Effect of Incorporating Aerodynamic Drag Model on Trajectory Tracking Performance of DJI F330 Quadcopter

by

Kashyap Sathyamurthy Nolastname

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved March 2020 by the
Graduate Supervisory Committee:

Wenlong Zhang, Chair
Sze Zheng Yong
Spring Berman

ARIZONA STATE UNIVERSITY

May 2020

ABSTRACT

Control algorithm development for quadrotor is usually based solely on rigid body dynamics neglecting aerodynamics. Recent work has demonstrated that such a model is suited only when operating at or near hover conditions and low-speed flight. When operating in confined spaces or during aggressive maneuvers destabilizing forces and moments are induced due to aerodynamic effects. Studies indicate that blade flapping, induced drag, and propeller drag influence forward flight performance while other effects like vortex ring state, ground effect affect vertical flight performance. In this thesis, an offboard data-driven approach is used to derive models for parasitic (bare-airframe) drag and propeller drag. Moreover, thrust and torque coefficients are identified from static bench tests. Among the two, parasitic drag is compensated for in the position controller module in the PX4 firmware. 2-D circular, straight line, and minimum snap rectangular trajectories with corridor constraints are tested exploiting differential flatness property wherein altitude and yaw angle are constant. Flight tests are conducted at ASU Drone Studio and results of tracking performance with default controller and with drag compensated position controller are presented. Root mean squared tracking error in individual axes is used as a metric to evaluate the model performance. Results indicate that, for circular trajectory, the root mean squared error in the x-axis has reduced by 44.54% and in y-axis by 39.47%. Compensation in turn degrades the tracking in both axis by a maximum under 12% when compared to the default controller for rectangular trajectory case. The x-axis tracking error for the straight-line case has improved by 44.96% with almost no observable change in the y-axis.

DEDICATION

To my father, my mother and all the researchers who are interested in quadrotor aerodynamic effects and control.

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to my advisor Dr. Wenlong Zhang for directing my progress during the pursuit of my research and for his invaluable inputs that have made me a better person today. I also want to thank my committee members Dr. Sze Zheng Yong and Dr. Spring Berman for their inputs and comments.

My loving parents who have always had my back no matter what the circumstance.

The coordination and support of lab managers Dr. Bruce Steele, Rhett Sweeney, and Nicholas Beck needs a special mention as without them experiments wouldn't have been possible.

A special thanks to Shatadal Mishra for his inputs and consistent support in navigating through the overall research process. I would also like to thank other members of quadcopter team Karishma Patnaik, and JJ for help with debugging and moral support.I would also like to acknowledge Hansol Moon a fellow lab member for helping me out with tolerance and fitting issues that arose during prototyping. Piyush Hota, for his inputs on sleeve design for propeller drag testing.

Last but not the least I would like to thank god for this wonderful opportunity that I think will be highlight of my time here in this university.

CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

Drones are emerging as more than just a recreational platform thanks to the electronics hardware which is becoming more and more compact, economical and yet continue to provide powerful computing. Currently, drones already are an integral part of airspace, assisting humans in providing solutions to the most critical problems.

Application of drones or unmanned aerial vehicles have significantly increased when compared to last decade and the market for commercial drones sector alone is projected to be about 2.91 million units by 2023 [1]. They are deployed in varied applications including but not limited to search and rescue [2] [3], package delivery [4] [5], human-drone interaction [6], aerial grasping and manipulation [7] [8], aerial inspection of power transmission lines [9] or of inaccessible areas [10], and operation in confined spaces[11].

Classification of drones can be done in three major categories viz. fixed wing, multicopters, and Vertical Take-Off and Landing (VTOL) drones. Fixed wing unmanned aerial vehicles have been aiding the military with reconnaissance and aerial surveillance missions for several years. Multicopters' hovering capability is widely exploited with also applications requiring fast agile flight. VTOL is a platform consisting of hybrid functionalities of both multirotors and fixed wing. The next generation of aerial transportation involves hybrid electric unmanned VTOL vehicles [12] which are scaled up versions of these smaller drones.

A new category of quadrotors is being researched and developed that can be added to the above classification of drones. These are reconfigurable quadrotors and unlike

usual quadrotors that have rigid arms, these have morphing characteristics both on the ground and mid-flight [13]. One useful application might be autonomous internal inspection of canals which have diameters that are small for rigid quadrotors to fly through. Another way of exploiting this unique property could be flying through narrow gaps even when performing aggressive maneuvers [14].

It is very crucial to consider the operating conditions of the environment in which these drones will be deployed in addition to the nature of the mission as it has direct influence on in-flight dynamics. For instance, at lower altitudes within urban areas the presence of wind gusts is minimal and so is the disturbance on multirotors when hovering. However, beyond certain altitudes or in a vast open land where multirotors are free from buildings, the incoming airflow might be unsteady and have higher velocity causing instabilities during flight that might lead to a crash. Moreover, during forward flight, due to the interaction of freestream air and rotating propeller, aerodynamic effects are induced which might hamper performance. Hence in order to better understand the interaction of the aerodynamics of the vehicle during different operating conditions, it is vital to develop physics-based or data-driven models that will eventually aid in improved flight performance. This not just applies to multirotors but all other forms of aerial vehicles, however, in this thesis the emphasis is on multirotor vehicles, specifically quadrotors.

1.1 Motivation

Often it is difficult to observe through human eyes the influence of an aerodynamic phenomenon on quadrotors. Depending upon the mission that they are deployed for and the location they operate in, aerodynamics might in turn affect performance.

Usually, aerodynamics is used to design efficient propellers that produce more lift than drag while consuming less power. However, recent works combine aerodynamics with control theory to better understand the stability and performance of the quadrotor system whilst operating under aerodynamic effects. Modeling these aerodynamic effects will not just be useful for accurately designing control systems for drones, moreover, it will give insights into other factors such as velocity beyond which such effects become existent or the geometry of the drone that is most vulnerable, limits on flight time and payload etc. An understanding of such effects can help researchers, and developers better design drones tailored to their applications. For instance, if its known that beyond a certain velocity the drone's attitude tracking becomes adversely affected due to wind disturbances, the drone can be configured in such a way that it autonomously selects an algorithm mid-flight that continues to ensure robust operation against these effects. Same is true for other case, if the operating conditions are in favor of drone, wherein the propellers need not depend on motors to supply power for maintaining same magnitude of thrust, then algorithms can be developed accordingly and flight time could be improved.

1.2 Organization of This Thesis

This thesis is divided into two halves. The first half consists of three chapters. Chapter 2 gives a brief overview of the various aerodynamic effects of quadrotors during vertical and translational flights. It is then followed by a detailed survey of existing literature that address aerodynamic effects modeling and compensation. In Chapter 3, an overview of the wind tunnel facility used for testing purposes was provided followed by details on parasitic (bare-airframe) drag and propeller drag

experiments. The second half deals with dynamics, control, and experiments. In Chapter 4 system dynamics incorporating parasitic the drag model is presented. Different types of trajectories that were implemented are discussed along with relevant mathematical equations wherever necessary in Chapter 5. Details of the testbed along with a discussion of results for each trajectory case are presented in Chapter 6 followed by concluding remarks and future work in Chapter 7.

1.3 Contributions of This Thesis

There are three main contributions of this thesis.

- As per [15], attempts are yet to be made to quantify and model parasitic drag due to airframe. Very few attempts have been made to investigate the relationship of parasitic drag with pitch angle using a data-driven approach from wind tunnel testing. A similar model for lateral characteristics can be extended because of the symmetry of the body frame of quadrotor.
- The physics that entail propeller drag due to forward flight of quadrotor was captured and modeled. Previously [16] modeled propeller drag but did not vary motor angular velocity. In this thesis propeller drag is modeled as a function of the angular velocity of the motor, along with pitch angle and forward flight velocity of quadrotor in body frame. This model can now serve as a reference for validation purposes for the model that is obtained numerically.
- There is a clear and open documentation provided for the integration of Optitrack, ROS, and PixHawk without having to use MAVROS. The link to the GitHub repository is also made available.

1.4 Assumptions Made in This Thesis

Two major assumptions were made in this thesis.

1. The mapping of independent variables: motor angular velocity, pitch angle, and body velocity are considered to be static with both parasitic and propeller drag. This stemmed from the fact that for a given operating condition (given v_B.Ω,θ) there exists a unique value of drag corresponding to that point.
2. The drag models obtained were for longitudinal characteristics only as the experimental setup would only let us change variables in one direction. However, since the quadrotor frame is symmetric, it was assumed that the same can be applied for lateral characteristics by replacing the pitch with a roll angle.

Chapter 2

LITERATURE REVIEW

2.1 Description of Aerodynamic Effects

1. Blade flapping

   During forward flight, the advancing blade of propeller encounters the incoming wind at a higher effective velocity than that of the retreating blade causing it to generate more aerodynamic lift than retreating blade which loses lift. This results in flapping up of the advancing and flapping down motion of the retreating blade once every revolution as shown in the Figure 1. In the Figure 1 α_{1s} represents flapping angle for a flexible propeller blade wherein an additional moment at the hub of the propeller is caused due to the flexing property.

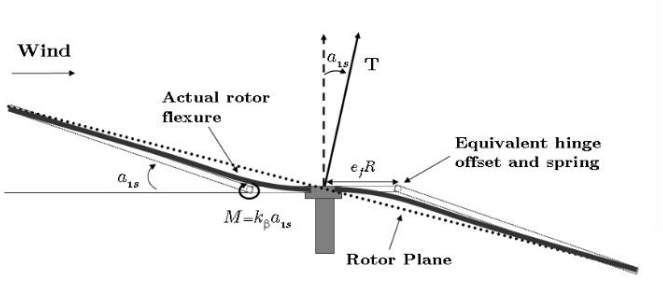

Figure 1: Flapping of Flexible Propeller Blades in Forward Flight.
Source: Adapted from [17].

Flapping occurs both in a longitudinal and lateral azimuthal location ψ with the maximum flapping angle occurring at specific azimuthal locations periodically. Due to this effect, the tip path plane of the propeller as can be seen in Figure 2 tilts both in longitudinal and lateral directions causing the thrust vector to

also tilt accordingly. As a consequence, there exists a horizontal component of the tilted thrust vector which induces a flapping drag and a moment about the center of gravity if the height of the rotors from the center of gravity is significant, which affects attitude stability.

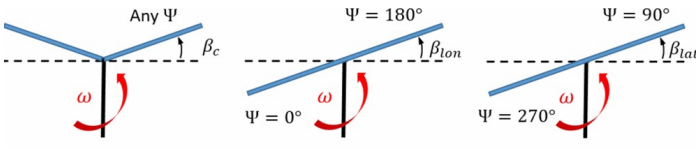

Figure 2: Coning Angle, Longitudinal and Lateral Flapping Angle of Blades. Source: Adapted from [18].

2. Total thrust variation in translational flight

   As the quadrotor pitches forward to achieve translational motion, there is a higher induced velocity due to the additional momentum from the stream of airflow entering the plane of the rotors. Increase in angle of attack also leads to an increase in thrust. Using suitable expressions, a mapping could be established along with look-up tables wherever necessary to compute the desired thrust from induced velocity for forward flight. This in turn helps in prevents high altitude tracking error.

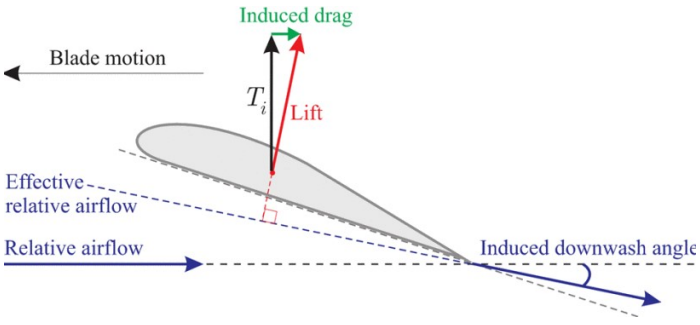

Figure 3: Induced Drag. Source: Adapted from [19].

3. Induced drag

   Induced drag is the drag induced as the lift is produced by the rotary blades as shown in the Figure 3. It exists as a by-product and increases as the lift increases. In [20] it is shown that induced drag is the maximum for small quadrotors with stiff blades during vertical-climb until hover. In the context of quadrotors, previous research incorporated a lumped parameter model of induced and flapping drag.

4. Parasitic drag

   Parasitic drag is the drag due to the frontal surface area of all the hardware that is exposed to the air other than the propeller itself. These parts of the drone are not designed to be streamlined as their practical use is to lift a payload and not to reduce drag. This form of drag is known to become dominant in quadrotors during forward flight at high speeds due to the bare-airframe along with accessories. For instance, the cross-sectional area of the battery alone might be significant contributions to the overall parasitic drag. It is worth mentioning that not a lot of work has been done to model this type of drag acting on quadrotors which also tends to influence attitude tracking performance.

5. Propeller drag

   The main lift producing component is the propellers and they contribute to a major portion of the overall drag due to their cross-sectional profile and also other factors such as surface roughness. Analytical methods such as Blade Element Theory (BET) or a hybrid Blade Element Momentum Theory (BEMT) can be employed to calculate aerodynamic forces and torques. However, they require that information related to geometry and profile of blades to be known which most of the manufacturers do not provide. Hence, a lot of simplifying

assumptions are to be made which when solved analytical might not lead to accurate estimation of the forces. Alternatively, an experimental approach can be used to also determine the aerodynamic forces on propellers and their coefficients.

6. Vortex ring state

   When the vehicle is descending with a velocity greater than twice that of velocity at hover the airflow beneath the blade randomly recirculates through the rotor disc but not completely upward as in windmill brake state. This is a very unstable state for the quadrotor to be in and usually leads to a crash. It is usually recovered by a pitch or roll command to exit the vertical state such that a translational motion is achieved.

2.2 Review of Relevant Work

There are two major problems investigated within the quadrotor literature related to aerodynamic effects. First is the accuracy of the thrust model for forward flight and compensation of first-order aerodynamic effects that tend to induce significant tracking error. In general, most work has considered the standard thrust and drag model which is directly proportional to square of the angular velocity of propeller rotation. This model doesn't seem to be accurate for flights which require quadrotors to travel with high velocity (beyond 4 m/s). In addition to requiring an accurate model for aerodynamic forces mapping specific to forward flight mode of operation of quadrotor, different forms of drag starts to become significant and the assumptions used in deriving model for axial climb or hover starts to deteriorate. Hence along with implementing a well-validated aerodynamic force model there is also a need to

compensate for drag forces and moments that become significant that in turn induce attitude instability and tracking performance.

One group [17] established a theoretical formulation of first order aerodynamic effects and initial attempt to modeling blade flapping by developing a custom test rig. Some experimental results were also presented but emphasis was more on the analytical modeling that is applicable to quadrotors with reference to helicopter literature. This rig was later used in their following study [21] which investigated two aerodynamic effects viz. blade flapping that induces attitude instability and total thrust variation affecting altitude tracking performance. Modeling of blade flapping angle was achieved by measuring the lateral forces and mapping it by established analytical expressions. Although they have also characterized blade stiffness coefficient to compensate for moments due to the rotor blade bending at the hub, there has been no information with regards to how they characterized this coefficient. Both works do serve as a good baseline for applying already established formulae from helicopter literature, however, not all the aerodynamic effects have been validated for. The velocity achieved in the stall turn maneuver performed to demonstrate compensation for flapping seems promising but since it is only one trajectory where purely longitudinal dynamics is involved, it is not conclusive enough to say the modeling approach might work for other trajectories as well.

Improvement in tracking performance doesn't necessarily need to stem from compensation alone, rather, an accurate estimate also does play a vital role. In [18] improved state estimation of rotor forces is proposed for 3 different acceleration models and validating them by applying them to an Extended Kalman Filter wherein the sensor measurements model of the filter is modified for incorporating for each acceleration model for respective validation. This study provides two useful insights.

One was to carefully assume a few parameters used for calculating rotor forces as constant as this allowed ease of computing the rotor forces which seemed very close to their model validation. Second, the importance of accurate estimation even before compensating for the aerodynamic effects is demonstrated. However, their scope was only limited to improved state estimation and other aerodynamic effects weren't modeled or compensated for.

In [20], an extensive analytical approach has been made to develop a nonlinear dynamic model that treats the forces and torques applied as a function of power output from the motors which are considered to be free inputs to the system. However, practical implementation of such a model would require an additional current sensor on-board which adds weight and more noise. Moreover, different forms of drag are modeled both for low velocity and high velocity flight but only a qualitative explanation has been provided. Conditions for mode switching for normal operating state, various rotor states in vertical flight along with ground effect has been discussed for designing a controller for switched states. Although, a simulation for a nonlinear model predictive control has been provided, it is not clear as to how much of an effect the aerodynamic effects are making in the response and models are yet to be validated by the author.

In [22] the propeller aerodynamic drag model consisting of the sum of flapping drag and induced drag for both stiff and flexible rotors is presented using analytical expressions from helicopter aerodynamics theory. Parameters and coefficients are directly taken from characterizations and assumptions made in [19]. Simulation results for two trajectories including hovering flight and for slightly aggressive trajectory with and without slowly varying wind velocity are presented. A drag compensated controller seems to perform improved position control even in the presence of wind. This study does presents a clear approach to their aerodynamic modeling but simulation results

presented are not comprehensive. Position profiles or any other statistical means aren't provided to observe the impact of augmented controller. Moreover, discussion of results was purely based on simulation and no experimental results have been presented.

There exist very few modeling efforts with experimental approach either by means of wind tunnel or system identification methods specifically, for forward flight. In [16] they have taken a wind tunnel based experimental approach for determining propeller thrust as well as drag for forward flight. However, this testing is done for the purpose of validating a analytical model (or map) that they propose which combines Blade Element Momentum Theory (BEMT) for vertical climb and Blade Element Theory (BET) for forward flight. Usually a hybrid BET and Momentum theory for a axial climb is used but in forward flight the small angle assumption doesn't hold well as flight velocities become significant and since the rotor disk is tilted with a non-zero angle facing the incoming freestream, the drag becomes non-trivial. They also demonstrate how standard thrust and drag model which is a quadratic dependence on angular velocity doesn't hold well beyond hover.

Quadrotor dynamics incorporating rotor drag and thrust augmentation was proved to be differentially flat in [23]. They identified their drag coefficients by flying the quadrotor in circular and lemniscate trajectories where they ran an optimization program which iterated for different combinations of drag coefficient values that yields the lowest root mean squared error (cost function). Later, they achieved compensation by using these coefficients of linear reference velocity terms, which were then fed-forward to the output of commanded thrust and torque. Additionally, some of the other contributions of this work include aerodynamic moment compensation in the

rate controller loop, and also validating the refined thrust model provided by [15]. Their root mean squared reduced by 50 % after compensating for rotor drag.

Some groups considered experimental modeling by performing regression and post-processing of onboard sensor data. In [15] they investigated the effect of incorporating lumped parameter models for induced drag and thrust. To determine relationship between ω_i, V_{zi}, and v_i they flew the quadcopter in upwards and downwards flight of -2 to 2 m/s logging motor speeds, adding differing weights to manipulate $T = mg/4$ and computing induced velocity from momentum theory. A linear relationship was obtained from linear regression between ω_i, V_{zi}, and v_i. They propose a refined thrust model which is built on top of the standard thrust model and blade-element momentum theory model as $T_i = k_\omega \omega_i^{2\smile} k_z V_{zi} \omega i + k_h (V_{hi})^2$ which is now purely a function of motor velocities and inertial velocities both of which can be measured using on-board sensors. For induced drag identification they use a model which was already derived in previous work and used the IMU data to record forward flight velocity and acceleration so that they can do a regression to determine induced drag coefficient k_d from the equation $\alpha_{IMU_x} = -\frac{k_d}{m} \omega_s V_x cos\theta + b_{ax}$. In [19], the effect of drag forces on both translational and rotational dynamics have been modeled separately. The force vector is made to be independent of vehicle orientation. They obtain drag coefficients via. offline batch estimation procedure of the accelerometer data. Disturbances due to other forms of drag (parasitic, flapping and induced drag) forces have been compensated using a large gain K_ω in the control design. A non-linear drag-augmented control scheme is applied, the effectiveness of which is presented in simulations and experiments.Experimental results show that the drag-augmented controller converges faster to setpoint when compared to the traditional controller when the vehicle starts to pick up speed. Although the control scheme compensates

for non-linearities due to aerodynamics while ensuring large domain of stability, the experimental results from step input alone is not sufficient to prove the effectiveness of compensation. Due to the limitations of testing space, the control scheme has been validated only for low to moderate speeds.

Some work also presents an alternative approach to addressing the problem of disturbance induced due to these aerodynamic effects. In [24] aerodynamic effects are not modeled; instead differential flatness property is exploited for feeding forward control terms from higher order reference inputs like jerk, snap along with yaw, yaw rate and yaw accelerations. Since snap is directly related to the angular accelerations, they choose to control the commanded torque. They achieve this lower-level control by closing the loop on motor speeds using measurements from optical encoder to directly control the motor speeds using a incremental nonlinear dynamic inversion (INDI) scheme which is able to bring down the root mean squared tracking error down to 4.0 cm under external disturbances. In [25], the effects of wind disturbances are modeled as unstructured, and unknown disturbances to the quadrotor. This means that there two unknown terms; one in position and the other in attitude dynamics. Their key idea is using multilayer neural network to determining the weights of the adaptive control terms and designing the controller such that the position and attitude errors are uniformly ultimately bounded. Although they present proof for their errors being uniformly ultimately bounded, there have been neither simulation nor experiments done which demonstrates the efficacy of their solution. In order to validate the effectiveness of compensation on trajectory tracking control, it is often a common practice in quadrotor research to use a motion capture system for both state feedback and recording ground truth data. However, if the capture volume is relatively small it is highly likely that quadrotors might have to exhibit smooth yet dynamically feasible

trajectories within confined space. One such work [26] provides a proof that quadrotor dynamics are differentially flat and in turn end up exploiting this property to automate the trajectory generation in terms of position in space and yaw angle. The highlight of this work in addition to generating minimum snap trajectories is imposing corridor constraints which allow for practical implementation of such moderately aggressive trajectories within confined environments. Although, [23] does extend their work by proving quadrotor dynamics subject to rotor drag is differentially flat, they haven't exploited corridor constraints to design trajectories within confined space but they have utilized simple circular and lemniscate trajectories.

To summarize, the efforts to modeling quadcopter aerodynamic effects has been progressive and more practical but the models themselves haven't been validated enough. Only few work demonstrate experimental results for one or two cases others predominantly present simulation results and due to lack of testing space aren't able to validate their model. Some modeling efforts directly adapted analytical models from helicopter literature. They further required a lot of simplifying assumptions to solve for induced velocity and also needed to determine more number of constant coefficients which highly depend on the information regarding geometry of the blade, that is often not easily available. Hence it is practical to stick to determining the models experimentally. Having said that, using on-board sensors for fitting logged flight data seems promising but it could contain sensor noise etc. Alternatively, an offboard approach can be used which might yield in a more accurate model hence a wind tunnel was chosen for this approach in this thesis. Physics pertaining to different components and phases of quadrotor motion can be captured and modeled using the wind tunnel. Since fewer works have attempted modeling of parasitic drag, a knowledge of which is very limited and compensation for this type of drag is also yet

to be done. Thus, in this thesis an offboard data-driven approach will be utilized to model parasitic and propeller drag along with compensation results provided with parasitic drag for the DJI F330 platform. Further, trajectories have been designed considering corridor constraints so that moderate to high velocities could be attained within the capture volume. Differential flatness property will be exploited to achieve trajectory tracking and feed-forward compensation as demonstrated by previous work. Relatively more experimental results will be provided to get a better insight into the tracking performance after compensating for such effects.

Chapter 3

AERODYNAMIC DRAG MODELING

In this chapter, first, the testing facility common to both experiments will be introduced. Then, tests to obtain the parasitic drag model along with reasons to use a simplified model will be discussed. Following this, the procedure to obtain propeller drag will also be detailed.

3.1 Description of Wind Tunnel Test Facility

The black wind tunnel shown in Figure 4 is a low-speed suction type open-return tunnel. On one side of the tunnel is a hexagonal shaped inlet with a honeycomb structure to streamline the incoming airflow. The inlet cone has a contraction ratio of 25:1 in order to accelerate the air to the desired velocity. The tunnel is equipped with an octagonal-shaped test section with 18.5 in. in diameter and 42 in. in length. Within the test section, the model is installed on to the force balance. The force balance present is an Aerolab 3 component Pistol Grip Balance, with $\pm 25 lbs$ normal force load limit, $\pm 10 lbs$ axial force load limit, and $\pm 50 in - lbs$ pitching moment limit. Usually a scaled-down model of an aircraft wing is placed in the tunnel, but these limits are more than enough for a quadcopter model in the tunnel. Force balance is usually oriented using a knob which lets the pitch angle change from -22 to 25 degrees. Following the test section is the diffuser section where a motor controller sits after which return circuit of the tunnel starts extending above near the ceiling until the exit which is located just above the inlet. Wind speed is adjusted by controlling

the angular velocity of a fan that sucks in air from the inlet using an ABB Variable Frequency Drive (VFD) controller. Wind velocity can be adjusted from 5m/s to 40m/s.

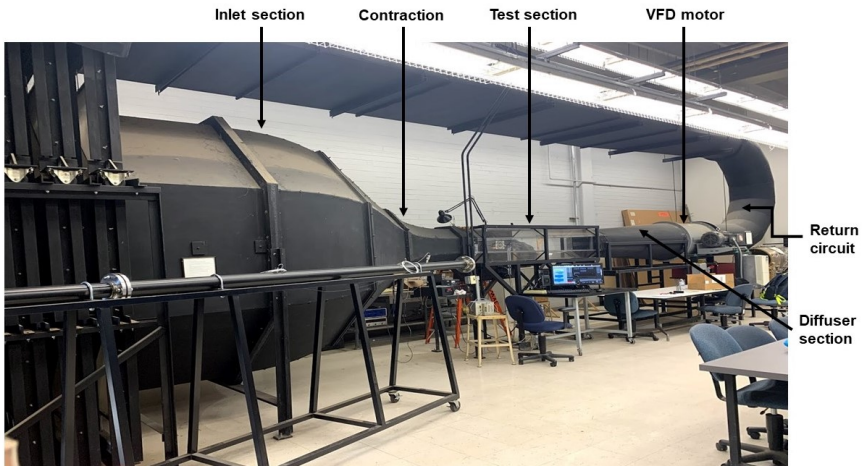

Figure 4: Low Speed Wind Tunnel Facility

3.2 Instrumentation

The wind tunnel is equipped with several instruments which measure different physical properties. It has a pitot static tube to measure incoming wind velocity, a dynamic pressure transducer to measure dynamic pressure, 2 thermocouples to measure ambient air temperature. It also has sensors to measure barometric pressure and humidity. Outside the wind tunnel there is instrumentation for measuring ambient pressure which will be input in the LabVIEW interface before starting of the experiment.

The analogue output from all the individual sensors is then combined at the multiplexer which converts them in to digital signals and streams at 330 KHz maximum.

The software is programmed to perform root mean squared averaging on the sampled raw data and pass it through a data reduction matrix.

3.3 Parasitic Drag Model

Parasitic or bare-airframe drag is the drag due to the frontal surface area which is exposed to the airflow. The larger the frontal surface area the more will be the drag. Rectangular frontal area of battery and motor without propellers installed are good source of parasitic drag at moderate to high speeds. Prior to this, related works [15],[22], were unable to obtain a model for parasitic drag analytically or experimentally. This could be because of the fact that there is no established analytical model available in literature that describes parasitic drag for quadrotors. This also means that onboard sensor data acquisition followed by offline batch-estimation for this type of drag would be difficult as there is no knowledge of relationship with other parameters available.

3.3.1 Testing and Data Acquisition

This facility allows for measuring forces a moment acting on the body due to the air flowing at a certain velocity and also change in pitch angle which become the two parameters that parasitic drag would depend on. The quadrotor within the test section closely simulates a drone in translational motion within an indoor environment. That is, quadrotor remaining stationary with air in motion around it is the same as the quadrotor moving past stationary air with certain velocity in body frame. Pitch angle change is achieved using a knob beneath the force balance.

First, the rigid body itself has a force component due to gravity when installed on

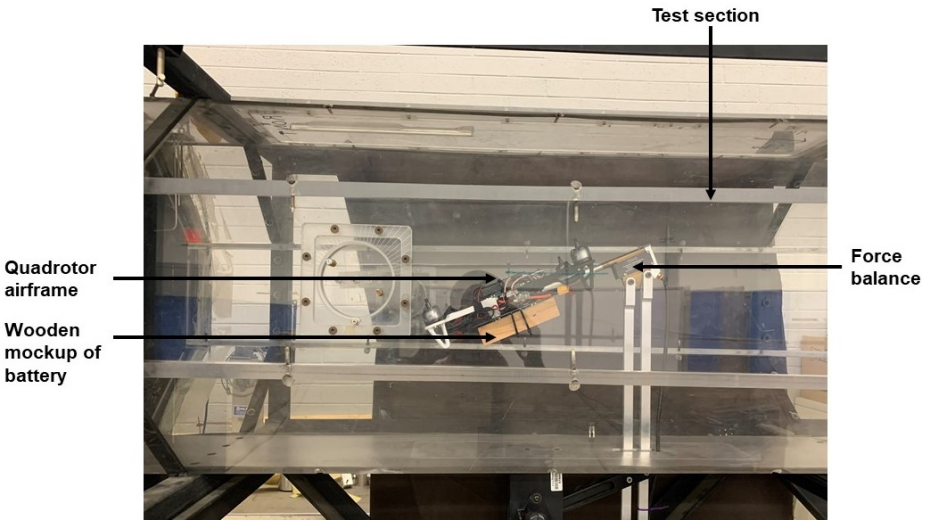

Figure 5: Quadrotor Wihout Propellers but with All Other Accessories in Wind Tunnel

the force balance. This needs to be subtracted to the forces measured with change in pitch angle and wind velocity to get the true body force and moment acting on the body. Hence with motor turned off, and taring the force balance after maintaining a zero pitch, the data was measured for full sweep of pitch angle as shown in Figure 5.

One run is considered as, for a given velocity, forces and moments are measured as pitch angle changes from -22 degrees to $+25$ degrees at every increment in pitch angle. Velocities were varied from 7m/s up until 28m/s. For quadrotors, 7 m/s itself is a very high velocity. Previously experiments have been done in trajectories where quadrotors have travelled beyond 8 m/s [21]. Tests were done beyond these velocities to get an insight into the change in the relation between parasitic drag for low and high velocity.

3.3.2 Model Fitting for Parasitic Drag

The raw body force data is post-processed to get drag force data as follows

$$D_{par} = F_{normal} \sin\theta + F_{axial} \cos\theta$$

Figure 6 represents that drag is varying as second order in velocity and linear in pitch angle for the range $-20 \leq \theta \leq 20$ degrees. For really high velocities, which usually are beyond the range of operation of quadrotor, the pitch angle within the aforementioned range varies as quadratic nature. This validates the standard expression for drag, $D \propto V^2$ for the range of velocity and pitch angle in the data collected.

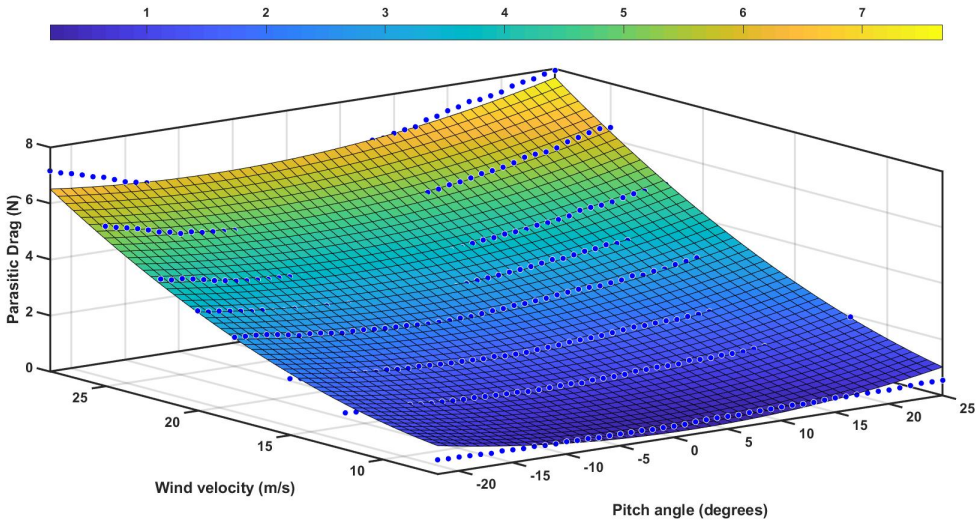

Figure 6: Surface Fitting for Obtaining Parasitic Drag Model

The maximum value of drag is $7N$ for highest velocity case, which is usually not attainable by quadrotors due to limits on the actuators, indoor space etc. The value at $7m/s$ is less than 1N.

In MATLAB a surface plot of fit (R^2 value was 0.9915) using the data and the corresponding model for parasitic drag is obtained as follows.

$$D_{par} = -0.1069 - 0.01293\theta - 0.01856v_B + 0.0015050\theta^2 + 0.001185\theta v_B + 0.008744{v_B}^2$$

where v_B is velocity in body frame and θ is the pitch angle.

3.3.3 Reasoning Behind Linear Model Approximation

There is one limitation to the estimated model of parasitic drag obtained from testing. Pitch angle cannot be used feed-forward and hence compensate for drag. This is so because pitch angle doesn't appear as one of the reference values. This is because, as it will be discussed in the later chapters, differential flatness property is exploited wherein reference values for trajectory is prescribed only by $[x, y, z, \psi]^T$ and their derivatives. Since pitch angle is not one among the differentially flat output, we cannot compensate for drag by feeding forward pitch angle dependent terms.

Due to this, the original parasitic drag equation reduces down purely linear in the velocity as

$$D_{par} = -0.1069 - 0.01856v_B + 0.008744{v_B}^2$$

Now, we make an assumption. The coefficient of the square of velocity term is almost small but it becomes as significant as the coefficient of linear velocity term only for velocity beyond 4m/s as the magnitude increases by an order. However, from the Figure 6, as the velocity decreases, the curve almost flattens out towards right. From [23],[15],[22] velocities beyond 4m/s is considered high. For the velocity range $0-5$ m/s, it is safe to assume drag is linear in velocity because corresponding coefficients 0.018 and 0.008 are not significantly different. This greatly simplifies for our implementation

as well. Hence the second order velocity term was also neglected from the previous equation. The final practical equation for parasitic drag then becomes

$$D_{par} = -0.1069 - 0.01856 v_B$$

.

3.4 Propeller Drag Model

The parasitic drag due to bare-airframe was low in magnitude for the range of operation of quadrotor. The other source of drag could be due to the propellers which were isolated from the airframe. The next plan to determine propeller drag model came into existence once the feasibility of installing an active propeller on to the force balance along with due consideration to vibration limits were determined. Two sleeve extension were designed so that vibration doesn't directly transfer to the force balance.

Unlike parasitic drag, propeller drag is another type that contributes to the overall drag but it is associated with different physics. Along with the lift induced drag of propellers, the interaction between dynamic wind and rotating propeller also contributes to drag due to propeller.

3.4.1 Experimental Setup and Data Acquisition

The setup from the previous experiment was slightly modified to incorporate some instrumentation that was used in static tests described in the next chapter.

Two sleeves were designed considering the range of motion of the pitch angle in

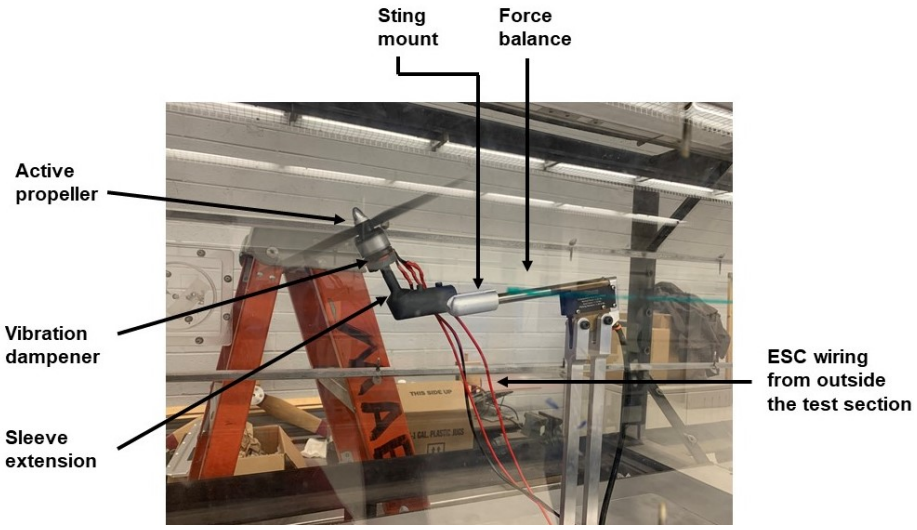

Figure 7: Active Motor and Propeller in a Wind Tunnel to Determine Propeller Drag

force balance, and the motor screw hole spacing and dimensions. These two sleeves were used to capture propeller drag data for the full range of sweep of pitch angle from 90 to 0 degrees. Propeller facing normal to the wind is 0 degrees and propeller facing into the wind is 90 degrees.

A reflective tape was attached to the motor. Angular velocity of motor was measured by standing near close proximity to test section and by pointing the laser from a laser tachometer at the rotating motor. The reflective tape will then cause a value to appear in the handheld tachometer.

The speed of propeller is controller by inputting PWM input ranging from $(0-255)$ via. a LabVIEW interface which is already programmed for open-loop motor control. Since, motor's rotation might cause any vibration, vibration dampener material was sandwiched between the motor base plate and the sleeve that was designed.

Reading for one case is considered as for a given wind velocity, body forces are measured as five sets of angular velocity and change in pitch angle from 0 to 90 degrees.

3.4.2 Model fitting for Propeller Drag

The resultant plot after post-processing is shown in Figure 8.

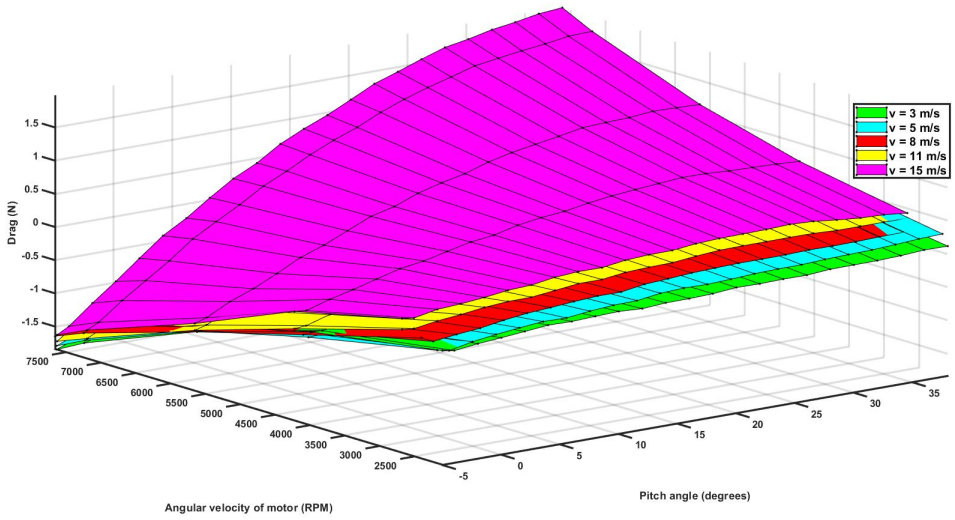

Figure 8: Surface Fitting for Obtaining Propeller Drag Model

Figure 8 is an isometric view of propeller drag as a function of motor angular velocity and pitch angle for given velocity. Data was collected for incremental velocities up to 15 m/s. From Figure 9, the relationship between propeller drag and velocity is linear with exception being for 15 m/s. In the x-axis, zero degrees of pitch angle corresponds to propeller hub facing normal to the wind and 90 degrees corresponds to propeller facing into the wind. There exists an anomaly where there is a sudden drop in value of drag and this is because the direction of the resultant force vector change as the propeller starts facing more and more into the wind. The thrust vector starts getting nullified by the incoming wind. There was no data collected in between 40-45 degrees due to the design limitation of the sleeves.

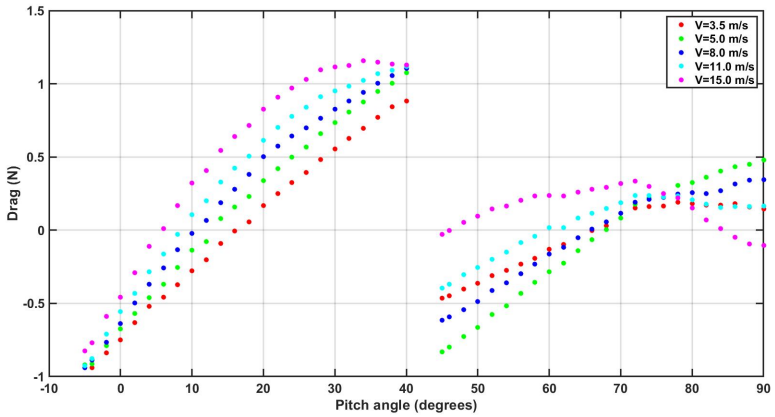

Figure 9: Relation of Propeller Drag with Velocity for a Given Motor RPM

The propeller drag model was fitted using higher order terms (R^2 =0.976) for improving the residuals and the normal distribution of the histogram of errors as follows

$$D_{prop} = 0.36891 - 0.041098\theta + 0.041432 v_B - 0.00024171\Omega + 1.8526\times10^{-5}\theta\Omega - 1.1287\times10^{-11}\theta\Omega^2$$

3.4.3 Limitation of the Model

Just like the parasitic drag model case, we cannot feed-forward motor angular velocity and pitch angle terms since the reference values are prescribed only using the differentially flat position and yaw states. It should also be noted that this model represents a different physics and it was not compensated for in the controller.

Chapter 4

QUADROTOR DYNAMICS

4.1 Coordinate Frames

There are two frames of reference which will be used to describe quadcopter motion. One is the body frame which is a right handed coordinate system with origin coinciding with the geometric center of the quadrotor. The x-axis points in the direction forward in between the two arms in the front and the z axis points against the gravity. The other is the inertial frame of reference fixed to the ground. This is also a right handed coordinate system with z-axis in the up direction against gravity.

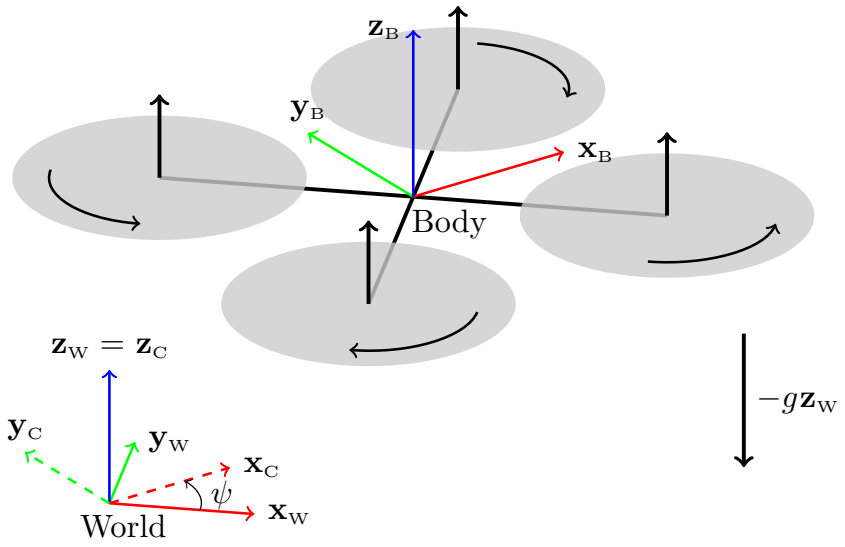

Figure 10: Free Body Diagram Representing Forces Acting on Different Coordinate Systems. Source: Adapted from [23].

4.2 Kinematics

For properly representing the kinematics in appropriate frames, it is first important to establish how we can get from one frame to another.

The body frame and inertial frame can be related by a rotation matrix R which represents a transition from the body frame to the inertial frame as given below. This matrix is derived by applying successive transformations viz.

$$R(\phi, \theta, \psi) = R(\psi)R(\theta)R(\phi)$$

by following the ZYX Euler angle conventions for undoing the yaw, pitch, and roll.

$$R = \begin{bmatrix} c_\psi c_\theta & s_\theta c_\psi s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ c_\theta s_\psi & s_\psi s_\phi s_\theta + c_\phi c_\psi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & s_\phi c_\theta & c_\theta c_\phi \end{bmatrix}$$

The position and velocity of the quadcopter in the inertial frame is given as $x = (x, y, z)^T$ and $\dot{x} = (\dot{x}, \dot{y}, \dot{z})^T$, respectively. If we have velocities defined in body frame and wish to represent in inertial frame then, for a given velocity vector v_B in the body frame, the corresponding vector in the inertial frame is given by Rv_B.

The roll, pitch, and yaw (Euler) angles in the body frame as $(\phi, \theta, \psi)^T$, with corresponding Euler rates as $(\dot{\phi}, \dot{\theta}, \dot{\psi})^T$. However, one should not be confused between Euler rates which corresponds to time derivative of Euler angles and, body rates which is angular velocity vector ω or $(p, q, r)^T$ which points along the direction of rotation.

After applying transformation, the relation between angular velocity and Euler

rates can be written as

$$\omega = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

where ω is the angular velocity vector in the body frame. However, we often require the reverse, that is, we require euler rates in terms of the angular velocity vector.

4.3 Thrust

To write down the dynamic equations of motion, it is first necessary to describe the external forces and torques acting on the quadrotor. Thrust is created by the propellers which in turn receive power from the motors. There are several analytical models that exist for determining thrust.

There is well known standard equation that thrust is directly proportional to the square of the angular velocity of motor.

$$T \propto \Omega^2 = T = k_T \Omega^2$$

where k_T is the thrust coefficient that depends on air density, and geometric properties of the blade. As mentioned before this model holds well for near hover condition and is considered for this thesis.

There are other such models as well which are suitable for hover condition. One of them is derived from Momentum theory as follows. For a quadrotor that is stationary about hover with no unsteady airflow nearby, the airflow through the rotor disc is only due to momentum imparted to the air by propellers. As per momentum theory, the velocity at free stream is zero and there is a velocity induced by the propeller

beneath the disc at hover which is given by

$$v_h = \sqrt{\frac{T}{2\rho A}}$$

where ρ is the density of the air and A is the area swept out by the rotor blades. The drawback of this theory is that it only depends on the geometry of the blade but does not consider other parameters representing operating conditions etc. However, momentum theory also holds well for hover condition. It is worth mentioning again that Blade Element Theory and hybrid Blade Element Momentum Theories are used to analyze the propeller aerodynamics accurately which is not the focus of this section.

The total thrust acting in the body frame due to all four motors is given by

$$T = \sum_{i=1}^{4} T_i = k_T \begin{bmatrix} 0 \\ 0 \\ \sum {\Omega_i}^2 \end{bmatrix}.$$

.

4.4 Torques

When a propeller rotates about its axis, a drag "force" is generated due to the friction developed due to the airflow of the rotating blade geometry. From aerodynamics it can be expressed as

$$F_d = \frac{1}{2}\rho C_d A v^2.$$

where ρ is air density, A is the reference area in exposed to the airflow, and C_d is a dimensionless constant called drag coefficient and it is unique for every blade's cross sectional profile. Velocity here is the angular velocity at the tip of the propeller.

The torque due to drag is given by

$$\tau_d = \frac{1}{2}R\rho C_D A(v^2) = \frac{1}{2}R\rho C_D A(\Omega R^2) = k_\tau \Omega^2$$

where Ω is the angular velocity of the propeller, R is the radius of the propeller, and k_{tau} is torque coefficient. This torque is needed to overcome the drag and continue to generate thrust is again proportional to square of the angular velocity of motor.

The torque about the z axis for the ith motor can then be written as

$$\tau_z = (-1)^{i+1} k_\tau {\Omega_i}^2.$$

where the $(-1)^{i+1}$ term is positive for the ith propeller if the propeller is spinning clockwise and negative if it is spinning counterclockwise. The total torque about the z axis is given as

$$\tau_\psi = k_\tau \left({\Omega_1}^2 - {\Omega_2}^2 + {\Omega_3}^2 - {\Omega_4}^2 \right)$$

If we assume $i = 1$ and $i = 3$ are the motors that create roll motion, then roll torque can be expressed as

$$\tau_\phi = L(k_T {\Omega_1}^2 - k_T {\Omega_3}^2) = L k_T ({\Omega_1}^2 - {\Omega_3}^2)$$

Similarly, the pitch torque can be expressed as

$$\tau_\theta = L k_T ({\Omega_2}^2 - {\Omega_4}^2)$$

where L is the distance from the propeller hub to the geometric center of the quadcopter. The torque acting on all the axes in the body frame can be given as

$$\tau_B = \begin{bmatrix} L k_T ({\Omega_1}^2 - {\Omega_3}^2) \\ L k_T ({\Omega_2}^2 - {\Omega_4}^2) \\ k_\tau \left({\Omega_1}^2 - {\Omega_2}^2 + {\Omega_3}^2 - {\Omega_4}^2 \right) \end{bmatrix}$$

4.5 Equations of Motion

The acceleration in the inertial frame are due to the thrust acting on the body, drag acting due to translational motion of quadrotor and gravity. In this thesis, we

specifically consider only parasitic drag which is a subset of the overall drag. Other types of drag exists like drag due to propellers as modeled before but has not been considered in our case.

After transforming thrust into the inertial frame using the rotation matrix R derived ealier, we can write the translational equation of motion as follows

$$m\ddot{x} = -mge_3 + Rf - R_{ref}D_{par}$$

where x is the position of the quadcopter, g is the acceleration due to gravity acting along the direction of unit vector e_3, D_{par} is a vector containing parasitic drag force, R_{ref} is rotation matrix constructed from reference yaw angle and f is the thrust vector in the body frame.

Unlike translational motion, the rotational equation of motion are with respect to the body frame. From Euler's rigid body rotational equation of motion. In vector form, Euler's equations are written as

$$I\dot{\omega} + \omega \times (I\omega) = \tau$$

where ω is the angular velocity vector, I is the inertia matrix, and τ is the external torques vector acting on the quadrotor. Rewriting the above equation

$$\dot{\omega} = \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z = I^{-1}\left(\tau - \omega \times (I\omega)\right). \end{bmatrix}$$

Due to quadrotor's symmetry we can write the inertia matrix as

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz}. \end{bmatrix}$$

Therefore, the final Euler rotational equations of motion in body frame is given as

$$\dot{\omega} = \begin{bmatrix} \tau_\phi {I_{xx}}^{-1} \\ \tau_\theta {I_{yy}}^{-1} \\ \tau_\psi {I_{zz}}^{-1} \end{bmatrix} - \begin{bmatrix} \frac{I_{yy}-I_{zz}}{I_{xx}}\omega_y\omega_z \\ \frac{I_{zz}-I_{xx}}{I_{yy}}\omega_x\omega_z \\ \frac{I_{xx}-I_{yy}}{I_{zz}}\omega_x\omega_y \end{bmatrix}$$

4.6 Characterization of Thrust and Torque Coefficients

There are two main reasons for conducting static bench tests for 8045 propeller that comes with the DJI F330 quadrotor.

1. To determine the torque and thrust relation with angular velocity of motor and to identify the respective coefficients.
2. To observe and address any issues related to vibration, or integration itself that will be encountered so that it can be resolved for before carrying out active propeller testing in wind tunnel using same experimental setup.

4.6.1 Description of Test Stand

Static testing was done using the static test rig show in Figure 11 that comprised of a PVC pipe oriented axially on to which a male adapter has been installed. This PVC pipe in turn is rigidly connected to a vertical aluminum beam by 4 screws. Strain gauges are taped down on select portion of the vertical beam for thrust measurement and on the PVC pipe near the portion where torque has to be measured. Calibration was done prior to actual tests to get the calibration constants. Since this test rig

is usually used for characterizing relatively larger spark ignition or glow plug type engines for RC aircrafts, the female PVC type fitting which allows the motor to be mounted was only suited for such engines. Hence a new mount had to be designed which would incorporate screw hole dimensions and spacing specific to the brushless DJI 2212/912kv motor that comes with the package along with correct internal thread type for the female mount. Thus the new mount serves those two purposes.

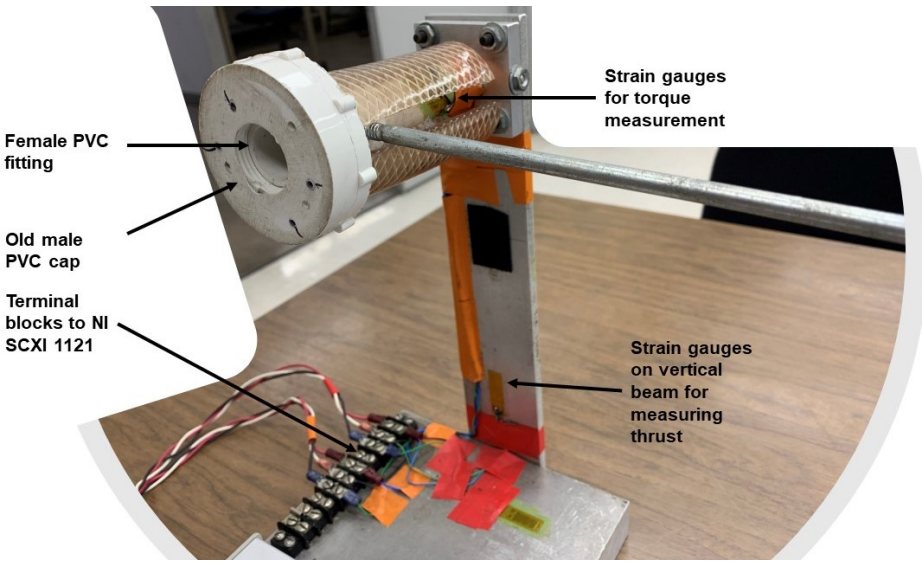

Figure 11: Static Test Stand

4.6.2 Design of New Motor Mount

The basis of the design for motor mount was inspired from existing PVC type fittings viz. female reducer bushing and cap type fitting. The threading on the existing mount implemented Iron Pipe Straight (IPS) standard $2in \times 3/4in$, $2in$ being the size of the pipe. The internal threading followed the National Pipe Taper (NPT) standard $3/4 \times 14$. $3/4$ being the internal diameter for the designed part as shown in Figure,

14 is number of revolutions for given pitch. The taper assures for a watertight fit beyond 40% engagement between the male adapter on the test rig and this motor mount design. 2 among the 4 holes are located a distance $9.525mm$ from center and the other two $8.025mm$ from center. Tolerance of $0.5mm$ or $0.019in$ was applied after creating holes in the part design. The thickness of the cap was given $0.1in$. Additional thickness of about $0.5in$ was provided by the vibration dampener giving a total of $0.165in$ so that the screw doesn't go deeper than stator portion of motor while screwing down to the mount. If the screw has been secured in a way that some portion of the depth of screw is interacting with the rotor part of electric motor, it might damage the motor and hence this was taken into account accordingly.

4.6.3 Working of the Setup

There are two parts to the functioning of the static test rig. There are two LabVIEW modules that were run simultaneously on the data acquisition computer. One was the open loop motor control via a mini ssc ii serial servo controller board through the ESC and the other was for recording angular velocity of motor in revolutions per minute (RPM) from photo transistor, induced force and torque in micro strain from strain gauge reading. Honeywell (HOA0149-001) infrared reflective sensor was used for the angular velocity measurement.

In the motor control module, the mini ssc ii serial servo controller was interfaced with the data acquisition computer through a serial communication. The communication port settings were baud rate (9600), with 8 bits (1 stop bit) with no parity as default. The mini servo controller itself needs 3 bytes of data among which the first byte of data is a sync marker, the second is servo channel to which the connection

has to be established, and the final byte of data is the a number which corresponds to the propeller setting (0-250). For safety purposes, the speed setting by default is set to zero when the program is stopped, restarted or run for the first time. The motor control algorithm is straightforward. It compares the current value of propeller setting to the previous propeller setting while the loop is running, which means, as long as the red stop button is been pressed by the user.

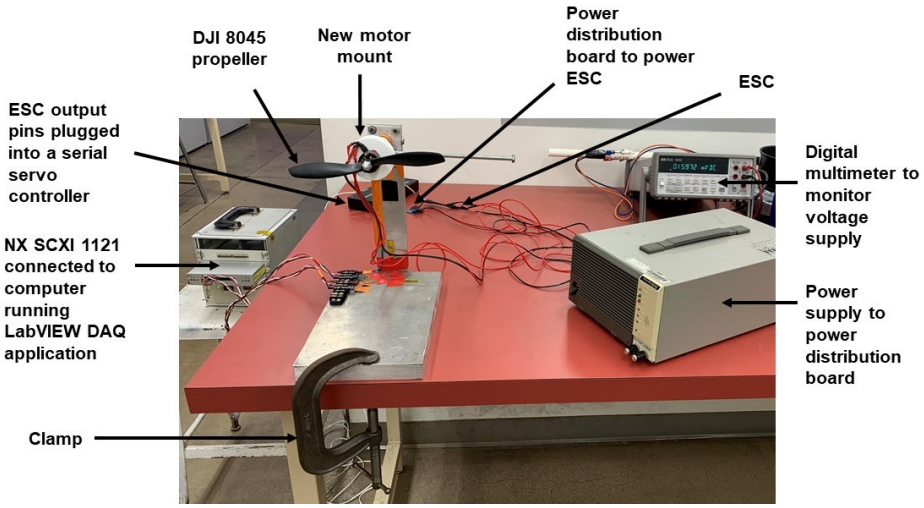

Figure 12: Front View of Static Test Facility

The propeller thrust acts in the direction away from the test rig when the motor is wired such that it is spinning in the correct direction. Care was taken to ensure that correct propeller type was installed on to motor that had the suitable power rating to drive the propeller. This force in turn pulls and creates a vertical deformation in the beam. Strain gauges which are taped on to the beam converts this deformation into strain and this strain is in turn mapped into thrust using a thrust calibration constant obtained after calibrating the instrument. The torque is also measured in similar manner where the torque induced in micro strain is multiplied by the torque calibration constant.

4.6.4 Thrust Calibration and Measurement

In order to calibrate for thrust measurements, the test stand was inverted upside down and the base of the test stand was clamped to the leg of the table as shown in the Figure 13. This ensured safety and correct load distribution when compared to holding the inverted setup by hand which was prone to human errors.

Load was placed on the vertical aluminium beam in a manner that the centre of the load placed coincides with the centre of the PVC pipe fitting. Load varying from $100g$ until $1200g$ was placed so that there is an axial deformation induced in the beam. This deformation is measured in the unit of microstrain. The Figure indicating the linear variation of deformation in microstrain for increasing load is show in the Figure 14.

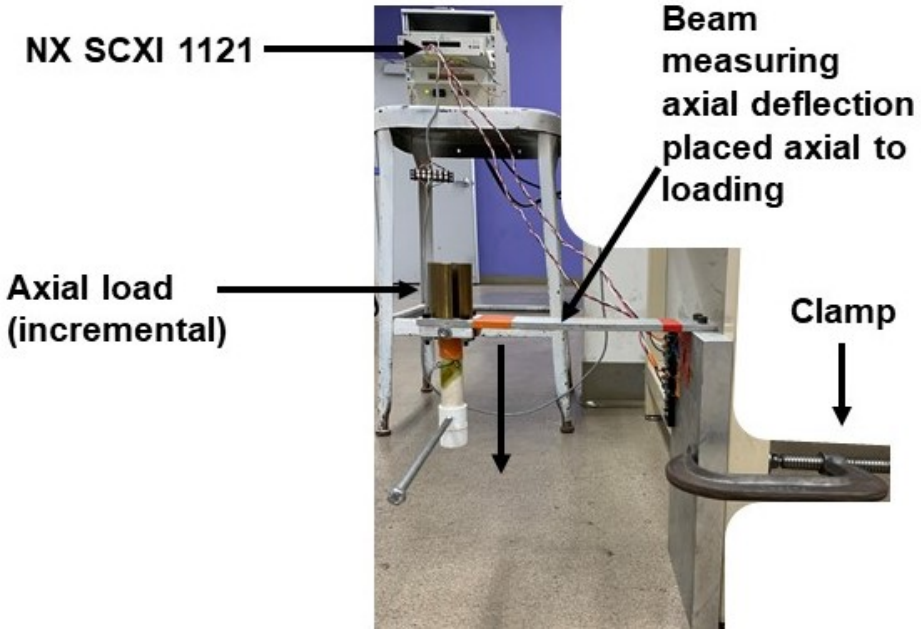

Figure 13: Placing Incremental Load Over the Axial Direction to Calibrate for Thrust Measurement

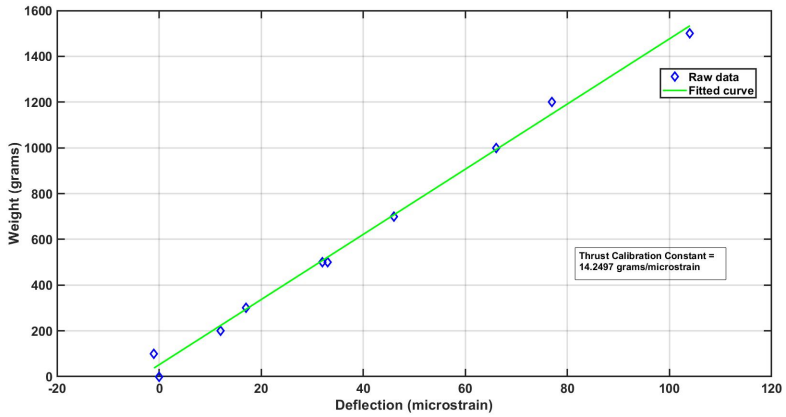

Figure 14: Thrust Calibration Curve: Variation of Deformation with Incremental Loading

The thrust calibration constant was multiplied with every deflection reading in order to obtain thrust in N. The thrust as a function of angular velocity of motors up to hovering motor RPM is given in Figure 15. The raw data was fit using custom polynomial equation $y = ax^2$ in the MATLAB's cftool. It should be noted that fitting a quadratic polynomial would although result in a better fit, it is not the correct fit for our purpose. From initial trials, fewer data points, resulted in a poor fit but from the Figure it is clear that a really good fit was obtained for identifying thrust coefficient. The thrust coefficient was identified to be 7.456×10^{-8}

4.6.5 Torque Calibration and Measurement

In order to calibrate for torque, a constant weight of 100 grams was placed at incremental position of 1 inch from the root to the tip of the horizontal beam fixed to the test setup as shown in the Figure 16.

The induced strain was measured for each position for the given loading. The

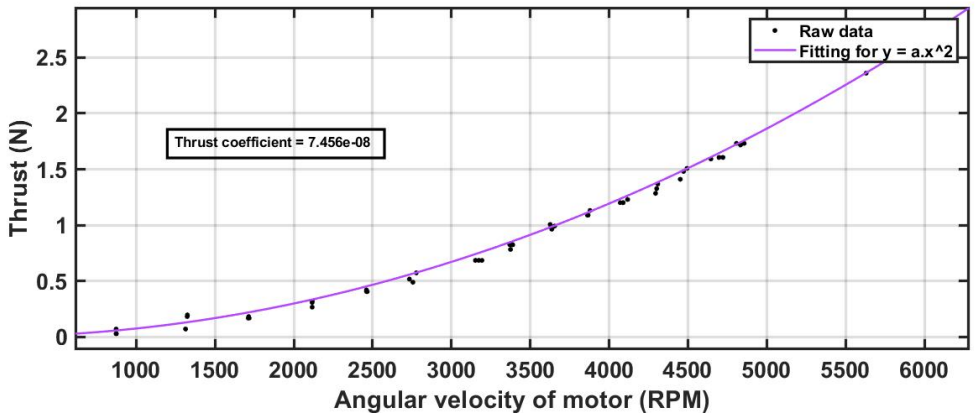

Figure 15: Thrust Profile with identified Thrust Coefficient

applied torque was plotted against the strain induced (in microstrain). The slope of the linear curve that best fits the raw data then yields the torque calibration constant as shown in Figure 17.

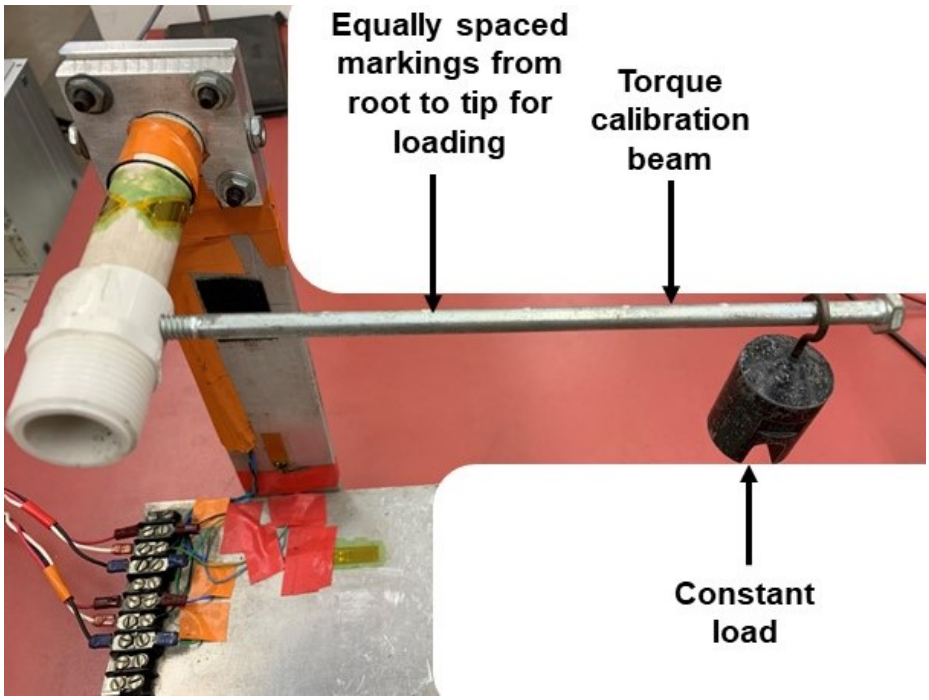

Figure 16: Torque Calibration by Incrementally Placing a Constant load in radial direction

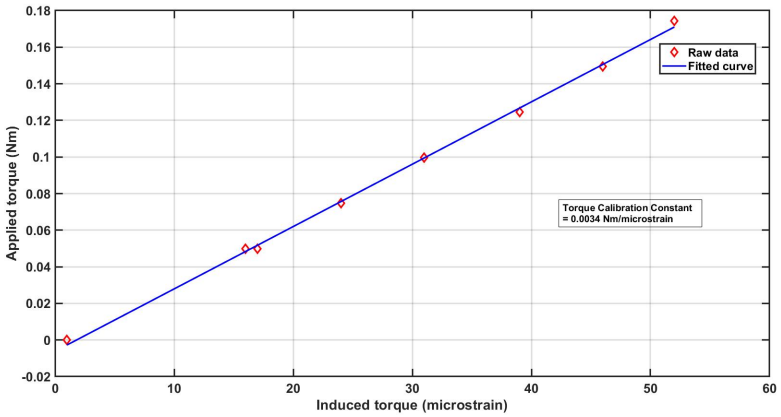

Figure 17: Torque Calibration Curve: Variation of Induced Torque with Applied Torque

This torque calibration constant was then multiplied along with every value of strain induced for increasing commanded angular velocity of motor and the product would correspond to the true torque generated due to the rotation of propellers. From the Figure 18, although there exists minor drift in measurements, custom equation $y = ax^2$ was fit with raw data and it seems fairly reasonable. The drift existed due to loose terminal connectors which needed to be screwed down. The torque coefficient obtained from this test was around 2.278×10^{-9}

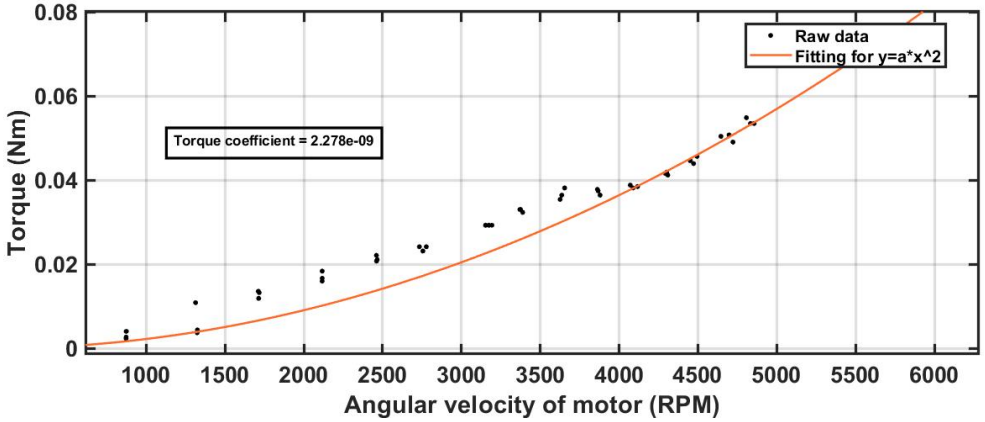

Figure 18: Torque Profile and Identified Torque Coefficient

4.7 Feed-forward Compensation

The overall system architecture along with trajectory planning, feed-forward compensation, and control is shown below.

From the reference position,velocity and acceleration (if provided) and using the modeled drag from offboard data-driven approach shown in previous chapter, a feed-forward force or moment can be added to the commanded force or moment. But in this thesis assuming that height of propellers from the centre of gravity is small, compensation for moment is not performed.

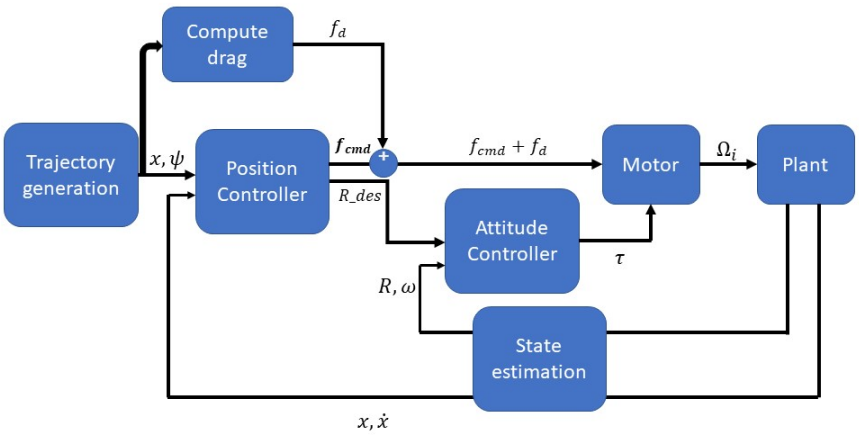

Figure 19: Feed-forward Drag Force Added to Commanded force From Position Controller

4.7.1 Computing Rotation Matrix from Reference Values

First, rotation matrix will be obtained considering the quadrotor body frame follows standard right handed coordinate system with z axis as up. However, the orthonormal basis will then be swapped as PixHawk controller's North-East-Down frame of reference. It will be mentioned in the upcoming chapter that using differential flatness property, we define a reference trajectory by position and yaw angle in 3 dimensional space as $\sigma = [x, y, z, \psi]^T$.

We know the body z-axis of the quadrotor is in the same direction as thrust vector then we can write z_B as

$$z_B = \frac{t}{||t||}, t = [\ddot{\sigma_1}, \ddot{\sigma_2}, \ddot{\sigma_3} + g]^T$$

This defines the body frame z-axis of quadrotor [26]. From the given yaw angle, we can write the unit vector that represents the rotation of inertial frame by the reference yaw angle.

$$X_C = [cos\sigma_4, sin\sigma_4, 0]^T$$

Now, x_B and y_B can be determined as follows.

$$y_B = \frac{z_B \times x_C}{\|z_B \times x_C\|}$$

$$x_B = y_B \times z_B$$

provided, singularity when z_B is parallel to x_C is not encountered. One fix for the singularity point of operation is given by [26]. It is observed that $-x_B$ & $-y_B$ are also consistent with yaw angle and body frame z-axis. In practice, check which one of the solutions is closer to actual orientation of quadrotor in order to calculate reference orientation matrix.The orthonormal basis are swapped as per NED convention.

Recalling from Chapter 3, simplified form a parasitic drag model excluding pitch angle and linear in velocity is $D_{par} = -0.01856 v_B - 0.1069$, where v_B is the reference velocity in the body frame of quadrotor. The computed drag force is in body frame and needs to be transformed into the world frame as $R_{ref} D_{par}$ using the rotation matrix obtained from above.

Chapter 5

TRAJECTORY GENERATION

In this chapter the math and theory behind trajectory generation is presented for all trajectories. MATLAB was used to design these trajectories. The position, wherever possible velocity and acceleration setpoints that were generated from these trajectories were then exported into a csv file which was read real-time and fed to the controller once the operator switches to offboard flight mode. It is to be noted that the desired setpoints were then fed as mentioned upon a trigger and not manually via a radio control by the operator.

5.1 Circular Trajectory

From one of the previous works [23], it can be noted that circular type trajectory can be a good validation platform for assessing x-y augmented performance. A combination of curved, longitudinal and lateral phases within the trajectory can assess its behavior to both longitudinal and lateral axis accelerations.

In order to obtain (x, y) coordinates for the circular trajectory, the cartesian coordinates representation of the circle was used. A simple program was written which loops over ϕ from 0 to 2π for a given radius with 120 sample increments and the coordinates were found using the equation of circle in cartesian form given below.

$$x = r\cos\phi \quad , y = r\sin\phi$$

.

5.2 Straight Line

Straight line trajectory was generated by sampling up to 120 waypoints in between the start and goal point. This was easily done using the linspace command in MATLAB to sample waypoints for x-axis. For y-axis, the same command but with constant start and end point was used to constraint the motion of quadrotor to avoid any lateral motion. In this case y-coordinate of 1 was used from start until end.

It is to be noted that, this trajectory was used predominantly to see any unsteady or sudden commanded motion of quadrotor with changes in the default controller code before even testing out the circular and rectangular trajectories.

5.3 Rectangular Trajectory with Corridor Constraints

Given a start point and a goal point, a smooth, dynamically feasible trajectory is to be generated with the help of constraints prescribed on the start and end states. In robotics, usually these waypoints, the output from path planning algorithm, are inputs to trajectory planning algorithm and functional $\mathcal{L}$ is a representation of work done to get from start to end state. From the Figure 20, if we assume $x(t)$ is the optimal function, it is clear that α is a scaling factor is applied for other trajectories from start $t = 0$ to end $t = T$.

The optimal trajectory can be mathematically expressed as

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \mathcal{L}^2 dt$$

where, $\mathcal{L}$ is a smoothing function which is analogous to work done. Choice of smoothing function directly depends upon the order of the system input as smoothness of a

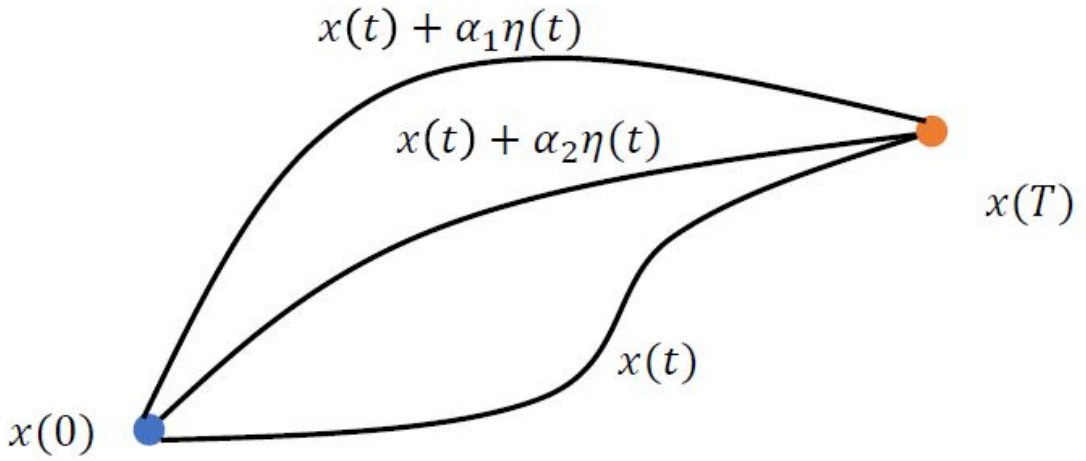

Figure 20: Work Done to Get from Start State to End State for Various Trajectories Including the Optimal Trajectory. Source: Adapted from [27].

trajectory is depicted by change in input function. Additionally, boundary conditions on $(n-1)^t h$ order derivatives can be applied to solve for $x^*(t)$.

From calculus of variations, we know that for a optimal function its derivative with respect to α should be zero.

Using Euler-Lagrange equation, the necessary condition for a function to be optimal for 1 dimensional case is

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) - \left(\frac{\partial \mathcal{L}}{\partial x}\right) = 0$$

For n^{th} order system, the functional and the optimal trajectory can be defined as follows.

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \mathcal{L}(x^n, x^{n-1}, ..., \dot{x}, x, t)dt$$

Applying Euler-Lagrange condition,

$$\frac{\partial \mathcal{L}}{\partial x} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) + \frac{d^2}{dt^2}\left(\frac{\partial \mathcal{L}}{\partial \ddot{x}}\right) - ... + (-1)^n \frac{d^n}{dt^n}\left(\frac{\partial \mathcal{L}}{\partial x^n}\right) = 0$$

Integrating and solving for $x(t)$ from the result of 5.3,

$$x(t) = c_0 + c_1(t) + ... + c_{2n-1}t^{2n-1}$$

Now solving for coefficients gives us the optimal trajectory. This is achieved by applying constraints on the boundary conditions.

5.3.1 Simple Example for 1 Dimension

To better explain the formulation from the previous section, lets assume that we have system input that is of 2^{nd} order. This example was drawn from [27].In this case, the trajectory obtained from the formulation state previously would be a minimum acceleration trajectory.

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \mathcal{L}(\ddot{x}, \dot{x}, x, t)dt$$

where, $\mathcal{L} = (\ddot{x})$ is choice of smoothing function.

Applying Euler-Lagrange's necessary condition for a function to be optimal,

$$\frac{\partial \mathcal{L}}{\partial x} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) + \frac{d^2}{dt^2}\left(\frac{\partial \mathcal{L}}{\partial \ddot{x}}\right) = 0$$

$$\frac{d^2}{dt^2}(2\ddot{x}) = 0$$

This results in a fourth order differential equation,

$$x^4 = 0$$

Integrating and solving for x(t), we obtain a cubic polynomial

$$x(t) = c_0 + c_1(t) + c_2(t)^2 + c_3(t)^3$$

Applying position and velocity constraints we have,

$$x(t=0) = a; x(t=T) = b$$

$$\dot{x}(t=0) = 0; \dot{x}(t=T) = 0$$

Writing it in matrix form,

$$\begin{bmatrix} a \\ b \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2T & 3T^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

This linear problem can be solved for in MATLAB by $x = A\backslash b$. If the A matrix is big and ill-conditioned then decomposition techniques should be used to overcome it.

5.3.2 Extension to Higher Dimensions

The same concept can be extended to higher dimensions by exploiting Euler-Lagrange equation which decouples individual axis.

$$(x^*(t), y^*(t)) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \mathcal{L}(\dot{x}, \dot{y}, x, y, t)dt$$

Applying Euler-Lagrange's necessary condition for a function to be optimal in each dimension,

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) - \frac{\partial \mathcal{L}}{\partial x} = 0$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{y}}\right) - \frac{\partial \mathcal{L}}{\partial y} = 0$$

Now, the trajectories can be solved for in x and y axes separately.

5.3.3 Multi-segment Multidimensional Trajectories

If we have intermediate waypoints also specified along with constraints on their n^{th} order derivatives then the type of trajectory generation problem now becomes multi-segment multidimensional trajectory with piece-wise polynomial basis.

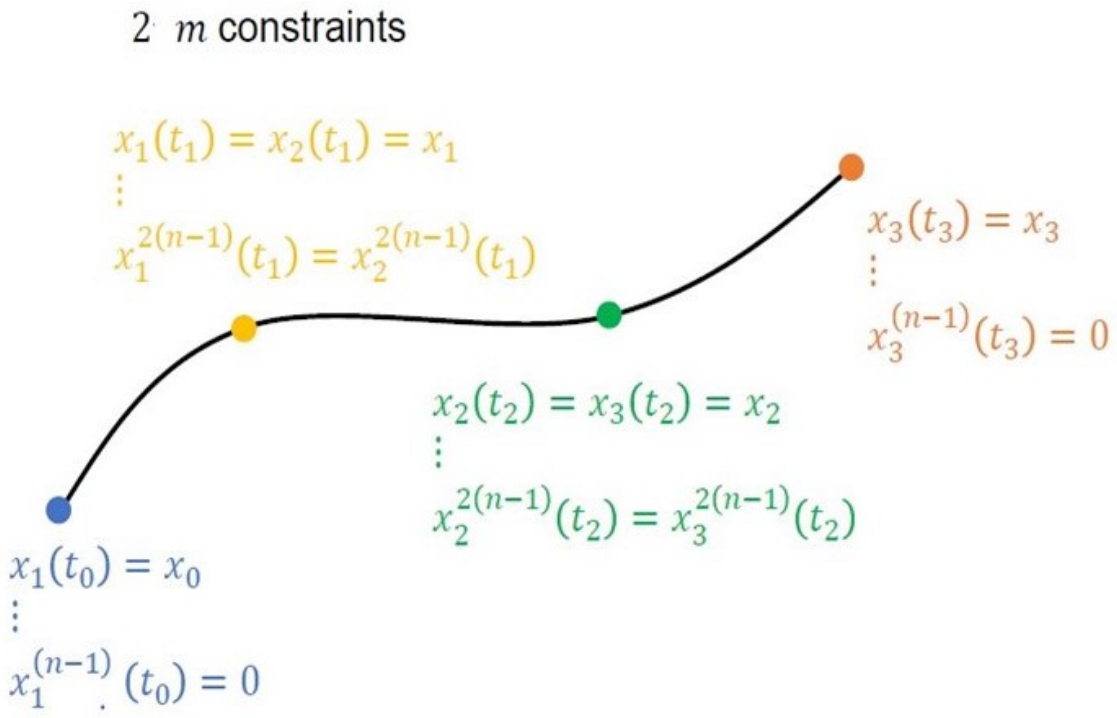

Figure 21: Multi-segment Polynomial Trajectories with Start, End and Intermediate Constraints Only. Source: Adapted from [27].

As shown in the Figure 21, there are $(m+1)$ time and 1 dimensional trajectory vectors. Unlike before, we have trajectories to be solved for within every segment by treating the state at that time instant to be starting point and next time instant to be an end point but for one trajectory. Hence, since there $(m+1)$ waypoints, there will be m segments overall.

5.3.4 Reason for Minimizing Snap

It was previously stated that, choice of the functional depends on the order of system input. In this case, we would like to minimize the square of the norm of the snap the reason for which is shown below.

$$m\ddot{x} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \sum_{i=0}^{4} F_i \end{bmatrix} + R_{ref} D_{par}$$

From Newton equation above, it is clear that the second derivative of position depends on $u1$ where $u1 = \sum_{i=0}^{4} F_i$ is the desired thrust and the rotation matrix R. The Rotation matrix depends on roll,pitch and yaw angles.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

From the equation above, angular velocity is a function of derivatives of Euler angles [28].

$$\begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} = I \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

The third equation above represents the control moment inputs $u2$, which is a function of the rate of change of angular velocity.

Hence, it is understood that :

1. From the last two equations, the derivative of angular velocity, in turn, the double derivative of Euler angles directly depicts that the second derivative of rotation matrix depends on $u2$.
2. Through R, the second derivative of position also depends on the control moment inputs. This makes sense as thrust when applied to a quadrotor needs to be prescribed with a direction, a pitch or roll angle in order for the quadrotor to pitch forward or roll accordingly.

Hence, from the above two points, we can say that the fourth derivative of position (snap) depends on $u2$. The choice of flat outputs is given by

$$\sigma = [x, y, z, \psi]^T$$

where $\sigma = [r, \psi]$, is a keyframe defined as a position and yaw angle in space [26]. The trajectory is defined a smooth curve in space of flat outputs. $\sigma(t) : [t_0, t_m] \rightarrow R^3 \times SO(2)$.

If we have a m keyframes specified at respective time instants, and if a quadrotor is supposed to pass through each keyframe remaining within a safe corridor, the simplest trajectory would be a straight line connecting the keyframes. However,for such a trajectory the quadrotor will come to rest at every keyframe especially a trajectory with sharp turns.

5.3.5 Minimum Snap Trajectory Problem Formulation

An optimal trajectory can be generated as mentioned earlier that ensures smooth transition between keyframes given some constraints.

Consider $(m + 1)$ keyframes specified at $(m + 1)$ time instants in the flat output

space.

$$t = [t_0, t_1, t_2, ..., t_m]$$

$$\sigma = [\sigma_0, \sigma_1, \sigma_2, ..., \sigma_m]$$

$$\sigma_T(t) = \begin{cases} \sum_{i=0}^{n} \sigma_{T_{i1}} t^i, & t_0 \leq t < t_1 \\ \sum_{i=0}^{n} \sigma_{T_{i2}} t^i, & t_1 \leq t < t_2 \\ \vdots & \\ \sum_{i=0}^{n} \sigma_{T_{im}} t^i, & t_{m-1} \leq t < t_m \end{cases}$$

The optimization program to minimize the integral of square of k_r^{th} derivative of position and the square of derivative of yaw angle without corridor constraints is given below [[26], eq. (10)].

$$min \int_{t_0}^{t_m} \mu_r \left\| \frac{d^{k_r} r_T}{dt^{k_r}} \right\|^2 + \mu_\psi \frac{d^{k_\psi} \psi_T}{dt^{k_\psi}}^2 dt$$

$$\text{s.t.} \quad \sigma_T(t_i) = \sigma_i, \quad i = 0, ..., m$$
$$\left. \frac{d^P x_T}{dt^P} \right|_{t=t_j} = 0, \quad j = 0, m; p = 1, ..., k_r$$
$$\left. \frac{d^P y_T}{dt^P} \right|_{t=t_j} = 0, \quad j = 0, m; p = 1, ..., k_r$$
$$\left. \frac{d^P z_T}{dt^P} \right|_{t=t_j} = 0, \quad j = 0, m; p = 1, ..., k_r$$
$$\left. \frac{d^P \psi_T}{dt^P} \right|_{t=t_j} = 0, \quad j = 0, m; p = 1, ..., k_\psi$$

We can formulate the problem as a quadratic programming problem by

$$\begin{aligned} \min \quad & c^T H c + f^T c \\ \text{s.t.} \quad & Ac \leq b \end{aligned}$$

where constants $\sigma_{Tij} = [x_{Tij}, y_{Tij}, z_{Tij}, \psi_{Tij}]$ and the vector c contains decision variables $\{x_{Tij}, y_{Tij}, z_{Tij}, \psi_{Tij}\}$. Here the objective function incorporates the minimization of functional while constraints on the flat outputs and derivatives can be applied as equality constraints and corridor constraint as inequality constraint.

5.3.6 Constraints

There are $(m+1)$ waypoints corresponding to $(m+1)$ time instants. On the $(m-1)$ intermediate waypoints there are two position constraints for a given time instant- one which corresponds to a end of one segment and the same time instant which becomes a starting point of another segment making it $2(m-1)$ constraints. This constraint is to ensure that the trajectory is consistent at the time instants.

In the start and end of the overall trajectory alone, there are only one position constraint specified which are unique for that time instant.

In total there are $2(m-1)+2 = 2m$ position constraints that are given by :

$$x_s(t=0) = p_s; x_e(t=T) = p_e; x_i(t_i) = x_{i-1}(t_{i-1}) = p_i$$

where the subscripts e and s denote start and end points respectively.

From the minimum acceleration example, we also need to impose constraints on $(n-1)^{th}$ order derivatives at the starting and goal points of the overall trajectory. Since position constraint is already prescribed, we are now left with 3 constraints for velocity, acceleration, jerk at both ends to be zero.

$$\dot{x_s}(t=0) = \ddot{x_s}(t=0); \dddot{x_s}(t=0) = 0$$

$$\dot{x_e}(t=0) = \ddot{x_e}(t=0); \dddot{x_e}(t=0) = 0$$

This adds another $3 \times 2 = 6$ constraints. Again, expanding the minimum acceleration example to n^{th} order, we need constraints to be imposed up to $2(n-1)^{th}$ order derivatives for the $(m-1)$ intermediate waypoints to ensure continuity. This means the position, velocity, acceleration etc. upto $2(n-1)^{th}$ order derivative should be the same as its adjacent trajectories.

$$\dot{x_i}(t_i) = \dot{x_{i-1}}(t_{i-1}); \ddot{x_i}(t_i) = \ddot{x_{i-1}}(t_{i-1})$$

$$\dddot{x_i}(t_i) = \dddot{x_{i-1}}(t_{i-1}); \ddddot{x_i}(t_i) = \ddddot{x_{i-1}}(t_{i-1})$$

$$x_i{}^5(t_i) = x_{i-1}{}^5(t_{i-1}); x_i{}^6(t_i) = x_{i-1}{}^6(t_{i-1})$$

That adds another $2(4-1) * (m-1) = 6(m-1)$ constraints. We know that for minimum snap, after applying Euler-Lagrange equation and solving, we will have a 7^{th} order polynomial with 8 coefficients. For piece-wise polynomial basis with m segments will required a total of $8m$ constraints to solve for the unknown coefficients.

Summing up all the constraints from above, we have $6 + 6(m-1) + 2m = 8m$ constraints. Thus, this confirms that we have enough constraints to obtain polynomials.

5.3.7 Applying Corridor Constraints

In [26], they define t_i as the unit vector along the segment from r_i to r_{i+1}. Then $d(t)$ be the perpendicular distance vector, $d_i(t)$ from segment i is defined as

$$d_i(t) = (r_T(t) - r_i) - ((r_T(t) - r_i).t_i)t_i$$

For each corridor, a width is prescribed on the inifinity norm, δ_i as

$$\|d_i(t)\|_\infty \leq \delta_i, \quad \text{while}, \quad t_i \leq t \leq t_{i+1}$$

This constraint is then incorporated into the QP as a inequality constraint by introducing η_c [26] intermediate points as

$$\left| x_W.d_i \left(t_i + \frac{j}{1+\eta_c}(t_{i+1} - t_i) \right) \right| \leq \delta_i, \quad \text{for} \quad j = 1, ..., \eta_c$$

Same applied for y_W and z_W as well.

After referring to [27] and solving for the above using quadprog function in MATLAB using [29], the following rectangular trajectory was obtained.

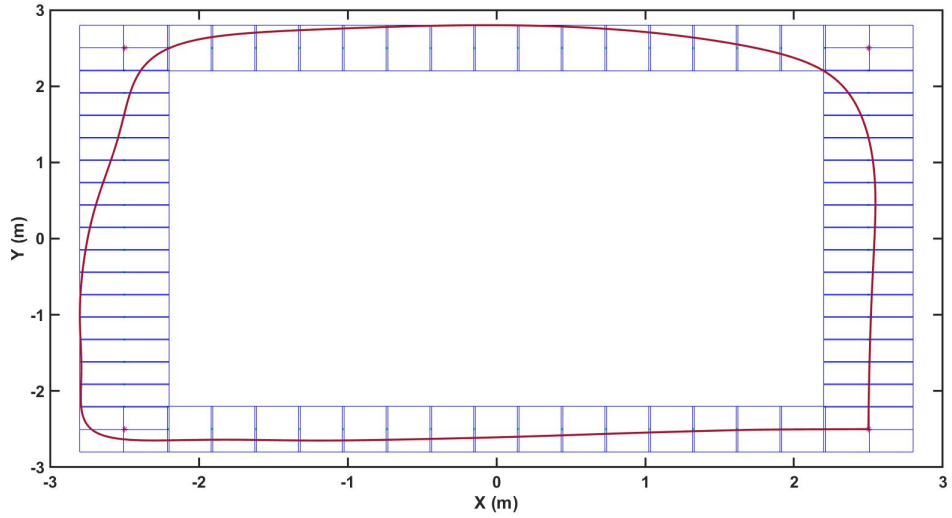

Figure 22: Rectangular Trajectory Generated Using Minimum Snap Algorithm

The blue rectangles denote that the trajectories are confined to the corridor length of $0.3m$. The maroon curve is the smooth trajectory.

Chapter 6

EXPERIMENTAL RESULTS

In this chapter, details regarding the experimental test bed including the motion capture system, its integration with the flight controller will be discussed. Following this, the way the flight tests were conducted along with results and discussions for each trajectory case will be presented.

6.1 Brief Description of Drone Studio

ASU Drone Studio is a former indoor basketball court converted into a large indoor motion capture studio spanning 10,000 sq. ft. area and 23 ft. high ceiling that was set up around Spring 2019 for enriching robotics research experience at Arizona State University. The motion capture system used here is Optitrack provided by NaturalPoint, Inc. This facility is a modular space, divided into 6 equally spaced zones, to aid 6 experiments to be conducted simultaneously. The studio is equipped with a total of 105 cameras, each zone having 17 Prime 17W cameras installed on 6m tall pillars which separate adjacent zones. These cameras can capture 360 frames per second enabling capturing high speed dynamics motion. Each zone's spacing, and the ease of ability to connect two adjacent zones or three continuous zones by porting all the camera data to a common up link switch, allows the user to expand capture volume for the case of multiple rigid body tracking applications such as robotics swarm. Up to 150 rigid bodies can be tracked simultaneously with a very high precision of

< 0.5 mm. Tracking with both passive and active markers is possible. Previously all 6 zones were used to capture and test aerial robot-ground robot motion collaboration.

6.2 Setup of Testbed

6.2.1 Rigid Body Creation and Data Streaming

To begin with, the setup of test bed as shown in Figure 23 will be detailed.The DJI F330 quadcopter was installed with six $12.7mm$ passive markers which were installed in an asymmetric and non-planar fashion around the centre of mass of the drone. The rigid body was created after aligning the forward direction of the drone to that of x-axis indicated in the motive software. If this was not done correctly there will be a yaw-offset present. Optitrack's inertial reference frame was not the same as the PixHawk's NED frame and hence a mapping was applied where in x and z of the Optitrack's inertial frame were mapped on to PixHawk's x and y frame, while the -y of inertial frame was mapped on to the z of PixHawk body frame. Now, the streamed pose is as per NED frame of reference which is what the low-level controller uses as reference. The mapping described previously was considering the data streaming is done with y-axis up in the Motive software although the onine guide tells otherwise. This was done because a custom bridge was implemented instead of MAVROS. Attitude transformation was done in a similar fashion as described on website [30] but again for y-axis being up instead of z.

Drone was operated in the zone B which had dimensions of $8m \times 8m \times 6m$ excluding the pillars which let the cameras to be mounted at height. Almost all cameras were in same plane except for the corner cameras which have field of view common to

the current and adjacent zone. These cameras are set up in the Motive software to capture data 120 frames per second. The camera system is connected via. Ethernet cables not exceeding length of 100m to 2 Power over Ethernet (PoE) network switches which are then are connected to a common up link switch that aggregates data from PoE network switches. The uplink switch is the final station station which is then connected to the zone's respective desktop computer running Motive via Ethernet cable distinguished with a colored tag for identification purpose.

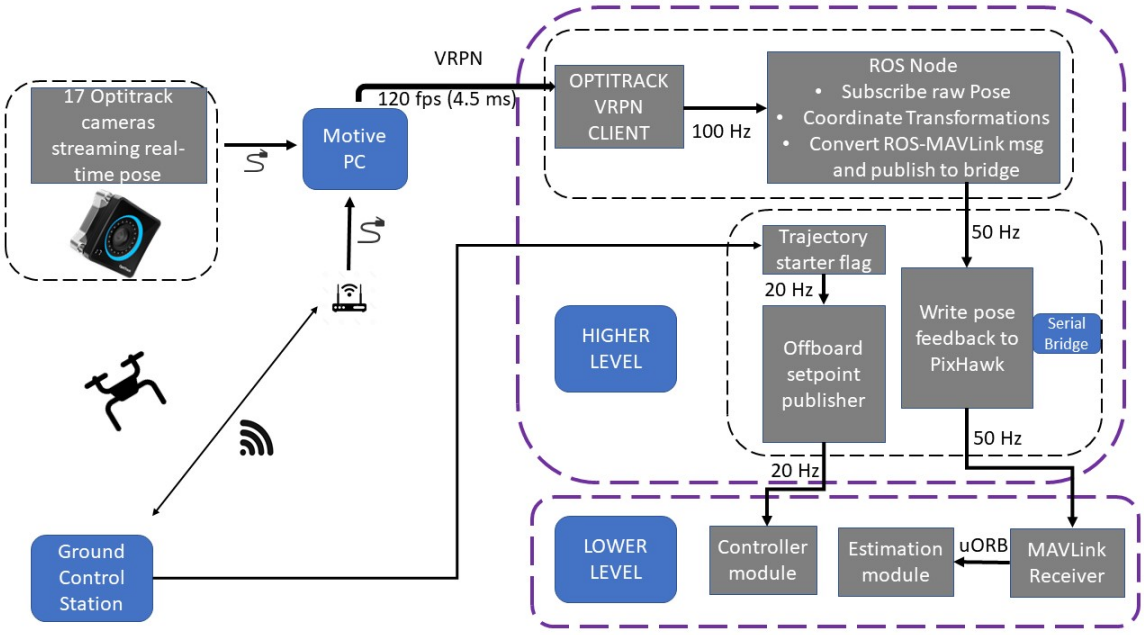

Figure 23: Full Setup of Components Involved and Workflow in Experimental System

6.2.2 Higher-level (companion) Computer Operations

Intel UP board [31] was selected as the companion computer due to the computing resource and low power consumption along with many USB ports allowing the user to

plug in multiple devices at once. In the higher level, three main programs were run real-time. They included :

1. A ROS based VRPN client that was on the wireless network as Motive workstation, to publish real-time raw pose data on the topic /vrpn_client_node/rigid_body_name/pose.
2. A custom ROS node that subscribed to the raw pose data and applied appropriate coordinate transformations as mentioned previously which are suitable with the PixHawk flight controller. This node then implemented ZMQ messaging libraries [32] to publish the pose that was fed back to PixHawk at 50 Hz. This is a very important step as ROS messages were then converted into ZMQ messages that were easily packed as per MAVLink protocol so that communication to PixHawk was established.
3. A custom bridge between companion computer and PixHawk was an essential C++ program that performed important functions. The original version of the bridge program [33] was already equipped with modules for doing socket type operations such as open, read from, write to a port, and close port along with multi-threading capabilities implemented using boost libraries [34]. There have been recent commits made to the repository after enabling support for communication with the ground station. In this thesis, the original bridge program along with ROS node with appropriate coordinate transformations has been made available [35] so that developers using Optitrack, PX4, and ROS can clone this repository on to their companion computer and instantly get started with offboard flight without using MAVROS. One benefit of using this custom bridge over MAVROS is the ease of adding more functionality while having complete knowledge of what messages are being sent to and received from

PixHawk. There was a module named bridge_subscribe() that subscribed to the pose published by the custom ROS node via ZMQ message libraries, packed them as an ATT_POS_MOCAP (current pose) MAVLink message and wrote them to the lower-level (PixHawk) at the configured baud rate using a WriteToPixhawk() function. It also read a heartbeat message from PixHawk, which also was an indicator that PixHawk was active and functioning. The other function of the bridge was to read a csv file that contained trajectory setpoints which were generated offline using a minimum snap trajectory generation algorithm or by simpler means, pack the (reference position, velocity, acceleration) into SET_POSITION_TARGET_LOCAL_NED MAVLink message and write this message at $20Hz$ to the lower-level when a flag is triggered by the user from the ground control station that supported Ubuntu OS over SSH connection. Trajectory starter flag, writing current pose, and writing reference values were executed as independent subthreads within the bridge.

6.2.3 Configuring the Lower-level for Offboard Flight

Low-level control of the drone (platform) shown in Figure 24 is achieved by using the widely adapted flight control hardware standard in the drone industry called the PixHawk flight controller. PixHawk stands for pixel-hawk which means "computer vision". This was developed by Lorenz Meier who back in 2008 was a master's student aspired to enable vision-based navigation using onboard computing for drones. Given that MEMS-based inertial sensors such as accelerometer, gyroscope, and magnetometer existed at that time, PixHawk came into existence as a byproduct during the pursuit

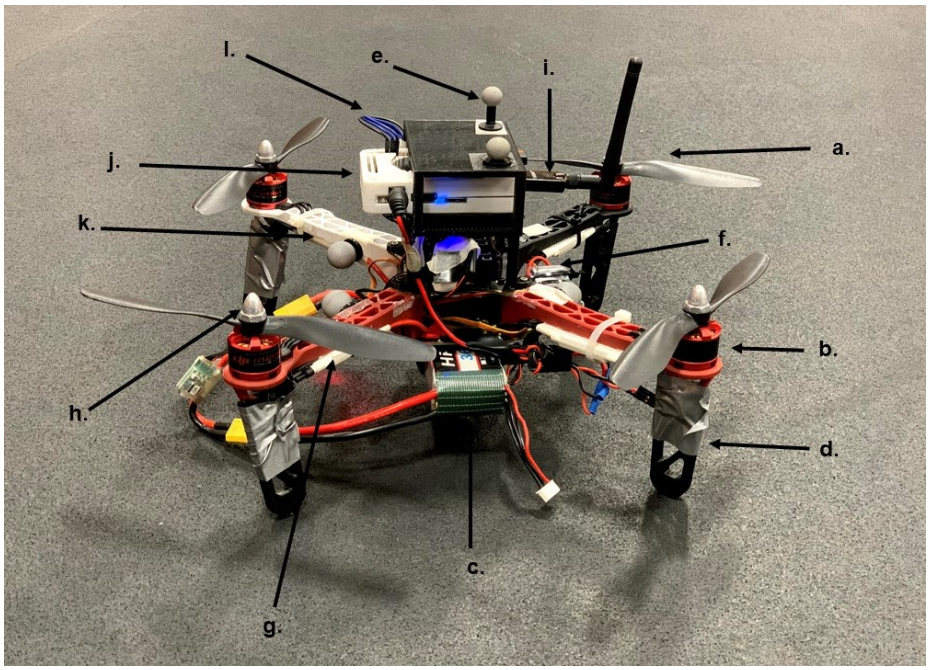

Figure 24: DJI F330 Quadrotor Platform Used for Real-time Flight Tests. (a.) DJI 8045 Propellers, (b.) 2212/920 kv motors, (c.) 4S LiPo Battery, (d.) Landing Gear, (e.) Reflective Markers, (f.) Battery Elimination Circuit, (g.) DJI OPTO 30A Electronic Speed Controller, (h.) Propeller Nuts, (i.) 5G Enabled Wifi Adapter, (j.) Intel UP Board Companion Computer, (k.) Arms, (l.) Serial UART Connection Between (j.) and PixHawk Controller. PixHawk and Receiver are beneath (j.) Not Labeled in Figure.

of building a micro aerial vehicle to participate in a competition. The goal of this flight controller hardware unlike others during that time was to aide vision-based navigation [36]. Another important hardware on the lower-level is PX4FLOW which is a combination of optical flow sensor [37] and a range sensor. This was initially used for localization of the platform used in this thesis, however, it failed to provide navigation and the drone shown in Figure 24 kept drifting away. This was because the optical flow sensor worked well only on terrain that had a pattern but the studio in which tests were to be conducted rather had a plain black surface. Moreover, indoor localization using multiple offboard cameras each having a different field of

view would yield more precise ground truth data and hence successful integration of the lower-level with the higher-level is vital.

To achieve successful real-time control of the drone, lower-level was configured to support offboard flight mode. In offboard mode, the PixHawk flight controller listens to setpoints sent by a companion computer that was physically wired by serial UART wires or by USB to tty converter. In this drone, no converter was used instead, two boards were connected directly using serial UART wires. From QGroundControl application, the following parameters were re-configured for offboard mode. PX4 $v1.9.0$ was used as the flight control software.

1. Since, EKF does not subscribe to ATT_POS_MOCAP MAVLink message, Local Position Estimator or LPE module was selected for indoor navigation. Local position estimator is an extended kalman filter for 3D position and velocity states and Q attitude estimator is quaternion based complementary filter for attitude. Both were enabled by setting SYS_MC_EST_GROUP parameter to 1.
2. To make PixHawk listen to yaw from motion capture, the ATT_EXT_HDG_M parameter was set to 2. Even with this setting and correct mapping of axes, a toilet bowling effect was visible from the motion of the drone. This behavior of downward spiral motion of drone with increasing radius is a result of consistent drift in yaw. It was later found by trial and error that, although not mentioned in the developer guide, the controller tends to fuse both onboard magnetometer and offboard yaw leading to fluctuations. When monitoring compass in QGroundControl, the heading would not settle at the actual heading of PixHawk. To resolve this issue, SYS_HAS_MAG was disabled and weight on the attitude quaternion filter was increased to 0.3 from the default value of 0.1.

3. For configuring the communication with the companion computer, SER_TEL2_BAUD was set to 921600 baud rate, MAV_1_MODE to onboard and MAV_1_CONFIG to was mapped to Telem 2 port as the serial UART connectors were physically wired to that high-speed port.

6.2.4 First Flight Test

After integrating the components of the system, configuring the Motive server for data streaming, and making appropriate changes in the lower-level end for offboard control, initial flight tests were done to confirm the closing of the loop. This was done by two means. Statically, placing the drone within the capture volume with the flight controller plugged into a laptop and monitoring the MAVLink inspector for messages in the QGroundControl application. Specifically, visually it was confirmed whether the ATT_POS_MOCAP message was appearing in the inspector indicating that the current pose was reaching from Motive to PixHawk. Secondly, using radio control, the drone was lifted off in manual mode, after it has gained a significant altitude, switching to altitude mode was done to visually assess z-axis performance. Another indicator that, drone is receiving feedback correctly, is a green LED flash upon transitioning in between modes. If altitude hold was itself sluggish with default gains, there could be other factors within the experimental setup that could be causing such performance and need to be debugged for before proceeding to test position control mode. In this case, an excellent altitude hold with the anticipated motion in the x-y plane was observed. Hence, in the following flights, the drone from altitude mode was quickly switched to position control mode. There existed minor oscillations about pitch and roll axes but these oscillations were accounted for other experimental factors such

as worn out markers and intermittent loss of rigid body while tracking. However, the position control hover performance was found satisfactory after addressing those issues. Those challenges and the way they were addressed are mentioned in 6.6.

6.3 Results From Real-time Flight Tests

For each trajectory the experiments were performed as follows. First, the quadrotor was taken-off in manual mode and once it gained sufficient altitude it was then switched to altitude mode. Once a stable altitude hold was visible a rather quick transition was made to position control mode. Following this, the quadrotor was transitioned into offboard flight mode where, the quadrotor was programmed to hover at the starting point of the trajectory. When a flag was triggered from the ground station, the drone would execute the trajectory 10 times.

This was done for the ease of studying the steady-state and transient behaviors separately by easily identifying the mode switching from the captured response so that the desired response could be trimmed. It should be noted that, only x and y response plots are shown as z axis and yaw setpoint was maintained constant.

6.4 Straight Line Tracking

To evaluate the tracking performance of both controllers for ten straight lines, first, have a look at the Figure 25.

From the figure, it can be seen that not a significant difference in tracking between compensated and default controllers are visible from the planar view. Only for 3 lines,

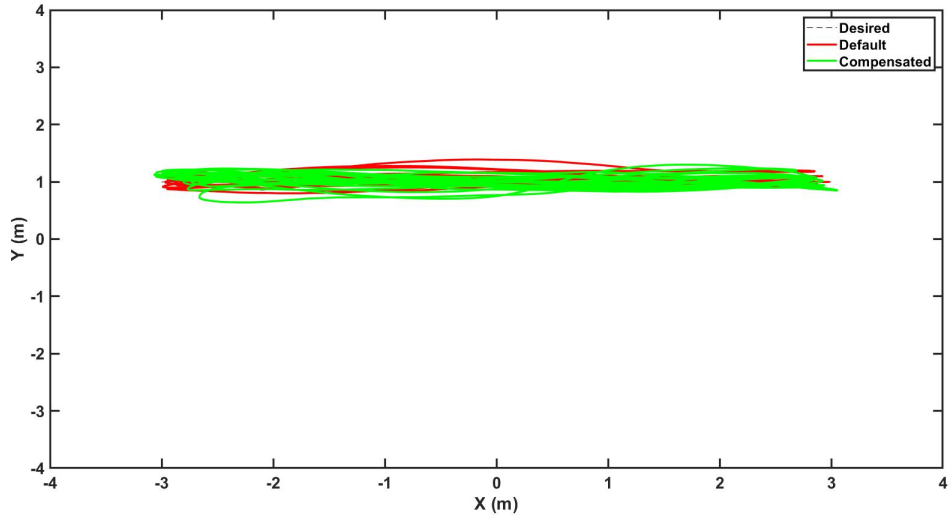

Figure 25: X-Y Planar View Comparison of Tracking of 10 Straight Lines

the default controller seems to overshoot in the y-direction, so only when we look at the other profiles a comment on the performance can be made.

Moreover, even by looking at the position profiles in the Figure 26, we can observe

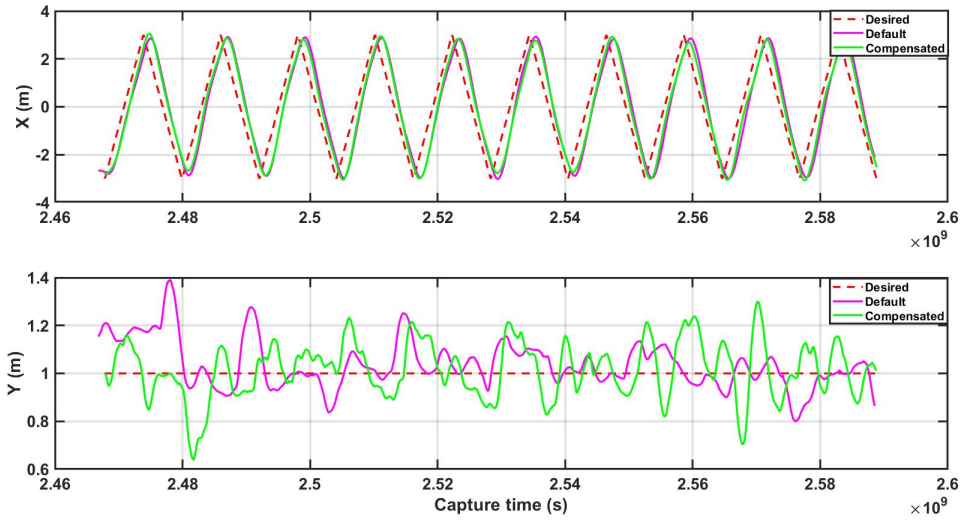

Figure 26: Position Profiles of 10 Straight Lines Tracking with and Without Drag Compensation

there is an overlap for x-axis tracking for both drag-augmented and default controllers which tend to track with delay. For the y-axis a maximum steady-state error of 40 cm for default and 75 cm for compensated controller is noted. Although no direct conclusions on improvement in tracking could be noted from position profiles, the relatively large root mean squared error in position can be justified from the average steady-state default controller performance which is tending to have many overshoot and oscillations. This is an indicator that the baseline controller performance needs to be improved either by additional tuning or by assessing the experimental factors that seem to affect its performance.

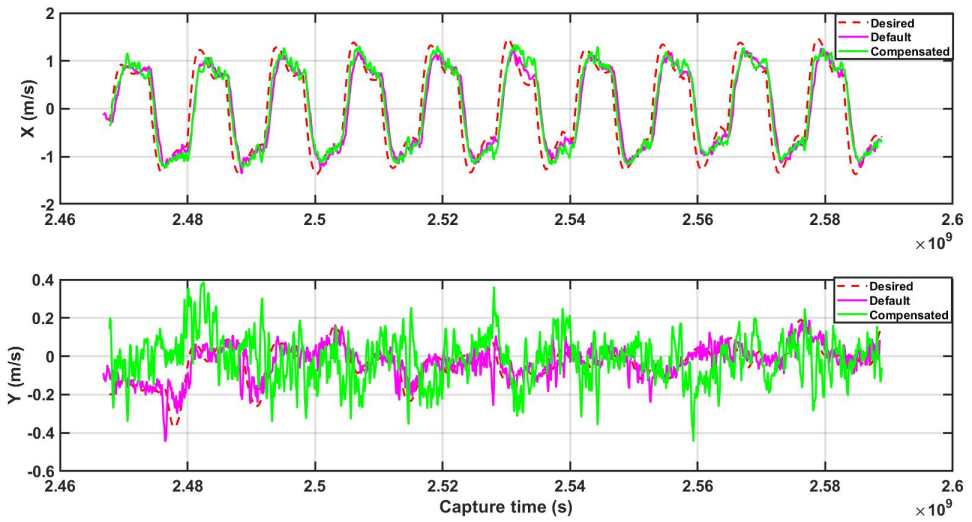

Figure 27: Velocity Profiles of 10 Straight Lines Tracking with and Without Drag Compensation

From the velocity profiles represented in Figure 27, it is again evident that there is no difference in the tracking between the compensate and the default controllers. The velocity profiles here have demonstrated to indicate the velocity attained by the

quadrotor. The maximum velocity attained is 1.2 m/s and these results indicate that at these velocities the effects due to compensation are not visible.

Table 1: Root Mean Squared Error in Tracking of 10 Straight Lines

Axis	Without compensation (m)	With compensation (m)	Difference (m)
X	1.4862	0.8179	0.6683
Y	0.1051	0.1174	−0.0123

Finally, the compensation performance for straight line case is summarized in the Table 1. Although not directly visible due to the density of lines in the position profiles, there seems to be a significant improvement in the x-direction performance in straight-line tracking. This was although not expected at such low speeds, reinforces the presence of compensation.

6.4.1 Circular Trajectory

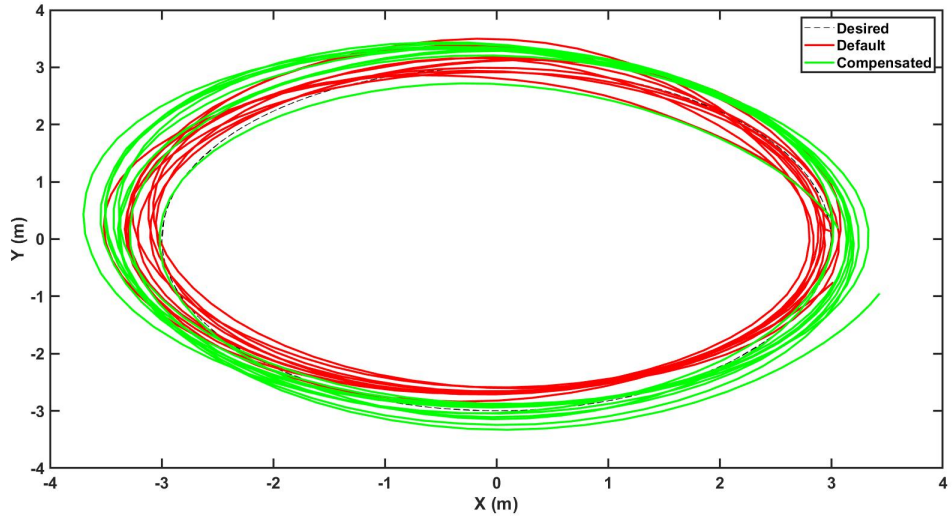

Figure 28: X-Y Planar View Comparison of Tracking of 10 Circular Trajectories

The Figure 28 above represents the top view of the tracking in the x-y plane for 10 circular trajectories. The green lines correspond to tracking with compensation while the red lines are for default case. The compensated trajectories were expected to track the reference closer when compared to default trajectories. This is visible from the Figure from $(-2.5, -2.5)$ until $(2.5, -2.5)$ where the compensated tracking is overlapping with the reference trajectory. From coordinates $(-2.5, -3)$ to $(2.5, -2.5)$, in the left half of the trajectories, the compensated trajectories tends to overshoot even the default trajectories, with maximum overshoot beginning around $(-2.5, 3)$ and decreasing around $(2.5, 3)$. Like mentioned before, there is no overshoot in the right half of the plane, where the trajectory actually begins. The overshoots on the left half of trajectories could be due to the feed-forward thrust term, which might be overcompensating and in turn decreasing the performance instead of remaining constant or decreasing at those instants.

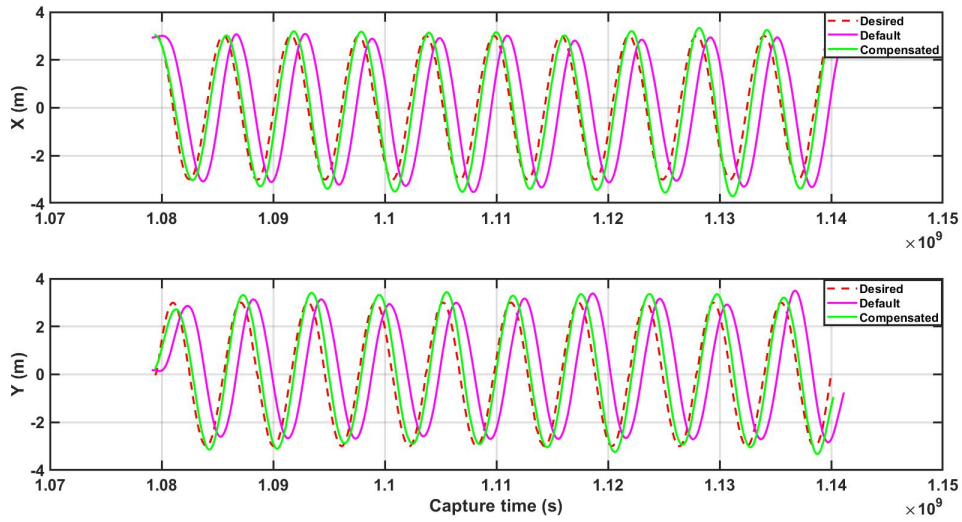

Figure 29: Position Profiles of 10 Circular Trajectories Tracking with and Without Drag Compensation

To have a better insight into tracking performance, consider the Figure 29. As seen from the figure, the compensated controller is aligned with the reference input from start until the end. The compensation behavior is appearing as a phase-shift in the default trajectory response. It is to be noted that, both the default and compensated controllers were not compensated for delay in the system, for, it wouldn't make for a suitable comparison. Moreover, overshoots are also visible in the compensated trajectory, and the overshoots persists more the in the y-direction when compared to x-direction wherein the overshoot of compensated controller are almost the same as default controller. After assessing the Figure 28 and Figure 29 it is clear that compensation has occurred and performance improvements in parts of the trajectories can be noted.

Before the actual improvement can be noted, velocity profiles can be analyzed for any additional details which the previous two figures couldn't convey.

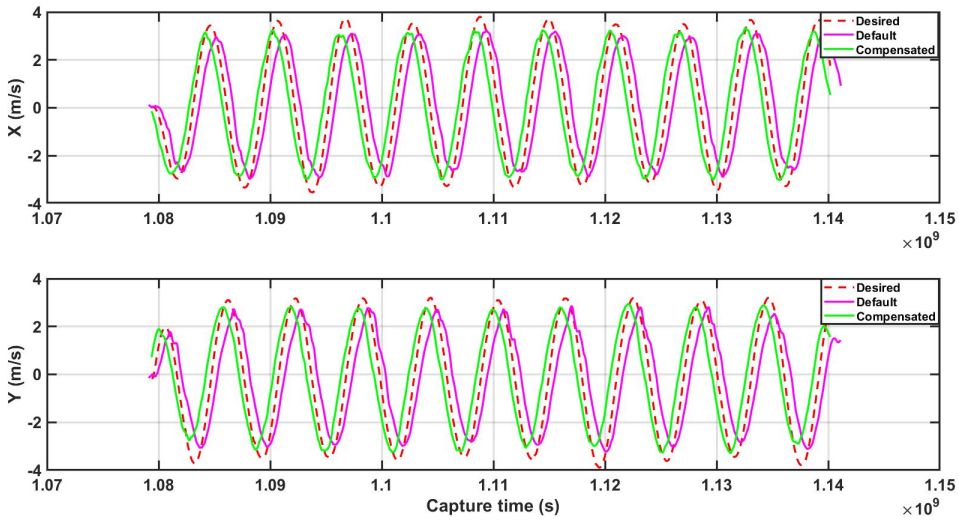

Figure 30: Velocity Profiles of 10 Circular Trajectories Tracking with and Without Drag Compensation

It can be seen from the Figure 30 that, the maximum reference velocities in both directions were about 3.3 m/s and that at certain time instants there are sudden peaks in reference velocity. Notice the time instants 1.11 in x-axis and 1.12 y-axis, a sudden peak in reference velocity is observed. For the same corresponding time instants on the position profile Figure 29 we can notice the overshoots being present, again, confirming the fact that at times the controller is overcompensating for the effect.

To summarize the performance for ten circular trajectories, refer to the Table 2 below. The root mean squared error in trajectory tracking has decreased in the x-axis tracking by 0.4811. In the y-axis, although in the position and velocity profiles, there were portions wherein the commanded velocities are higher, the tracking error has still decreased by 0.4316.

Table 2: Root Mean Squared Error in Tracking of 10 Circular Trajectories

Axis	**Without compensation (m)**	**With compensation (m)**	**Difference (m)**
X	1.0801	0.5990	0.4811
Y	1.0933	0.6617	0.4316

6.4.2 Rectangular Trajectory

From the Figure 31 we can observe the differences in tracking of compensated and default controller for rectangular trajectories. For the horizontal dimension both the compensated and default controllers tend to be densely embedded within each other indicating that the compensated controller performs very similar to the default controller. For the longitudinal dimension however, the compensated controller

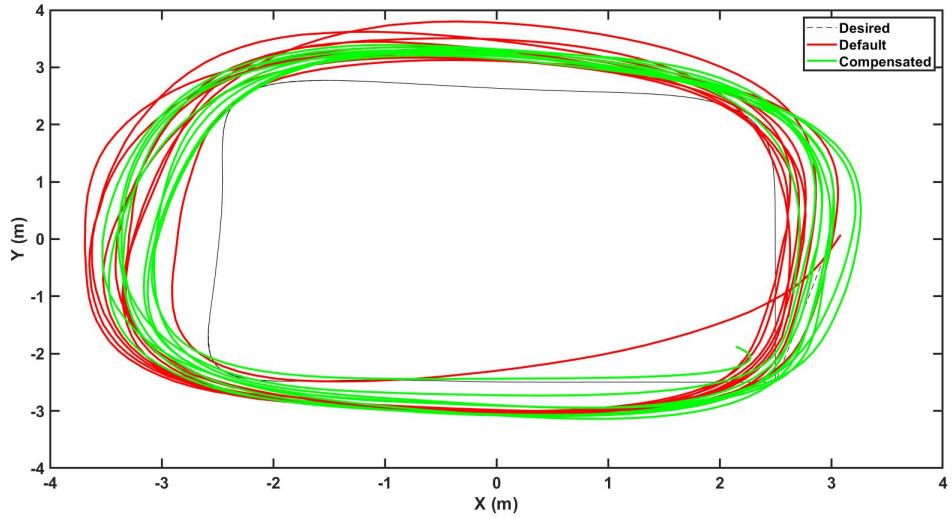

Figure 31: X-Y Planar View Comparison of Tracking of 10 Rectangular Trajectories

performs better in the left half of trajectory when compared to right half where it overshoots more than the default controller.

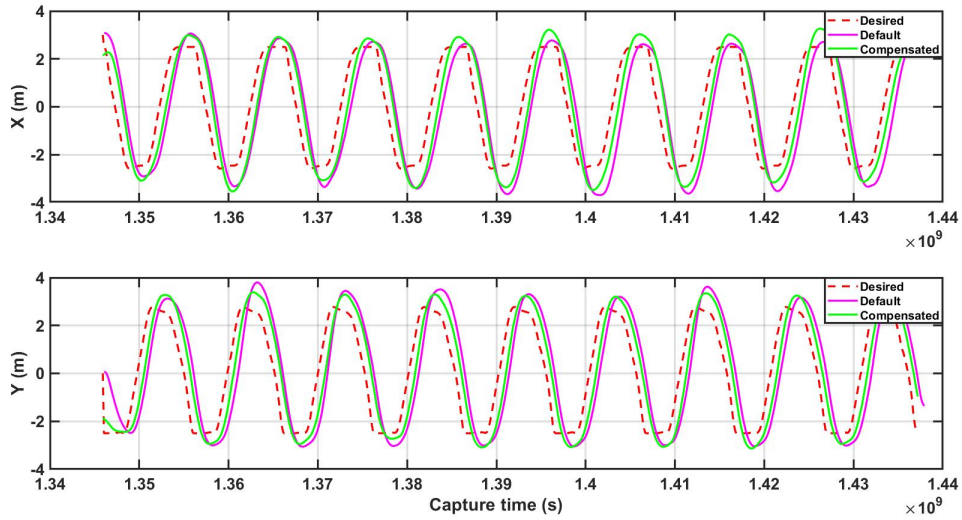

Figure 32: Position Profiles of 10 Rectangular Trajectories Tracking with and Without Drag Compensation

Figure 32 indicates that the drag augmented controller and the default controller

have very similar tracking performance with the lag still persisting in the default controller and slightly lesser lag in the compensated controller. The amplitude of the sine wave, the value of overshoot for the default controller in the y-direction seems larger than compensated controller, whereas the overshoot in the x-direction is observed to be more for compensated controller after 4 loops. Using position profiles, a conclusion seems unlikely to be drawn because the overshoots do not persist from beginning and exists locally only for certain loop or few loops towards the end of the trajectory.

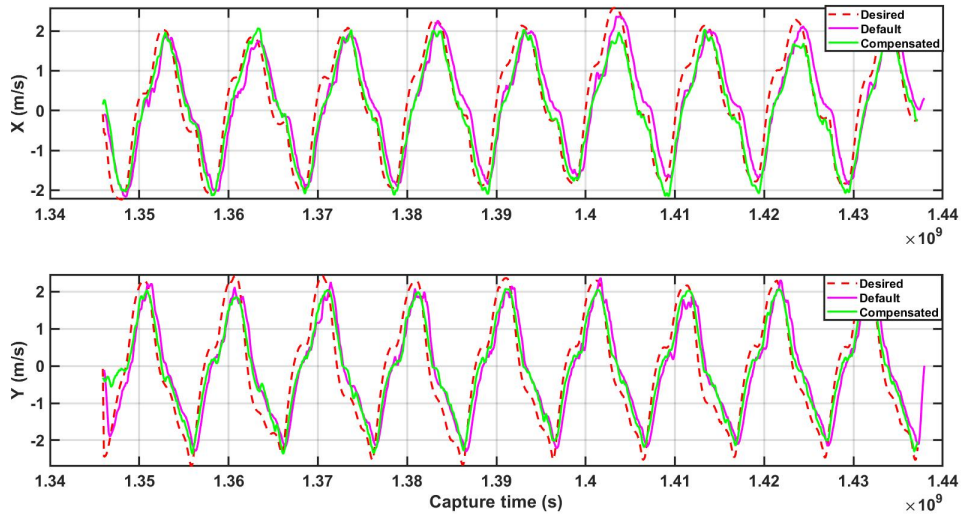

Figure 33: Velocity Profiles of 10 Rectangular Trajectories Tracking with and Without Drag Compensation

From the velocity profiles in Figure 33 yet again, the performance of both controllers is observed to be very similar and that they both seem to track reference velocity generated by the controller fairly closely.

Finally, the performance of the compensated controller for rectangular trajectory is summarized in the Table 3. From the numbers in the table, there is very minor

Table 3: Root Mean Squared Error in Tracking of 10 Rectangular Trajectories

Axis	**Without compensation (m)**	**With compensation (m)**	**Difference (m)**
X	0.9118	1.0369	−0.1251
Y	1.0042	1.0593	−0.0551

difference in y-axis performance and the x-axis performance has slightly degraded when compared to default controller. This might be due to the fact that at such low speed, there is no compensation happening due to which both the controllers have almost the same tracking performance.

6.5 Comment on Delay In The System

As it was seen in all the figures, there existed a delay in the system between the reference input and default controller response. There could be many reasons due to which this occurs but this is persisting because, by the time the quadrotor receives and corrects for a given state, there is already the next reference state fed to the system. Three main loops to be considered in this case are the frequency of the reference trajectory loop, frequency of controller loop, and the frequency of receipt of current pose message. As per documentation, the current pose message is to be received somewhere between $30 - 50$ Hz, for our experiments it was set to 50 Hz. It should be noted here that care was taken to choose the frequency of trajectory loop, that is, the trajectory setpoint publication shouldn't be too high that the drone already lags behind when a new setpoint is already published. Due to this choice of 20 Hz publication of reference trajectory was made. The cascaded-control architecture of outer-loop position and inner-loop velocity are already configured in such a way that the inner loop frequency is very high compared to outer loop frequency.

In addition to the fact that the receipt of state estimates, corrective control action

and publishing the PWM output signals are all not occurring at the same time instant could also be attributed to the processing power of the controller that directly influences the time delay in outputting the control action. In this drone, the oldest version (1) of the PixHawk flight controller hardware was used which rather had a 32-bit microcontroller and very less random access memory. Advanced version of PixHawk were available but were not opted for the purpose.

Having said the above, the final source of delay could be the delay in receiving the current pose message over a network. Although server side latency was recorded to be 4.2 ms from the Motive software, the latency in the receiving (higher-level) end was not measured and hence it still remains unknown in the experiment. Chances are high that, if this is high, then in reality quadrotor is receiving the pose at a later time instant and sending out control action for that instant.

It should be noted that, although the intent was to compare the effect of compensation without improving any performance or any experimental factors, the drag compensation, at least for circular trajectory case is tending to correct for delay in the system. This is due to the algorithm only and that the controller was not corrected for delay before compensation.

6.6 Addressing System Integration Challenges

There were quite a few issues that arose simultaneously that made the troubleshooting and debugging a challenging experience. Message dropouts, network issues such as latency, and inability to precisely calculate delay in the components of the system because of inconsistency in measurements made it difficult to predict the primary source

of the problem. The potential sources of problems were written down and instant flight tests addressing one issue at a time were attempted to notice any difference.

The first problem stemmed from the camera settings under the Motive software. At Arizona State University's Drone Studio Motive (version 2.1.1) was used to create rigid bodies, calibrate capture volume, and to collect precise ground truth data. It was initially challenging to determine the reason for the drone to ascend towards the ceiling of the capture volume at a high rate of climb when switching from manual or stabilized mode to altitude hold mode of PixHawk flight controller. This behavior occurred in consecutive trials leading to a hard landing followed by a crash. This action of the drone is an indicator that no feedback was being received resulting in no corrective control action being produced from the motors. It was not possible for a single person to simultaneously move the rigid body and monitor whether the rigid body or the asset is being tracked within Motive. It then turned out that among 5 markers that were installed, only 2 were being picked up the tracker. This could mean that software is unable to completely form a rigid body during real-time motion due to which no rigid body tracking data wasn't being logged during the time of flight. To confirm this flight data was captured for a short manual flight. When the data was exported and reviewed, almost all the rows were empty. After referring to the documentation [38], it was understood that camera gain and threshold settings were the closest settings to be configured to resolve the problem. Camera gain was increased from 1 (low) to 4 (medium). By doing so, this added more brightness to every pixel of the sampled image. Lowering the threshold reduces the filtering intensity such that the cameras can pick up worn out markers which might not reflect the infrared light with the same intensity a new marker of the same size would. But in this case, only

camera gain settings were changed every time the software was started and rigid body capture was successful.

Following this problem, it was repeatedly identified that the rigid body was losing tracking in specific areas of the capture volume. Losing tracking just before or around the border of capture volume was expected. However, the presence of blind spots inside the zones led to the doubt of poor calibration of the capture volume. The volume was re-calibrated this time, with more priority to ensure that the coverage or calibration wand was to overlap with that of the most probable region of operation of the drone during its flight. The author feels that the number of samples alone can be misleading quantifiers for the robustness of calibration. The problem was then resolved by again performing both walk tests and short manual flight tests to observe the performance.

Another cause of poor tracking other than calibration is the selection and position of markers. Although for this thesis, standard 15mm markers were used, the author when working with other drones containing more markers, with larger diameters offset at a height when compared to regular markers observed incredible tracking performance by the cameras for the same settings. Under the 2D camera view tab in Motive, for a similar-sized drone having 6 markers installed, the view was similar to one when stargazing into a dark sky. Almost all cameras were lit up with all markers which turned out to be a reassuring factor for not losing feedback data real-time in flight.

With these new configurations, altitude and position hold tests were initiated following a take-off in manual/stabilized mode. The operator was always in the loop to switch back to manual mode in case the drone went out of control in specific cases such as nearing the border of a capture volume. Initial altitude hold tests and position

hold tests turned out to be successful and the behavior was as expected. This did confirm the fact that the loop is closed and feedback is being received. However, there was still some instability observed while holding the position.

This lead to the third major integration problem which was the network router itself. The source doubt arose from the observation of constant discontinuities in the connection both from the higher-level computer end and also the host's end resulting in frequent powering off and on the network router. All ZMQ contexts were assigned with the IP address of the localhost to avoid constantly changing the code and building because of the reallocation of IP address when turning on and off the router. From this it was clear that, we needed a router preferably with 5G compatibility and also robust network support.

Upon switching from home wifi purpose NETGEAR router to a router for high-end gaming purposes, we noticed a significant improvement in the steady-state error in x,y,z position dropping from 40cm with the old router to under 10 cm without gain tuning with the new router. This again confirms the resolution of network-related issues. It should also be noted that replacing the router also resolved network delays with the same data rate configurations both from Motive and higher-level computer end.

Although efforts were made to measure latency in the network, replacing the router seemed to have resolved most of the network issues including latency. However, it is worth making a mention about those efforts here. In order to measure the server-side latency the author referred to the status bar in the bottom right- hand corner of the Motive software. By default, "Data" is chosen for monitoring purposes but "System" can be chosen to monitor server-side latency. As per the Motive documentation, the system parameter indicates the time measured between camera exposure until when

Motive has completely solved tracking data. In this case it was measured around 4 ms. In the client end, in the callback function of ROS subscriber node a sample counter variable and also a variable to measure total time was integrated with the remaining code. Using this, the total number of samples expected for the streaming rate for the program run time can be calculated manually and the difference between this and the actual number of samples received might help calculate the number of messages being lost. Since the author did not obtain consistent and repeatable results due to poor connectivity in the network, not much insight could be made from this effort. Finally, there are native NatNet applications available that can measure the system latency and total transit time.

Chapter 7

CONCLUSION

To conclude, two types of aerodynamic drag were modeled viz. parasitic drag and propeller drag. To the author's best knowledge this is the first time an attempt has been made to model parasitic drag with offboard data-driven approach. However, considering time constraints, only parasitic drag was compensated for in the position controller of the PX4 firmware (version 1.9.0). Simple straight lines and circular trajectories were generated. Differential flatness property was exploited to generate a rectangular minimum snap trajectory with corridor constraints applied suitable to operate within the drone studio space. Flight tests were conducted where the quadrotor was made to execute the same trajectory up to 10 times and the performance in the transient state was analyzed.

From root mean squared errors, it was observed that for circular trajectory both x and y-axis tracking was improved by compensation. The x-axis error reduced by 44.54% and y-axis by 39.47%. For the rectangular trajectory, however, almost no change in the tracking was observed. The root mean squared error value for both the axes indicate that the performance has been degraded by a maximum of 12%. The tracking error in the x-axis for straight lines has also observed to reduce by 44.96% with almost no noticeable change in its y-axis tracking error.

It is to be noted that, the above is the first set of results. Possible reasons for the lag of that existed in the default controller were discussed and few workarounds will be suggested in this chapter. The drag compensated controller behaved like a default controller with phase shift.

7.1 Limitations

Although a careful assumption on the order of velocity was made considering the range of operation of the quadrotor, mathematically the coefficients of the second-order terms were not negligible. Hence the correct relation of drag with velocity and the impact of considering second-order velocity term couldn't be justified.

Some of the model parameters viz. the motor angular velocity and the pitch angle could not be used for feed-forward compensation. Since reference terms can only be in terms of differentially flat outputs such as position, yaw, and their derivatives, pitch angle and motor angular velocity do not fit into the criteria and hence they are unable to be fed forward for compensation. There also exists no mapping for determining pitch angle and motor velocity corresponding to an operating point beforehand using the reference values. Hence this model couldn't be completely exploited for the main purpose of improving the accuracy of tracking performance.

The static mapping assumption between velocity and drag won't hold well for high-speed trajectories as the rate of change of velocity would then become significant for such trajectories.

Exploiting symmetry to assume a similar lateral model with corresponding parameters did not work very well. Drag compensation in the lateral axis did not seem to augment performance for almost all cases. Further experiments needs to be done to assess if compensation is indeed improving the performance.

7.2 Future Work

First and foremost, the baseline controller performance needs to be significantly improved to bring down the tracking error in the transient state from meters to under centimeters. A few workarounds to delay which is present in the system could be considered. An advanced version of PixHawk or any other flight controller hardware with enhanced processing power than the current STM32 microcontroller along with higher RAM could be made use of to overcome any processing latency. Among the three trajectories, the circular trajectory was the only one that recorded a maximum velocity of 3.3 m/s. Reference trajectories with a minimum velocity of 3.5 m/s or higher should be designed for the future so that improvement in tracking could be observed not just for one but all the trajectories. Along with this, compensation should be done for only one axis separately and then extend to multiple axis to further get an insight into the effect of compensation for undesired accelerations in multiple axes. Following this a known refined thrust model can be adapted from [15] for augmenting z-axis tracking performance as well.

In this work, only parasitic drag was compensated for, and that too only in the position controller loop assuming that the aerodynamic moments generated would be small. Further, this should be extended to compensate for propeller drag separately or if possible compensate for both the drag combined. Compensation for aerodynamic moments could also be achieved in the attitude control loop.

Most works assume drag is linear in velocity. A comparison can be made to see the difference in tracking error for the simplified model wherein the drag is linear in velocity to the original model obtained wherein the drag was found to be proportional to square of the velocity. If the latter proves to be holding well, then the bounds

where it is valid can be established and emphasis can be made to implement the latter instead.

REFERENCES

[1] Frost and Sullivan. *Commercial Drone Market to Hit 2.44 Million Units by 2023, says Frost and Sullivan*. 2020. URL: https://www.prnewswire.com/news-releases/commercial-drone-market-to-hit-2-44-million-units-by-2023--says-frost--sullivan-301037699.html.

[2] M. Agcayazi, E. Cawi, A. Jurgenson, P. Ghassemi, and G. Cook. "ResQuad: Toward a semi-autonomous wilderness search and rescue unmanned aerial system". In: *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 898–904.

[3] J. Cui, S. Phang, K. Ang, F. Wang, Y. Ke, S. Lai, Z. Yang, and B. Chen. "Drones for cooperative search and rescue in post-disaster situation". In: *2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. IEEE, 2015, pp. 167–174.

[4] Adlina AR. *UPS launches new drone technology for speedier, heavier deliveries*. 2020. URL: https://techwireasia.com/2020/03/ups-launches-new-drone-technology-for-speedier-heavier-deliveries.

[5] S. Sawadsitang, D. Niyato, P. Tan, and P. Wang. "Joint Ground and Aerial Package Delivery Services: A Stochastic Optimization Approach". In: (2018).

[6] F. Augugliaro and R. D'Andrea. "Admittance control for physical human-quadrocopter interaction". In: *2013 European Control Conference (ECC)*. EUCA, 2013, pp. 1805–1810.

[7] S. Mishra, D. Yang, C. Thalman, P. Polygerinos, and W. Zhang. "Design and Control of a Hexacopter With Soft Grasper for Autonomous Object Detection and Grasping". In: vol. 3. Dynamic Systems and Control Conference. Sept. 2018.

[8] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar. "Design, modeling, estimation and control for aerial grasping and manipulation". In: *IEEE International Conference on Intelligent Robots and Systems*. 2011, pp. 2668–2673.

[9] A. Zormpas, K. Moirogiorgou, K. Kalaitzakis, G. A. Plokamakis, P. Partsinevelos, G. Giakos, and M. Zervakis. "Power Transmission Lines Inspection using Properly Equipped Unmanned Aerial Vehicle (UAV)". In: *2018 IEEE International Conference on Imaging Systems and Techniques (IST)*. IEEE, 2018, pp. 1–5.

[10] J. Stumpe. *Drones to the rescue*. 2017. URL: https://aerospaceamerica.aiaa.org/features/drones-to-the-rescue.

[11] D. Mellinger, N. Michael, and V. Kumar. "Trajectory generation and control for precise aggressive maneuvers with quadrotors". In: *The International Journal of Robotics Research* 31.5 (2012), pp. 664–674.

[12] Uber Elevate. *Fast-Forwarding to a Future of On-Demand Urban Air Transportation*. 2016. URL: https://www.uber.com/elevate.pdf (visited on 09/30/2010).

[13] D. Yang, S. Mishra, D. M. Aukes, and W. Zhang. "Design, Planning, and Control of an Origami-inspired Foldable Quadrotor". In: *2019 American Control Conference (ACC)*. Vol. 2019-. 2019, pp. 2551–2556.

[14] D. Falanga, K. Kleber, S. Mintchev, D. Floreano, and D. Scaramuzza. "The Foldable Drone: A Morphing Quadrotor That Can Squeeze and Fly". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 209–216.

[15] J. Svacha, K. Mohta, and V. Kumar. "Improving quadrotor trajectory tracking by compensating for aerodynamic effects". In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 860–866.

[16] R. Gill and R. D'Andrea. "Propeller thrust and drag in forward flight". In: *2017 IEEE Conference on Control Technology and Applications (CCTA)*. Vol. 2017-. IEEE, 2017, pp. 73–79.

[17] G.M Hoffmann, H. Huang, S.L Waslander, and C.J Tomlin. "Quadrotor helicopter flight dynamics and control: Theory and experiment". In: *AIAA Guidance, Navigation, and Control Conference*. 2007.

[18] D. Sartori, D. Wou, L. Pei, and W. Yu. "A Revisited Approach to Lateral Acceleration Modeling for Quadrotor UAVs State Estimation". In: Institute of Electrical and Electronics Engineers Inc., 2018, pp. 5711–5718.

[19] S. Omari, M.-D. Hua, G. Ducard, and T. Hamel. "Nonlinear control of VTOL UAVs incorporating flapping dynamics". In: *IEEE International Conference on Intelligent Robots and Systems*. 2013, pp. 2419–2425.

[20] M. Bangura and R.E. Mahony. "Nonlinear Dynamic Modeling for High Performance Control of a Quadrotor". In: *ACRA 2012*. Australian Robotics and Automation Association, 2012.

[21] H. Haomiao, G.M Hoffmann, S.L Waslander, and C.J Tomlin. "Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3277–3282.

[22] J.-M. Kai, G. Allibert, M.-D. Hua, and T. Hamel. "Nonlinear feedback control of Quadrotors exploiting First-Order Drag Effects". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 8189–8195.

[23] M. Faessler, A. Franchi, and D. Scaramuzza. "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 620–626.

[24] E. Tal and S. Karaman. "Accurate Tracking of Aggressive Quadrotor Trajectories Using Incremental Nonlinear Dynamic Inversion and Differential Flatness". In: vol. 2018-. Institute of Electrical and Electronics Engineers Inc., 2019, pp. 4282–4288.

[25] M. Bisheban and T. Lee. "Geometric Adaptive Control for a Quadrotor UAV with Wind Disturbance Rejection". In: vol. 2018-. Institute of Electrical and Electronics Engineers Inc., 2019, pp. 2816–2821.

[26] D. Mellinger and V. Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011.

[27] Y. Aloimonos. *Vision, Planning and Control in Aerial Robotics*. Available at https://cmsc828t.github.io/. 2018.

[28] V. Kumar. *Motion Planning for Quadrotors*. Available at https://www.coursera.org/learn/robotics-flight. 2016.

[29] S. Mao. *minimum snap trajectory planning in MATLAB*. Available at https://github.com/symao/minimum_snap_trajectory_generation. 2017.

[30] Dronecode. *Using Vision or Motion Capture Systems for Position Estimation*. 2019. URL: https://dev.px4.io/v1.9.0/en/ros/external_position_estimation.html.

[31] Intel. *Up board series*. Available at https://up-shop.org/up-board-series.html.

[32] P. Hintjens. *ZeroMQ: messaging for many applications*. " O'Reilly Media, Inc.", 2013.

[33] S. Mishra. *A light framework for high-level and fcu communication*. Available at https://github.com/flyingmachines/litefw. 2019.

[34] C. Kohlhoff. "Boost. asio". In: *Online: http://www. boost. org/doc/libs/1* 48.0 (2013), pp. 2003–2013.

[35] K. Sathyamurthy. *Companion computer to PixHawk bridge*. Available at https://github.com/ksat4dev/CC2PixHawk_bridge. 2019.

[36] 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". In: *Autonomous Robots* 33.1-2 (2012), pp. 21–39.

[37] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys. "An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications". In: *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2013, pp. 1736–1741.

[38] NaturalPoint Inc. *Motive UI: Devices Pane*. 2019. URL: https://v22.wiki.optitrack.com/index.php?title=Devices_pane.