

State Estimation and Optimal Control for Racing Drones

In search of control algorithms for competing
against human pilots

Nilay Sheth



State Estimation and Optimal Control for Racing Drones

by

Nilay Sheth

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on September 26, 2019.

Project duration:	March 2019 – September 2019	
Thesis committee:	Dr. G.C.H.E. (Guido) de Croon	Control and Simulation, LR, TU Delft, supervisor
	Ir. C. de Wagter	Control and Simulation, LR, TU Delft, supervisor
	Prof. dr. K.G. Langendoen	Embedded and Networked Systems, EWI, TU Delft, supervisor
	Marco Zuniga, PhD	Embedded and Networked Systems, EWI, TU Delft

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Acknowledgements

Blaine Lee quotes Hellen Keller - *“There are red letter days in our lives when we meet people who thrill us like a fine poem, people whose handshake is brimful of unspoken sympathy and whose sweet, rich nature imparts to our eager impatient spirits a wonderful restfulness...”*. This project is dedicated to everyone who thrilled me and to everyone I look upto. Having this said, my amount of gratitude follows no specific order, is equal and abundant towards everyone mentioned or not mentioned in the acknowledgements.

Being able to experience a couple of wonderful journeys of life, this inquisitive journey of a Master’s programme in particular, was only possible by the encouragement and support of my Mom and Dad, my loving cousins - Allauki and Shivani and my entire family.


A big “thank you!!” to the entire MAVLab and my friends Federico and Yingfu who have played a great role in shaping this project. It has been a privilege to travel with Fede and to learn about his research interests! Fede’s observant review on my work encourages and assures me of the assumptions I make. Yingfu’s algorithms have encouraged me to write some of mine, after he kept telling us “You have an idea? Do it!”. Mario, Diana and Shuo have always made me feel at home with their friendship. Erik’s mentorship has helped me fix issues in no time! Anoosh and Jelle’s presence in the drone-race team has helped us all dream father.

With the fun loving nature of my supervisors Christophe and Guido I could start flying in no time and make exciting experiments with their drones. This report is a product of Christophe’s ideas of flying fast at the limits of saturation and Guido’s witty comments about looking beyond any usual levels of investigation! Koen’s attention to detail, his critical remarks on every scientific statement of the report along with the challenging and intriguing courses taught by him at TU Delft have been the highlight of my masters program!!

A lot of my research aptitude was nurtured under the mentorship of my seniors - Saurabh, Parita, Nachiket, Bavs and my batchmates Udit, Kewal, Rahul, Dhiraj, Anshuman from my undergraduate university, VJTI. I’d like to extend this acknowledgement to my professors Shailesh Sansare, from whom I derive my passion for engineering. Dr. Faruk Kazi and Akanksha Chauhan, from whom I derive the confidence to take up academic pursuits.

I’d like to thank my friends Mihir, Neel, Husain, Sukanya, Aditi, Kaushal who have been of great support during tough times, shared delicious food and have taken me for exciting trips with them. Maria’s sudden invitations for a bike ride always gave me a good break from my mundane schedule! I thank my peers Himanshu, Arvind, Erik, Jeoff, Alex and Joana from TU Delft - where I had the chance to study with everyone who were equally determined to bring a change to the world. A special token of gratitude to my friend Kanvi, whose opinions and feedback during the masters journey created a profound impact on my academic interests and whose company I truly enjoy!

I’d also like to thank Nikitas, Dr. Robert and Dr. Jyoti from the Space Systems group for their mentorship and friendship since the PocketCube project!

Apart from this, I’d like to thank  for the hours of playtime it gave me, unconsciously inculcating a sense of creativity and sometimes even resourcefulness during shortage of crucial parts.

Contents

1	Introduction	3
1.1	A brief introduction to autonomous drone racing	3
1.2	Scope	4
1.3	Conditions inside a drone-race	5
1.4	MAVLab's drone-racing pipeline	6
1.5	Requirements for the control module	7
1.6	Problem statement	8
1.7	Motivation	9
1.8	Organization	9
2	Introduction to Quadrotor Dynamics	11
2.1	A brief introduction to the area of state estimation	11
2.2	Quadrotor dynamics	13
2.3	The Newton-Euler modeling approach	14
2.4	Frames of reference and Rotation matrices	15
2.5	Difficulties in control and estimation for QRs	16
2.5.1	Observability.	16
2.5.2	Reachability	17
2.6	Discussion	17
3	State estimators for Quadrotor Control	19
3.1	Positional state estimation	20
3.2	Thrust.	20
3.2.1	Induced velocity method.	21
3.2.2	Hover-thrust/constant-altitude method	22
3.2.3	Thrust model augmented with drag and effectiveness based terms	22
3.2.4	Discussion	23
3.3	Drag forces	25
3.3.1	Lift induced drag/rotor drag	25
3.3.2	Blade flapping	25
3.3.3	Parasitic drag	26
3.3.4	Key take-away from the classification of drag forces	26
3.3.5	Drag parameter estimation	27
3.4	Modeling of QR dynamics.	29
3.4.1	Conventions and Symbols	29
3.4.2	The GRASPLab, UPenn and ANU estimator	29
3.4.3	The ETH-Zurich/CNRS France estimator	30
3.4.4	The UZH estimator	31
3.5	Summary of prior estimators	31
3.6	Requirements for a better lateral-state estimator	32
3.7	Proposed state estimator	33
3.8	Evaluation of different state estimators	34
3.8.1	Results: Maneuvers with high lateral velocities.	34
3.8.2	Results: Maneuvers with high angular velocities	35
3.9	Analysis	36

4	Optimal control	39
4.1	Position control for QRs	39
4.2	Existing QR control techniques	40
4.2.1	PID.	40
4.2.2	Minimum snap trajectories and Differential flatness.	41
4.2.3	Model predictive control.	42
4.2.4	Requirements for a better QR position controller	44
4.3	Proposed controller	45
4.3.1	A brief introduction to optimal control.	45
4.3.2	Proposed algorithm for control of racing drones	45
4.4	Results	50
4.5	Profiling.	51
4.6	Discussion	52
5	Conclusion and Recommendations	55
5.1	Conclusions.	55
5.2	Improved pipeline after this thesis	56
5.3	Applications of this work	57
5.3.1	Applications in the area of state estimation	57
5.3.2	Applications in the area of optimal control.	57
5.4	Recommendations	58
5.4.1	State estimation	58
5.4.2	Optimal control	58
5.5	Pontryagin's principle.	59
5.5.1	Existence of a unique analytical solution.	60
5.5.2	Pontryagin's minimum principle (PMP)	60
5.5.3	The Boston University algorithm.	60
A	Appendix	63
A.1	Scaling laws.	63
A.2	Modeling, Control and System Identification of the Moment models	63
A.3	Accelerometers for positional state estimation	65
A.4	Brachistochrone problem.	65
	Bibliography	67

Abstract

The e-sport of drone-racing involves human pilots to race against time. Recently, drone races have also gone fully-autonomous. As a result, these agile robotic platforms not only pose challenges of flying fast to the participating pilots but also create challenges for the flight control computers.

As a result, the concept of autonomous drone racing has gained significant attention from research groups around the world. These races aim to push the boundaries of perception and control algorithms, while simultaneously mitigating the real-world uncertainty of autonomous systems. While perception algorithms face challenges due to limited feature detection, high motion blur and computational requirements, control algorithms face challenges of convergence to the desired trajectories that are planned out in the race arena.

This thesis addresses the challenge of control for racing, which is responsible for guiding the drone to design and track desired trajectories for fast flights. The control sub-modules of racing drones are responsible for generating trajectories for fastest possible flights and also for obeying these generated commands. Additionally, the requirement of limited algorithm complexity is added to match the philosophy of computationally efficient algorithms at the Micro Air Vehicle Laboratory.

However, to address the requirements of these control sub-modules, the prerequisite of accurate state estimation always persists. Assigning control actions to a robot without information on the current state of the robot is rather unwise. As a result, this thesis first aims to perform accurate state estimation before designing controllers for time-optimal trajectory tracking. Again, another constraint of using only a single sensor (i.e. the Inertial Measurement Unit) is added to make the drone race in GPS denied environments.

As a result, the goal of the thesis is two-fold i.e. making accurate state estimators while using limited sensors and designing optimal controllers for taking the quickest trajectory through the arena. To achieve the goal of accurate state estimation, existing techniques are studied. Several features from each of these methods are selected to design a new estimator. To achieve the goal of time-optimal trajectory generation, firstly, the flaws of traditional control methods are pointed out. A new optimal-control technique is proposed, which makes use of fundamental principles dating back several decades. This principle is then fused along with present-day optimization solvers.

Finally, the proposed state estimation and control algorithm are compared against prior (benchmarked) techniques in the area. Compared to existing optimal control techniques, the proposed algorithm leads to faster trajectories and consumes less computational power onboard.



Introduction

A decade back, strategy games were the most popular genre of computer games. Board games also usually fall under a similar category of strategy games. Taking ‘optimal’ decisions to maximize the resources owned by an agent while estimating opportunities to conquer more is the usual goal in most of these strategy games. As a result, these games improved the decision making skills of a regular gamer. A similar genre of gaming called ‘serious gaming’ is designed for a real-world purpose rather than for pure entertainment.

Similar to the idea of serious gaming, autonomous drone racing aims to solve a purpose rather than just being present for entertainment. Interesting research questions are put forth when these agile robotic platforms are made autonomous, in turn being more purposeful than just adding adrenaline elements for the participating pilots. As a result, numerous research laboratories throughout the world have started participating in these races [9].

1.1. A brief introduction to autonomous drone racing

A typical drone race involves the challenge of flying through a series of checkpoints/gates, and finishing the course within minimum time. Autonomous drone races are usually held in big indoor arenas with illuminated gates through which drones must localize themselves against, plan their trajectories and fly through them in the minimum amount of time possible. These gates are usually horizontally placed on the x-y plane and their rough locations are made available priori to the race. As a result, reconstruction of the map is not fully required, and the objective of the race is more speed intensive and less exploratory. Also, racing drones are guided to the arena at fixed intervals from each other, while the arena is usually obstacle free reducing the chance of collisions with other drones or any structures of the arena. Figure 1.1 shows a typical drone racing arena, although this picture is photo realistically rendered for a drone racing simulator [9].



Figure 1.1: Drone-racing in FlightGoggles [9], source: youtube channel of Sertac Karaman, MIT AeroAstro.

A couple of autonomous drone races have been organized in the past by the prominent International

Conference on Intelligent Robots and Systems (IROS¹). Drone Racing League and Lockheed Martin² have offered huge prizes for designing algorithms that can beat human pilots. Stanford university and Microsoft Research have hosted their own "Game of Drones"³ for offering challenges in perception and control algorithms. The incentive for companies and universities collaborating at this level is to not only push research in the area of quadrotors but also into many different areas. (In this text, the terms racing drones and quadrotors imply the same meaning and will be called QRs in short).



Figure 1.2: Autonomous drone races are organized by IROS 2018 (left), Drone Racing League (center) and Stanford (right).

Autonomous drone races pose challenges in two major areas of research of perception and control. Perception algorithms use computer vision or neural networks to perceive the surroundings, so as to gain more information about the drone's position, velocity and orientation in the arena. Control algorithms issue commands to the propellers of the racing drone for making it track the correct trajectories. While doing so, these autonomous drones face numerous challenges like motion blur while executing perception algorithms, trajectory generation and tracking while executing control algorithms, non-linearities and aerodynamic effects and so on.

Impact: Research on control and perception algorithms done in drone racing can be transferred to other areas of research. Perception and control algorithms in case of QRs are based on Special Rotation group and the 3D-Euclidean space group. These configuration spaces are widely followed in any satellite's Attitude Determination and Control Systems (ADCS) and also widely used on factory assembly lines when industrial robots carry out welding, pick and place or packaging operations. If a lower dimensional version of the QR is considered, they can also replicate the dynamics of a self-driving car since QRs use similar control-perception pipelines for navigation [15]. If only a single rotor is kept under consideration, rotor aerodynamics of a helicopter can also be studied (in fixed blade pitch cases).

Due to the applications of quadrotors mentioned above and because of having numerous degrees of freedom in actuation, quadrotors can freely mimic many mechatronic systems. Algorithms running on such systems like self-driving cars, robotic manipulators and spacecrafts can first be tested on a QR. This can already give predictions about the stability, robustness and efficiency of the algorithms before the real-life deployment. Apart from the ability to serve as testing platforms, QRs have lately found direct applications in the area of agricultural monitoring, videography in sports and most importantly in search, rescue and surveillance operations.

On the basis of the points made above, it is apparent that research in the quadrotor area is not self-absorbed and provides plenty of incentives for collaboration between industry and academia. This collaboration also provides numerous commercial opportunities.

1.2. Scope

Autonomous quadrotors are agile robotic platforms on which sensors and actuators must be tightly coupled through perception, filtering and control algorithms, which make complicated calculations at a relatively high frequency. Due to this complexity a lot of scope for contribution arises. For narrowing down to a scope with respect to this thesis, an opportunity of contribution is found in the control and estimation area of QRs. The topic of control and estimation was chosen after observing the problems in flight trajectories from MAVLab's participation in earlier races.

The hierarchical diagram in Figure 1.3 enlists autonomous systems in general and then classifies them based on their respective degrees of freedom. QRs belong to the class of 3-dimensional systems, since they

¹Details about previous IROS drone races found at: <https://www.iros2018.org/competitions>

²More details about AlphaPilot found at: <https://www.herox.com/alphapilot>

³More details Game of drones hosted by NeurIPS: <https://www.microsoft.com/en-us/research/academic-program/game-of-drones-competition-at-neurips-2019/>

can actuate or hold any state in the 3D space as we perceive it. Moreover, due to high versatility of this robotic platform, it finds numerous applications in the field of autonomous systems, and hence deserves further classification depending on the applications it is being used in. A few challenges associated with each application are enlisted alongside each other in the purple block. The green block finally indicates that this thesis focuses on the control sub-module within the framework of drone-racing.

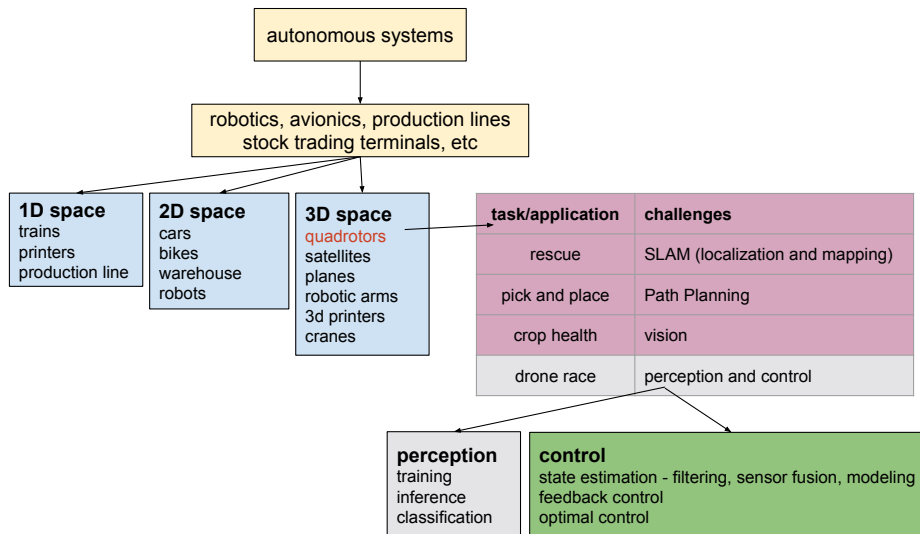


Figure 1.3: Zoomed out view of autonomous systems in general with respect to the scope of this thesis (in green).

As mentioned before, due to high versatility of the platform, the scope of this project could be extended to any control system, (e.g.: on self-driving cars, industrial robots and satellites), provided the control objectives are similar. This project tries to be more than "All board the hype train!" and does not intend to focus **purely** on tasks that QRs can do, but tries to make a contribution in the area of rigid body dynamics and control in general.

1.3. Conditions inside a drone-race

Out of the numerous challenges, the scope of this thesis has already been narrowed down to the control sub-module (Figure 1.3). The listing below describes the conditions at the drone-racing arena while addressing the associated control challenges:

1. **Sensors:** Most commonly, the frame of the QR is only allowed to carry a single camera and an inertial measurement unit (IMU). These two sensors satisfy the necessary conditions for a stable flight. Very frequently, the luxury of holding accurate localization data via GPS, Ultra Wide Band (UWB-localization) or LiDAR is not possible due to restrictions in the problem statement. Only occasionally, a laser time-of-flight sensor, barometric sensor or a SONAR sensor is mounted below the drone for height estimation.
2. **Actuators:** No additional actuators except the four propellers of the QR are present. If a QR must qualify as a racing-drone, the thrust to weight ratio of the vehicle must be greater than three. [20] suggests that the angular acceleration scales inversely to the length of the frame, while linear acceleration stays almost constant with change in scale. Section A.1 restates the derivations from [20] for comparing racing drone specifications. The Eachine Trashcan, the cover picture of the report, is the smallest drone in length present in MAVLab and hence could be classified as a racing-drone⁴.
3. **Conditions at the arena:** Drone races are held in indoor environments and it can be assumed that there are no wind disturbances that can disturb the trajectories of the QR. The velocities that the QR would see with respect to air would only be due to its own movement in space. Gates are well illuminated by spot lights for making perception less challenging, while they can be perturbed from their nominal location in each lap.

⁴Recently, the Trashcan is also made autonomous by MAVLab [19] and is the smallest autonomous racing drone at the time of its publication.

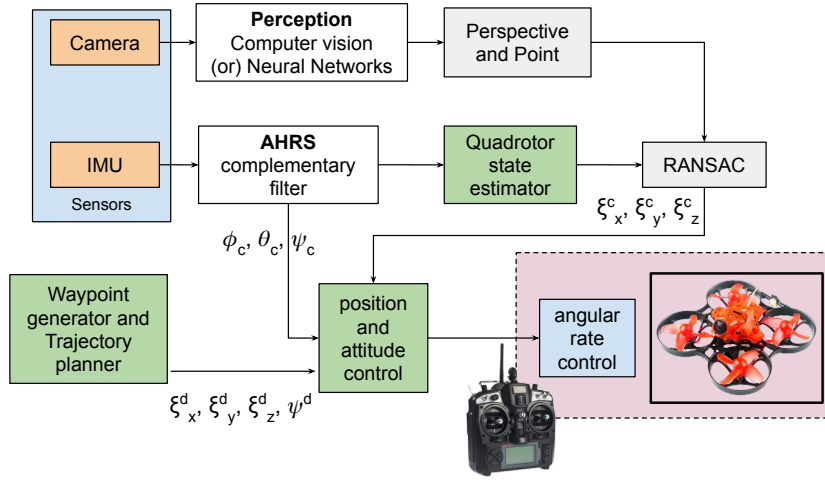


Figure 1.4: Individual sub-modules of MAVLab's drone race pipeline.

1.4. MAVLab's drone-racing pipeline

The closed-loop algorithm that is capable of facilitating drone-racing, is made up of several sub-modules as shown in Figure 1.4. When run together, these sub-modules are referred to as a 'pipeline' of the autonomous QR system. This is because the algorithms accept various inputs from sensors, perform a set of calculations and finally generate output commands for the QRs. The block in blue indicates the inputs to the pipeline while the one in purple indicates its output. The scope of this thesis is only limited to the blocks in green. Since there is non-trivial coupling between different modules of the pipeline, some bridging information is provided in the appendix of this thesis mainly highlighting how the vision and control modules are coupled. Since each module is critical to the functioning of the pipeline, some cross references might be made towards other modules while describing them in the listing below:

1. **Sensors:** Most common race drone setups come with an IMU and a monocular camera. However, performing control directly from the information provided by raw data from these sensors is not possible. Accurate high frequency measurements cannot be expected from the vision module, since it requires a lot of computational power. The IMU also usually does not give out the orientation or the position of the QR, but only gives out noisy and biased measurements of accelerations and rotational speeds of the body of the QR.
2. **AHRS:** The Attitude Heading and Reference system takes in noisy and biased readings from the accelerometer and gyroscope of the IMU, and fuses it using a complementary filter yielding accurate attitude estimates of the QR (later termed as ϕ, θ, ψ).
3. **Quadrotor state estimator:** The state estimator predicts the acceleration experienced by the QR depending on its bank angles. It also compensates for any effects of drag forces on the QR, giving close estimates of the actual accelerations that the vehicle experiences. It finally performs Euler integration to yield the velocity and position of the QR in the arena. The design of this block is most critical to the control module and is discussed in depth in this thesis. Meticulous design of this module also reduces a QR's dependence on the vision sub-modules for flying accurately.
4. **Perception:** The perception module is responsible for pre-processing, segmenting and finally post processing the images using the real-time camera stream. Pre-processing performs some scaling and distortion based transforms before segmentation. The segmentation algorithm thresholds any gate-like objects in the image frame. The post-processing algorithm runs a snake-gate algorithm [19] to return the corners of the gate in the camera frame of reference.
5. **Perspective and Point (PnP):** The PnP algorithm takes in the corners that are detected in camera frame and projects them to the world/inertial frame of reference. As a result, a QR can know its position in the arena simply by looking at the gate from a finite distance.

6. **Random Sample Consensus (RANSAC):** The output of PnP measurements follow a non-Gaussian model, yielding biased measurements when subjected to motion blur, mirroring effects, incorrect gate assignment, etc. The RANSAC module kicks out such biased outliers from the PnP measurement buffer. Hence this module provides vision-based corrections to the model-based predictions earlier made by the state estimator.
7. **Waypoint generator and trajectory planner:** This module holds a rough a priori map of the arena and selects next set of desired states that the QR should track. The desired states are a function of current position of the QR and the elapsed time of flight.
8. **Position and attitude control:** This module is responsible for generating appropriate commands to make the current states of the QR (ξ^c) converge to desired states (ξ^d). This module is also described in detail towards the end of the report since there are numerous techniques to make current states converge to the desired states.

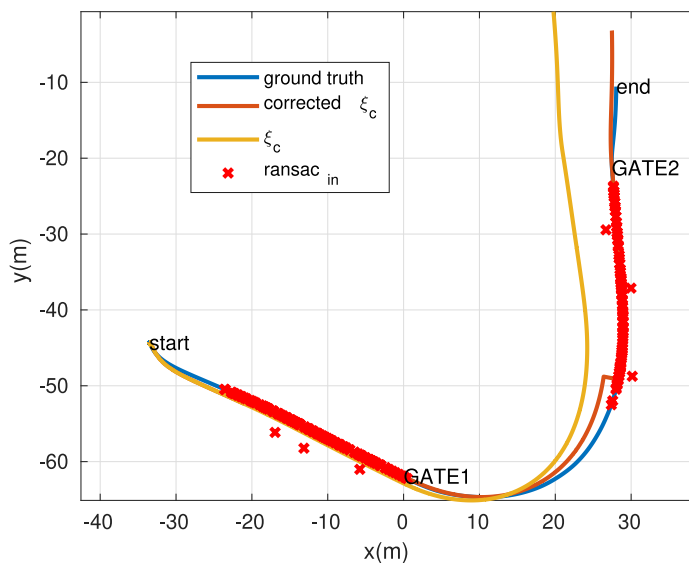


Figure 1.5: Top view of a drone-race flight trajectory highlighting the contribution of RANSAC to the state estimation module.

Figure 1.5 describes the nature of the outputs from the RANSAC and state estimation modules of a QR mid-flight. As the QR navigates through a course, the quality of its positional states (ξ_c) reduces with time. This can be seen in the figure by the drift between the current state (yellow) and the ground truth state (blue). The drift in these positional state estimates is corrected by the RANSAC module (corrected ξ_c). As seen from the red plot in Figure 1.5, the quality of current positional state estimates increases after the correction.

As seen in Figure 1.4, the control input u to the QR (purple block) is a function of the desired state (ξ_d) and the current state (ξ_c). i.e. $\hat{u} = f(\hat{\xi}^c, \xi^d)$. From this, it can be seen that the accuracy of control input improves as soon as the accuracy of current state estimate improves (i.e. when $\hat{\xi}^c = \xi_c$).

1.5. Requirements for the control module

The requirements to be addressed by the control module of the drone-race framework are as follows:

1. **Accurate:** The control algorithm must be able to *rely the least* on corrections made by the vision module. Relying on external sensors for control algorithms makes the control logic non-causal. After the non-causal point, predictions cannot be made. The only way to achieve good control in the absence of vision corrections is to have better state estimators that predict the state of the drone in future. This requirement of accurate state estimation is converted to become the aim of the first part of this thesis.
2. **Optimal:** To finish the drone-race track earliest, another requirement is added to the list. Flights through the desired states must take place in a time-optimal fashion. Also, while doing so, the control

algorithms must be able to allocate control inputs that use the entire flight capability⁵. While utilizing the entire flight capability, QR should reach the desired state in a time-optimal fashion while being very close to actuator constraints.

3. **Fast:** Control inputs must be generated fast enough to command the QR to fly through the gates. The time window for generating these commands when flying at high lateral velocities is very short. Failing to meet these time constraints might cause the QR to crash at the corners of a gate or miss the gate in fortunate circumstances.

A requirement also needs to be added to the above list because of the constraints coming from the testing platform. Experiments related to this thesis are carried out on a Parrot Bebop 1 QR. This QR runs a stripped down version of Linux (Busybox) on its ARM Cortex A9 processor. The Paparazzi toolchain⁶ is used to cross-compile a real-time application that threads at almost 512 Hz on the processor. This application is responsible for all the modules in the pipeline as shown in Figure 1.4. Starting from reading the cameras and the IMU, to sending control signals to each propeller is handled by this application. Since Paparazzi is thread-less and sequential in its execution, calculating control inputs for navigation should not hamper critical functions like the internal hover controller. For instance, inverting a huge non-sparse matrix can delay the integration of gyroscopes, which are responsible for calculating the stabilizing commands of the QR. This takes place when the delay due to extra function calls is more than the frequency of the periodic function of Paparazzi's stabilizing loop (512 Hz/2 ms).

1.6. Problem statement

To satisfy the requirements stated above, the problem statement for autonomous control of racing drones in context of this thesis can now be drafted. Figure 1.6 shows a QR starting at initial position x_0, y_0, z_0 , tar-

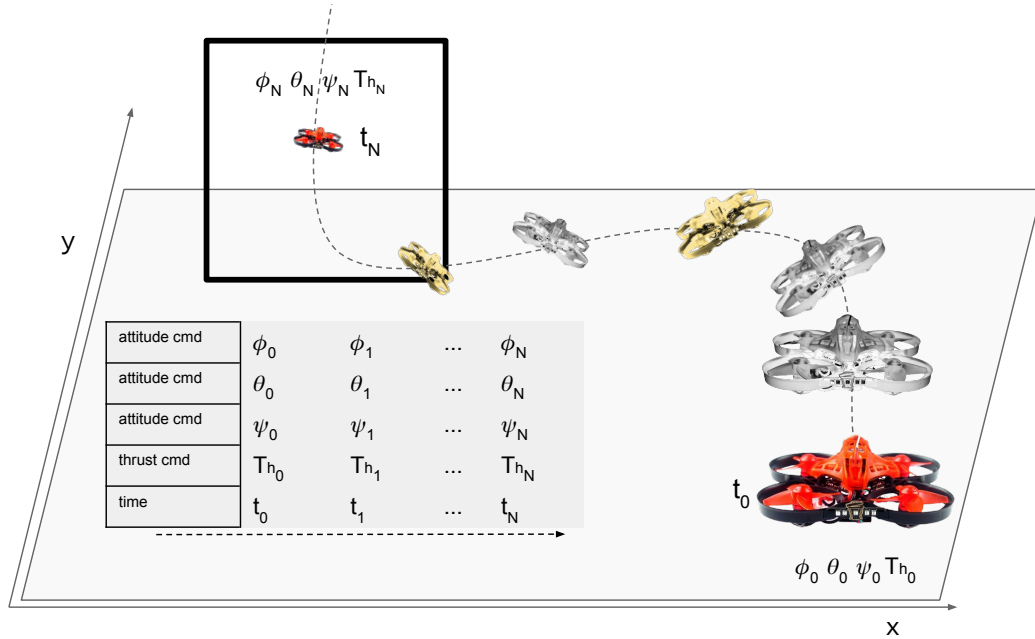


Figure 1.6: Objectives of the thesis - prediction and optimization.

getting a gate located at x_d, y_d, z_d . The QR must fly through this gate, irrespective of its initial condition (irrespective of its initial position or velocities in the arena). The QR must finish the entire track by making a series of such gate crossing maneuvers.

The QR must accurately predict its states at each time instant and simultaneously optimize for the best set of control inputs to make the QR reach the desired point in minimum time. The figure highlights two sub-problems:

⁵flight capability can be determined by the thrust to weight ratio, described further in Section A.1

⁶An open source autopilot, widely used in MAVLab, located on <https://github.com/paparazzi/paparazzi>

1. *The state estimation problem.* The **path** (denoted by black dotted lines) and the **pose** (denoted by gray shadows) are the positional and attitude predictions of the QR. These predictions are generated using the inputs from the Table in Figure 1.6 after they are given to the QR's mathematical model. These predictions must match the actual trajectory that is followed by the QR in real world.
2. *The optimal control problem.* Commands must be pre-calculated to not only make the QR at x_0, y_0, z_0 reach the point x_d, y_d, z_d , but also to make it do so in minimum-time. The table inside Figure 1.6 holds attitude and thrust commands that need to be applied to fly through the gate per time instant.

Quadrotor flight must hold accurate state estimates of all of its dynamical states while the flight is guided by time-optimal control maneuvers to reach its desired state. After doing so, the problem statement of the thesis can be addressed as follows:

Can the control module of an autonomous quadrotor flight be designed in such a way that it can beat human pilots by solving for unique time-optimal maneuvers?

OR

Develop a robust state-estimation framework that minimizes drift over time while reducing expectations of frequent corrections from the vision sub-module, and design an optimal control method that allows a QR to perform time-optimal maneuvers while being computationally inexpensive.

1.7. Motivation

The primary motivation of this thesis lies in bridging the theoretical field of linear algebra with the practical field of embedded systems. Pushing embedded systems to their limits while running rigorous real-time mathematical optimizations to fly quadrotors efficiently could provide a sense of satisfaction to the frugal community of embedded systems. It could disprove the axiom of demanding higher specifications of hardware everytime the task seems computationally heavy.

Secondly, should the objectives of this thesis be well met, they could be a possible solution towards yielding crash-free, minimum-time and energy efficient trajectories. These algorithms could be a possible contribution to our drone race team. This team definitely deserves an introduction - originating from the Micro Aerial Vehicle Lab (MAVLab⁷) of the Aerospace faculty, this team comprises of professors (also the supervisors of this thesis), PhD students and Master students. This group participates in drone races held all around the world and is currently participating in the famous AlphaPilot challenge hosted by Drone Racing League and Lockheed Martin. The team has had consecutive podium finishes in previously hosted IMAV and IROS drone race competitions. The philosophy followed by MAVLab is to fly fast using simple but robust vision and control algorithms while trying to sketch out time-optimal trajectories. Similar to the MAVLab team, numerous teams throughout the world participate in drone racing [9], revealing the recent interest in the field.

1.8. Organization

Chapter 2 is a supplementary introduction chapter that explains the symbols and conventions used throughout the report. It also gives a definition of state estimation and describes dynamics that are valid for QRs. It concludes by listing out the difficulties associated with state estimation and control in QRs.

Chapter 3 begins by describing forces that act on the body of QRs. It later lists and compares different types of state estimators that have been discussed prior in literature. After finding a scope of improvement, a proposal for a new estimator is made. Convergence metrics of different estimators are derived by performing two distinct maneuvers on the QR. It is shown that the proposed estimator has better convergence as compared to the prior ones in literature.

After deriving the tools for performing state estimation, Chapter 4 uses them for QR control. It follows the same structure as that of Chapter 3. It first introduces existing control techniques for QR control and then proposes a new optimal control technique. Chapter 4 compares the new candidate controller with existing controllers under different flight conditions. It does so in three ways. Firstly, by checking if the trajectory shape was met (enter the gate from the front panel with particular velocity), secondly by checking the time of

⁷More information about the lab can be found here: <http://mavlab.tudelft.nl/>

completion and thirdly by benchmarking the computation time of different control methods. It is seen that the proposed control method beats all the algorithms and finds its fit in the drone-race pipeline.

As a result, the prior work in state estimation and control is introduced in their own respective chapters. This makes it easier to compare the existing and proposed methods for QR estimation and control. The conclusion chapter verifies if the requirements stated in the introduction chapter are met. It gives recommendations to carry forward some ideas that remain unexplored in this thesis. It also discusses that a few ideas that are explored on QRs can also be transferred to other areas of research.

Various algorithms were explored and proposed during the thesis⁸. The workflow followed was to first semantically draft the algorithm depending on the requirements of the control modules. These algorithms were then typed out and plotted using MATLAB scripts. To prevent damaging drones at the lab, tests of these developed algorithms were made on MIT Flight-Goggles [9] ROS simulator⁹. Finally, if the algorithm matched the real-time constraints set by flight controllers of a real drone, they were ported to fly inside the CyberZoo of TU Delft.

⁸Our algorithms are open sourced at: <https://github.com/nilay994/superstate.git> and previewed at: <https://www.youtube.com/watch?v=gK487JaIZvo>

⁹FlightGoggles is sourced at: <https://github.com/mit-fast/FlightGoggles>

2

Introduction to Quadrotor Dynamics

After narrowing down the problem statement to the control module of racing-drones, this chapter takes a step further to split the control module into its sub-modules of state-estimation and control. This chapter also acts as a supplementary introduction chapter for presenting the symbols and conventions used along the entire span of this report. These conventions are important for both sub-modules of state estimation and control. After an introduction to QR conventions, the selection for a modeling technique for performing state estimation is made. This modeling of QR dynamics forms the backbone of state estimation.

2.1. A brief introduction to the area of state estimation

State estimation is a vast area under the control theory domain. It is defined as a mechanism that is able to provide an internal state of a given dynamical system. For the scope of this thesis report, state estimation would aim to give information about the positional state of the QR system.

In general, the estimated states for different systems could range from something tangible, for instance - the physical entities of positions, derivatives of positions, moments of inertia to something very abstract - the drag co-efficients of a satellite orbiting Earth. In network theory, states of a server system could be indicative of the traffic in the network or be information about the probability of packet drops. Shreve [29], well known for his contributions in the area of Brownian motion, explains strategies on portfolio optimization in financial market theory using stochastic theories. Shreve's algorithms can hold the portfolio value and states of market-risk that could be useful for making investment choices. Similarly, this thesis makes the QR hold information about the dynamic states that are used for making control decisions. After the states are estimated, the QR can make decisions about the control signals to follow trajectories. To expand more on the examples listed above, the end of this section encloses a couple of interesting examples on state estimation. With regard to the control systems based nature of this thesis, it is necessary to explain the area of state estimation further. It can be broken into two parts of modeling and filtering. Figure 2.1 explains the bifurcation of the control module from the place where the discussion of the scope of this thesis was concluded.

1. **Modeling:** The modeling phase involves listing down the forces in the free-body diagrams of the QR. These models are then represented using differential equations. Differential variables and their derivatives can be simply integrated in time to get the information about the state of a system. Control inputs to the system if known can also be accounted for in these differential equations. It can be noted that since the dynamical equations of the system are not an exact representation of the system, the estimates of the internal states will be erroneous.
2. **Filtering:** The filtering phase of state estimation takes over in such circumstances, correcting the erroneous estimates propagated by the differential equations of the modeling phase. This phase can be asynchronous, only seldom correcting the estimated states. This is also called as the measurement phase since sensor measurements are required to facilitate the filtering stage of state estimation. However, this, not being the main aim of this thesis, is explained only later in Section 5.2.
3. **Optimal state estimators:** In some special cases, there exists a theoretical optimum that yields unbiased and low variance estimates. This special case only exists in Linear Time Invariant systems with

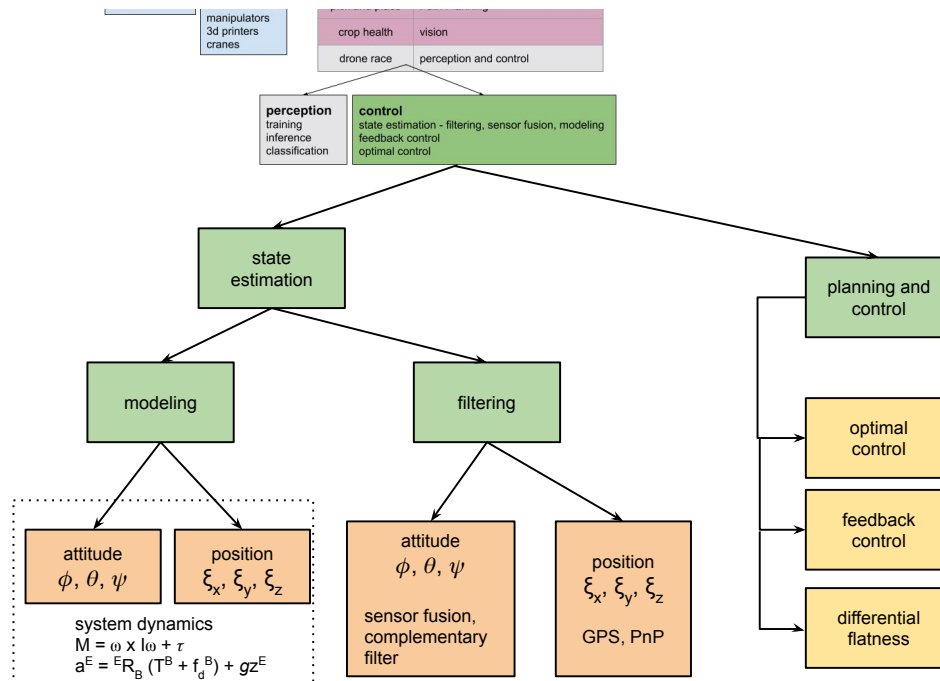


Figure 2.1: Control module of drone-race pipeline expanded.

linear measurement models. This type of state estimation is famously referred to as Kalman filtering in theory. If either process models or measurement models are non-linear, Extended Kalman filters and Unscented Kalman filters can be used for recursive parameter estimation.

Type of states to be estimated: Since race drones are supposed to fly through waypoints, a position controller must be used to track these commands. The controller would expect two inputs (a) the trajectory to be followed and (b) QR's relative position and velocity with respect to it. Hence estimations techniques for lateral positional states x^E, y^E, z^E of the QR are discussed in detail in this thesis.

Examples of state estimation: Cars find it difficult to perform positional state estimation inside tunnels

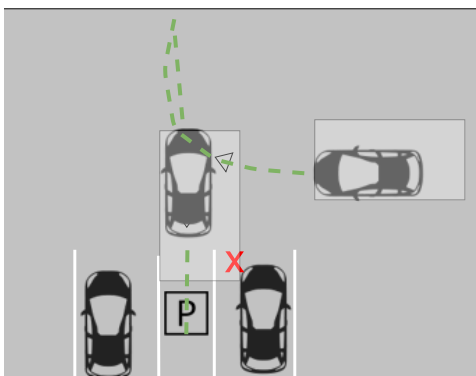


Figure 2.2: Possible collision inside parking spaces due to incorrect state estimation.

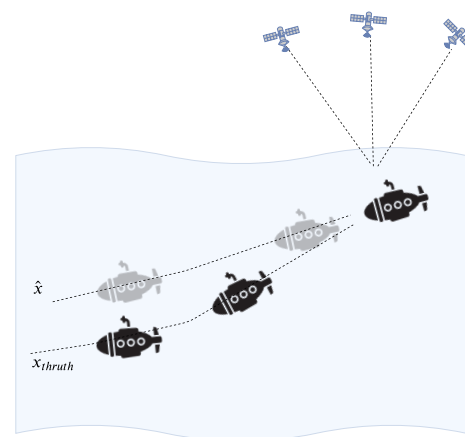


Figure 2.3: Submarines must come closer to the surface for correcting their drifting estimates.

and parking lots since GPS measurements are rather incorrect. In absence of accurate positional measurements, the state estimators of the car must completely rely on its dynamical model. However, using these coarse models it is difficult to drive the co-variance of the propagated states to lower values. Hence, in the absence of measurements from sensors, an accurate model of the dynamics of the vehicle is the only way to maintain accurate estimates of the states. A failure to match the requirement of an accurate model could im-

ply that the state estimates have drifted from the real world, and the vehicle now possesses incorrect values of its position. This is dangerous especially in a constrained configuration space like in Figure 2.2.

Submarines face a similar complication while spending long durations of time and distance underwater. Since GPS signals attenuate to a great extent at the depths of a submarine’s cruise level, GPS receivers cannot be used to triangulate the position of a submarine. In absence of GPS measurements, there are other methods to keep the state covariance low by using SONARs or by measuring underwater currents. Although the measurements from these sensors are unable to drive the co-variance of the state estimates to as low as a GPS measurement could have driven it. In such a case, the presence of precise dynamical models are essential for ensuring that the estimates in states drift less with respect to time (Figure 2.3).

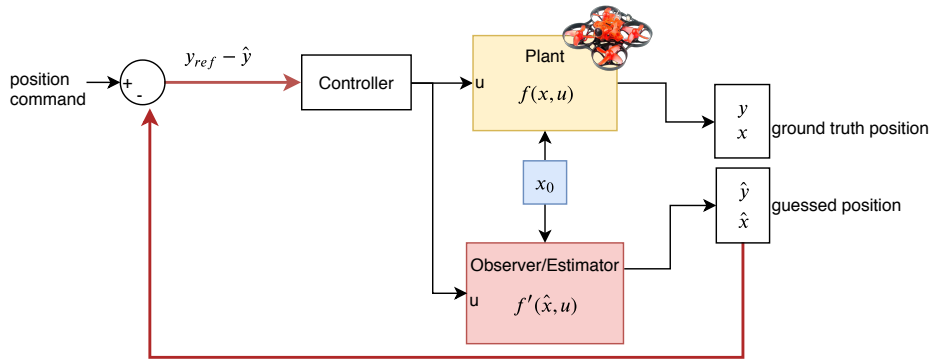


Figure 2.4: Observer or state estimator based feedback control.

When it comes to lateral positional states of autonomous QRs, it is essential to have low covariance state estimates for generating appropriate control signals while performing navigation. The reason for maintaining good state estimates for generating appropriate control signals can be explained by Figure 2.4. If the observer (the model in red) is not representative of the real world model, the estimates (\hat{x}) and outputs generated (\hat{y}) drift away with respect to x, y as time progresses. This drift leads to incorrect signals being sent to the controller ($y_{ref} - \hat{y}$). Hence, it is critical for QRs to have accurate observers/state estimators for flying correctly. In this manner, this chapter aims to make the the concept of state estimation irrevocable/mandatory for performing QR control.

2.2. Quadrotor dynamics

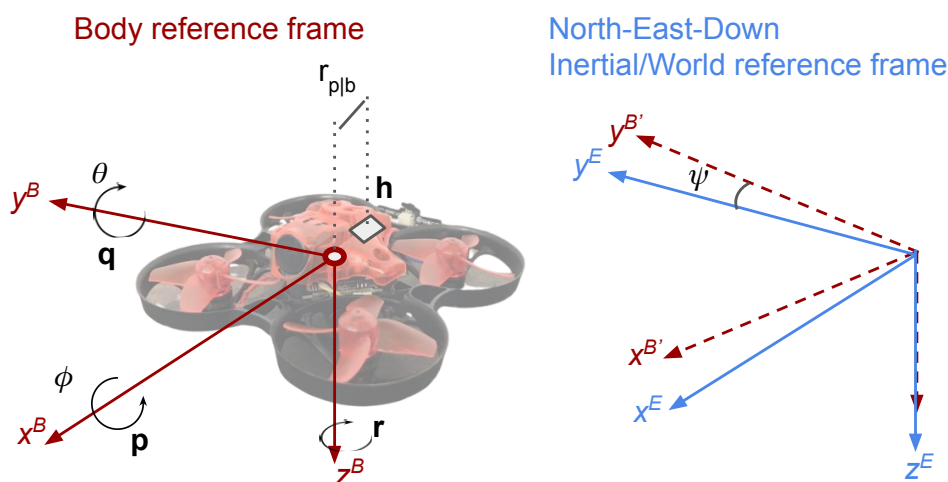


Figure 2.5: Axes and frame of reference conventions used in this report

The convention followed in this thesis for axes representations and co-ordinate frames is given in Figure 2.5. The world frame ‘E’ (also referred to as the inertial frame of reference) follows a right handed North-

East-Down convention. As a result, the higher a QR climbs, its position in the world frame becomes more negative. The IMU and body frame '**B**' also use a right hand reference system, z^b facing downwards, usually yielding negative values of gravitational acceleration on the z-axis of accelerometers when QR is in hover conditions. The frame '**B**' is an intermediate frame, which is derived after yawing the world frame of reference. This is done for purpose of alignment with the heading of the QR and for mapping correct control actions while navigating.

- p, q, r : angular velocity of the body of the QR.
- ϕ, θ, ψ : roll, pitch and yaw of the QR.
- $\dot{\phi}, \dot{\theta}, \dot{\psi}$: angular velocity of the QR with respect to the world frame.

Using these conventions, the QR must have a negative pitch to go forward. As the idea of "*All ways lead to Rome*" suggests, [5] discusses various methods for modeling of a QR. Different set of dynamical equations that are derived after following different modeling approaches, which are in the listing below, describe the same model of the QR.

1. The Euler Lagrangian approach gives a QR model after using kinetic and potential energy that a QR's body is subjected to. For performing positional state estimation of a QR in the world frame, it is not intuitive to estimate the energy acting on a QR. This model however, is widely used in outdoor flights of fixed-wing aircrafts.
2. The Newton Euler approach uses free body diagrams to estimate the forces acting on the model. This approach decouples the moment (rotational) and force (translational) model. It also provides an intuitive way of augmenting external un-balanced and un-estimated forces acting on a rigid body. Figure 2.6 replaces a QR with a rigid body which experiences two significant forces of drag and thrust (vectors in blue). The estimation of specific forces can give information about the instantaneous acceleration of the body. Specific force is a non-gravitational force acting on the body of a QR. In case of QRs, specific forces are mainly composed of the thrust produced by the propellers of the QR and drag forces faced by the body of the QR. Instantaneous accelerations can be used to infer the change in velocity and hence also the displacement of the rigid body. This modeling approach is used in this thesis because of two significant benefits: (a) Simplicity due to decoupled dynamics in the rotational and translational space. (b) ability of performing positional state estimation by measuring specific forces on the body of the QR.
3. The dual-quaternion approach for QR modeling is receiving some attention lately [17] [25]. The approach not only combines the rotational and translational model, but also represents the entire pose of an object in 3D space (position and orientation) into a single entity. For example, the pose of the head of a screw being drilled in, can now be represented as a single entity. Dual quaternions could potentially find their direct applications in powered descents made by SpaceX's FALCON rockets and Martian/Lunar landers [17]. It could also find applications on satellite attitude control and orbit raising algorithms. If QRs are modeled using this approach, they can mimic these satellites and landers by serving as testing platforms.

2.3. The Newton-Euler modeling approach

The model describing the dynamics of a QR evolving in \mathbb{R}^3 and SO^3 (special orthogonal group) space, which is subjected to non conservative forces $F \in \mathbb{R}^3$ and torques $\tau \in \mathbb{R}^3$ applied to its center of mass, is given by:

$\xi \in \mathbb{R}^3$	position of center of mass in E		
$V \in \mathbb{R}^3$	velocity of center of mass with respect to E		$\dot{\xi}^E = V^E$
${}^E R_B \in SO^3$	rotation matrix to go from B to E		$\dot{V} = a^E$
T^B	thrust in body frame		$a^E = {}^E R_B (T^B + f_d^B) + g z^E$
f_d^B	disturbance due to drag, etc.		$\dot{R} = R \hat{\omega}$
ω	angular velocity in the world frame		$I \dot{\omega} = -\omega \times I \omega + \tau$
I	inertia tensor		

The above equations are derived using the free body diagram given in Figure 2.6. The force and moments originating from the rotors (red entities in Equation 2.1) can be described further:

Blade element theory suggests that a blade rotating with a particular RPM generates a force and torque on the body of the QR. Summing up the forces produced by each propeller yields the net thrust generated by the QR, while a combined effect of moments generated by them provide angular motion in the QR. For a rotor with angular velocity ω , and radius r ,

$$\begin{aligned} T_B &= \sum_{i=1}^4 f_i \text{ where, } f_i = C_{Ti} \rho A_p r^2 \omega_i^2 \\ \tau &= \sum_{i=1}^4 \tau_i \text{ where, } \tau_i = C_{Qi} \rho A_p r^3 \omega_i^2 \end{aligned} \quad (2.2)$$

where C_{Ti} and C_{Qi} (dimensionless) are the thrust and moment co-efficients of the rotor blade. f_i is the steady-state thrust generated by a propeller that is not translating horizontally or vertically through space. When the propeller moves through space, the thrust produced by them at the particular rotor speed changes with respect to the thrust generated due to same rotor speed while hovering. The model that takes into account the dynamical factors of a blade in translation is explored in Section 3.2.

It is important to note that the conventions used in Equation 2.1 and Figure 2.6 are non trivial. They require correct intuitions of source of drag, Coriolis and centrifugal forces. The forces estimated, must also be compensated for in the correct frame of reference. Failing to take precautions while carrying out this math has led to incorrect equations to be published. Some corrections are found in Appendix A of differential flatness derivations in [21]. In our case, two allowable assumptions are made to avoid augmenting fictitious forces on the frame of the body of the QR. (1) The QR is always considered as a point mass. All points on its body are hence represented by a single point that is the center of mass. (2) The IMU is mounted exactly on the center of mass, avoiding any pseudo forces that may act on the accelerometer sensor.

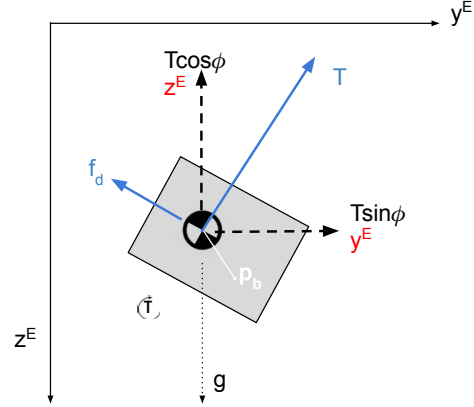


Figure 2.6: Free body diagram of the quadrotor subject to aerodynamic drag.

2.4. Frames of reference and Rotation matrices

As described earlier, there are two frames of reference, the inertial frame E and the body frame B . Rotation matrices can convert entities like lateral positions, velocities, accelerations from one frame of reference to the other. Note the two rotation matrices listed in Equation 2.6. Euler angles are used to denote the orientation of one frame with respect to the other. Depending on the sequence of rotations between the frame of reference, these representations are classified further. The ‘Proper Euler angles’ conventions involve sequential rotations, which means that the entities listed are the angles between each **consecutive** rotation. They usually follow the $(z-y-z)$ sequence of rotation. The Tait-Bryan representation (which is used in most aerospace applications) is used in our case. In this case, angles indicate orientation with respect to the original frame of reference and **not** with respect to each sequential rotation. The angles ϕ, θ, ψ in Figure 2.5 can be referred to as the angles in Tait Bryan representation. The processed readings after filtering the IMU data, return instantaneous angles with respect to the original frame of reference and not angles with respect to an intermediate rotational frame of reference. Hence, the Tait Bryan representation is used throughout the thesis to represent the orientation of the QR. Rotation matrices can hence be given as:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix}, R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}, R_z(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

Where ϕ, θ, ψ are the roll, pitch and yaw of the QR with respect to the world frame of reference. The rotation matrix generated after following the r-p-y or x-y-z sequence can be given as:

$$R = R_z(\psi)R_y(\theta)R_x(\phi) \quad (2.4)$$

This gives the complete rotation matrix to go from body entities to world entities as:

$$R = \begin{bmatrix} \cos\theta \cos\psi & \sin\phi \sin\theta \cos\psi - \cos\phi \sin\psi & \cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi \\ \cos\theta \sin\psi & \sin\phi \sin\theta \sin\psi + \cos\phi \cos\psi & \cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi \\ -\sin\theta & \sin\phi \cos\theta & \cos\phi \cos\theta \end{bmatrix} \quad (2.5)$$

The transpose of this rotation matrix can perform transformations from world entities to body entities.

$${}^W R_B = \underbrace{\begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix}}_{\text{body to world rotation matrix}} \quad {}^B R_W = \underbrace{\begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & S_\phi C_\theta \\ C_\phi S_\theta C_\psi + S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\phi C_\theta \end{bmatrix}}_{\text{world to body rotation matrix}} \quad (2.6)$$

The rotation matrix to go from angular velocities in the body frame to angular velocities in the world frame can be given by:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi \tan\theta & \cos\phi \tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (2.7)$$

2.5. Difficulties in control and estimation for QRs

Six degrees of freedom of QRs allow them to hold any pose¹ in the spatial \mathbb{R}^3 (x,y,z) and rotation S^3 ($R(\phi, \theta, \psi)$) space. However, to autonomously maneuver towards a particular pose from a different starting pose, two conditions from control systems should be met - (1) observability and (2) controllability of the positional states. In this thesis, the explanation of these concepts is not done meticulously, however it helps in illustrating the impending problems in the field of QR control.

Observability checks if the physical states of the QR can be read by the onboard computer. Reachability or controllability checks if the desired pose configuration can be directly achieved by sending commands to the actuators. In case of simple QRs that possess four propellers and a single IMU, the criteria for full rank of controllability and observability matrices are not met. To explain why observability and controllability are important for the scope of this thesis, a listing is made below explaining them briefly.

A continuous time linear system is considered while explaining the concepts below:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned}$$

- x: Continuous time states of the dynamical system. Positional and rotational states in this case.
- A: State propagation matrix. System dynamics make up the A matrix.
- B: Input matrix. The control inputs are fed into the system via this matrix.
- C: Output matrix. Selects states that are observable.
- D: Feedforward matrix: The outputs can also be a result of the inputs directly. The matrix D decides the feedforward mapping to the outputs from the inputs.

2.5.1. Observability

1. For being able to observe all states in \mathbb{R}^3 and S^3 , the observability criteria of control systems should be met. A system is said to be observable if the states $x(t)$ can be fully determined (reconstructed) using measurements $z(t)$ and control inputs $u(t)$ [15]. The observability criteria for a linear system is given by the rank of the matrix $\mathcal{O} = [C, CA, CA^2, \dots, CA^{n-1}]^T$. If the rank of this matrix equals the order of the dynamical system, then it implies that all states of the system can be directly observed. For more details on observability analysis, the paper by Hanley et.al.[10] describes the nonlinear techniques for observability analysis for states of a QR.

¹Except the QR being upside down for a long time.

2. Observability of the dynamic states implies whether or not the sensors (e.g. IMU) can read from the dynamical pose information of the QR (both attitude and positional states). Partial observability of only a couple of states could imply that estimators for the remaining states must be designed by the user.
3. 6-DOF IMUs can only directly observe the banking angles (ϕ, θ) and angular velocity (p, q, r) of the QR in the S^3 plane². This is after assuming that the AHRS (Attitude Heading Reference System) has already performed sensor fusion and filtering for estimation of angles in SO^3 .
4. For the remaining positional states in \mathbb{R}^3 , observers must be manually designed. The design of these observers is highlighted in Chapter 3.
5. As a result, it can be said that the observability of states in this thesis is limited by the capability of the sensors. The process of observer design, can however be skipped if additional sensors like GPS, Visual Inertial Odometry or Optical Flow sensors are used. Using these sensors could yield higher ranks of the observability matrix, leading to better observability of states in \mathbb{R}^3 .

2.5.2. Reachability

1. For being able to hold any of the states in \mathbb{R}^3 (x, y, z) and S^3 $(R(\phi, \theta, \psi))$, the reachability criteria must be met. A simple reachability analysis for linear systems is done by calculating the rank of the matrix $R = [B \quad AB \quad A^2B \quad \dots \quad A^{n-1}B]$.
2. Reachability of states is closely related to the actuated-ness of vehicles under study. If an actuator is absent in a particular degree of freedom of the vehicle, it can possibly make the state unreachable. This can be observed from Figure 2.5, where there are missing actuators in the lateral ξ_x, ξ_y axes. It is not possible for QRs to move sideways unless they change their attitude by banking.
3. The ability of being able to reach states in the under-actuated degree of freedom while missing actuators is answered by the study of non-holonomic underactuated³ systems. As a result, these systems can still achieve the desired states without implicitly having an actuator in that particular direction.
4. The moment equations (defined in Equation 2.1), indicate that the change in actuator RPM can bring about actuation in all the three axes of rotation.
5. The force equations (defined in Equation 2.1), indicate that there must be a rotation for a translation to take place along ξ_x^E, ξ_y^E .
6. sidenote: An interesting study by Romero et. al [26], uses a set of eight propellers to solve the underactuated problem. Figure 2.7 illustrates a modified QR from their study, which now has actuators along the under-actuated axis. Four additional actuators f_5, f_6, f_7, f_8 are introduced along the lateral under-actuated axes (ξ_x^E, ξ_y^E) that can work in tandem to produce the resultant vector in black.

2.6. Discussion

Due to the standard setup with the actuators (4 propellers) and sensors (one IMU), the axes of an autonomous QR have the following properties:

1. Rotational actuation and estimation is possible about all three axes⁴.
2. Along the lateral z-axis, positional estimation is sometimes made possible if there are additional sensors for height measurement. Also full actuation is possible about this axis by increasing or decreasing the RPM of all the actuators simultaneously.
3. Along lateral positional axes of (ξ_x, ξ_y) neither a direct actuation nor estimation is possible⁵.

Table 2.1 summarizes the difficulties in estimation and control with respect to each axis for a standard QR.

²Since accelerometers can not be of help in the yawing plane, drifts because of the integrated gyroscope measurements must be compensated by visual techniques or by incorporating a compass.

³The study on underactuated systems is used in multiple areas nowadays. Differential flatness is one of the solutions exploited to algebraically assign control actions to underactuated systems depending on the output desired. This now finds its application in robotics, self-driving cars, industrial robots and self-landing rocket booster stages.

⁴This is assuming that the IMU also has magnetometer/vision based corrections for avoiding drifts in yaw.

⁵This is assuming that there are no extra actuators and no GPS/SLAM sensors are present.

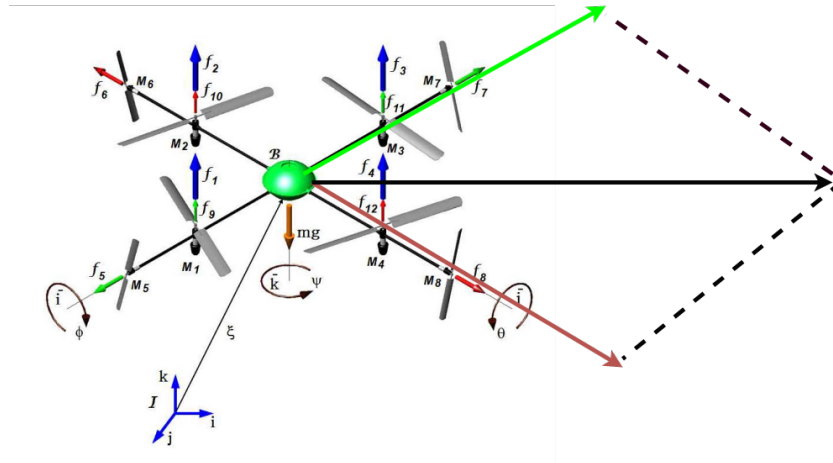


Figure 2.7: The eight-rotor fully-actuated QR [26].

↓Axes Properties ⇒	Directly observable	Directly controllable	Solution for observability
ϕ	✓	✓	estimated by IMU sensor fusion
θ	✓	✓	estimated by IMU sensor fusion
ψ	✗	✓	requires magnetometers or visual aid
Lateral x	✗	✗	requires GPS/VIO/OF or state estimation
Lateral y	✗	✗	requires GPS/VIO/OF or state estimation
Lateral z	✗	✓	requires GPS/VIO/OF or state estimation

Table 2.1: Table of controllability and observability with respect to each axis.

Since QRs must fly through positional setpoints on the arena, estimation and control along lateral positional axes are most critical. However, from the above table it is clear that neither estimation or control is directly possible about these axes. These difficulties are taken up as the problem statement of this thesis. The chapters that follow this supplementary introduction hence discuss the last three rows of Table 2.1. The only way to achieve these objectives in drone-race settings (only IMU is allowable) is by knowing the dynamics of QRs as accurately as possible. The next chapter discusses state estimation using these QR dynamics to make the lateral axes observable.

3

State estimators for Quadrotor Control

It was earlier established that possessing knowledge of the states can increase the quality of flight through the gates. Aristotle's saying "*Knowing yourself is the beginning of all wisdom!*", if extrapolated to our case, hints on the importance of possessing the estimates of all the states of the QR as accurately as possible. This would include estimation of the states of positions, orientations and their derivatives $[(X = \xi_x, \xi_y, \xi_z, \phi, \theta, \psi), \dot{X}, \ddot{X}]$. Table 2.1 establishes that the estimation and control of the rotational states ϕ, θ, ψ is not a pressing issue anymore. As a result, this chapter focuses on the estimation of the positional states.

The prior chapter highlights that any state estimation techniques consists of two sub-modules (a) modeling and (b) filtering¹. Only the first sub-module of modeling is discussed in this chapter. The filtering module is not a primary goal of this thesis and hence discussed only in the appendix within Section 5.2. Performing accurate modeling also satisfies the first objective of this thesis (as stated earlier in Figure 1.6), which is about accurate propagation of the estimated states. Precise models established by this chapter can later be used to assign accurate control actions to the actuators for navigating through the gates of the drone race arena (discussed in Chapter 4).

Figure 3.1 clarifies the scope of this chapter. To explain the figure further- QR dynamics can be split into lateral and rotational dynamics. It has been highlighted earlier in Table 2.1 that stable rotational dynamics is already a solved problem in QR control (indicated by purple boxes in the figure). The figure indicates that the chapter focuses only on the estimation of the lateral states of QR under the block of translational dynamics (box in green).

Organization of the chapter: This chapter is divided into three parts. The first part explains the two significant forces of thrust produced by the propellers and drag effects that are induced in forward flights. It concludes with the identification of drag coefficients which are helpful for designing a state estimator.

The second part of the chapter discusses prior work in the area of modeling, where each paper has included different terms in order to improve accuracy of the model. After discussing the pros and cons of models currently in use, the requirements for a new estimator are discussed that could fit well in the framework of drone-racing. After these requirements of a new state estimator are established, a proposal for a new state estimator is made. Detailed models used by the new estimator can assist in accurate lateral-state estimation of the QR.

The third part devises experiments that emulate different drone racing maneuvers to compare the new

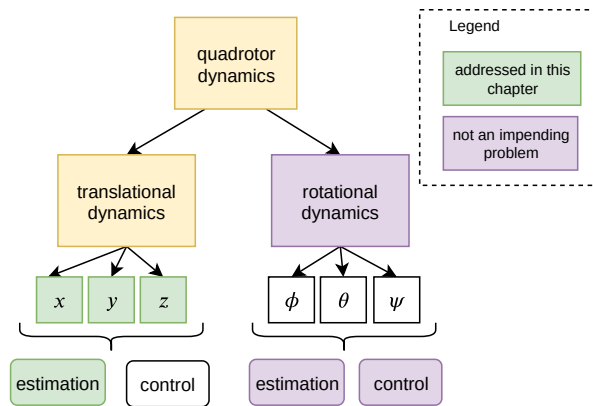


Figure 3.1: Chapter 3 only discusses translational dynamics along x^B, y^B, z^B (boxes in green).

¹In control systems jargon, the steps performed for modeling and filtering are similar to the prediction propagation and measurement update step of state estimation.

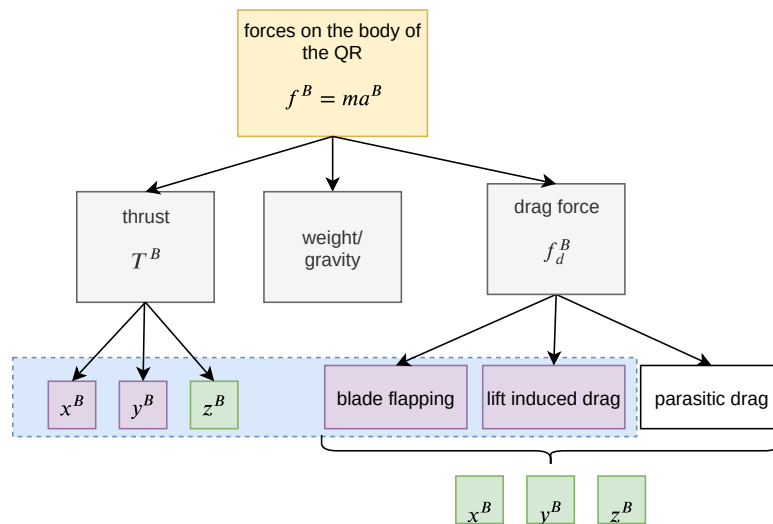


Figure 3.2: classification of types of forces acting on the body of the QR.

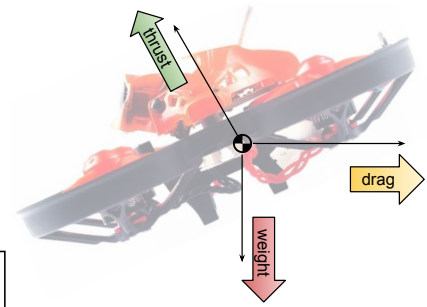


Figure 3.3: forces on the body of the QR.

estimator with prior estimators proposed in literature using some key-metrics. The chapter concludes by achieving the aim mentioned in the green box of Figure 3.1.

3.1. Positional state estimation

As Newton's laws dictate, for the QR to change motion through space, external unbalanced forces must act on the body of the QR. It is essential to find out these unbalanced forces acting on the body frame of the QR for estimating positions of the QR. A change in the velocity of the vehicle can be inferred from the integration of these unbalanced forces. Displacement of the vehicle can be observed after integrating these obtained instantaneous velocities of the vehicle at each time instance. The displacement can be added to the initial (known) position of the QR, to return its absolute position in the inertial frame of reference. As a result, to perform positional state estimation, estimating the forces on the frame of the body of the QR is critical.

It is possible to use the accelerometer for reading the values of the above mentioned 'external unbalanced forces'. However this is a rather incorrect method and cannot be used in the drone-race framework. A detailed explanation for the reasons to this are listed in the appendix within Section A.3. To go back to the rather correct method of positional state estimation, the forces on the body of the QR must be studied. When flying indoors in the absence of wind, the QR's body is subjected to two significant sources of external unbalanced forces:

1. **Thrust:** generated by each propeller of the QR. For a normal configuration, thrust only acts along the z^B axis of the QR as shown by the leftmost green box in Figure 3.2.
2. **Drag:** an opposing force on the body of the QR as a result of motion of the QR. Drag forces originate due to multiple reasons and the major contributors of these forces are classified in Figure 3.2. Drag forces offer their contribution along all the axes of the body frame (green boxes in Figure 3.2).

The boxes in purple indicate that there is some cross coupling between drag and thrust forces. The cross coupling arises due to changing direction of resultant thrust vector generated from the body of the QR. This resultant vector is a due to the generated thrust and drag (combined) and will be explained in sections to follow.

3.2. Thrust

Thrust is the most significant contributor of the forces that are induced on the body of the QR. In hover positions, thrust is the only contributor of forces induced on the body of the QR ($T = -m \times 9.81 m/s^2$). However, this is not the case in forward flight and other non-hover regimes. To study the thrust in these non-hover regimes, different thrust models have been proposed, which give a close representation of the actual thrust that might be produced at different angles of attacks and lateral velocities.

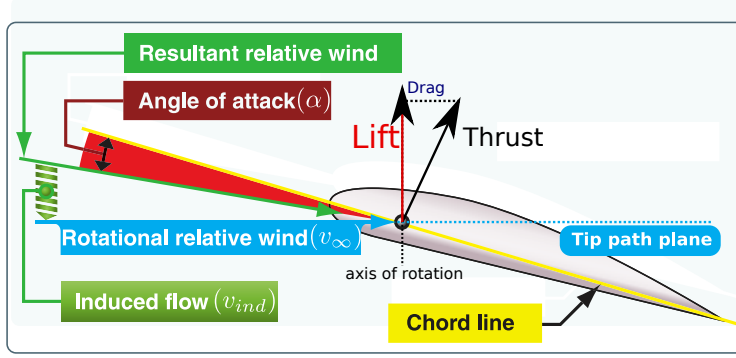


Figure 3.4: Angle of Attack and Free-stream velocity [1]

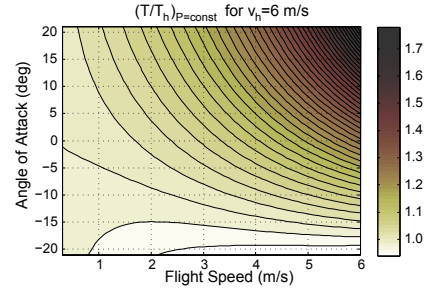


Figure 3.5: The ratio of thrust generated with respect to hover thrust at different resultant relative wind and angle of attacks [14]

A coarse thrust model usually fails to capture the coupled dynamics that arise when the QR is in non-hover regimes. A fine model requires a higher number of terms to represent the net thrust generated, requiring rigorous system identification procedures to estimate them. In such a case, numerous parameters could be possibly added, for e.g. density of air, pitch angles of the rotor and battery state of charge. Subsections below list the three most commonly used thrust models that were proposed earlier in literature. The thrust section concludes with a discussion on picking one of the three thrust models that could be suitable for the drone-race framework.

Having accurate thrust models improves the lateral state estimates in the $x^E - y^E$ plane, although having accurate thrust models is most critical for the estimates in the z^E axis. The z^B and z^E axes are usually aligned (in near hover states) and the major component of forces in the body frame (thrust) is along z^B . As a result, thrust has the highest contribution of forces in the z^E axis. As a result, having a good estimation of thrust can provide the QR with good altitude estimates.

The assumption made while discussing different methods to model thrust is that the QR flight is always ensured to be in normal flying regimes and no vortex ring states are induced. Vortex ring states are discussed at the end of this section using Figure 3.9.

3.2.1. Induced velocity method

Blade element theory suggests that the force generated by a rotating airfoil can be given by the force-rpm equation below. As earlier stated in Equation 2.2, summing the forces produced by each propeller yields the net thrust generated by the QR:

$$T^B = \sum_{i=1}^4 f_i \text{ where, } f_i = C_{Ti} \rho A_p r^2 \omega_i^2$$

However, this expression for f_i holds only when the QR is hovering. Blade element theory proposes different equations to give an accurate thrust in non-hover regimes. During forward flight, an increase in the incident free-stream velocity seen by the propellers increases the effective thrust produced by the blades. The free-stream velocity is essentially the velocity of wind with respect to the body of the QR and is represented by v_∞ . For finding accurate values of thrust generated by the propellers in the non-hover regime, [14] suggests that the induced flow through each blade of a propeller must be analyzed. Relevant equations from the paper are re-stated here and the pros and cons associated with the induced velocity method are highlighted here. A few terms must be introduced before stating the equations used in [14].

- V_{ind} : Induced velocity respect to the airfoil as shown in green.
- V_∞ : Free-stream air velocity given by $\sqrt{V_x^E{}^2 + V_y^E{}^2}$ (assuming wind velocity is zero).
- α : Angle of attack: Angle between the airfoil chord line and the resultant relative wind as indicated in red.
- T_h : Thrust required to maintain hover-state. For this case, $T_h = -mg$
- V_h : Lateral velocity of the QR represented in body frame, is given by $\sqrt{V_x^B{}^2 + V_y^B{}^2}$.

$$V_{ind} = \frac{V_h^2}{\sqrt{(V_\infty \cos \alpha)^2 + (V_\infty \sin \alpha + V_{ind})^2}}, \quad V_h = \sqrt{\frac{T_h}{2\rho A}}, \quad \text{and the thrust ratio is given by, } \frac{T}{T_h} = \frac{V_h}{V_{ind} + V_\infty \sin \alpha} \quad (3.1)$$

It can be noted that the above expression² for V_{ind} is quartic in nature. The sidebar in Figure 3.5 indicates the ratio of increase in thrust observed at non-hover positions. This sidebar is generated using the quartic expression above. From Figure 3.5, an observation can be made that at higher lateral velocities, thrust becomes highly sensitive to the angle of attack at the blades [14].

To be able to select appropriate thrust ratios at different induced flows, the above quartic polynomial must be solved in real time at high-frequencies onboard the flight-controllers. Algebraic quartic solvers or Newton's iterative methods are required to be ported for solving this equation. After the roots to the quartic polynomials are found, the resulting induced flow can be used for selecting appropriate thrust ratios. Selection of correct thrust ratios can help the QR maintain a constant altitude. Because of these complicated root-finding methods, [14] recommends using a look-up table to match the required thrust to maintain altitude at non-hover conditions. A look-up table can be setup by discretization of the 2-D space in Figure 3.5. However, due to questionable real-time implementation of solving for induced velocities and thrust ratios, another interesting approach can be explored.

3.2.2. Hover-thrust/constant-altitude method

Most models in literature exploit the hover-thrust method [19] when change in altitude is not desired/expected. The thrust produced is almost enough for the QR to maintain stable altitude when flying through the arena. Instead of induced free-stream air velocities and quartic polynomials, this model gives the net thrust generated by all the propellers to be given by:

$$T^B = \frac{-g}{\cos \theta \cos \phi} \quad (3.2)$$

Another advantage of using this model is that it does not require instantaneous rotor RPMs, angle of attacks or vehicle velocity reported back from the flight controllers. The disadvantage of this model is significant. The model enforces QR to maintain a constant altitude while off-loading the responsibility of controlling the altitude via height based sensors. It also does not include effects of increased blade effectiveness at higher lateral velocities, causing the QR to increase the altitude while banking. Dead-reckoning while using this model is only acceptable in short-horizons depending on the frequency of altitude changing maneuvers.

3.2.3. Thrust model augmented with drag and effectiveness based terms

Given the bounds of the flight envelope and ignoring vortex states, [31] simplifies the thrust described in blade-element theory under normal flight-regimes. After simplifying the equations established by blade element theory, [31] simplifies the thrust equation from

$$T_i = c_1 \omega_i^2 \left(c_2 \left(1 + \frac{3}{2} \mu_i^2 \right) - \lambda_i \right), \quad \text{where,} \quad \begin{aligned} \text{advance ratio: } \mu_i &= \frac{V_{hi} + v_{hi}}{r \omega_i} \\ \text{inflow ratio: } \lambda_i &= \frac{V_{zi} + v_{zi}}{r \omega_i} \end{aligned}$$

to

$$T^B = k_\omega \omega_s^2 - k_z V_{zi} \omega_s + k_h V_{hi}^2$$

- v_z^B - ascent/descent velocity along the z^B axis of the QR.
- $\omega_s = \frac{\sum_{i=1}^4 \omega_i}{4}$: average propeller speed in RPM returned by paparazzi's telemetry modules.
- c_1, c_2 : Dimensionless rpm to thrust co-efficients.

T^B can be estimated from synchronized and timestamped data from OptiTrack and Paparazzi.

$$T^B = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} B \\ \mathbf{R}_E \end{bmatrix}}_{\text{pprz attitude}} \underbrace{\begin{bmatrix} a_x^E & a_y^E & -9.81 + a_z^E \end{bmatrix}^T}_{\text{optiTrack}}$$

²It can be noted that lateral velocities and angle of attack information required by Equation 3.1 can be estimated with some accuracy using state estimation.

Entities in **fuchsia** are considered known and can be estimated from the OptiTrack and Paparazzi setup. The hidden catch in the above model is that the thrust is represented as a linear combination of known and unknown entities. Thrust can be written in a matrix-vector representation as the equation below:

$$T = \begin{bmatrix} \omega_s^2 & V_{zi}\omega_s & V_{hi}^2 \end{bmatrix} \begin{bmatrix} k_\omega \\ k_z \\ k_h \end{bmatrix}$$

A flight identification experiment exciting the lateral velocities V_z and V_h observations over time, yields the following observation matrix-vectors. Its pseudo-inverse can give the parameters to be estimated:

$$\begin{bmatrix} T|_{t0} \\ T|_{t1} \\ \vdots \\ T|_{tn} \end{bmatrix} = \begin{bmatrix} \omega_s^2|_{t0} & (V_{zi} \times \omega_s)|_{t0} & V_{hi}^2|_{t0} \\ \omega_s^2|_{t1} & (V_{zi} \times \omega_s)|_{t1} & V_{hi}^2|_{t1} \\ \vdots & \vdots & \vdots \\ \omega_s^2|_{tn} & (V_{zi} \times \omega_s)|_{tn} & V_{hi}^2|_{tn} \end{bmatrix} \begin{bmatrix} k_\omega \\ k_z \\ k_h \end{bmatrix}, \quad \begin{bmatrix} \hat{k}_\omega \\ \hat{k}_z \\ \hat{k}_h \end{bmatrix} = \begin{bmatrix} \omega_s^2|_{t0} & (V_{zi} \times \omega_s)|_{t0} & V_{hi}^2|_{t0} \\ \omega_s^2|_{t1} & (V_{zi} \times \omega_s)|_{t1} & V_{hi}^2|_{t1} \\ \vdots & \vdots & \vdots \\ \omega_s^2|_{tn} & (V_{zi} \times \omega_s)|_{tn} & V_{hi}^2|_{tn} \end{bmatrix}^\dagger \begin{bmatrix} T|_{t0} \\ T|_{t1} \\ \vdots \\ T|_{tn} \end{bmatrix}$$

$$\hat{T}^B = \hat{k}_\omega \omega_s^2 - \hat{k}_z V_z \omega_s + \hat{k}_h V_{hi}^2 \quad (3.3)$$

The parameters that are identified by the experiment are tabulated in Table 3.1. These parameters are used to estimate/predict the instantaneous thrust force produced at that instantaneous rotor speed and lateral velocities. This prediction of thrust is illustrated in Figure 3.6.

parameter	value
k_ω	-0.00001438 $m/s^2 RPM^2$
k_z	0.001514 $(RPM(s))^{-1}$
k_h	-0.044797 m^{-1}

Table 3.1: values after parameter identification of thrust co-efficients

The term $-\hat{k}_z V_z \omega_s$ provides damping to the thrust acceleration produced. This can also be explained using the angle of attack analogy from Figure 3.4. When the QR ascends, it sees a higher induced flow. This results in a different 'resultant relative wind', which decreases the angle of attack of the blades with respect to the rotational relative wind. Decrease in angle of attack directly reduces the thrust produced.

The difference between the estimated thrust \hat{T}^B and the thrust recorded by ground truth T^B gives the residual. The residual is expected to follow a Gaussian distribution since the type of estimator exploited uses a bias-less minimum variance least squares method. The pattern of residual is important since a non-Gaussian fit signifies faults in the experiment performed or that an incomplete model was assumed (e.g. failing to capture transients, missing terms etc.). In our kind of experiment, a non-Gaussian fit could be originating from:

- The representation for thrust (Equation 3.3) is incorrect or misses significant terms.
- Or the experiment failed to excite the dynamics of thrust and its co-efficients/variables.
- Or the experiment performed was biased towards one particular maneuver.

After carefully performing the experiment, a Gaussian pattern of residual is obtained as illustrated in Figure 3.7.

3.2.4. Discussion

The following subsection includes some information about the extreme regimes that occur during aggressive flights and warns about the states to be avoided while the trajectory generation step. To understand these regimes better, the concept of angle of attack is introduced first.

During ascent, a reduction in the effective angle of attack (α) is observed due to change in the induced flow as illustrated in Figure 3.4. This reduction is observed because of the addition of induced flow ($V_{zi} = V_\infty - V_{bz}$) to the rotational relative wind (v_∞). As a result, the effective thrust that is produced during ascent is lower than expected.

During descent, an increase in effective angle of attack is observed due to the converse of the above explained phenomenon. As a result, the effective thrust that is produced while descent is higher than expected.

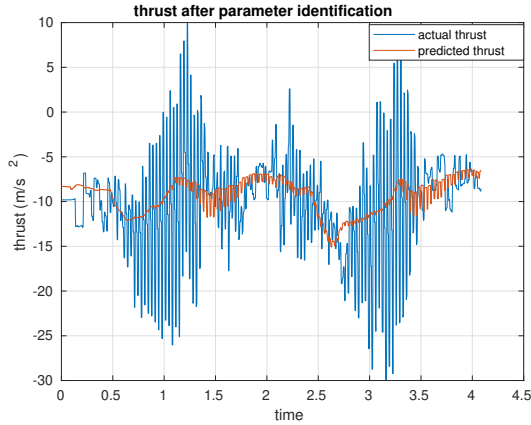


Figure 3.6: Predicted thrust \hat{T} after fitting.

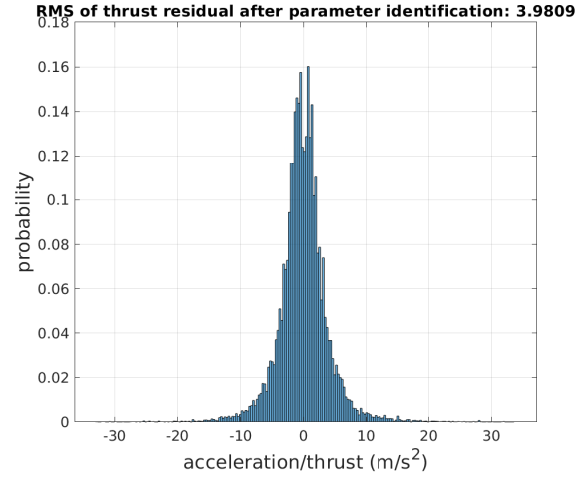


Figure 3.7: residual $(T - \hat{T})$ follows a Gaussian distribution, signifying a good quality fit.

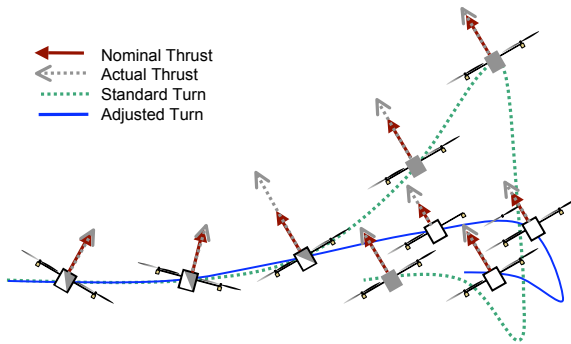


Figure 3.8: reduce thrust upfront for predicted change in angle of attack, figure from [14]

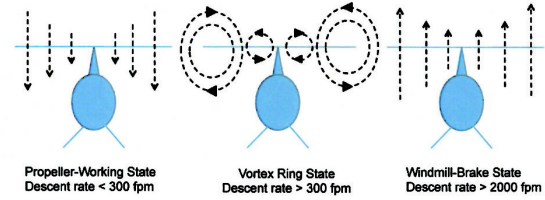


Figure 3.9: different regimes of QR flight, extreme left: desired regime, middle: low thrust VRS regime to be avoided, source: Tim Tucker rotorandwing.com

However, descents need to be carefully dealt with. Performing a descent at higher velocities beyond a threshold can break the assumptions made by blade momentum theory due to the induction of vortex ring states. In these states, there is a significant loss of effective thrust, in turn increasing the rate of descent while also causing significant turbulence. This can be avoided by performing a translation while descent. As a result, the QR does not fall under its own turbulent wake.

Another example of increase in thrust due to increase in angle of attack is illustrated by Figure 3.8. The figure illustrates that thrust models must reduce generated thrust abruptly upfront, in case a sudden change in angle of attack is expected during forward flight. If the generated thrust is not reduced upfront, change in angle of attack augments additional thrust, changing the altitude of the vehicle even if there is no command to do so (green path in Figure 3.8). Since there is no stopping in drone-racing, these states are rarely encountered. A QR performing laps in a drone race, usually generates trajectories that do not decelerate to such an extent.

The evasive method of filtering: To evade the tenuous modeling process for accurate thrust/altitude estimation, filtering methods complemented with coarse dynamic models can be used for improving the accuracy of estimation. Nowadays, time of flight sensors (ToF) are available in small packages, which can be mounted below the QR. ToFs have a short range for making measurements, however the expected trajectory in indoor drone-racing is not higher than a couple of meters. As a result, the above models can be fused using a complementary filter or Exponential Moving Average (EMA) filters to yield bias-less altitude measurements. The bias of the thrust measured by the accelerometer or by the dynamic QR model can hence be made observable by the presence of these sensors. This can yield bias-less thrust estimates.

3.3. Drag forces

In the very first part of the chapter, thrust coefficients and thrust models were discussed. Similarly in the second part, drag coefficients and drag models must be discussed.

In the absence of external wind or other disturbance forces, the QR only experiences forces due to its own movement through air. As a result, the only significant unestimated force except the thrust generated by the propellers is the drag force on the body of the QR. This drag force is augmented in both the force and moment models as described by the f_d term of Equation 2.1. Different types of drag forces and their effects are explained in the following subsections, since these forces and effects are fundamental for lateral state estimation. As explained in Table 2.1, this report does not investigate the induced drag in moment models, since they can be compensated by high gains in the closed loop control of the attitude loops [24].

3.3.1. Lift induced drag/rotor drag

The forces generated as highlighted earlier in Equation 3.3 can be classified into lift force, drag force and real collective thrust. When the resultant vector of lift and drag force is observed, there occurs a backward inclination of the net aerodynamic force of thrust with respect to the airfoil motion. Lift induced drag is unavoidable since it originates due to the inherent profile of the blades. Figure 3.4 indicates the contribution of this type of drag force with the dotted line. According to [20], induced drag is proportional to the lift generated by the airfoil. As a result, in case of higher RPMs during agile maneuvers, a higher damping due to lift induced drag is observed in the accelerations. The significance of some of the terms in state estimators discussed below are explained by lift induced drag. (T in Equation 3.11 and ω_s in Subsection 3.4.2).

3.3.2. Blade flapping

The concept of blade flapping is well explained by blade-momentum theory and flying handbooks of helicopters [1]. During forward flight of a rotating blade, higher lift is generated by the advancing rotor blade as compared to the trailing rotor blade due to the difference in their respective tip velocities with respect to the free-stream air. Flexible rotors can cause the advancing blade to rise with respect to the trailing blade of the translating propeller.

When the QR attains a constant velocity in forward flight, a steady state flapping angle is observed (α in Figure 3.11). This flapping angle creates a difference between the direction of actual thrust (T_{flap}) produced and the direction of the normal to the body of the QR (T_{ideal})³. If the body of the QR is banking with a certain angle, the direction of thrust applied by the propellers does not always remain normal to the bank angle of the body. As a result, due to the effect of blade flapping, the direction of thrust which is guessed by the IMU mounted on a body of the QR is rather coarse.

In Figure 3.10, v_∞ denotes the free stream air velocity induced due to QR's own velocity of v_x^E . Due to these above effects, in forward flight, blade flapping introduces additional forces and moments on the body of the QR. The additional force can be given by the term $TA_{flap}R^T\hat{\xi}$, and is derived as follows:

Similar to Equation 2.2, the lift and drag generated by a blade can be given by:

$$L = \frac{1}{2}\rho C_L A V^2, \quad L \propto V^2, \quad D = \frac{1}{2}\rho C_D A V^2, \quad D \propto V^2$$

For blade 1, $V = v_\infty + \omega r$ while for blade 2, $V = v_\infty - \omega r$.

The total lift generated by both blades can be given by $L_1 + L_2$ and the total drag produced by them can be given by $D_1 - D_2$. Substituting for the relative rotation velocity for each blade,

$$L_1 + L_2 = K[(v_\infty - \omega r)^2 + (v_\infty + \omega r)^2], \quad D_1 + D_2 = K[(v_\infty - \omega r)^2 - (v_\infty + \omega r)^2]$$

$$\text{Lift} = k_1 v_\infty^2 + k_2 \omega^2 r^2, \quad \text{Drag} = k_3 \omega v_\infty$$

QRs that use rigid propellers do not bend easily when subjected to aerodynamic forces. This does not have a big impact on the direction of thrust produced and it is rather parallel to the T_{ideal} vector in Figure 3.11.

³In hover conditions, the direction of thrust and the direction of the normal to the body are parallel and flapping angle is zero.

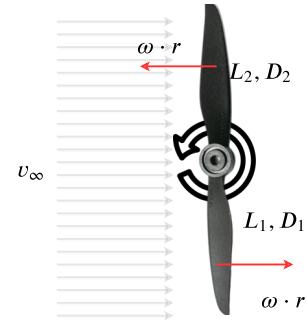


Figure 3.10: Relative relative wind seen with respect to the leading and trailing blade.

As a result, when transitioning to forward flight from hover positions, QRs that use rigid propellers directly transmit these aerodynamic forces to the frame of the body of the pitching QR. Forces that are imparted on the motor hub of a pitching QR can be modeled as moments acting on the CoM of the QR. As a result, small QRs that use rigid propellers must account for these terms in its moment model as well. Since Table 2.1 highlights that high gain controllers can eliminate disturbances caused in the moment model, this thesis only discusses the consequences of blade flapping on the translational force models which is important for position-tracking of QRs.

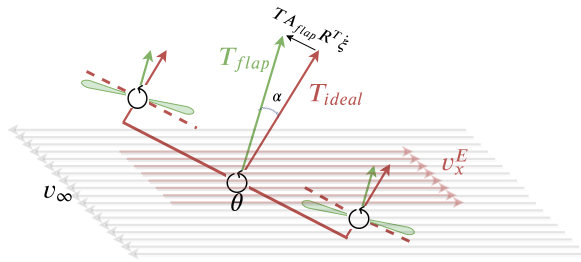


Figure 3.11: Rotation of ideal thrust vector (from red to green) due to blade flapping.

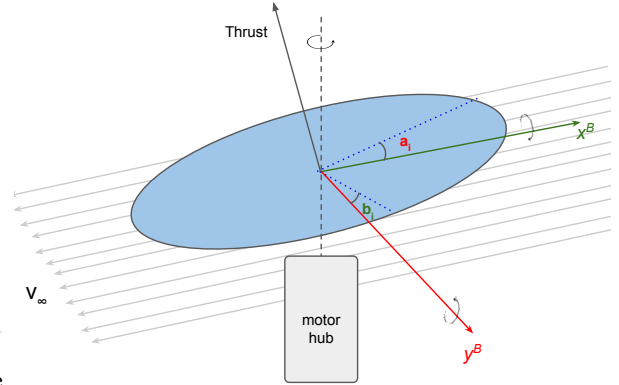


Figure 3.12: Plane of area swept by blade is now both pitching and rolling with respect to the original plane of blade rotation due to blade flapping.

sidenote: Blade flapping creates an offset of thrust in more than one plane. The offset of thrust is not only in the backward direction, but is also observed in the sideways direction. To explain this further, something called ‘gyroscopic torque’ needs to be introduced first. Gyroscopic torques occur when two moments acting on mutually perpendicular axes cause a resultant moment about the third axis. Considering independent rotors, a moment is induced as a result of the rotation of the blade about z^B and another moment is induced by the blade flapping component about the y^B axis. As a result there is a resultant moment or gyroscopic torque induced about the x^B axis. Figure 3.12 illustrates the effect of these moments on the blade of the QR. The moments generated modify the entire plane of rotation of the blade of the QR. Due to the original blade flapping moment, an offset of a_i is created and due to the gyroscopic torque an offset of b_i is created with respect to the original plane of the blade.

3.3.3. Parasitic drag

This type of drag force is due to the components of the body that are not capable of generating lift. This includes non-lifting components like the motors, the frame of the QR, the battery and the flight controller. Parasitic drag of the QR in forward flight can be given by:

$$f_D^{Bpar} = K_{par} |V^B| V^B \quad (3.4)$$

When flying faster, more frontal area of the body of the QR along the respective axis of movement is exposed to the incoming stream of air. As a result, the incoming air-stream creates a force that is proportional to the square of the velocity of the QR in a direction opposite to the QR's movement. As concluded in [2], for flight envelopes of QRs flying at moderate speeds up to 10m/s, parasitic drag may often be ignored. If these second order aerodynamic effects were pre-dominant in the flight, the tail of the plots in Figure 3.13 would look different. The tail of the plots correspond to high velocity segments recorded during the flight.

3.3.4. Key take-away from the classification of drag forces

1. The drag force arises due to blade flapping and is described by the black vector in Figure 3.11 as $TA_{flap}R^T\dot{\xi}$. As a result, the drag force (f_D^B) generated due to blade flapping is proportional to lateral velocity of the QR ($\dot{\xi}$).
2. Due to the inherent profile of the blade, the faster a blade rotates, it is subjected to higher induced drag. In the earlier section, the lift induced drag force arising has been shown to be proportional to the blade

RPM. After tracking the blade RPM of the Bebop, it was observed that the blade RPM of the respective rotor significantly increases in forward flight. As a result, even lift induced drag can be considered proportional to the free stream velocity v_∞ . The free stream velocity in our case, is completely a function of vehicle's own velocity through space. Hence, the drag forces arising due to above effects are directly proportional to vehicle's own velocity ($\dot{\xi}$). As a result, drag forces on the body (f_D^B) generated due to the lift induced drag is proportional to lateral velocity of the QR ($\dot{\xi}$).

3. Since all types of drag forces on the body (f_D^B) are proportional to the lateral velocity of the QR ($\dot{\xi}$), they can all be lumped into one parameter.
4. If the proportionality co-efficient of this lumped parameter can be identified, it can help estimate the forces on the body frame of the drone.

The next section performs system identification of the lumped drag parameter described above. Identifying this parameter will help conclude the sections on thrust and drag identification, which are the two main contributing forces in QR flight. As these forces act on the body frame of the QR, the forces can be rotated back to the world frame using the orientation information from the IMU. These forces (now expressed in the world frame) are simply subtracted with the gravitational acceleration. These compensated accelerations can now be dead-reckoned using kinematic equations to finally obtain the positional state estimates of the QR.

3.3.5. Drag parameter estimation

As noted from the earlier section describing drag forces, the forces experienced by the QR on the body frame are proportional to the vehicle velocity. Determining this proportionality constant is relatively trivial since only the first order effects of aerodynamic drag (effects proportional to vehicle velocity) need to be estimated. It can be later observed that the second order aerodynamic effects due to parasitic drag are not visible at these lateral velocities. This is because the bank angles required to achieve those high velocities are usually not in the flight envelope. When bank angles increase significantly, they expose more non-lifting surface area to the incoming flow of air, making the forces become proportional to the square of velocity.

The experiment for finding out the linear proportionality constants can be done in the following steps. The first step for estimating the drag co-efficients is to make a dataset. It involves making symmetric maneuvers for equal time duration while exciting lateral accelerations on the frame of the QR and having monotonic increasing time while making these observations.

1. Symmetric maneuvers: this is essential for unbiased samples in the dataset (equally bounded roll and pitch banking),
2. (of) equal time durations: essential for almost equal number of datapoints generating unbiased samples (equal time intervals for rolling and pitching maneuvers),
3. (while) exciting the lateral acceleration: essential for exciting the dynamics responsible for the drag parameter,
4. (having) monotonically increasing time: time stamps from the dataset must monotonically increase and are also preferred to have equal intervals.

The pre-processing step calculates various entities that can be calculated from the OptiTrack⁴ observations. It includes the following steps:

1. Pre-processing positional OptiTrack information to yield positions, velocities and accelerations in the inertial frame of reference.
2. Matching the IMU time stamps of orientations (ϕ, θ) with OptiTrack information.
3. Extracting body frame velocities and accelerations after using IMU and OptiTrack data.

After these entities are calculated, the equations for drag parameter estimation can finally be formulated. The onboard IMU measures the body's specific force and angular velocities. When the IMU is placed on the body of the QR, it becomes a non-inertial frame of reference with respect to the Earth frame. As a result, the IMU is now subjected to gravitational forces, centripetal forces and Coriolis forces. However, if it is

⁴OptiTrack is a visual tracking system that uses time of flight calculations to point out the ground truth pose of an object inside the lab

assumed that the IMU is mounted along the CoM of the QR, the centripetal force becomes zero. As suggested by the drag force estimators in [31], [10] and the key-take ways mentioned above, the acceleration on the body of the QR can be written as a function of rotor speed and lateral speed of the QR:

$$\dot{v}_x^b = -g \sin \theta - \underbrace{2\omega_z v_y^b + 2\omega_y v_z^b}_{\text{Coriolis force}} + \underbrace{k_{dx} \omega_s v_x^b}_{\text{drag force}}, \quad \dot{v}_y^b = g \cos \theta \sin \phi - \underbrace{2\omega_x v_z^b + 2\omega_z v_x^b}_{\text{Coriolis force}} + \underbrace{k_{dy} \omega_s v_y^b}_{\text{drag force}} \quad (3.5)$$

Accelerations from the IMU can be read out and the above equation can be rearranged as:

$$\begin{aligned} \dot{v}_x^b + g \sin \theta - 2\omega_z v_y^b - 2\omega_y v_z^b &= k_{dx} \omega_s v_x^b \\ \dot{v}_y^b - g \cos \theta \sin \phi + 2\omega_x v_z^b - 2\omega_z v_x^b &= k_{dy} \omega_s v_y^b \end{aligned} \quad (3.6)$$

After reaching steady state bank angle i.e. $\omega = 0$, there is no Coriolis force acting on the accelerometer.

$$\mathbf{a}_x^B = k_{dx} \omega_s \mathbf{v}_x^B, \quad \mathbf{a}_y^B = k_{dy} \omega_s \mathbf{v}_y^B$$

formulating it as a least squares problem,

$$\begin{bmatrix} \mathbf{a}_{x|t=0}^B \\ \mathbf{a}_{x|t=1}^B \\ \vdots \\ \mathbf{a}_{x|t=N}^B \end{bmatrix} = \begin{bmatrix} 1 & \omega_{s|t=0} \mathbf{v}_{x|t=0}^B \\ 1 & \omega_{s|t=1} \mathbf{v}_{x|t=1}^B \\ \vdots & \vdots \\ 1 & \omega_{s|t=N} \mathbf{v}_{x|t=N}^B \end{bmatrix} \begin{bmatrix} b_x \\ k_{dx} \end{bmatrix}, \quad \therefore \begin{bmatrix} \hat{b}_x \\ \hat{k}_{dx} \end{bmatrix} = \begin{bmatrix} 1 & \omega_{s|t=0} \mathbf{v}_{x|t=0}^B \\ 1 & \omega_{s|t=1} \mathbf{v}_{x|t=1}^B \\ \vdots & \vdots \\ 1 & \omega_{s|t=N} \mathbf{v}_{x|t=N}^B \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{a}_{x|t=0}^B \\ \mathbf{a}_{x|t=1}^B \\ \vdots \\ \mathbf{a}_{x|t=N}^B \end{bmatrix} \quad (3.7)$$

Performing this pseudo-inverse yields the unbiased, minimum variance estimate of the drag parameter. The

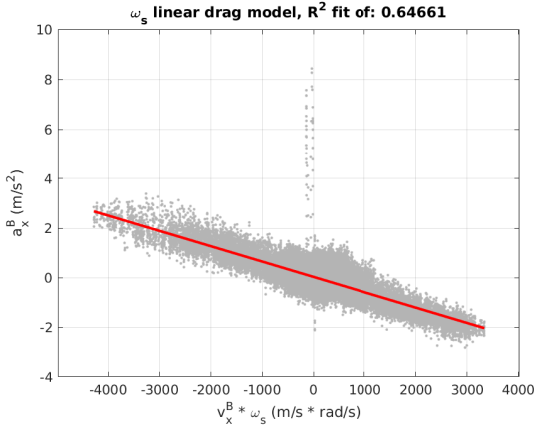


Figure 3.13: drag co-efficient identification for x^B .

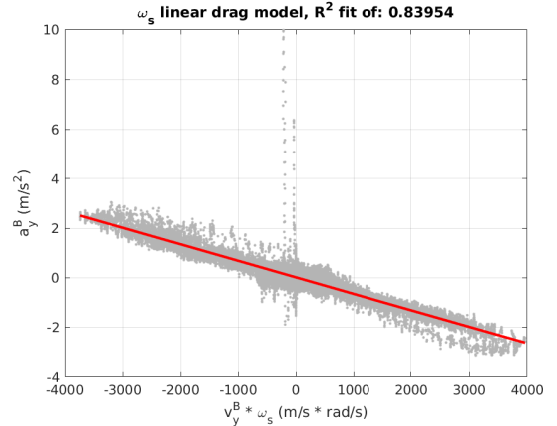


Figure 3.14: drag co-efficient identification for y^B .

same identification can be repeated for the lateral y^B axis for k_{dy} yielding Figure 3.13 and Figure 3.14. The following observations can be made from this experiment:

1. The red line in the figures indicate the drag parameter, which reprojects the body frame accelerations on the red line given an estimated body frame velocity. From now on, if either of the entities (acceleration of the body or the velocity of the body) is known, the other can be estimated using the identified drag co-efficient.
2. Few papers perform drag estimation using similar techniques. Parameter estimation is performed using each of their estimators presented in [31], [21] and [24] for the sake of comparison. This yields the following table with fit values. The R^2 values indicate the quality of the fit, higher R^2 implies a better fit.

The k_{dx}, k_{dy} that are estimated from the dataset are stored for later use on different set of flights. If the estimated parameters are used for performing state estimation on the same dataset, the experiment can give excellent results. However, these metrics are not relevant since the experiment qualifies under a over-fitted dataset. For recursive estimation recursive method for drag estimation mid-flight can be made using EKF (Extended Kalman filters).

Drag fit quality	GRASPLab, UPenn[31]	UZH[21][20][19]	ETH-CNRS [24]
k_{dx}	$0.000619rad^{-1}$	$0.5395s^{-1}$	$-0.051(srads)^{-1}$
k_{dy}	$0.000667rad^{-1}$	$0.5822s^{-1}$	$-0.567(srads)^{-1}$
R^2 of k_{dx}	0.6466	0.4096	0.6505
R^2 of k_{dy}	0.8395	0.8336	0.8424

Table 3.2: R^2 and drag values of fit of drag parameter using different state estimators

3.4. Modeling of QR dynamics

Having estimated the drag parameters, they can now be used in the design of accurate lateral-positional state estimators. Moving to the second part of the chapter, sections below describe different models proposed prior to this thesis. It later moves on to propose and test a new model that could be useful in the drone-race framework.

There have been quite a few papers that explore the area of lateral state estimation on Newton-Euler force models while using only the IMU information [10] [19] [20] [24] [31]. In the subsections below, each of their proposed models alongwith their derivations have been re-introduced, so they can be compared against one another. At the end of this section, the R^2 based convergence metric compares these methods with respect to the ground truth.

Free-body diagrams and Newton-Euler models have already been introduced in the earlier chapter. Newton's method is used to estimate the lateral accelerations on the body frame by making use of the free body diagrams. Knowing the rotation matrices described in the earlier chapter, the body accelerations can be converted to Earth/inertial frame accelerations. Performing Euler integration on these world frame accelerations can yield the displacement of the QR in the inertial frame of reference. To revise again, these lateral-positional state estimates (ξ_x, ξ_y, ξ_z) are critical to drone racing and indoor navigation, since control modules depend on them for planning accurate flights through the gates. The states to be estimated are hence the lateral positions, accelerations and the velocity of the QR in the inertial frame of reference.

3.4.1. Conventions and Symbols

- In the sections below, matrix R is the rotation matrix converting body frame entities to the world frame.

$$R = {}^E R_B \text{ and } R^T = {}^B R_E$$

- $\dot{\xi}$: velocity in the inertial frame of reference.
- $\ddot{\xi}$: acceleration in the inertial frame of reference.
- z^E - the z-axis of the inertial frame of reference, gravity acts on this axis.
- g - gravity (9.81 m/s^2)
- T^B : thrust generated by all the propellers on the body of the QR, thrust is expressed in the body frame of reference.
- f_d^B : drag forces experienced by the QR in the body frame of reference.
- v_h^B : lateral velocity in the $x^B - y^B$ plane of the body frame of the QR.
- v_z^B : ascent/descent velocity along the z^B axis of the QR.
- $\omega_s = \frac{\sum_{i=1}^4 \omega_i}{4}$: instantaneous rotor speed (averaged for 4 propellers).

3.4.2. The GRASPLab, UPenn and ANU estimator

The state estimators presented in [20] and [31] use the idea of lift-induced drag to scale the drag coefficient with the propeller speeds. Let P be a projection matrix which obscures the z^E estimates and let the average propeller speed in RPM be given by ω_s . The lateral (x-y) velocity of the QR represented in the body frame can be given by:

$$v_h^B = PR^T \dot{\xi}, \quad \text{where } P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The linear drag model in [20] gives the acceleration in body frame by:

$$f_d^B = k_d \omega_s v_h^B$$

from the dynamical equations stated in Equation 2.1 it is known that,

$$\ddot{\xi} = T^B R z^E - g z^E - R f_d^B$$

substituting for f_d^B yields,

$$\ddot{\xi} = T^B R z^E - g z^E - R(k_d \omega_s v_h^B), \quad \therefore \ddot{\xi} = T^B R z^E - g z^E - k_d \omega_s R P R^T \dot{\xi}$$

The models in [31] simplifies the fundamental thrust equation given by blade momentum theory. As a result, the thrust equation:

$$T_i = c_1 \omega_i^2 \left[c_2 \left(1 + \frac{3}{2} \mu_i^2 \right) - \lambda_i \right], \text{ which is simplified to, } T_i = k_\omega \omega_i^2 - k_z V_{zi} \omega_i + k_h V_{hi}^2 \quad (3.8)$$

where k_ω is the thrust co-efficient mapping the angular velocity of the propellers to the thrust, k_z is the drag co-efficient in the z^B axis. V_{zi} is the induced velocity of the air through the propellers. A simplification made assumes the induced velocity through the propellers to be the same as V_{bz} . k_h is called the 'collective' term in the thrust. This term is adopted from helicopter dynamics, indicating that there is an increase in effective thrust produced with increase in lateral body velocities along the horizontal plane. In other words, when the vehicle is translating fast along the horizontal axes, the thrust generated by the blades at "x" RPM, is higher than the thrust that would have been generated at the same "x" RPM while hovering.

Thrust estimates from the above equations could be used to yield better z-positional estimates. However, this paper uses a projection matrix "P" to remove the influence of thrust based acceleration on the altitude estimation of the QR.

3.4.3. The ETH-Zurich/CNRS France estimator

Another state estimation technique by [24] is proposed, which scales the drag co-efficients with the instantaneous thrust being produced by the QR. Applying the Newton Euler formulation for the translational model,

$$m \ddot{\xi} = R \sum_{i=1}^4 F_i + m g e_3 + F_{aero} \quad (3.9)$$

The translational velocity of each propeller is now estimated. This helps in estimating the advance ratio of the blades, which later helps in constructing the flapping matrix. (Advance ratio of a blade is defined as the ratio of the freestream air speed to the propeller speed). The velocity of each propeller can be given by:

$$v_{ri} = R^T (\dot{\xi}) + \omega \times d_i$$

where ω is the angular velocity of the body. d_i is the distance of the rotor hub from the CoM of the QR.

The advance ratio for each propeller can now be given as:

$$\mu_{ri} = \frac{|v_{ri,1,2}|}{\omega_i r}$$

where r is the radius of propeller. It was earlier observed that flapping occurs along both x^B, y^B axes creating a flapping angle between the body of the QR and the tip path plane. The flapping angles a_i, b_i as earlier illustrated in Figure 3.12 are given by:

$$\begin{bmatrix} a_i \\ b_i \end{bmatrix} = \begin{bmatrix} \frac{c_a}{1 - \frac{\mu_{ri}^2}{2}} & \frac{-c_b}{1 + \frac{\mu_{ri}^2}{2}} \\ \frac{-c_b}{1 + \frac{\mu_{ri}^2}{2}} & \frac{c_a}{1 - \frac{\mu_{ri}^2}{2}} \end{bmatrix} v_{ri,1,2}, \quad \begin{bmatrix} a_i \\ b_i \end{bmatrix} \approx \begin{bmatrix} c_a & -c_b \\ c_b & c_a \end{bmatrix} v_{ri,1,2}$$

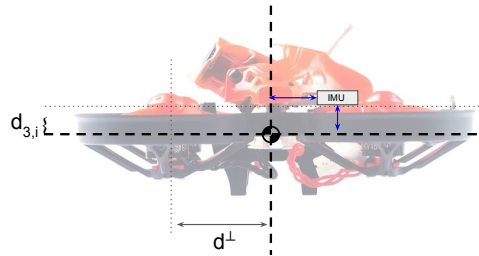


Figure 3.15: distance from centre of mass of each propeller.

$[a_i, b_i]$ given above are the lateral and longitudinal flapping angles which disturb the thrust produced as described in Figure 3.11. a_i, b_i are also directly proportional to the lateral velocity of the vehicle in the body frame. These flapping angles can now be substituted in the prior Equation 3.9:

$$\sum_{i=1}^4 F_i = -Tz^E - c_T A_{flap} \sum_{i=1}^4 \omega_i v_{ri} \quad \text{where, } A_{flap} = \begin{bmatrix} c_a & -c_b & 0 \\ c_b & c_a & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

substituting for v_{ri} ,

$$\sum_{i=1}^4 F_i = -Tz^E - c_T A_{flap} \sum_{i=1}^4 \omega_i R^T(\dot{\xi}) + \omega \times d_i \quad (3.10)$$

splitting lateral dynamics,

$$\sum_{i=1}^4 F_i = -Tz^E - T A_{flap} (R^T \dot{\xi} - \underbrace{\omega \times d_{3,i} z^E}_0) + \underbrace{A_{flap} \omega \times \sum_i T_i d_i}_0$$

There are two assumptions made by [24] at this point. The first assumption is that $d_{3,i} \ll 1$, which could hold true for most QR designs as in Figure 3.15. The second assumption is that the drag models are validated after reaching a state state bank angle, with $\omega \approx 0$. This removes the last two terms from the above equation, yielding:

$$\sum_{i=1}^4 F_i = -Tz^E - T A_{flap} R^T \dot{\xi}$$

substituting in Equation 3.9,

$$m\ddot{\xi} = R(-Tz^E - T A_{flap} R^T \dot{\xi}) + mgz^E + F_{aero} \quad (3.11)$$

$$\text{where, } F_{aero} = k_{parasitic} |v_{x,y}|^2$$

It can be noted by A_{flap} 's last row that z-estimates have been obscured out.

3.4.4. The UZH estimator

Unlike the above models, the estimators in [21] do not eliminate the z-estimates via projection matrices.

$$cZ_B - \begin{bmatrix} \mathbf{x}_B & \mathbf{y}_B & \mathbf{z}_B \end{bmatrix} \begin{bmatrix} d_x & 0 & 0 \\ 0 & d_y & 0 \\ 0 & 0 & d_z \end{bmatrix} \begin{bmatrix} \mathbf{x}_B^\top \\ \mathbf{y}_B^\top \\ \mathbf{z}_B^\top \end{bmatrix} \cdot \mathbf{v} = \mathbf{a} + g\mathbf{z}_W \quad (3.12)$$

which if translated to conventions and symbols used in this report can be written as:

$$\ddot{\xi} = g z^E - TR^T z^B - R^T DR \dot{\xi} \quad (3.13)$$

The term d_z stems from Equation 3.3 although not explicitly mentioned. Although, this model misses out on the lift-induced drag term, which imparts higher damping to the forces on the body of the QR when the propellers are spinning at a higher RPM.

3.5. Summary of prior estimators

This section re-states the state estimators derived above, and re-writes them using the conventions used in this report.

1. GRASPLab, UPenn and Australian National University [31] [20]

$$\begin{bmatrix} \dot{v}_x^B \\ \dot{v}_y^B \\ \dot{v}_z^B \end{bmatrix} = w_s \begin{bmatrix} -k_{dx} & 0 & 0 \\ 0 & -k_{dy} & 0 \\ 0 & 0 & 0 \end{bmatrix} [{}^B R_W] \begin{bmatrix} \dot{v}_x^E \\ \dot{v}_y^E \\ \dot{v}_z^E \end{bmatrix}, \quad \begin{bmatrix} \dot{v}_x^E \\ \dot{v}_y^E \\ \dot{v}_z^E \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + [{}^W R_B] \begin{bmatrix} \dot{v}_x^B \\ \dot{v}_y^B \\ T_h \end{bmatrix}$$

where w_s is the average prop speed

2. ETH-Zurich/CNRS France [24]

$$\begin{bmatrix} \dot{v}_x^B \\ \dot{v}_y^B \\ \dot{v}_z^B \end{bmatrix} = T_h \begin{bmatrix} -k_{dx} & 0 & 0 \\ 0 & -k_{dy} & 0 \\ 0 & 0 & 0 \end{bmatrix} [{}^B R_W] \begin{bmatrix} v_x^E \\ v_y^E \\ v_z^E \end{bmatrix}, \quad \begin{bmatrix} \dot{v}_x^E \\ \dot{v}_y^E \\ \dot{v}_z^E \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + [{}^W R_B] \begin{bmatrix} \dot{v}_x^B \\ \dot{v}_y^B \\ T_h \end{bmatrix}$$

where T_h is the thrust produced by the QR along the z^B axis

3. UZH [21]

$$\begin{bmatrix} \dot{v}_x^E \\ \dot{v}_y^E \\ \dot{v}_z^E \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + [{}^B R_W] \begin{bmatrix} 0 \\ 0 \\ T_h \end{bmatrix} - \begin{bmatrix} -k_{dx} & 0 & 0 \\ 0 & -k_{dy} & 0 \\ 0 & 0 & -k_{dz} \end{bmatrix} \begin{bmatrix} v_x^E \\ v_y^E \\ v_z^E \end{bmatrix}$$

3.6. Requirements for a better lateral-state estimator

The systems of dynamical equations obtained must make the model causal, nonlinear and time invariant.

1. **Quality of estimates:** It is important to select a drag model that gives better convergence, lower R^2 epsilons and drift less from the ground truth.
2. **Limited system identification steps:** The model must be representative of the actual drone by accounting for as many external forces and torques that the designer is aware of. However, parameters in this model should be able to be identified within a few flights that excite the relevant dynamics. Models should also be resistant to over-fitting towards a particular drone, for e.g. the radius of the blade must not be accumulated in the identified constants, the distance of the motor hub from the centre of mass must not be accumulated. In [30], the QR is exposed to high speed wind tunnel tests while stepwise regressors fit the model for accurate state estimation. This thesis can not use these methods, since they are QR specific.
3. **Willingness to look in the future/causality:** The final requirement for the estimator is that the QR's

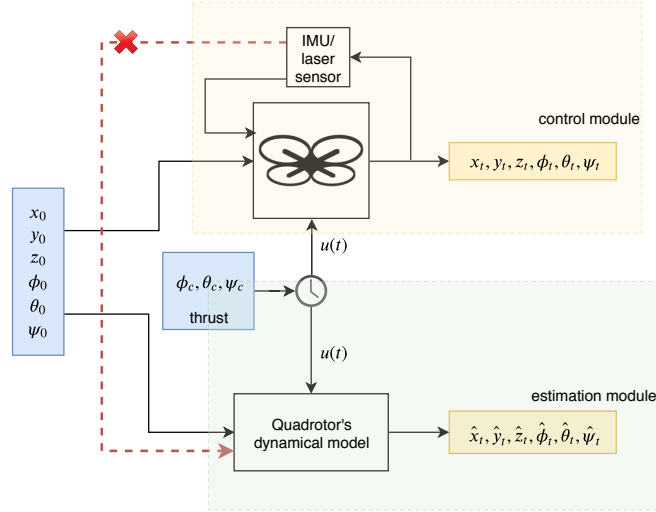


Figure 3.16: red cross indicating prevention of non-causal systems by not considering measurements from the future.

dynamical model should be free from expectations of measurements that would be made in the future. An example of a measurement dependent state estimator uses an accelerometer for performing sensor-fusion between the acceleration predictions from the drag models and the sensor. In this manner, the bias of the accelerometer, which gives a huge drift when dead-reckoned, will not be used and only the high frequency changes of these estimates would be. Equation 21 from [20] is restated here, which suggests usage of accelerometers in a complementary observer system.

$$v_h \approx -\frac{1}{g} [\mathbb{P}_h \hat{R} D \hat{R}^T \mathbb{P}_h^T]^{-1} ({}^A a_h + g \mathbb{P}_h \hat{R} \vec{z}) \quad (3.14)$$

The terms in the equation are explained in Subsection 3.4.1. The additional term $^A a_h$ in the above equation is the accelerometer measurement. The measurement update rule after receiving accelerometer updates was then given as:

$$\dot{\hat{v}}_h = -g\mathbb{P}_h^\top (\hat{R}\vec{z} + \hat{R}D\hat{R}^\top \mathbb{P}_h^\top \hat{v}_h) - k_w(\hat{v}_h - v_h)$$

However, the model that is required for in our framework of drone-race, should **only** be a function of the current pose of the QR and the inputs given to the QR by the controller. Augmenting sensor information in the model can give convergent estimates, however it has two disadvantages.

- (a) Accelerometer measurements must be carefully filtered. They must be compensated and calibrated correctly before performing sensor-fusion. The reasons for these extra precautions are listed in Section A.3.
- (b) Secondly, performing sensor-fusion in order to bypass the design of robust state estimators reduces the chance of making long horizon estimates while performing optimal control. Since sensors would not be active in the feed-forward region while performing optimal control, the QR would **only** be relying on the design of state estimators for prediction of trajectories.

A ‘causal’ model in control systems implies something similar. It prevents expectations of having measurements from the sensors in the future. If the future measurements are included in the dynamical model of the QR as in Equation 3.14, it would make the model immediately non-causal. Figure 3.16 highlights the recommended structure of the observer/estimator. The causal estimator in the green box can predict the states of a QR, which **only** is a function of the inputs to the system $u(t)$ and the current states (x_0, y_0, z_0, ψ_0) .

4. **Time invariant systems:** If the goal of designing state estimators is for performing optimal control, then a requirement of time in-variance should be introduced. For each time that the optimal control algorithm is invoked, the state space representation of the dynamic QR must be fixed, until the control horizons have been reached. This is because an optimal control sequence would have been generated at t_0 using a fixed representation of the QR model. If this assumption is not made, complicated adaptive algorithms or MPC with multi-objective parameter estimation should be used for being able to deal with changing parameters in the model.
5. **Speed:** In drone racing, reaching highest possible lateral velocities is crucial for finishing the course in minimum time. This creates some advantages and disadvantages for the state estimators. The advantage being that there is a lesser drift in the state estimates over time, since the time span of covering the course while flying at higher velocities is considerably reduced. Another advantage of flying fast is that it facilitates more observations and corrections offered by the vision pipeline, since the gates, which act as localization markers for the drones, are visible early-on in the trajectory. The disadvantage to flying fast is that it could invite some unexpected forces to act on the body of the QR. It is suspected that second order aerodynamic effects and vortex states are the most common phenomena that can occur when flying fast. Second order aerodynamic effects can not be neglected when the QR banks to extreme angles, while exposing a huge cross section of its non-lifting surface to the incoming free stream wind velocity. Banking to agile angles against the direction of motion for stopping or descending with high lateral velocities can invite vortex states, leading to sudden decrease in effective thrust that is being produced. These trade-offs must be taken into account while designing the state estimation and control framework.

3.7. Proposed state estimator

The proposed method includes the increased damping due to increase in rotor speed by scaling drag coefficients, while also not obscuring out the z^B frame estimates. It takes the good features of scaled drag coefficients from [24] and [20] and lateral z-axes from [31] and [21].

$$\ddot{\xi} = TRz^E - gz^E - R\omega_s \begin{bmatrix} k_{dx} & 0 & 0 \\ 0 & k_{dy} & 0 \\ 0 & 0 & k_{dz} \end{bmatrix} R^T \dot{\xi} \quad (3.15)$$

It can be assumed that there is a second-order transfer function/fitted polynomial between ω_s and ω_{cmd} depending on the battery SoC [21]. This thesis assumes that $\omega_s = \omega_{cmd}$, since the Bebop has a closed loop for RPM control. As a result, the above equation can still remain causal.

$$T_i = k_\omega \omega_i^2 - k_z V_{zi} \omega_i + k_h V_{hi}^2 \quad (3.16)$$

Features	GRASPLab, UPenn[31]	UZH[21]	ETH-Z/CNRS[24]	Proposed
x_{est}	✓	✓	✓	✓
y_{est}	✓	✓	✓	✓
z_{est}	✗	✓	✗	✓
variable drag ⁵	✓	✗	✓	✓

Table 3.3: features of the state estimators

3.8. Evaluation of different state estimators

State estimation was usually performed at slower lateral velocities when the above models were tested. As put by the famous statistician W. Edwards Deming on experimental bias- *“Your system is perfectly designed to give you the results that you get.”*, this section devises experiments to stray away from conducive results. Testing the robustness of the proposed estimator in harsh drone-race conditions also assists in generating and comparing the key-metrics between different methods while exciting the relevant dynamics. In these conditions, there are high changes of performing high-velocity maneuvers with high-angular body rates for tracking abrupt attitude commands. Hence, two separate tests are devised for testing the performance of discussed estimators in both of these scenarios of: (a) high lateral velocity and (b) high angular rate maneuvers.

3.8.1. Results: Maneuvers with high lateral velocities

During high speed flights, it is likely that the governing dynamics explained by the linear drag models do not hold. It might be possible that second order aerodynamic effects become significant and sometimes also vortex states occur when the QR stops in its own wake. For this reason, the QR is excited to relatively high lateral velocities, characterizing the accuracy of different state estimators in such conditions. The trajectory used for exciting high lateral velocities is given as follows:

1. An L-shape trajectory is commanded.
2. Yaw setpoint is always zero, the QR does not change heading mid-flight.
3. Pitches forward for a couple of seconds, rolls right for the rest of the flight.
4. A sinusoidal altitude is commanded throughout the maneuver as seen from the third subplot of Figure 3.19.

↓ RMS e_ξ estimator ⇒	UZH/Delft	Proposed	CNRS
$e(\xi_x)$	0.2814	0.0431	0.1078
$e(\xi_y)$	1.2341	0.1098	0.2693
$e(\xi_z)$	1.5437	0.1773	0.7979

Table 3.4: RMSE from the ground truth **positions** in the L shaped maneuver *units in meters

The height of the flight was constantly changed to show the effect of improvement in trajectories after the z-estimation is accurate. Figure 3.18 illustrates the differences between different position estimators which were described earlier. Table 3.4 gives the comparison for the key-metric of positional accuracy of each estimator. Tracking errors are the Root Mean Square values of the absolute errors between positional estimates with respect to the ground truth.

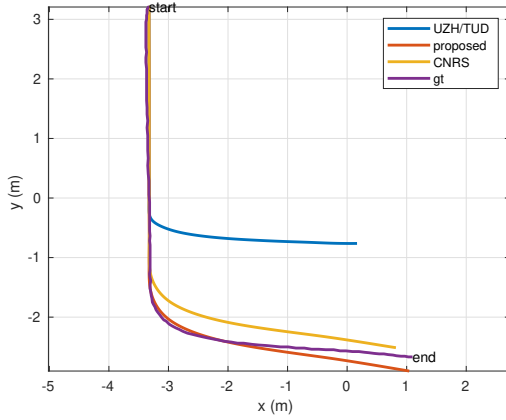


Figure 3.17: Top view: L-shape maneuver- comparison between different state estimators.

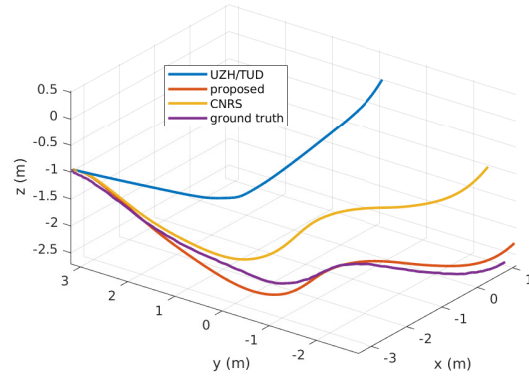


Figure 3.18: Isometric view: L-shape maneuver

3.8.2. Results: Maneuvers with high angular velocities

When rigid bodies are subjected to higher angular velocities, pseudo-forces like centrifugal and Coriolis force act on the body frame, leading to unexplained dynamics apparent to an observer on a non-inertial frame of reference (in our case the B frame). Maneuvers exciting these higher angular velocities are performed to check the robustness of the proposed estimator.

The most frequent maneuver in drone racing is rolling to correct the lateral offset from the gate, while having a constant pitch angle (earlier depicted in Figure 1.6). In this case, the QR has already acquired enough velocity in v_x^B to fly towards the gate. However, it must now roll at high angular velocities about the x^B axis, to cancel out the lateral positional offset so to fly through the center of the gate. Due to initial lateral velocity and a high commanded angular velocity (roll axis), a potential Coriolis force can be induced on the frame of the QR given by:

$$f_C^B = \omega \times v = \begin{bmatrix} \omega_x \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} v_x \\ 0 \\ 0 \end{bmatrix} \approx 0$$

As a result, there is no pseudo force acting on the body of the QR when rolling during a forward flight. To check the robustness of the above proposed estimator in rotationally excited flight, the QR is now subjected to high yaw-rates while maintaining a constant bank angle. This creates circular trajectories.

$$f_C^B = \omega \times v = \begin{bmatrix} 0 \\ 0 \\ \omega_z \end{bmatrix} \times \begin{bmatrix} v_x \\ 0 \\ 0 \end{bmatrix} \approx \begin{bmatrix} 0 \\ f_{Cy}^B \\ 0 \end{bmatrix}$$

1. Circular shaped maneuver, relatively high yaw rates.
2. Commanded a constant pitch angle, roll angle and yaw-rate.
3. Constant altitude, thrust generated almost constant throughout the maneuver.
4. Tried to excite high lateral velocities in the body frame.

↓ RMS e_ξ estimator ⇒	UZH	Proposed	University of Côte d'Azur (CNRS)
$e(\xi_x)$	0.3523	0.3668	0.3551
$e(\xi_y)$	0.3205	0.2498	0.2103
$e(\xi_z)$	0.8857	0.3242	0.7422

Table 3.5: RMSE from the ground truth **positions** during the circular maneuver *units in meters.

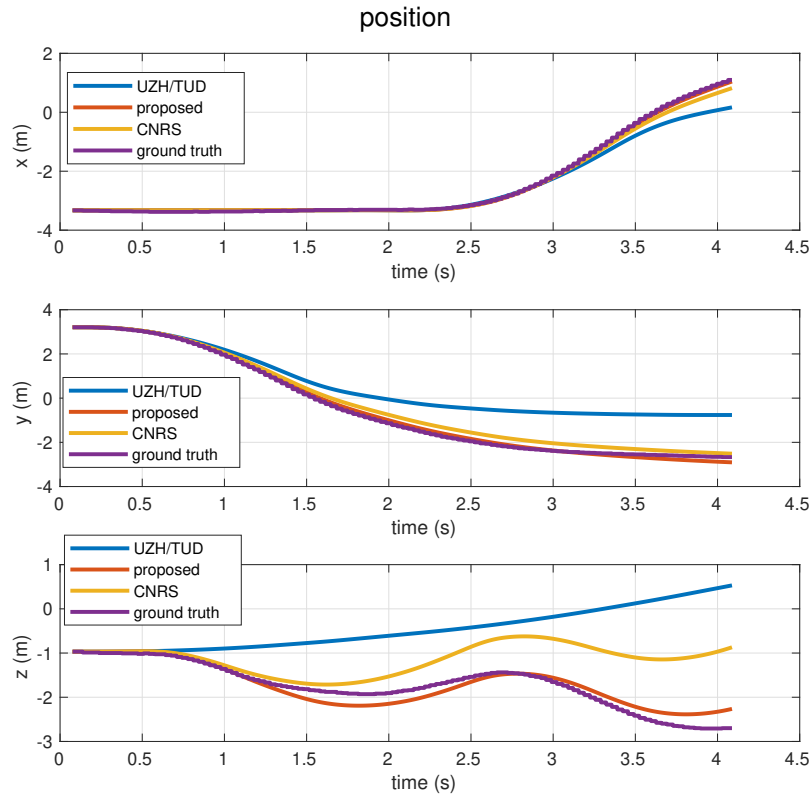


Figure 3.19: comparison of estimated position with time

3.9. Analysis

After performing several experiments (these two in specific), the RMSE values indicate that the proposed estimator performs better than the ones presented earlier in literature. These RMSE values can also be used for making a good guess about the process noise co-variance before running an Extended Kalman filter.

The proposed estimator also meets the requirements of time in-variance, causality and requires only a single system identification experiment for drag estimation.

Rigorous system identification steps must be performed to obtain accurate state estimates as is the case in [30]. Also, [21] runs system identification between battery power and produced thrust, which must be done if the speed controllers do not have a closed loop in the angular RPM setpoints. System identification helps to generate the same thrust at any time instant through the flight, even if the battery discharges with time.

In case of poor estimation along the z^B axis, rotation matrices could transfer the drifts to x , y estimations as well. It is very likely that the thrust being generated has poor estimation because of poor R^2 values earlier found from Equation 3.3. The transfer of poor estimation from one estimate to another is a function of bank angle of the QR. When the QR banks to higher angles for fast flights, the estimates between different axes are even more coupled. As a result, coarse estimation of thrust can decrease the quality of estimates in other axes. On the upside, accurate state estimates in one axis could also lead to good estimations along other axes.

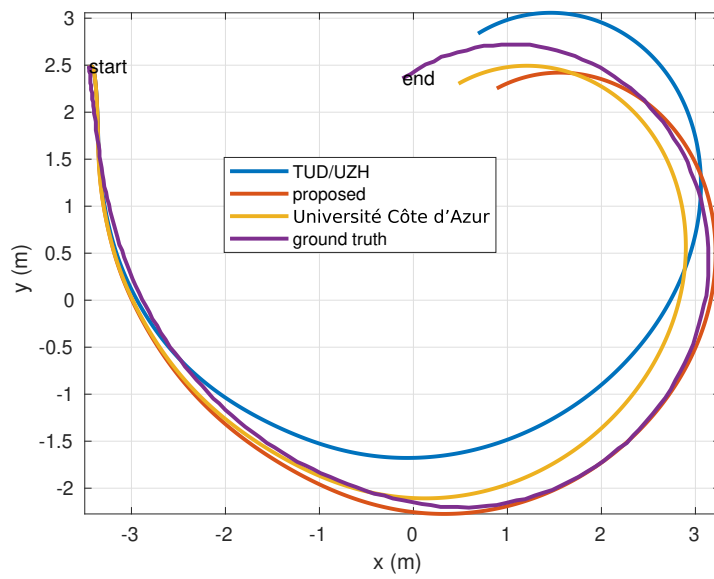


Figure 3.20: Top view: Circular maneuvers- comparison between different state estimators.

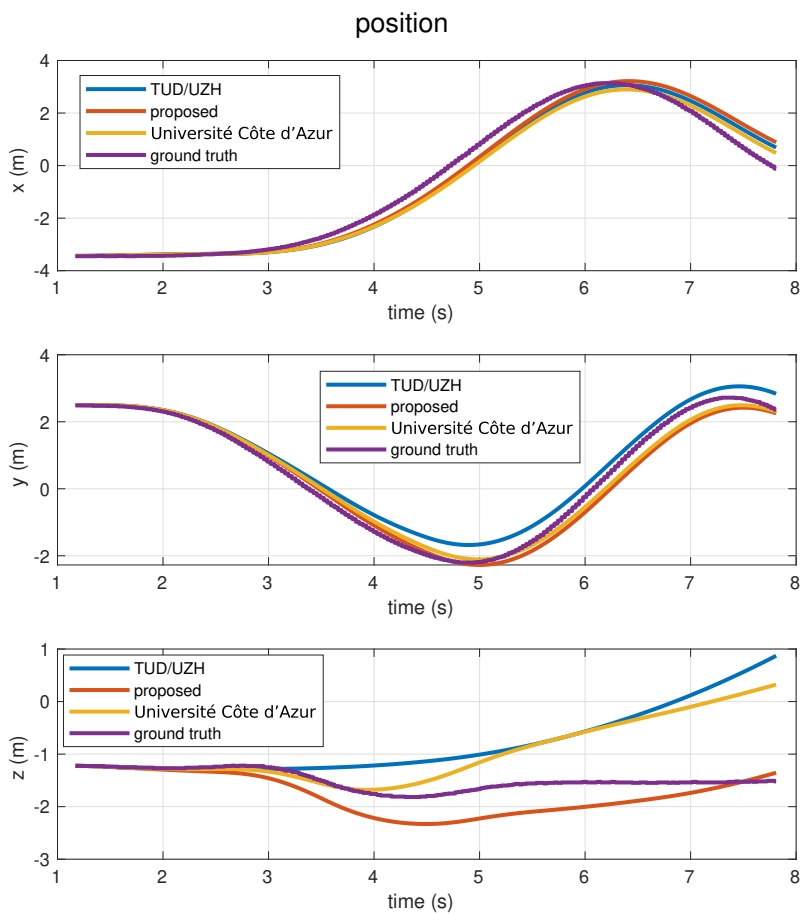


Figure 3.21: Comparison of the estimated position with time using different estimators.

4

Optimal control

In the earlier chapter, the application of state estimation was limited to improving the accuracy of the positional states of a QR. This chapter illustrates that the application of state estimation can also be extended to the control of QRs. Since the QR states, which were earlier un-observable are now observable via state estimation, QR control along all the axes is now possible (Table 2.1 lists controllability, observability and the axes associated).

As a result, after possessing accurate knowledge of all the positional and rotational states, a QR can control itself autonomously, as soon as a control law is established. The QR that was barely self-aware till the last chapter can now transition into an autonomous vehicle that can be commanded to track trajectories around the drone-racing arena.

This chapter firstly introduces traditional QR control techniques and illustrates potential problems that can arise while using them. After this introduction, requirements are established for designing a controller that can fit well in the drone-race pipeline. The chapter concludes by plotting different trajectories that are generated while using different controllers alongside a table of key metrics.

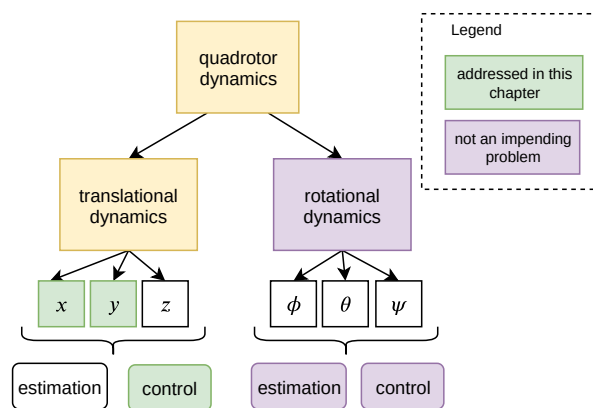


Figure 4.1: Scope of this chapter is limited to control along the lateral positional axes.

4.1. Position control for QRs

With a working position control loop, a QR can navigate to any point in the inertial frame of reference of the drone race arena. Positional control loops usually run at lower frequencies of around 30Hz, finally sending their commands to the attitude and thrust control loops. However, the manner in which this position control is achieved can vary significantly. The QR can reach the destination using one of the several ways:

- The time optimal way: The trajectory followed makes the QR reach its desired position in minimum time. For the drone-racing scenario, the first type of maneuver is important. Flying through the assigned gates in minimum-time irrespective of QR's initial position or velocity is critical for finishing first.
- The snap/jerk optimal way: The trajectory followed makes the QR reach the desired position without any jerks or snaps (higher order derivatives of acceleration). This simply implies that the trajectories are smooth splines that guide the QR to the desired state. This method is generally used to guide vehicles through obstacles or generate circular trajectories.
- The energy optimal way: The trajectory followed makes the QR reach the desired position using minimum energy delivered from the battery. For flying through certain waypoints in time optimal fashion,

QRs increase their altitude and later make descent through the waypoint with higher velocities. Such maneuvers are not energy optimal. Energy optimal maneuvers take into account the actuator effort and battery state of charge.

However, the techniques described above are rather advanced control techniques. These type of techniques require some knowledge of the dynamical models of the system, a complicated software environment setup and fast processors to solve the optimization problems in real time.

As a result, optimal control techniques have not been a very attractive solution to users and the traditional techniques of PID control have found their way to the position control of QRs. These techniques, unlike optimal control techniques, do not require a lot of prior theoretical knowledge, complicated softwares or fast processors. Although, they do require the user to understand the tuning intricacies associated with PID control.

4.2. Existing QR control techniques

For supporting a brief introduction on control of an autonomous QR, the sub-sections below discuss existing methods for control of QR before picking and proposing a technique that could be a candidate in the drone-race framework.

4.2.1. PID

As stated earlier, Proportional Integral Derivative techniques which are most commonly used in the control systems area, find its application in QR control as well. To coin the working of this controller in simple terms: Proportional gains map control actions to the actuators depending on the distance from the setpoint. Derivative gains change the speed of the mapped actions. Integrator gains make sure that small convergence errors to the setpoint disappear over time.

Small convergence errors called as steady state tracking errors in the context of control systems, are not of specific interest in fast flights. Pilots rarely care about trajectory tracking errors and aim to fly through the gates as quickly as possible. For the same reason, an integral term is absent in the control modules of racing drones.

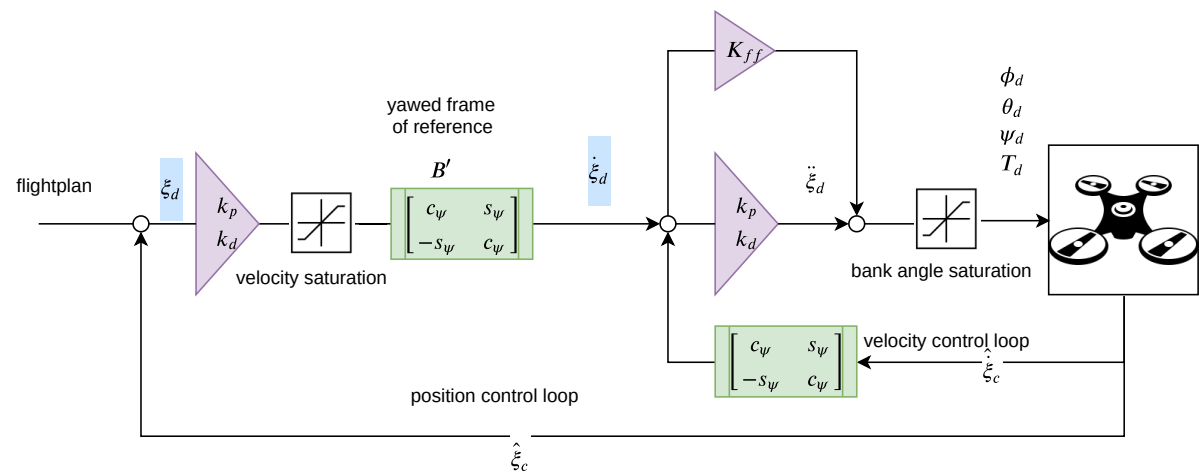


Figure 4.2: inputs and outputs of the traditional cascaded PD control algorithm

Figure 4.2 explains the cascaded PD controller used in the QR control area. Blocks in purple indicate the parameters to be tuned to make the controller track errors robustly. The saturation blocks also need to be tuned and are present right after the gain blocks. Most of the times setpoints are so far away, that a direct control action can command the QR to perform maneuvers that exceed their flight envelopes. The saturation blocks prevent this excess demand, mapping adequate control inputs according to flight envelope and actuator constraints. The green boxes perform transformations to map the entities from the world frame of reference (E) to a yawed frame of reference (B').

A benefit of something called ‘feedforward control’ comes alongside state estimation. If the drag forces that prevent the vehicle from reaching the setpoint velocity can be estimated, then the setpoint bank angles

are simply augmented with the drag forces that are expected to arrive at the next time step. This drag compensation demands the introduction of a feedforward term in the outer loop control to acknowledge the contribution of lateral accelerations during the lateral position control (K_{ff}). A similar feed-forward compensation term is also suitable for controlling the inner loops or the moments of the QR. However, full-actuation around these axes is possible, and the effect of drag on moments can be directly compensated by high feedback gain controllers [24].

As mentioned earlier this technique of PID is a simple tool that brings a lot of power in the hands of an amateur user. Although when it comes to the framework of drone-racing, PID control poses certain disadvantages:

1. Strenuous tuning procedures must be performed to find a decent control behavior. An individual tuning procedure must be performed for each decoupled loop of the cascaded controller in the above figure. This tuning procedure also includes the feedforward terms and saturation blocks. Having incorrect gains can induce oscillations in the attitude of the QR, potentially causing crashes.
2. PD controlled QRs tend to slow down near the positional setpoints (gates) and sometimes even completely stop at the gates. However there is no stopping in drone racing!. To prevent this abrupt stopping throughout the arena, some lateral velocity is augmented in the controller's velocity setpoints. This has deteriorating effects on the performance of the control loops.
3. There is no information about the expected time of completion of any maneuver. The guarantee of convergence to the reference velocity and positional setpoints is also not directly made possible.

4.2.2. Minimum snap trajectories and Differential flatness

Minimum snap trajectories minimize the higher order derivatives of acceleration, for generation of smooth trajectories through the arena. The cost function minimized here is:

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

The Euler Lagrange representation of the above optimization function can be written as:

$$\frac{\partial \mathcal{L}}{\partial x} - \frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{x}} \right) + \frac{d^2}{dt^2} \left(\frac{\partial \mathcal{L}}{\partial \ddot{x}} \right) - \frac{d^3}{dt^3} \left(\frac{\partial \mathcal{L}}{\partial x^{(3)}} \right) = 0$$

The polynomial based trajectory that is generated after solving the Euler-Lagrange optimization yields:

$$x(t) = c_5 t^5 + c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t + c_0$$

Boundary values can be substituted for these polynomials and its derivatives by knowing priori conditions of the arena. They can help estimate the co-efficients of the above trajectory polynomial. After the co-efficients are forward-substituted back to the polynomial, a minimum snap trajectory throughout the arena can be found out (see Figure 4.3).

Differential flatness is usually coupled with minimum snap algorithms for mapping the reference states generated by the trajectory polynomial to the input commands at each time instance ($\phi_d(t), \theta_d(t), \psi_d(t)$). This technique, despite being able to make a lot of contributions in this area of planning, control and differential flatness, introduce some problems if used in the current drone-race pipeline:

1. Path planning can be performed accurately in stationary scenarios (e.g. from the take-off point). Although, these trajectories need to be re-planned mid-flight when the QR moves away from its desired trajectories. In the dynamic re-planning scenario, if the QR is subjected to different distances and terminal time (referred to as segment times for time intervals between gates), co-efficient identification becomes very sensitive, yielding unstable solutions. Unstable polynomial solutions generated after differential flatness can command the QR to perform incorrect maneuvers. This makes re-planing mid-flight almost impossible using the standard constrained solvers that are frequently presented in literature.
2. The path planning algorithms mentioned in literature minimize jerk, snap or higher orders of derivatives of acceleration. The primary objective, however in case of drone-racing, is to minimize the total time of the maneuver.

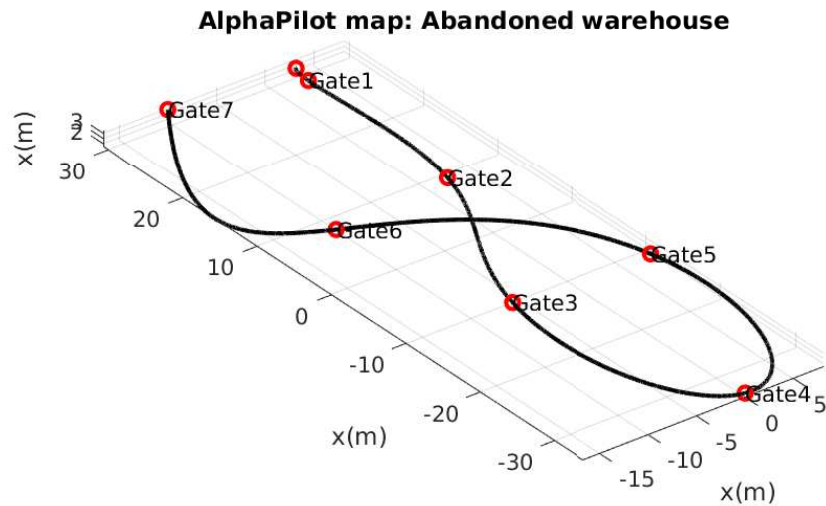


Figure 4.3: Minimum snap trajectory generated by Anoosh Hedge, MAVLab AlphaPilot team.

3. This method does not always make use of the entire flight envelope, it instead caps the maximum commanded velocities, while etching out a non time-optimal (but snap-optimal) trajectory. According to Pontryagin's principle, the current method of minimum snap and differential flatness does not qualify for time optimality. (Pontryagin's principles widely discussed in [23] give the necessary and sufficient conditions for ensuring time-optimality, further discussed in the recommendations section of this thesis).

The only possible solution to reduce the problems caused due to dynamic re-planning is to have a static polynomial based trajectory fixed beforehand. Instead of using feedforward based differential-flatness, the QR must track static trajectories using PD control algorithms mentioned in the above section. Although, the problem of time-optimality still remains unsolved.

4.2.3. Model predictive control

Model predictive control (MPC) has been an attractive solution for dynamical systems for achieving control objectives. Optimal solutions returned from MPC controllers can be used as control inputs for the upcoming time steps, gradually leading to the completion of the control objective. In MPC controllers, the number of quadratic optimizations performed per second are proportional to the non-linearity involved in the process model and the number of disturbances and measurements that arrive. Every time a control action is applied, MPC controllers rerun optimizations to predict the optimal control action that can be applied in the next discretized step.

The relevance of MPC in the field of QR control, especially for racing drones is established by a recently published paper that ranks the control strategies used by the best drone-race teams around the world [9]. However a couple of shortcomings are associated with their simulator-based MPC approach, especially its implementation on a computationally lite processor.

The type of optimal control problem the drone-race framework needs to solve is non-linear in nature, with both equality and inequality constraints. The equality constraints come from the dynamics of the QR that are worked out in detail in the earlier chapter, while the inequality constraints come from the actuator

constraints and flight envelopes.

$$\begin{aligned}
 & \underset{\xi(\cdot), \dot{\xi}(\cdot), u(\cdot)}{\text{minimize}} && T \\
 & \text{subject to:} && \\
 & \forall t \in [0, T] : \dot{\xi}(t) = v(t) && \\
 & \forall t \in [0, T] : \ddot{\xi}(t) = {}^E R_B (F_T^B + f_D) - g z^E && (4.1) \\
 & t = 0 : \xi(0) = \xi|_{t=0} && \\
 & t = T : \xi(T) = \xi_d|_{t=T} \quad (\text{desired states}) && \\
 & \forall t \in [0, T] : lb \leq u(t) \leq ub &&
 \end{aligned}$$

The symbols used in the equations are described in the earlier chapter. The cost function ‘T’ is a free-variable of terminal time and is called a Mayer term in the optimization context. The constraints are simply the dynamics of the QR. The other constraints are the initial and terminal states of the maneuver. In dynamic conditions, the terminal state constraint can be replaced by a user’s reference input.

Many different solvers can be used to obtain an optimal control sequence for the above problem. The most commonly used is the ACADO toolkit (open source algorithm collection for Automatic Control and Dynamic Optimization) [12], which uses qpOASES solvers from the ABB research group [7]. Apart from ACADO’s open source toolkit, FORCES takes up the challenge of running optimal controllers on real-time embedded systems[15].



Figure 4.4: AMZ Racing’s driverless car runs the FORCES optimizer by embotech [15]

Coming back to optimal controllers for QRs, ACADO toolkit’s cpp version is tested using the equations stated in Equation 4.1. A quick run of QR dynamics on it yields Figure 4.5. The velocity plot in the second row of Figure 4.5 resembles a familiar saturated acceleration-deceleration duo, which satisfies Pontryagin’s minimum principle. The principle suggests that every time optimal maneuver includes saturated input sequences and an optimal switching time when the input should be banded between its lower and upper bounds. It can be seen from the velocity ‘y’ plot of

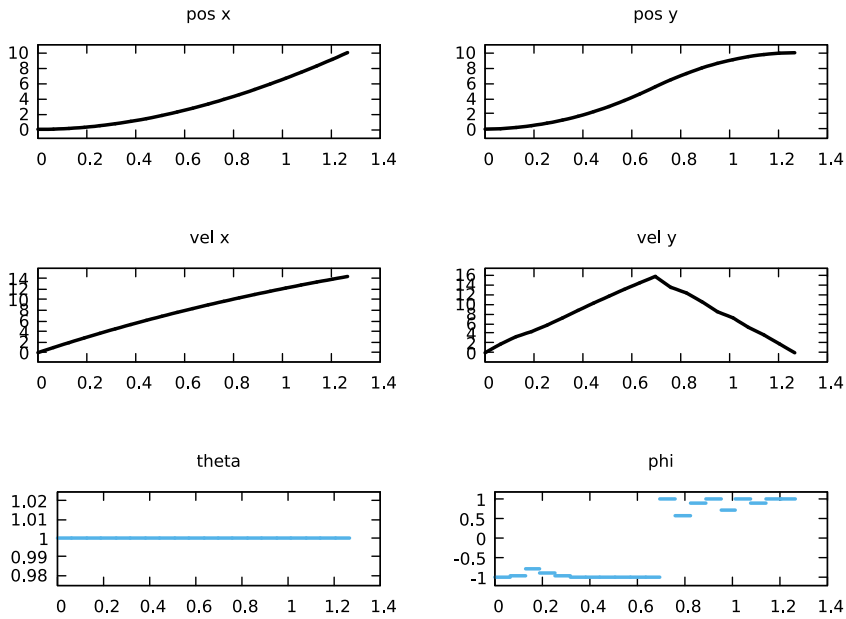


Figure 4.5: Bang-bang input sequence on ϕ to reach desired states in minimum time

Figure 4.5, that the QR accelerates longer than it decelerates in the given maneuver. This is because of the

presence of drag forces on the body of the QR, which demands the QR to accelerate for higher durations than decelerate in order to reach the target with zero-velocity. In spite of these results, MPC controllers miss out on the following:

1. The implementation of an optimal-time cost function using ACADO on real-time embedded processors is not currently supported.
2. The MPC controller accounts for costs even in intermediate states along the trajectory, whereas drone-race poses only two objectives. (a) of reaching the center of the gate with zero sideways velocity and (b) doing so in minimum time. Hence only the terminal cost of a maneuver is important in the optimization. Including intermediate states helps improve the quality of trajectory tracking, but at the cost of increased computational effort.
3. The above equations stated in Equation 4.1 fall under the non-linear MPC class. Non-linear MPC increases tracking accuracy, but adds more to the computational complexity by having to perform linearizations and sequential optimizations at every time step.
4. MPC methods in ACADO's QR equations do not take advantage of the linear drag model's dynamical equations established in the earlier chapter and hence can only be used in short time horizons.

4.2.4. Requirements for a better QR position controller

To propose a set of requirements for a better controller that fits well in the framework of drone racing, a few shortcomings of the aforementioned control techniques if used for drone racing, are listed below:

1. The gains inside cascaded PD control become tiresome to tune and bound. The ideal controller must be intuitive to tune and at the same time must respect the actuator bounds.
2. PD control misses out on indicating the possibility of meeting the control objectives. There is no means for indicating if PD control can help converge to the desired states within a given time interval. The ideal controller must be able to indicate if the desired state is reachable after respecting the actuator bounds, dynamic constraints and terminating time.
3. Minimum snap and PD methods both fail to guarantee time optimality. The ideal controller should perform time-optimal maneuvers, while (ideally) saturating the QR's input bounds.
4. MPC methods introduce complexity by introducing intermediate states, discretization and sequential quadratic programming techniques to deal with non-linearities. The current codegen from ACADO also does not support optimizing for terminal time. The proposed controller must not take the luxury of directly using the MPC framework, but instead propose its own method of reducing computations and run on a real-time system.
5. The ideal controller cannot keep the CPU busy for too long, as instructed by the single threaded Paparazzi autopilot. As a result, the hard real-time deadline for yielding an optimal solution must not exceed 10ms.
6. The ideal controller should be able to converge to the control objectives and terminal states irrespective of different starting conditions of different lateral velocities, heading and QR position. To make the case more specific, the ideal controller must enter only from the front-panel of the gate maintaining a particular desired velocity.

It can be observed that optimal controllers are the way to go, and suit well within the framework of drone-racing. They can guarantee time-optimal maneuvers and convergence to desired positions and velocities at the respective gate location. However for being able to execute them on a real-time embedded system of a QR, the optimization needs to be computationally efficient. The next section discusses how a rudimentary MPC that fits well in MAVLab's drone-race pipeline can be executed on a real-time system.

4.3. Proposed controller

The following paragraph helps revise concepts from the prior chapter while bridging the gap between control and estimation. Differential equations established in Equation 3.15 help propagating the model in time and predicting the states held at any point in the future. Free body diagrams (Figure 2.6) help in accounting the forces and torques on the body. The forces can be scaled to give the accelerations. Simply dead reckoning the accelerations (performing first order Euler integration), to obtain lateral positional information has yielded reasonable results over a short time-span (see Figure 3.19).

Robust state estimation performed in the earlier chapter can help define the system dynamics with higher accuracy leading to less drift in the predictions from the ground truth. As a result due to less drift, the re-planing or re-scheduling of the control inputs can be done sparsely. It is also now possible that a single quadratic optimization function call can schedule accurate control inputs for a longer time horizon.

Predicting trajectories and possibilities of meeting control objectives is only made possible by good state estimators, which was exactly the purpose of the last chapter. As famously put by John Wooden, "*Without proper self-evaluation, failure is inevitable*". which in this context could translate to: without state estimators, the objective of meeting control objectives is far fetched.

It can be observed that optimal controllers are the way to go, and suit well within the framework of drone-racing. They can guarantee time-optimal maneuvers and convergence to desired positions and velocities at the respective gate location. However for being able to execute them on a real-time embedded system of a QR, the optimization needs to be computationally efficient. The next section discusses how a rudimentary MPC that fits well in MAVLab's drone-race pipeline can be executed on a real-time system.

4.3.1. A brief introduction to optimal control

Quadratic optimization forms the base of MPC and is used in various areas, starting from portfolio optimization for maximizing returns in the finance market to the area of control systems [4]. Having no intention to jump into details of this vast field of optimization and stochastic systems, this section mentions an example of an optimal control benchmarking performed at ETH-Zürich. Also, an interesting account from history is added to the appendix of this report (see Section A.4) to support the motivation behind pursuing optimal control.

A non-trivial example of optimal control in the area of QRs is given by [11], in which they send commands for displacement along the z^E axis in a time-optimal manner. A trivial solution to this problem would be to simply increase the thrust optimally and slow down close to the setpoint while the commanded angular velocities are zero. However, their research finds that the time optimal solution was to pick up a lot of ascent speed by enormous increase in thrust, except when it reaches close to the setpoint, it must flip upside down by commanding some angular velocity. Facing upside down, it now uses the force of gravity and the thrust generated by its propellers (which are also facing downwards) to decelerate from the enormous acceleration it gained in the initial stage. It finally flips upside up near the setpoint and reaches the desired setpoint in a time-optimal fashion. However, in case of this thesis, the maneuvers are limited to the x-y plane since there are limited altitude changes in the drone-racing arena. Also, the maneuvers are limited by the flight envelope of not banking more than 35 deg.

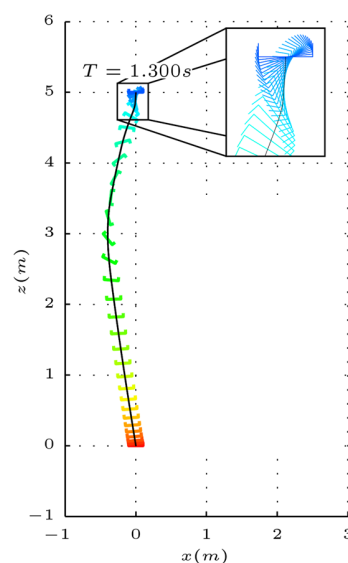


Figure 4.6: Time optimal ascent to height of 5m [11].

4.3.2. Proposed algorithm for control of racing drones

MPC problems are only a subset of optimal control problems that have a fixed time horizon [12]. As a result, the goal of making time-optimal trajectories is difficult to realize with an MPC controller. A different subset of optimal control problems that minimize time should hence be explored. A new technique that can have the benefits of optimal control, but still be computationally lite, is in search. This section puts forth a proposal for such a controller. Since the controller can be explained mathematically and graphically, equations and derivations alongwith a flowchart (see Figure 4.8) is made. An interesting intermezzo included in the paper from Magicc Lab at BYU [18] models the effect of drag on acceleration as a mass damper system.

- \dot{u}/\dot{v} : acceleration of the QR in the inertial frame along the x^E/y^E direction respectively.
- u/v : velocity of the QR in the inertial frame along the x^E/y^E direction respectively.
- μ : drag co-efficient along the body x^B axis.

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -g \sin \theta - \frac{\mu}{m} u \\ g \sin \phi \cos \theta - \frac{\mu}{m} v \end{bmatrix} \quad \text{taking the Laplace transform yields, } u(s) = \frac{-gm}{\mu s + 1} \theta(s) \quad (4.2)$$

The representation of linear drag equations as transfer functions in Equation 4.2, provides the motivation to also find the state space representation to it. Let lateral velocity be given as the derivative of position $\dot{x} = u$, lateral acceleration for our case can be written as $\ddot{x} = -g \sin \theta - k_d \dot{x}$

$$\begin{bmatrix} \ddot{x} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} -k_d & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ x \end{bmatrix} + \begin{bmatrix} g \\ 0 \end{bmatrix} \sin \theta$$

This is similar to the model that was discussed in the earlier chapter, under MAVLab's method for state estimation, but this model is represented in a state space form. It also directly compensates for drag forces in the world frame. This can be extended in both the lateral positional axes, i.e. along x^E and y^E . Also, another simplification can be made since the control input is bounded due to actuator constraints. The control commands sent in case of a Bebop¹ or the simulated drone on ROS² - does not bank more than ± 35 deg. Hence the approximate linearization of $\sin \theta = \theta$ and $\sin \phi \cos \theta = \phi$ can be made around this linearization point. The combined state-space model after linearization can hence be given as:

$$\dot{x} = Ax + Bu \quad (4.3)$$

$$\begin{bmatrix} \ddot{x} \\ \dot{x} \\ \ddot{y} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -kd_x & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -kd_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ x \\ \dot{y} \\ y \end{bmatrix} + \begin{bmatrix} 9.81 & 0 \\ 0 & 0 \\ 0 & 9.81 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \phi \end{bmatrix} \quad (4.4)$$

The following section focuses on formulating a basic optimal control problem using the state space formulation above. First, the above state-space is discretized using zero order hold (ZOH), with a sampling time of $h = 0.1$ seconds.

$$\begin{bmatrix} \dot{x}_{k+1} \\ x_{k+1} \\ \dot{y}_{k+1} \\ y_{k+1} \end{bmatrix} = \Phi \underbrace{\begin{bmatrix} \dot{x}_k \\ x_k \\ \dot{y}_k \\ y_k \end{bmatrix}}_X + \Gamma \underbrace{\begin{bmatrix} \theta_k \\ \phi_k \end{bmatrix}}_U \quad (4.5)$$

After obtaining the solutions from this discretized state-space, the control inputs are supposed to be sent over to the QR at the same rate, i.e. at 10 Hz. For simplicity in symbol conventions, the discretized state-propagation matrix Φ and Γ are represented as A and B respectively.

Estimating time horizons: The time horizon is initially guessed using the expression:

$$T_0 = 2 \times \frac{\Delta pos}{vel_{avg}} = 2 \times \sqrt{\frac{(\xi_c - \xi_d)^2}{(\dot{\xi}_c - \dot{\xi}_d)^2}}, \quad \text{and propagation steps: } N = T/h$$

Bang-bang control makes a system reach its desired states in a time-optimal manner. (Detailed explanation of bang-bang control in the recommendation section of this thesis). Applying bang-bang means sending control signals that are (almost) always saturated. The %bang can be estimated using the following formula:

$$\%bang = \frac{\sum \cup}{N \cdot ub} \times 100, \quad \text{where, } \cup \in \{lb, ub\}$$

where lb, ub are the lower and upper bounds of each element in the input sequence respectively. Instead of doing this manually, Lagrange multipliers at the solution of the quadratic program (λ) can be used to

¹Implementation of this controller on Bebop is pushed here: https://github.com/tudelft/paparazzi/tree/drone_race_opt/

²Implementation of this controller on MIT-FlightGoggles is pushed here: <https://github.com/nilay994/superstate/tree/master/mpcROS>

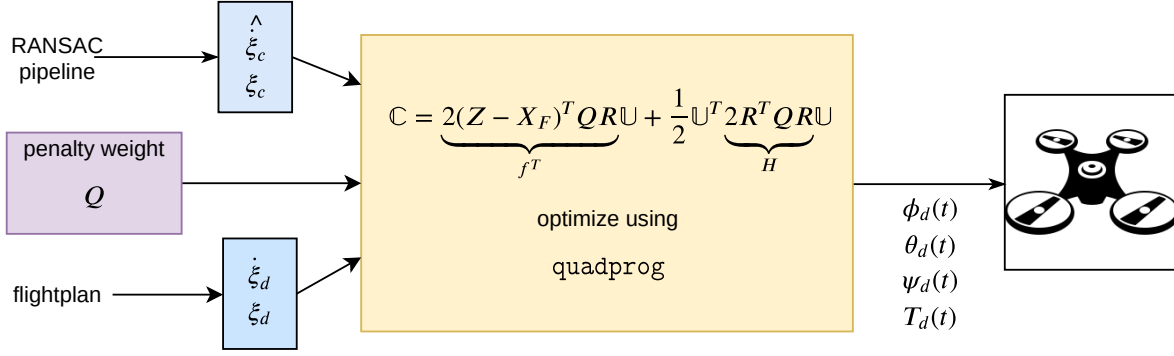


Figure 4.7: Inputs and outputs of the optimal control algorithm, symbols explained in Equation 4.6

keep a watch on closeness to constraints. However, they don't provide a good metric of % of banging. The next candidate time horizon is divided by an exponentially rising iterate given by: $iterate = iterate \times 1.2$, $iterate(t_0) = 1$. This iterative scaling provides a good candidate N for time horizon selection and usually can converge to minimum time horizon within a maximum of 3 iterations. The quadratic optimization adapts automatically to the new time horizon while not exceeding reasonable λ .

$$T_{iterate} = \frac{T_0}{iterate}$$

Propagate states to prepare for quadratic optimization: The terminal state (X_N) must be parameterized in terms of the initial state (X_0) and the input sequence U . This can be done by:

$$X_1 = AX_0 + BU_0$$

similarly, at the consecutive time step,

$$X_2 = AX_1 + BU_1$$

substituting X_1 in the RHS of X_2 ,

$$X_2 = A^2 X_0 + ABU_0 + BU_1$$

generalizing for N time steps,

$$X_N = A^N X_0 + A^{N-1} BU_0 + A^{N-2} BU_1 + \dots + BU_{N-1}$$

regrouping the terms and separating the input vector U_i ,

$$\therefore X_N = \underbrace{A^N X_0}_Z + \underbrace{[A^{N-1} B \quad A^{N-2} B \quad \dots \quad B]}_R \underbrace{\begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_{N-1} \end{bmatrix}}_{\text{to find}} \therefore X_N = Z + RU \quad (4.6)$$

where U is the input sequence to be optimized, X_N is the terminal state reached using optimal input sequence. X_D is the desired state or the reference state instructed by the user.

Cost function: Unlike the traditional MPC methods, the cost function (see Equation 4.6) is only subject to the terminal penalty of going through the gates. A penalty matrix of MPC methods usually also holds an "integral cost", which is minimized mid-flight by penalizing the deviation of intermediate states. If the optimizer can not promise lower values of the cost function after minimization, then it symbolizes that the maneuver would not be possible given the input bounds of the actuators (U) and the initial conditions (X_0). The proposed approach skips the idea of integral costs and only focuses on achieving the desired state at the end of the trajectory, without worrying about the trajectory taken in the middle before reaching the desired state. As a result, the cost function can be re-written as:

$$\mathbb{C} = (X_N - X_D)^T P (X_N - X_D) \quad (4.7)$$

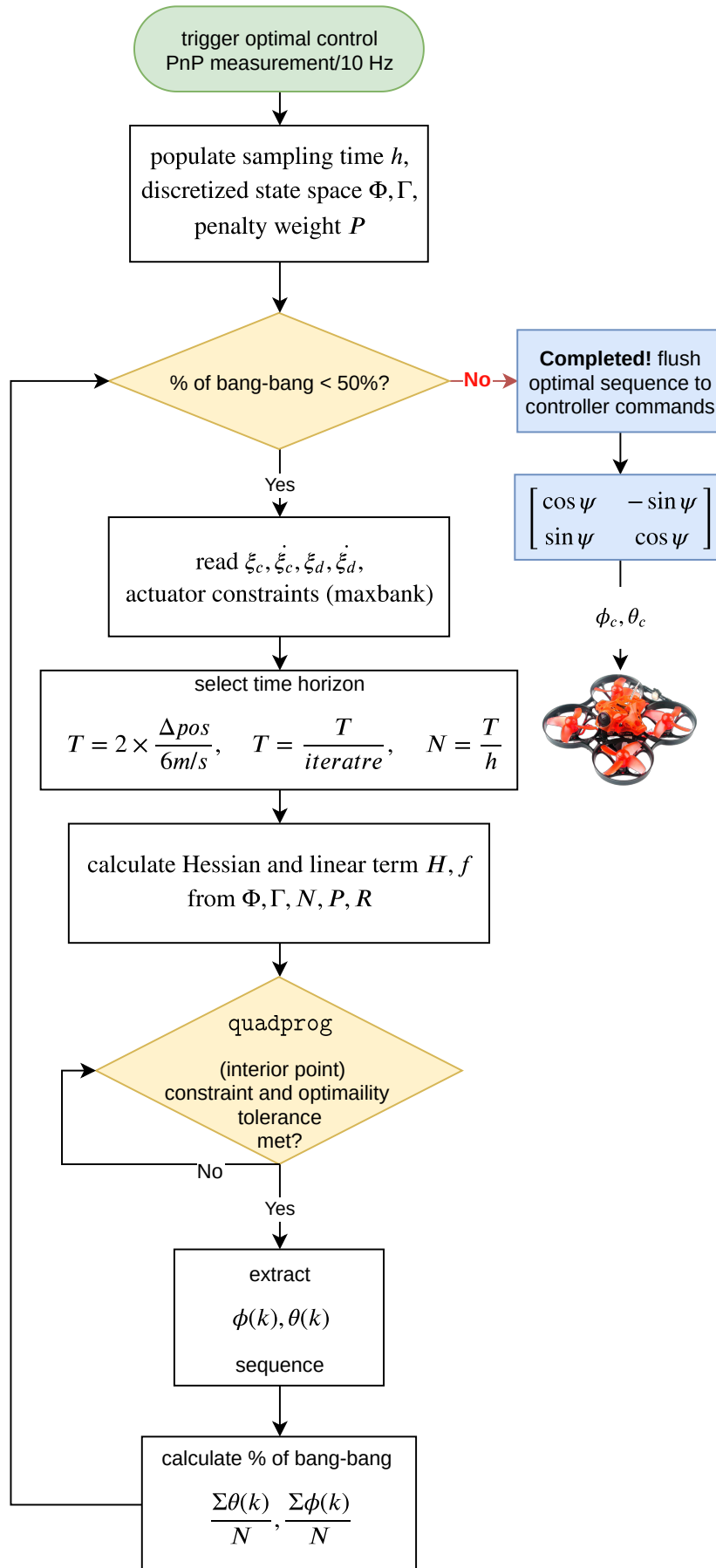


Figure 4.8: Flowchart for the implementation of the proposed controller

where deviations in position are reprimanded more than deviations in velocity, by using the penalty weighting matrix:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

Populating the matrices for starting quadprog can be done by substituting Equation 4.6 in Equation 4.7,

$$\begin{aligned} C &= (Z + R\mathbb{U} - X_D)^T P (Z + R\mathbb{U} - X_D) \\ C &= \underbrace{X_D^T P X_D}_{\text{constant}} - X_D^T P X_F - X_F^T P X_D + X_F^T P X_F \end{aligned}$$

Constants in a quadratic program optimization can be ignored,

$$C = -2X_D^T P X_F + X_F^T P X_F$$

Substituting back $X_F = Z + R\mathbb{U}$,

$$\begin{aligned} C &= [(Z + R\mathbb{U})^T - 2X_D^T] P (Z + R\mathbb{U}) \\ C &= \underbrace{Z^T P Z}_{\text{constant}} + Z^T P R\mathbb{U} + \mathbb{U}^T R^T P Z + \mathbb{U}^T R^T P R\mathbb{U} - \underbrace{2X_D^T P Z}_{\text{constant}} - 2X_D^T P R\mathbb{U} \end{aligned}$$

Ignoring the constants, the Hessian and linear term for quadratic optimization can be formulated as:

$$C = \underbrace{2(Z - X_D)^T P R\mathbb{U}}_{f^T} + \frac{1}{2} \mathbb{U}^T \underbrace{2R^T P R\mathbb{U}}_H$$

These terms can be programmed into an interior point based quadratic optimizer in tools like quadprog (MATLAB)/qpOASES (C++) which follow the structure:

$$\min_x \frac{1}{2} x^T H x + f^T x \quad \text{subject to:} \begin{cases} A \cdot x \leq b \\ A_{\text{eq}} \cdot x = b_{\text{eq}} \\ lb \leq x \leq ub \end{cases}$$

Hessian conditioning: The Hessian term (H) inside the quadratic cost function should be positive semi-definite as stated by the requirement of standard quadratic program solvers. A direction for the next step in the optimization becomes immediately possible for positive semi-definite Hessians. To ensure this, a scaled identity matrix is added to the Hessian term via the process of 'ridge regression' [8]. On our case, it is observed that adding an un-scaled identity matrix to the Hessian, has an impact on the rate of roll and pitch commands in the input sequence generated. However, since it is desired to have bang-bang inputs, a slow rate of commands is not desirable. Hence, the diagonal matrix to be added is scaled to bang-bang quickly between saturation limits.

$$H = \frac{H' + H'^T}{2} + 0.15I_{2N \times 2N}$$

These modifications to the Hessian possibly lead to fewer KKT optimality iterations. **Uniqueness:** It is observed that the Hessians are always positive-definite and have positive eigen-values irrespective of the initial and desired states of the QR. Because of these properties, the quadratic program optimization gives a solution which is the global minimum of the objective function given these constraints. As a result, the %bang optimal control sequence and the trajectories generated are unique, and no other solution exists, which can make the QR follow the same time-optimal trajectory.

Control input assignment After the quadratic program is solved after reaching KKT optimality conditions, control input vector \mathbb{U} can be parsed for the pitch (θ) and roll (ϕ) commands. However, it happens that all the calculations that were performed earlier, are in the B' reference frame (see Figure 2.5). To be able to assign correct control inputs to the QR, the commands must be provided in the Earth reference frame. This can be done using a simple yaw-rotation transformation:

$$\begin{bmatrix} \phi_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} \cos \psi & -\sin \psi \\ \sin \psi & \cos \psi \end{bmatrix} \begin{bmatrix} \phi'_k \\ \theta'_k \end{bmatrix}$$

Using this transformation, the control sequences generated by quadprog would guide the QR through the correct trajectories, irrespective of the heading of the QR.

It can also be noted that this method not only generates the command inputs at 10Hz, but it also generates state trajectories, or the reference path to be followed by the QR. This additional information about the reference path is directly discarded at the moment. In case the feed-forward command inputs are not sufficient to ensure a safe trajectory through a gate, the reference path can be used to augment some feedback information. This is well explained in [21] which uses differential flatness to combine feedback based trajectory offset information and feedforward control input.

4.4. Results

Since a few methods were described above, a comparison of the proposed controller must be made with respect to the traditional control methods. The first comparison compares the PD control methods with the proposed method for trajectory tracking. Figure 4.9, Figure 4.10 and Figure 4.11 are generated after a typical drone-race maneuver. The maneuver is performed on MIT's FlightGoggles drone-race simulator. A couple

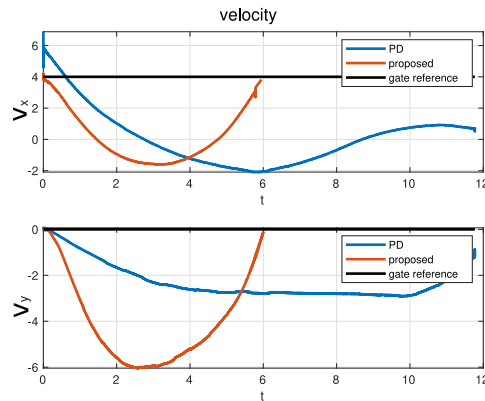


Figure 4.9: Achieving desired gate velocity of $v_x^E = 4 \text{ m/s}$.

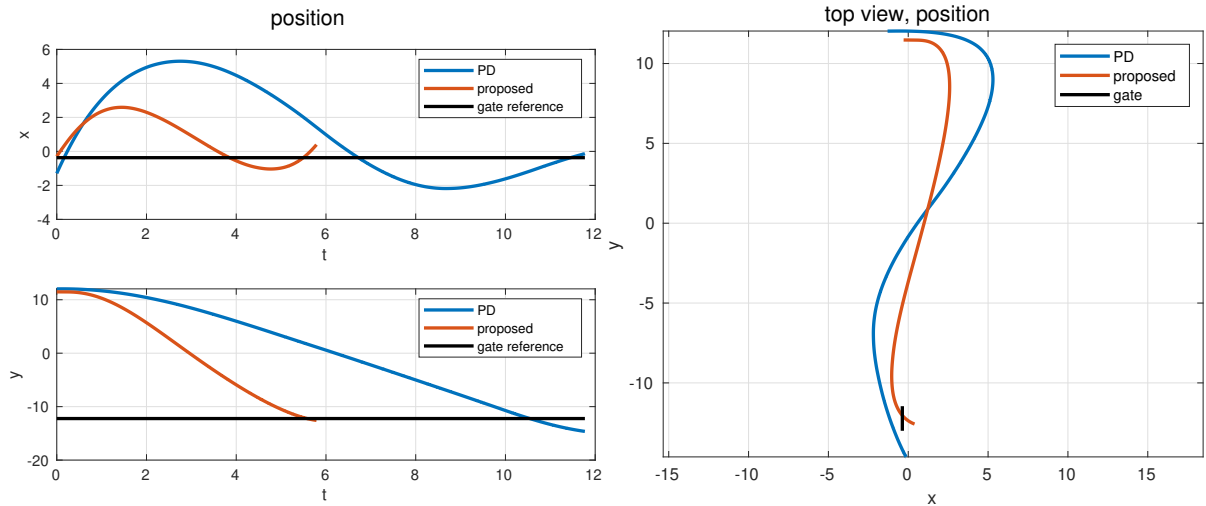


Figure 4.10: QR position with time.

Figure 4.11: QR position from the top view of the arena.

of observations can be made after this experiment:

1. PID control methods are not able to meet the requirement of forward velocities at the gate. The proposed optimal method performs the required maneuver along the x-y plane and make the QR converge to the desired gate velocity of $v_x^E = 4 \text{ m/s}$. It can also be observed in Figure 4.11, that the optimal control method meets the requirement of passing through the front panel of the gate with forward velocity.

2. It can be seen from Figure 4.10 and Figure 4.9 that the proposed controller reaches the reference within a shorter time interval (almost half the amount of time) as compared to the PD control methods.
3. (not seen in the plots) PD controllers introduce a very oscillatory behavior when the QR arrives closer to the gate. This can be solved by frequently switching the setpoints and making the QR move throughout the arena for all times.

Another set of maneuvers are made to compare the proposed method with the minimum-snap method.

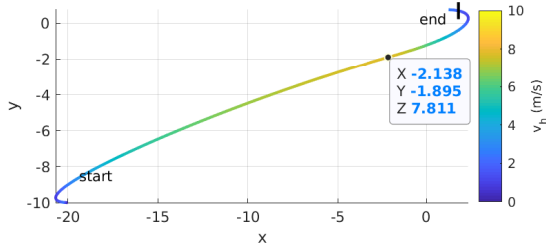


Figure 4.12: Proposed method makes the QR reach the end state with a maximum velocity of 7.8 m/s.

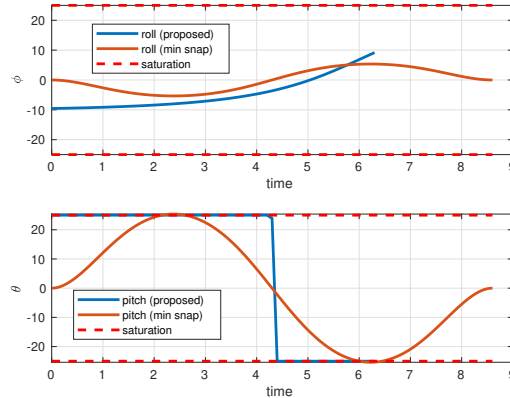


Figure 4.13: Commands sent to the QR when using (a) proposed method (b) minimum snap method.

A couple of observations can be made after this experiment:

1. The proposed method converges to the setpoints (reaches the gate) earlier than the minimum-snap method. This can be seen in Figure 4.13, where the plots in blue already converge at 6 seconds while the plots in red (minimum snap) are still generating commands till 8.5 seconds through the flight.
2. The proposed method uses entire flight capability by giving inputs that are almost close to the saturation of the actuators. It can be seen from Figure 4.13 that minimum snap methods do not use the entire flight capability, since there is unfilled area between the pitch commands (blue plot) and the saturation limits in red.

4.5. Profiling

ACADO toolkit's .cpp version comes with an interesting feature of codegen, which can generate the calculations for discretization, linearization and Runge-Kutta integration as a standalone code that can be flashed onto an MCU. With this feature, the real-time embedded system can perform some hard-coded calculations before invoking the qpOASES solvers. Even after these efforts to minimize the time taken for prior calculations, discretization, linearization and Runge-Kutta integration take considerable CPU time [13] out of the total time of that iteration. However, codegen currently does not support optimizations when Mayer terms are included in the dynamic optimization. As a result, the trajectories and control inputs generated in Figure 4.5 can not be generated on a real-time platform.

As discussed in the requirements section for the proposed estimator, the execution time of the optimal control algorithm is critical. The optimal control sequence must be generated within strict time bounds while calling the subroutine in Figure 4.8. The proposed solution makes some assumptions for simplifying and making the trajectory generation problem faster by:

1. Reducing external overhead: Avoiding use of any external libraries for Model Predictive Control. The C code of the proposed controller uses only a single library of qpOASES [7] to perform quadratic optimization.
2. Linearization and Discretization: Both these steps are executed upfront at the initialization of the proposed solution. Discretization is only done when the drag parameters of the proposed state estimators

have changed. Linearization at every time-step is not required, since the process model is already linearized while drafting the algorithm.

3. Allocation for system matrices: Only the Hessian and linear term that are required for qpOASES are stored in the memory. While they store single precision floating point elements, Their size is $2N \times 2N$ and $2N \times 1$ respectively. If the target is 20 metres away, the time horizon is almost set to 4 seconds, assuming the average velocity of 5 m/s. This leads to $N = 4.0/0.1 = 40$ samples. The hessian and linear term would hence have a size of 80×80 and 80×1 respectively. These sizes are acceptable while running on ARM Cortex A9.
4. Zero Order Hold: System dynamics are discretized with the maximum possible sampling time to reduce the size of the matrices above. System identification of attitude commands (see Section A.2) gives a rise time of around 0.1 seconds and a settling time of 0.4 seconds. The sampling period should have been $h = 0.4$ seconds, however it lead to a very coarse set of commands being sent out to the QR. To generate ample number of commands to the QR, $h = 0.1$ seconds is chosen. However, if sampled at such higher frequencies, the state space must be made cognizant of the transient responses of the attitude of the QR. This is added to the list of recommendations for taking this work ahead.

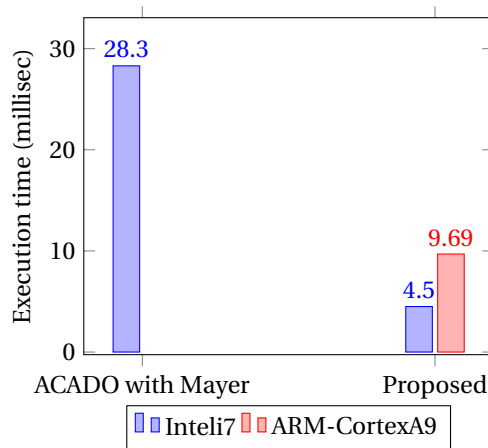


Figure 4.14: Profiling the time-optimal ACADO algorithm (left) and the proposed optimal control algorithm (right)

4.6. Discussion

PD control requires a constant inflow of measurements for assigning control inputs. Optimal control on the other hand, would have already scheduled control inputs at the arrival of the previous measurement. As a result, optimal control decouples itself from the high frequency feedback loop of Figure 4.2. It also reduces the expectations from the vision pipeline of delivering high frequency measurements and state estimates. This decoupling hence frees up computational resources to perform other calculations.

Comparison between different control techniques:

After proposing a new optimal control technique, it would be nice to compare all the techniques in the earlier sections with the new proposed technique. The comparison could highlight the importance of the new optimal controller in the drone-race pipeline.

↓ features, control method ⇒	PD	Minimum Snap	ACADO	Proposed Optimal
Perception aware (ψ only)	✓	✓	✓	✓
Change in model parameters	✓	✓	✓multi-objective optimize	✓re-discretize
Time optimal guarantee	✗	✗	✗not available in codegen	✓closeness to constraints
Convergence guarantee	✗not default, but possible	✗	✓	✓residual of optimization
No hectic tuning required	✗	✓	✓	✓only heuristic
Saturation aware control	✗	✗	✓	✓
Computational time taken	low	high	high	medium
Trajectory generation	✗	✓	✓	✓

Table 4.1: Comparison between different control techniques

All control techniques can be made perception aware to some extent. Perception awareness implies augmenting vision based objectives of field of view, inside the optimization of the trajectories. It must be noted that when it comes to optimal control, perception awareness comes at a cost of optimization of other objectives as well [6]. For instance, it might not be ideal to pitch to saturation limits, if the gate is not visible in the image frame of the monocular camera. If max-bank cannot be achieved in order to keep the gate in view, time-optimality can not be guaranteed (Pontryagin's principle - explained in the recommendations). Keeping the gate in view is critical since being unable to see the gates for long can cause state estimators to drift, increasing the chance of crashing at the boundaries of the gate.

All the control techniques can adapt to change in model parameters. The parameters that can vary include the drag and thrust parameters in the force model along all the three translational axes.

- Since PD control has only one term in its feed-forward that is drag dependent and the rest of PD control is not model based, it can adapt well to change in model parameter k_d .
- Since ACADO comes with multi-objective optimization, a recursive least squares based estimator can be augmented into the optimal control cost functions. This enables ACADO to estimate the change in k_d after a considerable number of measurements are made. The new estimated k_d is used from then on to perform MPC.
- Minimum snap methods are not constrained by the model of the QR, but only by the snap in the trajectory. Hence they remain unaffected to change in model parameters. However, the drag parameter in the differential flatness mappings must be updated.
- While using the proposed method as the control module, Equation 4.5 must be re-discretized everytime the estimated drag parameter changes.

However, it can be concluded that the presence of optimal control is critical in the drone-race scenario for meeting control objectives of time-optimality and high forward velocity through the gates. Compared to a state-of-the-art optimal control technique, the proposed algorithm leads to faster trajectories and consumes less computational power onboard.

However, the following limitations still remain with the current approach:

1. Non-linearities are not addressed. When flying at near saturation angles, the banking of the QR is well outside the linear region $\theta \geq 20$ deg, $\phi \geq 20$ deg. As a result, linearization about the point $\theta = 0$ and $\phi = 0$ do not hold correct. The consequences of still using the state space in Equation 4.4 can only make the QR track the trajectory in a coarse manner. However, this issue can be solved by asynchronously re-planning for the same desired states mid-way through the flight.
2. This process is iterative, and performs a iterative descent to find the candidate time horizon. However, this could directly be done by switching to Sequential Quadratic Programming (SQP) based algorithms instead of the current interior point algorithm. It also requires addition of the 'Bolza' term of the free-variable of time horizon in the cost function.
3. This method assumes that the altitude of the QR is the same as the altitude of the desired waypoint. The proposed method currently can only make near time optimal trajectories in a 2D space. However, since autonomous drone-racing rarely requires frequent change of altitude, this can be allowed.
4. If the state space is discretized at smaller time intervals than the settling time of attitude commands, then it must be made aware of slow attitude dynamics by augmenting attitude based states in the state-space of Equation 4.6.
5. The qpOASES library was not built with architecture specific flags turned on. If extra efforts are taken with the `Makefile`, the results in the profiling section can definitely improve.

Conclusion and Recommendations

5.1. Conclusions

Three requirements of an ideal control sub-module were stated in the introduction of this thesis. The first requirement was to be able to possess accurate state estimates of the QR position over long time horizons. The second requirement was to be able to design time-optimal trajectories for QRs to be able to fly the track in minimum time. The third requirement expected these state estimators and optimal controllers to run within milliseconds on a real-time embedded system platform. Each of these requirements are re-assessed in the subsections below. The problem statement of this thesis is restated here for evaluation:

Develop a robust state-estimation framework that minimizes drift over time while reducing expectations of frequent corrections from the vision sub-module, and design an optimal control method that allows a QR to perform time-optimal maneuvers while being computationally inexpensive.

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

(a) Accuracy

The accuracy of various estimators were discussed in Chapter 3. Apart from the requirement of being most accurate, Chapter 3 also added additional requirements of causality and time invariance to the state estimators. Causality prevents sensor-fusion techniques of state estimation, while time invariance prevents the QR models from changing while optimal control horizons are active. These requirements make the state estimators of QRs compatible for optimal-control.

Chapter 3 concluded that the proposed estimator's position error metrics with root mean square error of 0.22 m over 4 seconds (along the x-y plane), are good enough for its use in the drone-race pipeline. These predictions were made by only using the information about initial states and the future inputs to the QR. The requirement of causality and time-invariance were also met, since there is no presence of sensor measurements in the representation of Equation 3.15.

(b,c) Optimality

The requirement of generating and tracking time-optimal trajectories while using saturated control inputs was made in the introduction. Time-optimal maneuvers can make the QR complete the course of the arena within minimum time. Another set of requirements were added in chapter 4, which include intuitive tuning of gains, information about convergence to setpoints and finally a requirement of entering gates from the correct direction.

As seen from Table 4.1, these requirements are successfully satisfied by the proposed algorithm. A QR can fly through designated gates from the correct direction, irrespective of its starting conditions. It can also indicate how far the generated trajectories are from the desired states from the residual cost after minimization. It is intuitive to tune the penalty matrix of the proposed controller, since the user can directly select the penalty on position violation or velocity violation at terminal state.

The proposed algorithm in Figure 4.8 is only able to generate semi time-optimal trajectories at the moment. The iterations for time-optimality are stopped if the control sequence of the current iterate are saturating more than 55% of the time. Asking the controller to saturate for a long time has an impact on the terminal cost and also cause higher trajectory tracking errors. These cause of these errors is suspected to be because of using QR models which are always linearized about $\phi = 0, \theta = 0$, while the QR is flying in non-linear regimes.

(d) Speed

The introduction section lists a hard real-time constraint for solving the trajectory generation problem on a computationally lite processor of a QR. This was already considered while writing the algorithms in Figure 4.8. The proposed solution reduces external overhead, skips onboard linearization and discretization, uses limited memory and performs output-hold operations to buffer the low frequency commands that are sent to the QR.

Chapter 4 also gives a proof by profiling the proposed algorithm on a processor of the Bebop QR (see Figure 4.14). It can be concluded that the proposed algorithm is a good candidate for becoming the new control sub-module of MAVLab's drone racing pipeline.

5.2. Improved pipeline after this thesis

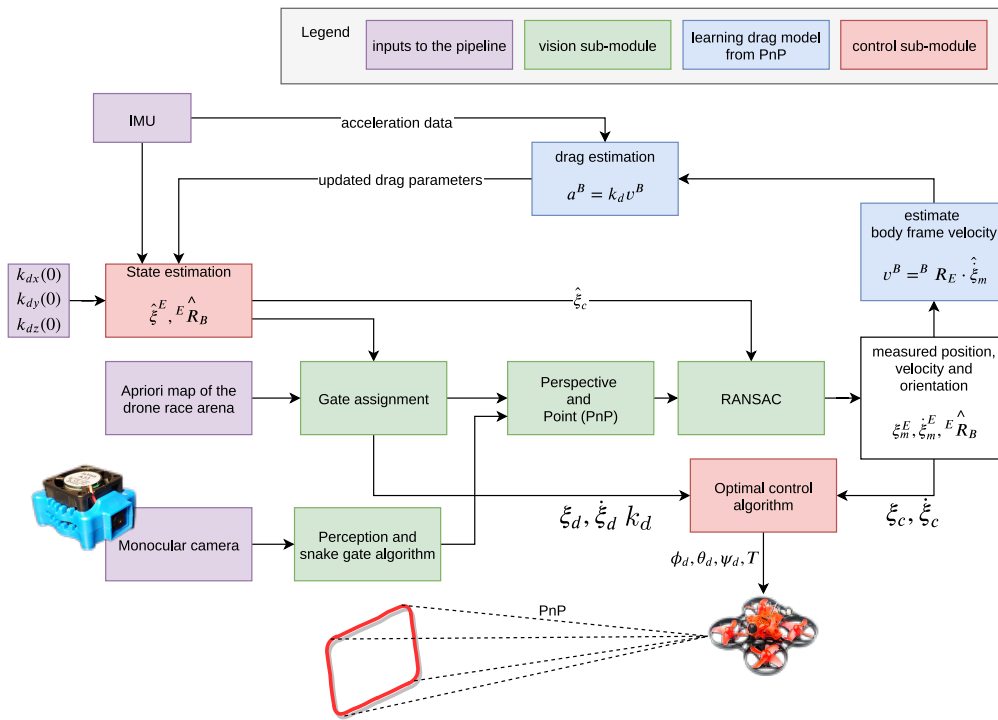


Figure 5.1: Being able to perform robust state estimation and optimal control can lead to a parameter adaptive closed-loop system.

A slightly modified pipeline can be suggested for use after various suggestions in this thesis that are made to improve the state estimators and control algorithms of MAVLab's drone racing pipeline. There is one additional contribution made by the introduction of the new pipeline. The drag parameters can now also be identified mid-flight, improving the quality of the model as we fly.

While the QR flies around using the proposed optimal controller, the vision sub-module corrects the position and velocity estimation which was made by the model of the QR. If a sufficient number of such measurements are available, the QR can update the drag parameters used inside the QR model. As a result, the tedious system identification process suggested in Section 3.3 can be skipped. The drag co-efficient update law comes with three assumptions:

1. The accelerometers have been calibrated and the readings available to the autopilot are bias-less acceleration measurements along x^B and y^B .

2. The PnP measurements must be arranged chronologically, having monotonically increasing time stamps.
3. The initially assumed drag coefficients are within realistic bounds of $0.2 < k_d(0) < 0.9$.

This new pipeline is currently under test on MIT's FlightGoggles racing-drone simulator. As a result, this thesis concludes that given the scenario of drone-racing using computationally lite autopilots, accurate state estimators and coarse optimal control algorithms are the way to go. By following this approach, state estimation is as accurate as it can be and the optimal control algorithm is not computationally intensive. Every sub-module in the control pipeline is able to meet the hard real-time deadline of the autopilot. The autopilot can safely send high frequency PWM signals to the ESC, read the IMU at high-frequencies, perform attitude control and still run these state estimation and optimal control sub-modules listed in the previous chapters.

5.3. Applications of this work

The introduction of this thesis stated that research on QRs can be extrapolated to different areas of research. This section lists them below:

5.3.1. Applications in the area of state estimation

State estimators explored in this thesis can be extended to various fields of research. This subsection presents setup of state estimation in GRACE and GRACE-FollowOn missions launched by NASA.

The primary purpose of this mission is to map the local gravitational potential of the surface of Earth. Performing state estimation on these satellites not only include the positional states of the satellite in orbit but also the gravitational field underneath. Infact, possessing accurate information of the positional states is only a secondary objective of the mission and these states are only the by-product of accurate gravitational field estimation.

Obtained gravitational estimates are used in the dynamical equations of the satellites to propagate the positional states which are essentially the orbital positions. The highlight here is, that the gravitational potential could also be an augmented state to be estimated, whose accuracy improves everytime GRACE makes a successful measurement using its lasers. This reduces the covariance in the belief of the gravitational potential states, which in turn also helps in improving the quality of the positional estimates due to a better dynamical model. This reduction in covariance and increase in quality of the estimated gravitational field goes on recursively till the resolutions and epsilons of the mission's objective are met.

This thesis aimed to carry out similar techniques of state estimation when it came to QRs. The effects of drag forces that are discussed in Section 3.3 are non-negligible while propagating the dynamics of the drone. As a result, the drag coefficients are parameterized similar to how GRACE missions parameterize the gravitational potential fields, with the objective of having better lateral positional estimates.

5.3.2. Applications in the area of optimal control

Optimal control explored in this thesis can have various applications starting from portfolio management in financial markets to power division in hybrid vehicles.

Portfolio optimization: As seen in [29], dynamic programming and optimal control can be used in the field of finance. Billions of euros are pushed daily through international markets while brokers try to optimize position and portfolios in the market while mitigating their risk and maximizing returns. Usually each action of buying, holding or shorting can be modeled as an input to the market, while the market itself can be modeled as an MDP (Markov Decision Process). Usually the MDPs do not lead to straight-forward quadratic cost functions as indicated in case of drone-racing, but they definitely have an optimal policy to reach the optimal value (maximizing returns) without getting reprimanded frequently (mitigating risk). However, it might be possible to maximize returns very early on, in a short-time horizon by using similar time-optimal algorithms as proposed in this thesis.

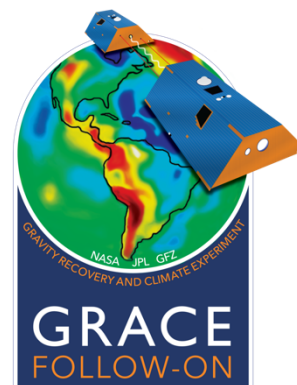


Figure 5.2: GRACE gravimetry mission, source: gracefo.jpl.nasa.gov Two satellites, amusingly named Tom and Jerry chase each other in orbit to model the gravitational potential of the Earth using their precision lasers. Gravitational potential of Earth represented in a colored mask behind the satellites.

It was observed in this thesis, that the positional states of the QR drift away from the ground truth with time. As a result, the accuracy of states deteriorate with time, increasing the uncertainty of control behavior. To avoid this, measurements must update the states frequently to be able to take accurate control actions. In Fin-tech, a similar approach of High-Frequency trading (HFT) is used for making decisions in the market with a short time model. The algorithms must make an investment decision before the short term model diverges from the behavior of the actual market. As a result, HFT creates small trades within short intervals of time to optimize the returns from the market without having a long term accurate model.

5.4. Recommendations

This section provides ideas for extending this work. This section is divided similar to the divisions in chapters. Recommendations in the area of state estimation are followed by recommendations in optimal control.

5.4.1. State estimation

It was concluded that the proposed state estimator can not predict for long time horizons. However, if the requirement of optimal control is removed, the state estimators can be allowed to be non-causal. In other words, various sensor-fusion techniques can now be used, which usually give accurate estimates than the proposed method. The red cross in Figure 3.16 can hence be removed, giving way for sensor fusion in almost every time step of model propagation.

If state estimation has to be extended using these classical methods, following a dual quaternion modeling technique can be recommended. These have multiple benefits and are described below:

Dual Quaternion State Estimation: Dual quaternions which were earlier described in Chapter 2, combine the representation of translational vectors and orientation data into a single entity and hence coupled control objective becomes easier to define. A couple of papers have explored powered descent on Lunar/Martian surfaces (which also lack GPS like our case) using the concept of dual quaternions [17]. Dual quaternion Multiplicative Extended filter (DQ-MEKF) in [5] also states that DQs are not only beneficial for control, but also lead to more accurate estimations of pose. DQs make it relatively easy to enforce the terminal constraints of zero lateral velocity and standing nose-up 90 deg during touchdown in a coupled sense. This is similar to the case for QR control for going through the gate with zero side-ways velocity while pitching to saturation. Perception Aware based MPC in [6] performs a similar task of adding intermediate state constraints to the dynamics of the QR flight for keeping feature rich objects in the field of view of cameras. This directly improves the quality of state estimates, consequently also improving the quality of control inputs. DQs can also be useful in this case.

Deep reinforcement learning

An interesting study on hexapod robots [22] was done by the BAIR group at UC-Berkeley on model-based deep-reinforcement learning for performing navigation. The difference in this approach is that the robot switches between (a) providing learning to a parameterized neural network and (b) using this learnt model for performing MPC based navigation. Abundant information (close to ground-truth) is given by onboard sensors on the hexapod for this approach to work.

A similar approach could be followed in QR parameter estimation. If a few additional sensors are allowed on the drone-race arena, these can generate enough information for parameter estimation using deep reinforcement learning. While Unscented Kalman Filtering based approaches are sensitive to initial conditions and require some heuristic while tuning it, deep-reinforcement learning can learn the parameters of these highly nonlinear models with ease. On similar lines, symbolic regression methods for identifying dynamic models have also started gaining popularity.

5.4.2. Optimal control

1. It was observed that ACADO's codegen did not support the drone-race objective of time optimization. Performing time optimal control using Equation 4.1 without codegen takes 28.3 ms to be completed

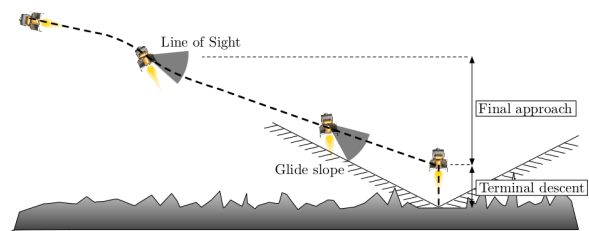


Figure 5.3: Powered descent on a lander with line of sight and glide slope constraints [17].

on a desktop computer (i7-8750H CPU @ 2.20GHz × 12). A shorter execution time can be expected after codegen can support this time-optimization objective, which could make Equation 4.1 a good candidate for execution on QR's computationally lite processor.

2. The Eigen libraries that are being used on the autopilot for hessian computations can be optimized if the autopilot is cross-compiled using the NEON flag. NEON is the optimized SIMD instruction set for ARM Cortex A8/A9 processors on which the Bebop currently runs.
3. Another recommendation would be to also include information about the various delays involved in the pipeline after the optimal control sequence is generated. This can be done by system identification of the attitude control loops. The identified state-space can be augmented to the state space model of the QR in Equation 4.4.
4. It was observed that the optimal control sequence generated by Figure 4.8 was not enough for accurate trajectory tracking. To abate tracking problems, [21] suggests using feedforward control inputs from the optimal control algorithm and feedback corrections from differential flatness. As a result, the combined controller can not only track the attitude commands, but also the positional commands of the trajectory. This feedback/feedforward duo is restated here: To match the accelerations required by the trajectory,

$$\mathbf{a}_{\text{des}} = \mathbf{a}_{\text{fb}} + \mathbf{a}_{\text{ref}} - \mathbf{a}_{\text{rd}} + g\mathbf{z}_W$$

To match the position and velocity commands required by the trajectory,

$$\mathbf{a}_{\text{fb}} = -\mathbf{K}_{\text{pos}}(\mathbf{p} - \mathbf{p}_{\text{ref}}) - \mathbf{K}_{\text{vel}}(\mathbf{v} - \mathbf{v}_{\text{ref}})$$

These errors are mapped to the thrust force being produced by the QR. Meanwhile, the required attitude can be tracked using:

$$\omega_{\text{des}} = \omega_{\text{fb}} + \omega_{\text{ref}}$$

5.5. Pontryagin's principle

Another recommendation would be to try the bang-bang based optimal control strategy originating from Pontryagin's maximum principle. Bang-bang control, as the name suggests, saturates the input to the plant as much as the actuator constraints can allow. This type of control can also theoretically make the initial state of a plant converge to a certain terminal state in a time-optimal fashion. In bang-singular systems, the input to actuators can also be a 'zero' value for non singular period of time-interval.

Bang-bang control could be an extension or simplification of the proposed controller in chapter 4. For second-order state-space models, a couple of conditions must hold for being eligible for bang-bang control.

- The second order system must be controllable, i.e. the matrix $G = [B \quad AB]$ must be full rank.
- Second order systems with full rank on controllability imply that the switching between the minimum allowable input to maximum allowable input can only happen once. For systems with lower ranks on their controllability matrix, bang-singular approach must be followed.
- A unique switching time solution is also a property of second order systems with full controllability.

These type of systems are classified in literature as normal-time-optimal-control (NTOC) systems.

In case of the models described above, the controllability matrix can be given by

$$G = \begin{bmatrix} 1 & k_d \\ 1 & 0 \end{bmatrix} \quad \text{where, } k_d \neq 0 \quad \text{and, } \text{rank}(G) = 2$$

Considering a linear system with actuator constraints, where $u = \phi$ is the roll angle of the QR in radians.

$$\begin{aligned} \dot{x} &= Ax + Bu, \quad |u| \leq 1 \\ \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} -0.5 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u \end{aligned} \quad (5.1)$$

Solving the differential equations for the above state-space representation,

$$\begin{aligned} x_1(t) &= e^{-0.5t} x_1(0) + 2u(1 - e^{-0.5t}) \\ x_2(t) &= x_2(0) + x_1(t)t \end{aligned}$$

5.5.1. Existence of a unique analytical solution

Let the initial time be given by $t_0 = 0$, switching time given by t_s and terminal time of reaching the desired state given by t_f . An analytical solution can be calculated from the solved differential equations to find out values for u between t_0 and t_s and between t_s and t_f . A simple assumption to be made for simplicity of the proof can be that the input bangs to u_{max} from t_0 to t_s and then later bangs to u_{min} from t_s to t_f . However, the states of the system can not be inferred during the switching time, i.e. there is no knowledge of $x_1(t_s)$ and $x_2(t_s)$. Hence, boundary conditions (states at t_s and terminal time t_f) cannot be found out using these differential equations, making it difficult to find an analytical solution for switching time and terminal time. For the same reason, another method using Pontryagin's Maximization Principle can be tried out to find out the solution to this problem.

5.5.2. Pontryagin's minimum principle (PMP)

From the above derivations for the controllability of the system, the state space described in Equation 5.1 fits under the NTOC category. Hence, a single unique switching-time solution that takes the QR from one position and velocity in the arena to another reference position and velocity can be found out using the PMP principle. Hamiltonian Jacobi Bellman (HJB) equations along with PMP can be evaluated to find out the analytical solution for the optimal switching time. Without loss of generality, it can be assumed that for a terminal state $x_f = 0$ and initial state $x(0) = x_0$, the cost function for time-optimality can be given by:

$$J = \int_0^T 1 dt$$

The Hamiltonian for this cost function can be given by:

$$H = 1 + \lambda^T (Ax + Bu) = 1 + (\lambda^T A)x + (\lambda^T B)u$$

Applying the conditions for optimality, yields the evolution of the cost function with respect to varying co-state vectors:

$$\begin{aligned} \dot{x} &= \frac{\partial H}{\partial \lambda} = Ax + Bu \\ -\dot{\lambda} &= \frac{\partial H}{\partial x} = A^T \lambda, \quad \begin{bmatrix} \dot{\lambda}_1 \\ \dot{\lambda}_2 \end{bmatrix} = \begin{bmatrix} -0.5 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \end{aligned} \quad (5.2)$$

The input u which minimizes the cost function H can be given as,

$$u = \operatorname{argmin} H = -\operatorname{sgn}(\lambda^T B)$$

The optimal solution always satisfies the above equations since Pontryagin's principle provides a necessary condition. It follows that the input is always either +1 or -1, depending on $\lambda^T B$ (in this case only the first term '1' of cost function H can be minimized by selecting the scalar control input as $u = -\lambda^T B$).

$$u = -\operatorname{sgn} \begin{bmatrix} \lambda_1 & \lambda_2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = -\operatorname{sgn} \lambda_1$$

From Equation 5.2, the evolution of costates can be given as:

$$\dot{\lambda}_1 = -0.5\lambda_1 + \lambda_2, \quad \therefore \lambda_1(t) = e^{-0.5\lambda_1(t)} + 2\lambda_2(1 - e^{-0.5t})$$

$$\dot{\lambda}_2 = 0, \quad \therefore \lambda_2(t) = \lambda_2(0)$$

However the initial conditions of λ_1, λ_2 are unknown, and assuming a priori on them to find the u that minimizes H is heuristic or rather search based. Also, looking at the equations above, an analytical solution for switching time, terminal time or optimal control sequence is not possible.

5.5.3. The Boston University algorithm

Another method for finding the optimal control sequence had to be investigated since no analytical solution for the inputs was available using the above methods. This method also tends to follow a search fashion, although it does so within a very minuscule search space [28]. The method propagates system dynamics from the initial conditions x_0 to reach the states at switching time $x(t_s)$. It later propagates the states in a backward fashion from $x(t_f)$ to $x(t_s)$.

The method proposed in [28] is exactly suitable for the linear drag model, since their paper specifically discusses second order systems that use a similar canonical form. After substituting the drag coefficients in their second order state space, state trajectories for the QR case can be solved. These trajectories which are propagated from different ends of the maneuver, finally intersect in the phase plane plot at one unique point, which yields the state of the system at the switching time ($x(t_s)$). After $x(t_s)$ is found, the values for t_s, t_f, u can be analytically solved for.

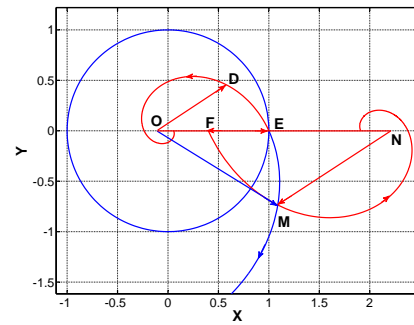
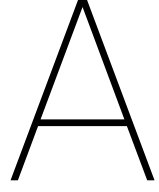


Figure 5.4: A phase plane plot can yield the optimal switching time and terminal time of the bang-bang maneuver[28] (given by coordinates of M).



Appendix

A.1. Scaling laws

A section from IEEE's magazine article [20], gives an interesting peak into why tiny quadrotors could be more agile and useful in the drone race scenario. In drone racing, agility is key. Unlike the drones used in search and rescue operation which need to perform SLAM at slow speeds of 1m/s, racing drones must be configured to push the flight envelope by being able to accelerate quickly. In such cases, the racing drone must be able to produce high linear and angular accelerations.

To study which type of drone would be ideal for drone-racing, the Parrot Bebop is compared against the Eachine Trashcan. The average angular velocity of Bebop1's propeller while flying is approximately $n = 8000$ rpm. In SI units the angular velocity of the blade hence approximately is $\omega_i \approx 700$ rad/s. The tip speed of the 7cm propeller is almost $V_b = r\omega \approx (0.07 \times 700) \approx 50$ m/s. The Mach number ratio is $M = (50/343) = 0.15$ which is well within the subsonic regime. Hence the assumptions made by the Mach scaling for compressible flows can be applied. (If Mach scaling isn't applicable, Froude scaling laws for in-compressible flows are assumed).

The ability to produce thrust is given by:

$$T = c_T \rho A r^2 \omega^2, \quad T \propto \omega^2 r^4$$

where T is the thrust produced, ρ is the density of air, A is the area swept by the blade, ω is the angular velocity of the blade. We assume that the radius of the propeller r decides the length of the quadrotor's frame and hence the mass. As a result the mass of the quadrotor scales as $m \propto r^3$.

The linear acceleration hence scales by $a \propto \omega^2 \frac{r^4}{r^3}$.

Similarly, the angular acceleration scales as $\alpha \propto \omega^2 \frac{r^5}{r^3}$.

Mach scaling quotes $\omega \sim (1/r)$ assuming that the blade tip speed (V_b) is constant while scaling up/down the size of the quadrotor. Substituting the scaling law in the above relations,

$$a \sim \frac{1}{r} \quad \alpha \sim \frac{1}{r^2}$$

This goes to say that smaller the size of the quadrotor frame, (which is linearly related to radius of the propeller) leads to higher angular and lateral acceleration capability. Hence a tiny TODO:

A.2. Modeling, Control and System Identification of the Moment models

Table 2.1 mentioned that the moment and attitude control is a solved problem. However they require some introduction and are appended in the appendix. **Modeling:** This subsection restates the Netwon-Euler based moment models mentioned in [5].

$$I\dot{\Omega} = -\Omega \times I\Omega + \sum_{i=1}^4 (\tau_{M_i} + \tau_{r_i}) + \tau_d \quad (A.1)$$

where,

$$\tau_{r_i} = k_\beta a_{s_i} + \mathbf{r}_i \times \mathbf{f}_i \quad (A.2)$$



Figure A.1: The Eachine Trashcan has the same size as a propeller of the Parrot Bebop.

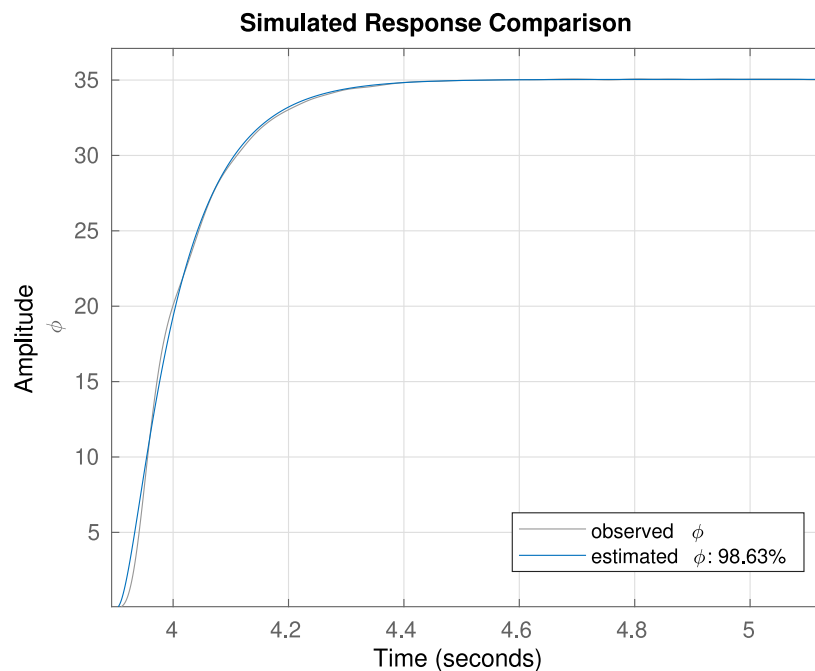


Figure A.2: System identification of the roll maneuver.

$$\tau_{M_i} = C_{Q_i} \rho A_p r^3 \omega_i |\omega_i| \hat{\mathbf{k}} \quad (\text{A.3})$$

and τ_d represents the drag moment vector.

Control: However, as stated in [24], the moment models can simply be dealt with high gain feedback controllers. Commonly autopilots (like BetaFlight and Paparazzi) have very well tuned PD control loops for controlling the rate of the QR. As a result, a simple ‘inner-loop’ is enough to stabilize the attitude of the QR. However, after plugging in the gains and making this PD based inner-loop, system identification for these models must be done.

System Identification: Doing the system identification step is critical for the models described in the optimal controllers above. The proposed controller has assumed that there is instantaneous convergence to the commanded angles. The consequences of directly flushing the optimal control commands to the PD controller must be studied.

A second order system identification is done for identifying the parameters of the inner loop. The transfer function found, has a good fit of 98.63% to the ground truth roll angles of the QR. Figure A.2 shows the transient angles generated by the transfer function with respect to the ground truth angles.

$$\phi(s) = \frac{508.3}{s^2 + 57.9s + 507.6} \phi_{cmd}$$

It is found that the settling time of the roll angles exceeds the discretization intervals of the state space

in Equation 4.5. The settling time when a step input of 35deg is commanded is almost 0.4 seconds. As a result, the commands flushed from the optimal controller would take longer to converge as compared to the assumed discretization interval of 0.1 seconds. This information must be augmented in the state space of Equation 3.15 as described in [3]. As a result, the optimal control sequence generated would already be cognizant of the settling time of the innerloop of the tuned attitude controller.

A.3. Accelerometers for positional state estimation

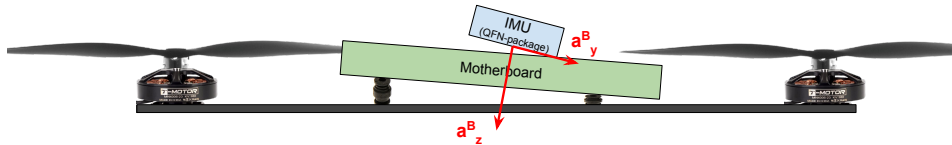


Figure A.3: Bias due to the accelerometer's misaligned axes.

Most frequently QRs use an MPU6050 (QFN package) soldered to the motherboard as their inertial measurement unit. These accelerometers are usually mis-aligned with the body axes of the QR, due to two reasons. (a) The motherboard of QRs commonly sit on dampers for avoiding transmission of vibrations from the motors to sensitive electronics soldered on the motherboard. These dampers deform over time, creating an offset between the QR's body frame axes and the IMU's sensing axes. (b) Some amount of heat produced by the BLDC motor controllers can transfer to IMU's solder points, disturbing the levels of mount points of the IMU on the motherboard¹. This creates mis-alignment of the axes dynamically over time. Ignoring these mis-alignment errors can cause significant drift if the accelerations measured are directly dead-reckoned.

Secondly, if the position of the IMU is very different from the center of mass of the QR, then the accelerometer is subjected to additional centrifugal and Coriolis forces which must be compensated for. Underestimating the consequences of offset of IMU's position from the CoM can cause significant drifts if the measured accelerations are directly dead-reckoned. These pseudo-forces are illustrated by an IMU company called 'x-sens' in one of their publications [16].

$$a_{ii}^n = a_{nn}^n + \underbrace{2\omega_{ie}^n \times v_n^n}_{\text{Coriolis force}} + \underbrace{\omega_{ie}^n \times \omega_{ie}^n \times p^n}_{\text{Centrifugal force}}$$

Even if the errors due to mis-alignment and placement on the motherboard are taken care of by carefully designing the motherboard of the QR, the IMU brings with itself its own inherent manufacturing errors. These errors include non-zero cross-sensitivity and non-identity scaling factors between different axes.

TODO: Add estimate accelerometer bias and fuse it with existing model compare Rsq values.

A.4. Brachistochrone problem

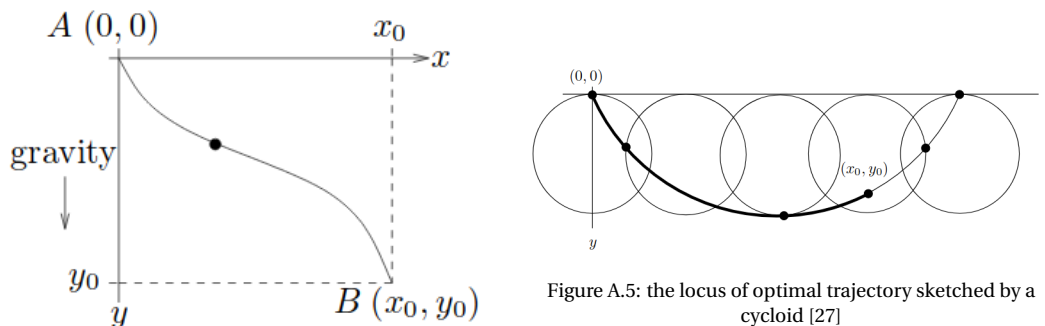


Figure A.4: the problem presented by Bernoulli in 1696 [27]

¹The temperature sensitivities are also very critical in magnetometers or compasses. They come with special temperature corrections since hysteresis in magnetic elements is sensitive to temperature.

An interesting account of problem was presented in the year 1696, which forms the basis of the motivation behind trajectory optimization problems [27]. Johann Bernoulli (1667-1748) posted a problem about sketching a time-optimal trajectory when traveling from point A to B, under the influence of gravity. Along with a figure alike the Figure A.4, he quoted *"I hope to gain the gratitude of the whole scientific community by placing before the finest mathematicians of our time a problem which will test their methods and the strength of their intellect. If someone communicates to me the solution of the proposed problem, I shall publicly declare him worthy of praise"*.

During the time of publication, a lot of mathematicians were actively vying in the area of calculus and it did not take long for solutions to come in from Galileo, Newton, l'Hôpital, Johann's own brother Jacob and Tschirnhaus. The solution to the optimization problem is non-trivial (a straight line joining points A to B is not the solution). If the slope of the trajectory is comparatively higher in the beginning, body picks up a higher velocity early on and total time can be reduced! However the correct solution is a bit far from just having higher slopes during the start.

Proof: conservation of energy would give us the relation between the kinetic and potential energy: $K.E. = P.E.$ and hence $\frac{1}{2}mv^2 - mgy = 0$. The instantaneous velocity at any point on the trajectory is given by $v = \frac{ds}{dt} = \sqrt{2gy}$ where s is the arc length of the distance traveled on the trajectory. $\delta s = ((\delta x)^2 + (\delta y)^2)^{\frac{1}{2}}$. Total time is found by integrating the segment of time traveled:

$$T = \int_{\text{curve}} \frac{ds}{\sqrt{2gy}} = \frac{1}{\sqrt{2g}} \int_0^{y_0} \sqrt{\frac{1 + \left(\frac{dx}{dy}\right)^2}{y}} dy \quad (\text{A.4})$$

The value of T must be a global minimum, making the object reach from point A to B in the minimum time possible. Fortunately, a very elegant solution (as in Figure A.5) emerges from the optimization of the cost function described above. The solution is the locus of points described by a cycloid given by:

$$x = r\theta - r \sin\theta, \quad y = r - r \cos\theta \quad (\text{A.5})$$

where r is the radius of the circle rotating on the horizontal plane. If any object travels from point A to point B, ie $(0, 0)$ to (x_0, y_0) in Figure A.5, it reaches the destination with minimum time. The optimal control problem for drone racing in this section comes with a similar objective - of generating appropriate commands to travel from point A to point B in minimum time, while not violating the constraints and dynamics of QRs.

Bibliography

- [1] Federal Aviation Administration. Helicopter flying handbook, 2017. URL https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/helicopter_flying_handbook/.
- [2] Moses Bangura, Robert Mahony, et al. Nonlinear dynamic modeling for high performance control of a quadrotor. *Australian Robotics and Automation Association*, 2012.
- [3] P. Bouffard. On-board Model Predictive Control of a Quadrotor Helicopter: Design, Implementation, and Experiments. *Ucb/Eecs-2012-241*, page 67, 2012. URL <http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-241.html>.
- [4] Stephen Boyd and Lieven Vandenbergh. *Convex optimization*. Cambridge university press, 2004.
- [5] Pedro Castillo-García, Laura Elena Muñoz Hernandez, and Pedro García Gil. In *Indoor Navigation Strategies for Aerial Autonomous Systems*, pages 31 – 50. 2017. ISBN 978-0-12-805189-4. URL <http://www.sciencedirect.com/science/article/pii/B9780128051894000032>.
- [6] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. PAMPC: Perception-Aware Model Predictive Control for Quadrotors. 2018. doi: 10.1109/IROS.2018.8593739. URL <http://arxiv.org/abs/1804.04811>.
- [7] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.
- [8] Jeff Gill and Gary King. What to do when your hessian is not invertible: Alternatives to model respecification in nonlinear estimation. *Sociological methods & research*, 33(1):54–87, 2004.
- [9] Winter Guerra, Ezra Tal, Varun Murali, Gilhyun Ryou, and Sertac Karaman. FlightGoggles: Photorealistic Sensor Simulation for Perception-driven Robotics using Photogrammetry and Virtual Reality. 2019. URL <http://arxiv.org/abs/1905.11377>.
- [10] David J. Hanley and Timothy W. Bretl. An Improved Model-Based Observer for Inertial Navigation for Quadrotors with Low Cost IMUs. (January), 2016. doi: 10.2514/6.2016-0105.
- [11] Markus Hehn, Robin Ritz, · Raffaello D’andrea, M Hehn, · R Ritz, and · R D’andrea. Performance benchmarking of quadrotor systems using time-optimal control. 33:69–88, 2012. doi: 10.1007/s10514-012-9282-3.
- [12] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. Acado toolkit—an open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.
- [13] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10):2279–2285, 2011. ISSN 00051098. doi: 10.1016/j.automatica.2011.08.020. URL <http://dx.doi.org/10.1016/j.automatica.2011.08.020>.
- [14] Haomiao Huang, Gabriel M. Hoffmann, Steven L. Waslander, and Claire J. Tomlin. Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3277–3282, 2009. ISSN 10504729. doi: 10.1109/ROBOT.2009.5152561.
- [15] Juraj Kabzan, Miguel de la Iglesia Valls, Victor Reijgwart, Hubertus Franciscus Cornelis Hendriks, Claas Ehmke, Manish Prajapat, Andreas Bühler, Nikhil Gosala, Mehak Gupta, Ramya Sivanesan, et al. Amz driverless: The full autonomous racing system. *arXiv preprint arXiv:1905.05150*, 2019.

- [16] Manon Kok, Jeroen D Hol, and Thomas B Schön. Using inertial sensors for position and orientation estimation. *arXiv preprint arXiv:1704.06053*, 2017.
- [17] Unsik Lee and Mehran Mesbahi. Optimal Power Descent Guidance with 6-DoF Line of Sight Constraints via Unit Dual Quaternions. *AIAA Guidance, Navigation, and Control Conference*, pages 1–21, 2015. doi: 10.2514/6.2015-0319. URL <http://arc.aiaa.org/doi/10.2514/6.2015-0319>.
- [18] Robert C Leishman, John Macdonald, Robert C ; Leishman, John ; Macdonald, Randal W ; Beard, and Timothy W Mclain. Quadrotors and Accelerometers: State Estimation with an Improved Dynamic Model BYU ScholarsArchive Citation. *Original Publication Citation*, 34(1):28–41, 2013. ISSN 1066-033X. doi: 10.1109/MCS.2013.2287362. URL <https://scholarsarchive.byu.edu/facpub>.
- [19] Shuo Li, Erik van der Horst, Philipp Duernay, Christophe De Wagter, and Guido CHE de Croon. Visual model-predictive localization for computationally efficient autonomous racing of a 72-gram drone. *arXiv preprint arXiv:1905.10110*, 2019.
- [20] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics and Automation Magazine*, 19(3):20–32, 2012. ISSN 10709932. doi: 10.1109/MRA.2012.2206474.
- [21] Antonio Franchi Matthias Faessler and Davide Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *CoRR*, abs/1712.02402, 2017. URL <http://arxiv.org/abs/1712.02402>.
- [22] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.
- [23] D Subbaram Naidu. *Optimal control systems*. CRC press, 2002.
- [24] Sammy Omari, Minh Duc Hua, Guillaume Ducard, and Tarek Hamel. Nonlinear control of VTOL UAVs incorporating flapping dynamics. *IEEE International Conference on Intelligent Robots and Systems*, pages 2419–2425, 2013. ISSN 21530858. doi: 10.1109/IROS.2013.6696696.
- [25] B Razgus, E Mooij, and D Choukroun. Relative navigation in asteroid missions using dual quaternion filtering. *Journal of Guidance, Control, and Dynamics*, 40(9):2151–2166, 2017.
- [26] Hugo Romero, Sergio Salazar, and José Gómez. Real-Time Stabilization of an Eight-Rotor UAV Using Stereo Vision and Optical Flow. *Unmanned Aerial Vehicles: Embedded Control*, 25(4):237–263, 2013. doi: 10.1002/9781118599938.ch12.
- [27] Amol Sasane. *Optimization in Function Spaces*. Courier Dover Publications, 2016.
- [28] Zhaolong Shen and Sean B Andersson. Minimum time control of a second-order system. In *49th IEEE Conference on Decision and Control (CDC)*, pages 4819–4824. IEEE, 2010.
- [29] Steven E Shreve. *Stochastic calculus for finance II: Continuous-time models*, volume 11. Springer Science & Business Media, 2004.
- [30] Sihao Sun, Rudi Schilder, and Coen C. de Visser. Identification of Quadrotor Aerodynamic Model from High Speed Flight Data. (January), 2018. doi: 10.2514/6.2018-0523.
- [31] James Svacha, Kartik Mohta, and Vijay Kumar. Improving quadrotor trajectory tracking by compensating for aerodynamic effects. *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, pages 860–866, 2017. doi: 10.1109/ICUAS.2017.7991501.