure 3: Three important phases of the DelFly MAV flapping cycle: (a) wings fully opened, (b) wings clapping, (c) wings peeling apart.

2.2 Wind-tunnel set-up

The tests presented here were carried out in the *Open Jet Facility* of TU Delft. This close-circuit low-speed wind-tunnel offers an open test section of nearly $3\,\mathrm{m} \times 3\,\mathrm{m}$ and is thus greatly suitable for MAV free-flight experiments. The wind speed, controlled by the wind-tunnel software via the fan rpm, can reach up to $35\,\mathrm{m/s}$. We have noticed that the wind speed control was not very accurate at very low speeds, where our tests were performed (0.5 to $1.5\,\mathrm{m/s}$), resulting in a steady state error. Thus, in the results we will state the wind

speed set-point V_W^{SP} rather than the (unknown) actual speed V_W.

The DelFly has been tracked by twelve motion tracking cameras (OptiTrack Flex 13, NaturalPoint, Inc.) attached at the ceiling of the wind-tunnel room (Fig. 1), providing the MAV position and orientation at a rate of up to 120 Hz with a mean error of $\approx 0.2\,\mathrm{mm}$ per marker. The position and heading were sent on-board at a rate of 30 Hz, via the Paparazzi Ground Control Station, using the WiFi datalink.

2.3 Axis systems

We are using the axis systems defined by ISO 1151-1 and -2, see Fig. 4. In addition to the usual geographic, flight-path, air-path, and body-fixed axis systems, we introduce the *wind-tunnel reference system* in order to define the (desired) motion in the wind-tunnel: the x_w-axis points opposite the wind velocity vector, z_w points down and y_w completes the right-handed Cartesian system. As the origin is located at the geographic system's origin, the wind-tunnel reference system differs from the geographic only by rotation.

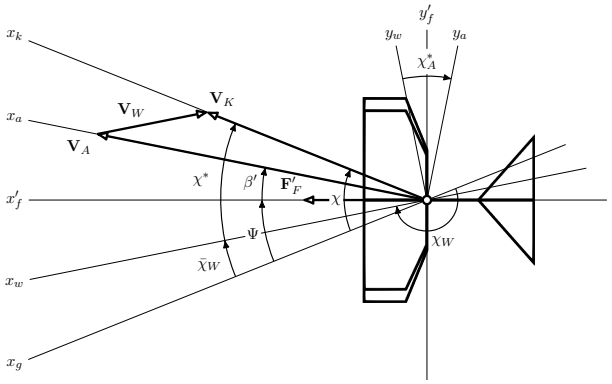


Figure 4: Axes of the horizontal flight dynamics. Projections of axes or quantities into the drawing layer are marked by $\odot'$. *Based upon [28].*

The body-fixed coordinate system is defined by the body's main axes: the x_f-axis points along the fuselage towards the nose, the z_f-axis points opposite to the direction of the vertical stabilizer, and the y_f-axis pointing starboard completes the right-handed system. Its origin is placed at the centre of the tracked body (defined by the four LED markers), which we assume to be close enough to the centre of mass.

3 CONTROL APPROACH

For the problem of precision position control we propose a hierarchic control scheme according to Fig. 5. In contrast to the classical flight-control hierarchy, an additional *speed-thrust control* block was inserted between attitude and guidance control to deal with the coupling of lift and thrust, induced by varying effects of flapping and gliding at different flight speeds.

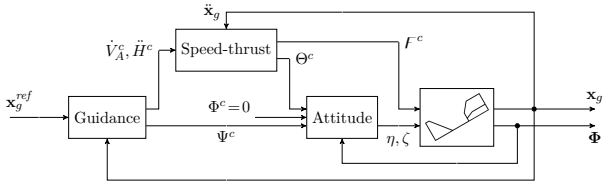


Figure 5: The hierarchic control scheme for the DelFly control approach.

In the longitudinal x_a-z_a-plane, the motion is determined by the air-speed V_A and the change in height $\dot{H}$. For the speed-thrust controller, we make the following assumptions about the flight condition of the DelFly:

- the air-speed vector lies in the horizontal plane;
- the thrust vector is aligned with the air-speed.

These assumptions are fully satisfied in steady-state flight in the wind-tunnel, and sufficiently met for slight deviations from the set-point.

3.1 Guidance control

Flying in the wind-tunnel, the direction of the aircraft velocity $\vec{\mathbf{V}}_A$ is (in steady-state) pointing opposite the wind velocity vector $\vec{\mathbf{V}}_W$, that is we have $\vec{\mathbf{V}}_A = -\vec{\mathbf{V}}_W$ and there is no motion relative to ground ($\vec{\mathbf{V}}_K = \vec{\mathbf{V}}_A + \vec{\mathbf{V}}_W = \mathbf{0}$, see Fig. 4). As the wind-direction is constant, we introduce the non-rotating wind-tunnel reference axis system $[x_w, y_w, z_w]$ opposing wind azimuth χ_W and note the angle between normal earth-fixed x_g-axis and wind-tunnel reference x_w-axis, clockwise positive, by $\bar{\chi}_W = \chi_W - \pi$.

The guidance control approach presented here is based on non-linear dynamic inversion [29]. Dynamic inversion control enjoys the benefits of directly injecting the desired closed-loop system behaviour, *i.e.* the coefficients of the characteristic polynomials, via the control law. In the axes of the wind-tunnel reference system, assuming constant wind speed ($\dot{V}_W = 0$), we end up with the following component-wise accelerations in steady-state flight:

$$\ddot{x}_w = \dot{V}_A - \dot{V}_W = \dot{V}_A \tag{1}$$

$$\ddot{y}_w = m^{-1}\left(F_F \sin(\Psi - \bar{\chi}_W) - F_D \sin(\chi_A - \bar{\chi}_W)\right) \tag{2}$$

$$\ddot{z}_w = -\ddot{H}. \tag{3}$$

For small angles $\Psi - \bar{\chi}_W$ and $\chi_A - \bar{\chi}_W$, the lateral equation reduces to

$$\ddot{y}_w = m^{-1} F_F \left(\Psi - \bar{\chi}_W\right). \tag{4}$$

Using $\dot{V}_A$, $\ddot{H}$, and Ψ as commands to the underlying control loops we have the control laws

$$\dot{V}_A^c = \nu_x \tag{5}$$

$$\Psi^c = \lambda_{\Psi\ddot{y}} \nu_y + \bar{\chi}_W \tag{6}$$

$$\ddot{H}^c = \nu_z, \tag{7}$$

http://www.imavs.org/

http://www.imavs.org/

where the lateral ratio $\lambda_{\Psi\ddot{y}}$ is introduced to cancel $m^{-1}F_F$ in Eq. 4. The pseudo-commands are given by

$$\nu_x = d_{\dot{V}_A x_w}\Delta\dot{x}_w + k_{\dot{V}_A x_w}\Delta x_w \quad (8)$$

$$\nu_y = d_{\ddot{y}_w y_w}\Delta\dot{y}_w + k_{\ddot{y}_w y_w}\Delta y_w + i_{\ddot{y}_w y_w}\int \Delta y_w \,\mathrm{d}t \quad (9)$$

$$\nu_z = d_{\ddot{H} z_w}\Delta\dot{z}_w + k_{\ddot{H} z_w}\Delta z_w, \quad (10)$$

where $i_{\ddot{y}_w y_w}$ is a lateral integral gain added to correct steady-state offsets in the heading controller. Thus, the closed loop systems resemble second-order behaviour. The lateral integral gain $i_{\ddot{y}_w y_w}$ and ratio $\lambda_{\Psi\ddot{y}_w}$ have been tuned in the wind-tunnel.

3.2 Speed-thrust control

Controlling the behaviour in the longitudinal plane is the core task of the designed controller. Since the generation of lift and thrust is highly coupled, the speed-thrust control law needs to command a suitable combination of pitch angle and flapping frequency. In Fig. 6, the overall control strategy is shown: pitch angle and throttle are commanded by a forward control block from the forward and vertical acceleration set-points, which are being adjusted by the feedback loop.

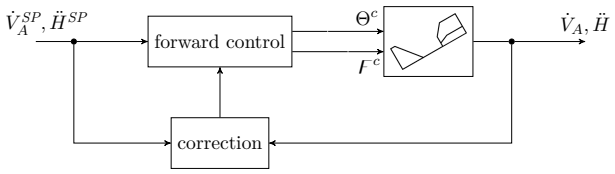


Figure 6: The overall control strategy for speed-thrust control.

Forward control Using static wind-tunnel measurements for different wind speeds, pitch angles, and throttle commands [30], a linear relationship between thrust F_F , lift F_L and pitch angle Θ and throttle F was found by first-oder Taylor linearisation:

$$\begin{bmatrix} F_F \\ F_L \end{bmatrix} = \begin{bmatrix} F_F(\Theta_0, F_0, V_W) \\ F_L(\Theta_0, F_0, V_W) \end{bmatrix} + \mathbf{F}_{\mathcal{E}}(V_W)\begin{bmatrix} \Delta\Theta \\ \Delta F \end{bmatrix}, \quad (11)$$

where Θ_0, F_0 denote the condition of no acceleration (equilibrium) and $\mathbf{F}_{\mathcal{E}}$ the matrix of force gradients at the equilibrium; the lift and thrust force in equilibrium is $F_{F0} = F_{D0}(V_W)$ and $F_{L0} = F_G$.

Considering our initial assumption ($\Delta F_F = m\dot{V}_A, \Delta F_L = m\ddot{H}$), we obtain a wind speed-dependent forward control law by inversion of Eq. 11 as

$$\begin{bmatrix} \Theta^c \\ F^c \end{bmatrix} = \begin{bmatrix} \Theta_0 \\ F_0 \end{bmatrix} + m\mathbf{F}_{\mathcal{E}}^{-1}\begin{bmatrix} \dot{V}_A^{SP} \\ \ddot{H}^{SP} \end{bmatrix} \quad (12)$$

As $\mathbf{F}_{\mathcal{E}}$ has been derived for different wind speeds, our forward control law follows a *gain scheduling* approach.

Feedback correction In feedback loop, we employ a simple PI control law in order to correct model uncertainties and reject disturbances

$$\begin{bmatrix} \dot{V}_A^{\mathrm{fb}} \\ \ddot{H}^{\mathrm{fb}} \end{bmatrix} = k\begin{bmatrix} \Delta\dot{V}_A \\ \Delta\ddot{H} \end{bmatrix} + i\int \begin{bmatrix} \Delta\dot{V}_A \\ \Delta\ddot{H} \end{bmatrix} \mathrm{d}t, \quad (13)$$

where the integrated error can be calculated as $\int \Delta\dot{V}_A \,\mathrm{d}t = \int \dot{V}_A^{SP} \,\mathrm{d}t - V_A; \int \Delta\ddot{H} \,\mathrm{d}t = \int \ddot{H}^{SP} \,\mathrm{d}t - \dot{H}$.

As $\mathbf{F}_{\mathcal{E}}^{-1}$ features a guess of direction of the force gradients, we add the correction *before* the forward control; thus, the combined forward-backward control law results into

$$\begin{bmatrix} \Theta^c \\ F^c \end{bmatrix} = \begin{bmatrix} \Theta_0 \\ F_0 \end{bmatrix} + m\mathbf{F}_{\mathcal{E}}^{-1}\Big(\begin{bmatrix} \dot{V}_A^{SP} \\ \ddot{H}^{SP} \end{bmatrix} + k\begin{bmatrix} \Delta\dot{V}_A \\ \Delta\ddot{H} \end{bmatrix} + i\int \begin{bmatrix} \Delta\dot{V}_A \\ \Delta\ddot{H} \end{bmatrix}\Big). \quad (14)$$

3.3 Semi-adaptive control approach

The combined forward-backward controller, as presented so far, compensates model errors in both force gradients and equilibrium conditions by feedback. We enhance the control approach by a quasi-adaptation stage to initially adapt to the equilibrium condition (Θ_0, F_0); we call this extended approach *semi-adaptive*.

First, we can use a position feedback to find the "true" equilibrium condition, without de-stabilizing the system:

Lemma 1. *Let $\hat{\Theta}_0$, $\hat{F}_0$, and $\hat{\mathbf{F}}_{\mathcal{E}}$ be close guesses of the pitch angle, throttle command, and matrix of force gradients at equilibrium, respectively. Given an initial longitudinal position (x_{w0}, H_0) and a feedback law*

$$\begin{bmatrix} \Theta^c \\ F^c \end{bmatrix} = \begin{bmatrix} \hat{\Theta}_0 \\ \hat{F}_0 \end{bmatrix} + m\hat{\mathbf{F}}_{\mathcal{E}}^{-1}\gamma\begin{bmatrix} x_{w0} - x_w \\ H_0 - H \end{bmatrix}, \quad (15)$$

for a suitable gain $\gamma \in \mathbb{R}_{>0}$ it holds

1. *there is a position (x'_w, H') close to (x_{w0}, H_0) where the equilibrium condition is met;*
2. *in the neighbourhood of (x'_w, H') this position will be approached.*

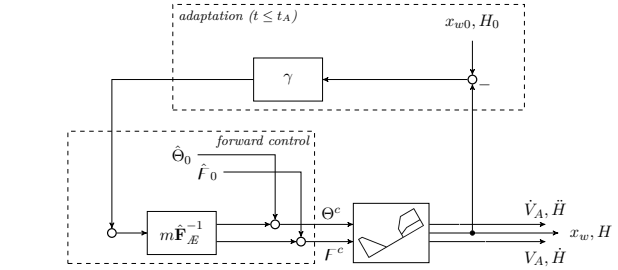
Proof. See Appendix. □

That is, the stable position (x'_w, H') is approached and Eq. 15 yields the true pitch angle $\Theta_0 = \Theta^c(x'_w, H')$ and throttle command $F_0 = F^c(x'_w, H')$.

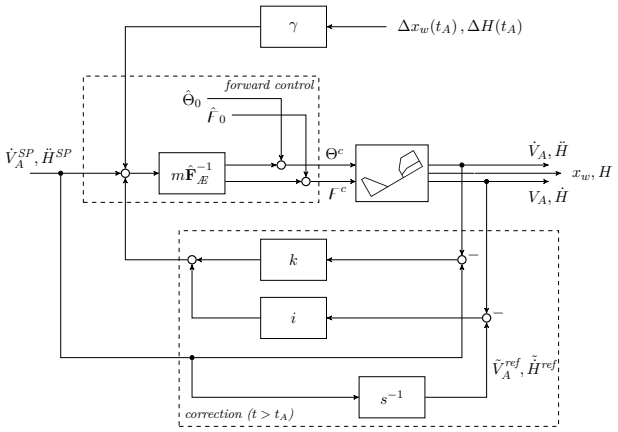
We now split our control approach into two subsequent stages: *adaptation* and *correction*, see Fig. 7. First, the position feedback of Eq. 15 stabilizes the position; as soon as the equilibrium is found by Lemma 1 (time t_A), the controller switches to the correction stage (and remains here).

3.4 Attitude control

We use elevator and rudder of the DelFly to control pitch angle and heading, respectively. While roll is also affected by the rudder, it quickly returns to zero thanks to passive stability. We employ standard PID feedback and use directly the error in rates for the derivative part.



(a) Adaptation stage, $t \leq t_A$.



(b) Correction stage, $t > t_A$.

Figure 7: The two-stage semi-adaptive control approach: at time $t = t_A$, the semi-adaptive controller switches from the adaptation (a) to the correction (b) stage.

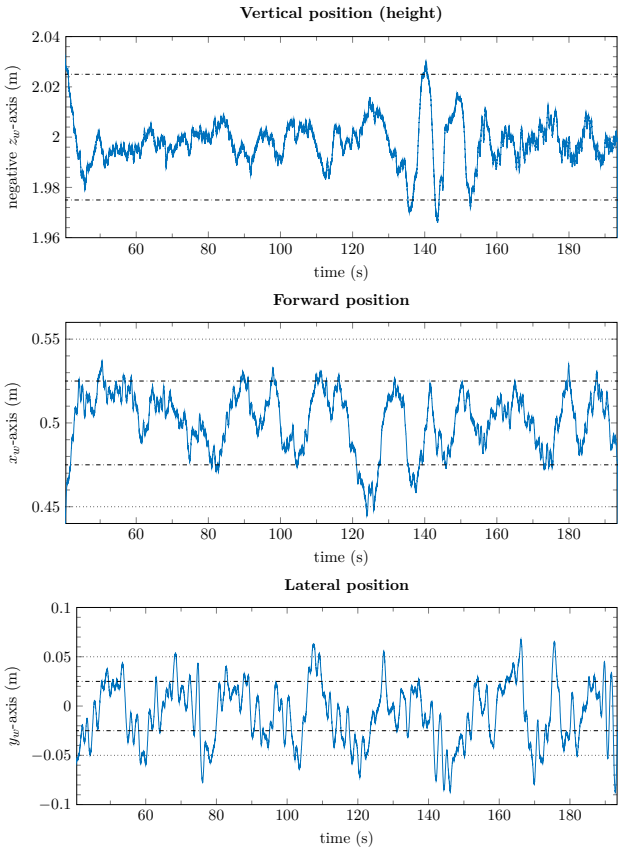


Figure 8: Steady-state position (by OptiTrack); dashed and dotted lines are indicating $\pm 2.5\,\mathrm{cm}$ and $\pm 5.0\,\mathrm{cm}$, respectively. (*wind speed* $V_W^{SP} = 0.8\,\mathrm{m/s}$*; semi-adaptive speed-thrust control with* $k = 0$*,* $i = 300\,\%$.)

4 RESULTS

If not stated otherwise, the wind-tunnel speed controller was set to $V_W^{SP} = 0.8\,\mathrm{m/s}$. Because the wind speed measurement was not available on-board, the wind tunnel set point speed was used instead to choose the forward gains of the speed-thrust controller.

4.1 Precision in steady-state

The steady state results are shown in Fig. 8. We achieved the highest steady-state precision using only an integral gain $i = 300\,\%$ in the correction stage. The control precision in the vertical direction was around $\pm 1.0\,\mathrm{cm}$, except for a disturbance around time $t = 140\,\mathrm{s}$. The precision in the forward axis was around $\pm 2.5\,\mathrm{cm}$.

The performance in the lateral axis is not influenced by the integral gain but heavily oscillating; thus, a lateral precision of $\pm 5.0\,\mathrm{cm}$ is achieved most of the time. We further discuss the lateral results in Section 4.2.

The RMS error over the time interval displayed in Fig. 8 (approx. $150\,\mathrm{s}$) was $0.9\,\mathrm{cm}$, $1.7\,\mathrm{cm}$ and $3.2\,\mathrm{cm}$ for vertical, forward and lateral direction, respectively.

4.2 Robustness to wind speed

While flying, we increased the wind speed set-point stepwise from 0.7 up to $1.3\,\mathrm{m/s}$ (Fig. 9); neither forward nor feedback gains were changed during this test.

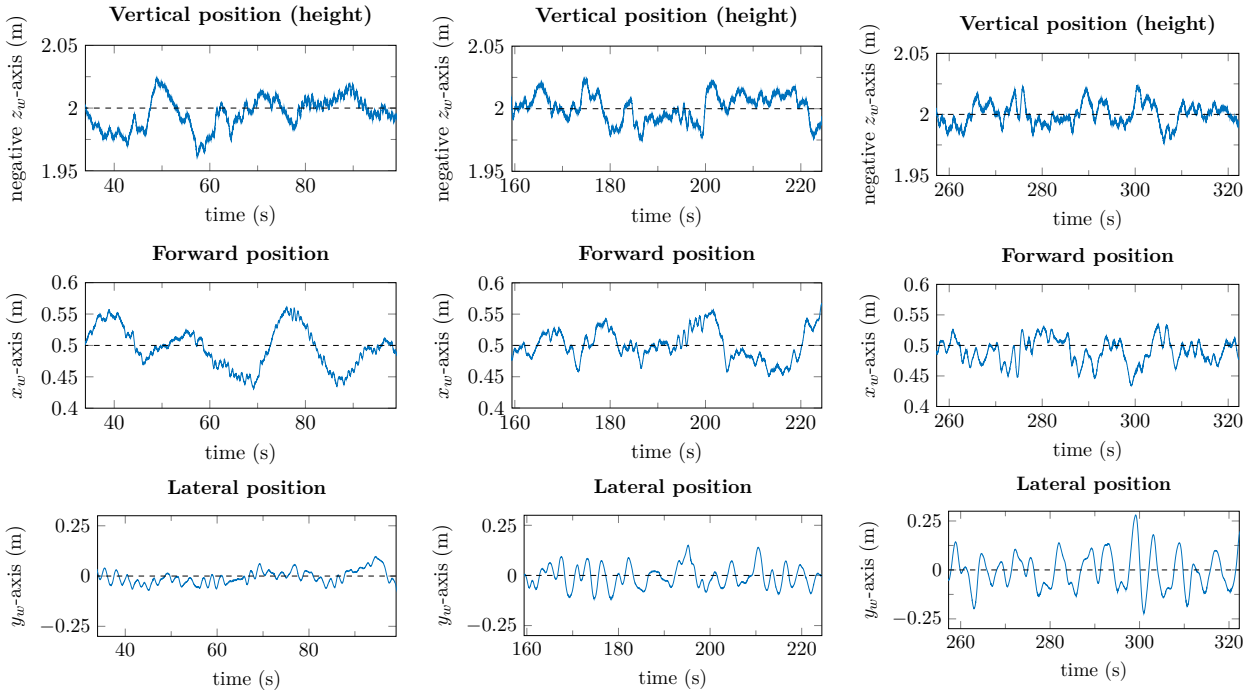
In the vertical and forward axes, the control approach is clearly robust and the precision remains unaffected. As one can see from the RMS error in Tab. 1, the accuracy is even increasing slightly with the wind speed.

V_W^{SP}	$0.7\,\mathrm{m/s}$	$1.0\,\mathrm{m/s}$	$1.3\,\mathrm{m/s}$
RMS z_w (cm)	1.35	1.10	0.90
RMS x_w (cm)	3.26	2.47	2.25
RMS y_w (cm)	3.44	5.52	9.22

Table 1: RMS position error for different wind speeds.

The precision in lateral axis, however, deteriorates for increasing wind speed; so does the RMS error. Since the rudder is more effective in low-pitch, fast-forward flight and a change of heading induces an increased lateral velocity com-

http://www.imavs.org/



(a) Wind speed $V_W^{SP} = 0.7\,\mathrm{m/s}$. (b) Wind speed $V_W^{SP} = 1.0\,\mathrm{m/s}$. (c) Wind speed $V_W^{SP} = 1.3\,\mathrm{m/s}$.

Figure 9: Results for different wind speeds. (*semi-adaptive speed-thrust control with* $k = 0$, $i = 200\,\%$.)

ponent, the lateral control loop as third-order integrator system is more sensitive to increased wind speed and thus airspeed and pitch angle. By gain scheduling or dynamic inversion control, these couplings can be taken into account.

4.3 *Height steps for different poles*

The gains $d_{\ddot{H}z_w}, k_{\ddot{H}z_w}$ of the guidance control law are determined by the desired poles of the closed loop

$$\left(s^2 + d_{\ddot{H}z_w} s + k_{\ddot{H}z_w}\right) z_w(s) = \left(s - \varsigma_{z1}\right)\left(s - \varsigma_{z2}\right) z_w(s) = 0 \tag{16}$$

that is,

$$d_{\ddot{H}z_w} = -\varsigma_{z1} - \varsigma_{z2} \tag{17}$$

$$k_{\ddot{H}z_w} = \varsigma_{z1}\varsigma_{z2} \tag{18}$$

Desiring a "smooth" approach without overshoot, we chose pole pairs $\varsigma_{z1,2} \in \{-1, -3, -5\}$ on the left-hand real axis. The step responses for these pole pairs were tested in a flight arena where the DelFly flew at constant forward speed in circles with radii of 2 m, simulating the wind tunnel flight. The step responses are compared in Fig. 10.

The system response is as fast as desired and without overshoot beyond the accuracy of 2.5 cm for all three pole pairs; for the later pairs however, there is an initial deadband with a succeeding steeper rise indicating a delayed reaction of the underlying controller.

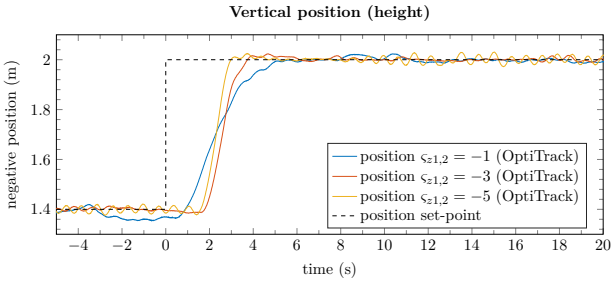


Figure 10: Height step response for different guidance poles, $\varsigma_{z1,2} \in \{-1, -3, -5\}$.[2]

5 CONCLUSION

We performed the first high-precision flight of a flapping-wing micro air vehicle in a wind-tunnel, maintaining an accuracy of ± 5 cm in all axes for most of the time; in the vertical axis in particular, we achieved an accuracy of ± 2.5 cm and below, up to ± 1 cm for tens of seconds. This performance both in position accuracy and in duration is more than sufficient in order to perform time-resolved stereo PIV.

Although the controller was tuned for a single wind speed (0.8 m/s), the longitudinal performance remained unaffected

[2] For $\varsigma_{z1,2} = -1$, the height set-point $H^{SP}(t < 0)$ was just below 1.4 m.

http://www.imavs.org/

for the whole speed range tested (0.7 to $1.3\,\mathrm{m/s}$). The performance of the separated lateral controller decreased for higher speeds. This should be improved by gain scheduling or by dynamic inversion that take into account the roll and yaw coupling effects and the varying rudder effectiveness with speed. In addition—thanks to the modular design of the controller—the precision can be further improved by a fully-adaptive speed-thrust controller as well as by augmented filtering of position and attitude data originating from the off-board motion tracking and on-board inertial measurement.

To conclude, despite the lack of an accurate model of the DelFly MAV over its full flight envelope, we solved the flapping-wing position control problem successfully and, furthermore, in a new and advanced manner. Our work enables future investigations of the flapping wing aerodynamics *in-flight*, which should bring better understanding of the underlying aerodynamic mechanisms and result in advanced, more agile and more efficient flapping wing MAV designs.

REFERENCES

[1] M. Keennon, K. Klingebiel, H. Won, and A. Andriukov. Development of the Nano Hummingbird: A Tailless Flapping Wing Micro Air Vehicle. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, Nashville, US-TN, Jan. 2012. American Institute of Aeronautics and Astronautics.

[2] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood. Controlled Flight of a Biologically Inspired, Insect-Scale Robot. *Science*, 340(6132):603–607, 2013.

[3] G. C. de Croon, K. M. de Clercq, R. Ruijsink, B. Remes, and C. De Wagter. Design, Aerodynamics, and Vision-based Control of the DelFly. *International Journal of Micro Air Vehicles*, 1(2):71–97, 2009.

[4] 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 and Biomimetics*, 6, 2011.

[5] S. Deng, T. Xiao, B. van Oudheusden, and H. Bijl. Numerical Investigation on the Propulsive Performance of Biplane Counter-flapping Wings. *International Journal of Micro Air Vehicles*, 7(4):431–440, Dec. 2015.

[6] G. K. Taylor and A. L. R. Thomas. Dynamic Flight Stability in the Desert Locust *schistocerca gregaria*. *The Journal of Experimental Biology*, 206:2803–2829, 2003.

[7] J.-S. Lee and J.-H. Han. Experimental Study on the Flight Dynamics of a Bioinspired Ornithopter: Free Flight Testing and Wind Tunnel Testing. *Smart Materials and Structures*, 21, 2012.

[8] C. Rose and R. S. Fearing. Comparison of Ornithopter Wind Tunnel Force Measurements with Free Flight. In *IEEE International Conference on Robotics and Automation*, pp. 1816–1821, Hong Kong, CN, 2014. Institute of Electrical and Electronics Engineers.

[9] J. V. Caetano, M. Percin, B. W. van Oudheusden, B. D. W. Remes, C. De Wagter, G. C. H. E. de Croon, and C. C. de Visser. Error analysis and assessment of unsteady forces acting on a flapping wing micro air vehicle: free flight versus wind-tunnel experimental methods. *Bioinspiration & biomimetics*, 10(5):056004, Oct. 2015.

[10] S. K. Ghosh, C. L. Dora, and D. Das. Unsteady Wake Characteristics of a Flapping Wing through 3D TR-PIV. *Journal of Aerospace Engineering*, 25(4):547–558, Oct. 2012.

[11] H. Ren, Y. Wu, and P. G. Huang. Visualization and characterization of near-wake flow fields of a flapping-wing micro air vehicle using piv. *Journal of Visualization*, 16(1):75–83, 2013.

[12] 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(1806), 2014.

[13] S. Deng, M. Percin, and B. van Oudheusden. Experimental Investigation of Aerodynamics of Flapping-Wing Micro-Air-Vehicle by Force and Flow-Field Measurements. *AIAA Journal*, 54(2):588–602, 2016.

[14] H. Ben-Gida, A. Kirchhefer, Z. J. Taylor, W. Bezner-Kerr, C. G. Guglielmo, G. A. Kopp, and R. Gurka. Estimation of Unsteady Aerodynamics in the Wake of a Freely Flying European Starling (*sturnus vulgaris*). *PLoS ONE*, 8(11), 2013.

[15] F. T. Muijres, L. C. Johansson, Y. Winter, and A. Hedenström. Leading Edge Vortices in Lesser Long-nosed Bats Occurring At Slow But Not Fast Flight Speeds. *Bioinspiration and Biomimetics*, 9, 2014.

[16] P. Henningsson, D. Michaelis, T. Nakata, D. Schanz, R. Geisler, A. Schröder, and R. J. Bomphrey. The complex aerodynamic footprint of desert locusts revealed by large-volume tomographic particle image velocimetry. *Journal of the Royal Society Interface*, 12(108), 2015.

[17] C. De Wagter, A. Koopmans, G. C. de Croon, B. Remes, and R. Ruijsink. Autonomous Wind Tunnel Free-Flight of a Flapping Wing MAV. In *Advances in Aerospace Guidance, Navigation and Control*, pp. 603–621. Springer, Berlin, DE, 2013.

http://www.imavs.org/

[18] J. Nowak. *Windkanal-Freiflugmessungen zur Bestimmung flugmechanischer Kenngrößen*. PhD thesis, RWTH Aachen University, Aachen, DE, 2010.

[19] J. N. Wiken. *Analysis of a Quadrotor in Forward Flight*. Master's thesis, Massachusetts Institute of Technology, Cambridge, US-MA, 2015.

[20] X. Deng, L. Schenato, and S. S. Sastry. Flapping Flight for Biomimetic Robotic Insects: Part II – Flight Control Design. *IEEE Transactions on Robotics*, 22(4):789–803, 2006.

[21] D. B. Doman, M. W. Oppenheimer, and D. O. Sigthorsson. Wingbeat Shape Modulation for Flapping-Wing Micro-Air-Vehicle Control During Hover. *Journal of Guidance, Control, and Dynamics*, 33(3):724–739, 2010.

[22] S. B. Fuller, M. Karpelson, A. Censi, K. Y. Ma, and R. J. Wood. Controlling Free Flight of a Robotic Fly Using an Onboard Vision Sensor Inspired by Insect Ocelli. *Journal of the Royal Society Interface*, 11, 2014.

[23] L. Roberts, H. Bruck, and S. Gupta. Modeling of Dive Maneuvers in Flapping Wing Unmanned Aerial Vehicles. In *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics*, p. 6, 2015.

[24] P. Chirarattananon, K. Y. Ma, and R. J. Wood. Single-loop control and trajectory following of a flapping-wing microrobot. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 37–44. Institute of Electrical and Electronics Engineers, May 2014.

[25] P. Chirarattananon, K. Y. Ma, and R. J. Wood. Perching with a robotic insect using adaptive tracking control and iterative learning control. *The International Journal of Robotics Research*, Feb. 2016.

[26] G. C. de Croon, M. Perçin, B. D. Remes, R. Ruijsink, and C. De Wagter. *The DelFly: Design, Aerodynamics, and Artificial Intelligence of a Flapping Wing Robot*. Springer, Dordrecht, NL, 2016.

[27] B. D. W. Remes, P. Esden-Tempski, F. van Tienen, E. Smeur, C. D. Wagter, and G. C. H. E. de Croon. Lisa-S 2.8g Autopilot for GPS-based Flight of MAVs. In *IMAV 2014: International Micro Air Vehicle Conference and Competition*, pp. 280–285, Delft, NL, Aug. 2014.

[28] R. Brockhaus. *Flugregelung*. Springer, Berlin, DE, 2nd, revised edition, 2001.

[29] J.-J. Slotine and W. Li. *Applied Non-Linear Optimal Control*. Prentice-Hall, Upper Saddle River, US-NJ, 3rd edition, 1991.

[30] J. A. Koopmans and M. Karásek. Static force measurement of the Delfly flapping-wing MAV (*working title*). unpublished, 2015.

APPENDIX A: PROOF OF LEMMA 1

Proof. Let Θ_0, F_0, $\mathbf{F}_{Æ}$ be the true pitch angle, throttle command, and matrix of force gradients in equilibrium;

1. If the equilibrium condition is met, *i.e.* $\Theta^c = \Theta_0$, $F^c = F_0$, Eq. 15 yields

$$\begin{bmatrix} x'_w \\ H' \end{bmatrix} = \begin{bmatrix} x_{w0} \\ H_0 \end{bmatrix} + m^{-1}\hat{\mathbf{F}}_{Æ}\gamma^{-1} \begin{bmatrix} \Theta_0 - \hat{\Theta}_0 \\ F_0 - \hat{F}_0 \end{bmatrix} \tag{19}$$

and $|x_{w0} - x'_w|$, $|H_0 - H'|$ are small for a suitable γ.

2. Let (x_w, H) be a position in the neighbourhood of the equilibrium (x'_w, H'); rearranging Eq. 19 yields

$$\begin{bmatrix} \Theta^c \\ F^c \end{bmatrix} = \begin{bmatrix} \hat{\Theta} \\ \hat{F} \end{bmatrix} + m\hat{\mathbf{F}}_{Æ}^{-1}\gamma \begin{bmatrix} x_{w0} - x'_w \\ H_0 - H' \end{bmatrix} = \begin{bmatrix} \Theta_0 \\ F_0 \end{bmatrix}$$

and inserting into Eq. 15, we get

$$\begin{bmatrix} \Theta^c \\ F^c \end{bmatrix} = \begin{bmatrix} \hat{\Theta} \\ \hat{F} \end{bmatrix} + m\hat{\mathbf{F}}_{Æ}^{-1}\gamma \begin{bmatrix} x_{w0} - x'_w \\ H_0 - H' \end{bmatrix} + m\hat{\mathbf{F}}_{Æ}^{-1}\gamma \begin{bmatrix} x'_w - x_w \\ H' - H \end{bmatrix} = \begin{bmatrix} \Theta_0 \\ F_0 \end{bmatrix} + m\hat{\mathbf{F}}_{Æ}^{-1}\gamma \begin{bmatrix} x'_w - x_w \\ H' - H \end{bmatrix} \tag{20}$$

Finally, recalling

$$\begin{bmatrix} \dot{V}_A \\ \ddot{H} \end{bmatrix} = m^{-1}\mathbf{F}_{Æ}(V_W) \left(\begin{bmatrix} \Theta \\ F \end{bmatrix} - \begin{bmatrix} \Theta_0 \\ F_0 \end{bmatrix} \right) \tag{21}$$

and assuming $\Theta = \Theta^c, F = F^c$ we get

$$\begin{bmatrix} \dot{V}_A \\ \ddot{H} \end{bmatrix} = \mathbf{F}_{Æ}\hat{\mathbf{F}}_{Æ}^{-1}\gamma \begin{bmatrix} x'_w - x_w \\ H' - H \end{bmatrix} \tag{22}$$

As $\hat{\mathbf{F}}_{Æ}$ is a close guess of $\mathbf{F}_{Æ}$, we can assume $\mathbf{F}_{Æ}\hat{\mathbf{F}}_{Æ}^{-1}$ is diagonal and positive; thus we have

$$\begin{aligned} \dot{V}_A &= \lambda_1\gamma\,(x'_w - x_w) \\ \ddot{H} &= \lambda_2\gamma\,(H' - H) \end{aligned} \tag{23}$$

where $\lambda_1\gamma, \lambda_2\gamma > 0$ and the equilibrium position (x'_w, H') is stable in its neighbourhood.

□

http://www.imavs.org/

Real-time Simultaneous Localization and Mapping for UAV: A Survey

Jiaxin Li, Yingcai Bi, Menglu Lan, Hailong Qin, Mo Shan, Feng Lin, Ben M. Chen
National University of Singapore

http://www.imavs.org/

ABSTRACT

Simultaneous Localization and Mapping (SLAM) refers to the problem of using various sensors like laser scanner, RGB cameras, RGB-D cameras, etc, to estimate the position of the robot, and concurrently construct the 2D/3D map of the environment. The SLAM community has made great progress in the past few decades. So far the 2D SLAM problem with range finders is considered as solved, while the real-time 3D SLAM, especially robust and high quality visual SLAM on UAVs, remains an open problem. This article aims to give a picture of the evolution and very recent development of SLAM algorithms, and emphasis is given on real-time SLAM methods that are suitable for Unmanned Aerial Vehicles (UAVs).

1 INTRODUCTION

Autonomous UAVs, robots, vehicles are receiving more and more attention in academia and industry. Although the industry has noticed the potential applications of unmanned systems, including driver-less cars, building inspection, surveillance etc, lots of problems remain unsolved. Most of problems are caused by the failure, inaccuracy or instability of perception. In outdoor environment, GPS provides accuracy localization and partly solved the perception problem. In situations that GPS is unavailable, or high accuracy of localization or mapping is needed, accurate and robust SLAM algorithms are required. SLAM is especially difficult for UAVs because of the strict requirement of real-time processing, and the limitation of UAV's payload.

1.1 Localization

During navigation, a robot's positions at discrete time instants are related by rigid motion transformation $T \in \mathbb{R}^{4\times4}$. Aiming to solve T and concatenate T into a trajectory, localization have always been a hot topic in both research and application, and the emergence of Global Positioning System (GPS) have partly solved this problem in outdoor environments. For GPS denied situations, solutions like Wifi/Ultra Wide Band (UWB) positioning, Motion Capture Camera System (VICON) are proposed. These solutions require extern hardwares, making them impractical or too expensive for most indoor applications. Therefore, on-board localization, which utilizes only on-board sensors to estimate positions, becomes a popular and feasible solution for most robots. Localization is especially indispensable for autonomous UAVs because they can not "stop" in the air like ground robots. Localization is sometimes called odometry in robotics. In particular, Visual Odometry (VO), i.e. odometry using cameras, is experiencing rapid growth recently.

1.2 Mapping

The significance of mapping comes mainly in three aspects. First of all, a map supports tasks like path planning and obstacle avoidance, which are basic requirements of autonomous navigation. Secondly, the map itself is the objective for many robot applications. Besides providing intuitive visualization, it allows further analysis of the explored space, including dimension evaluation, object recognition, etc. Thirdly, proper mapping will improve the accuracy and robustness of localization. One of the most important features of mapping is loop closure, which allows robots to recognize a place visited previously and optimize it's estimated trajectory accordingly, therefore drift is reduced or even rejected. Also, the ability of recognizing a place in the existing map enables robots to recover from odometry failure.

In the early ages, localization and mapping are considered separately, but later researchers found that SLAM is kind of a "chicken and egg" problem, i.e., localization and mapping are highly interrelated. A map is needed for accurate localization, while localization is needed for mapping. Or in another word, mapping or localization can be solved if one of them is known accurately. Therefore, current Visual Odometry algorithms typically include the mapping function as well, although the map may not be suitable for path planning or other applications. The only difference between modern odometry and SLAM is whether loop closure, or global map optimization is available [1].

Classical SLAM methods in Section 4 prefer to jointly estimate pose and map. Later more advanced methods in Section 5 usually employ the interleave idea of Parallel Tracking and Mapping (PTAM) [2] and put localization and mapping into two parallel threads.

2 RELATED WORKS

Proposed by the recent review by Cadena et al. [1], the history of SLAM can be roughly divided into three ages. In the classical age, 1986 - 2004, the mainstream of the community is the probabilistic formulation and filtering techniques. Reviews of first 20 years' research were published by Durrant

http://www.imavs.org/

Whyte and Bailey [3, 4]. The subsequent period, 2004-2015, is called the algorithm analysis age, where the fundamental properties, including observability, convergence and consistency, were investigated [5]. Many theories from the Computer Vision community, such as the SfM, were widely used by SLAM algorithms. Optimization based methods were developed, and believed to outperform the classical filtering based SLAM. The community was creating many open-source projects and pushing SLAM to practical applications. Currently the community is entering the third period, the robust perception age, where robust performance and high-level environment understanding are the focal points.

At the crossing of the algorithm analysis age and the robust perception future, emerging SLAM algorithms are marching towards the edge of real world application. The ORB-SLAM and LSD-SLAM published in the last two year are regarded as the most promising SLAM that can be applied on UAVs. Current reviews of SLAM are either outdated to include such recent progress [3–6], or focus on the theory development [1]. A survey introducing the real-time SLAM algorithms will provide the researchers in robotics with a clear picture, and help to apply or improve SLAM in the context of UAV.

This paper focuses on the milestones that can achieve real-time robust performance on UAV platforms, using either laser scanners or cameras. We starts by describing the modern architecture of SLAM in Section 3. Section 4 and 5 follows the anatomy of [1], to summarize the the classical and modern SLAM algorithms.

3 MODERN FRAMEWORK

A SLAM system can be partitioned into two parts, the front-end and back-end, shown in Figure 1. Usually the front-end requires techniques of computer vision and signal processing, in order to abstract the geometry information into mathematical model and feed it into the back-end. The back-end is in charge of optimizing the model, often called factor-graph, to refine the pose and map.

The sensor dependent front-end takes raw data from the sensors and pre-processes it. The pre-processing includes feature extraction, short term and long term data association, etc. In the case of featured-based visual SLAM in Section 5.2, detected image feature points are associated to 3D geometrical points. For direct tracking methods in Section 5.3, tracking between frames are conducted by front-end as well.

In Section 5, the output of front-end is usually modeled as a factor graph, where positions and space structures serve as nodes, and the rigid body transformation T as the edges connecting the nodes. Mathematically, this is a Maximum A Posteriori (MAP) problem formulated as 1. Measurements $Z = z_k : k = 1, \cdots, m$ are expressed as a function of $X = x_k : k = 1, \cdots, m$, $z_k = h_k(x_k) + \epsilon_k$, where x_k represents the unknown variable like position or space structure, i.e. the nodes in the factor graph. Under the assumption of zero mean Gaussian noise ϵ_k, (1) becomes (2), which can be optimized with iterative Gauss-Newton or Levenberg-Marquardt algorithms. The detailed formulation of the MAP problem can be found in [1].

$$X^* = \arg\max_X \mathbb{P}(X|Z) = \arg\max_X \mathbb{P}(Z|X)\mathbb{P}(X) \quad (1)$$

$$X^* = \arg\min_X \sum_{k=0}^{m} \frac{1}{2} \|h_k(x_k) - z_k\|^2_{\Omega_k} \quad (2)$$

For classical SLAM in Section 4, instead of constructing such MAP problem, nonlinear filtering approaches including Extended Kalman Filter (EKF) and Particle Filter (PF) are used to jointly estimate the position and map. A systematic review can be found in [3, 4].

According to some research [7], the accuracy of MAP estimation is believed to be better than that of the filtering method. But some advanced filtering systems may have equivalent performance as well, such as the Multi-State Constraint Kalman Filter [1].

4 FILTERING BASED METHODS

The idea of explicitly formulating SLAM as the probabilistic problem originated at the 1986 IEEE Robotics and Automation Conference [3]. Similar to the modern architecture of Section 3, the features and landmarks are extracted by the front-end, and the back-end is the joint estimation of localization and landmark maps. Later, EKF and PF were widely used in the back-end.

4.1 Probabilistic Formulation

At time instant k, the unknown variables are explicitly defined into position and orientation x_k, landmark location m_k. Control vector u_k defines the drive of transition from x_{k-1} to x_k. Measurement z_k is the observation of m_k at the state of x_k. In addition, the variable sets are defined in (3). Under the probabilistic formulation, the SLAM is implemented in an iterative time-update (4) and measurement update (5) fashion, where motion model is described by $P(x_k|x_{k-1}, u_k)$ and observation model described by $P(z_k|x_k, m)$. The visualization of the filtering inference is shown in Figure 2(a).

$$\begin{aligned} X_k &= \{x_0, x_1, \cdots, x_k\} = \{X_{k-1}, x_k\} \\ U_k &= \{u_0, u_1, \cdots, u_k\} = \{U_{k-1}, u_k\} \\ m &= \{m_1, m_2, \cdots, m_n\} \\ Z_k &= \{z_0, z_1, \cdots, z_k\} = \{Z_{k-1}, z_k\} \end{aligned} \quad (3)$$

$$P(x_k, m|Z_{k-1}, U_k, x_0) = \int P(x_k|x_{k-1}, u_k) P(x_{k-1}, m|Z_{k-1}, U_{k-1}, x_0) dx_{k-1} \quad (4)$$

$$P(x_k, m|Z_k, U_k, x_0) = \frac{P(z_k|x_k, m) P(x_k, m|Z_{k-1}, U_k, x_0)}{P(z_k|Z_{k-1}, U_k)} \quad (5)$$

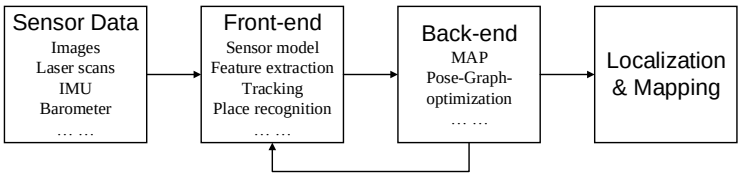


Figure 1: Modern architecture of SLAM consists of front-end and back-end.

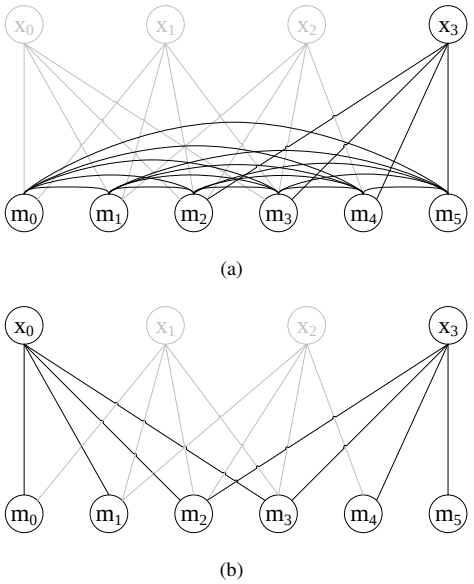


Figure 2: (a) Visualization of inference under the filtering framework. All landmarks m, and their correlation are maintained and updated. (b) Inference of the keyframe-based method. Only motion and map of keyframes are maintained.

4.1.1 EKF SLAM

The most intuitive way to simplify (4) and (5) is to employ EKF with the linear Gaussian assumption. Motion model and observation model are linearized so that EKF is applicable, and real-time performance can be achieved. However, the linear Gaussian assumption is usually not the practical case, and may lead to divergence, or significant estimation error. Also, EKF filtering is fragile to incorrect data association.

4.1.2 FastSLAM

To overcome the fragile linear assumption of EKF SLAM, FastSLAM was introduced by Montemerlo et al. [8]. Instead of linearizing the motion model $P(x_k|x_{k-1}, u_k)$, the non-Gaussian motion model is estimated with Monte Carlo sampling, i.e. Particle Filter. Direct sampling with m and X_k is computational infeasible because of their high dimension. Rao-Blackwellization is used to partition the joint estimation of m and X_k into product of independent Gaussian distributions, so that the sampling is greatly accelerated. The detailed derivation of FastSLAM can be found in [3, 8].

4.2 Laser SLAM

Proved by research like [9], the above probabilistic architecture is able to achieve practical performance with 2D laser scanner. A significant issue in filtering based laser SLAM, is the feature extraction and data association problem. Traditional features are modeled as lines, circles, corners, etc [10]. The major problem of such geometric features is that they are environment sensitive. For example, features relying on corners will fail in a mess environment, where walls can not be scanned by the laser. An improvement is the scan correlation, which regards the raw laser scan as the feature, and the alignment of the raw scans can be efficiently solved with Iterative Closest Point (ICP) [11].

In practice, the mapping between measurements and landmarks is rarely known [12]. In EKF SLAM, maximum likelihood estimation for each observation is commonly used to solve data association. Some enhanced algorithms solve the best association of all observations at the same time [13]. Compared to EKF, FastSLAM is much less prone to inaccurate data association. Actually, FastSLAM can be extended to sample on data associations, so that it can be estimated simultaneous with robot paths [12]. In [12], Montemerlo and Thrun showed an impressive SLAM result with FastSLAM using a single 2D laser. In the experiment of driving for over 4km, the position error was less than 10 meters. In conclusion, it is proved that FastSLAM is much better than EKF SLAM in terms of accuracy and robustness.

4.3 Visual SLAM

Compared to the success of filter based laser SLAM, vision-only SLAM remained unsolved, because of scarcity of computational power, lack of depth information, and the difficulty of extracting or associating features. Various filtering techniques, including EKF [14], Unscented Kalman Filter (UKF), Particle Filter, sparse information filter, were attempted under the filtering framework mentioned above.

Stereo cameras are used to overcome the depth problem [15]. Davison and Murray [15] achieved 5Hz SLAM using fixating active stereo, and proved that a small number of landmarks is enough to provide accurate pose estimation. In some

http://www.imavs.org/

work, plane assumption was applied on downward looking cameras, to further simply the depth estimation.

Feature detector and descriptors like SIFT, SURF are widely used nowadays because of their robustness and convenience for data association, i.e. matching landmarks with image measurements. However, these features were computational too expensive before 2006. As the workaround, salient image patches, line segments were popular.

MonoSLAM by Davison et al. [14] is a significant milestone of Visual SLAM, because it is the first real-time monocular SLAM that achieve the speed of 30Hz with adequate accuracy and robustness for robotics. MonoSLAM is under the EKF frameworks where the state vector contains position, orientation, linear velocity, angular velocity, and the 3D position of landmarks. Using the famous good-feature-to-track detector by Shi and Tomasi, a landmark is represented by a salient image region, the depth of the patch, the orientation of its norm, and the uncertainty of its position. Efficient data association is achieved by searching features with a precalculated area, instead of in the whole image.

5 ANALYSIS BASED METHODS

Poor scalability is the main drawback of filtering SLAM. Because the joint distributions of all landmarks have to be maintained and updated all the time, the computational cost become unacceptable eventually, shown in Figure 2(a). This scalability problem is serious for Visual SLAM because images contain much more features than laser scans.

To mitigate such limitation, analysis based methods, or the so called optimization based methods, are proposed to maintain only a small subset of motions and maps. These subsets that represent the trajectory and map are called keyframes. Shown in Figure 2(b), although the number of nodes in the graph is larger, the interconnections between nodes remain sparse. Keyframe based graph can be optimized efficiently even if there are large number of motions and features. In computer vision, graph of Figure 2(b) is referred as Bundle Adjustment (BA). In many SLAM algorithms, the graph consists of only positions x. In that case, by applying the MAP formulation in Section 3, open-source libraries including g2o are able to optimize a graph with tens of thousands of nodes within one second.

By making a systematic comparison between filtering based and analysis based Visual SLAM methods, Strasdat et al. [7] came to the conclusion that keyframe based BA outperforms filtering SLAM in terms of accuracy, robustness and speed.

5.1 Laser SLAM

Research on 2D laser SLAM has been decreasing since 2006, mainly because of the success of algorithms like FastSLAM, lack of 3D information, and the significant progress of Visual SLAM. However, filtering based laser SLAM is mainly designed for 2D motions in structured environment. It is still impractical to apply it on UAV. New algorithms adopts the frontend-backend structure in Figure 1. Researchers concentrate on the front-end, while the back-end is kind of standardized as pose graph optimization. ICP [16] is a mature choice of front end, while it suffers from the high computational cost. Polar Scan Matching (PSM) utilizes the polar coordinates to perform scan matching. Normal Distribution Transform (NDT) aligns laser scans to a mixture of normal distributions.

Proposed by Kohlbrecher et al. in 2011 [17], HectorSLAM is currently one of the most competitive 2D laser algorithms. The front-end scan matching is achieved by aligning the laser scan with the map learned so far, using the Gauss-Newton approach. To improve convergence, pyramid like multi-resolution map is implemented with a coarse-to-fine matching scheme. Kohlbrecher et al. showed that their front-end was so accurate and robust that the back-end post-graph optimization could be neglected in most situations. To estimate the complete 6 DOF motion, a EKF is used to fuse information from sensors like IMU, with the 2D laser SLAM.

5.2 Featured Based Visual SLAM

Representing the environment with landmarks or features has been a popular choice since the 1980s when researchers began working on SLAM. Feature based map is compact, so that the requirement for computational power is significantly reduces. Also, both salient image patches and modern feature detectors are robust, and data association can be done efficiently with features. In the age of analysis based SLAM, feature detectors, including ORB [18–22], FAST [2], SURF [23], etc., were widely used in short-term tracking and loop closure.

The beginning of analysis SLAM is marked with the publish of PTAM [2], who first proposed to replace the filtering framework with a tracking thread and a mapping thread. PTAM maintains a global map composed of 3D points and the corresponding salient image patches. The tracking thread consists of data association and motion only BA. Data association, i.e. associating patches on a new frame with 3D points in map, is done by a coarse-to-fine search. After data association, motion only BA is conducted to refine the previously predicted motion. Simultaneously, the mapping thread utilizes the tracking result, to triangulate unmatched features into 3D points, which will be inserted into the global map. Then the local BA is applied to refine both the tracking result and the global map. The experiments showed that PTAM significantly outperforms the MonoSLAM. Ever since PTAM, most Visual SLAM algorithms adopted similar structure of separated tracking and mapping.

Strasdat et al. [24] presented a monocular SLAM algorithm where motion only BA was used for tracking. Within the keyframe framework, [24] proposed three dimensional information filters to initialize the 3D feature points. Loop closure were conducted with 7 DOF similarity transformation, where the scales of keyframes were considered. In [25],

http://www.imavs.org/

http://www.imavs.org/

Strasdat et al. proposed the idea of covisibility graph, and tracking within a local map.

With stereo cameras, the depth of feature points can be triangulated, which greatly simplifies the tracking and mapping process. In libviso2 [23], SURF feature points are extracted with triangulated depth. Efficient feature point matching is achieved with coarse-to-fine searching. Visual odometry is done by minimizing the projection error of 3D feature points from the previous time instants into the current left and right frames. With the motion estimated by visual odometry, dense map construction is conducted with the ELAS stereo matching. With similar strategy, libfovis [26] achieved real-time SLAM on UAV with RGBD cameras, where loop closure was implemented using Calonder randomized tree descriptors and RANSAC.

In 2015, the paper [18] and source code of ORB-SLAM were published, and is believed to be the best feature based SLAM so far. ORB-SLAM is a complete system with functions of tracking, mapping, loop closure, relocalization from tracking lost. The ORB features, which can be extracted in milliseconds, are used throughout the system in all functions. Tracking is initialized from previous frame or global relocalization, and then refine with motion only BA in a local map extracted from the covisibility graph. Local mapping is done by inserting keyframes, optimizing using local BA, and culling keyframes. Loop closure candidates are searched using the DBoW2 approach, which is also used for relocalization from tracking lost. Similarity transformations between current frame and loop candidates are computed similar to the method of [24], and then fused into the covisibility graph and essential graph. Global map optimization is achieved over the essential graph. The ORB-SLAM has shown impressive performance with monocular and depth cameras.

5.3 Dense Direct Visual SLAM

Despite that feature based methods have proved themselves to be effective, they are faced with two serious problems. The first is that, the features are far too sparse to represent the environment, which hinders further applications with the environment, including semantic understanding, object recognition, human interaction, etc. Secondly, they make use of only the information of the features and discard all other pixels of the images. In 2011, dense methods were proposed to take advantages of all pixels, and build dense 3D map.

One of the pioneers is the Dense Tracking and Mapping (DTAM) by Newcombe et al [27], which achieved real-time performance with GPU support. Dense reconstruction is implemented between any keyframe and many other nearby frames. A regularized cost function is constructed with regard to the inverse depth, in order to consider both photometric error and the smoothness of the estimated depth. By introducing a coupling term, the smoothness part is optimized in a similar way of ROF image denoising, while the photometric error part can be optimized simply using exhaustive search. Paralleled with dense mapping, each RGB frame is tracked against the densely built map by minimizing the photometric error, in order to solve for the Lie algebra $se(3)$ that defines the 6 DOF motion.

The vision group from Technical University of Munich (TUM) published a series of paper to demonstrate their dense direct SLAM system. Kerl et al. [28] utilized the depth from RGBD camera, and used a similar optimization scheme with [27] to achieve odometry in 30Hz with CPU. In 2014, the Large Scale Direct Monocular SLAM (LSD-SLAM) was published by Engel et al. [29], and demonstrated impressive capacity of densely reconstructing the environment accurately. At the back-end, the map is continuously improved with pose-graph optimization, and loop closure is done with FAB-MAP [30]. Later, the stereo [31] and fisheye [32] extension of LSD-SLAM were released to make use of the known depth, or the wide view angle.

KinectFusion by Newcombe et al. [33] is another milestone for dense SLAM, though it is only real-time with GPU support. Different from previous Visual SLAM where map is represented with point clouds or grids, KinectFusion maintains a volumetric, truncated signed distance function (TSDF) representation. Tracking is conducted with pyramid ICP.

5.4 Semi-dense Methods

In between the feature based and direct SLAM, some researchers tried to take advantages of both methods. Semi-direct Visual Odometry (SVO) is a popular visual odometry algorithm put forward by Forster et al. [34]. The map is represented by 3D locations of FAST feature points, and the 4×4 image patches of the features. Whenever a keyframe is created, FAST features are extracted and inserted into a depth filter, which is in charge of updating depth and evaluating the uncertainty. In the tracking thread, there are 3 steps. The first is to initialize motion against the previous frame by minimizing the photometric error. In the second step, for each projected feature in the current frame, its location is refined by minimizing the photometric error between its patch and the closest keyframe's feature patch. In the third step, motion only BA or local BA is applied to refine the pose of the current frame. Because of the semi-direct scheme, SVO is so efficient that it can be implemented on embedded systems.

6 CONCLUSION

Over decades of development, 2D SLAM methods with laser scanners are considered mature. Modern algorithms like HectorSLAM [17] are adequately robust for UAV applications. Visual SLAM on UAV is still an open problem, mostly because algorithms are fragile to UAV's agile motion and the unknown uncertainty of the environment. Recent algorithms including ORB-SLAM and LSD-SLAM, are potential candidates that can be applied on UAV.

REFERENCES

[1] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian D Reid, and John J Leonard. Simultaneous

localization and mapping: Present, future, and the robust-perception age. *arXiv preprint arXiv:1606.05830*, 2016.

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

[3] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[4] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[5] Gamini Dissanayake, Shoudong Huang, Zhan Wang, and Ravindra Ranasinghe. A review of recent developments in simultaneous localization and mapping. In *2011 6th International Conference on Industrial and Information Systems*, pages 477–482. IEEE, 2011.

[6] Josep Aulinas, Yvan R Petillot, Joaquim Salvi, and Xavier Lladó. The slam problem: a survey. In *CCIA*, pages 363–371. Citeseer, 2008.

[7] Hauke Strasdat, José MM Montiel, and Andrew J Davison. Visual slam: why filter? *Image and Vision Computing*, 30(2):65–77, 2012.

[8] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Aaai/iaai*, pages 593–598, 2002.

[9] MWM Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on robotics and automation*, 17(3):229–241, 2001.

[10] Juan Nieto, Tim Bailey, and Eduardo Nebot. Scan-slam: Combining ekf-slam and scan correlation. In *Field and service robotics*, pages 167–178. Springer, 2006.

[11] Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18(3):249–275, 1997.

[12] Michael Montemerlo and Sebastian Thrun. Simultaneous localization and mapping with unknown data association using fastslam. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1985–1991. IEEE, 2003.

[13] José Neira and Juan D Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on robotics and automation*, 17(6):890–897, 2001.

[14] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[15] Andrew J Davison and David W Murray. Simultaneous localization and map-building using active vision. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):865–880, 2002.

[16] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994.

[17] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 155–160. IEEE, 2011.

[18] Raul Mur-Artal, JMM Montiel, and Juan D Tardós. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[19] Raúl Mur-Artal and Juan D Tardós. Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. *Proceedings of Robotics: Science and Systems, Rome, Italy*, 1, 2015.

[20] Raúl Mur-Artal and Juan D Tardós. Orb-slam: Tracking and mapping recognizable features,. In *MVIGRO Workshop at Robotics Science and Systems (RSS), Berkeley, USA*, 2014.

[21] Raúl Mur-Artal and Juan D Tardós. Fast relocalisation and loop closing in keyframe-based slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 846–853. IEEE, 2014.

[22] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[23] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968. IEEE, 2011.

[24] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*, 2010.

[25] Hauke Strasdat, Andrew J Davison, JMM Montiel, and Kurt Konolige. Double window optimisation for constant time visual slam. In *2011 International Conference on Computer Vision*, pages 2352–2359. IEEE, 2011.

[26] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *International Symposium on Robotics Research (ISRR)*, volume 2, 2011.

[27] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[28] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Robust odometry estimation for rgb-d cameras. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3748–3754. IEEE, 2013.

[29] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014.

[30] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[31] J. Engel, J. Stueckler, and D. Cremers. Large-scale direct slam with stereo cameras. In *International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[32] D. Caruso, J. Engel, and D. Cremers. Large-scale direct slam for omnidirectional cameras. In *International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[33] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[34] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22. IEEE, 2014.

http://www.imavs.org/

Relative Localization for Quadcopters using Ultra-wideband Sensors

Kexin Guo, Zhirong Qiu, Wei Meng, Thien Minh Nguyen and Lihua Xie

http://www.imavs.org/

ABSTRACT

In this paper a relative localization (RL) system for quadcopters based on ultra-wideband (UWB) ranging measurements is proposed. To achieve the relative localization, UWB modules are installed on the quadcopters to actively measure distances and exchange data package to a hovering quadcopter which is equipped with an identical UWB module. Since instantaneous distance-only measurements cannot provide enough information for relative position estimate, a combination of nonlinear and linear trajectory is ultilized to fulfill the RL in this paper. To reduce the estimate error, the heading in the linear path phase is designed and chosen online. This intial relative position estimate produced by the algorithm will be further fed to the flight control loop to aid the navigation of the quadcopters. Flight tests have been conducted to validate the performance of UWB based relative localization algorithm.

1 INTRODUCTION

Micro unmanned aerial vehicles (UAVs) are promising to play more and more important roles in both civilian and military activities. Currently, the navigation of UAVs is critically dependent on the localization service provided by the Global Positioning System (GPS), which suffers from the multipath effect [1] and blockage of line-of-sight, and fails to work in an indoor, forest or urban environment [2].

In certain environments such as forest and multi-functional office, even though UAV can execute self-localization and navigation through LiDAR sensor, camera, etc. However, for a long-time flight, the percent of position drift from simultaneous localization and mapping (SLAM) system can degrade significantly so that it can no longer support autonomous flight. Theoretically, when UAVs are close to each other, the localization solution for UAVs can be improved by using relative position estimates. Hence, a relative localization system can be useful to aid the navigation for UAVs in a blind environment.

The authors Kexin Guo (GUOK0005@e.ntu.edu.sg), Zhirong Qiu (QIUZ0005@e.ntu.edu.sg), Thien Minh Nguyen (E150040@e.ntu.edu.sg) and Lihua Xie (elhxie@ntu.edu.sg) are with School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798. The author Wei Meng (tslmeng@nus.edu.sg) is with National University of Singapore, Singapore 117411.

The study on the cooperative localization can be dated back to as early as the 90s of last century, when in [3] a repeated move-and-stop actions were employed with static robots serving as landmarks to localize the moving robots. A huge body of works has flourished after 2000, and can be roughly classified into two categories [4, 5] as based on whether fusion or optimization method is used.

The majority of cooperative localization literature is dedicated to fusion method, either by particle filter or by extended Kalman Filter. By assuming that the location of each robot is independent from the others', [6] applied the method of non-parametric belief propagation to marginalize the joint distribution as the estimate of each agent. By carefully examining the timing of marginalization in a Markov network, [7] was able to account for the cross correlation with a delayed estimate for each robot in the team. Later it was extended in [8] in a more convenient information filter form, with a central processing center dealing with the relative measurements. The fusion by covariance intersection method is considered in [9] and [4]. Based on discrete-time range measurements from an agent to a source, [10] addresses the position estimation problem of a drifting source relative to an agent in three-dimension by designing a Kalman filter with globally exponentially stable error.

The other works are based on the optimization. Assuming that in each time step the motions and observations are independent among different robots, [11] obtained the estimate by solving a MLE problem. For the distributed implementation, it suggested an optimum seeking method by fixing the neighbor's position, and then exchanging the suboptimal estimate for another round of optimization. [12] considered a MAP formulation, and achieved the cooperative localization by distributed conjugate gradient method. When the measurements are relative positions, Jacobian algorithm was used to solve the distributed localization in [13]. Moving horizon estimation (MHE) and convex optimization were applied to perform multirobot localization with constraints and unknown initial poses in [5].

Compared with the global and relative position information, range measurement is relatively easy to obtain among multi-UAVs. In this case, recently many researchers focus on range based relative localization [14, 10]. Typically the distance from a moving UAV to its neighbors can be estimated from RF signals. Among different types of RF ranging techniques, UWB ranging technology is robust to multipath and non-line-of-sight (NLOS) effects, and can achieve a centimeter level ranging error. The ultra wide bandwidth enables

http://www.imavs.org/

UWB modules to avoid the interface with other types of RF signals such as WiFi and remote control signals in the flight environment. Besides, the ultra-short duration pulses can be exploited to simultaneously obtain precise range and data information, which provides extra information from neighbors for relative localization.

In this paper, we establish a relative localization system for quadcopters based on UWB range measurements. To achieve the relative localization, an UWB module is installed on the quadcopter to actively send ranging requests and data package to neighboring UAVs. From a practical and computation point of view, only a part of distance and neighbor's height measurements go through calibration and relative localization algorithm. Flight tests with three UAVs have been conducted to evaluate our developed relative localization algorithms.

In the following section, we will formulate the relative localization (RL) problem. In developing a solution to the problem, we begin to examine how to estimate the relative position with smaller errors in a dynamic agent group and propose an UWB based relative localization algorithm (see section 2). After that, the demonstration of the proposed relative localization for 3 UAVs in a woods near environment is presented in section 3. We conclude our work in section 4.

2 UWB BASED RELATIVE LOCALIZATION

2.1 Problem Formulation

For each UAV in the team, relative localization aims to determine the position of the neighboring UAV relative to itself. If the bearing and distance of the neighbor can be obtained, then the relative localization is readily achieved. However, the relative bearing is quite difficult to achieve when the neighbor is located outside the operational range of camera. On the other hand, the distance measurement can still be stably achieved with the help of UWB. In this case, a possible method is to combine the distance measurements and the motion displacements for estimating the relative location. As shown in the Figure 1(a), for the two UAVs P and Q, relative localization aims to estimate the vector $\overrightarrow{P_iQ_i}$ from the displacements $\overrightarrow{\delta P_i}$ and $\overrightarrow{\delta Q_i}$, as well as the distances d_i, $i = 1, \ldots, n$. In this paper, we shall focus on the case when one of the UAVs is static, say Q in Figure 1(b). Specifically, we shall first discuss the proposed localization method under the assumption that the displacement $\overrightarrow{\delta P_i}$ can be accurately measured, and then revise the corresponding method to include the error in $\overrightarrow{\delta P_i}$.

2.2 Direct Estimation by Trilateration

2.2.1 Algorithm

Denote $\overrightarrow{QP_0} = (x, y)$ and $\overrightarrow{P_0P_i} = (\Delta x_i, \Delta y_i)$, then we have

$$d_i = \sqrt{(x + \Delta x_i)^2 + (y + \Delta y_i)^2}, i = 0, 1, \ldots, n. \quad (1)$$

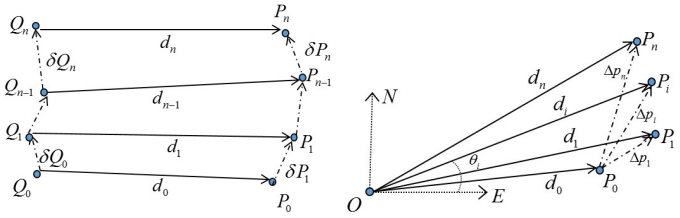


Figure 1: Relative localization for two UAVs

In case of ranging error of d_i's, we can estimate (x, y) by solving the nonlinear least square problem

$$\begin{aligned} &\min \sum_{i=0}^{n} (d_i - \sqrt{(x + \Delta x_i)^2 + (y + \Delta y_i)^2}) \\ &= \min \sum_{i=0}^{n} (d_i - f_i(x, y)). \end{aligned} \quad (2)$$

The solution to the above problem is usually achieved by the Gauss-Newton method, which is based on the linear approximation of f_i around the current estimate $(\hat{x}, \hat{y})$ as

$$\begin{aligned} f_i(x, y) \approx\ & f_i(\hat{x}, \hat{y}) \\ & + \frac{1}{f_i(\hat{x}, \hat{y})} (\hat{x} + \Delta x_i, \hat{y} + \Delta y_i) \cdot (x - \hat{x}, y - \hat{y}). \end{aligned} \quad (3)$$

Replacing the approximation (3) in (2), problem (2) can be transformed into a linear least square problem whose normal equation is given by

$$\begin{pmatrix} d_0 - \hat{f}_0 \\ d_1 - \hat{f}_1 \\ \vdots \\ d_n - \hat{f}_n \end{pmatrix} = \begin{pmatrix} \hat{x}/\hat{f}_0 & \hat{y}/\hat{f}_0 \\ \hat{x}_1/\hat{f}_1 & \hat{y}_1/\hat{f}_1 \\ \vdots & \vdots \\ \hat{x}_n/\hat{f}_n & \hat{y}_n/\hat{f}_n \end{pmatrix} \begin{pmatrix} x - \hat{x} \\ y - \hat{y} \end{pmatrix}, \quad (4)$$

or $\Delta d = J \Delta p$ where $\hat{x}_i = \hat{x} + \Delta x_i$ and $\hat{y}_i = \hat{y} + \Delta y_i$. Now we have

$$\Delta p = \begin{pmatrix} x - \hat{x} \\ y - \hat{y} \end{pmatrix} = (J'J)^{-1} J' \Delta d \quad (5)$$

The newly obtained (x, y) can be treated as the new estimate to update J and Δd for a new normal equation. The iteration will stop until $|x - \hat{x}| < 0.01$ and $|y - \hat{y}| < 0.01$ and the problem (2) is solved.

However, we still need an initial guess to solve (2), which comes from the solution of a related least square problem dealing with the squared distances of (1). Actually, squaring both sides of (1) and subtracting the first equation $d_0^2 =$

$x^2 + y^2$ from the other n equations, we get the following linear system

$$2\begin{bmatrix}\Delta x_1 & \Delta y_1\\ \Delta x_2 & \Delta y_2\\ \vdots & \vdots\\ \Delta x_n & \Delta y_n\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix} = \begin{bmatrix}d_1^2 - d_0^2 - (\Delta x_1^2 + \Delta y_1^2)\\ d_2^2 - d_0^2 - (\Delta x_2^2 + \Delta y_2^2)\\ \vdots\\ d_n^2 - d_0^2 - (\Delta x_n^2 + \Delta y_n^2)\end{bmatrix} \quad (6)$$

or $2A\begin{bmatrix}x\\ y\end{bmatrix} = D$. Now an initial guess of $\overrightarrow{QP_0}$ can be given by

$$\begin{bmatrix}x\\ y\end{bmatrix} = \frac{1}{2}(A'A)^{-1}A'D. \quad (7)$$

Remark 2.1. *Although the equation* (6) *is able to give an exact solution when there is no ranging error, the introduction of squared distances would increase the error of calculation and only achieves a rough estimate. In fact, if the range measurement d is a Gaussian random variable with distribution $N(\bar{d}, \sigma^2)$, then the variance of d^2 is given by $4\bar{d}^2\sigma^2 + 2\sigma^4$ dependent on $\bar{d}^2$. On the other hand,* (4) *employs the original measurements as the input and a rough initial guess is able to guarantee the convergent solution in a few iterations. Hence we employ the rough estimate from* (6) *as the initial guess of* (2) *and obtain the estimate from* (4)*.*

2.2.2 Error Analysis

From (5) the error covariance of the solution of (2) can be approximated by $E[\Delta p' \Delta p] = \sigma^2(J'J)^{-1}$, where $E[\cdot]$ denotes the expectation of $\cdot$, if the ranging error is assumed to be independent from each other and has a variation of σ^2. Noticing that each row of J is a normalized vector, we can rewrite J as

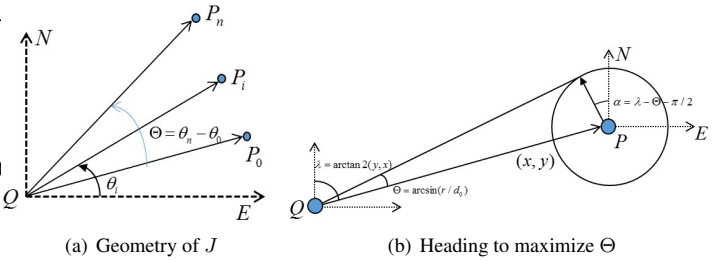
$$J = \begin{bmatrix}\cos\theta_0 & \sin\theta_0\\ \cos\theta_1 & \sin\theta_1\\ \vdots & \vdots\\ \cos\theta_n & \sin\theta_n\end{bmatrix} = \begin{bmatrix}c & s\end{bmatrix} \quad (8)$$

and obtain $(J'J)^{-1}$ as

$$(J'J)^{-1} = \frac{1}{\det(J'J)}\begin{bmatrix}s's & -c's\\ -c's & c'c\end{bmatrix}.$$

Now the mean squared error is given by the trace of $\sigma^2(J'J)^{-1}$ as

$$MSE = \frac{\sigma^2(n+1)}{\det(J'J)} \quad (9)$$



(a) Geometry of J (b) Heading to maximize Θ

Figure 2: Heading diagram

where $n+1$ is the sample size. In the meanwhile, direct computation shows that

$$\begin{aligned}\det(J'J) &= \sum_{i\neq j}\cos^2\theta_i \sin^2\theta_j - \sum_{i\neq j}\cos\theta_i\sin\theta_i\cos\theta_j\sin\theta_j\\ &= \sum_{i\neq j}(\cos^2\theta_i\sin^2\theta_j - \cos\theta_i\sin\theta_i\cos\theta_j\sin\theta_j)\\ &= \sum_{i\neq j}(\cos\theta_i\sin\theta_j - \cos\theta_j\sin\theta_i)^2\\ &= 2\sum_{0\le i<j\le n}\sin^2(\theta_i - \theta_j),\end{aligned} \quad (10)$$

where each addend is the area spanned by the unit vectors $[\cos\theta_i \ \ \sin\theta_i]$ and $[\cos\theta_j \ \ \sin\theta_j]$. If we assume that the UAV can only obtain the accurate displacement over a short distance r and $r << ||\overrightarrow{QP_0}||$ (namely $(\Delta x_i)^2 + (\Delta y_i)^2 << x^2 + y^2$ for each i), then $\det(J'J) \approx 2\sum_{0\le i<j\le n}(\theta_i - \theta_j)^2$. Now the MSE is only dependent on the angles spanned by the corresponding unit vectors, and aligning each P_i along a straight line would not affect the localization accuracy. Therefore, we may let the UAV fly along a straight line during the relative localization. Moreover, it's reasonable to assume that each small increment of θ is equal, or $\theta_{i+1} - \theta_i = \omega$. In this case, $\det(J'J) \approx 2\sum_{0\le i<j\le n}(i-j)^2\omega^2 = (\frac{n^4}{6} + \frac{2n^3}{3} + \frac{5n^2}{6} + \frac{n}{3})\omega^2$ and

$$MSE \approx \frac{6\sigma^2}{(n^4 + 4n^3 + 5n^2 + 2n)\omega^2} = \frac{6\sigma^2}{(n^2 + 4n + 5 + \frac{2}{n})\Theta^2}$$

where $\Theta = \theta_n - \theta_0$. We can see that a major reduction of the mean squared error can be from an increase of Θ. In light of this observation, the UAV is to move along the line which maximizes Θ, or the angle $\angle P_0QP_n$ as in Figure 2(a). We can see that the maximum $\Theta = \arcsin(r/d_0)$ if the UAV is to move a short distance of r for the relative localization as shown in Figure 2(b). Also, the heading in the North-East frame can be found as $\alpha = \lambda - \Theta - \frac{\pi}{2}$, where $\lambda = \arctan 2(y, x)$.

http://www.imavs.org/

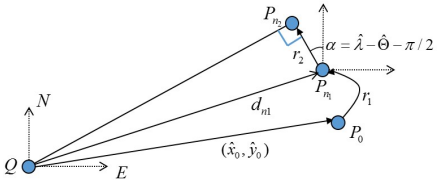


Figure 3: Two phases in the relative localization

2.2.3 Application to Relative Localization

Now we can apply the error analysis to the relative localization as follows. Actually, a linear path is not enough to uniquely determine the vector $\overrightarrow{QP_0}$ due to the ambiguity, while a nonlinear path nails it. Therefore, we propose the relative localization as a combination of a nonlinear path and a linear path. To be specific, assume that the UAV is to fulfill the relative localization in a short distance r and it first moves along some nonlinear path of fixed length r_1. After the nonlinear phase, we can actually get a rough estimate of (x_0, y_0) as $(\hat{x}_0, \hat{y}_0)$, as well as that of $\overrightarrow{QP_{n_1}}$ as $(\hat{x}_{n_1}, \hat{y}_{n_1})$. As suggested from the last subsection, we need to move along some direction by a length of $r_2 = r - r_1$ such that $\angle P_0 Q P_n$ reaches the maximum. Obviously the heading should be chosen as $\alpha = \hat{\lambda} - \Theta - \frac{\pi}{2}$, where $\hat{\lambda} = \arctan 2(\hat{y}_{n_1}, \hat{x}_{n_1})$, $\Theta = \arcsin(r_2/d_{n_1})$. These two phases are also depicted in Figure 3.

2.3 Inclusion of Displacement Error in the Algorithm

This section evaluates the performance of the proposed relative localization algorithms considering that both the distance measurements and the odometry estimates are subject to noise. Specifically we assume the range measurements of each UAV are subject to independent zero-mean Gaussian noise and the odometry estimates are subject to two independent sources of noise.

To apply the direct estimation method, we still need to account for the displacement error, which can be solved by the following EKF. In fact, the system model can be simply modeled as follows:

$$\begin{aligned} x(k+1) &= x(k) + \Delta x(k) + \zeta_x(k), \\ y(k+1) &= y(k) + \Delta y(k) + \zeta_y(k), \\ d(k) &= \sqrt{x^2(k) + y^2(k)} + \eta(k), \end{aligned} \tag{11}$$

where $(x(k), y(k))$ denotes the relative position of the moving UAV at the time step k, $(\Delta x(k), \Delta y(k))$ is the displacement, $d(k)$ is the UWB ranging, and $\zeta_x(k), \zeta_y(k), \eta(k)$ are the corresponding noises. With the initial estimate given by the direct trilateration, we can apply the EKF for relative localization.

With Figure 3, we summarize the relative localization of the moving UAV P from the static UAV Q as below:

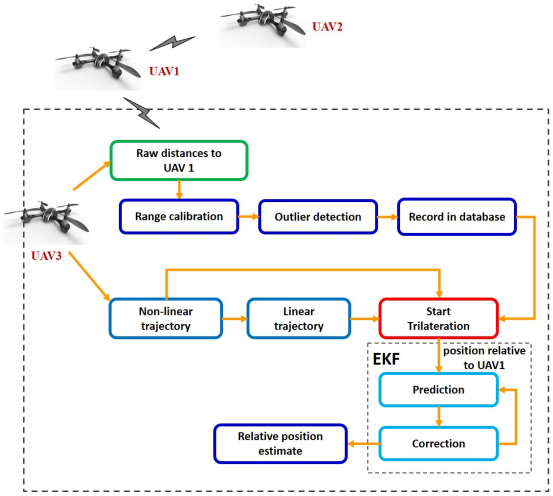


Figure 4: UWB-based relative localization workflow

I (1) P moves along a nonlinear path of fixed length r_1 to the point P_{n_1} and collect the distance measurements d_i and displacement $(\Delta x_i, \Delta y_i)$ in the NE frame, $i = 0, \ldots, n_1$.
(2) Obtain estimate of $\overrightarrow{QP_{n_1}}$ as $(\hat{x}_{n_1}, \hat{y}_{n_1})$ by trilateration algorithm described in section 2.2.1. Choose heading in the next phase by $\alpha = \hat{\lambda} - \Theta - \frac{\pi}{2}$, where $\hat{\lambda} = \arctan 2(\hat{y}_{n_1}, \hat{x}_{n_1})$, $\Theta = \arcsin(r_2/d_{n_1})$ and r_2 is the flight distance in the next phase.

II (1) With the heading of α, P moves along a linear path of fixed length r_2 to the point P_n and collect the distance measurements d_i and displacement $(\Delta x_i, \Delta y_i)$ in the NE frame, $i = 0, \ldots, n$.
(2) Obtain estimate of $\overrightarrow{QP_0}$ as $(\hat{x}_0, \hat{y}_0)$ by trilateration algorithm described in section 2.2.1, as well as the error covariance matrix $M = \sigma^2 (J'J)^{-1}$.

III The initial guess $(\hat{x}_0, \hat{y}_0)$ and M will be used to initialize the EKF based on the model (11). d_i and $(\Delta x_i, \Delta y_i)$ are used to generate the final estimate of $\overrightarrow{QP_n}$.

3 FLIGHT TESTS WITH QUADCOPTERS

To validate the proposed UWB-based relative localization approach, flight tests of 3 UAVs have been conducted in like-forest environments. The detailed hardware configuration is depicted in [15]. Below, we shall discuss the experiments in details.

3.1 UWB based Relative Localization Workflow

Figure 4 illustrates the workflow of the UWB based relative localization system. To achieve the range-based rela-

http://www.imavs.org/

http://www.imavs.org/

tive localization, all of the UAVs are equipped with the same UWB module. As the two-way time of flight ranging method described in [15], the UWB module on a hovering quadcopter actively sends ranging requests to neighboring UAVs for distance measurement and communication. Once a distance is obtained by UAV2 and UAV3, it will be calibrated by linear regression where the calibration parameters are determined by a series of experiments in different environments[15]. The calibrated range then goes through the outlier detection before it is stored in the database. To reduce the computation and avoid excessive repetition of similar data, only the selected distance measurements and neighbor's information are recorded and this selection depends on the movement of each UAV. Note that trilateration is applied to trigger the algorithm and its output serves as the initial state estimate for EKF. The follow-up localization is sustained by EKF in a recursive way.

Each part of our UWB based relative localization system is elaborated in section 2.2 and 2.3. Path direction during the linear phase is analyzed in section 2.2.3

3.2 UWB based Relative Localization for 3 UAVs Flight

The location of our test site was closen at Dover in Singapore with a area of 30 $\times$ 40 m. To validate the performance of our proposed relative localization, the ground truth reference for calibration is necessary. Since that, a baseline, which is parallel to a ditch with 233 degrees with respect to the north, was selected as our reference. The UAVs are aimed to navigate to or hover over the relative targets we set preliminarily (yellow flags on the ground in Figure 7) and this performance will demonstrate the availability of UWB based relative localization. Some experimental details will be introduced as following.

UAV1 was set in the middle of UAV2 and UAV3 initially with 20 meters away from them as shown in Figure 5. Once UAVs took off, UAV1 hovered over the start point and UAV2 and UAV3 flied along a preset non-linear trajectory to collect distance data and generated initial relative position estimates, namely phase I. During phase II, UAV2 and UAV3 flied as a linear trajectory to improve the accuracy of the relative position estimates by using the proposed trilateration algorithm. Then in phase III, UAV2 and UAV3 flied directly to their own desired targets (the red stars in Figure 5) set in relative coordinates and the video shot during 3 UAVs flight is shown in Figure 6. Finally all of these 3 UAVs hovered over their own targets with reference to the landmarks (yellow flags on the ground in Figure 7) measured by tape measure before flight test. It can be seen from Figure 7, once accomplishing the entire phases UAV2 and UAV3 hovered over their own landmarks with the position error less than 1 meter (Figure 8).

4 CONCLUSION AND FUTURE WORK

The relative localization strategy proposed in this paper utilizes UWB radios to estimate the relative position of moving UAVs to a static one over a short distance. The error analysis of the trilateration method is employed to reduce

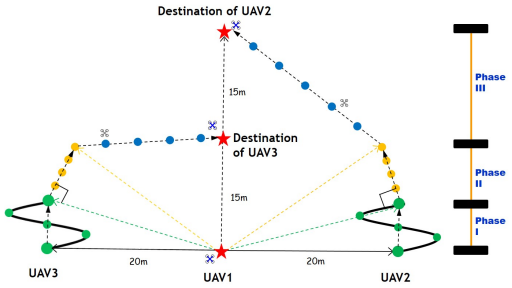


Figure 5: Demo scenario at Dover



Figure 6: Flight video shot of 3 UAVs at Dover.



Figure 7: Performance of the relative localization for 3 UAVs.

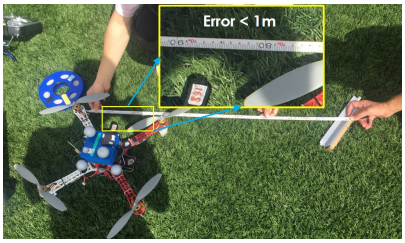


Figure 8: Accuracy of the relative localization measured after landing.

the localization error. Basically, the UAV is first to move along some nonlinear path for a rough estimate, based on which a heading will be chosen for the UAV to move along a straight line aiming to improve the localization error. The rangings and displacements over the course are first fed to the nonlinear least squares algorithm for an initial guess, which serves to initialize the EKF. The proposed trilateration algorithm and EKF cooperate to achieve highly precise relative position estimate for autonomous flight of the quadcopters. Flight tests have been conducted to validate the performance of the proposed UWB-based relative localization, and the results demonstrate its capability of providing accurate relative position for the quadcopters.

Future works will be steered to the case of relative localization when all the three UAVs are moving simultaneously. Integrated with certain self-localization algorithms, a more robust and accurate localization technique for long term flight in GPS denied environments can be investigated. Besides, UWB based UAV swarming may be explored and demonstrated in the future.

http://www.imavs.org/

ACKNOWLEDGMENT

We would like to thank Mr. Abdul Hanif Zaini, Mr. Bing Cheng Yeap and Mr. Dong Wei for their assistance in numerous flight tests. We also thank Dr. Rodney Teo for the technical support.

REFERENCES

[1] P. Scerri, R. Glinton, S. Owens, D. Scerri, and K. Sycara, "Geolocation of rf emitters by many uavs," in *AIAA Infotech@ Aerospace 2007 Conference and Exhibit*, 2007, p. 2858.

[2] T. Samad, J. S. Bay, and D. Godbole, "Network-centric systems for military operations in urban terrain: the role of uavs," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 92–107, 2007.

[3] R. Y. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994, pp. 1250–1257.

[4] L. C. Carrillo-Arce, E. D. Nerurkar, J. L. Gordillo, S. Roumeliotis, *et al.*, "Decentralized multi-robot cooperative localization using covariance intersection," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1412–1417.

[5] S. Wang, D. Gu, L. Chen, and H. Hu, "Single beacon-based localization with constraints and unknown initial poses," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 4, pp. 2229–2241, 2016.

[6] J. Schiff, E. B. Sudderth, and K. Goldberg, "Nonparametric belief propagation for distributed tracking of robot networks with noisy inter-distance measurements," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 1369–1376.

[7] K. Y. Leung, T. D. Barfoot, and H. H. Liu, "Decentralized localization of sparsely-communicating robot networks: A centralized-equivalent approach," *Robotics, IEEE Transactions on*, vol. 26, no. 1, pp. 62–77, 2010.

[8] H. Mu, T. Bailey, P. Thompson, and H. Durrant-Whyte, "Decentralised solutions to the cooperative multi-platform navigation problem," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 47, no. 2, pp. 1433–1449, 2011.

[9] H. Li and F. Nashashibi, "Cooperative multi-vehicle localization using split covariance intersection filter," *Intelligent Transportation Systems Magazine, IEEE*, vol. 5, no. 2, pp. 33–44, 2013.

[10] P. Batista, C. Silvestre, and P. Oliveira, "Ges source localization based on discrete-time position and single range measurements," in *Control & Automation (MED), 2013 21st Mediterranean Conference on*. IEEE, 2013, pp. 1248–1253.

[11] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Localization for mobile robot teams: A distributed mle approach," in *Experimental Robotics VIII*. Springer, 2003, pp. 146–155.

[12] E. D. Nerurkar, S. Roumeliotis, A. Martinelli, *et al.*, "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 1402–1409.

[13] P. Barooah, W. J. Russell, and J. P. Hespanha, "Approximate distributed kalman filtering for cooperative multi-agent localization," in *Distributed Computing in Sensor Systems*. Springer, 2010, pp. 102–115.

[14] A. Cornejo and R. Nagpal, "Distributed range-based relative localization of robot swarms," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 91–107.

[15] K. Guo, Z. Qiu, C. Miao, A. H. Zaini, C.-L. Chen, W. Meng, and L. Xie, "Ultra-wideband-based localization for quadcopter navigation," *Unmanned Systems*, vol. 4, no. 01, pp. 23–34, 2016.

Salient Object Detection Using UAVs

Mo Shan*, Feng Lin*, and Ben M. Chen§
*Temasek Laboratories, National University of Singapore
§Department of Electrical and Computer Engineering, National University of Singapore

ABSTRACT

A salient object detection approach is proposed in this paper, tailored to the aerial images collected by Unmanned Aerial Vehicles (UAVs). In particular, the aerial images are classified. The selected image is segmented using superpixels, and then a weak saliency map is constructed based on image priors. Positive and negative samples are selected in accordance with the weak saliency map for training a boosted classifier. Consequently, the classifier is used to produce a strong saliency map. The weak and strong saliency maps are integrated to locate the candidate objects, and false alarms are pruned after post-processing. Experiments on aerial images collected above meadow and roof demonstrate the effectiveness of the proposed approach.

Keywords: Saliency detection, UAVs

1 INTRODUCTION

UAVs have already been widely employed for a range of tasks, including inspection and surveillance. Furthermore, the advent of visual inertial odometry presented recently in [1, 2], which utilizes on-board camera and IMU in a tightly coupled manner, reduces their dependence on GPS dramatically and enables the UAVs to operate in clustered environment. For instance, researchers in the SFLY project have demonstrated that the UAVs are capable of performing localization and mapping in GPS-denied environment [3].

This paper studies salient object detection in aerial images. Specifically, illegal dumping detection is selected as a case-study. It refers to the unauthorized waste disposal of garbage, appliances or furniture upon public or private properties. A patrol team is usually required to deter such offensive activities. Meanwhile, the automatic illegal dumping detection from the images captured by UAVs is very critical to lower the burden of manual labor. Since the amount of images obtained from the on-board camera is significant, it is troublesome to scan through every image. Consequently, detection algorithms are necessary to select those images that may contain dumped objects. Normally, the waste is left at the places where it stands out from the environment, such as a plastic bag on the meadow, and this makes the dumped object to be salient. Therefore salient object detection becomes handy for detecting illegal dumping automatically. This problem may seem trivial at first glance, because the dumped waste is usually quite distinguishable from its environment. However, other man-made objects may exist in the scene as well, such as basketball court on a meadow, or windows on the roof. The existence of these outliers makes it more challenging to detect the garbage.

Saliency detection could be divided into three categories: bottom-up, top-down, and hybrid approaches [4]. The first category includes [5, 6, 7, 8, 9], and considers primitive information, such as the intensity contrast, color histogram, and global distribution of colors, whereas the second category contains [10, 11], and concerns about the application of prior knowledge for the specific task at hand. An example of bottom-up saliency detection is to detect a red ball against a while background, while top-down approach is used when the cyclists are searching for the bicycle lane [12].

In this paper, a task driven salient object detection algorithm is proposed. The images are pre-processed by scene classification. For the images that are likely to contain debris, a weak saliency map is generated from the input image based on prior knowledge, and then both positive and negative training samples are selected to produce a strong saliency map. Those two saliency maps are combined and thresholded to identify salient objects. Post-processing is carried out to prune outliers. The proposed approach is implemented in MATLAB based on the open-source code of [13] †. Aerial images collected by our UAV are used for the experiments.

The remainder of the paper is organized as follows: Section II presents a brief literature review; the methodology is described in Section III; Section IV contains the experiments for salient object detection; Section V consists of the conclusion.

2 RELATED WORKS

2.1 Bottom-up approaches

One of the early bottom-up models is presented in [5]. It is based on a biologically-plausible architecture related to the feature integration theory. The primitive visual features considered include intensity contrast, color contrast, and local orientation, all of which are computed at multiple scales. To integrate the feature maps with different modalities, a normalization operator is designed to promote the unique maps, mimicking the cortical lateral inhibition mechanisms. The feature maps are combined into conspicuity maps, which are normalized and summed to produce the saliency map. While

† http://202.118.75.4/lu/publications.html

http://www.imavs.org/

http://www.imavs.org/

this framework could be tailored to different tasks via dedicated feature maps, it may not be able to detect objects salient for feature types not implemented.

Another biologically-plausible bottom-up approach is proposed in [6]. The structure of graph algorithms is exploited to compute saliency efficiently. Specifically, Markov chains are defined for the images, and the equilibrium distribution is used as activation. Because each node is independent, this process can be computed in a parallel way. In the normalization phase, a graph is constructed to concentrate activation into key locations. This approach outperforms [5] by 14% on human fixation prediction, and it could be extended to multiresolutions for improved performance. Nevertheless, this work favors the central-bias and thus may not be suitable to detect salient objects in the image periphery.

The recent work [9] relies on a convolutional neural network (CNN) trained from ImageNet dataset to perform feature extraction. The features are extracted from three rectangular windows enclosing the target region, its surrounding regions, and the whole image, in order to evaluate the visual contrast. Fully connected layers are trained from these multiscale CNN features to infer the saliency score. The saliency maps are refined to maintain spatial coherence and fused to obtain an aggregated map. The drawbacks of this approach include its requirement of training datasets and long processing time if GPU is not used.

2.2 *Top-down approaches*

As for the top-down approach, [10] focuses on salient object detection by incorporating the high level concept. The problem is modeled by a condition random field (CRF) to combine multiscale contrast, center-surround histogram, and color spatial distribution as local, regional as well as global salient features. Temporal cues are also exploited for dealing with sequential images. Being only able to detect a single salient object is one of the remaining issues for this approach.

An alternative top-down model is described in [11]. The image is decomposed into multiscale segmentations. A random forest regressor is learnt to map the regional descriptors representing the contrast, property and backgroundness to a saliency score. An aggregated saliency map is obtained from fusing the saliency maps of different segmentation levels. The key differences of this approach is that it computes a contrast vector instead of a value, and it combines the features to generate the integrated saliency map, other than combining the saliency maps generated by varied features. This method requires the collection of training samples with groundtruth labels as well.

2.3 *Hybrid approach*

In addition to the two categories mentioned above, there are hybrid approaches as well. In [13], a novel salient object detection approach is proposed. Image priors and multiple features are exploited to generate positive and negative training samples for bootstrap learning. Since the training samples are selected using the bottom-up model, the off-line training and groundtruth labeling are alleviated. This is critical as there is a lack of datasets for illegal dumping detection. Our work is similar to [13] with the following contributions:

1. Saliency map is used to detect illegal dumping using aerial images captured by a UAV.
2. A simple yet efficient pre-processing algorithm is used for scene classification.
3. The color and size priors are used to take into account the features of the scene and the objects to be detected.
4. Steps for post-processing the saliency map to prune the outliers are proposed.

3 METHODOLOGY

Referring to the overview in Fig.1, the proposed approach consists of scene classification as pre-processing, generating a saliency map, which is thresholded to identify the candidates for the dumped waste, and post-processing to prune the false alarms.

3.1 *Pre-processing*

UAVs are often needed to survey a building to detect illegal dumping, and it is common that the building is surrounded by meadow and tree. Since the debris is most likely to exist on the meadow and roof, the aerial images are classified into three categories, namely meadow, roof, and tree images. The salient object detection is only carried out in the meadow and roof images. In this way, not only the number of false alarms is reduced, but also the computational time is saved by discarding the tree images.

In scene classification, green regions are detected first, and then blob detection of these regions is performed to count the cluster number. If there is few clusters, then the scene is probably the meadow. In contrast, the existence of more clusters indicate that the green regions are not connected, and hence the image is likely a tree image. If no green region is found, the probability of the image depicting the roof is high.

The green region detection is based on [14]. The input RGB image is transformed to grayscale image by

$$I = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B \qquad (1)$$

where R, G, B are the red, green, and blue channels of the input, I is the grayscale image. Then I is subtracted from G to obtain the green part of the image, followed by applying the median filter to suppress the noise. The resulting image is converted to a binary one by using the threshold of $thresh_{gr}$. Afterwards, blob detection is performed in the binary image, and the green regions whose area is smaller than $thresh_{ga}$ are discarded as noise. Suppose the number of the remaining blobs is k, then the scene is classified as meadow if $0 < k \leq thresh_k$, or tree if $k > thresh_k$. If $k = 0$, then the scene depicts roof. In the experiments, the parameters are set to $thresh_{gr} = 0.07$, $thresh_{ga} = 1000$, and $thresh_k = 3$.

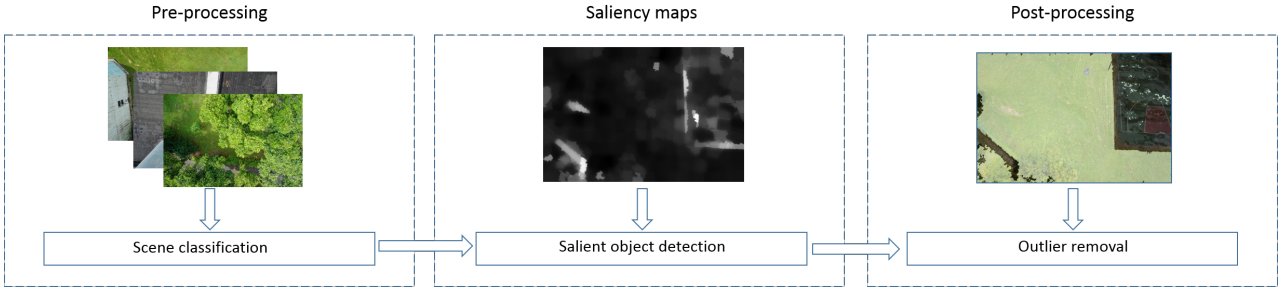


Figure 1: Overview of the proposed approach for salient object detection.

3.2 *Saliency detection*

A weak saliency map is constructed using color and size priors. As for the color prior, two cases are tested: meadow and roof, since it is common to find debris on the meadow, and it is also important to detect debris on the roof using UAVs because it is difficult for people due to out of view.

The color prior of a pixel p for the meadow is defined as

$$S_c(p) = \sqrt{(h(p) - H_m)^2 + (s(p) - S_m)^2 + (v(p) - V_m)^2} \quad (2)$$

where h, s, v are the Hue, Saturation and Value channels of the RGB image. H_m, S_m, V_m are the typical values for the meadow in the HSV colorspace, and $H_m = 0.2$, $S_m = 1$, $V_m = 0.5$. This color prior essentially computes the distance of the pixel to the green color. In this way, non-green objects are assigned with higher weights, and thus they are more likely to be chosen as positive training samples.

Similarly, the color prior for the roof is $S_c(p) = s(p)$, which is the value of the Saturation channel. Based on the observation that the Saturation for roof is low, the higher the Saturation, the more likely that it is not part of the roof, indicating the pixel belongs to objects such as debris.

The superpixels proposed in [15] are computed in the image to exploit the size prior, defined as $S_s(p) = 1 - A$, where A is the normalized region of the superpixel that pixel p belongs to. The size prior penalizes large area, because these regions often correspond to buildings, and are unlikely to be dumped objects.

Besides color and size priors, another useful criterion for determining salient objects is the difference from the image border, assuming that the majority of the border region contains only background. To compute the distance of each superpixel to those close to the boundary, the RGB, CIELab, and Local Binary Pattern (LBP) features are used. Therefore, the per-pixel saliency could be obtained from

$$S_{weak}(p) = S_c(p) \times S_s(p) \times d(p) \quad (3)$$

where $S_c(p), S_s(p), d(p)$ are the color prior, size prior, and the superpixel difference to the image border respectively. This weak saliency map is smoothed by Graph Cut.

Next, the weak saliency map is used to generate the training samples for the strong saliency map. The superpixels whose average saliency values are below the lower threshold is selected as negative training sample, whereas those with average saliency values exceeding the higher threshold are chosen as positive training samples.

Since how to integrate different features for training is unclear, Multiple Kernel Boosting (MKB) presented in [16] is employed. Several weak classifiers, which are Support Vector Machines (SVMs) with linear, polynomial, RBF, and sigmoid kernels, are combined to form a strong classifier. The details of the boosting process is described in [13]. The aggregated classifier is then used to generate a pixel-wise strong saliency map S_{strong}. The map is smoothed by Graph Cut and guided filter.

To address the multiscale issue, superpixels with four different granularities are generated, producing four pairs of weak and strong saliency maps. Averaging these maps gives the multiresolution weak and strong saliency maps.

To summarize, a weak saliency map is generated based on priors, and then a strong saliency map is constructed using the training samples. The former detects fine details due to its bottom-up model, while the latter focuses on global shapes. As the two maps are complementary, they are integrated to produce the final saliency map as

$$S_{aggregated} = \frac{S_{weak} + S_{strong}}{2} \quad (4)$$

The final saliency map $S_{aggregated}$ is thresholded by $thresh_{sa}$ to identify objects that possess a high saliency value.

3.3 *Post-processing*

The post-processing algorithm is described in Algorithm 1. The first step is to detect homogenous regions. This is necessary to exclude the false alarms existed on non-homogenous regions, such as the basketball court besides the meadow, or the sidewall on the roof.

The meadow is considered first. Green region detection as described in the pre-processing step is conducted. For the roof, the image is converted from RGB colorspace

http://www.imavs.org/

Algorithm 1 Post-processing algorithm

```
Detect homogenous region
if The scene is meadow then
    Detect green region
    thresh_gr ← 0.05
end if
if The scene is roof then
    Threshold the HSV image
    thresh_sl ← 0, thresh_su ← 0.1
end if
Retain the salient objects on the homogenous region
for Each salient object candidate do
    if Rectangularity < thresh_rt then
        Remove outlier
    end if
    if (Area < thresh_al or Area > thresh_au) then
        Remove outlier
    end if
    if (ECD < thresh_dl or ECD > thresh_du) then
        Remove outlier
    end if
end for
```

to HSV colorspace, and thresholded using pre-defined lower and upper thresholds for the Saturation channel, where $thresh_{sl} = 0$, $thresh_{su} = 0.1$, since the roof part has very low saturation. The holes in these binary maps are filled to produce masks for homogeneous regions.

Even though the salient object could be detected on the homogenous regions, there may still exist some outliers such as the boundaries of the meadow or the building. To remove these outliers, the rectangularity of the detected object could be taken into account, since the garbage usually consists of boxes or appliances that are rectangular. The rectangularity could be computed as the ratio of pixels belonging to the object to the total pixels in its bounding box. The objects whose rectangularity is lower than the threshold $thresh_{rt}$ are discarded.

Besides rectangularity, the size is also taken into account to prune the objects that are either too small, such as the shadow, or too large, such as the entire wall. The parameters for size include the area and the equivalent circular diameter (ECD), computed by $\sqrt{4\pi \times area}$, which specifies the diameter of a circle with the same area as the detected object. Their lower and upper thresholds are $thresh_{al}$, $thresh_{au}$, and $thresh_{dl}$, $thresh_{du}$ respectively.

4 EXPERIMENT

4.1 *Image collection*

The images used in the experiments are captured when the UAV surveys a building surrounded by a meadow, as shown in Fig. 2 [‡]. Black plastic bags and tree branches are placed on the meadow and the roof to simulate illegal dumping. The UAV operates at about 35m, and the height of the building is 18m. In other words, the height in the aerial images for garbage detection on meadow is about 35m, while the height for garbage detection on roof is about 17m.



Figure 2: Stitched map of the test site, generated by Pix4Dmapper. The locations of plastic bag and tree branches are marked in red.

The camera carried on-board is Sony A6000, with a focal length of 16mm. The original resolution of the image is 6000×4000, and to reduce the computational time, it is downsampled to 640×427.

4.2 *Scene classification*

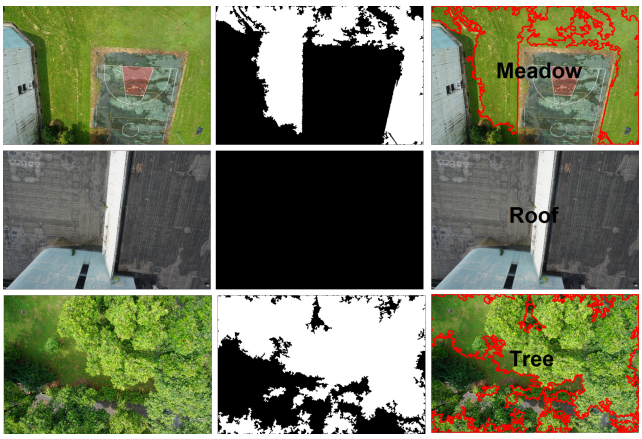


Figure 3: Scene classification results. From left to right: original image, binary image after green region detection, classification result with blobs marked in red.

The images are classified into different categories prior to salient object detection. From Fig. 3, it is evident that the

[‡]These images used for the experiments are available at https://github.com/shanmo/IMAV2016-Dataset

http://www.imavs.org/

green regions can be effectively detected in the images collect by the UAV. Moreover, the number of blobs in meadow image is smaller than that in the tree image, where the green regions tend to be discontinuous. Furthermore, there is no green region in the roof image. With the help of the scene classification, illegal dumping detection will only be performed in the meadow and roof images to save computational time.

4.3 Garbage detection on the meadow

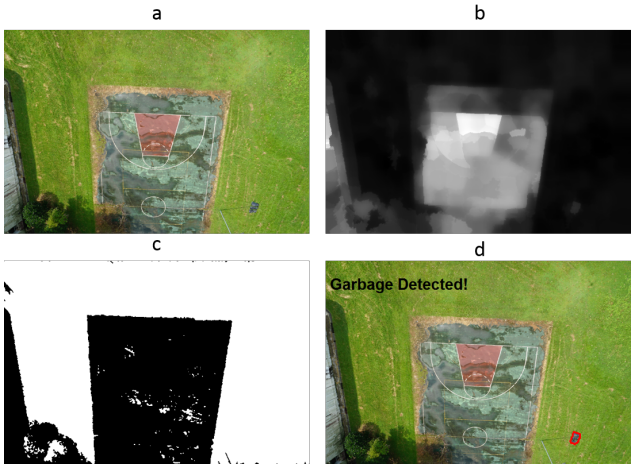


Figure 4: Garbage detection result on the meadow. a: original image. b: saliency map. c: mask image of the meadow region. d: result of detected garbage marked in red. Best viewed in color.

The experimental results for the illegal dumping detection on a meadow will be presented in this section, where some black plastic bags are placed on the meadow as the dumped waste. The rectangularity threshold $thresh_{rt}$ is set to 0.5. The size thresholds are $thresh_{al} = 30$, $thresh_{au} = 600$, and $thresh_{dl} = 20$, $thresh_{du} = 40$ respectively. The saliency threshold is $thresh_{sa} = 0.1$.

As shown in Fig. 4, there is a plastic bag on the meadow, which could be clearly observed in the saliency map. Moreover, the mask image of the meadow effectively covers the entire meadow region, such that the salient objects on the basketball court can be removed as outliers. After post-processing, the boundaries of the meadow could be further removed, and the resulting saliency map only contains the plastic bag.

For comparison, using the same original image from Fig. 4, the saliency map from other methods are displayed in Fig. 5. It shows the saliency map generated from the approaches proposed in [6, 9] and the original version of [13] respectively. Since the dumped waste is not in the center, the methods that emphasize central-bias, for instance [6] and [13], may not work well.

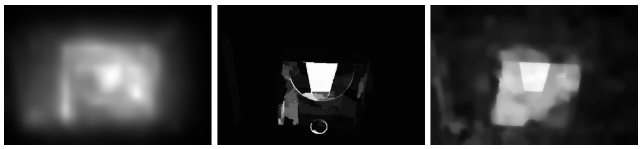


Figure 5: Comparison of saliency maps. From left to right: saliency maps generated by [6], [9], [13].

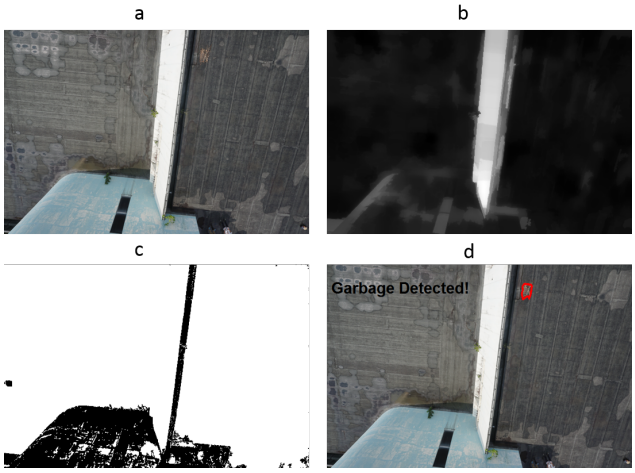


Figure 6: Garbage detection result on the roof. a: original image. b: saliency map. c: mask image of the roof region. d: result of detected garbage marked in red. Best viewed in color.

4.4 Garbage detection on the roof

This section presents the results of the proposed approach for garbage detection on roof, and the dumped waste are tree branches. The rectangularity threshold $thresh_{rt}$ is set to 0.6, which is higher than that of the previous experiment, because the water pond on the roof whose boundaries have irregular shapes produce many outliers. The size thresholds are $thresh_{al} = 300$, $thresh_{au} = 600$, and $thresh_{dl} = 20$, $thresh_{du} = 30$. The saliency threshold is also set to $thresh_{sa} = 0.1$ as used in the meadow case.

It could be observed in Fig. 6 that although the tree branches are not as salient as the plastic bag shown in Fig. 4, it could still be detected.



Figure 7: Comparison of saliency maps. From left to right: saliency maps generated by [6], [9], [13].

To compare the results of different saliency detection al-

http://www.imavs.org/

gorithms, the same image from Fig. 6 is used, and the saliency maps are shown in Fig. 7. Similar to what is observed in Fig. 5, the algorithms assuming that the object is close to the image center fail to detect the tree branches.

4.5 Discussion

Some may contend that instead of using saliency, HSV thresholds could be applied to the meadow or roof regions to detect the plastic bags or tree branches, as they possess unique colors such as black or brown. Admittedly, color thresholding may work well as the meadow or roof regions are quite homogeneous. However, the type of garbage is not limited to only two kinds, and thus saliency is preferable over color because of its capabilities to detect all kinds of dumped waste.

Though the proposed approach is extensible to detect varied kinds of garbage, there are also certain limitations associated with its extensibility. For instance, sometimes the lighting pole on the meadow is detected as dumped waste, because it stands out from the scene.

Another drawback of our work is its limited dataset size, which contains few images for each category, and thus the number of images that have debris in the scene is very small. Hence it is difficult to evaluate the proposed approach quantitively. To obtain statistically meaningful results, a larger dataset is needed. In addition, the small dataset also hinders the usage of CNNs for scene classification and garbage detection as the training samples are quite limited. One possible way is to apply transfer learning to resolve this issue.

5 CONCLUSION

In this paper, a new saliency detection algorithm is proposed, which consists of pre-processing, saliency detection and post-processing. During the first stage, the image is classified based on the number of green blobs. For the second stage, color and size priors are designed to obtain a weak saliency map. This map is used to generate the training samples for a boosted classifier. The classifier then produces a strong saliency map, which is fused with the weak saliency map. In the third stage, the outliers that are located on the non-homogenous regions, or inconsistent with the size requirements are pruned. Experimental results show that the proposed approach could perform illegal dumping detection on aerial images captured above the meadow and roof. Consequently, the proposed approach could be exported to other salient object detection applications as well.

ACKNOWLEDGMENT

The authors would like to thank the members of NUS UAV Research Group for their kind support, and Prof Lu Huchuan for the generous sharing of their code.

REFERENCES

[1] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.

[2] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[3] Davide Scaramuzza, Michael C Achtelik, Lefteris Doitsidis, Felice Friedrich, Elias Kosmatopoulos, Alessio Martinelli, Markus W Achtelik, Maria Chli, Savvas A Chatzichristofis, Laurent Kneip, et al. Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in gps-denied environments. *Robotics & Automation Magazine, IEEE*, 21(3):26–40, 2014.

[4] Kate Duncan and Santonu Sarkar. Saliency in images and video: a brief survey. *Computer Vision, IET*, 6(6):514–523, 2012.

[5] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (11):1254–1259, 1998.

[6] Jonathan Harel, Christof Koch, and Pietro Perona. Graph-based visual saliency. In *Advances in neural information processing systems*, pages 545–552, 2006.

[7] Ming Cheng, Niloy J Mitra, Xumin Huang, Philip HS Torr, and Song Hu. Global contrast based salient region detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(3):569–582, 2015.

[8] Xiaodi Hou and Liqing Zhang. Saliency detection: A spectral residual approach. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[9] Guanbin Li and Yizhou Yu. Visual saliency based on multiscale deep features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5455–5463, 2015.

[10] Tie Liu, Zejian Yuan, Jian Sun, Jingdong Wang, Nanning Zheng, Xiaoou Tang, and Heung-Yeung Shum. Learning to detect a salient object. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(2):353–367, 2011.

[11] Huaizu Jiang, Jingdong Wang, Zejian Yuan, Yang Wu, Nanning Zheng, and Shipeng Li. Salient object detection: A discriminative regional feature integration approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2083–2090, 2013.

http://www.imavs.org/

[12] Simone Frintrop, Erich Rome, and Henrik I Christensen. Computational visual attention systems and their cognitive foundations: A survey. *ACM Transactions on Applied Perception (TAP)*, 7(1):6, 2010.

[13] Na Tong, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Salient object detection via bootstrap learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1884–1892, 2015.

[14] Arindam Bose. How to detect and track red, green and blue objects in live video, 2013–2014.

[15] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, 2012.

[16] Fan Yang, Huchuan Lu, and Yen-Wei Chen. Human tracking by multiple kernel boosting with locality affinity constraints. In *Computer Vision–ACCV 2010*, pages 39–50. Springer, 2010.

http://www.imavs.org/

Simulation of an Obstacle Avoidance Algorithm in a Dynamic 2D Environment

J.S Coetzee*, Dr. WJ Smit†
Department of Mechanical and Mechatronic Engineering
Stellenbosch University
Stellenbosch, South Africa

http://www.imavs.org/

ABSTRACT

This paper describes the methods followed to more accurately predict how a combination of D* Lite and the Virtual Force Field method would react when sensor as well as pose uncertainties are considered during the map building process. These uncertainties are then simulated using MATLAB where the main focus is the effect that these uncertainties have on the combined algorithms' output and whether the same results can be obtained by using simplified assumptions.

1 INTRODUCTION

Obstacle avoidance algorithms can be broadly classified into two different categories, global and local path planning [1]. Global path planning is mainly done offline, whereas local collision avoidance must be fast, reactive and usually carried out online to ensure the safety of the vehicle by compensating for previously unknown obstacles.

The key distinction between the two obstacle avoidance categories is the amount of information available regarding the robots' immediate environment. If all the information regarding the obstacles is available, global path planning will be the best approach. On the other hand, if the information regarding the environment is incomplete or unreliable, local information based on sensor data will have to be used to navigate [2].

Apart from using one or a combination of the two approaches to provide a basic solution, there are other criteria the algorithms must preferably satisfy. These criteria include the robot's dynamics, implementation time, limited computational resources, robustness as well as the optimality of the code being implemented [1].

Although many techniques for both local and global collision avoidance have been proposed in the recent literature [1, 2, 3], there is still a great interest around the globe to better address obstacle avoidance.

The aim of this paper is to simulate a combination of two different avoidance algorithms with a probability map building technique to see how it will react in a dynamic environment. Therefore, it was decided to use a slower global path planning algorithm (D* Lite) with a fast reacting local obstacle avoidance algorithm (Virtual Force Field) while building a map of the surroundings by taking into account sensor noise as well as pose uncertainty.

In Section 2 a general background regarding the map building process used in this paper is given. This is followed by an overview of the Virtual Force Field method as well as D* Lite. The simulations, as well as their results, will be discussed in Section 3 followed by the conclusion and future work in Section 4.

2 BACKGROUND

2.1 Map Building

In an ideal world, hard assignment regarding the occupancy of a grid cell (one for occupied and zero for empty) can be used since everything about the robot and sensor is known. However, sensor measurements are noisy, which implies that the position of the robot is not accurately known. Therefore, it would be more realistic to assign a probability value to each grid cell, representing the certainty of a cell being occupied.

Since the sensor is mounted on the multicopter, all measurements received will be given relative to its pose at that time [4]. Suppose, m_i is the outcome of cell i being occupied given the current pose and measurement (z) from the multicopter and sensor respectively. Assuming, for now, that the multicopter estimates its pose correctly, the grid update formula is given by [5]:

$$\log\left(\frac{p(m_\mathrm{i}|z_{1:\mathrm{t}})}{p(m_\mathrm{i}^\mathrm{c}|z_{1:\mathrm{t}})}\right) = \log\left(\frac{p(m_\mathrm{i}|z_\mathrm{t})}{p(m_\mathrm{i}^\mathrm{c}|z_\mathrm{t})}\right) + \log\left(\frac{p(m_\mathrm{i}|z_{1:\mathrm{t\text{-}1}})}{p(m_\mathrm{i}^\mathrm{c}|z_{1:\mathrm{t\text{-}1}})}\right) - \log\left(\frac{p(m_\mathrm{i})}{p(m_\mathrm{i}^\mathrm{c})}\right) \quad (1)$$

The last term in Equation 1, also given in Equation 2, is known as the prior probability log-odds ratio of a cell. In the absence of any prior knowledge regarding the environment, this term is set to zero, i.e. $p(m_\mathrm{i}) = 0.5$.

$$\log\left(\frac{p(m_\mathrm{i})}{1-p(m_\mathrm{i})}\right) \quad (2)$$

The second term on the right-hand side of Equation 1 is the probability of cell i being occupied. This value takes into

*Email address: 16484649@sun.ac.za
†Email address: wjsmit@sun.ac.za

account all previous measurements regarding cell i whereas the first term on the right-hand side of Equation 1, also given in Equation 3, is known as the inverse sensor model. This term only contains new information given by the sensor at time t. The main reason log-odds is used is to enable the update equation to integrate new measurements into the grid through addition, avoiding truncation errors as well as the multiplication of probabilities.

$$p(m_{\mathrm{i}}|z_{\mathrm{t}}) \tag{3}$$

To convert the log-odds ratio (given on the left hand side of Equation (1)) back to a probability value, the following steps can be taken:

$$\lambda = \log\left(\frac{p(m_{\mathrm{i}}|z_{1:\mathrm{t}})}{1-p(m_{\mathrm{i}}|z_{1:\mathrm{t}})}\right) \tag{4}$$

$$p(m_{\mathrm{i}}|z_{1:\mathrm{t}}) = \frac{e^{\lambda}}{1+e^{\lambda}} \tag{5}$$

Therefore cell i will be updated accordingly, taking into account all prior knowledge regarding obstacles as well as new sensor measurements.

Although this algorithm assumes a static environment, it will also update the map sub-optimally for moving obstacles by creating a smeared representation in the evidence grid [4]. For more information about the inverse sensor model and the pose uncertainty of the multicopter, refer to Section 3.

2.2 *Virtual Force Field*

This Virtual Force Field is a global path planning algorithm with local collision avoidance. Historically, this algorithm was mainly used for ground robots and only recently started being used in multicopters. One of the most significant benefits of this method is the fact that no knowledge of the multicopter model is neccesary [6]. The Virtual Force Field function consists of an attraction field that pulls the multicopter to the goal position and a repulsive field eminating from obstacle to ensure that it travels safely [1]. The multicopter can therefore be seen as a particale with a positive electrical charge. If all the obstacles are given a positive charge as well, there will be a repulsive force on the multicopter. On the other hand, if the goal is given a negative charge, there will be an attraction force on the multicopter [7]. This method then calculates the resultant force along with the resultant direction of the combined forces to plan its next position as seen in Figure 1.

An alternative way of implementing this method is by taking the gradient of the resultant force to calculate the next position. This can be compared to a marble on the floor. If the floor is slanted in a certain direction, the marble will start rolling till it reaches the bottom [7], as seen in Figure 2.

By using the gradient method, a path to the goal position can be generated by checking the neighbouring values at each

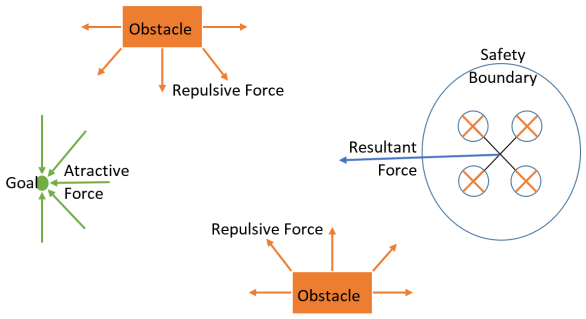


Figure 1: Potential field vector diagram.

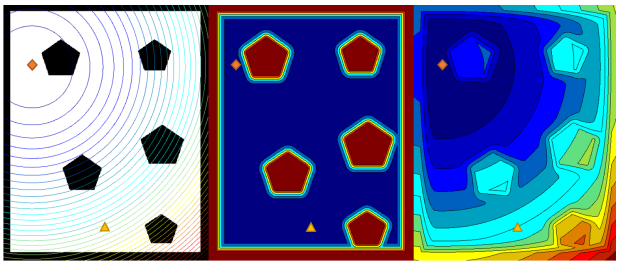


Figure 2: Potential field gradient diagram. The triangle and the diamond respectively represents the start position goal position of the robot. The first image represents the attractive force around the goal position, while the second image represents the repulsive force from each obstacle. The last image represents the sum of these forces.

cell and continuously moving to the cell with the lowest F_{total} value, where F_{total} is given by:

$$F_{\mathrm{total}}[i,j] = F_{\mathrm{att}}[i,j] + F_{\mathrm{rep}}[i,j] \tag{6}$$

$$F_{\mathrm{att}}[i,j] = \sqrt{(x-x_{\mathrm{goal}})^2 + (y-y_{\mathrm{goal}})^2} \tag{7}$$

$$F_{\mathrm{rep}}[i,j] = \begin{cases} 0 & \text{if } r > r_{\mathrm{safe}} \\ 2^{(r_{\mathrm{safe}}-r)} & \text{if } r < r_{\mathrm{safe}} \end{cases} \tag{8}$$

The current distance of the multicopter to the nearest obstacle is given by r where r_{safe} is the safety boundary as seen in Figure 1.

The Virtual Force Field method does not explicitly avoid moving obstacles and has a tendency to get caught in local minima. Therefore, the multicopter can get caught in cluttered environments before reaching its goal [1], but these issues can be addressed with the proper implementation of a global path planning algorithm [1, 8] as addressed in Section 3.3.

2.3 *D* Lite*

The D* Lite algorithm used in this paper is based on the second version of the D* Lite pseudo code given in [9]. According to [10], D* Lite is easy to understand and extend,

http://www.imavs.org/

http://www.imavs.org/

while being at least as efficient as Focused Dynamic A* (D*) itself. Even though they are algorithmically different, they still exhibit the same behavior. Both algorithms search from the goal vertex to the current vertex of the multicopter by using a heuristic function to focus their search, whilst also using similar ways to minimize reordering of the priority queue [10].

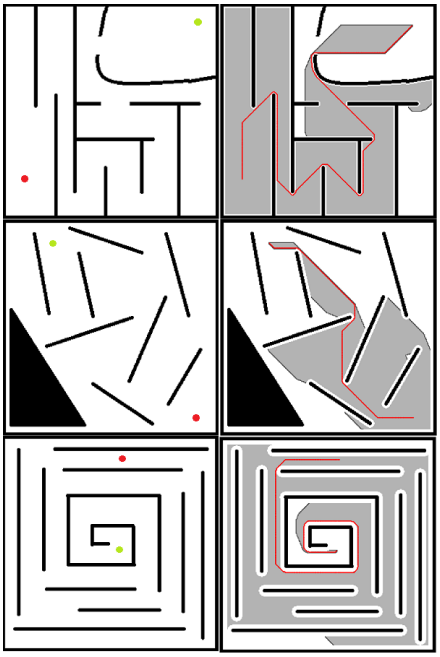


Figure 3: D* Lite global path planning (static environment). The red indicates the shortest path, while the light gray indicates all the vertices evaluated to generate the path. Dark gray indicates the vertices that will be expanded next, if needed. These paths were calculated with a 2, 4 and 6 m boundary around the obstacles respectively(1 pixel $=$ 1 m).

D* Lite is classified as a global path planning algorithm because the multicopter will have to maintain an internal map of its surroundings to calculate a path, as can be seen in Figure 3. When the map is updated with the relevant sensor data, the D* Lite algorithm will determine if the new entries into the map will have an effect on the current path and recalculate if necessary, as seen in Figure 4.

The algorithm is able to re-plan faster than planning from scratch each time as it modifies its previous search results when needed. In other words, it efficiently recalculates the shortest path from the current cell of the multicopter to the goal cell by recalculating only the relevant cells that have changed or have not been calculated before.

The algorithm has to obey two conditions [9, 10]. The first is that the heuristic function $h(s, s')$ has to be non-negative and forward-backward consistent. That is, it has to adhere to $h(s, s'') \leq h(s, s') + h(s', s'')$ for all vertices $s, s', s'' \in S$[1]. The second is that the cost of the shortest path from any vertex $s \in S$ to $s' \in S$ has to be more than or equal to the value of the heuristic function for the same vertices i.e. $h(s, s') \leq c^*(s, s')$. Both the heuristic and cost function used in the end can be described by Figure 5.

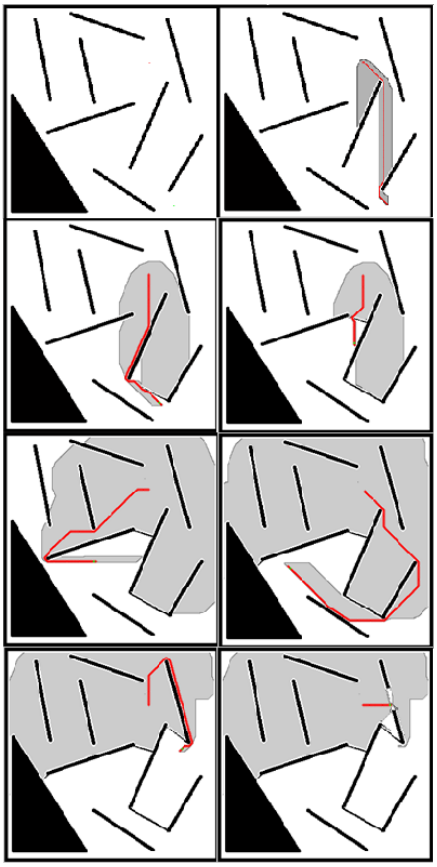


Figure 4: D* Lite global path planning (dynamic environment). The red indicates the shortest path, while the light gray indicates all the vertices evaluated to generate the path. Dark gray indicates the vertices that will be expanded next, if needed. The path was calculated with a 2 m boundary around the obstacles (1 pixel $=$ 1 m).

3 SIMULATIONS AND RESULTS

To accurately simulate how the multicopter will react when using D* Lite and the Virtual Force Field methods described in Section 2, the map of the surrounding area must be built as if the measurements were taken from an actual multicopter. To do this, the amount of measurements being taken by the lidar[2] in one revolution has to be taken into account

[1] S denotes the finite set of vertices of the graph.

[2] PulsedLight lidar sensor combined with a continuously rotating servo motor, giving up to 160 readings per revolution with a maximum distance of 40m when not rotating.

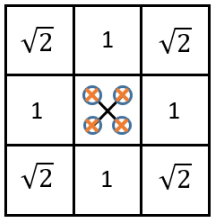


Figure 5: Heuristic and cost function used for D* Lite algorithm.

along with measurement and pose uncertainty as discussed in Section 3.1 and 3.2 respectively.

With a more realistic map of the environment being built, it will be possible to more accurately simulate how D* Lite combined with the Virtual Force Field method will work as can be seen in Section 3.3.

3.1 Incorporating Measurement Uncertainty into Inverse Sensor Model

When using Equation (1) to update the probability of a cell, an ideal inverse sensor model should return $p(m_i|z_t) = 0$ for all the cells in front of the obstacles, $p(m_i|z_t) = 1$ for the cell containing the obstacles and $p(m_i|z_t) = 0.5$ for all the cells behind the obstacles as seen in Table 1. Therefore, the function representing an ideal sensor model can be seen in Figure 6 and is defined as [5]:

$$g(r) = \begin{cases} 0 & \text{if } r < Z - \frac{L}{2} \\ 1 & \text{if } Z - \frac{L}{2} \leq r < Z + \frac{L}{2} \\ 0.5 & \text{otherwise} \end{cases} \tag{9}$$

Where r represents the distance from the current position of the multicopter to the cell that is being updated and Z represents the actual measurement from the sensor. Since the centre of the cells are used to calculate the distance to the sensor, the peak at Z may be completely missed if the centre of the cell has a slight offset from Z. To compensate for this, another parameter, L, is introduced, signifying a band of r values that receive a probability corresponding to definitely occupied ($p(m_i|z_t) = 1$). A natural choice for the value of this parameter is the diagonal distance between two corners of a grid cell.

$p(m_i\|z_t)$	$\log\left(\frac{p(m_i\|z_t)}{1-p(m_i\|z_t)}\right)$	Interpretation
0	$-\infty$	definitely free
0.5	0	unknown
1	∞	definitely occupied

Table 1: Correspondences between the probabilities and the log odds ratios along with their respected interpretations [11].

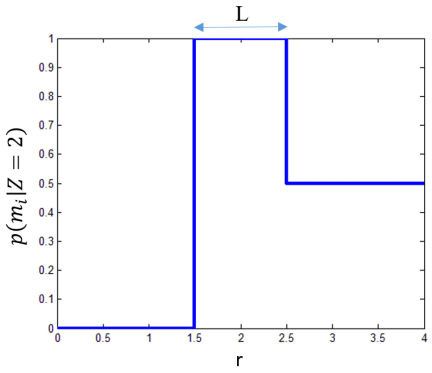


Figure 6: Ideal sensor model.

When collecting measurement data, it may be noisy. Assuming that the noise is normally distributed around the measured value Z, with a standard deviation of σ. A probability density function (PDF) of such a Gaussian distribution can then be generated, as seen in Figure 7, and is defined as:

$$f(r; Z, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(r-Z)^2}{2\sigma^2}} \tag{10}$$

To incorporate the noise into the ideal sensor model, a convolution between Equation 9 and 10 had to be performed giving the following piecewise defined function [11, 5]:

$$\begin{gathered} \text{For } r \in \left(-\infty, Z - \frac{L}{2}\right) \\ (f * g)(r) = 0 \\ \text{For } r \in \left[Z - \frac{L}{2}, Z + \frac{L}{2}\right) \\ (f * g)(r) = -\frac{1}{2}\text{erf}\left(\frac{-Z}{\sqrt{2}\sigma}\right) + \frac{1}{2}\text{erf}\left(\frac{r - 2Z + \frac{L}{2}}{\sqrt{2}\sigma}\right) \\ \text{For } r \in \left[Z + \frac{L}{2}, Z + \infty\right) \\ (f * g)(r) = -\frac{1}{4}\text{erf}\left(\frac{r - 2Z - \frac{L}{2}}{\sqrt{2}\sigma}\right) + \frac{1}{2}\text{erf}\left(\frac{r - 2Z + \frac{L}{2}}{\sqrt{2}\sigma}\right) \\ -\frac{1}{4}\text{erf}\left(\frac{-Z}{\sqrt{2}\sigma}\right) \end{gathered} \tag{11}$$

Where erf represents the error function, r represents the distance from the current position of the multicopter in the map to the cell being updated and Z represents the measured distance given by the sensor. To get a better understanding of how the sensor model in Equation 11 works alongside Equation 1, refer to Figure 8 and 9. The cells in front the obstacle got their probability decreased while the cell containing the obstacle got an increase in probability whereas the cells behind the obstacle stayed unaffected as would be expected. The inverse sensor model, however, is not used as shown in

http://www.imavs.org/

Figure 8. This is because a value of 0 will be mapped to $-\infty$, implying that new measurements will not be able to change the value of the cell any more as seen in Table 1. To avoid this problem, the sensor model was given both an upper and a lower bound value [11].

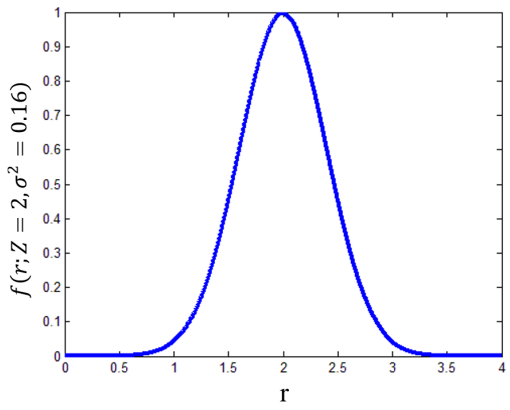


Figure 7: Probability density function.

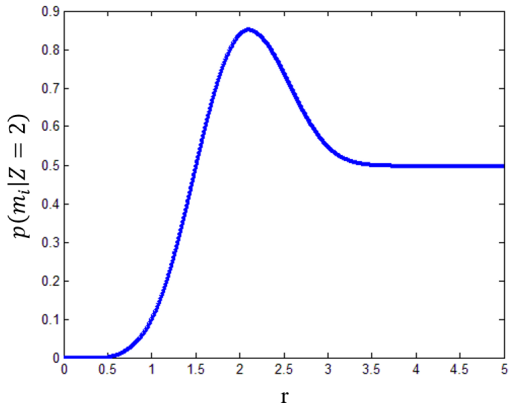


Figure 8: Gaussian inverse sensor model $L = 1$ and $\sigma^2 = 0.16$.

To see what effect the inverse sensor model has on the readings, refer to Figure 10. As the map gets darker in some places, the certainty that no obstacles are present grows while, as the map gets lighter, the certainty that an obstacle is indeed present, increases. Using a lidar as the measurement instrument, it was found that the ideal sensor model would give similar results as the Gaussian inverse sensor model if the same upper and lower bounds were implemented on it. Therefore, the Gaussian inverse sensor model derived would be more suitable for sensors with bigger measurement uncertainty like sonar or stereo vision and an ideal-like inverse sensor model can be used alongside the lidar for all intents and purposes.

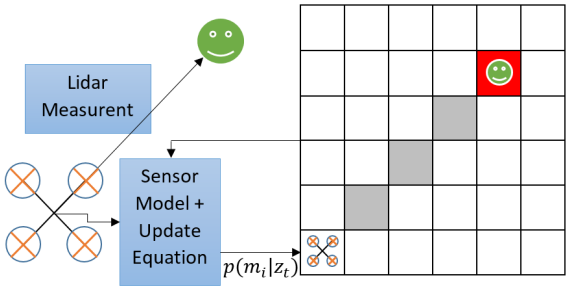


Figure 9: Map being updated by using previous map information along with new measurement information.

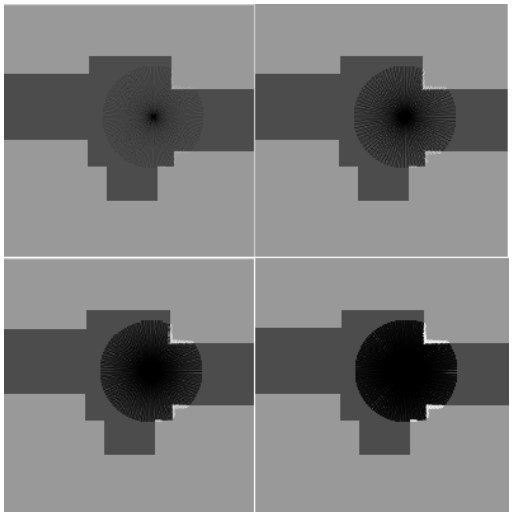


Figure 10: Map being updated using the Gaussian inverse sensor model. $L = 1.4$ and $\sigma = 0.3$. First image (top) $= 1$ scan. Second image (top) $= 5$ scans. First image (bottom) $= 10$ scans. Second image (bottom) $= 20$ scans.

3.2 *Pose Uncertainty*

To model the pose uncertainty of the quadcopter, measurements were taken with both a Piksi real time kinematic (RTK) GPS and the quadcopter controller. Since the RTK GPS is accurate up to 2 cm, horizontally, it was used as a ground truth measurement to determine the controllers' position estimation error. This data was then processed and a normal distribution was fit to it. The results of these distributions can be seen in Table 2. To incorporate this data into the model, the cumulative distribution function (CDF) first has to be calculated. Therefore, given a PDF with mean μ and variance σ^2, the CDF, as seen in Figure 11, can be generated with:

$$F_x(x; \mu, \sigma^2) = \frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\left(\frac{x - \mu}{\sqrt{2}\sigma}\right) \qquad (12)$$

http://www.imavs.org/

	North		East	
	μ [m]	σ [m]	μ [m]	σ [m]
Test 1	-0.3813	0.2571	-0.1819	0.1864
Test 2	0.5838	0.2951	-0.0847	0.0913
Test 3	-0.6745	0.3395	0.5166	0.2779
Test 4	-1.2974	0.3270	0.7304	0.3291

Table 2: PDF distribution values obtained through testing for both the Northern and Eastern error (Given in North, East and Down (NED) coordinate system). Each test was done over a 15 min time frame.

To sample from a known one-dimensional PDF, a uniform random number, u, between 0 and 1 first has to be generated. This number can then be used to calculate the inverse of the CDF. This method is also known as the inverse transform method [5] and can be found by using:

$$x = \sqrt{2}\sigma \, \text{erf}^{-1}(2u - 1) + \mu \tag{13}$$

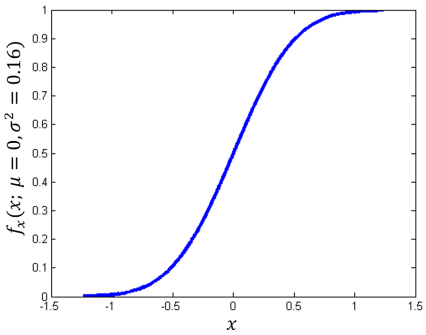


Figure 11: Cumulative distribution function.

These values are then used to update the map. It will be assumed that the same measurement could have been made from any of the positions generated by Equation 13. Since the values were randomly generated from a normal distribution, it makes sense to give an equal weight to each of them, summing to 1. To find the total value a cell will be updated with, each of the weighted probabilities affecting that cell have to be added together, refer to Figure 12 for a visual representation. Equation 1 can therefore now be re-written as follows to take into account pose uncertainty [11, 5]:

$$\log\left(\frac{p(m_\text{i}|z_{1:\text{t}})}{p(m_\text{i}^\text{c}|z_{1:\text{t}})}\right) = \sum_{j=1}^{M} w_\text{t}^{[j]} \log\left(\frac{p(m_\text{i}|z_\text{t}^{[j]})}{p(m_\text{i}^\text{c}|z_\text{t}^{[j]})}\right) + \log\left(\frac{p(m_\text{i}|z_{1:\text{t-1}})}{p(m_\text{i}^\text{c}|z_{1:\text{t-1}})}\right) - \log\left(\frac{p(m_\text{i})}{p(m_\text{i}^\text{c})}\right) \tag{14}$$

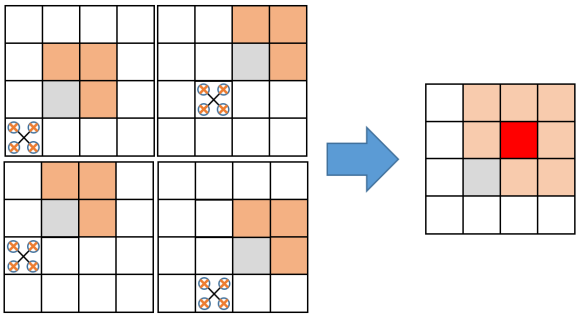


Figure 12: Illustration of how a map will be updated with pose uncertainty.

Since the maximum μ value from Table 2 is -1.2974 m and the maps are generated as images where each pixel represents 1 m, the effect of the pose uncertainty is difficult to see. One noticeable difference in the results, however, is the blurred effect that comes from Equation 14 as illustrated in Figure 12. Due to the pose uncertainty, the map generated from sensor measurements of the environment will also move in accordance with this uncertainty. To illustrate this, an error of 7 m in both the Northern as well as Eastern direction was simulated as seen in Figure 13. Since the pose uncertainty always changes when flying (as shown in Table 2), the map the multicopter builds of the environment will always be different from the previous flight as the map is updated relative the multicopters current pose.

Unfortunately, the pose uncertainty update formula is computationally intensive as a large number of samples has to be generated from the CDF to accurately represent the PDF of the multicopter's pose uncertainty each time new sensor data has to be incorporated into the map. However, it works well when simulating the uncertainty in a robots' pose. Luckily, when actually flying, this equation does not need to be used as the map will be updated according to the multicopters' pose estimation. Therefore, the map relative to the multicopter will be correct and can be used to navigate safely.

If a global map of a static environment is available and no sensor measurements can be taken, then an extra safety boundary can be added to the obstacles. In other words, this will basically enlarge the obstacles to compensate for the uncertainty in the multicopter's pose, ensuring its safety when flying without extra sensor information from the environment.

3.3 Combining Virtual Force Field with D Lite*

To be able to fly autonomously in a partially known or unknown environment, it was decided to use a combination of the three different techniques discussed in Section 2 to ensure the safety of the multicopter. As indicated previously, the idea is to let the multicopter plan a global path while still being able to react locally. D* Lite was therefore used as

http://www.imavs.org/

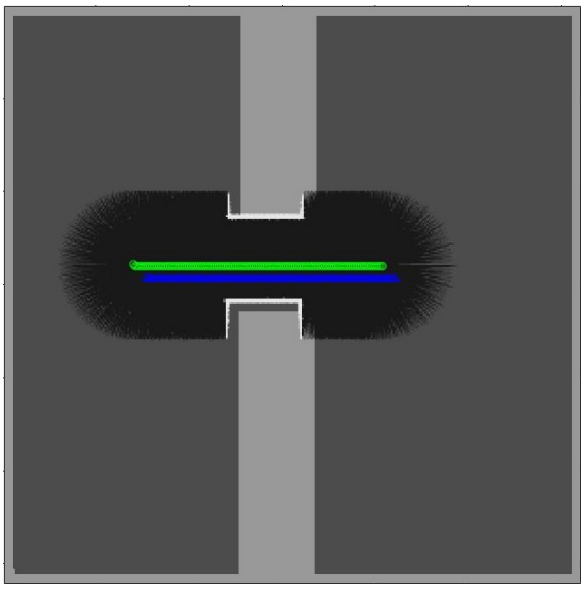


Figure 13: Illustration of how a map built by the multicopter will look over a map of the actual environment. A pose error of 7 m in both the Northern and Eastern direction was chosen for the illustration. Blue line: actual position. Green line: position estimation of the multicopter. Light gray illustrates the actual environment, whereas black and white represents the map the multicopter built. Black being no obstacles whereas white represents obstacles.

a global path planner by calculating a path from the multicopter's current position to the goal position. In a static environment, it will only have to compute the path once, but in a dynamic or unknown environment, the algorithm will be executed continuously until the goal position is reached. Even though the path may stay the same, the algorithm still has to check whether the addition or removal of obstacles affects the path after every scan. This continuous re-checking/planning can therefore get computationally intensive. One way of limiting this is by only re-checking/planning every few steps. The only problem with this is that if an obstacle is scanned, blocking the path to the goal while D* Lite is not active, the multicopter will fly straight into it.

For this reason it was decided to combine the global path planner with the Virtual Force Field method. By combining these two methods, they can help compensate for each others shortcomings i.e. D* Lite won't have to continuously recalculate, while the Virtual Force Field won't get stuck in local minima as the multicopter is being "pulled" to the goal position by making temporary attraction points on the calculated global path.

With the algorithm as described above, the amount of steps that can be taken before D* Lite has to recalculate the path, can be specified. If these steps are more than the sensor range, the multicopter will still move along the previously calculated path and update the map as usual. If an unexpected obstacle appears on this path, the map will be updated accordingly while the Virtual Force Field will assure the safety of the multicoper by maintaining the necessary safety distance.

The safety boundary used by D* Lite has to be the same as the boundary used by the Virtual Force Field, otherwise D* Lite might calculate a path that goes through obstacles that are too close to each other, causing the multicopter to still get trapped in a local minima.

Both D* lite and the Virtual Force Field does not specifically avoid moving obstacles, but since the map building process will generate a smeared representation for a moving obstacle, they will be avoided sub-optimally until the map is updated correctly again.

4 CONCLUSION AND FUTURE WORK

In Section 3 a map building technique was successfully developed and implemented by taking into account both sensor and pose uncertainties. For the sensor model it was found that the Gaussian inverse sensor model gave similar results to the ideal sensor model if the same upper and lower bounds were applied. This is only the case because a lidar with small measurement uncertainty was simulated. The Gaussian sensor model, however, could still prove to be useful when used alongside sensors with bigger measurement uncertainties like sonar or stereo vision. With the multicopter's pose uncertainty, it was found that the update model created a blurred representation of the obstacles. As each lidar reading update required a significant amount of samples from the CDF to accurately represent the PDF, it became computationally intensive. Because the map is constantly updated according to the new lidar information, the uncertainty in the pose (and therefore the update equation) is not needed for the map building process when flying. Therefore, the map relative to the multicopter can be assumed correct and safe to travel. If, however, a multicopter has to be flown autonomously without a distance sensor, a map of the environment can be used if an additional uncertainty is added around the obstacles. Thus, moving the uncertainty from the multicopters' position to the obstacle and ensuring the safety of the vehicle.

With a constantly updating map in place, a combination of both D* Lite and the Virtual Force Field method was successfully implemented. Due to the fact that the D* Lite algorithm can get computationally intensive when flying in a constantly changing or unknown environment, it was decided to only recalculate the global path every few steps, when needed. The multicopter therefore follows the global path by making a temporary attraction point on it as it moves. When an unforeseen obstacle appears, the Virtual Force Field will ensure that the multicopter stays at a safe distance until a new global path can be calculated. Therefore, these two algorithms compliment each other since D* Lite is now less computationally intensive and the Virtual Force Field method does not get

http://www.imavs.org/

stuck in local minima any more. The code is, however, not without its flaws. If the safety boundary generated by D* Lite is smaller than the safety boundary generated by the Virtual Force Field, the algorithm can still fall into a local minima as the global path will be able to go through denser packed obstacles. This, however, can easily be fixed by always ensuring they are the same value.

In future work, the algorithm will be integrated with the flight controller and other components as depicted in Figure 14. Extensive tests will then be done on the multicopter simulator, jMAVSim, to see how the multicopter will react in various situations. Lastly, actual flight tests will also be completed to verify the simulated results as well as the assumptions made.

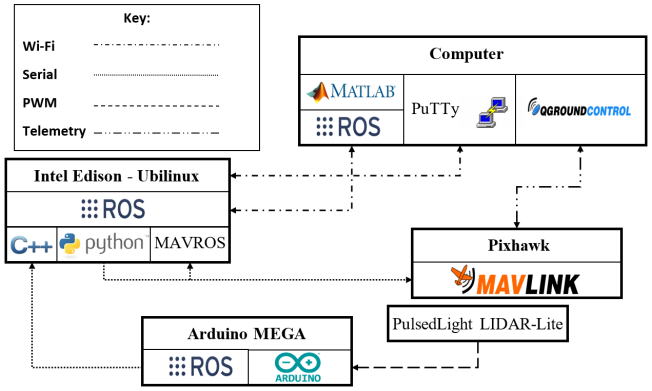


Figure 14: Hardware and software setup along with communication protocols used.

REFERENCES

[1] Anusha Mujumdar and Radhakant Padhi. Evolving Philosophies on Autonomous Obstacle/Collision Avoidance of Unmanned Aerial Vehicles. *Journal of Aerospace Computing, Information, and Communication*, 8(2):17–41, 2011.

[2] Michael Hoy, Alexey S. Matveev, and Andrey V. Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33(03):463–497, 2015.

[3] C. Goerzen, Z. Kong, and B. Mettler. A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. *Journal of Intelligent and Robotic Systems*, 57(1-4):65–100, 2010.

[4] Sebastian Scherer, Sanjiv Singh, Lyle Chamberlain, and Srikanth Saripalli. Flying Fast and Low Among Obstacles. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 2023–2029. IEEE, 2007.

[5] Daniek Joubert, Willie Brink, and Ben Herbst. Pose Uncertainty in Occupancy Grids through Monte Carlo Integration. *J Intell Robot Syst Journal of Intelligent & Robotic Systems : with a special section on Unmanned Systems*, 77(1):5–16, 2015.

[6] Tobias Paul, Thomas R. Krogstad, and Jan Tommy Gravdahl. Modelling of UAV formation flight using 3D potential field. *Simulation Modelling Practice and Theory*, 16(9):1453–1462, 2008.

[7] Luca De Filippis, Giorgio Guglieri, and Fulvia Quagliotti. Path Planning Strategies for UAVS in 3D Environments. *Journal of Intelligent & Robotic Systems*, 65(1-4):247–264, 2012.

[8] David Droeschel, Matthias Nieuwenhuisen, Marius Beul, Dirk Holz, Jörg Stückler, and Sven Behnke. Multilayered Mapping and Navigation for Autonomous Micro Aerial Vehicles. *Journal of Field Robotics*, 33(4):451–475, 2016.

[9] Sven Koenig and Maxim Likhachev. Improved fast replanning for robot navigation in unknown terrain. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 1, pages 968–975. IEEE, 2002.

[10] S Koenig and M Likhachev. D* Lite. *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 476–483, 2002.

[11] Daniek Joubert. Adaptive occupancy grid mapping with measurement and pose uncertainty. 2012.

http://www.imavs.org/

Small Height Duct Design for 17” Multicopter Fan Considering Its Interference on Quad-copter

Stremousov K.*, Arkhipov M.* **, Serokhvostov S.* **
* Moscow Institute of Physics and Technology, Department of Aeromechanics and Flight Engineering 16, Gagarina str., 140180, Zhukovsky, Moscow Region, Russia
** Central Aerohydrodynamic Institute, 1, Zhukovsky str., 140180, Zhukovsky, Moscow Region, Russia

http://www.imavs.org/

ABSTRACT

The 17 inch ducted fan with a 50mm height was studied in hover. Numerical simulations were provided by solving RANS equations with SST turbulence model using actuator disc with radial distribution of pressure difference according to numerical and experimental investigations of 17” propeller in hover. Optimal airfoil for axisymmetric duct was found. The 3D numerical simulations of four ducted fans interference were carried out. The geometry of duct was improved in term of power consumption considering interference of quad copter ducted fans.

1 INTRODUCTION

For the last decade multicopters became one of the most popular aircraft for the set of applications. These remotely piloted aircraft (RPA) can be used as cheap and safe devices for area monitoring from the air, security assistance, taking part in a rescue operations, inspection of defects in places that are difficult of access for human or robots. Moreover, multicopters are sufficiently stable, quiet and non-polluting aircraft with the low energy consumption in comparison with the other devices. One of the main features of such RPA is the ability to hover for a long time over same point. For the hovering time to be increased the energy consumption from the on-board energy source should be minimized while the propeller thrust remains constant. It could be provided by optimization of controllers and other electrical devises or increasing the propeller efficiency. The latter could be solved by installing the duct. Also duct protects propeller from contact with obstacles. For the study the 17 inch ducted fan with a 50mm height was chosen. Small height of the duct was chosen to satisfy the wind stability requirement. Thus the ducted fan thrust was set as 9N which is corresponding to the hovering regime.

2 METHODOLOGY

The principal functions of the duct are:

•Producing extra lifting force by the installing an airfoil at the appropriate incidence;

•Reducing the propeller induced drag;

•Protecting from contacts of the fan with surrounding obstacles.

Previous theoretical and experimental investigations give the following optimal parameters for a duct [1]:

•The front edge of the ring should be rounded and the rear edge must be sharp;

•Internal diameter of the duct should decrease firstly, then this diameter ought increase downstream;

•It is highly recommended to install the propeller in the narrowest part of the duct;

•The duct height b_a must be equal to 60% of propeller diameter D_p;

•The distance between duct front edge and propeller must be equal to 40% of duct height b_a;

•The optimal airfoil thickness is 18%;

•The airfoil incidence is 7°;

•The duct diameter D_d is equal to 1.26 of propeller diameter D_p.

The features described above are either represented in figure 1.

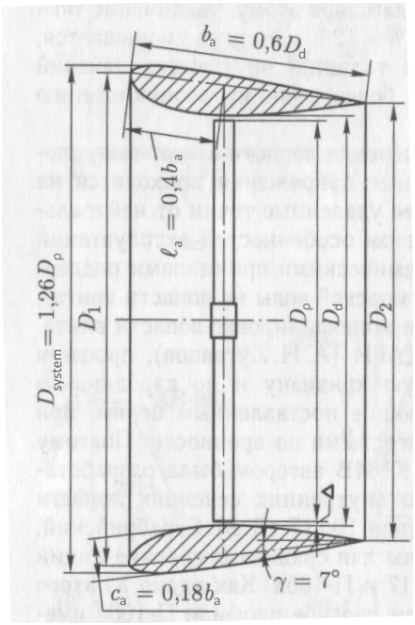


Figure 1: Features of a system.

http://www.imavs.org/

However, these features are set for the high duct which is unacceptable for multicopters because of wind stability. It is very difficult for multicopter to balance sideforce, including side wind flow. Thus it was decided to consider the small height duct ($b_a \sim 10\%$) which combines rather acceptable wind stability and advantages of normal duct. Therefore the main objective of the investigation was to maximally decrease the power consumption for the constant ducted fan thrust.

Firstly the foil optimization was continued for the axisymmetric task. Previously found airfoil [2] was improved by 19% using neuronet methods of optimisation. The optimisation task was solved numerically using an actuator disk with a radial distribution of pressure difference which was found previously from the numerical simulation of the 17" propeller [3].

Next step is numerical study of the ducted fans interference on quad-copter and minor shape improvement. Numerical simulations were provided by solving RANS equations with SST turbulence model on 25mn structured mesh. Pressure difference radial distribution on the actuator disk was the same as in the airfoil optimisation task.

3 AXISYMMETRIC TASK

Axisymmetric case was solved numerically using 2D structured mesh on the system of airfoil and actuator disc. The two-dimensional mesh which define 1 of the duct contains about 200 000 elements with wall cell thickness of 0.015 mm which provides maximum value of Y_+=0.932. As it was mentioned above the simulations were provided by solving RANS equations with SST turbulence model with the actuator disc. Surface of a duct was set as no slip wall. Every single simulation was stopped only after fixation of fourth significant digit in magnitude of the aerodynamic forces acting on a foil. The radial pressure difference distribution on the actuator disk was taken from the numerical simulation of the 17" propeller in hover [3]. The pressure difference on the quarter chord line of the propeller and approximation function is given in figure 2. Thus the approximation function was:

$$\triangle p = k*(23922*r^2-861*r+52)*(th(72,5r-0.21125)+1) \quad (1)$$

where r is a local value of radius. The maximum deviation of approximation function therefore is less than 0.5%. In each separate solver task the thrust remained constant at 9N by increasing or decreasing coefficient k of pressure difference on the actuator disk.

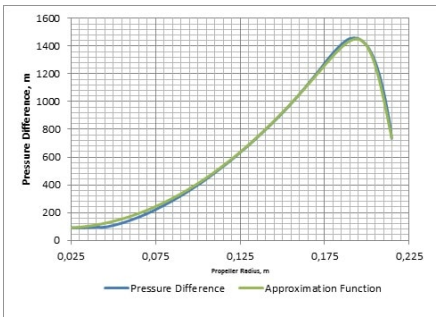


Figure 2: Pressure difference between upper and lower surfaces of the propeller on its quarter chord line.

During the previous and present studies, the main foil feature was understood. For such a small duct height it is not possible to develop a classical duct with an appropriate front edge to turn the flow and diffusor behind the propeller. Only one part could be used. If diffusor will be set alone the separation on its front edge will not allow diffusor to work optimally, moreover, in some cases, it is not working at all. Thus the main feature of the airfoil for small height duct should be turning of the flow to air ram on the propeller approximately perpendicularly.

Optimization task was based on varying of 51 parameters: one parameter is leading edge radius and other 50 are foils coordinate points. The optimization task was provided in 2D case with setting the mass flow, which corresponds to the mass flow obtained from the propeller simulation [3], over the upper surface of the airfoil. Airfoil which gives maximum thrust was chosen for the follow-up study. The foil obtained is shown in figure 3. This airfoil was also studied for two parameters: optimal position of the actuator disk along the chord and optimal incidence of the airfoil.

Optimization task was based on 51 parameters of foil coordinate points. The foil obtained is shown in figure 3. This airfoil was also studied for two parameters: optimal position of the actuator disk along the chord and optimal incidence of the airfoil.

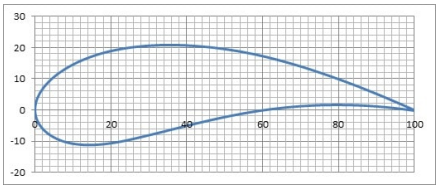


Figure 3: Airfoil obtained from the optimization task.

To find the optimal position of the actuator disk it was set consecutively from the front to the rear edge of the airfoil with a step of 0.1 of foil chord b_a. The results of the simulations are given in figure 4. According to the airfoil angle of incidence (23,56 for this case) the function of the power

consumption is in good agreement with the postulation of the installation of the propeller in the narrowest part of the duct

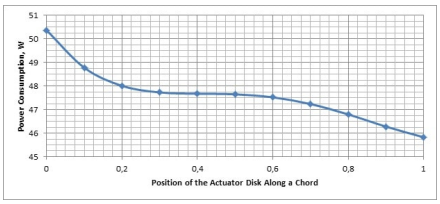


Figure 4: Power consumption as a function of actuator disk position along the airfoil chord.

Firstly airfoil obtained was set to the angle of incidence of 45.6 as a previously studied foil [2] and then was consecutively decreased with a step of 2 up to 17. The function of the power consumption is represented in figure 5.

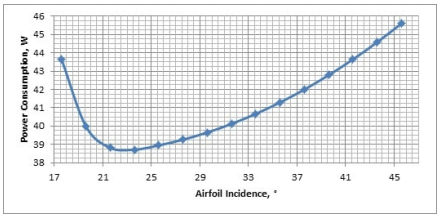


Figure 5: Power consumption as a function of the airfoil angle of incidence.

It is well enough seen that the minimal power consumption is 38.68W in comparison to 47.24W on the propeller alone. The power consumption decreases until the leading edge separation occurs on incidence of 21.6 then it significantly increases. The pressure coefficient distribution and separation on the leading edge depending on angle of the foil incidence is shown in figures 6-8.

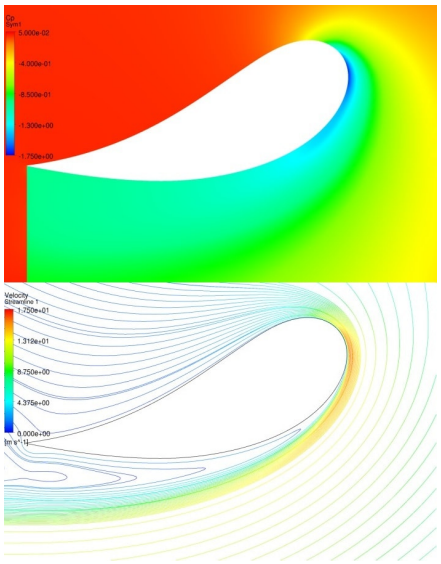


Figure 6: Pressure coefficient distribution (upper). Separation on front edge (lower). Angle of incidence 17,6°

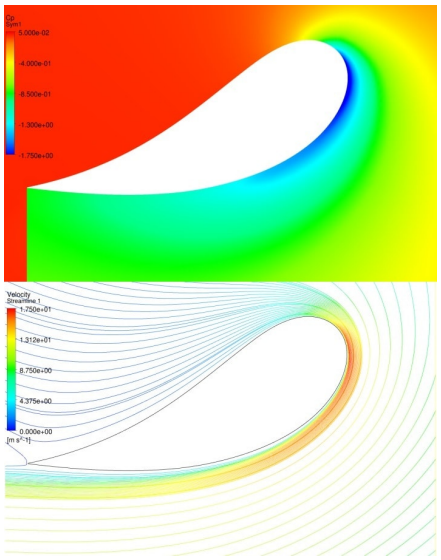


Figure 7: Pressure coefficient distribution (upper). Separation on front edge (lower). Angle of incidence 21,6°

http://www.imavs.org/

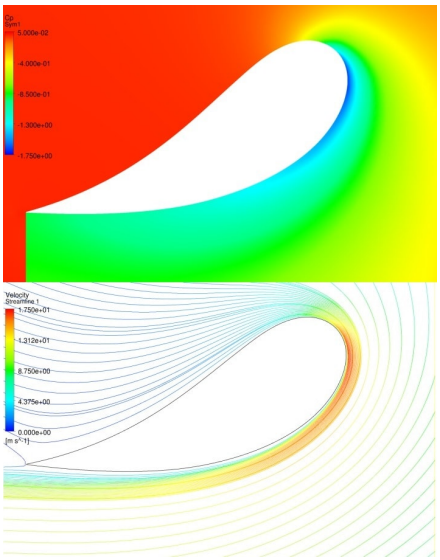


Figure 8: Pressure coefficient distribution (upper). Separation on front edge (lower). Angle of incidence 25,6

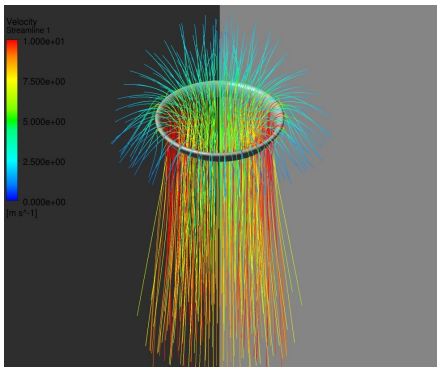


Figure 9: Streamlines that passes the actuator disc.

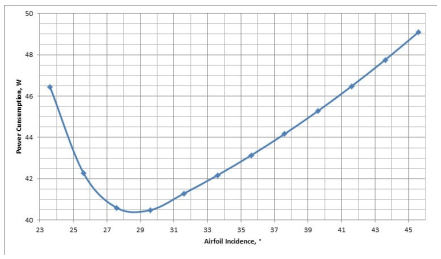


Figure 10: Power consumption as a function of the airfoil angle of incidence in 3D case.

4 DUCT SHAPE OPTIMIZATION CONSIDERING ITS INTERFERENCE ON QUAD-COPTER

The interference of four ducted fans was studied numerically using rotational periodicity symmetry conditions on the planes of geometrical symmetry. The structured meshes for the analysis have cylindrical C-H grid topology and contain about 25 million cells with wall cell thickness of 0.01 mm which provides maximum value of Y_+ = 0.759. As it was mentioned above the simulations were provided by solving RANS equations with SST turbulence model with the actuator disc with radial pressure difference distribution. Surface of a duct was set as no slip wall. Every single simulation was stopped only after fixation of fourth significant digit in magnitude of the aerodynamic forces acting on a duct.

Axisymmetric duct with airfoil obtained was simulated. Due to the geometrical features of the case the flow pattern slightly differ from the 2D case. The local angle of attack averagely increases by 6 because flow that passes the actuator disc in this case comes from the quarter of the space above duct (figure 9) in distinction from the axisymmetric case where flow comes from the entire front half sphere. Thus the power consumption dependency from the foil incidence became to the following view (figure 10):

The dependency shows sharp decrease from 23.56 to 27.56 degrees of incidence. After that there is a steady growth of power consumption. The minimum power consumption for the case of interference increased by 1.8W up to 40.5W. In comparison the power consumption of propeller in the case of interference is 49.6W.

Figures 11-12 represents fast flow junction. Nearby in one diameter below the duct streams from all four ducts attaches to each other and then evaluate as a one stream.

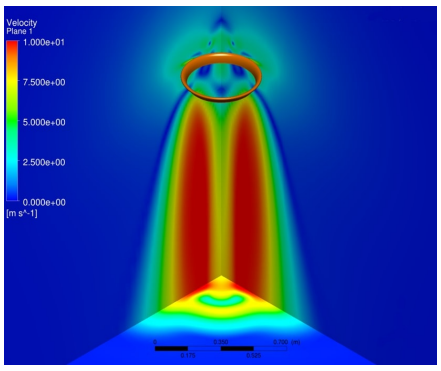


Figure 11: Velocity distribution on the symmetry plane and plane 1.5m under duct.

http://www.imavs.org/

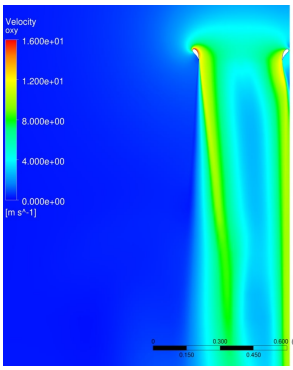


Figure 12: Velocity distribution on the plane parallel to XOY at the maximum radius placement.

The flowfield near the planes of symmetry necessitates varying the foil incidence in this area. The strong pressure increase occurs on the duct inside surface that lead to increase in power consumption to provide thrust of 9N. Typical Cp and velocity distributions in this section are represented in figures 13-14.

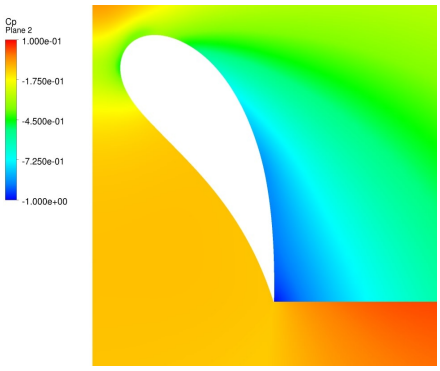


Figure 13: Pressure coeffitient distribution on the foils nearest to the symmetry.

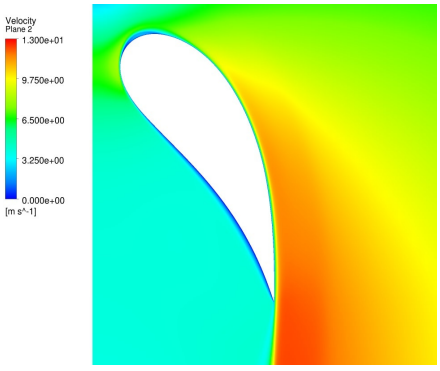


Figure 14: Velocity distribution on the foils nearest to the symmetry.

The resulting amounts of power consumption for the different angles of incidence of foil nearest to the symmetry plane are given below. Thus the optimal angle of incidence of the foils closest to the symmetries is 19.56 degree and the lowest power consumption appears to be 39.69 watts.

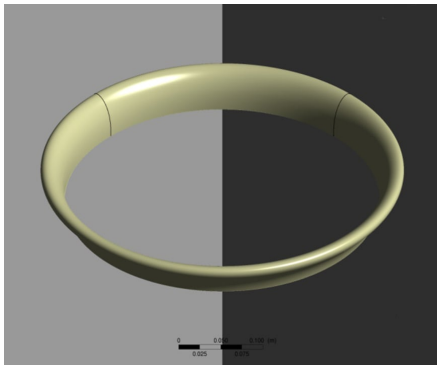


Figure 15: Airfoils nearest to the symmetry planes.

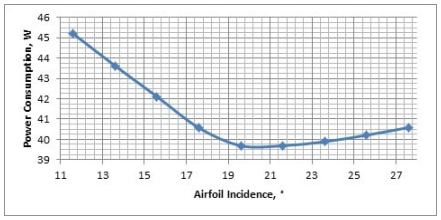


Figure 16: Power consumption as a function of incidence of airfoil nearest to the symmetry.

5 CONCLUSION

New airfoil for a small height duct obtained, the results of its simulation is in good agreement with an experimental data and represent an improvement over the previously published. The method of radial pressure difference distribution showed better agreement with an experimental data previously published. The method is thus very promising.

The optimal position of the propeller and foil incidence are found. The behavior of the flow for a different foil incidences described.

The power consumption on quad-copter in hover decreased by about 20%. The methods of the future study of the ducted fan interference are found.

REFERENCES

[1] S.P. Ostrouhov. *Propellers and Ducted Fans Aerodynamics*. FIZMATLIT, Moscow, 2014.

[2] Arkhipov M.E., Serokvhostov S.V., Semelin A.E., Ageev N.D., and Volkova A.O. Design of ducted fans of small

http://www.imavs.org/

height for multicopter. In *6TH EUROPEAN CONFERENCE FOR AERONAUTICS AND SPACE SCIENCES (EUCASS)*, 2015.

[3] Arkhipov M.E., Serokhvostov S.V., and Streltsov E.V. Experimental and numerical study of propeller in hover. In *2nd Extremal and Record-Breaking Flights of the RPAS and the Aircraft with Electrical Power Plant*, 2014.

http://www.imavs.org/

Specifying Complex Missions for Aerial Robotics in Dynamic Environments

Martin Molina[1]*, Adrian Diaz-Moreno[1], David Palacios[1], Ramon A. Suarez-Fernandez[2], Jose Luis Sanchez-Lopez[2], Carlos Sampedro[2], Hriday Bavle[2] and Pascual Campoy[2]
[1]Department of Artificial Intelligence, Technical University of Madrid, UPM, Spain
[2]Computer Vision Group, Centre for Automation and Robotics, CSIC-UPM, Spain

http://www.imavs.org/

ABSTRACT

To specify missions in aerial robotics, many existing applications follow an approach based on a list of waypoints, which has been proven to be useful and practical in professional domains (agriculture of precision, creation of terrain maps, etc.). However this approach has limitations to be used in other problems such as the one defined in the IMAV 2016 competition (e.g., a search and rescue mission). In this paper we describe a language to specify missions for aerial robotics that tries to overcome such limitations. This language has been designed as part of a complete software framework and architecture for aerial robotics called Aerostack. The paper describes the result of experimental evaluation in real flight and its adequacy for the IMAV 2016 competition.

1 INTRODUCTION

There are a number of available software applications that allow operators to specify a mission for aerial vehicles using a ground control station. Examples of such applications are [1]: MP (Mission Planner), APM Planner 2, MAVProxy, QgroundControl, and PC Ground Station (DJI). However, the use of waypoint lists to specify a mission presents important limitations [1, 2]. One of the most significant limitations is that they are rigid descriptions that are not able to adapt to mission circumstances. The specification is normally based on a fixed list of waypoints that cannot change dynamically (e.g., in the presence of dynamic obstacles). The specification follows a main sequential workflow and it is difficult to specify alternative flows (e.g., iterations, branches, conditional waypoints, alternative flows).

In order to cope with these limitations, mission specification languages can follow more powerful planning approaches. For example, there are classical planning approaches in artificial intelligence (e.g., with STRIPS-like operators). However, to be used successfully in robotics, they need to be adapted or combined with other solutions to react efficiently to the environment changes, monitor mission execution and integrate with reactive behaviors [3]. Some practical planning solutions in robotics are task-based specifications, Petri nets, or behavior-based specifications besides others (rule-based reactive planners or finite state machines)

In this paper, we present the results of our recent research work to create a language to specify complex missions for aerial robotics, such as the IMAV 2016 indoor competition. The goal of this mission is a search and rescue task with different sub-missions related to search and map an unknown environment and transport objects to certain locations.

Our proposed solution has been inspired by the task-based approach for planning but includes also a particular definition of basic concepts (e.g., actions and skills) and it is combined with an event-driven reactive planning approach, to facilitate adaptation to changes of the environment. The solution has been integrated as part of the software framework Aerostack (www.aerostack.org).

The remainder of the paper is structured as follows. Section 2 describes the characteristics of our proposed language. Section 3 describes how to verify the specification. Section 4 presents how our planning system was implemented in the software framework Aerostack. Finally, Section 5 describes the results of the experimental evaluation based on real flights and its ability to specify a mission for the IMAV 2016 competition.

2 MISSION SPECIFICATION IN TML LANGUAGE

We designed the language called TML (Task-based Mission specification Language) using XML syntax to be readable by both humans and machines. TML provides a logical level using more natural and intuitive conceptual notions that avoids the need of specifying technical details at programming level (e.g., what specific processes must be executed for each goal or in which order they must be executed).

Example 1 shows a specification of a simple mission in TML language. In this language, the mission is defined with the tag <mission> and the attribute name and the body of the mission is separated in two main parts: task tree and event handlers.

*Email address: martin.molina@upm.es

[1]Mission Planner http://planner.ardupilot.org/planner/index.html,
APM Planner 2.0: http://planner.ardupilot.org/planner2/index.html,
MAVProxy: http://dronecode.github.io/MAVProxy/html/index.html,
QgroundControl: http://qgroundcontrol.io,
PC Ground Station: http://www.dji.com/es/product/pc-ground-station

http://www.imavs.org/

2.1 Task tree

TML uses the notion of *task* as a basic component to structure a mission with a modular organization. For example, in the case of the IMAV 2016 competition, tasks can identify the different parts of the global mission such as take off from a moving platform, entering the building, pickup objects, etc. The mission is specified with a set of tasks organized in a task tree. Each task is specified using the tag <task> and the attribute {name}.

```
<mission name="Mission example">
  <task name="Initial take off">
    <action name="TAKE_OFF"/>
  </task>
  <task name="Navigate avoiding obstacles">
    <skill name="RECOGNIZE_VISUAL_MARKERS"/>
    <skill name="AVOID_OBSTACLES"/>
    <task name="Stabilize at origin">
      <action name="STABILIZE"/>
    </task>
    <task name="Go to destination">
      <action name="GO_TO_POINT">
        <argument name="point"
                  value="(3.0,4.0,1.5)"/>
      </action>
    </task>
  </task>
  <task name="Final landing">
    <action name="LAND"/>
  </task>
  <event_handling>
    <event name="Land command recognized">
      <condition parameter="visualMarker"
                 comparison="equal" value="3"/>
      <action name="LAND"/>
      <termination mode="ABORT_MISSION"/>
    </event>
  </event_handling>
</mission>
```

Example 1: Mission specification in TML language.

We use the concept of *action* to express an elementary goal that the aerial robot is able to achieve by itself, using its own actuators. An illustrative set of actions related to IMAV 2016 competition is the following: take off, go to a point, move forwards, explore space, pick up item, drop item, rotate yaw, and land.

To represent a particular robot's ability we use the concept of *skill*. An illustrative set of skills related to IMAV 2016 competition is the following: avoid obstacles, self locate by markers, recognize buckets and recognize items. Skills can be active or inactive in a particular robot. In general, skills have influence in the behavior of actions. Thus, we can understand skills as global modifiers for sets of actions.

The notion of skill is useful as an intuitive concept to help operators express more easily what complex abilities should be active, without considering low-level technical details. Internally, a skill is automatically supported by a set of running processes. Thus, the activation of skills is associated to the increase of resource consumption (memory space, processing time, battery charge) so it is important to deactivate unnecessary skills when it is possible.

In TML, the body of a terminal task (i.e., a task that is a terminal node in the task tree) must specify an action with the tag <action> and the attribute {name}. An action can include optionally arguments. Arguments use the tag <argument> and the attributes {name, value}. The body of a terminal task can also specify several skills to be active with the tag <skill> and the attribute {name}.

Action	Description
FLIP	The vehicle performs a flip in a certain direction (argument direction with the values {front, back, right, left}). The direction by default is to the front.
GO_TO_POINT	The vehicle moves to a given point. The point can be expressed using absolute coordinates (argument point) or coordinates relative to the vehicle (argument relative point).
LAND	The vehicle descends vertically (through the z axis) until it touches the ground. It is assumed that the ground is static.
ROTATE_YAW	The vehicle rotates the yaw a number of degrees (argument angle).
STABILIZE	The vehicle tries to cancel all the perturbations and turbulences that may affect its system such as movement speeds and attitude speeds.
TAKE_OFF	The vehicle takes off from its current location to the default altitude. If the vehicle is flying, this action is ignored.
WAIT	The vehicle waits on the air for a specified number of seconds (argument time).

Table 1: List of actions (partial) used in TML language.

The body of a non-terminal task can include skills that are active during the task execution. The body of a non-terminal task does not include actions. Instead, the body includes one or several simpler tasks that are executed in sequence.

The linear sequence of task execution can be modified with repetitions and conditioned executions. The tag <repeat> and the attribute {times} is used to repeat a task several times. The tag <condition> and the attributes {parameter, comparison, value} are used to establish a condition to execute a task. The allowable values for the attribute comparison are {equal, less than, less than or equal to, greater than, greater than or equal to, not equal to}. Example 2 illustrates the use of these tags.

```
<task name="Flip and move if green">
  <task name="Flip three times">
    <repeat times="3"/>
    <action name="FLIP"/>
  </task>
  <task name="Move if green">
    <condition parameter="observed color"
               comparison="equal" value="green"/>
    <action name="GO_TO_POINT">
      <argument name="point"
                value="(3.0, 4.2, 1.5)"/>
    </action>
  </task>
</task>
```

Example 2: Task body with conditions in TML language.

2.2 Event Handlers

We have combined the task-based solution with reactive planning to provide more adaptability to changes of the environment. To represent conditions about the world state, we use the concept of event. An event is the occurrence of a significant change in the state of the external environment or

the state of the own robot. Events can be related to normal situations (e.g., the detection of a specific visual marker) or abnormal/undesired situations (time out, discharged battery, etc.).

In TML, the mission specification includes a section for event handlers defined with the tag <event_handling> to describe what to do in the presence of events. Event handlers are similar to condition-action rules, a typical representation used in reactive planners. Events are defined with the tag <event> and the attribute {name}. Each event includes a list of conditions (in conjunctive form), a list of actions to be done and an ending step defined with the tag <termination> and the attribute {mode}. Events can use a particular type of condition with the attribute {currentTask}. This is useful to express that the event happens while a particular task is being executing. Example 3 illustrates this case in TML language.

```
<event name="take off failure">
  <condition currentTask="initial take off"/>
  <condition parameter="action confirmation"
             comparison="equal" value="failure"/>
  <termination mode="ABORT_MISSION"/>
</event>
```

Example 3: Event declaration in TML language.

The task tree and the event handlers are integrated during the execution time in the following way:

1. The interpreter navigates in the task tree following a depth-first control strategy. When the interpreter reaches a terminal node of the tree, it requests the execution of the corresponding action with the active skills. The order of this control strategy can be modified by the user with repetition and condition sentences.

2. While the interpreter waits until a requested action finishes, it verifies the conditions of event handlers. If an event handler verifies its condition, it can request additional skills to be active, can request additional actions to be done or can change the normal execution flow (abort mission, abort task, jump to task, etc.).

3 VERIFICATION OF THE MISSION SPECIFICATION

The text in TML language written by the operator must be automatically analyzed to verify that it does not include errors generated by mistakes or misunderstood concepts. This is important to guarantee a robust operation avoiding system crashes. In general, the verification of a mission specification can include two types of analysis:

- Language validation. This corresponds to the lexical, syntax and semantic validation of the mission specification according to the grammar defined for the TML language.
- Physical feasibility verification. This verification checks if the specified mission can be performed in practice considering constraints of the physical world. This validates, for example, that a point to reach is not too close to an obstacle or that the distance to cover is not too long for the remaining battery.

The verification of physical feasibility is a complex task with different dimensions. It includes (1) the *individual verification* of the feasibility of each particular action, (2) the *contextual verification* to check the feasibility of an action or a skill in relation to other skills and actions that occur at the same time (3) the *temporal verification* of the complete mission taking into account the temporal evolution of the whole set of actions and skills.

The relative time when the verification is performed is another significant distinction. The verification can be done before the mission starts. For example, it is possible to verify in advance that a certain spatial point is too far to be reached, considering the maximum charge of battery. On the other hand, the verification can be done *during* the mission execution, considering unexpected changes in the environment such as the moving obstacles, bad illumination, etc.

We designed a solution to verify the physical feasibility using a computational model with a constraint-based representation. This model uses variables, parameters, functions and constraints. Variables $\{x_i\}$ represent the dynamic values of physical references and magnitudes (e.g., destination point, current charge of battery, etc.). Parameters $\{k_i\}$ represent constant values for physical magnitudes related to the performance of the robot such as maximum speed, battery consumption rate of the vehicle, etc. Parameters can be divided into vehicle-independent parameters (general for any kind of vehicle) or vehicle-specific for each category of vehicle. Functions represent spatio-temporal and motion functions (see Table 2) such as the length of a trajectory, distance to the closest obstacle, maximum distance covered with certain battery charge, required speed to reach a point at certain time, etc.

Function	Description
Distance(x_1, x_2)	Distance from point x_1 to point x_2
DistanceBattery(x_1, x_2)	Maximum distance covered with battery charge x_1 and consumption rate x_2
DistanceObstacle(x_1)	Distance from point x_1 to the closest obstacle
Length(x_1)	Length of trajectory x_1
Speed(x_1, x_2, x_3)	Required speed to departure now from point x_1 and arrive at point x_2 at time x_3
Trajectory(x_1, x_2)	Trajectory from point x_1 to point x_2 (generated by a trajectory planner)

Table 2: Example functions used in the verification model.

Constraints $\{c_i\}$ are conditions about the physical world that must be satisfied. Examples of these conditions are: the destination point must be safe from obstacles, there must be enough battery for the movement, and the destination point must be reachable at an acceptable speed. The previous conditions are represented with the following three constraints (with functions, variables and parameters):

c_1: *DistanceObstacle*$(x_2) > k_2$

http://www.imavs.org/

c_2: *Length(Trajectory*$(x_1, x_2)) < $*DistanceBattery*$(x_3, k_1)$
c_3: *Speed*$(x_1, x_2, x_4) < k_3$

where the variable x_1 is the current point, x_2 is the destination point, x_3 is the current battery charge, and x_4 is the planned time of arrival to the destination; and the parameter k_1 is the battery consumption rate, k_2 is the minimum free acceptable space between obstacle and vehicle, and k_3 is the maximum speed of the vehicle.

The whole set of constraints is divided in subsets according to the categories of actions. For example, there is a subset of constraints for actions related to rotation motions, another subset of constraints for actions related to translation motions, etc. For a given action, the verification procedure reviews only the subsets of constraints that correspond to categories to which the given action belongs.

This type of model is generic to be reusable for different physical platforms. Only the vehicle-specific parameter values must be manually configured.

4 THE TML INTERPRETER

We implemented an interpreter of the TML language with a set of computational processes in the software framework Aerostack. This section summarizes Aerostack and the set of processes that support the interpretation of the TML language.

4.1 The Aerostack software framework

Aerostack [4] is a software framework for aerial robotics, with a multilayered architecture supported by a distributed inter-process communication. Aerostack uses an architecture based on an extension of the hybrid reactive/deliberative paradigm, i.e., an architecture that integrates both a deliberative and reactive approaches [5]. The architecture includes five layers (Figure 1):

- The reactive layer with low-level control with sensor-action loops.
- The executive layer that accepts symbolic actions from the deliberative layer and generates detailed behavior sequences for the reactive layer. This layer also integrates the sensor information into an internal state representation.
- The deliberative layer generates global solutions to complex tasks using planning.
- The reflective layer helps to see if the vehicle is actually making progress to its goal and helps to react in the presence of problems (unexpected obstacles, faults, etc.) with recovery actions.
- The social layer includes communication abilities, which is important to establish an adequate communication with human operators and other robots.

The first three layers correspond to the popular hybrid design known as the three-layer architecture [6]. The reflective layer is based on cognitive architectures [7, 8] to simulate certain self-awareness able to supervise the other layers. The social layer has been proposed in multiagent systems and other architectures with social coordination (e.g., [9]). It is worth to highlight that the higher level layers, reflective and deliberative, must work slower (0.1 - 10 Hz.) than the lower level layers, executive and reactive (10 - 1000 Hz.).

4.2 Processes for mission planning

To implement an interpreter for our approach for mission specification, we have upgraded Aerostack, reusing some of the available components, and programming and integrating some additional ones. The processes that support the planning system perform the following tasks asynchronously, generating outputs when they receive new inputs that requires an action to be taken (Figure 2):

- The *mission planner* interprets the TML specification written by the human operator. The mission planner uses such a specification to generate step by step the next action to perform and, optionally, the set of skills to be active, considering the current state of the environment.
- The *action specialist* verifies the physical feasibility of actions and predicts the performance of actions (e.g., time required, distance to cover, required charge of battery, etc.).
- The *trajectory planner* generates collision-free trajectories that can be followed by the drone.
- The *yaw planner* generates a yaw value for the aerial platform specified either as a point to look or as a yaw to look.

The planning system communicates with processes that perform the following tasks:

- The *manager of actions and skills* accepts the actions and skills requested by the mission planner, expressed as symbolic descriptions (e.g., take off, move to a point, etc.), and translates them into specific orders for the motion controllers and the activation of certain running processes. It responds asynchronously to the requests.
- The *action monitor* supervises the correct execution of initiated actions and detects either when they are completed or when they have failed. It works synchronously, monitoring the actions actively.
- The *event detector* detects the presence of significant events to be considered by the mission planner (e.g., the recognition of a particular visual marker). It synchronously monitors the considered events.
- The *problem manager* collects errors and initiates the corresponding recovery actions. The detected problems are informed to the mission planner to react in consequence. It watches out for new problems synchronously.

We reused the trajectory planner and the yaw planner (see [10] for details about these components) and we adapted the

http://www.imavs.org/

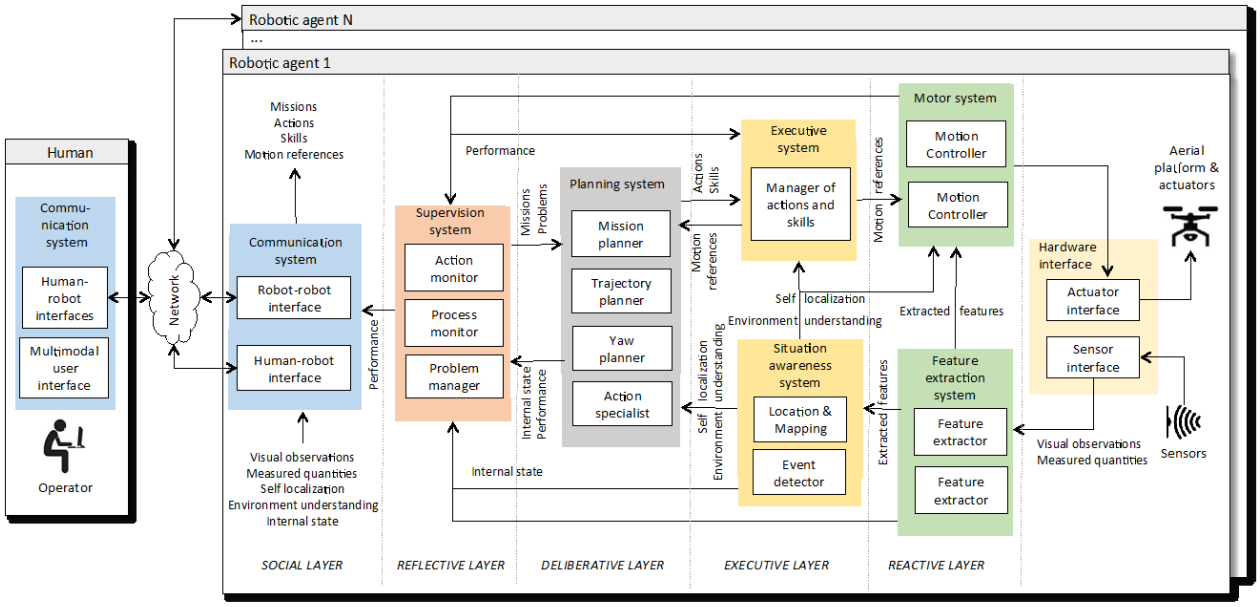


Figure 1: Main components of the Aerostack multi-layered architecture [4]

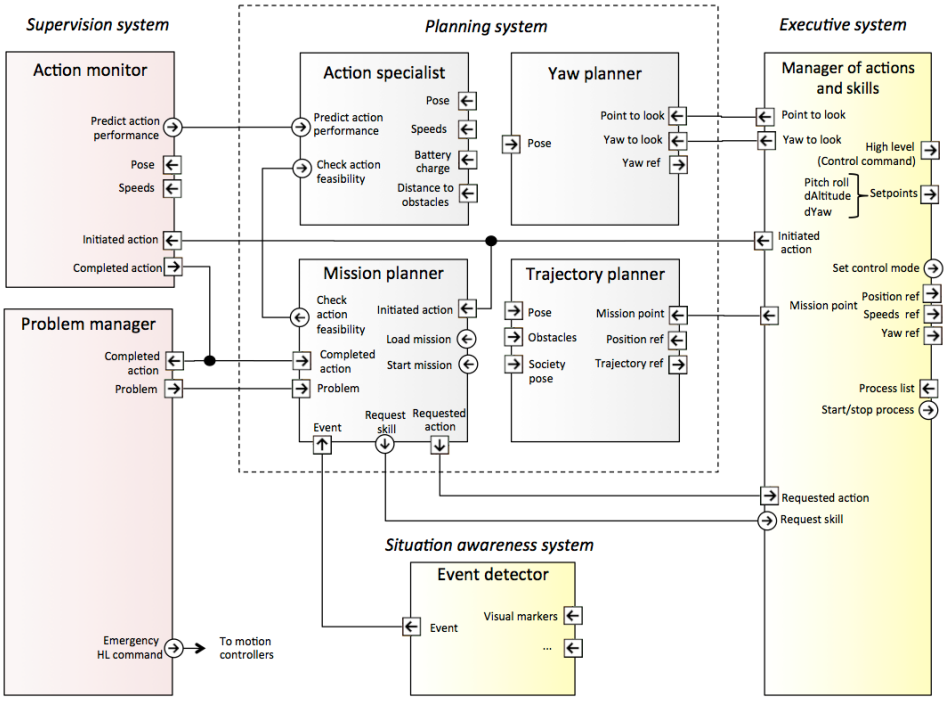


Figure 2: Detail of processes used in Aerostack to interpret the TML mission specification language.

http://www.imavs.org/

manager of actions and skills. Since Aerostack is a software framework platform independent, the resulting mission planning system also has this property. Only certain platform-dependent parameters must be configured.

5 EVALUATION

This section presents the results of evaluations that we performed for the TML language: (1) flight tests and (2) mission specification for the IMAV 2016 competition.

5.1 Flight tests

The TML language was tested in real flight using different mission cases with various degrees of complexity. This section describes one of these cases, whose formulation was based on a surveillance mission. The goal of this mission is to search for a subject in a spatial area avoiding any type of obstacle. When the aerial robot recognizes the presence of the subject, it performs an alert action (e.g., notify the location of the object, start an alarm sound, etc.) and returns to the starting point.

For the sake of simplicity, we defined an indoor mission where the robot performs a search routine (a squared trajectory with a side of 2 meters) and detects obstacles or the searched subject using visual markers (ArUco markers). These simplifications facilitate the practical development of the real flight and allow testing the functionality of the TML language. The experiment was carried out with the aerial platform Parrot AR Drone 2.0. We used the same specified mission considering several scenarios with changes in the environment with different locations of obstacles and searched subject. Figure 3 shows graphics representing the trajectories followed by the robot in the different scenarios.

The described experiment demonstrates that the TML language can be used to specify and execute a mission for an aerial vehicle in a dynamic environment. The experiment shows how the robot adapts correctly its behavior to changes in the environment, considering different locations of obstacles and the searched subject.

It is worth to clarify that this behavior was achieved in two different decision planes. On the one hand, the process *mission planner* supports an adaptive behavior corresponding to the moment when the subject is detected, according to the specified reaction written explicitly in TML language (return to the starting point).

On the other hand, the process *trajectory planner* supports an adaptive behavior corresponding to the presence of obstacles, generating collision-free trajectories to be followed by the robot. In summary, this experiment corresponds to a representative type of mission based on surveillance but presents also similarities to other types of missions (e.g., search and rescue, terrain mapping, etc.) where the aerial robots must show certain degree of autonomy.

5.2 IMAV 2016 Competition

We analyzed the ability of the TML to be used as specification language for the mission of the IMAV 2016 indoor competition. This mission includes partial goals such as taking off from a moving platform, entering the building via different entrance (doorway, chimney or window), pickup objects and release them to the correct locations, exiting the building, landing on the moving platform, and mapping of the indoor environments.

Appendix A shows an example of specification that we wrote in TML language for this mission. For this specification, we made certain assumptions about actions and skills. For example, we assumed that the robot is able to do specific actions (pick up item, drop item), specific skills (recognize bucket, recognize item), or general actions (go to a rectangle identified by a colored frame, like the door in the competition, or explore the space building a map).

According to these assumptions, we were able to write in TML the corresponding IMAV 2016 mission, which demonstrates that this language provides enough expressive power to formulate such a type of missions. In particular, tasks were useful to structure modularly the different parts of the mission: take off at operating zone, enter building, explore building, etc. Event handlers were useful to react to recognized items and buckets while the robot is exploring the building. The concept of actions and skills were also useful to model the different abilities of the robot for this mission.

6 CONCLUSION

In this paper, we have described TML, a language that follows a task-based approach to specify missions for aerial robotics. The TML language uses the concept of task as a basic notion to structure the mission and other intuitive concepts (actions, skills, events) that can help to be easily learned and used by operators.

We implemented a TML interpreter as part of the Aerostack software architecture. Thanks to the use of this framework, the implementation of the TML interpreter is independent on aerial platforms and able to be used in real flights. The TML interpreter is freely available for the research community as part of the Aerostack architecture (www.aerostack.org). The evaluation of TML showed that this language is expressive enough to specify a mission such as the one defined for the IMAV 2016 competition. The real flight experiments with the interpreter of TML in Aerostack showed also an adequate adaptation to environment changes.

Our future work includes adding improvements to this language such as new and more complex actions and skills. We also plan to build a graphical user interface that help operators to specify missions based on TML. This interface can use graphical notations and restricted menu options. Another line of future work is to combine TML with a multi-agent approach to support distributed and coordinated planning (e.g., [11]).

http://www.imavs.org/

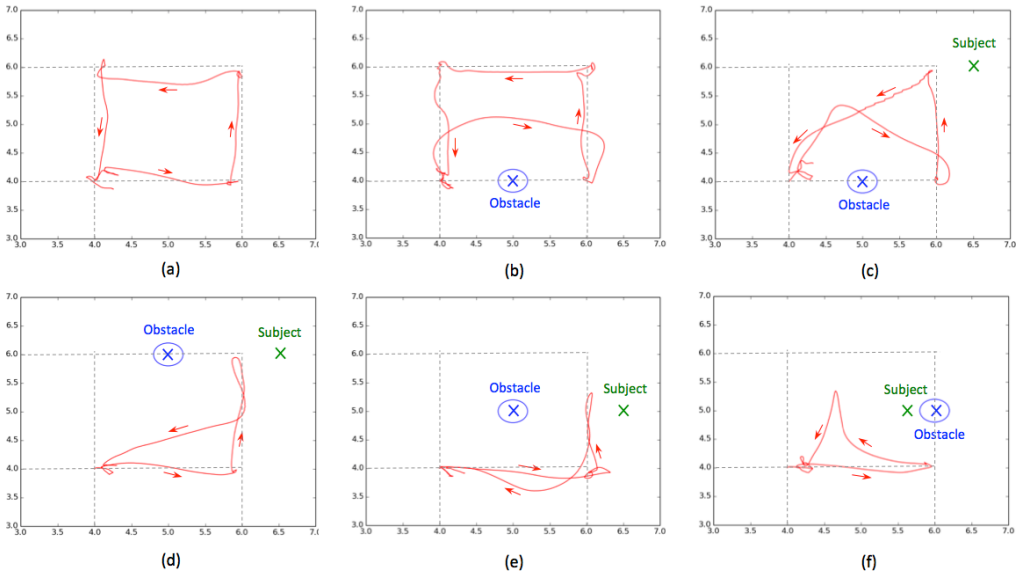


Figure 3: This figure shows different trajectories followed by our aerial robot in real flights during the execution of the same mission specified in TML language. The experiment shows how the robot adapts correctly its behavior to changes in the environment, considering different location scenarios of an obstacle and the searched subject. In the figure, the origin of the movement is the point (4.0, 4.0). The graphic (a) corresponds to the squared trajectory followed by the robot in a scenario without any influence from the environment.

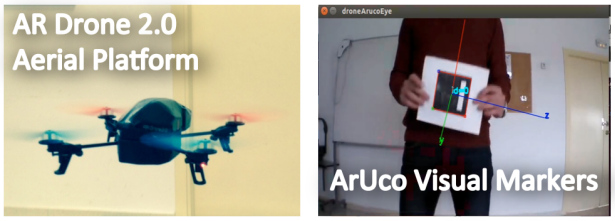


Figure 4: Platform and markers used in our experiments.

http://www.imavs.org/

ACKNOWLEDGEMENTS

This research work has been supported by the Spanish Ministry of Economy and Competitiveness through the project VA4UAV (Visual autonomy for UAV in Dynamic Environments), reference DPI2014-60139-R.

REFERENCES

[1] E. Santamaria, P. Royo, C. Barrado, E. Pastor, J. Lpez, and X. Prats. Mission aware flight planning for unmanned aerial systems. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Honolulu (HI), 2008.

[2] B. Schwartz, L. Nagele, A. Angerer, and B. A. MacDonald. Towards a graphical language for quadrotor missions. In *Workshop on Domain-Specific Languages and models for Robotic systems*, 2014.

[3] A. L. Rothenstein. *A Mission plan specification language for behaviour-based robots*. PhD thesis, University of Toronto, 2002.

[4] J. L. Sanchez-Lopez, R. A. Suarez Fernandez, H. Bavle, C. Sampedro, M. Molina, J. Pestana, and P. Campoy. Aerostack: An architecture and open-source software framework for aerial robotics. In *The 2016 International Conference on Unmanned Aircraft Systems ICUAS*, Arlington, VA, 2016.

[5] R. C. Arkin, E. M. Riseman, and A. Hansen. Aura: An architecture for vision- based robot navigation. In *proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, 1987.

[6] E. Gat. On three-layer architectures. In David Kortenkamp, R. Peter Bonnasso, and Robin Murphy, editors, *Artificial Intelligence and Mobile Robots*. AAAI Press, 1998.

[7] A. Sloman. What sort of architecture is required for a human-like agent. In M. Wooldridge and A. Rao, editors, *Foundations of Rational Agency*. Kluwer Academic Publishers, 1999.

[8] R. J. Brachman. System that know what they're doing. *IEEE Intelligent Systems*, 17(6):67–71, 2002.

[9] B.R. Duffy, M-Dragone, and D.M.P. O'Hare. The social robot architecture: A framework for explicit social interaction. In *In Proceedings Cognitive Science Workshop, Android Science-Towards Social Mechanisms*, Stresa, Italy, 2005.

[10] J.L. Sanchez-Lopez, J. Pestana, P. de la Puente, R. Suarez-Fernandez, and P. Campoy. A system for the design and development of vision-based multi-robot quadrotor swarms. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 640–648, May 2014.

[11] C. Sampedro, H. Bavle, J. L. Sanchez-Lopez, R. A. Suaarez-Fernandez, A. Rodrguez-Ramos, M. Molina, and P. Campoy. A flexible and dynamic mission planning architecture for uav swarm coordination. In *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, Arlington, VA, 2016.

APPENDIX A: MISSION FOR IMAV 2016

This appendix shows a mission specification in TML language for IMAV 2016 Indoors Competition. This corresponds to an example where only one item is picked up.

```
<mission name="Indoor Competition IMAV 2016">
  <skill name="AVOID_OBSTACLES"/>
  <!-- TASK: Take off at operating zone -->
  <task name = "Take off at operating zone">
    <skill name="SELF_LOCATE_BY_MARKERS"/>
    <task name = "Take off to start the mission">
      <action name="TAKE_OFF"/>
    </task>
    <task name = "Memorize operating zone">
      <action name="MEMORIZE_POINT"/>
        <argument name="coordinates"
                  label="operating zone"/>
    </task>
  </task>
  <!-- TASK: Enter building -->
  <task name = "Enter building">
    <task name = "Go to the door">
      <action name="GO_TO_RECTANGLE">
        <argument name="frame color" value="red"/>
      </action>
    </task>
    <task name = "Enter through the door">
      <action name="MOVE_FORWARDS">
        <argument name="distance" value="1.5"/>
      </action>
    </task>
    <task name = "Memorize where is the door">
        <action name="MEMORIZE_POINT">
            <argument name="coordinates"
                      label="door coordinates"/>
        </action>
    </task>
  </task>
  <!-- TASK: Explore building -->
  <task name = "Explore building">
    <skill name="RECOGNIZE_ITEMS"/>
    <skill name="RECOGNIZE_BUCKETS"/>
     <action name="EXPLORE_SPACE">
       <argument name="build map" label="yes"/>
    </action>
  </task>
  <!-- TASK: Transport item A -->
  <task name = "Transport item A">
    <condition label="item A coordinates"
               comparison="known" value="yes"/>
    <condition label="bucket A coordinates"
               comparison="known" value="yes"/>
    <task name = "Go to item A">
        <action name="GO_TO_POINT">
            <argument name="coordinates"
                      label="item A coordinates"/>
        </action>
    </task>
    <task name = "Pick up item A">
        <action name="PICK_UP_ITEM"/>
    </task>
    <task name = "Go to bucket A">
        <action name="GO_TO_POINT">
            <argument name="coordinates"
                      label="bucket A coordinates"/>
        </action>
    </task>
    <task name = "Drom item A">
        <action name="DROP_ITEM"/>
    </task>
  </task>
  <!-- TASK: Exit building -->
  <task name = "Exit building">
    <task name = "Return to the door">
      <action name="GO_TO_POINT"/>
        <argument name="coordinates"
                  label="door coordinates"/>
    </task>
    <task name = "Correct orientation">
      <action name="ROTATE_YAW"/>
        <argument name="angle" label="90"/>
    </task>
    <task name = "Exit building through the door">
      <action name="MOVE_FORWARDS"/>
        <argument name="distance" value="1.5"/>
    </task>
  </task>
  <!-- TASK: Land at operating zone -->
  <task name = "Land at operating zone">
    <skill name="SELF_LOCATE_BY_MARKERS"/>
    <task name = "Go to operating zone">
      <action name="GO_TO_POINT"/>
        <argument name="coordinates"
                  label="operating zone"/>
    </task>
    <task name = "Final landing">
      <action name="LAND"/>
    </task>
  </task>
  <!-- Event handlers -->
  <event_handling>
  <!-- EVENT: Item A found -->
  <event name="Item A found">
      <condition current_task="Explore building"/>
      <condition parameter="Item A found"
                 comparison="equal" value="yes"/>
      <action name="MEMORIZE_POINT">
        <argument name="coordinates"
                  label="item A coordinates"/>
      </action>
      <termination mode="CONTINUE"/>
    </event>
  <!-- EVENT: Bucket A found -->
    <event name="Bucket A found">
        <condition current_task="Explore building"/>
        <condition parameter="Bucket A found"
                   comparison="equal" value="yes"/>
        <action name="MEMORIZE_POINT">
            <argument name="coordinates"
                      label="bucket A coordinates"/>
        </action>
        <termination mode="CONTINUE"/>
    </event>
  </event_handling>
</mission>
```

Example 4: Mission specification file for IMAV 2016.

http://www.imavs.org/

Spring for clapping X-wing Micro Air Vehicles

Yao-Wei Chin, Yu-Kai Luo, Zi-Yuan Ang, and Gih-Keong Lau

School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 639798

http://www.imavs.org/

ABSTRACT

Four-winged ornithopters like Delfly make use of clap and fling to enhance thrust generation. Yet, clapping of the four wings (forming a cross) requires high energetic cost and their wing kinetic energy is not recovered using rigid-body mechanism. To help recover the wings' kinetic energy, we designed and developed a compact T-shaped elastic flexures, replacing a pivot revolute joint for this X-wing micro air vehicles (MAV). The elastic mechanism is made of flexible polyimide film and rigid carbon fiber plates. The polyimide film is capable of storing elastic strain energy for large bending. Higher elastic storage in the film hinges improved the clapping speed of the X-wing flapper to generate higher thrust without incurring additional energetic costs. Hence, it is useful to enhance the thrust of low-frequency clappers. This X-wing clapper of a 13.4g total weight is able to generate thrust up to 20.7g force and demonstrates tethered hovering and climbing.

1 INTRODUCTION

Dragonfly are agile flyers which have two wing pairs in tandem. Their four-wing configuration inspired man-made flapping-wing flight vehicles or ornithopters with four wings. The four wings are two criss-cross wing planes. When the two criss-crossing wing planes are opened, their leading edges crosses like an X. When the two wing planes are close, their two leading edges becomes parallel and their wing foils overlap. Examples of the X-wing ornithopter include the Luna [1], which is powered by a long twisted rubber band, and the DelFly [2], which is powered by an electric motor. Four-winged ornithopters have good flight stability by moving the two pairs of wings in countering directions, minimizing the body rocking [2]. In contrast, two-winged ornithopters are more prone to body rocking due to asymmetric inertial force induced by flapping wings.

Typically, an electric motor via a double crank mechanism was used to drive a scissor-like bi-plane or X wing clappers, such as Delfly or Silverlit Wing-Master I-Bird. Two pairs of X wings claps together once or twice during either ends of wing stroke. For example, X-wing Delfly adopted the clap-and-fling mechanism to increase the thrust-power ratio (a measure of efficiency) by about 50% [2, 3]. Each of the two wing planes is pivoted about a center revolute joint, which consists of a hollow sleeve and a pin shaft. Such joint is subjected to friction and unable to store elastic energy. One possible solution is to use elastic elements to convert the kinetic energy of the wings into elastic energy which is subsequently recovered during stroke reversal.

Elastic energy storage has been used in non-clap two-winged ornithopters for power saving [4, 5]. For example, Baek and Fearing [5] integrated a discrete helical spring to the crank-slider mechanism for two flapping wings and showed experimentally that the measured power saving is as much as 19%, near the resonant frequency [5]. However, there was no compact spring to be integrated into the limited space at the pivot of the four-winged clapping mechanism. Traditional cross flexures take space more than a pivot does. Hence, the objective of this research is to develop a compact flexural mechanism to help recover wing kinetic energy and assist faster clapping of X-wing FWMAV. In this way, the X-wing FWMAV should be more energy efficient while maintaining or even improving its thrust generating capability.

Hovering is a challenge for FWMAV because the wings have to beat fast and large stroke to produce sufficient thrust to lift its weight. Heavier FWMAVs must either beat larger wings or beat wings faster to hover, hence expends much higher kinetic energy. Elastic energy storage is in need to recover the higher wings' kinetic energy by using stiffer spring. Yet there are challenges to design a compact, stiff, yet durable spring for the X-wing FWMAV because the large-stroke bending and high reciprocating torque may break the elastic hinges.

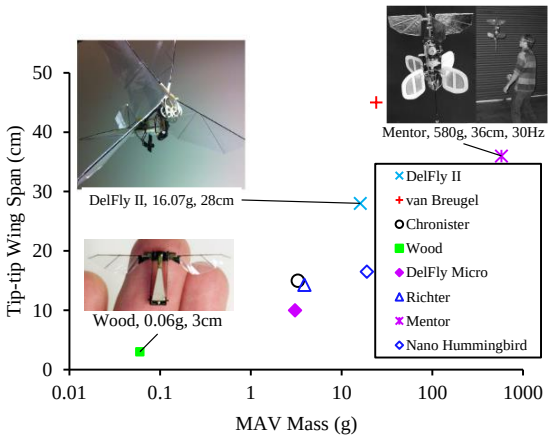


Figure 1 Wingspan and mass of flapping wing micro air vehicles reported or claimed to hover or fly slowly, from [1] [6] [7] [8] [9] [10].

2 PRINCIPLE AND DESIGN

Flight by flapping wings was expected to consume more energy than rotary flight because kinetic energy of flapping wings is lost within a quarter cycle of acceleration and deceleration. In a quarter wing-beat cycle, motor accelerates the wings to maximum angular speed with maximum wings' kinetic energy at the mid stroke. However, this wing kinetic energy is lost when motor decelerates the wings to a stop at the wing's extreme position.

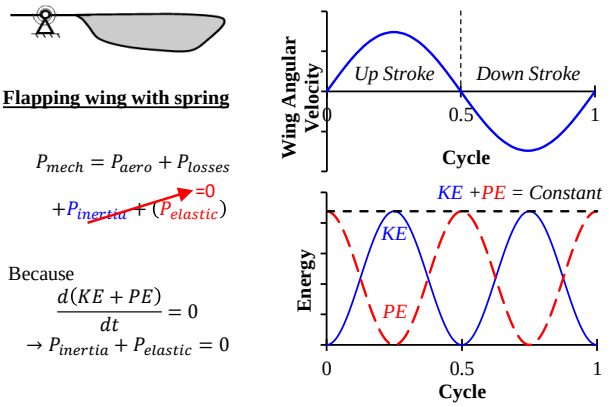


Figure 2 Kinetic energy (KE) lost in wing reciprocation can be recovered with spring hinge that can store equivalent elastic energy (PE).

The loss of wing kinetic energy can be stored as elastic energy in a spring hinge for wings instead. The spring hinge recovers kinetic energy when it elastically deforms from neutral mid stroke to slow down the wing towards end stroke. In a subsequent quarter stroke, its energy release assist the wing reversal to mid stroke. For full energy recovery, the capacity of elastic storage (i.e. of spring) needs to match the wings' maximum kinetic energy change. For a wing of rotation inertia J_0, which beats sinusoidally with a stroke of ϕ at a wingbeat frequency f, its maximum kinetic change over a quarter wingbeat cycle is:

$$\mathrm{KE}_{wing,max} = \frac{1}{2}\pi^2\phi^2 f^2 J_0 \quad (1)$$

A wingbeat cycle of the X-wing consists of clap and fling strokes. The wing kinematic varies over the quarter cycles of clapping and flapping. More specifically, the maximum wing speed of the clapping quarter cycle is different from that of the flinging quarter cycle. This is due to the non-linear crank-rocker drive and elastic hinge assistance. The total kinetic energy change over a wingbeat cycle is a sum of the kinetic energy changes during the wing clapping and during the wing flinging:

$$Total\ \mathrm{KE}_{wing,max} = \frac{1}{2}J_0\left(\omega^2_{max,clap} + \omega^2_{max,fling}\right) \quad (2)$$

Spring design is not trivial work for an X-wing clapping mechanism. The spring must be compact, lightweight and tough enough. Coil springs are not suitable because they take up too much space. Instead, frictionless leaf springs like film flexures are more suitable to replace revolute joints. To enable large wing stroke, the flexural hinges need to bend more than 90° without breaking. Thin metal sheet, which made stiff springs, break when subjected to such a large angle of bending. Flexible polyimide film hinge bordered by rigid plates of carbon fiber reinforced polymer were found useful for Robobee [11] [7] [12] of less than 12cm wing span. It was not clear if the film hinges can be scaled up to store more elastic storage for larger FWMAV [13] of greater than 24cm wing span. This work shows that polyimide film hinges work for greater-span wings and can store elastic storage for X wing (two cross wing planes) when the hinges are configured in a compact T shape (see Figure 3).

The T-shaped compliant mechanism consists of a mirror-pair of polyimide flexural hinges. Each polyimide hinge connects two rigid plates, namely a wing base and a vertical ground. The polyimide hinge acts a frictionless pivot for the wing base, which swings about the vertical ground at the symmetry plane. Each wing base carries a wing plane that spans both sides of the pivot. The wing plane has its leading edge of wing foil reinforced by a carbon spar. The other wing plane has it leading edge offset from the other one so that the cross spars of the two wing planes do not collide with each other.

Upon assembly, the hinges are bent 90 degrees from the vertical. Elastic recoil of the hinges assists clapping of the opposite wing planes at the horizontal plane. In the subsequent quarter stroke of flinging (reverse to clapping), a force is needed to peel the clapped wings apart and to deform the elastic hinge. While a wing rotates by an angle of θ from the horizon, the elastic flexure bends an angle of β. The elastic energy storage stored in the flexure is the work done to the flexure:

$$W_{elastic} = \int_{\beta_1}^{\beta_2} K\beta\, d\beta \quad (3)$$

where K is the torsional stiffness of the flexure, which is the ratio of bending moment to the bending angle given as:

$$K = \frac{dM}{d\beta} = \frac{EI}{L},\ where\ I = \frac{bh^3}{12} \quad (4)$$

where E is the Young modulus, I is the area moment of inertia, L is the length, b is the width and h is the thickness of the polyimide film used as the flexure. The polyimide film used is 0.127mm (5mil) thick Kapton HN (E=2.5 GPa).

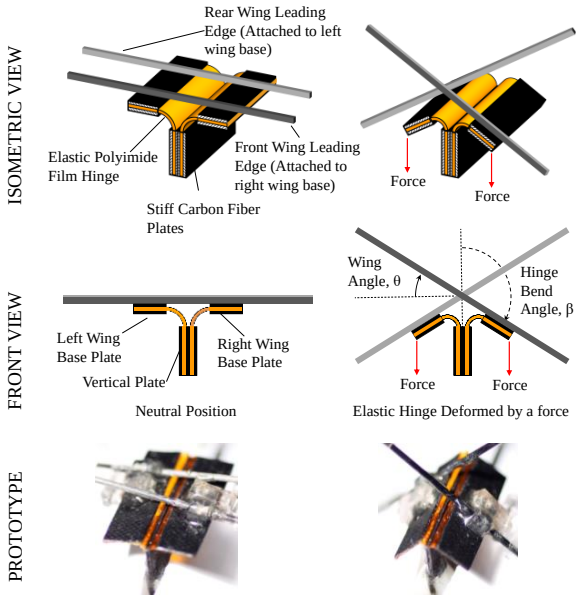


Figure 3 X-wing clapper with elastic hinge. The elastic hinge tends to return the wing base plate to vertical attitude, but is stopped at the horizontal by the opposite leading edge. As such the wings clapped together at the horizon. A force is required to rotate the wing base downwards to fling open the wings.

http://www.imavs.org/

http://www.imavs.org/

This elastic energy stored in the flexure is released upon reversal of the wing stroke. Release of the elastic energy speeds the wings to clap at the horizontal neutral position. The higher the elastic storage, the faster the wing claps. To show the effect of increasing elastic energy storage, we made three sets of hinges: 5 mm, 10 mm, and 20 mm, given the same 1mm hinge length (See Figure 6). Too stiff a hinge may elastically resist the start of a rotary motor.

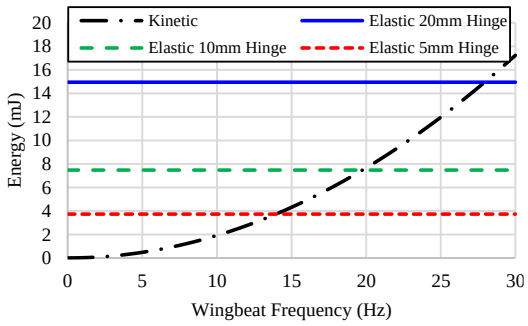


Figure 4 Calculated wing kinetic energy of a wing pair (sinusoidal wing stroke assumption) and the elastic energy storage in a single PI hinge with each different hinge width for a wing stroke of 50° from horizon.

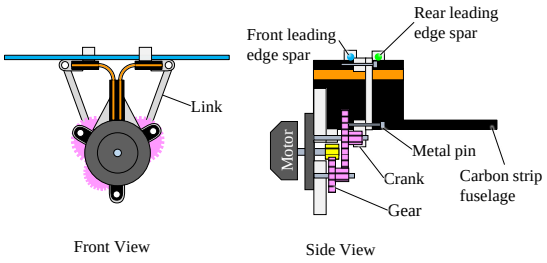


Figure 5 The double crank rocker mechanism with brushless motor and gearbox with 1:10.67 for the clapping mechanism.

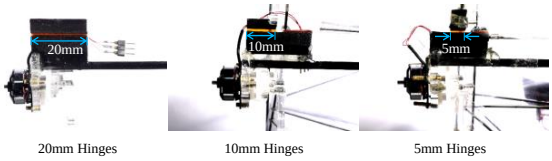


Figure 6 Elastic clappers with hinge width decreasing from 2cm to 0.5cm to show the effect of elastic stiffness.

The X-wing consists of two overlapping sheets of 25µm thick Mylar foils, which are reinforced spanwise by 1mm thick carbon rods and chordwise by 0.5mm thick carbon rods. Each sheet of foil spans the left and right sides of a middle pivot. (Figure 7a). A half of the foil has its leading edge attached to the front spar, while the other half has its leading edge attached to the rear spar. The attachment is done by slotting the rod through each foil sleeve. As the two spars criss-cross open, the foil is bent to form a V-shape about the middle pivot. (Figure 7b). The wing model in SolidWorks estimates J_0 as 5097g.mm^2 about the rotational axis at foil centre. Assuming sinusoidal wing stroke, the kinetic energy of the wings increases with the second order of wingbeat frequency according to equation (1), as shown in Figure 4.

Actuation of the wing base plates is provided through a double crank-rocker mechanism, driven by a DC brushless motor (Hobbyking AP03 7000kv) (See Figure 5). The output torque of the brushless motor is amplified 10.67 times through a speed reduction gearbox to spin the 3mm crank. The rigid linkages and the body frame to hold the gear shafts and motor are made of laser cut acrylic blocks.

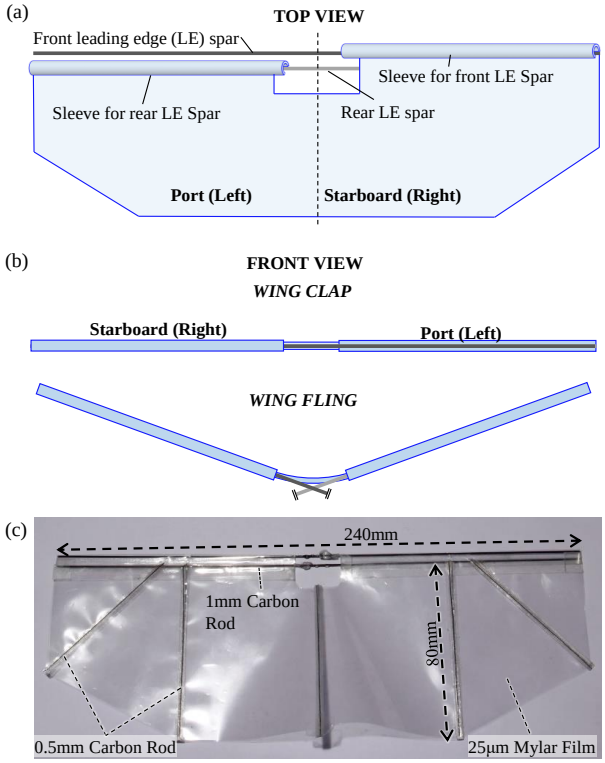


Figure 7 (a) Top view of the top wing foil design. Front LE spar is slotted into the starboard sleeve of the foil. Rear LE Spar is slotted into the port sleeve of the foil (b) Front view of the top wing foil when the wings clap and fling open. The top foil is folded to a V-shape by the LE spars. (c) X-wing with two overlapping sheets of wing foils with reinforcement spars.

3 TESTING METHODS

First the static elastic behaviour of the hinges is measured by deadweight loading on the wing leading edge at a fixed distance away from the hinge (Figure 8). The wing base was disengaged from the crank to free it from the friction of the gears and motor. Increasing weights induced a larger moment M to bend the hinge to larger angle β. This moment-angle relationship must be determined experimentally because the PI hinge yielded at large angle which was unaccounted in theory, though it maintained elasticity. This actual moment-angle relationship is used to determine the elastic energy stored:

$$W_{elastic} = \int_{\beta_1}^{\beta_2} \mathrm{M}(\beta) d\beta \qquad (5)$$

If the static load on the leading edge is removed abruptly, the elastic hinge recoils the wing towards the horizon. The stiffer hinges clap the wings faster when released from the same

deflection (40° wing angle which corresponds to a hinge bending angle of 130° bending angle). This recoil time of the hinges is observed using a high speed camera.

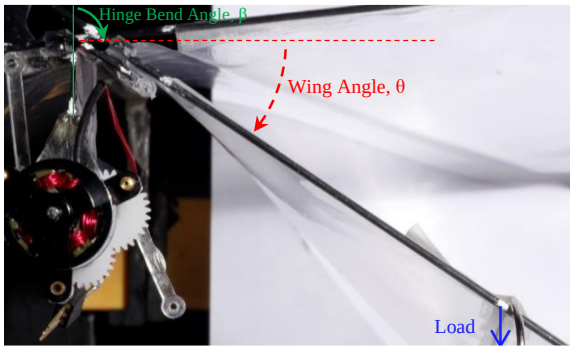


Figure 8 Static loading on the wing to determine the elastic hinge behaviour.

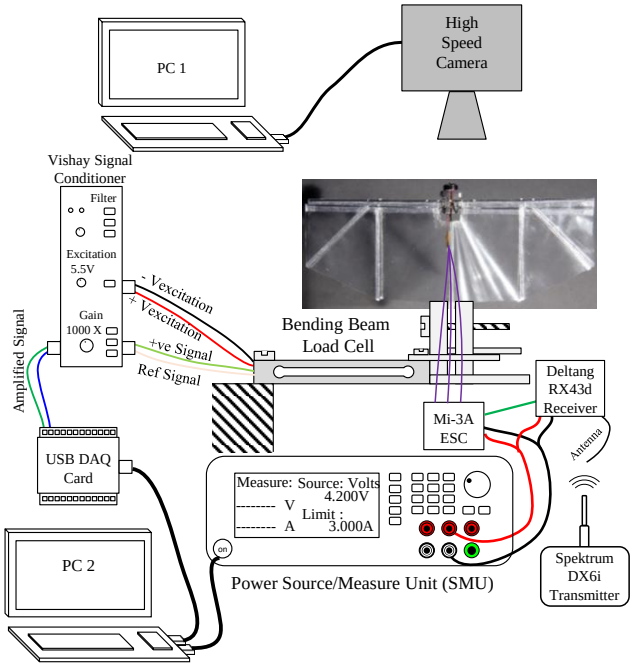


Figure 9 Tethered flight test setup for the measurement of thrust and input power to motor and high speed video of the wing kinematics.

Secondly, a tethered flight test was done by running the X-wing clapper on a load cell, as shown in Figure 9. The clapper was driven by a brushless motor which is controlled remotely with a transmitter through a receiver and electronic speed controller (ESC). A source/measure unit (Agilent B2902A SMU) provides constant-voltage power input to the ESC and receiver, and measure the supplied current simultaneously at 2000Hz sampling rate.

Simultaneously, a load cell (Zemic Q47-20*6-6, 120g capacity) was used to measure the thrust generated by the clapping wings. The output signals from the load cell were amplified 1000 times using a signal conditioner before it was logged using a data acquisition card (NI USB6009 DAQ card) at 2000Hz sampling rate.

In addition, the wing clapping motions were recorded in video of 512×512 pixels at a frame rate of 2000 frame per second using Photron FASTCAM 1024 high-speed camera. The wing leading edge motions were tracked to yield the instantaneous wing position and wing angular speed in the stroke plane, and the hinge bending angle. The vertical carbon fiber plate was also tracked to provide the correct angle reference line, because the clapper body rocked and rolled a little during wing oscillation (Figure 10).

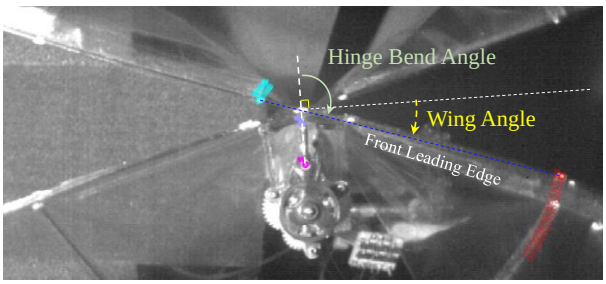


Figure 10 Tracking of the leading edge motion and vertical plate in the high speed snapshot for 2cm hinges X-wing.

4 RESULTS AND DISCUSSIONS

4.1 *Elastic Hinge Behaviour*

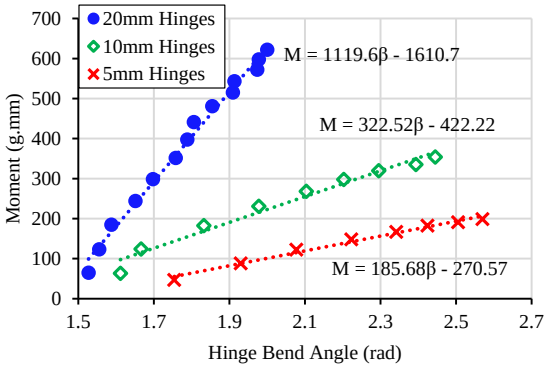


Figure 11 Bending moment required to bend the hinges.

Figure 11 shows the bending moment required to bend the hinge increases linearly with the bending angle. The 2cm hinge is almost 3.5 times stiffer than the 1cm hinge and 6 times more than the 0.5cm hinge. The elastic energy storage is proportional the stiffness. However, too much bending causes the yielding of the plastic hinge. The plastically deformed plastic hinge may not return fully upon removal of the deadweight. Yet, they exhibit consistent elastic characteristics.

The elastic energy stored is released and converted to the wing kinetic energy that assist wing clapping. In presence of air damping, the 20mm hinge carrying a wing plane (spanning over the left and right) recoiled to the 0° wing angle within 0.0765s. Similarly the 10mm hinge carrying the same wing recoil back to 0°, but slower at 0.093s. The 5mm hinge is the

http://www.imavs.org/

slowest to recoil back for 0.1200s (see Figure 12). In the wing reciprocation, this quasi-static spring back effect would manifest in a faster wing clap than the fling.

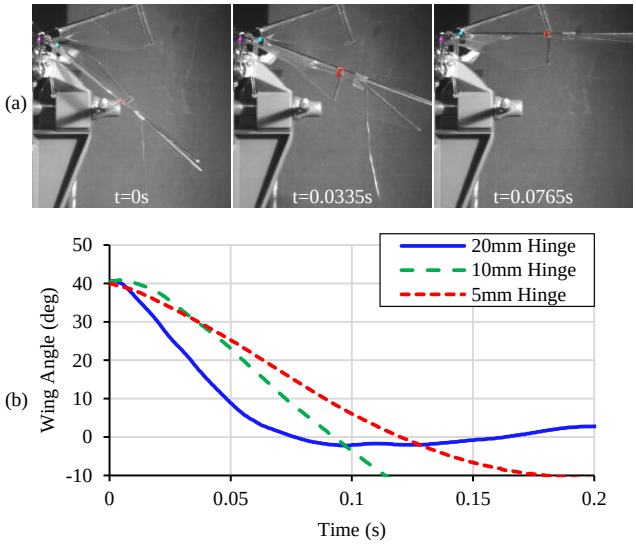


Figure 12 (a) High speed snapshot of the wing spring back for the 2cm hinge. (b) The time taken for the leading edge to return to horizontal.

4.2 *Wing Kinematics*

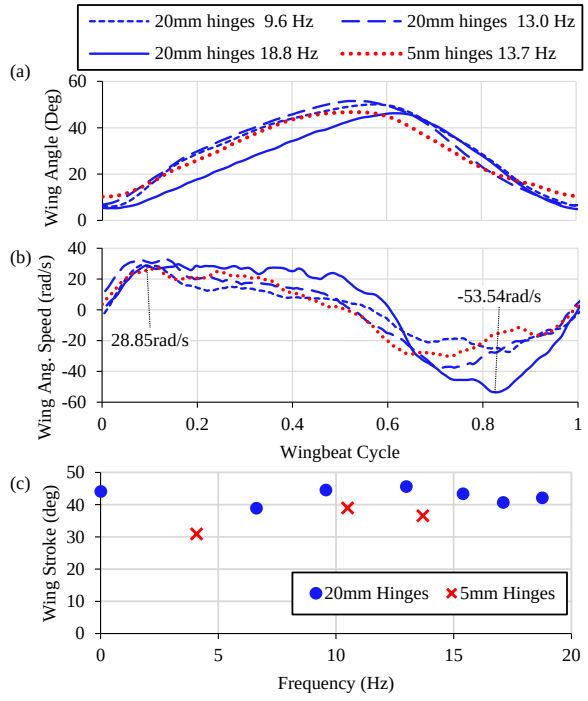


Figure 13 (a) Wing angular position and (b) Wing angular velocity in a wingbeat cycle at different wingbeat frequencies. (c) Wing stroke along wingbeat frequency for 2cm hinges and 0.5cm hinges.

During a quarter cycle of wingbeat, the elastic hinges assist the motor to clap wings faster. In a subsequent quarter cycle of wing fling, it elastically resists the wing motion. The work done by motor against the elastic hinges is stored as elastic energy, which will be released in the subsequent quarter wingbeat cycle. As a result the wing clapping is faster than the wing fling as shown in Figure 13. At first, the wings fling open with a short burst of high angular speed, but slowed down at larger wing angle. During the quarter cycle of wing fling, a part of the motor power is used to deform the elastic hinge, storing as elastic energy. With larger elastic storage, the motor clapped the wings at higher speed, and the clap duration is shorter than the fling.

As the wingbeat frequency increases, the clap peak angular speed increases significantly, but the maximum fling angular speed change slightly. Figure 14 shows that the wing kinetic energy in stroke plane increases with the wing beat frequency as shown in. The same wing on different hinges show different maximum wing kinetic energy given the same wingbeat frequency due to the impact of spring recoil on the wings' speed profile (See Figure 13b). However, the elastic energy storage remains constant because the wing stroke did not change much along frequency. At higher frequency, the wing kinetic energy increases beyond the elastic energy storage.

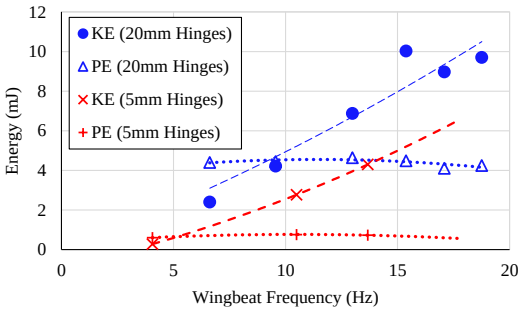


Figure 14 Total maximum kinetic energy of clap and fling stroke for a single wing pair and the elastic energy storage in a single hinge along wingbeat frequency.

4.3 *Thrust Generation and Energetic Cost*

Faster clap is useful to generate more powerful thrust, as shown in Figure 15(a). Given the same wingbeat frequency, the 20mm hinges help clap the wings faster, and thus increases the thrust generation. Stiffer hinges are good for faster clap with no extra power needed, as shown in Figure 15 (b). This showed that elastic energy storage primarily boosted the wing clapping speed for larger thrust generation given same wingbeat frequency. Hence, the optimal elastic hinges improved the efficiency of the thrust generation.

At a maximum thrust of 20.7g, the X-wing with 2cm hinges is able to produce sufficient thrust to lift off and climb. In a tethered flight with onboard ESC, receivers and battery, the X-wing could lift its weight of 13.4g and climb, as shown in Figure 17.

http://www.imavs.org/

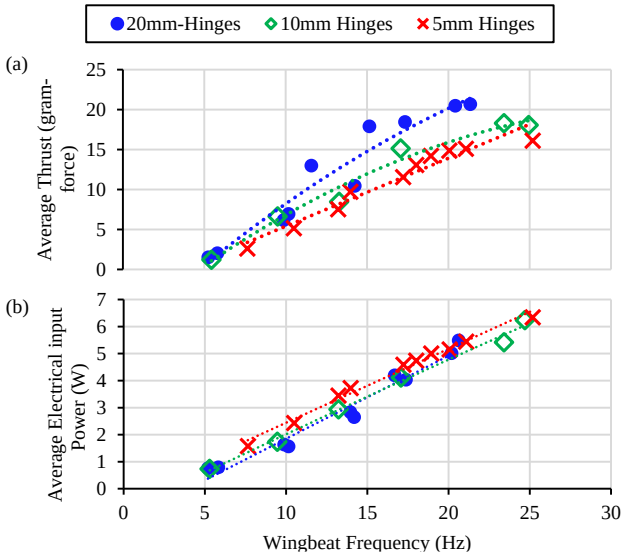


Figure 15 (a) Average thrust generation along wingbeat frequency. (b) Average electrical input power along frequency.

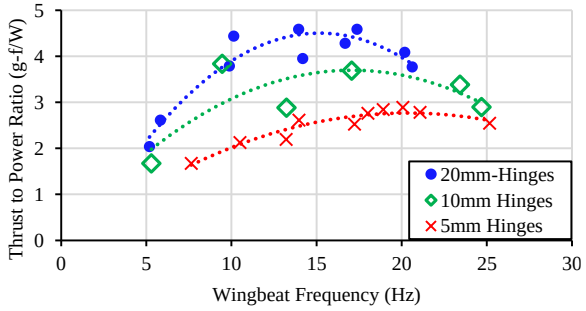


Figure 16 Ratio of the average thrust generation to the electrical power input to motor along frequency.

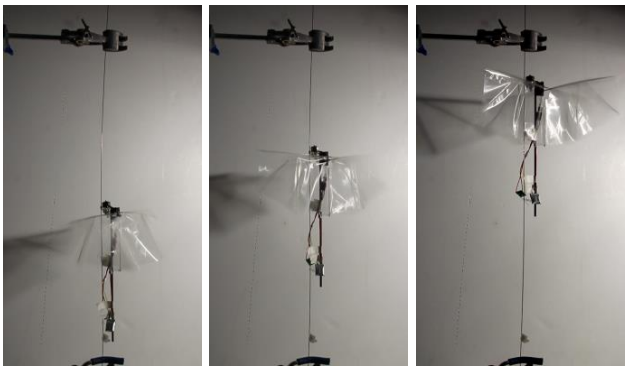


Figure 17 Photo of the 2cm hinges X-wing lifting its weight and climb in tethered flight demonstration.

5 CONCLUSION

We introduced an X-wing clapper mechanism integrated with elastic polyimide hinges to store sufficient elastic energy. The elastic polyimide hinges could store sufficient elastic energy for a large bending angle without breaking. This work shows that 4.2mJ elastic energy storage by using a polyimide film hinge is good at improving the energetic efficiency to 4.5 gram lift force per watt for the flapping 24cm-span X-wings. This present a 50% improvement of the maximum lift to power ratio as compared to that using a soft hinge of 0.8mJ. The X-13.4g wing clapper with matched elastic hinges is able to produce 20.7g thrust for lift off and climbing.

ACKNOWLEDGEMENTS

The authors wish to thank Temasek Laboratories @ National University of Singapore and Nanyang Technological University for their support of this project.

REFERENCES

[1] The Ornithopter Zone, "Full History of Ornithopters," [Online]. Available: http://www.ornithopter.org/history.full.shtml. [Accessed 2015].

[2] D. Lentink, S. Jongerius and N. Bradshaw, "The scalable design of flapping micro air vehicles inspired by insect flight," in *Flying insects and robots*, Springer-Verlag Berlin Heidelberg, 2009, pp. 185-205.

[3] K. M. E. De Clercq, "Flow Visualization and Force Measurements on a Hovering Flapping-wing MAV 'DelFly II'," Delft University of Technology, Delft, The Netherlands, 2009.

[4] R. Madangopal, Z. A. Khan and S. K. Agrawal, "Biologically Inspired Design Of Small Flapping Wing Air Vehicles Using Four-bar Mechanisms and Quasi-steady Aerodynamics," *Journal of Mechanical Design*, vol. 127, no. 4, pp. 809-816, 2005.

[5] S. S. Baek, K. Y. Ma and R. S. Fearing, "Efficient Resonant Drive of Flapping-Wing Robots," in *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, USA, 2009.

[6] M. Keennon, K. Klingebiel, H. Won and A. Andriukov, "Development of the Nano Hummingbird: A Tailless Flapping Wing Micro Air Vehicle," in *60th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, Nashville, Tennessee, 9-12 January 2012.

[7] R. Wood, "Liftoff of a 60mg flapping-wing MAV," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, 2007.

[8] C. Richter and H. Lipson, "Untethered Hovering Flapping Flight of a 3D-Printed Mechanical Insect," in *12th International Conference on Artifical Lift (Alife XII)*, Odense, Denmark, 2010.

[9] F. v. Breugel, Z. E. Teoh and H. Lipson, "A Passively Stable Hovering Flapping Micro-Air Vehicle," in *Flying Insects and Robots*, Springer-Verlag Berlin Heidelberg, 2007, pp. 171-184.

[10] P. Zdunich, D. Bilyk, M. MacMaster, D. Loewen, J. DeLaurier, R. Kornbluh, T. Low, S. Stanford and D. Holeman, "Development and Testing of the Mentor Flapping-Wing Micro Air Vehicle," *Journal of Aircraft*, vol. 44, no. 5, pp. 1701-1711, 2007.

[11] R. Wood, "Design, fabrication, and analysis of a 3DOF, 3cm flapping-wing MAV," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, 2007.

[12] R. Wood, R. Nagpal and G.-Y. Wei, Flight of the Robobees, Scientific American, March 2013.

[13] R. Sahai, K. C. Galloway, M. Karpelson and R. J. Wood, "A flapping-wing micro air vehicle with interchangeable parts for system integration studies," *2012 IEEE/RSJ International Conference onIntelligent Robots and Systems (IROS)*, pp. 501--506, 2012.

http://www.imavs.org/

Systematic Modeling of Rotor Dynamics for Small Unmanned Aerial Vehicles

Limin Wu*, Yijie Ke and Ben M. Chen
Unmanned Systems Research Group
Department of Electrical and Computer Engineering, Faculty of Engineering
National University of Singapore, 21 Lower Kent Ridge Rd, Singapore 119077

http://www.imavs.org/

ABSTRACT

This paper proposes a systematic modeling approach of rotor dynamics for small unmanned aerial vehicles (UAVs) based on system identification and first principle based methods. Both static state response analysis and frequency-domain identifications are conducted for rotor, and CIFER software is mainly utilized for frequency-domain analysis. Moreover, a novel semi-empirical model integrating rotor and electrical speed controller is presented and verified. The demonstrated results and model are promising in UAV dynamics and control applications.

1 INTRODUCTION

Rotor dynamics plays a crucial role in understanding flight dynamics for small UAVs with rotor configuration, and the challenges include complex propeller aerodynamics and system hardware response. In current UAV development, the rotor dynamics is usually simplified by utilizing low frequency response information only, such as static thrust and torque. To further improve flight maneuverability, dynamics of the rotor system has to be investigated.

Typically, modeling methods for such system include system identification and first-principle modeling method [1]. Both methods are attempted and realized in our work. On the one hand, system identification can extract the information of system response around a certain trim condition, such as static state and frequency-domain response. The accuracy of this method depends heavily on experiment setup and data processing techniques. In our work, the CIFER software is mainly utilized for frequency response analysis, which can provide a reliable estimation on dynamics response and has been commonly used in flight dynamics identification [3, 4]. On the other hand, first-principle modeling relies on a deep understanding of underlying physics. As the rotor subsystem in UAVs usually consists of motor, propeller and electronic speed controller (ESC), modeling of such integrated subsystem is a challenge, which is also commonly overlooked in the literature to our best knowledge. Thus we propose a semi-empirical method based on experiment data, in order to approximate the subsystem dynamics for flight dynamics and control applications.

The paper is organized as follows: Section 2 presents the experiment setup. Section 3 illustrates the system identification method for rotor response, including both static state analysis and frequency domain analysis. Responses of thrust force, torque and propeller angular speed are recorded and investigated. Section 4 presents a novel semi-empirical method integrating brushless DC motor, propeller and ESC. Conclusions are summarized in section 5.

2 EXPERIMENTS AND CONDITION

The experiment aims to capture dynamics of rotor system under desired condition. Setup of our work is shown in schematics in Figure 1 and 2.

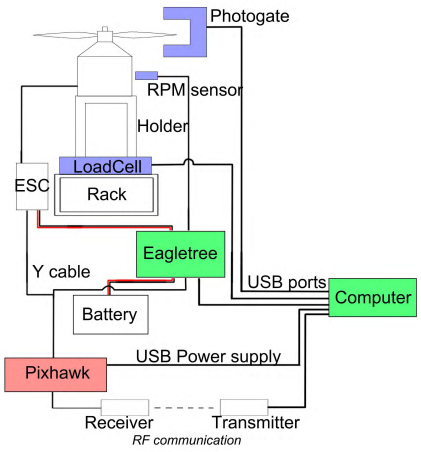


Figure 1: Schematic of rotor test hardware setup

Equipments are listed below in Table 1. The input of rotor system is normalized throttle from 0.0 to 1.0. It is generated by Pixhawk at fixed data rate of 150Hz. Outputs of system are thrust and torque as well as angular speed of rotor. Loadcell can record force and torque in six degree of freedom with fixed logging rate at 1000Hz. Angular speed in RPM can be measured by Eagletree and Photogate devices. Eagletree has a low recording rate of 10Hz and Photogate can record at extremely high frequency which is able to capture every single

*Corresponding author.
Email: a0112849@u.nus.edu

http://www.imavs.org/

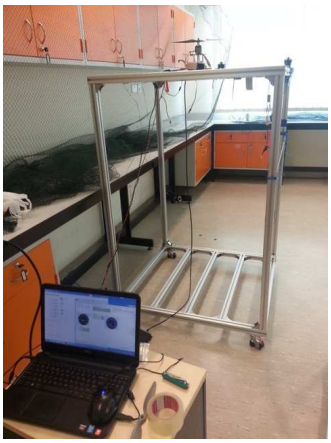


Figure 2: Rotor test setup

Name	Description
APC 11X5.5P Propeller	rotor propeller
Scorpion 3026 motor	brushless DC motor
Scorpion Commander 90A	ESC
4-cells LiPo battery	power supply
optical RPM sensor	angular speed measuring device
Photogate	angular speed measuring device
ATI load cell	6 DOF force and torque logging
Pixhawk	PWM throttle input generator
EagleTree elogger V4	battery voltage and current recording
rack and holder	supporting platform
computer	sensor data collection

Table 1: LIST OF EQUIPMENTS

revolution of rotor. Power is supplied by LiPo battery instead of DC power supply because battery has limited discharge rate. And we want to study the dynamics of rotor with the same discharge rate in power source as our actual UAVs.

3 SYSTEM IDENTIFICATION METHOD

Both static state analysis and frequency-domain identification methods are utilized for rotor dynamics based on our experiment setup. The static state response can help identify step-response features for interested states, while frequency response can extract richer dynamics information around a trim condition. In this section, main results of these two methods will be presented, and we are mainly concerned with the propeller thrust, torque and angular speed.

3.1 Static state analysis

3.1.1 Stimulus signal

Stimulus signal for static state analysis is shown in lower part of Figure 3, which includes 10 step functions with amplitude increased from 0.1 to 1.0 with step size 0.1. Duration of each step function is 5 seconds therefore total duration is 50 seconds.

3.1.2 Angular speed channel

Result from step signal test indicates the steady-state angular speed when throttle varies from 0.1 to 1.0 with step size 0.1. Table 2 and Figure 3 show the result. As can be observed in

Throttle	Steady-state Angular speed (rad/s)
0.10	228.18
0.20	423.17
0.30	566.85
0.40	689.58
0.50	737.54
0.60	788.02
0.70	832.84
0.80	892.74
0.90	937.35
1.00	941.54

Table 2: RELATION BETWEEN THROTTLE AND STEADY-STATE ANGULAR SPEED

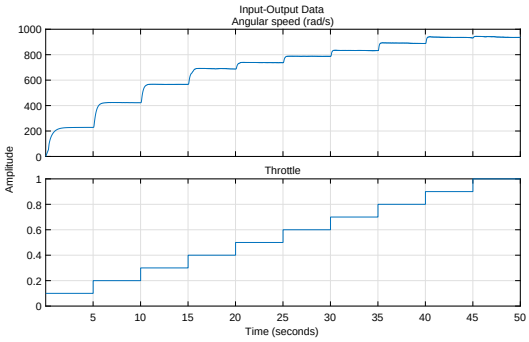


Figure 3: Plot of measured angular speed with corresponding throttle in step signal test

figure and table, angular speed changes proportional to throttle. The relation can be summarized as function:

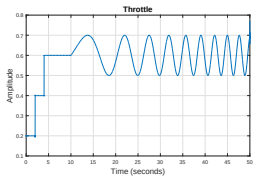
$$\omega = -1080u^2 + 1952u + 42 (0 \leqslant u \leqslant 1) \qquad (1)$$

where ω is angular speed in rad/s and u is normalized throttle.

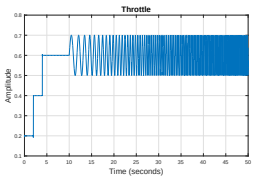
3.2 Frequency domain analysis

3.2.1 Stimulus signal

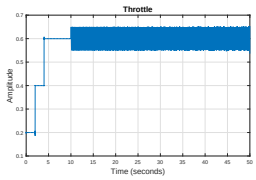
Stimulus signal for frequency domain analysis is selected to be a sinusoidal wave swept from 0.05Hz to 20Hz at throttle equals to 0.6 equilibrium point. The frequency range is selected based on our flight control design as higher frequency exceeds greatly the control bandwidth. However, in order to eliminate effect from battery voltage drop, this chirp signal test has been divided into three experiments:



(a) Experiment 1



(b) Experiment 2



(c) Experiment 3

Figure 4: Chirp stimulus signal

- Experiment 1: Chirp signal swept from 0.05Hz to 0.5Hz, test duration 50s includes 10s warm-up.
- Experiment 2: Chirp signal swept from 0.5Hz to 5Hz, test duration 50s includes 10s warm-up.
- Experiment 3: Chirp signal swept from 5Hz to 20Hz, test duration 50s includes 10s warm-up.

All three chirp signals have been plotted in Figure 4.

3.2.2 Thrust channel

Following figures show the relationship between throttle and thrust in both time domain (Figure 5, Figure 6 and Figure 7) and frequency domain (Figure 8), and this set of data is collected during the chirp signal rotor test. The first 10 seconds warm-up section and mean value will be removed by CIFER during data preprocessing as the response near equilibrium is our interest.

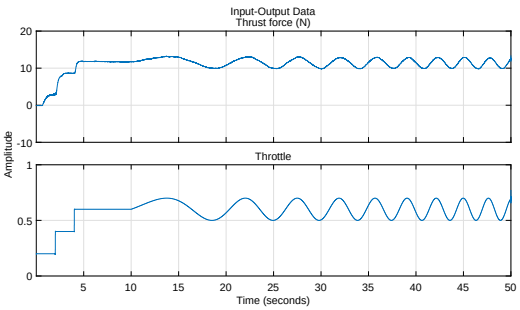


Figure 5: Thrust channel, time domain data of experiment 1

The system in thrust channel has a decreasing gain with frequency. The plot of coherence in Figure 8 indicates the accuracy of frequency estimation. For rotorcraft, recommended

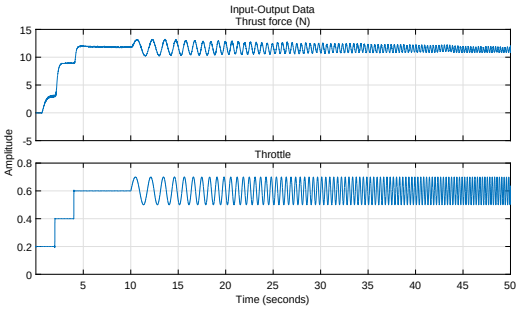


Figure 6: Thrust channel, time domain data of experiment 2

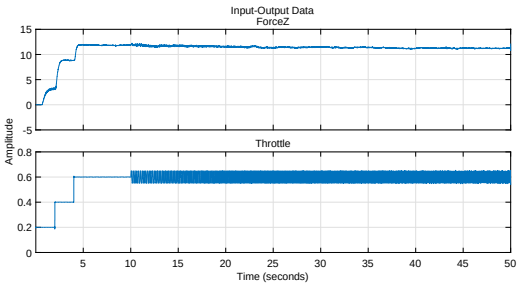


Figure 7: Thrust channel, time domain data of experiment 3

threshold coherence value is 0.6 according to [4]. Therefore, Figure 8 tells that the estimation is credible within the range of frequency up to 30 rad/s. Phase plot suggests that system have a -135 degree phase at 30 rad/s. Hence the system can be approximated by a second order system within confidence range. Then parameter identification is applied and identification result is:

$$\frac{8859}{(s+9.35)(s+61.52)} \tag{2}$$

which DC gain = 15.40, Bandwidth = 9.12 rad/s.

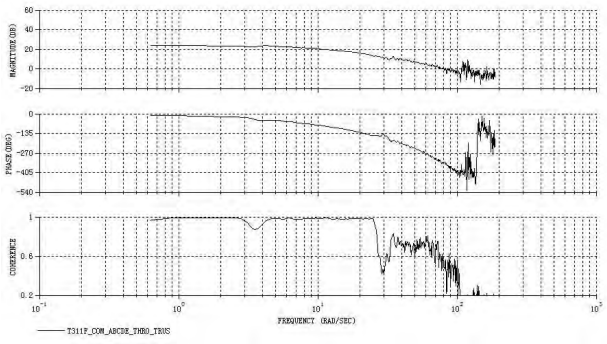


Figure 8: Thrust channel, Bode plot estimated by CIFER

http://www.imavs.org/

http://www.imavs.org/

3.2.3 Torque channel and angular speed channel

Similar to thrust channel, the analysis is conducted for torque channel and angular speed channel. Frequency response of torque channel is shown in Figure 9 and estimated transfer function is:

$$23.96\frac{(s+6.02)}{(s+16.94)(s+33.97)} \tag{3}$$

which DC gain = 0.25, Bandwidth = 129.52 rad/s.

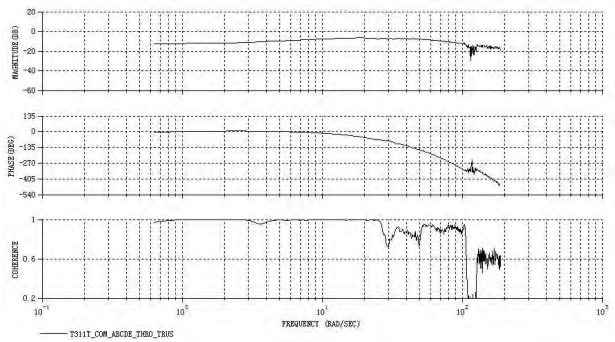


Figure 9: Torque channel, Bode plot estimated by CIFER

Frequency response of angular speed channel is shown in Figure 10 and estimated transfer function is:

$$\frac{225961}{(s+9.39)(s+45.34)} \tag{4}$$

which DC gain = 530, Bandwidth = 9.01 rad/s.

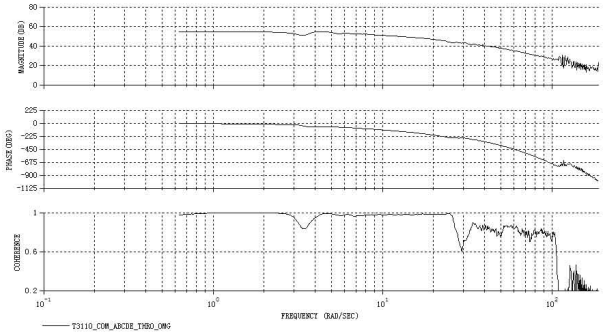


Figure 10: Angular speed channel, Bode plot estimated by CIFER

4 SEMI-EMPIRICAL MODEL OF ROTOR SUBSYSTEM

In previous section, the rotor system is considered as a black-box. The result of identification cannot describe what is happening inside the black-box. In this section, the model of rotor system will be divided into three parts named ESC model, BLDC model and propeller model. Figure 11 shows the block diagram of the the semi-empirical model of rotor subsystem.

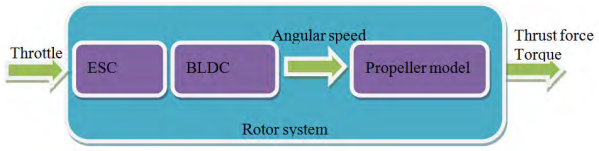


Figure 11: Components of rotor system model

4.1 *Propeller model*

Propeller model describes the response from angular speed to thrust and torque. According to blade element method, thrust and torque are proportional to the square of angular speed. Therefore, propeller model can be simply constructed as follow:

$$\begin{aligned} F &= K_f\omega^2 + F_{offset} \\ T &= K_t\omega^2 + T_{offset} \end{aligned} \tag{5}$$

where F is thrust force, T is torque, and ω is angular speed in rad/s. K_f ,K_t ,F_{offset} and T_{offset} are unknown coefficients to be identified. This model has been estimated via polynomial fitting based on experimental data obtained, the result is:

$$\begin{aligned} K_f &= 1.814657 \times 10^{-5} \\ F_{offset} &= -4.376713 \times 10^{-1} \\ K_t &= 2.798821 \times 10^{-7} \\ T_{offset} &= -5.700314 \times 10^{-3} \end{aligned} \tag{6}$$

4.2 *ESC model and BLDC model*

Previous group research suggested a ESC model [5]:

$$\begin{aligned} \omega_d &= throttle^2 K_a + throttle K_b + K_c \\ e_\omega &= \omega_d - \omega \\ u_{eq} &= K_p e_\omega + K_I \int e_\omega dt \end{aligned} \tag{7}$$

where ω_d is desired angular speed and ω is actual angular speed. *throttle* is normalized throttle between 0 and 1. K_a, K_b and K_c are polynomial coefficients describing relationship between throttle and ω. These three parameters have been estimated in section 3 static state analysis. u_{eq} is the equivalent voltage directly input to BLDC, and K_p, K_i are PI gains of control law. The BLDC model is [5]:

$$J_r\dot{\omega} = K_m i - K_r\omega^2 - K_s \tag{8}$$

$$L\frac{di}{dt} = u - Ri - K_e\omega \tag{9}$$

For low inductance BLDC, the above form is further simplified as:

$$J_r\dot{\omega} = -K_s - \frac{K_m K_e}{R}\omega - K_r\omega^2 + \frac{K_m u}{R} \tag{10}$$

where u is the input voltage, J_r is rotor inertia, K_m, K_s and K_e are motor electrical parameters, and *R* is the resistance. The output of ESC u_{eq} is equivalent to u, input of BLDC model.

As only throttle input and angular speed ω are measurable, a nonlinear model of above subsystem is derived as below with an augmented state x_a:

$$\begin{aligned} \dot{\omega} = & \quad -\tfrac{K_s}{J_r} - \tfrac{K_m K_e}{R J_r}\omega - \tfrac{K_r}{J_r}\omega^2 + \tfrac{K_m K_i}{R J_r}x_a + \tfrac{K_p K_m}{R J_r}\omega_d \\ \dot{x_a} = & \quad -\omega + \omega_d \end{aligned} \tag{11}$$

where $K_s, K_m, K_e, K_p, K_i, R$ and J_r are parameters under identification. All parameters are positive.

4.3 Model identification method

Parameter identification is achieved by Matlab function "nlgreyest", which estimates the unknown nonlinear grey-box model parameters based on measured data. The function employs minimization schemes with embedded line searching methods for parameter estimation. Initial value and range of parameters are estimated according to first principle.

4.4 Estimation of parameter range

- Effective motor resistance
  R is effective Motor Resistance. It can be found in data sheet of motor.

- Rotor inertia
  J_r is the moment of inertia of propeller and rotor. This parameter can be estimated by considering them as a thin rod rotating about its middle point. Corresponding formula to calculate J_r is:

$$J_r = \frac{mr^2}{12} \tag{12}$$

  where m is mass of rod and r is radius of the thin rod.

- Equivalent drag coefficient
  $K_r\omega^2$ represents reaction torque from propeller. Therefore, K_r is equivalent to drag coefficient K_f obtained in section 4.1.

- Motor torque constant
  Motor Torque constant K_m can be calculated by applying following formula:

$$K_m = \frac{60}{2\pi K_v} \tag{13}$$

  where K_v is motor velocity constant given in motor data sheet.

- Back EMF constant
  Back EMF constant K_e. In the three phase BLDC motors the relationship is approximately equal to [6]:

$$K_e = \sqrt{\frac{3}{2}} K_m \tag{14}$$

- Static friction torque constant
  K_s is static friction torque constant. It can be measured by torque meter. It usually has much less effect to torque when compared with back EMF and reaction torque of propeller. Hence, it is assumed that this term has 2 order of magnitude smaller than $K_r\omega^2$ term and $\frac{K_m K_e}{R}\omega$ term in equation (10).

- Proportional Gain
  K_p is proportional gain of control law. It can be estimated by using instantaneous angular acceleration while rotor is changing from one trim condition to another. Formula used to estimated K_p is:

$$J_r\dot{\omega} = \frac{K_m}{R}K_p e_\omega \tag{15}$$

  where $\dot{\omega}$ is angular acceleration and e_ω is difference angular speed between two trim conditions.

- Integral Gain
  K_i represents integral gain of control law. It can be estimated by observing the transition from one trim condition to another. equation (7) and equation (10) can be applied to both trim conditions, thus following equations can be obtained:

$$0 = -\frac{K_s}{J_r} - \frac{K_m K_e}{R J_r}\bar{\omega_1} - \frac{K_r}{J_r}\bar{\omega_1}^2 + \frac{K_m K_i}{R J_r}\int e_{\omega 1} dt \tag{16}$$

$$0 = -\frac{K_s}{J_r} - \frac{K_m K_e}{R J_r}\bar{\omega_2} - \frac{K_r}{J_r}\bar{\omega_2}^2 + \frac{K_m K_i}{R J_r}\int e_{\omega 2} dt \tag{17}$$

  $\bar{\omega_1}$ and $\bar{\omega_2}$ are steady-state angular speed for two trim conditions respectively, and both values are measurable. The difference between $\int(e_{\omega 1}dt)$ and $\int(e_{\omega 2}dt)$ can be obtained by integrating e_ω over transition interval between two trim conditions. Therefore, K_i can be estimated by minusing equation (17) with equation (16).

Based on estimated initial values, parameter ranges are defined as shown in Table 3. For parameters calculated from given value in datasheet, range is defined as $\pm 10\%$ of initial value. For parameters calculated from measured data in experiment, range is defined as $\pm 50\%$ of initial value.

4.5 Result and comparison

With aforementioned parameter range and propeller model, Matlab function "nlgreyest" is used to identify the parameters, and the results are shown in Table 4.

Figure 12 and Figure 13 show a comparison between ω from identified semi-empirical model and measured value. In order to verify model fidelity, NRMSE fitness value is used in Matlab, as defined by

$$fit\% = 100\%(1 - \frac{\|p - \bar{p}\|}{\|p - mean(p)\|}) \tag{18}$$

http://www.imavs.org/

http://www.imavs.org/

Parameter	Initial value	Range
R	0.014	$\pm 10\%$
J_r	3×10^{-5}	$\pm 50\%$
K_r	2.8×10^{-7}	$\pm 10\%$
K_m	0.01	$\pm 10\%$
K_e	0.012	$\pm 10\%$
K_s	10^{-2}	$\pm 50\%$
K_p	2.6×10^{-4}	$\pm 50\%$
K_i	7×10^{-2}	$\pm 50\%$

Table 3: RANGE OF PARAMETERS

Parameter	Value
R	0.0154
J_r	4.5×10^{-5}
K_r	3.08×10^{-7}
K_m	0.009
K_e	0.0132
K_s	1.5×10^{-2}
K_p	1.3×10^{-4}
K_i	0.069

Table 4: IDENTIFIED PARAMETERS

where p is the validation data collected during experiment and $\bar{p}$ is the output generated by BLDC model. The 75.72% fitness shows that our model prediction fits well with real experiment data.

Another model validation standard is Theil inequality coecient[7] (TIC), which is defined by

$$TIC = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{p}-p)^2}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{p})^2} + \sqrt{\frac{1}{n}\sum_{i=1}^{n}(p)^2}} \tag{19}$$

where n is the total sample amount, p is the validation data collected during experiment and $\bar{p}$ is the output generated by BLDC model. TIC is a normalized value between [0, 1], and zero indicates a perfect matching. In practice, the threshold of TIC is commonly set at 0.25 [8]. As for TIC-based validation, results for Experiment 1 and 2 are 0.020 and 0.025 respectively. Both TIC value are much lower than the threshold value 0.25. Therefore, results indicate that identified BLDC model has sufficient accuracy.

After linearized at throttle = 0.6 trim condition, BLDC model is combined with ESC and propeller models, then compared with transfer function obtained through identification approach in section 3. Input is throttle and output is thrust force. Bode plots are shown in Figure 14.

Bode plot of system identified from semi-empirical model has a DC gain equals to 21.8dB and bandwidth 4.95 rad/s. For transfer function identified by CIFER, DC gain is 23.7dB and bandwidth is 9.12 rad/s. The comparison (Figure 14)

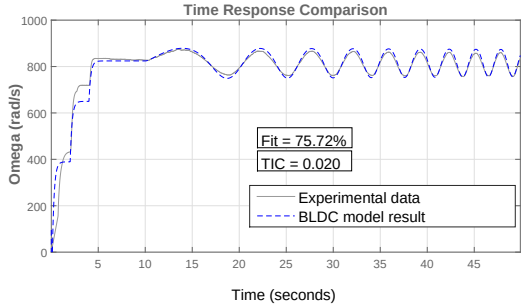


Figure 12: BLDC model result compare with measured angular speed (Experiment 1)

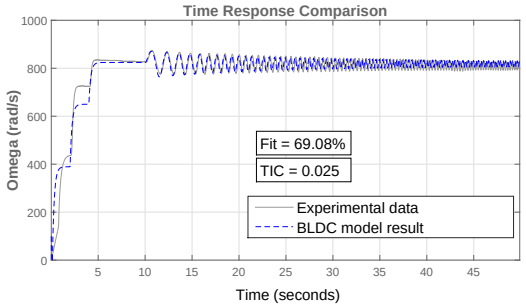


Figure 13: BLDC model result compare with measured angular speed (Experiment 2)

shows that our semi-empirical model can fit well with CIFER identification up to 10 rad/s, and the accuracy implies that our model is capable of predicting rotor dynamics and can be integrated in real-time UAV dynamics and control applications.

5 CONCLUSIONS

To conclude, this paper presents a systematic modeling approach for rotor dynamics integrating brushless DC motor, propeller and ESC. First, static state analysis is done to indicate the behavior under various steady states. Then, frequency responses of dynamics in various channels are successfully generated. Transfer function models are estimated and proven to be reliable for up to 20 rad/s. Finally, a novel semi-empirical model is presented and validated by experimental data. The model is proven to fit well with frequency response results up to 10rad/s and is promising in real-time implementation for UAV dynamics and control.

REFERENCES

[1] B. M. Chen, T. H. Lee, K. Peng and V. Venkataramanan, *Hard Disk Drive Servo Systems*, 2nd ed. New York, NY: Springer, 2006.

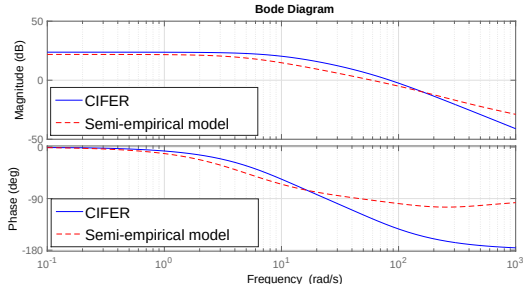


Figure 14: Bode plots of systems estimated by CIFER and by Semi-empirical model. Thrust channel

[2] Ljung L, "Linear Model Identification," in *System identification toolbox*, South Natick, MA:The MathWorks Inc., 1988.

[3] *Comprehensive Identification from Frequency Responses Users Guide*, UCSC, Santa cruz, CA, 2012.

[4] G. Cai, B. M. Chen, T. H. Lee. *Unmanned rotorcraft systems*, New York, NY:Springer Science & Business Media, 2011, pp. 90-94 .

[5] Yijie Ke, "An empirical model to rotor dynamics with brushless DC motor," Internal report in NUS unmanned system research group.

[6] *General motor terminology*, The Motor & Motion Association, S. Dartmouth, MA, 2015.

[7] M. B. Tischler, R. K. Remple , *Aircraft and rotorcraft system identification: Engineering methods with flight test examples*, VA:American Institute of Aeronautics and Astronautics, 2006.

[8] G. Cai, T. Taha, J.Dias amd L. Seneviratne, "A framework of frequency-domain flight dynamics modeling for multi-rotor aerial vehicles,". *Journal of Aerospace Engineering*, May. 2016.

http://www.imavs.org/

Tailless Control of a Double Clap-and-Fling Flapping Wing MAV

CHAN Woei Leong,* Nguyen Quoc Viet, and Marco DEBIASI
National University of Singapore

http://www.imavs.org/

ABSTRACT

This paper presents the study on a tailless control mechanism designed to tilt the wing roots of a double clap-and-fling flapping wing micro-air-vehicle (MAV). The MAV has made stable hovering flight with tail and stabilizers, the work presented in this paper paves the way for the implementation of tailless flight control on the MAV. A wing tilting mechanism driven by three linear servos was designed and fabricated. Study shows that tilting the wing roots creates a linear trend of pitching and rolling moments if the centre of gravity (CG) falls on the correct location. Wing kinematics was captured and it was found that the higher drag due to higher angle of attack could be a contributing factor to the pitching and rolling moments. Another contributing factor could be the lift dissymmetry of the opposite pairs of wings. The effects of pitch-roll coupling are also presented.

1 INTRODUCTION

Tailless flapping wing is one of the most challenging aspects of flapping wing flight, mainly due to the complexity of the mechanism, the coupling effect of forces and moments that is yet to be fully understood, and the stability problem that arises once the tail is removed. Furthermore, there are quite a few types of flapping wings, including two-winged platforms that generate thrust or lift mainly by delayed stall and wake capture, four-winged platforms that utilize clap-and-fling, and tandem wing platforms that exploit the interactions of the forewings and hind wings. Implementing tailless control on each type of platform could be very different due to their differences in aerodynamics characteristics. However, despite the challenges, tailless control comes with the advantages of biomimicry, less sensitive to disturbance due to the absence of tail surfaces, and higher manoeuvrability.

The first step to achieve tailless control is to design a symmetrical gearbox. Such gearboxes ensure hovering equilibrium can be achieved. The Temasek Laboratories of the National University of Singapore has been developing flapping wing MAVs, including the Odopter [1] that encompasses a hybrid concept, and the FolowerFly [2] that is designed to hover and utilizes a symmetrical flapping mechanism. Our previous efforts on tailless control were focused on bench testing of two-winged platforms [3, 4]. Although the outcome was satisfactory, two-winged platforms tend to suffer from severe vibration and the lift generation is not as efficient as a four-winged clap-and-fling platform. Hence the decision to implement tailless control on the FlowerFly was made.

Similar effort has been done by different research groups, including Karasek et al. [5, 6], Takagi et al. [7], and Phan et al. [8]. The mechanisms used are mainly Type 1: changing the stroke angle and flapping rate to create lift and drag dissymmetry, which is harder to implement, or Type 2: manipulating the trailing edge of the wing roots, similar to the mechanism discussed in this paper. All of these efforts focus on two-winged platforms.

Some successful tailless flapping wing platforms include the Nano Hummingbird [9], FESTO BionicOpter [10], FESTO eMotionButterflies [11], and the Robobee [12, 13]. Among these platforms, the Nano Hummingbird which utilizes the aforementioned Type 2 mechanism, stands out as the most impressive standalone platform. Although the Robobee which utilizes Type 1 mechanism is a tethered platform, it is the most outstanding in terms of fabrication techniques and miniturization.

This paper focuses on the implementation of a Type 2 tailless control mechanism and its implementation on a four-winged clap-and-fling flapping wing platform. Section II of this paper elaborates the flapping mechanism, the wing, and the wing tilt mechanism. The experiment setup is being explained in Section III. The experiment results are presented and discussed in Section IV. Finally, the concluding remarks end the paper.

2 FLAPPING MECHANISM, WING, AND WING TILT MECHANISM

2.1 Flapping Mechanism

The flapping mechanism of the FlowerFly MAV [2] is used in this study. It is a four-winged flapping mechanism which utilizes double clap-and-fling. The wings clap at the front and rear at the end of down stroke and clap at the right and left at the end of upstroke. This unique design fully utilizes the whole volume around the MAVs body, maximizing the wing swept volume for lift generation. It is designed to be symmetrical such that the net lift acted at the centre of the flapping mechanism, allowing the MAV to hover at equilibrium with tail attached.

*Email address(es): tslcwl@nus.edu.sg

2.2 Wing

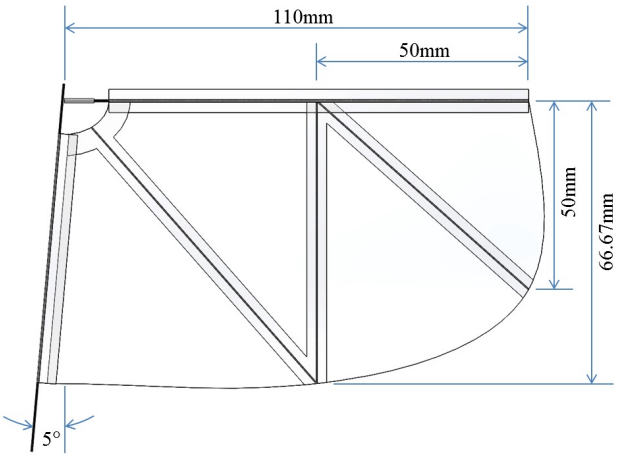


Figure 1: Wing dimension.

The wing has a root-to-tip wingspan of 110mm and the maximum chord length of 66.67mm. At the wing root, there is a 5° of slack angle to encourage wing rotation.

The wing membrane is 10μm Mylar film. The leading edge, wing root, and stiffeners are carbon rods at diameter of 0.6mm, 0.5mm, and 0.25mm respectively. The carbon rods are attached on the wing membrane with 23.4μm transparent polyester stickers. The total weight of one wing is 0.29g.

2.3 Wing Tilt Mechanism

The wing tilt mechanism is driven by three linear servos. Two servos are attached on the body right under the flapping mechanism, and one attached at the bottom of the body as shown in Figure 2. Servo 1 and 2 are used to push and pull the pitching arms. The pitching arms are attached on the rolling arm, which is driven by Servo 3. The wing roots are attached to the pitching arms such that the wings can be tilted longitudinally (forward and rearward) using the pitching arms, and laterally (rightward and leftward) using the rolling arm.

The arms and the linkages are mainly CNC milled out of Delrin plastic, but the arms of Servo 1 and 2 are connected to the mechanism through ball links to minimize the coupling between pitch and roll. Due to limitation on space, the mechanism can tilt the wing roots within limited range of $\pm 2°$ for both pitch and roll inputs.

2.4 Coordinate System and Sign Convention

The average tilting angle of the wing roots at longitudinal and lateral directions are given the designation of δ_p and δ_r respectively, which stand for pitch input and roll input. At neutral position, δ_p and δ_r are both zeros. In this study, tilting the wing root rearward and rightward are positive.

The coordinate system used is consistent with body-fixed coordinate system of aircraft flight dynamics: the origin at the centre of gravity, positive X, Y and Z directions coincide with

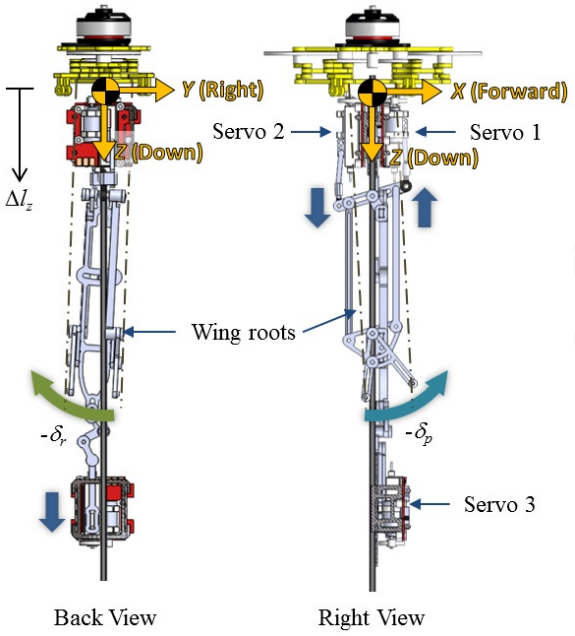


Figure 2: Wing tilt mechanism and coordinate system.

forward, rightward, and downward respectively as shown in Figure 2. The centre of gravity is assumed to be at the centre of the stroke plane. The force and moment data presented in this paper were transformed to this coordinate system, except for Section 4.3 where the assumed CG is shifted at Δl_z to evaluate the effect of CG shift.

3 EXPERIMENTAL SETUP

3.1 Force and Moment Measurement

Unlike our previous studies on two-winged platforms [3, 4], the four-winged platform produces less moment fluctuations, allowing a smaller load cell to be used for force and moment measurement at higher resolution. The flapping wing platform is attached on an ATI Nano17 Titanium as shown in Figure 3. The sampling rate used was 5000Hz.

Shunt resistors were used for current measurement. The power was supplied to the system through two 31 AWG enamelled wires as shown in Figure 4. The wires are at the right thickness to make sure the force and moment measurements, and the power supplied to the system are not compromised. The flapping frequency and wing tilt angles were controlled using a remote control radio. The flapping frequency was measured using a magnet attached on the main gear and a Hall Effect switch. The same signal from the Hall Effect switch was used as trigger signal for load cell and high speed camera.

3.2 Wing Kinematics Capture

The wing kinematics of the front left wing was captured using similar technique in our previous studies [4].

http://www.imavs.org/

http://www.imavs.org/

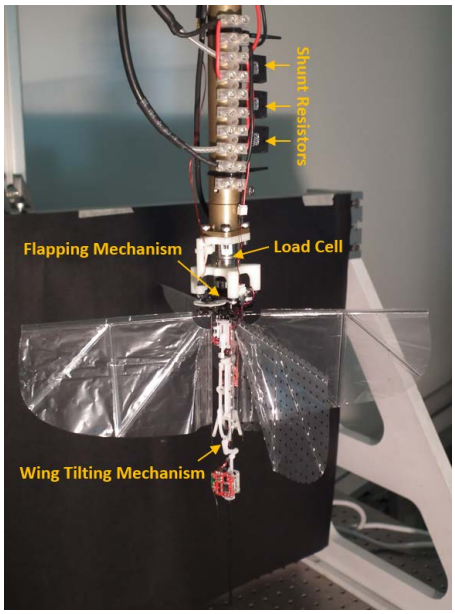


Figure 3: Experimental setup for force and moment measurement.

High speed videos of the flapping wings with markers were recorded using the Phantom Miro 320S high speed camera at 5000Hz from three different viewing angles. Using one camera, the three viewing angles were achieved by fixing the camera at one position, and rotate the flapping wing platform. The trigger signal ensures the video recording always starts at the same wing position. The wing kinematics repeatability was satisfactory. Figure 5 shows the snapshots of high speed video from three viewing angles. Theoretically, two viewing angles are enough to reconstruct the wing kinematics, but three viewing angles reduces the blind spots significantly. The wing kinematics was reconstructed using the open source Matlab code from Hedrick [14].

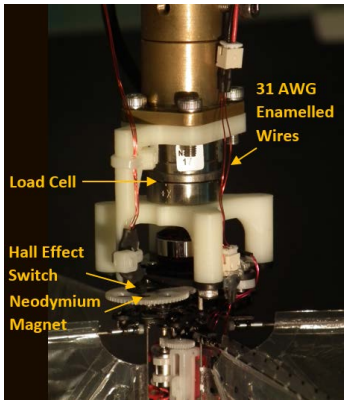


Figure 4: Close-up view of the experimental setup.

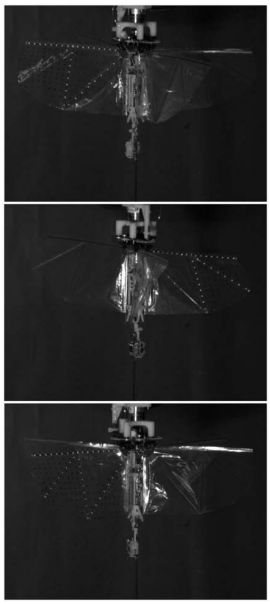


Figure 5: Snapshots of high speed video from three viewing angles.

4 RESULTS AND DISCUSSION

4.1 Pitch Input

Pitch input was tested at $-1.5° \leq \delta_p \leq 1.5°$ with $\delta_r = 0°$. Figure 6 and 7 show the cycle-averaged forces and moments versus the pitch input.

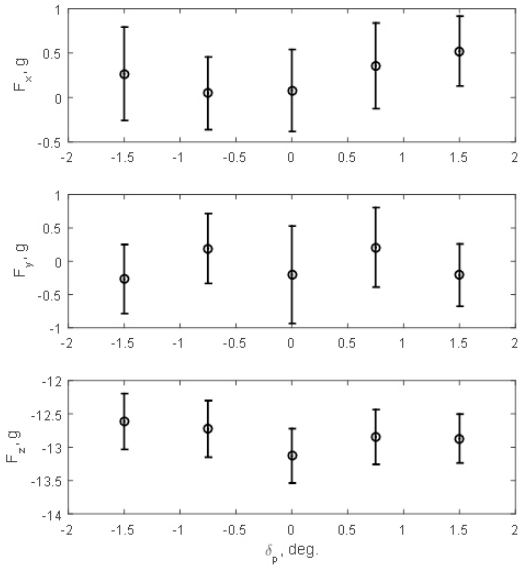


Figure 6: Cycle-averaged forces vs. pitch input.

http://www.imavs.org/

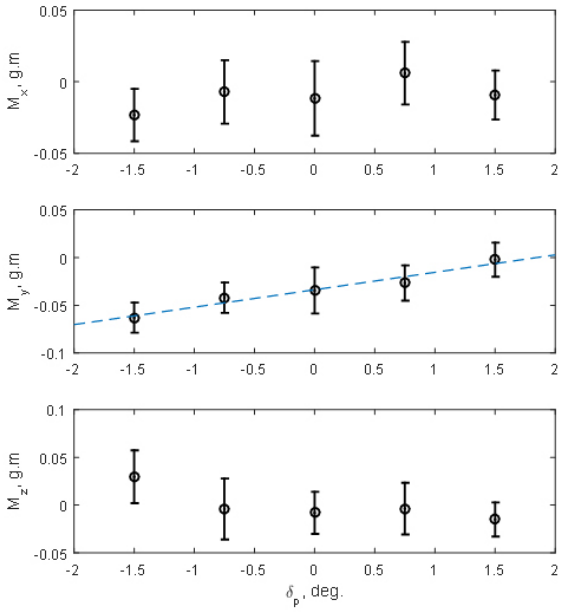


Figure 7: Cycle-averaged moments vs. pitch input.

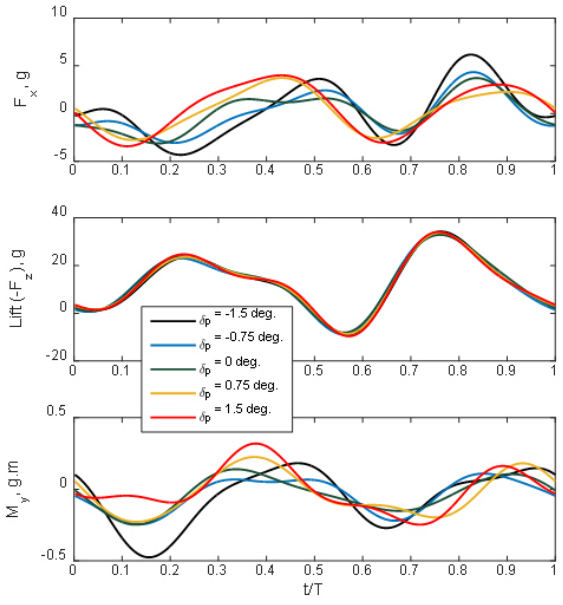


Figure 8: Time-resolved forces and pitching moment for various pitch inputs.

The pitching moment exhibits a linearly increasing trend in the figure. Other force and moment components are less affected by the input. The lift ($-F_z$) shows small decreases when the wings are tilted to either directions, but the drop is within 0.5g. The yawing moment (M_z) is also slightly affected nonlinearly, but it is likely due to small unequal tilt of left and right wings because the wings were tilted by different servos, it is very difficult to ensure both sides are tilted at the exact same angle.

The time-resolved forces and pitching moment is shown in Figure 8. The data starts at the beginning of upstroke, upstroke ends at t/T = 0.53, and down stroke ends at t/T = 1, consistent with the identified stroke angle shown in Figure 9. From Figure 8, tilting the wing roots rearward has very little effect on the lift ($-F_z$), but it causes significant increase on the forward force (F_x) during upstroke, staring at t/T = 0.15. This is consistent with the larger rotation angle at the 0.55 wingspan shown in Figure 10. Larger rotation angle during upstroke translates to larger angle of attack, thus creating larger drag, which results in larger forward force. Larger forward force should contribute to larger pitch up moment because the centre of force should be located lower than stroke plane, where the CG is assumed to be. The forward force and pitching moment are highly correlated, both are most affected by the wing roots tilting during the upstroke. Pitching moment can be contributed by the forward force and lift dissymmetry between the front and rear wings. It is quite difficult to evaluate the front-rear lift dissymmetry because the load cell measures only the total lift, but the wing kinematics might provide a probable explanation.

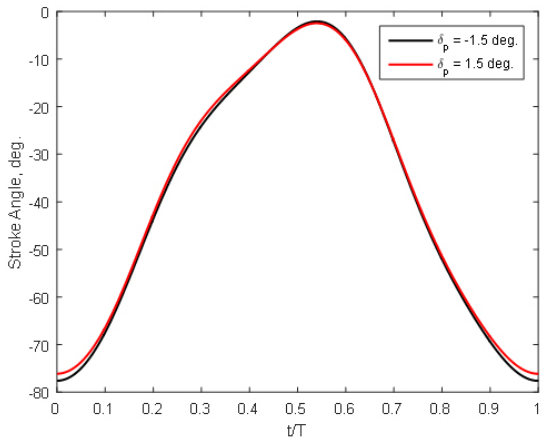


Figure 9: Stroke angle of the pitch input cases.

Figure 11 shows the clap-and-fling between the right and left front wings. Only the cross section of the left front wing is reconstructed, the right front wing is mirrored. Timestamps are shown on the figure so that the wing cross section can be associated correctly with data in Figure 8 to 10. The wings are far apart during the clap-and-fling, which is something that requires further improvement on the flapping mechanism. It

is also due to the distortion caused by data projection in a flat plane.

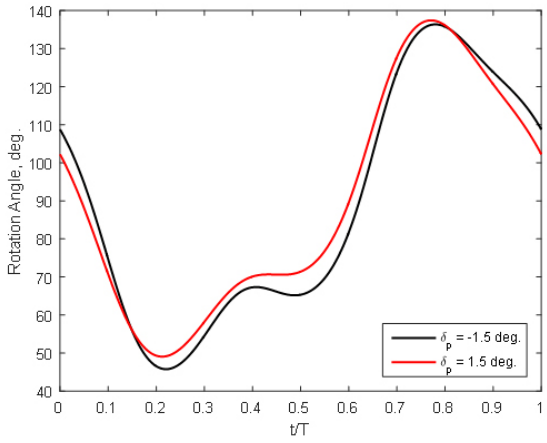


Figure 10: Rotation angle at 0.55 wingspan of the pitch input cases.

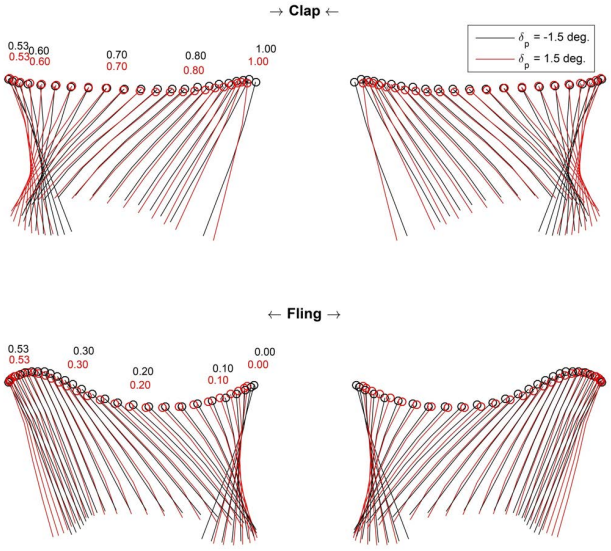


Figure 11: Clap-and-fling at 0.55 wingspan of the pitch input cases.

As seen in Figure 11, as the wings are tilted rearward ($\delta_p = 1.5°$), the wings are under tension when they clap. It causes the wing to have slightly lesser stroke angle and wing rotation occurs earlier. The earlier wing rotation could produce more effective clap-and-fling, which produces higher lift on the front wings compared to the rear wings, and contributes to the higher pitching moment.

In summary, tilting the wing roots to the rear can create a linearly increasing trend in the pitching moment while the other force and moment components are less affected. The increase of pitching moment could be due to increase of drag during upstroke and the lift dissymmetry between front and rear wings caused by more effective clap-and-fling on the front wings.

4.2 Roll Input

Roll input was tested at $-1.5° \leq \delta_r \leq 1.5°$ with $\delta_p = 0°$. Figure 12 and 13 show the cycle-averaged forces and moments versus the roll input. Similar to the pitching case, the rolling moment exhibits a linearly increasing trend in the figure. Other force and moment components are less affected by the input. The lift (-F_z) shows small decreases when the wings are tilted to either directions, but the drop is within 0.5g. Unlike the pitching case, the yawing moment (M_z) is not affected at all. Different from the pitching arms, the rolling arm is driven by one servo only. Due to the similarity of how the wings are being tilted in both the rolling and pitching cases, the fact that the yawing moment is not affected in the rolling case proves that the effect on yawing moment in the pitching case is caused by unequal tilt of the left and right wings.

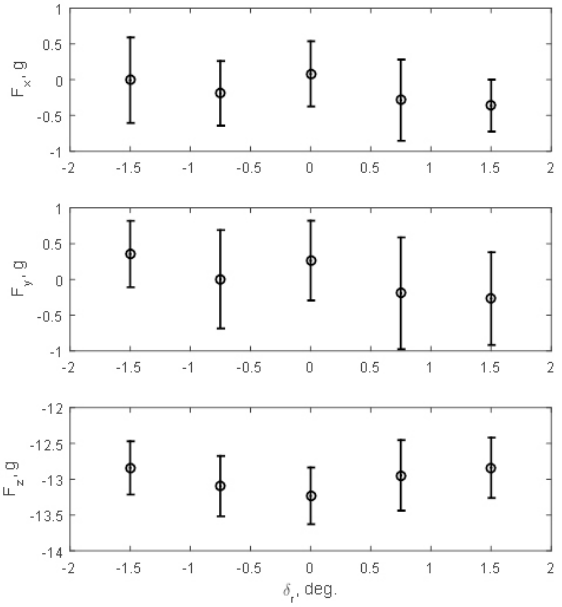


Figure 12: Cycle-averaged forces vs. roll input.

The time-resolved forces and rolling moment is shown in Figure 14. Similar to the pitching case, the data starts at the beginning of upstroke, upstroke ends at $t/T = 0.53$, and down stroke ends at $t/T = 1$, consistent with the identified stroke angle shown in Figure 15.

http://www.imavs.org/

http://www.imavs.org/

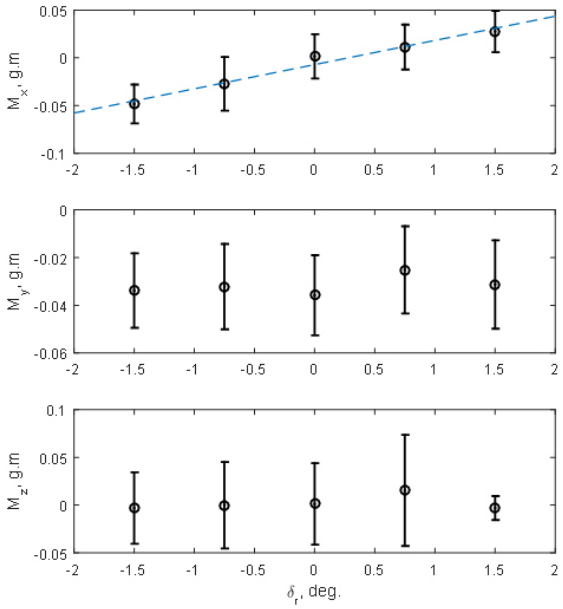


Figure 13: Cycle-averaged moments vs. roll input.

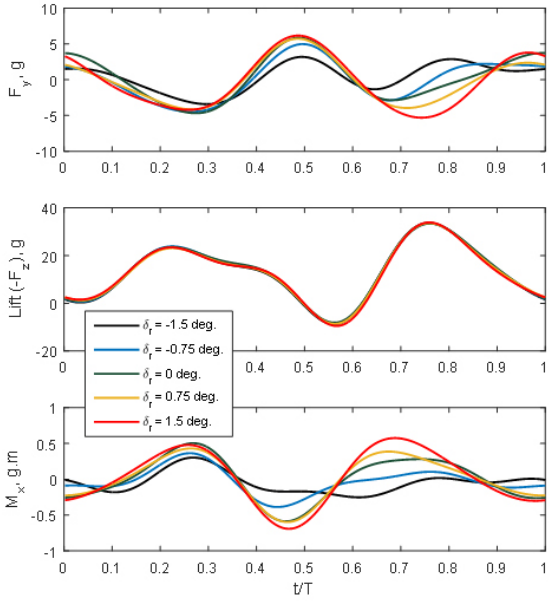


Figure 14: Time-resolved forces and pitching moment for various roll inputs.

From Figure 14, tilting the wing roots rightward has very little effect on the lift ($-F_z$), but it causes significant magnitude increase on the rightward force (F_y) during down stroke, staring at t/T = 0.65. This is consistent with the lesser rotation angle at the 0.55 wingspan shown in Figure 16. Lesser rotation angle during down stroke translates to larger angle of attack which results in higher drag. The drag causes larger negative rightward force during down stroke. The negative rightward force contributes to the roll right moment.

The rightward force and the rolling moment are inversely correlated, both are most affected by the wing roots tilting during the down stroke. Rolling moment can be contributed by negative rightward force and lift dissymmetry between the left wings and the right wings.

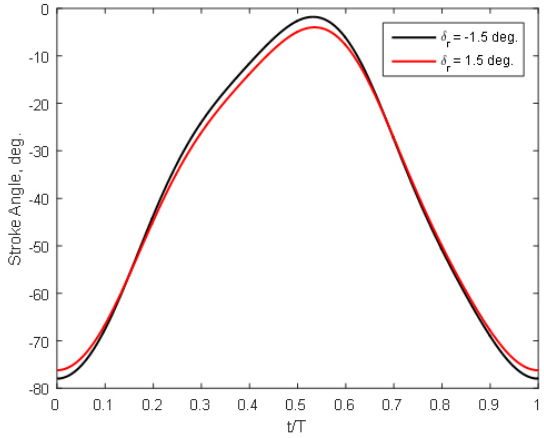


Figure 15: Stroke angle of the roll input cases.

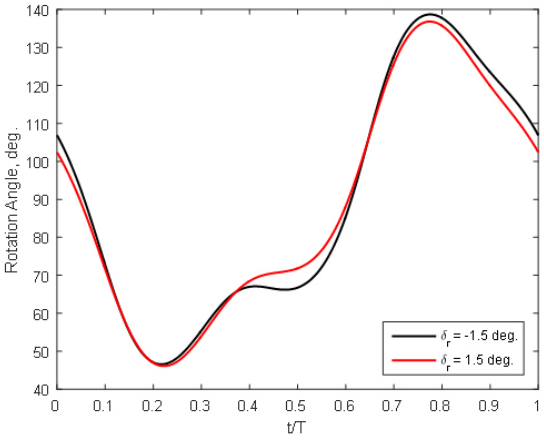


Figure 16: Rotation angle at 0.55 wingspan of the roll input cases.

Figure 17 shows the clap-and-fling between the left front and the left rear wings. Only the cross section of the left front wing is reconstructed, the left rear wing is mirrored.

Timestamps are shown on the figure so that the wing cross section can be associated correctly with data in Figure 14 to 16. Due to reasons mentioned earlier, the wings seem to be further apart than reality.

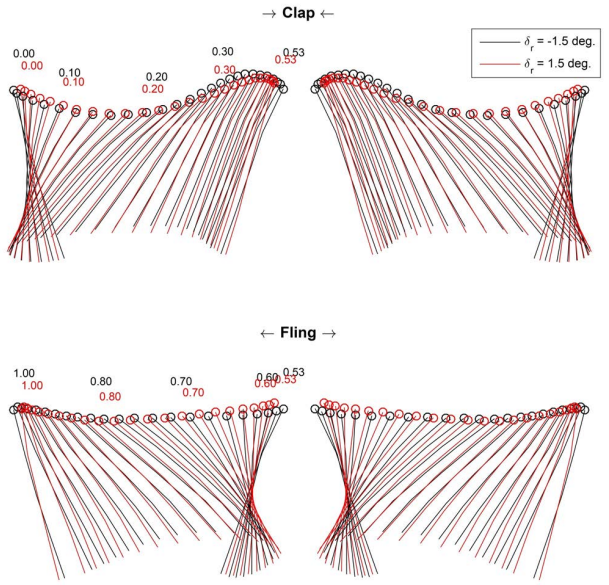


Figure 17: Clap-and-fling at 0.55 wingspan of the roll input cases.

Similar to the pitching case, as the wings are tilted rightward ($\delta_r = 1.5°$), the wings are under tension when they clap. It causes the wing to have slightly lesser stroke angle and wing rotation occurs earlier. The earlier wing rotation could produce more effective clap-and-fling, which produces higher lift on the left wings compared to the right wings, and contributes to the higher rolling moment.

In summary, tilting the wing roots to the right can create a linearly increasing trend in the rolling moment while the other force and moment components are less affected. The increase of rolling moment could be due to larger drag during down stroke and the lift dissymmetry between left and right wings caused by more effective clap-and-fling on the left wings.

4.3 Effect of Centre of Gravity Shift

It is expected that if the CG is shifted far upward or downward, the pitching and rolling moments will be dominated by the effect of forward and rightward forces respectively. Given the fact that the cycle-averaged values of the forces do not exhibit any trend with the tilt angles, it is very likely that the observed linear trend of pitching and rolling moment shown in Figure 7 and 13 would only occur if the CG is within certain range where the effects of lift dissymmetry are dominant.

To evaluate this, the moments were transformed into different coordinate systems. The coordinate systems were shifted up and down from the original position by Δl_z. Positive Δl_z indicates shifting down. the figure of merit in this evaluation are the linear fit slope and the norm of residual of the linear fit. The linear fit slope indicates the effectiveness of the moment generation, the higher the number, the more effective it is. The norm of residual indicates the linearity of the trend, the lower the number, the more linear it is.

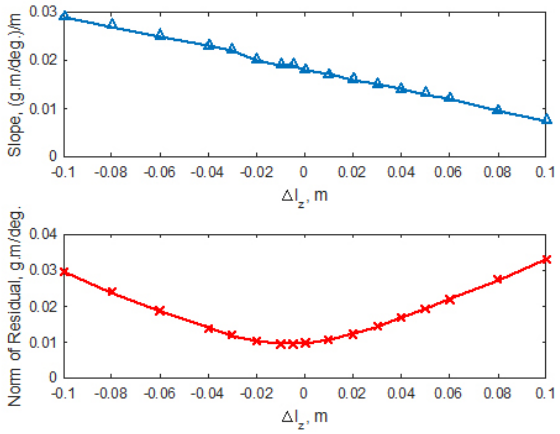


Figure 18: Slope and norm of residual of linear fit of pitching moment vs. pitch input at various CG locations.

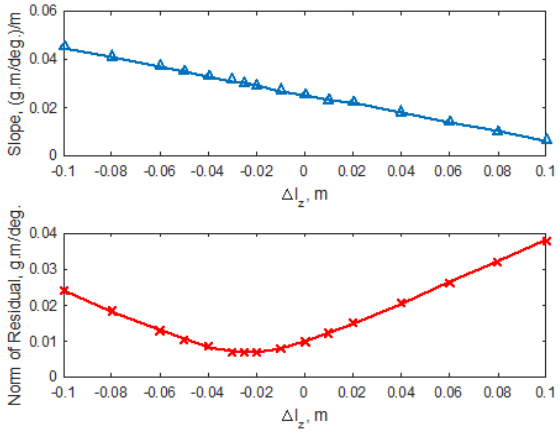


Figure 19: Slope and norm of residual of linear fit of rolling moment vs. roll input at various CG locations.

As shown in Figure 18 and 19, the higher the CG is, the more effective the moment generation is. In other words, having the CG positioned very low would make the wing tilt mechanism useless. Having the CG positioned very high is also not desirable because the trend might become highly nonlinear. The best location of the CG is between 0 to 30mm above the stroke plane as far as linearity is concerned. This

http://www.imavs.org/

finding is very interesting because the effectiveness and linearity of this tailless control mechanism on similar flapping wing platform has to be taken into account during the design stage of the flying machine.

4.4 *Roll-Pitch Coupling*

There are two types of couplings between pitch and roll. One is the mechanical coupling which is caused by the design of the mechanism. In this case, the pitching arms are installed on the rolling arm. Rotating the rolling arm would change the pitching arm angle by a little. Another type of coupling is the aerodynamic coupling, which is caused by the aerodynamics effect when the wings are tilted forward-rearward and left-right at the same time.

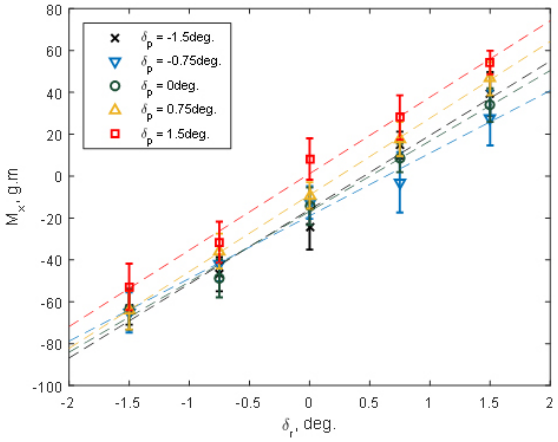


Figure 20: Coupling effect on rolling moment.

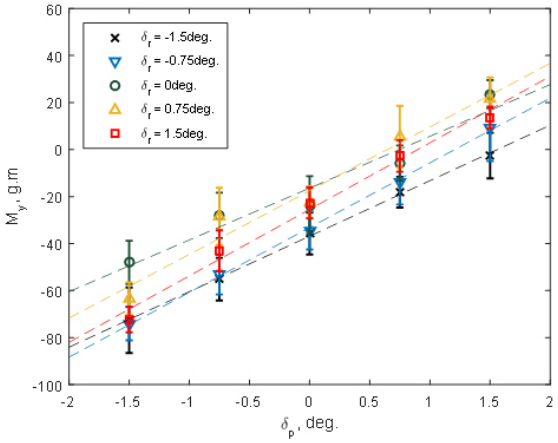


Figure 21: Coupling effect on pitching moment.

The mechanical coupling of this mechanism is very small due to the small tilting angles and the utilization of ball link, but aerodynamic coupling is unavoidable. This paper does not take the effort to explain the coupling, but simply to investigate the significance of the coupling effect.

Figure 20 and 21 show the coupling effect on rolling moment and pitching moment. Generally, the linear relationship retained, but the trim point (zero moment) shifted. The coupling seems to have more significant effect on the trim point than the linearity. While the linear slopes are quite similar, the trim point shifts in an ambiguous pattern.

5 CONCLUSION

A tailless control mechanism was designed, fabricated, and tested on a four-winged double clap-and-fling flapping wing platform. The mechanism tilts the wing roots in forward-rearward and left-right directions creating pitching and rolling moments respectively. Tilting the wings rearward and rightward create positive pitch up and roll right moments respectively. The change in pitching moment is caused by the change in forward force and the front and rear wings lift dissymmetry. But the linear trend is very likely to be cause by the lift dissymmetry because as the CG is shifted far up and far down, where the forward force effect is dominant, the linearity is not retained. Similarly, the linearity of the rolling moment is likely to be caused by the left-right wings lift dissymmetry.

From the wing kinematics, the lift dissymmetry is believed to be cause by the earlier wing rotation at the end of the stroke which resulted in a more effective clap-and-fling. However, this claim has to be further investigated.

The best CG position to maintain linearity of the moments is found to be in between 0 to 30mm above the stroke plane. This is quite a high CG configuration, and very difficult to achieve on flying platform.

Coupling between pitch and roll control are also shown in the paper. Generally, when pitch and roll inputs are applied together, the linearities of the moments are retained, but result in an ambiguous shift of the trim point.

ACKNOWLEDGEMENTS

Authors appreciate the financial support from Future System and Technology Directorate of Singapore (Programme No. PA 9014100626).

REFERENCES

[1] Q.V. Nguyen, W.L. Chan, and M. Debiasi. Hybrid design and performance tests of a hovering insect-inspired flapping-wing micro aerial vehicle. *Journal of Bionic Engineering*, 13(2):235–248, 2016.

[2] Q.V. Nguyen, W.L. Chan, and M. Debiasi. Performance tests of a hovering flapping wing micro air vehicle with double wing clap-and-fling mechanism. In *International Micro Air Vehicle Conference and Competition (IMAV)*, 2015.

http://www.imavs.org/

[3] W.L. Chan, Q.V. Nguyen, and M. Debiasi. Pitch and yaw control of tailless flapping wing mavs by implementing wing root angle deflection. In *International Micro Air Vehicle Conference and Competition (IMAV)*, 2014.

[4] W.L. Chan, K.S. Cheng, and Q.V. Nguyen. Investigation on tailless control of a flapping wing mav. In *International Micro Air Vehicle Conference and Competition (IMAV)*, 2015.

[5] M. Karasek, I. Romanescu, and A. Preumont. Pitch moment generation and measurement in a robotic hummingbird. *International Journal of Micro Air Vehicles*, 5(4):299–310, 2014.

[6] M. Karasek, A. Hua, Y. Nan, M. Lalami, and A. Preumont. Pitch and roll control mechanism for a hovering flapping wing mav. In *International Micro Air Vehicle Conference and Competition (IMAV)*, 2014.

[7] M. Takagi, T. Nishimura, and T. Miyoshi. Development of a roll and yaw moment generation mechanism with flapping amplitude control. *Journal of Aero Aqua Biomechanisms*, 4(1):56–62, 2015.

[8] H.V. Phan and H.C. Park. Pitch, roll, and yaw moment generator for insect-like tailless flapping-wing mav. In *Proc. SPIE 9797, Bioinspiration, Biomimetics, and Bioreplication*, 2016.

[9] M. Keennon, K. Klingebiel, H. Won, and A. Andriukov. Development of the nano hummingbird: a tailless flapping wing micro air vehicle. In *50th AIAA Aerospace Sciences Meeting*, 2012.

[10] H. Frontzek, E. Knubben, N. Gaissert, R. Mugrauer, G. Mugrauer, A. Jebens, and K. Jebens. Bionicopter. In *FESTO Report 54814 en 4/2013*, 2013.

[11] H. Frontzek, E. Knubben, R. Mugrauer, G. Mugrauer, A. Jebens, K. Jebens, C. Rabe, and N. Gaiert. emotionbutterflies: Ultralight flying objects with collective behaviour. In *FESTO Report 50058 en 4/2015*, 2015.

[12] R.J. Wood. Design, fabrication, and analysis of a 3 dof, 3cm flapping-wing mav. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.

[13] K.Y. Ma, P. Chirarattananon, S.B. Fuller, and R.J. Wood. Controlled flight of a biologically inspired, insect-scale robot. *Science*, 340(6132):603–607, 2013.

[14] T. L. Hedrick. Software techniques for two- and three-dimensional kinematic measurements of biological and biomimetic systems. *Bioinspiration and Biomimetics*, 3(3):034001, 2008.

http://www.imavs.org/

Towards a real-time aerial image mosaicing solution

Alexander Kern*, Markus Bobbe and Ulf Bestmann
TU Braunschweig, Germany

http://www.imavs.org/

ABSTRACT

Real-time aerial image mosaicing is a crucial task for future search and rescue missions. Solving the correspondence problem, estimating a valid transformation and visualizing the result is computational intensive. It becomes more challenging if the flying platform is a small micro air vehicle (MAV), which limits the available margin for payload significantly.

This paper proposes a robust algorithm that is able to create and update photo maps in a fixed period of time. The approach uses a high number of features and strict match filtering to allow robust matching without additional sensor information. Subsequently the ability of todays most common single board computers (SBC) to run the presented algorithm is examined. Together with the selection of a lightweight board camera the setup is less than 100 grams allowing even small MAVs to generate maps in real-time.

1 INTRODUCTION

In recent years the development of unmanned aircrafts considerably reduced the costs and effort required to generate aerial images. As a result aerial mapping is getting more common in various fields such as agriculture or construction side documentation. Common mapping results are 2D pseudo-orthophotos and 3D surface meshes. Both are commonly generated using structure from motion or photogrammetry workflows which are started after the survey flight and usually take several hours.

The capability for cost and time efficient aerial mapping is also needed by various emergency response units. In civil catastrophe scenarios for instance after earthquakes or tsunamis these maps can be used by rescue workers to get an overview of the situation. While a 3D mesh of the area generated by structure from motion or photogrammetry can fulfill the need for an all around perspective at best, it is computationally intensive and therefore too time consuming. Certainly time is of critical importance to maintain an effective coordination of the emergency aid to save human lifes.

This paper aims to provide a solution for a 2D real-time mapping implementation. In its first half a suitable mosaicing algorithm is discussed and presented. Afterwards the hardware requirements are determined and compared to hardware available today. The ultimate goal is to create a lightweight standalone package capable of performing the image acquisition and mapping task independently from the aerial vehicle it is attached to.

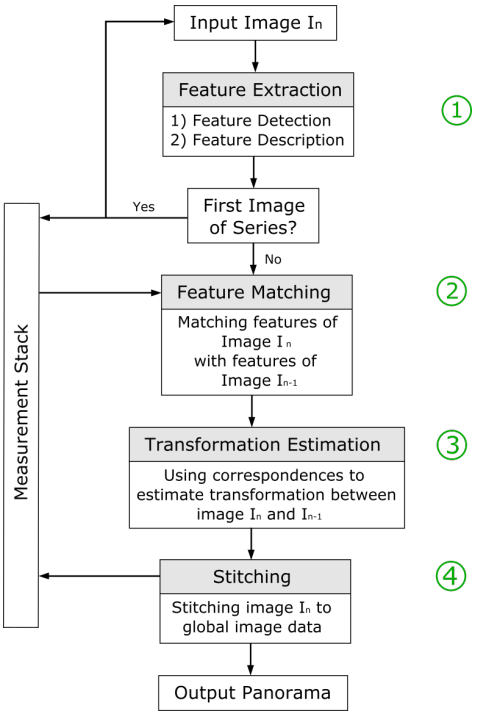


Figure 1: General approach to image mosaicing.

2 REAL-TIME CAPABLE MOSAICING ALGORITHM

In this section the focus lies on the algorithms used for image mosaicing. The choice is highly influenced by the requirements emerging from real-time performance. While classic approaches use bundle adjustment to minimize the reprojection error over all images in a single, iterative step to get a global consistent solution, it is not feasible for real-time applications. The computational time is too high and will increase with every image taken as the complexity increases with $O((m+n)^3)$, where m is the number of images and n is the number of structure points [1]. The real-time performance imposes the requirement of a constant maximum time frame for every stitching iteration. This time frame has to be independent from the number of images that have already been taken. To allow such performance the extensive OpenCV 3.0 C++ library is well suited and is therefore used for this task.

*contact: al.kern@tu-braunschweig.de

http://www.imavs.org/

2.1 Feature Extraction

In figure 1 the general approach for image mosaicing is displayed. As soon as a new image is acquired and sent to the pipeline, the first step is initiated. This step combines detection of markable points and description of the pixels near environment. For this task the ORB algorithm is used. It is based on the FAST keypoint detector and the BRIEF description of these keypoints [2]. The fact ORB uses a binary descriptor allows very efficient logical operations and comparisons, which fits the need for a real-time capable system excellently.

2.2 Feature Matching

If the input image is the first of the series it is then saved with all its features, since there is no other data it can be stitched to. After the acquisition and feature extraction of the next image the search for corresponding points is initiated. To ensure a fixed time frame the current input image is only stitched to the previous, thereby matches between these two sets of features have to be found. It is the most crucial part as it influences the final transformation notably. After generating a list of matches by brute force comparing their descriptor it is therefore beneficial to filter these results. A common method to identify good matches is the ratio test [3]. It compares the best and the second best match for each feature by their euclidean distance. A good match has a high chance of not having a second best match too close, which would be an indicator for a homogenous image section.

Additionally the previously introduced characteristic of ORB as a binary descriptor is used. It describes the features near environment with 1 and 0. Comparing two descriptors bit by bit and quantifying the difference by the hamming distance a proposition about the likeliness of two features can be done. That way the set of raw matches can be reduced to a set of best matches. Figure 2 shows the movement of the matches from two consecutive frames. On the top the movement of the raw matches is displayed, whereas on the bottom the filtered matches can be seen. In figure 3 the resulting stitched scene based on 43 aerial images is displayed. It should be noted, that reducing the matches drastically can result in decreasing transformation stability despite their quality, as they may not be uniformly spread across the image leaving more degrees of freedom. A good compromise between quality and quantity of matches should therefore be intended and can be achieved by selecting proper parameters for the pipeline depending mainly on the processing power and the overall performance requirements.

To improve the overall stability of the algorithm by minimizing the total drifting error it is furthermore beneficial to use an initial guess of the position to compute image neighboorhood relations. This can be done by kalman or particle filtering, additional sensor data like GPS or simple linear extrapolation of image centroids. If neighbours can be found within a defined range, matched features with the previous frame can be reprojected into the neighboring images to increase the number of observations and therefore ultimately improve transformations. At this step advanced state-of-the-art algorithms perform local bundle adjustment to optimize camera pose and 3D world points for even higher accuracy [4]. Considering the increasing complexity accompanied with numeric optimization a simpler approach for our system was chosen. The neighbourhood relations are used to compute additional matches between more than two frames resulting in reduced error propagation, which is evaluated in chapter 4.

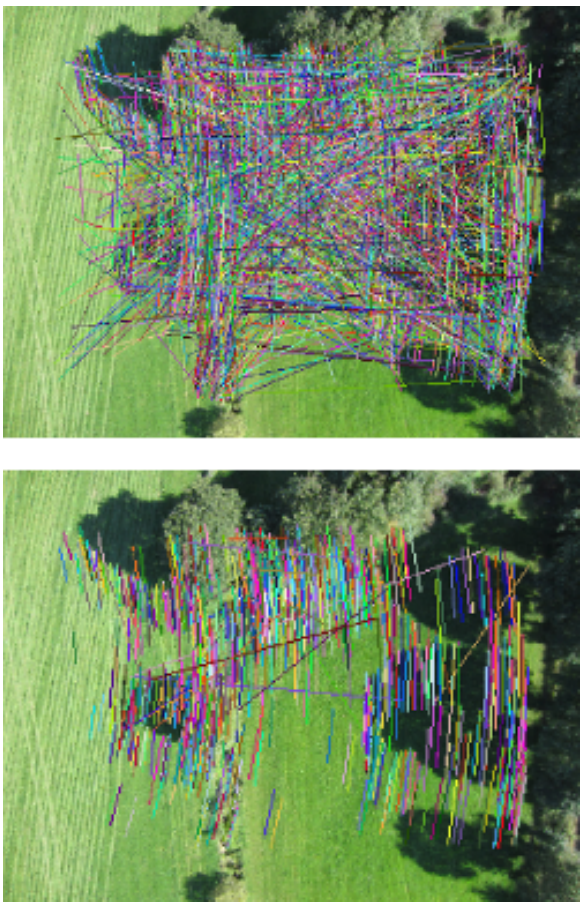


Figure 2: Movement of the matched features from two consecutive frames with unfiltered matches on the top and filtered by ratio test and hamming distance on the bottom.

2.3 Transformation estimation

After generating features and determining their corresponding points the transformation estimation starts. The most common method for real-time capable mosaicing approaches is the identification of the homography. It describes the relative position of two planes by a 3x3 matrix with 8 degrees of freedom, called perspective transformation. How-

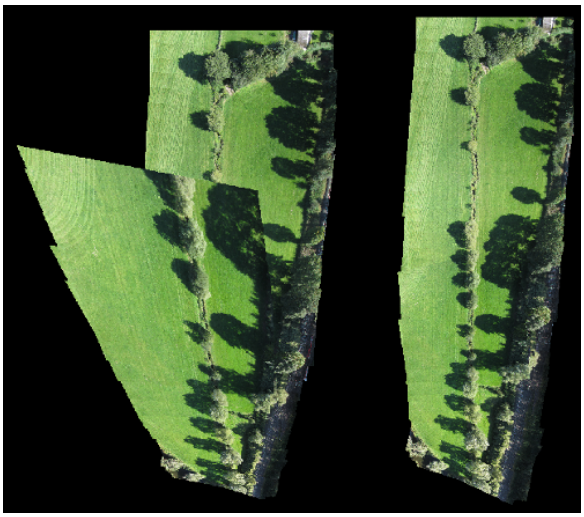


Figure 3: Stitched result with unfiltered matches on the left and with ratio and hamming filtering on the right for a test dataset.

ever in practice it has proven to be more reliable to use pitch and roll stabilised image data for mapping, reducing the overall complexity of the homography to

$$H = \begin{bmatrix} A & t \\ O^T & 1 \end{bmatrix} \tag{1}$$

where A represents a 1D rotation and scaling matrix and t the translation vector. Thus with 3 corresponding points the similiar transformation can be identified. It is important to be aware, that while estimating the homography the minimal geometric error between input and reference image is calculated. However the reference image in the pipeline is always the last stitched image. To achieve global consistent solutions it is therefore convenient to first transform all features used for calculation into the reference coordinate system and then compute the homography [5]. That way the input image is aligned relatively to the global reference and not to the untransformed previous image. Additionally random sampling consistency (RANSAC) is applied to reduce the number of outliers and obtain the maximum transformation accuracy during the process.

2.4 Image composition

In case a valid transformation matrix was found the input image must then be composited visually with the rest of the data. This step contains a high risk of breaking the real-time requirement, as the growing global map must be updated in a fixed time frame. Therefore only the section of the image that actually changed has to be considered. To achieve this a common approach is to project the edges of the input image into the global reference system using the previously estimated homography. In conclusion the maximum dimension of the warped input image is known and can be reduced to a region of interest for the visualization.

Additionally to increase performance even more the visualization process can be decoupled from the rest of the pipeline in aspects of image resolution. While working on lower resolution data for fast but stable feature extraction and matching the calculated matrix can be scaled afterwards. This approach also implies sending only raw image data and the corresponding transformation informations to the user. Once received, the frame can be aligned in high resolution to the global reference by using the complete processing power of the ground station. Furthermore implementing it that way, lost data packages during uplink to the UAV will not be a relevant problem as the solution on-board stays consistent the whole time.

3 HARDWARE SELECTION

In the previous section a potential real-time capable image mosaicing algorithm was descriped. The next step is to choose a lightweight hardware that is able to satisfy the defined requirements.

3.1 SBC Selection

The performance of the proposed algorithm is mainy influenced by two parameters. First is the image resolution, second the number of features that are extracted. However these parameters are not independent from each other. A high resolution image has a lot of details that can be detected as markable points. The higher the resolution, the more markable points and the more features can be extracted. To ensure matching features it is therefore necessary to extract as many markable points as possible while keeping the computational time low. This is aggravated by the fact that during matching brute force is used. In conclusion every feature is compared with each other resulting in an $O(n^2)$ complexity.

Following the approach given by [6] the selection of a sufficently capable single board computer (SBC) is now possible. Their framework quantifies the performance of the developers computer in OpenCV by running two different standard algorithms and measuring their respective processing time T_1 and T_2. The first is a simple $(ComplexityC_1 = 0\%)$, the second a complex one $(C_2 = 100\%)$. By interpolating linear between these two sampling points the user is able to identify the complexity C_{alg} of his own algorithm with

$$C_{alg} = \frac{T_{alg} - T_1}{T_2 - T_1} C_2 + C_1 \tag{2}$$

In the next step [6] tested common SBCs with both algorithms defining T_1 and T_2 on common SBC hardware. With a known C_{alg} on the developers computer T_{alg} on the SBC can then be estimated using

$$T'_{alg} = \frac{C_{alg} - C_1}{C_2} (T'_2 - T'_1) + T'_1 \tag{3}$$

http://www.imavs.org/

http://www.imavs.org/

Table 1 shows the resulting processing performance of the proposed mosaicing algorithm with a working resolution of 480 x 360 pixels and roughly 500 extracted features per image on different SBCs. Furthermore table 2 shows the specifications of the boards allowing to choose the best matching component concerning size and weight. Overall the Brix board is the best compromise by performance and weight. It also outperforms the Intel NUC while being less than half as heavy. However considering the goal of a very lightweight setup the Odroid XU3 offers the best solution. In our finals system an Odroid XU4 was selected as it comes with the same processing hardware but less periphery.

Board	Estimated fps
ITX i7	31.3
Brix	53.5
NUC	43.3
Odroid XU3	12.3
Odroid U3	8.1
ITX atom	9.3
Jetson	6.6
Rasp. Pi B	0.6

Table 1: Processing performance for the proposed algorithm with different SBCs [7].

3.2 Camera specifications

At first it is important to determine the general mission requirements. In an emergency aid scenario a human should be indentifiable on the aerial images. Therefore a resolution of 5 pixels per 30 cm should at least be maintained resulting in a ground sampling distance (GSD) of 6 cm / pixel. Taking the image resolution defined in section 3.1 into account this leads to an image ground dimension of 28.8 x 21.6 m. The vehicle operation altitude can vary but was chosen to be at least 30 m to prevent collisions with high trees or other obsticles.



Figure 4: UI-1221LE camera developed by IDS.

For a lightweight setup the use of board cameras is a promising option. In figure 4 the μEye UI-1221LE developed by IDS is displayed. With a resolution of 0.36 Megapixel, a maximum of 87.2 fps and global shutter it is a suitable camera for this mission. Following the definition of the GSD

$$GSD = h_{rel} \frac{d}{f \, n_{Pixel}} \tag{4}$$

with the relative altitude h_{rel} above ground, d as the camera chip width and n_{Pixel} as image width the focal length of the camera lens can be deduced. Applying h_{rel} = 30 m, d = 4.52 mm, n_{Pixel} = 480 pixels and GSD = 6 cm / pixel the focal length of the camera is estimated with equation 4 to

$$f = h_{rel} \frac{d}{GSD \, n_{Pixel}} = 4.7mm \tag{5}$$

Finally to guarantee a stable stitching process a high overlap between two consecutive frames is recommendable. Tests showed that an overlap of at least 75% is required for a stable stiching process.

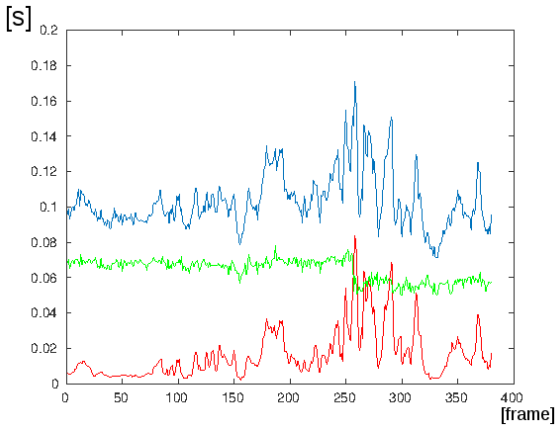


Figure 5: Processing time distribution for each image with image loading (green), feature detection (red) and total processing time (blue)

4 EVALUATION

The proposed algorithm was tested on the determinded hardware using an available dataset. At first the performance of the SBC is analyzed and compared with the estimation made in the previous section. Subsequently the accuracy of the calculated trajectory is evaluated using a GPS reference. The data set used was published by [8] in 2016 and includes 381 images of a village captured by a UAV in an height of 165 m. Following section 3.2 the image resolution should be at least 480x360 pixels with a height of 30 m and 75% overlap. The latter parameters are overly fulfilled by the data set, which results in higher stability of the algorithm with less ground resolution. This can not be transferred directly to the

Name	Processor	Memory	Weight [gram]	Power@100% [Watt]	Volume[cm^3]
mini-ITX I7	Intel i7-4770S	16GB	684	68	1815
Brix	Intel i7-4500	8GB	172	26	261
NUC	Intel i5-4250U	8GB	550	20	661
Odroid XU3	Samsung Exynos 5422	2GB	70	11	131
Odroid U3	Samsung Exynos 4412	2GB	52	7	79
mini-ITX atom	Intel Atom D2500	8GB	427	24	1270
Jetson	Cortex A15	2GB	185	13	573
Rasp. Pi B	ARM1176JZF-S	512MB	69	4	95

Table 2: Processing performance for the proposed algorithm with different SBCs [7].

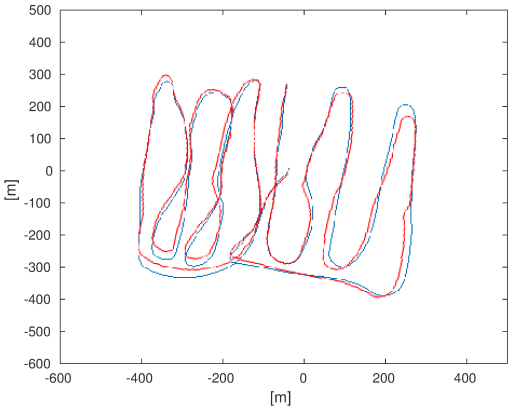


Figure 6: Trajectory generated by the GPS receiver (blue) and the stitching algorithm (red) with additional neighbour frames

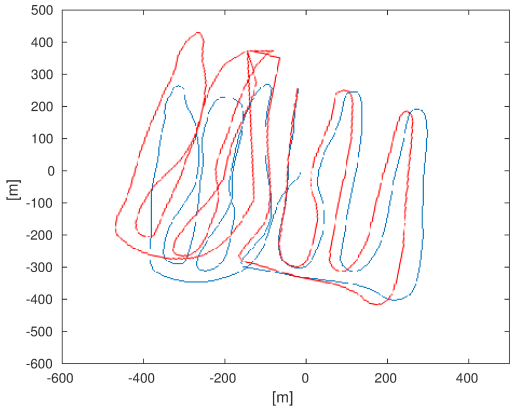


Figure 7: Trajectory generated by the GPS receiver (blue) and the stitching algorithm (red) without additional neighbour frames

mission requirements, as the resolution is not enough to identify humans safely. But it should be sufficent to verify the general performance and accuracy capabilities. Additionally the resolution can be adjusted in the real scenario as proposed in section 2.4 for visualization purposes.

4.1 SBC Performance

The extrapolations made in section 3.1 indicate the Odroid XU4 to run the stitching pipeline at 12.3 FPS. Using the given data set a mean runtime of 103 ms per frame was achieved. This concludes 9.7 FPS for the designed system making it slightly slower than estimated. The processing time for each image is displayed in figure 5. It can be noticed that image reading from the harddrive (green) shapes the mean processing time significantly, while feature matching (red) defines the overall variance. Despite the small variations around the mean value the complexity can be identified as constant per frame.

4.2 GPS - Image trajectory comparison

To evaluate the accuracy of the transformations and therefore the quality of the mapping process a comparison between UAV trajectories will be displayd. The trajectory produced by the proposed algorithm was analyzed by following the x- and y-coordinates of image centroids in the global reference frame. The scale was extracted by identifying markable centroids from the images and measuring their distance in satellite images. Calculating the meters per pixel and applying this informations to the rest of the data produced the red output in figure 6. The GPS trajectory measured by the UAV (standalone, single frequency receiver) on the other hand is displayed in blue. Figure 7 in contrast shows the calculated trajectory for the same data set without additional neighbour frame matching. It can be noticed, that the overall visual consistency only fits the first leg flown. Even though an alignment of the trajectories was found visually by assuming the error to be minimal at start, the error propagation of the transformation estimation obviously grows and affects the mapping solution negatively. However for future analysis another possibility might be to align the starting points, calculate a least squares transformation for the first centroids and applying this to the rest of the data. That way error propagation can be directly calculated between every single measurement, allowing a more distinct analysis.

http://www.imavs.org/

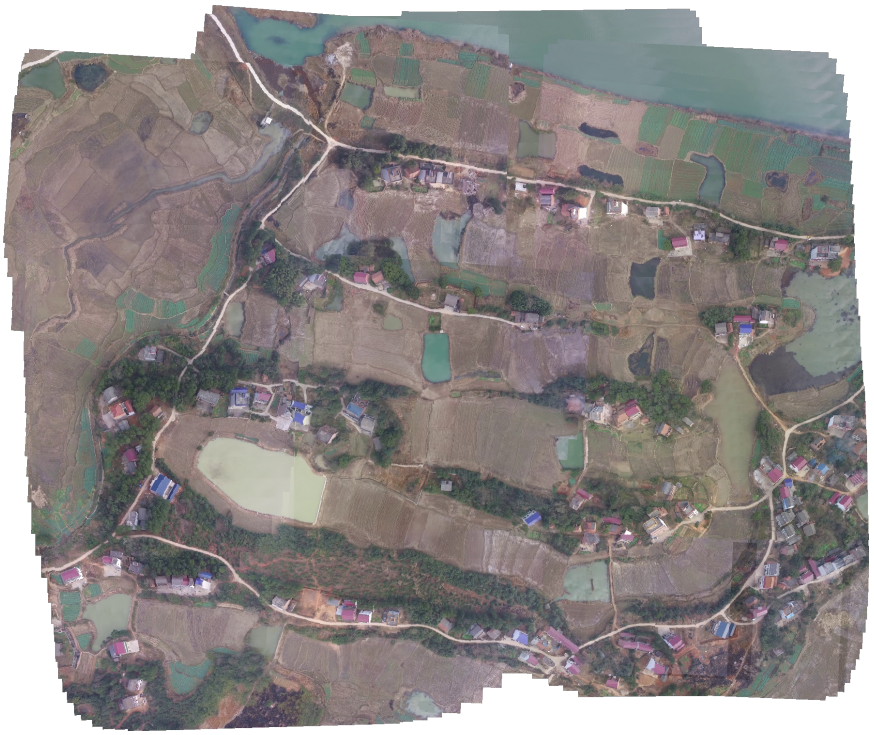


Figure 8: Public image sequence data set visualized by the proposed stitching algorithm

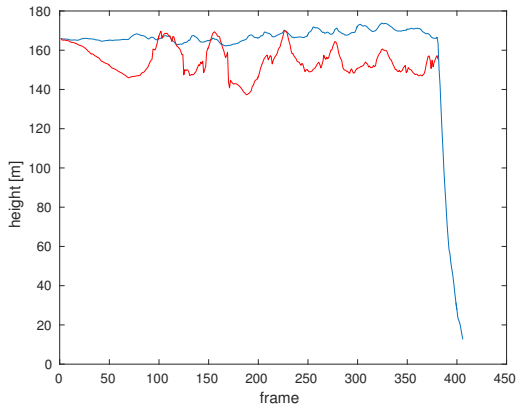


Figure 9: Height measurements by GPS (blue) and stitching algorithm (red)

Additional analysis can be achieved by decomposing the homography into rotation, scale and translation. Subsequently by multiplying the resulting scale with the initial UAV height of 165 m a direct comparison to the GPS height measurements can be done. Figure 9 shows this comparison. On the left the total height measured by image processing is displayed in red while GPS data is in blue. Significant is the sinusoidally characteristic of the plot. This is also an indication for the reduced error propagation achieved by neighbour frame matching. The first leg flown by the UAV ends at frame 77 followed by the second leg until frame 105. In this exact period the error drops down to zero. This repeats constantly following the flight routine of the UAV with a steady growing error offset.

5 CONCLUSION

A lightweight setup for image mosaicing was introduced together with a real-time capable stitching algorithm. The feature based approach solves the problem using only image data and no additional sensor informations. Tests with a public dataset showed promising results reducing the error propagation through homography estimation by extracting high amounts of markable points and filtering the matches afterwards.

The evaluation of available SBCs revealed that the Odroid XU4 is capable of running the pipeline with an average of 9.7 FPS. In combination with a board camera like the UI-1221LE the whole setup is light and can further be developed to a standalone module making real-time mapping available for a variety of MAVs. Final flight tests with the defined setup will show if an additional camera stabilization is required and are planned for the very near future.

http://www.imavs.org/

REFERENCES

[1] Kaushik Mitra and Rama Chellappa. A scalable projective bundle adjustment algorithm using the l infinty norm. *Sixth Indian Conference on Computer Vision, Graphics Image Processing*, 2008.

[2] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. *2011 International Conference on Computer Vision*, 2011.

[3] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[4] Montiel J. M. M. Mur-Artal, Raúl and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[5] Taygun Kekec, Alper Yildrim, and Mustafa Unel. A new approach to real-time mosaicing of aerial images. *Robotics and Autonomous Systems*, 62:1755–1767, 2014.

[6] Dries Hulens, Jon Verbeke, and Toon Goedem. How to choose the best embedded processing platform for on-board uav image processing? *VISSAPP 2015*, 2015.

[7] Dries Hulens. Embedded processing board selection tool. www.eavise.be/hulens/selectiontool.html, 2016.

[8] Gang Wan Zhenbao Liu Shuhui Bu, Yong Zhao. Map2dfusion: Real-time incremental uav image mosaicing based on monocular slam. 2016.

http://www.imavs.org/

Visual Homing for Micro Aerial Vehicles using Scene Familiarity

Gerald J.J. van Dalen,* Kimberly N. McGuire and Guido C.H.E. de Croon
Delft University of Technology, The Netherlands

http://www.imavs.org/

ABSTRACT

Autonomous navigation is a major challenge in the development of Micro Aerial Vehicles (MAVs). Especially when an algorithm has to be efficient, insect intelligence can be a source of inspiration. An elementary navigation task is homing, which means autonomously returning to the initial location. A promising approach uses learned visual familiarity of a route to determine reference headings during homing. In this paper an existing biological proof-of-concept is transferred to an algorithm for micro drones, using vision-in-the-loop experiments in indoor environments. An artificial neural network determines which control actions to take.

1 INTRODUCTION

A major challenge in robotics is to navigate autonomously through an unknown environment. Especially in indoor scenes, where no Global Positioning System (GPS) system is available, the entire navigation problem is not yet solved.

Current navigation algorithms either require expensive sensors or significant computation power. Especially Simultaneous Localization and Mapping (SLAM) methods have shown to be successful in real-time navigation, given enough computational power on-board a vehicle or good sensors. Most Micro Aerial Vehicles (MAVs) do not have such sensors and cannot perform heavy computations on-board the vehicle.

In order to find suitable navigation algorithms for MAVs, insects can be a source of inspiration, since they constantly have to deal with complex navigation problems while only having small-sized brains [1]. Different algorithms have already been created based on observations done on insects. A well-known example is using optic flow to get a sense of velocity, which is known to be done by insects [2]. Integrating this estimate for localization is called visual odometry. The obtained location estimate is employed in higher level navigation algorithms. Still, these algorithms are not readily available for tiny MAVs yet. One of the higher level skills employed by insects is the ability to return to the nest location. This is referred to as *homing* [3]. It would be an important enabler for MAVs, if they could use similarly high-level, but computationally efficient algorithms for navigation.

*Email address: gjj.vandalen@gmail.com

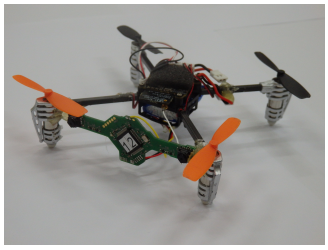


Figure 1: Pocket drone: a micro quad rotor containing a Lisa-S autopilot and a stereo camera [5]. While this pocket drone can already fly, stabilize and avoid obstacles, in this paper we investigate efficient insect-inspired algorithms that will allow it to navigate in an unknown environment.

A promising homing algorithm is proposed by Baddeley et al., where familiar views along a route are used to determine the correct direction to an earlier visited location [4]. This is a *visual* homing algorithm, since cameras are used as driving sensor. Instead of focussing on the contruction of a detailed (or coarse) map, Baddeley et al. propose that homing can be performed just by means of recognizing which direction seems most familiar to a robot. Furthermore, they use a small neural network to store and recapitulate a route in order to find the initial location. Potentially, this is very useful for MAV navigation algorithms, since it deals with limited storage capacity found on many small platforms, like the pocket drone shown in Figure 1.

In an effort to find efficient navigation algorithms for MAVs, this paper investigates the practical application of the scene familiarity algorithm on MAVs. The focus is on how robust familiarity is to determine control actions.

First, section 2 discusses the state-of-the-art in autonomous visual navigation on drones. Then, section 3 explains the scene familiarity method as introduced by Baddeley et al. Section 4 shows simulations and experiments for different environments, to overcome current shortcomings in the implementation described by Baddeley et al. Finally, closed-loop simulation flights are performed and presented in section 5, to show a more realistic use-case of view familiarity for MAV homing.

2 RELATED RESEARCH

This section gives a brief overview of previous research done to visual navigation and specifically visual homing. Visual SLAM is the most commonly used algorithm in camera-

http://www.imavs.org/

driven robotics. An example is shown in Motard et al., where an AIBO robot[1] must navigate back to its charging station [6]. Still, visual SLAM algorithms in real-time require much computational resources, since (visual) processing, mapping and self-localization must be performed simultaneously. Since most MAVs have limited computational resources, visual SLAM often cannot be run in real-time, which makes it less suitable for homing.

In 1983, Cartwright & Collett introduced the Snapshot Model [7]. The framework they presented gives an explanation of the navigation capabilities of bees when traveling between different food sources. The visual matching is done by a direct comparison of an image on the retina with a stored snapshot. The landmark approach is further extended by the addition of visual beacons [8]. A disadvantage of this is that many images have to be stored.

A similar approach uses Average Landmark Vectors (ALVs) to represent landmarks [9]. ALVs, introduced by Lambrinos et al. in 1998, are averages of the heading vectors to all landmark locations [10]. The homing vector is determined with respect to this ALV. ALV homing stores the location of interest as a vector, which is more efficient in computation and storage, than storing an entire image. However, due to its simplicity, ALV homing is also more prone to errors.

Scene familiarity methods refer to recognition of a traversed route, without specific information about the goal location. This means, a robot must always move into the *most familiar* direction. In the ideal case, this would automatically mean that the agent returns to the goal location. In 2012, a scene familiarity method is proposed for visual homing of desert ants [4]. The scene familiarity method proposed by Baddeley et al. is quite new and not yet used in robotic applications. The next section reviews their paper in depth.

3 THE SCENE FAMILIARITY METHOD

In an effort to find a biologically more plausible alternative to map-based navigation methods and the snapshot model described in the previous section, the scene familiarity homing method is introduced [4]. To show that homing navigation could take place without the use of visual odometry, a method is presented where views along the entire route determine the heading in which to proceed. Conceptually, this means that during a training run images in the direction of the route are stored. Then, when using the algorithm for homing, images taken around the robot are compared to these stored views in order to determine the most familiar direction.

When the homing capabilities are tested, the agent is placed back at its initial location. From there, homing is done by performing 360° scans of the world and comparing images taken in each direction with all images stored. A familiarity value of a single image is obtained by calculating the Sum of Squared Differences (SSD) of raw pixel values, as defined in Equation 1 [11].



Figure 2: Binary panoramic image used in Baddeley et al. [4].

$$F(I) = -\arg\min_i \sum_{x,y} \left(I(x,y) - V_i(x,y)\right)^2 \qquad (1)$$

In this equation, $F(I)$ indicates the familiarity of view I, $I(x,y)$ is the current view and $V_i(x,y)$ are the stored views. It can be seen that the stored image that gives the closest match to the current image is used as familiarity value. The agent can rotate on the spot or use an omni-directional camera to obtain familiarity values in all directions. After determining the most familiar direction (by maximizing the values obtained with Equation 1), the simulated agent is moved in that direction.

The stored panoramas are binary images and have dimensions of 90 by 17 pixels (Figure 2). The resolution is such that each pixel in horizontal direction is equivalent to a rotation of 4°. During homing, familiarity is evaluated for steps of 1 pixel, such that Equation 1 is evaluated 90 times. The maximum outcome of this results in the most familiar direction.

Due to the large memory needed for storing images and the computational requirements, the algorithm in the current form is not yet suitable for implementation on-board a small robot. Baddeley et al. therefore also study an unsupervised Infomax neural network to approximate familiarity [12]. The network is a two-layer neural network, where the linear combination of an input and the network weights represent familiarity. A lower value indicates *more familiar*. The training rule therefore adapts the weights such that the value is lower for every input encountered during training.

Baddeley et al. showed the validity of scene familarity with virtual robotic ants in a simulated environments. However, they use an environment of binary sceneries, which are not representative for the scenes through which a robot must navigate. Moreover, the simulation is set up such that moving the image by one pixel in the horizontal direction is equivalent to a rotation of the agent of 4°. These direct relations to rotation and pixel difference are not realistic for real-life cameras. Furthermore, the algorithm has only been tested on relatively small distances, since images are stored every $4cm$ and movements of $10cm$ per timestep are made. When the method is implemented in robotics, the robot should be able to cover longer distances to make it more useful.

Recently, Gaffin et al. have published a detailed analysis on scene familiarity in realistic, indoor environments [13].

[1] http://www.sony-aibo.co.uk/

Distinguishing familiarity is both analyzed in rotation and translation, for raw pixel matching between images of different resolutions. A rail mounted camera is used to perform a MATLAB-driven experiment.

In our analysis of the scene familiarity method, we will use a simulator containing realistic sceneries, vehicle dynamics and camera parameters. A translation and rotation analysis will be performed as well, however, next to raw pixel values, we will also investigate alternative image representations, to determine which one is more suitable for recognizing familiar views. Closed-loop simulations with an MAV are presented and we show the use of an Infomax neural network as well, since this helps in meeting the limited storage requirements of an MAV. We hope to better understand autonomous navigation for small MAVs. In future work, the methods of Gaffin et al. should be applied to our simulation, to compare the obtained results.

4 FAMILIARITY ANALYSIS

In the previous section, the original simulation results presented by Baddeley et al. are discussed [4]. Based on this, a key question remains whether the algorithm will work in more realistic environments. In this and the following sections, an analysis of an indoor simulated environment is presented in combination with different image representation methods. First, the tested image representations and calculated performance measures are introduced. Then, simulation results of these different methods in multiple environments are shown. To validate this, similar results are shown on real imagery.

4.1 Methods

To test the usability of familiarity of scenes for visual homing, we investigate the familiarity sensitivity during both rotation and translation. Analyzing rotation is done by performing a 360° turn at a fixed location in the environment, in steps of 5°. A single image is stored and used as trained view and all other views experienced during this rotation are compared to this. The hypothesis is that familiarity should improve when the heading difference between the current view and the stored image decreases.

Translation is analyzed by evaluating familiarity in a grid of locations, with a fixed heading. Again, a single image is used as training sample and the familiarity is expected to improve when the distance to the trained view gets smaller. Results of this should show the sensitivity of familiarity with both increasing distance (in two directions) and increasing heading angle.

The following image representations are compared:

- **Raw pixel values** The sum of squared differences of each pixel in two images outputs a similarity score [11], as shown in Equation 1.
- **Texton histograms** Textons are small distinct image patches, which can be extracted from an image [14]. When clustered with a texton dictionary, histograms are formed which represent an image.
- **Hue Saturation Value (HSV) color histograms** Color histograms contain a classification of each pixel based on color intensity.

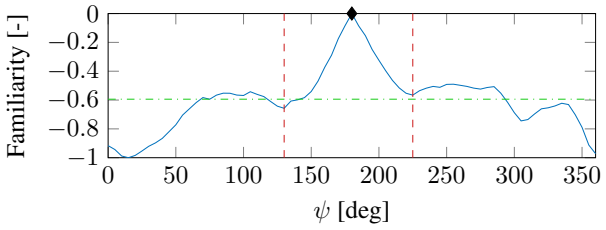


Figure 3: Rotation on the spot at a constant location in a simulator. Unfiltered images of 48 by 32 pixels are taken every 5° and compared to a stored image at a heading angle of 180°. The red dashed lines indicate the BoA bounds and the green dashed line shows the mean familiarity.

The performances of the different methods are evaluated by 1) looking at how distinct a view close to the trained view is, compared to other views and 2) what the probability is that the correct (i.e., trained) view is selected as most familiar, since that direction will be chosen for homing. Figure 3 shows an example of a familiarity evaluation when rotating on the spot. The trained image is positioned at an angle of 180° and, in this example, image matching is done using the SSD of raw pixel values. The performance is evaluated using the following measures:

- **Peak ratio** The peak ratio is defines as:

$$PR = \frac{\max F - \mu_F}{\max F - \min F} \tag{2}$$

  In this equation, F refers to the familiarity values shown in Figure 3 and μ_F is the mean of all familiarity values (i.e., the green line in the figure). The higher the peak ratio is, the more distinct a peak is.
- **Basin of Attraction (BoA)** The basin of attraction shows how far an agent can be off from the trained view, before diverging from the correct direction. It is evaluated by finding all local optima (both minima and maxima) and looking between which minima the agent converges towards the trained optimum familiarity (maximum).
- **Correlation coefficient** This is used to estimate the correlation between two neighboring heading angles, differing by 5°. Here, the Pearson product-moment correlation coefficient is used, where 1 indicates full positive correlation between two neighboring angles, -1 means full negative correlation and 0 means no correlation.

The BoA is considered to be most important, since it determines how far an agent can be off the route (i.e., the correct

http://www.imavs.org/



(a) (b)

Figure 4: Examples from the scenery used in SmartUAV simulations (a) and the validation Cyberzoo environment (b).

heading), while still being able to converge back to the correct path with a gradient-like search. The peak ratio is mainly useful when an agent has no clue where to go; if the agent makes a 360° turn and the trained peak is very distinct, the probability of continuing in the right direction is high. The correlation coefficient gives a measure for how continuous a familiarity curve is. When the correlation is low, it could happen that spikes occur in the familiarity curve, which may give wrong results.

4.2 SmartUAV Simulations

This section shows analyses for sceneries in the SmartUAV simulator. SmartUAV is made for Guidance Navigation & Control (GNC) research on MAVs and specializes in the use of vision as primary sensor. The simulator is written in C++ and sensors and controllers can be connected using a block interface. This makes it easily extendable and the level of simulation fidelity can be adapted by changing complexity of vehicle dynamics, sensor dynamics and realism of the environment.

The tested environment is based on a sports hall located in Delft (the Netherlands). The dimensions are 30 by 60 meters. Figure 4a shows an example view of the sports hall. This environment is used for both familiarity analysis and closed-loop simulations.

As mentioned, both rotational and translational familiarity sensitivity will be tested. For familiarity estimation, SSD values of raw pixels, SSD values of texton histograms and SSD values of HSV color histograms are used and compared. The familiarity sensitivity to yaw rotations is most important for view familiarity-based homing. Each turn taken during homing is made based on the familiarity values for different heading angles. To analyze familiarity for different headings, different image representations are compared by calculating the BoAs, peak ratios and correlation coefficients. An MAV is simulated at a single location and stores a representation of one view. This view is matched to images in all other directions to get a measure of familiarity. This is done in a grid of locations in the sports hall, to get imagery in the center of the room, as well as close to walls. For each location, the BoA, peak ratio and correlation coefficient can be calculated.

Table 1 summarizes these performance measures for the different methods. The calculated BoAs, peak ratios and correlation coefficients are averaged for all locations and the standard deviations are included as well. Good performance is characterized by large BoAs (i.e., it is likely that the correct heading is found), large peak ratios (i.e., the correct familiarity value is distinct compared to familiarities in other directions) and correlation coefficients close to 1 (i.e., continuous and not too noisy).

	Raw pixels	**Textons**	**HSV**
BoA average	37.3%	36.7%	6.90%
BoA std. dev.	16.5%	12.0%	3.77%
Peak ratio average	0.57	0.43	0.53
Peak ratio std. dev.	0.10	0.076	0.13
Corr. coeff. average	0.98	0.98	0.80
Corr. coeff. std. dev.	0.051	0.0091	0.14

Table 1: Average performance metrics during rotation, for each image matching method in the simulated sports hall.

Looking at the results, it can be seen that the BoAs for raw pixel matching and texton histogram matching perform similarly. HSV histogram matching performs much worse, which is also seen in the lower correlation coefficient. This indicates more local optima, which inherently decreases the BoA. The peak ratio is best with raw pixel matching, although the differences between the different methods are quite small.

To illustrate the results shown in the table, familiarity curves are shown in Figure 5. The top plot shows raw pixel matching, the middle texton histogram matching and the bottom one HSV histogram matching. The blue, solid lines indicate the average familiarity curves for all locations in the environment, the red dashed lines indicate two times the standard deviation and the gray lines show some example familiarity curves at individual locations in the sports hall. The results are scaled such that the average lies between 0 and 1.

As expected, all average curves show a single peak at the trained locations (i.e., at 180°). The HSV histogram result however, shows a less predictable outcome, with a larger amount of local optima. This is in line with the lower BoAs and correlation coefficients shown in Table 1.

To test familiarity sensitivity with translation only, images taken in a grid pattern are analyzed. In the sports hall the trained view is obtained in the centre of the room, which is matched against views from the entire room, keeping the heading angle constant. In contrast to rotation, translational motion is not directly controlled. For homing, only the heading angle is adjusted in order to reach the correct destination. This means that good performance in translation is characterized by a familiarity that does not change too much for small displacements. Stated differently: when a 360° turn is performed, it is advantageous when the familiarity curves are similar for proximate locations, so that good homing performance is achieved even when exploration and homing routes do not perfectly align. Figure 6 shows the results in the sports

http://www.imavs.org/

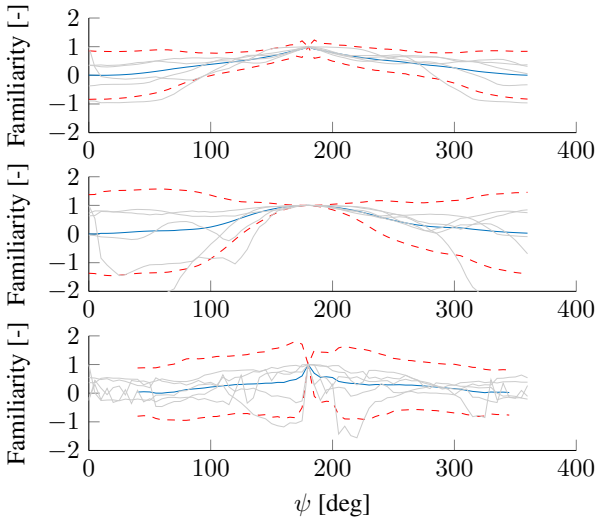


Figure 5: Average rotation on the spot of 231 locations in the sports hall environment in SmartUAV. Unfiltered images of 48 by 32 pixels are taken every $5°$ and compared to a stored image at a heading angle of $180°$. The red dashed lines indicate the 2σ bounds and the gray lines are some example familiarities. The top, middle and bottom plots indicate raw pixel matching, texton histogram matching and HSV histogram matching respectively.

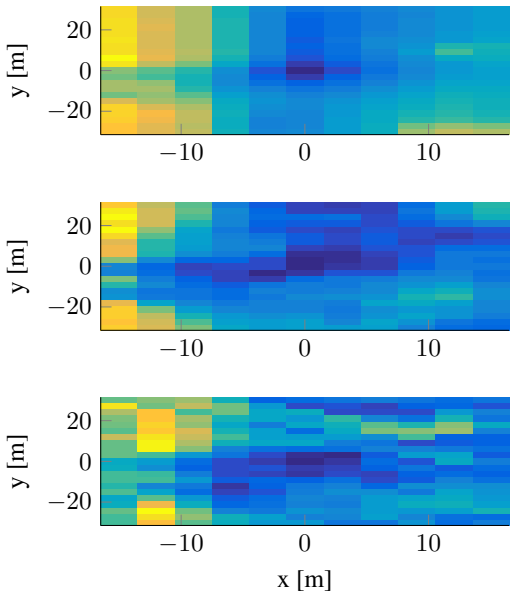


Figure 6: Varying x and y positions in a SmartUAV simulation in a sports hall, with constant heading angle. Unfiltered images of 48 by 32 pixels are taken in a grid pattern and compared to a stored image at the center of the grid (x=0 and y=0). The top figure uses raw pixel matching, the middle figure texton histograms and the bottom figure HSV histograms.

hall environment, for raw pixel matching, texton histogram matching and HSV color histogram matching. The colors indicate the familiarity of a certain location, where dark blue is most familiar.

From the figures it is clear that raw pixel matching shows the most distinct global optimum. Texton and HSV histogram matching however, show a larger region of optimal familiarity. This can be useful when the robot is slightly off-track, because rotational performance will be similar on different locations. However, both methods show several local minima, which can be disadvantageous for homing.

4.3 Validation Experiment

The previous analysis is done in simulation. To validate this, an experiment is shown using real imagery taken in an indoor environment. The environment used is the Cyberzoo; a flight arena located at the TU Delft, as shown in Figure 4b.

Validation is done for both rotation and translation. For rotation, videos of rotations on the spot are recorded, containing 25 videos in a grid of 5 by 5 meters. The average BoAs, peak ratios and correlation coefficients are computed, as in the simulations presented in the previous section. The results, including the corresponding standard deviations, are shown in Table 2. The first observation is that the BoAs are much smaller than in simulation. This is explained by more spikes (and hence local optima) in the results, which is confirmed by the lower correlation coefficients. It is however, in contrast with the observation in the previous section that more realistic environments yield higher BoAs.

The second observation is that texton and HSV histogram matching show slightly better BoAs than raw pixel matching. Due to the small differences and the large standard deviations however, no significant conclusions can be drawn from this. The corresponding rotation plots are shown in Figure 7.

	Raw pixels	**Textons**	**HSV**
BoA average	9.13%	12.7%	11.7%
BoA std. dev.	3.38%	6.57%	4.24%
Peak Ratio average	0.53	0.41	0.37
Peak Ratio std. dev.	0.054	0.095	0.093
Corr. Coeff. average	0.82	0.92	0.84
Corr. Coeff. std. dev.	0.093	0.025	0.14

Table 2: Familiarity performance metrics for each image matching method in the Cyberzoo environment.

Translation is validated by comparing images taken facing the same direction, in a grid of 49 locations. The results are quite similar to the simulation results and are shown in Figure 8. Again, the result for raw pixel matching shows a very narrow peak at the trained location. This can be disadvantageous for homing, since a small offset from the training

http://www.imavs.org/

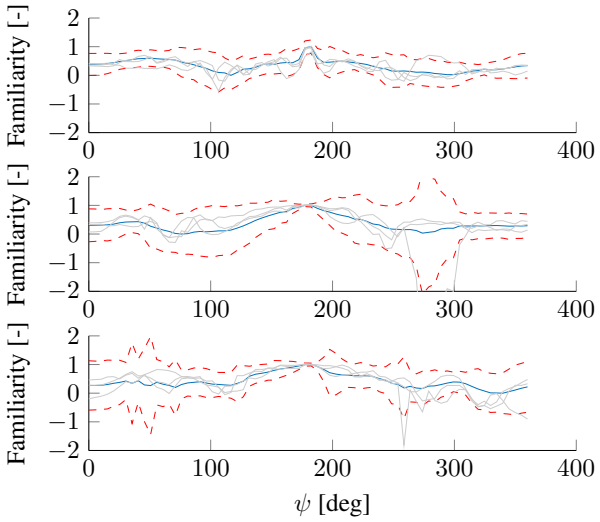


Figure 7: Average rotation on the spot of 25 locations in the Cyberzoo environment. Unfiltered images of 64 by 36 pixels are taken every $5°$ and compared to a stored image at a heading angle of $180°$. The red dashed lines indicate the 2σ bounds and the gray lines are some example familiarities. The three plots indicate raw pixel matching, texton histogram matching and HSV histogram matching respectively.

path can cause divergence from this path. When looking at the texton histogram matching result, it can be seen that two clear optima are present. Even though the surrounding region has quite similar familiarity values, the local optimum at $x = 3$ and $y = 2$ might result in wrong convergence.

Looking at both rotation and translation of HSV histogram matching, it can be observed that the real-life results are better than those made in simulation. This can be explained by more distinct colors in the validation imagery, such that more bins in the HSV histogram are filled.

5 CLOSED-LOOP SIMULATION FLIGHT

As mentioned in the previous sections, the recognition of views during rotation performs best for both raw pixel matching and texton histogram matching. Especially in simulation, the BoAs of these two methods are comparable. When observing familiarity during translations, both texton and HSV histogram matching show a large central region of similar familiarity. As explained earlier, this can be advantageous for homing, since recognizing the correct heading during rotations probably yields the same result for proximate locations. When looking at closed-loop results it is therefore expected that texton histogram matching will perform better than the other two methods.

To show a closed-loop simulation, a simulated robot is placed in the sports hall environment. A route is learned by flying backwards (with a speed of $0.5m/s$), such that the front

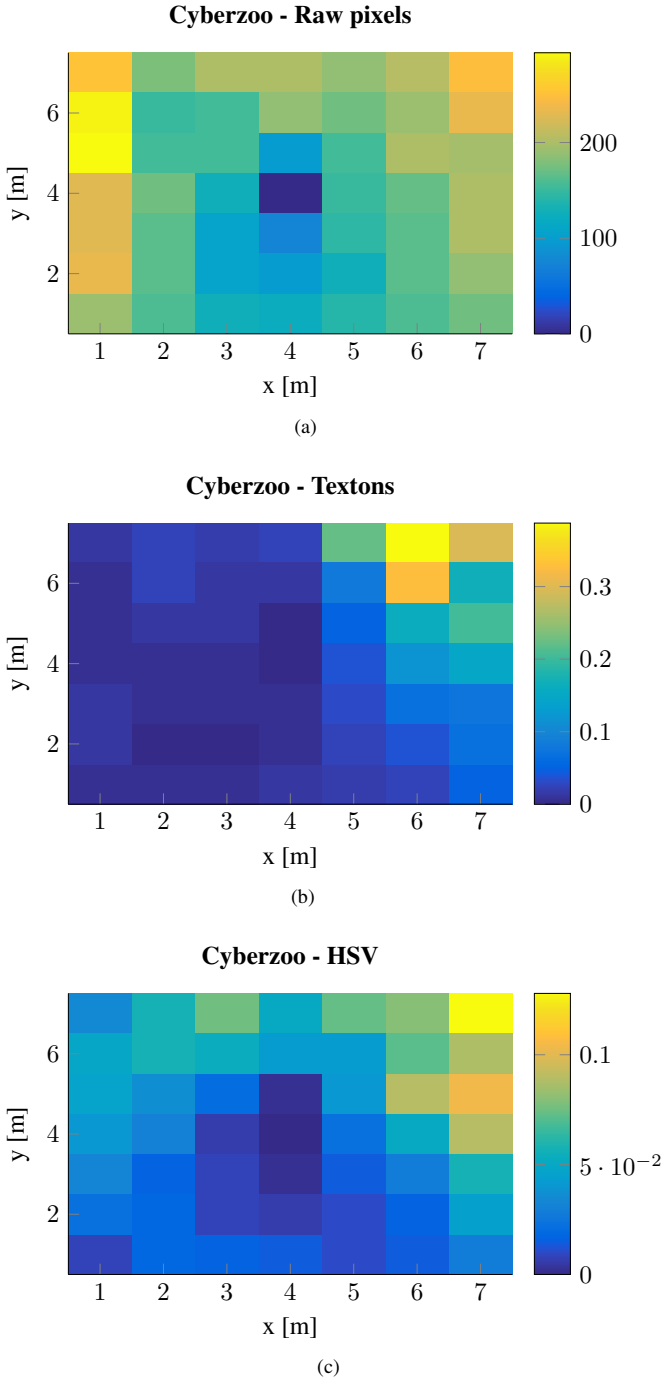


Figure 8: Varying x and y positions using pictures of the Cyberzoo environment, with constant heading angle. Unfiltered images of 64 by 48 pixels are taken in a grid pattern and compared to a stored image at the center of the grid (x=4 and y=4).

http://www.imavs.org/

http://www.imavs.org/

camera looks in the homing direction, which is necessary to use scene familiarity for homing. One third of the image taken at the center is used for training. When homing is initiated, the robot starts flying forward with a constant speed of $0.5m/s$ and the heading is constantly determined using view familiarity. This is done by selecting one third of the image giving the best match with one of the trained views. The center of this image patch is converted to an angle, to which the MAV is steered. Views are obtained from a forward looking camera, with a field of view of $90°$. The result is shown in the left part of Figure 9. Here, the blue solid line is the training route, starting at $x = 4m$ and $y = 12m$, which are arbitrarily chosen. A route of approximatelly $20m$ is flown.

From the results it can be seen that both texton histogram matching and HSV histogram matching approximatelly reach the initial location. The main difference is that texton histogram matching performs turns with a small delay, where HSV histogram matching turns too early. The delay can be explained by low frequency: because all possible patches are extracted from each image, texton histogram matching operates at approximatelly $1Hz$, where HSV histogram matching operates at approximatelly $20Hz$. Texton histogram matching can be significantly improved by using sub-sampling of textons, instead of extracting all. For HSV histogram matching it could be questioned whether it only performs well because the flying direction is approximatelly straight. When homing is done by matching raw pixels (performed at approximatelly $5Hz$), the robot diverges from the trained route. It does, however, follow the curvature of the trained path. The fact that raw pixel matching works worst suggests that differences in familiarity when performing small translational movements causes views to be hard to recognize.

As mentioned, the Infomax neural network can be used as function approximator of familiarity [12]. To test this in closed-loop, the three methods are all represented in a neural network. For both texton and HSV histogram matching a network with 50 inputs is defined (i.e., each histogram forms one input vector to the network). The number of novelty neurons is arbitrarily chosen to be 200. Furthermore, the number of epochs is set to 500. It turned out that a lower number of epochs gives significantly worse performance. In further simulations or flight tests this should be tuned by testing multiple numbers of both novelty neurons and epochs. For raw pixel matching, the image is scaled down to 16 by 12 pixels, which gives 192 inputs to the network. Larger dimensions as input cannot be performed in real-time. The number of novelty neurons and epochs are kept the same.

The results using an Infomax network can be seen in the right part of Figure 9. It is clear that the results are slightly worse than with a perfect memory (i.e., by keeping a database of images, texton histograms or HSV histograms). It does however, look quite similar to the perfect memory case, which suggests that the assumption that Infomax is only used as approximator for views is quite good.

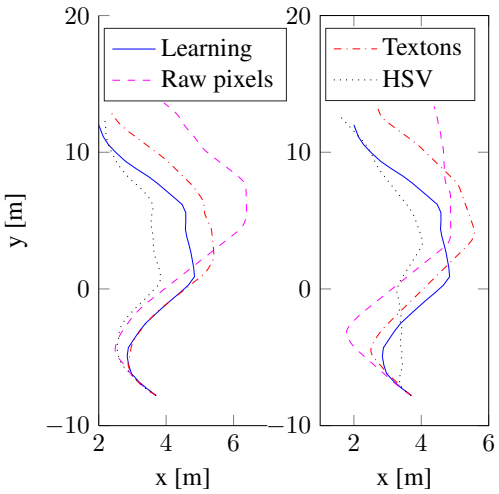


Figure 9: Closed-loop homing simulation in the sports hall environment in SmartUAV. On the left, a perfect memory is used; on the right the Infomax neural network is applied.

6 DISCUSSION

When first looking at the rotational analysis, it was observed that raw pixel and texton histogram matching performed best. When looking at the translation results, raw pixel matching shows the most distinct peak. Because position of the robot is not directly controlled, it is advantageous that a large familiar region appears in translation, so that a small displacement of the robot does not change the homing performance. This was especially the case for texton histogram matching and HSV histogram matching. This suggests that texton histogram matching would perform best, which is confirmed by the closed-loop results. Surprisingly, HSV histogram matching shows very good performance in closed-loop. A reason for this can be that generating and storing HSV histograms is computationally very efficient, which allows for a low timestep. This means corrections are made very quickly so that the robot does not diverge too much. It does not say however, that HSV histogram matching would perform well when divergence already happened.

When evaluating the closed-loop tests in this paper, some limitations can be identified. First of all, it is only tested in simulation. Although the fidelity of the simulation is higher than the simulations performed by Baddeley et al., it is questionable whether the same results would be obtained in a real flight. Furthermore, additions can be proposed to make the algorithm more robust. An example is to use active rotation instead of using the inherent field of view of the forward looking camera, such that bigger turns can be made. Alternatively, a camera with a larger field of view can be added. Another possibility is the use of visual odometry to get a rough estimate of the path taken. Odometry could be used to prevent severe divergence from the correct route. Since the experi-

ment enforces small turns only, it cannot yet be concluded that the method works well for diverse trajectories.

Another point of discussion is that the main reason scene familiarity can be a viable approach for visual homing of MAVs is computational efficiency. The only way this is tested in this paper, is by performing closed-loop real-time simulations on a laptop computer. When implementing the algorithm on-board an MAV, the real-time performance may be inadequate due to a slower micro-processor. The one exception was HSV histogram matching, because both the computations needed to extract histograms, and the storage capacity are limited. In this paper however, all textons were extracted from each image. Usually, it suffices to randomly pick a set of textons, which would drastically improve computational performance. The storage of a texton histogram is similar to storing an HSV histogram. A huge advantage of using a neural network is that the storage capacity is constrained. Even though this means that the network can *forget* earlier trained views (which is also investigated by Baddeley et al.), it allows control over the often very limited storage capacity on MAVs. Training on the other hand, is quite slow; especially when having to train each sample 500 times.

7 CONCLUSION AND RECOMMENDATIONS

This paper investigates the applicability of the scene familiarity homing method, observed from insect behavior, to MAVs. The scene familiarity method is introduced as proof of concept for desert ants to use the recognition along a route to find their way home. Next to this, an unsupervised neural network was used to keep storage of familiarity compact.

The concept of only using recognition along a route is a very interesting one. The analysis shows the closed-loop performance is good. The reason the method is promising, is the computational efficiency. Especially HSV histogram matching showed surprisingly good closed-loop performance while running quite fast. For the other two image representations the algorithm works in real-time on a laptop, although the frequencies in the current implementations are low.

It is concluded that using texton or HSV histogram matching is useful for visual homing on small robots. Once a route is lost, the risk of divergence is quite high. This must be further investigated. It seems very useful to combine scene recognition with existing methods like visual odometry. Especially when some more thought is put in optimizing the algorithm and sensor usage (like multiple or omnidirectional cameras), two computationally efficient methods can be combined to succesfully perform homing.

REFERENCES

[1] Z. Mathews, M. Lechon, J.M. Blanco Calvo, A. Dhir, A. Duff, S Bermudez i Badia, and P.F.M.J. Verschure. Insect-like mapless navigation based on head direction cells and contextual learning using chemo-visual sensors. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, New York, USA, oct 2009. IEEE.

[2] M.V. Srinivasan. Where paths meet and cross: navigation by path integration in the desert ant and the honeybee. *J Comp Physiol A*, 201(6):533–546, may 2015.

[3] R. C. Nelson. Visual homing using an associative memory. *Biological Cybernetics*, 65(4):281–291, aug 1991.

[4] B. Baddeley, P. Graham, P. Husbands, and A. Philippides. A model of ant route navigation driven by scene familiarity. *PLoS Computational Biology*, 8(1), jan 2012.

[5] B.D.W. Remes, P. Esden-Tempski, F. Van Tienen, E. Smeur, C. De Wagter, and G.C.H.E. De Croon. Lisa-s 2.8g autopilot for gps-based flight of mavs. Delft University of Technology, 2014.

[6] E. Motard, B. Raducanu, V. Cadenat, and J. Vitria. Incremental on-line topological map learning for a visual homing application. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, New York, USA, apr 2007. IEEE.

[7] B. A. Cartwright and T. S. Collett. Landmark learning in bees. *Journal of Comparative Physiology*, 151(4):521–543, 1983.

[8] T. S. Collett. Insect navigation en route to the goal: Multiple strategies for the use of landmarks. *Journal of Experimental Biology*, 199:227–235, 1996.

[9] D. Lambrinos, R. Möller, T. Labhart, R. Pfeifer, and R. Wehner. A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems*, 30(1-2):39–64, jan 2000.

[10] D. Lambrinos, R. Möller, R. Pfeifer, and R. Wehner. Landmark navigation without snapshots: the average landmark vector model. In *Proceedings of Neurobiology Conference Göttingen*, 1998.

[11] J. Zeil, M.I. Hofmann, and J.S. Chahl. Catchment areas of panoramic snapshots in outdoor scenes. *Journal of the Optical Society of America A*, 20(3):450, mar 2003.

[12] A.J. Bell and T.J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, nov 1995.

[13] D.D. Gaffin and B.P. Brayfield. Autonomous visual navigation of an indoor environment using a parsimonious, insect inspired familiarity algorithm. *PLOS ONE*, 11(4):e0153706, apr 2016.

[14] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *Int J Comput Vision*, 62(1-2):61–81, apr 2005.

http://www.imavs.org/