Faculty of Aerospace Engineering

Trajectory Library based Guidance for Drones in Unknown Environment

A feasibility study

S. Wang

November 17, 2017

TU Delft Delft University of Technology

Challenge the future

Trajectory Library based Guidance for Drones in Unknown Environment

A feasibility study

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

S. Wang

November 17, 2017

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Trajectory Library based Guidance for Drones in Unknown Environment"** by **S. Wang** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: November 17, 2017

Readers:

dr.ir. Q. P. Chu

dr.ir. C. C. de Visser

S. Li

dr. M. A. Mitici

Acronyms

CL	Closed-loop
GA	Genetic Algorithm
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INDI	Incremental Nonlinear Dynamic Inversion
MILP	Mixed-integer Linear Programming
MIP	Mixed-integer Programming
NDI	Nonlinear Dynamic Inversion
NED	North-East-Down
NLP	Nonlinear Program
QP	Quadratic Program
TUD	Delft University of Technology

List of Symbols

Greek Symbols

Ω	Rotational speed of rotor
ω	Angular velocity
ϕ	Roll angle
ψ	Yaw angle
θ	Pitch angle

Roman Symbols

b	Width of the Parrot Bebop
F_*	Force
D_*	Drag
g	Gravitational constant
I	Moment of Inertia
J	Cost function
l	Length of the Parrot Bebop
M_*	Moment
m	Mass of the Parrot Bebop
p	Roll rate
q	Pitch rate
r	Yaw rate
T	Thrust force
u	Body velocity in x axis

V_*	Velocity
v	Body velocity in y axis
w	Body velocity in z axis

Contents

Chapter 1

Introduction

The field of UAVs has received increased attention because of their simplicity and high maneuverability. The research of UAVs began with military uses, where the UAVs are used for surveillance and air strikes. Recent developments in technology enable a wide range of successful civil applications, including monitoring & inspection for civil engineering(Chan, Guan, Jo, & Blumenstein, 2015), UAV precision agriculture(Zhang & Kovacs, 2012), aerial imaging & mapping(Colomina & Molina, 2014; Nex & Remondino, 2014), etc. In some cases, the drones are stabilized with an inner-loop controller while the "human in the loop" approach is adopted to remotely control the drones' positions. Several disadvantages can be identified with such approaches, such as shorter range and low flexibility. A higher level of autonomy is required to further exploit the potential of UAVs.

Advanced autonomy relies on precise information of the drone's states and its surrounding environment. Usually, the Inertial Measurement Unit (IMU) yields the UAV attitude, while position is provided by an external localization system such as Global Positioning System (GPS), VICON[1] or OptiTrack[2]. However, GPS signals may not be available at all times, especially in indoor environments. Several investigations have been performed to tackle the challenge of UAV navigation in absence of external localization system. For instance in Ref. (He, Prentice, & Roy, 2008), the problem of motion planning for autonomous quadrotor flight without GPS is concerned. A laser range-finder is equipped to estimate the drone position and attitude. On-board camera is also a commonly used sensor for real-time feedback of the environment. Recent research efforts at Delft University of Technology (TUD) have developed such an autonomous drone which can sense the gates and navigate itself through the gates successfully with the help of down-facing cameras.

However the use of on-board camera suffers from several drawbacks. Firstly, the velocity of the quadrotor is estimated by analyzing the texture on floor. This constraints the flight condition, i.e., agile maneuvers cannot be achieved due to the high computational

[1]VICON. `www.vicon.com`
[2]OptiTrack. `http://optitrack.com`

effort required by the algorithm. Moreover, the approach cannot be applied to environments with little texture on the floor. Secondly, after detecting the gate, the drone takes significant amount of time to position itself to the center of the gate.

In order to tackle the issues with the current existing vision-based navigation approach, a novel guidance method, *based on a set of pre-stored dynamically optimal trajectories,* is explored in this project. The trajectory library is designed and optimized to guide the drone through obstacles. These trajectories are first tested with position feedback provided by the OptiTrack. For each trajectory, the control inputs are recorded and saved. They are then "replayed" in the second flight test where the position feedback is not available.

1-1 Problem statement and research objectives

Based on the approach, the research question of the thesis can be stated as follows:

Verify the feasibility of a novel guidance method based on a set of pre-stored trajectories that are dynamically optimal.

The navigation method can be verified feasible if the quadrotor is capable of following the designed trajectory with small deviations. The research can be divided into two parts. The objectives of each part are listed below.

1. Preliminary research phase
    (a) Literature study on the quadrotor modeling and state of the art control techniques.
    (b) Literature study on the commonly used trajectory generation method and their pros and cons.
    (c) Generate collision-free and dynamically feasible trajectories in the simulation environment.
    (d) Perform simulation with the generated trajectories and quadrotor model.
    (e) Verify whether the navigation approach is feasible with the simulation results.
2. Flight test phase
    (a) Program and test on-board controllers to follow the designed trajectory.
    (b) Perform closed-loop flights and record the control inputs.
    (c) Repeat the same trajectory without Optitrack signals and analyze the results.
    (d) Compare the results from multiple open-loop flights and draw conclusions on the feasibility of the proposed method.
    (e) Derive recommendations for future research.

For the flight experiment, the Parrot Bebop drone, depicted in Figure 1-1, is used. All experiments are conducted in the Cyber Zoo[3] at TUD.

[3]Cyber Zoo. `http://tudelftroboticsinstitute.nl/labs/cyber-zoo`

[4]Parrot Bebop. `https://www.parrot.com/us/sites/default/files/styles/product_teaser_display/public/ps/2691-large-parrot-2691jpg.jpg?itok=RNsHN9d7`

Figure 1-1: Parrot Bebop[4]

1-2 Thesis scope and focus

In this thesis, the main focus is on the feasibility analysis of the proposed method. The success of the approach relies on the hypothesis that open-loop flights, in short intervals, can achieve same level of accuracy as the closed-loop flights. To verify the hypothesis, both simulations and flight tests are performed, and flight tests are considered the main method of verification.

After the trajectory generation and optimization, a quadrotor model and a simple PID feedback controller is used to simulate the designed trajectory. Fully non-linear quadrotor model is chosen, for accurate representation of the Bebop drone dynamics. However, the aerodynamic coupling and disturbances are not modeled in this study. Only a simple PID based feedback controller was explored here due to the limited scope of the project. However it is recommended in general to use a combined feedforward-feedback controller for better performance. In fact, it is one of the main recommendations of the thesis.

Many external factors were identified that influence the performance of the experimental flights. To name a few, the hardware limitations of the Bebop drone, unmodeled aerodynamic forces and disturbances, etc. This study aims to discover the extend to which the proposed method is applicable with the existing hardware and experimental conditions, rather than to tackle each problem that occurred during the experiments. To improve the performance of the method, the aforementioned factors can be studied. Due to limitations in time and facility, they are not covered in this work. However several suggestions are proposed in the recommendation section.

1-3 Thesis outline

The thesis is structured as follows,

Chapter 1 introduces the background of the project. The proposed method is illustrated together with the research objectives and scope. The thesis is divided into two parts. The main results and conclusions of the project are presented in part I, as a scientific paper. In Part II, the appendices contain the detailed descriptions and relevant results of the project. Appendix A includes a literature survey on the quadrotor model and control techniques. State-of-the-art methods for trajectory optimization are presented in appendix B, followed

by a short summary of the literature survey in appendix C. The simulations performed in MATLAB are elaborated in appendix D and the additional results are included in appendix E. Finally, chapter 6 summarizes and concludes the entire project.

Part I

Scientific Paper

Trajectory Library Based Guidance for Drones in Unknown Environment

S. Wang*, S. Li† and dr.ir. C. C. de Visser‡
Delft University of Technology, 2629 HS Delft, The Netherlands

UAVs(Unmanned Aerial Vehicles) have gained popularity in various fields, and many applications require the drones to perform tasks in indoor environments. One major challenge in UAV indoor guidance is the absence of accurate GPS(Global Positioning System) signals. In this work, a novel guidance method is investigated which uses a library of precomputed trajectories which can be used in GPS-denied environments. Trajectory library is constructed and it stores the sequence of control inputs, which are selected to be the Euler angles and thrust force, obtained from closed-loop flight. Flight test results suggest that the approach is feasible for short intervals, and the standard deviations between open-loop flights are shown to be within acceptable range. The relationships between velocity, time and standard deviation are also explored.

Nomenclature

θ = Pitch angle, deg
ϕ = Roll angle, deg
ψ = Yaw angle, deg
T = Thrust force, mN
Q = Hessian matrix, -

Acroynms
UAV Unmanned Aerial Vehicle
GPS Global Positioning System
NLP Nonlinear Program
QP Quadratic Program
INDI Incremental Nonlinear Dynamic Inversion
MSE Mean squared error
OL Open-loop (flight)
CL Closed-loop (flight)

Subscript
ref Reference
cmd Command

I. Introduction

The field of UAVs has received increased attention because of their simplicity and high maneuverability. The research of UAVs began with military uses, where the UAVs are used for surveillance and air strikes. Recent developments in technology enable a wide range of successful civil applications, including monitoring & inspection for civil engineering,[1] UAV precision agriculture,[2] aerial imaging & mapping,[3,4] etc. In some cases, the drones are stabilized with an inner-loop controller while the "human in the loop"

*MSc Student, Control & Simulation, Faculty of Aerospace Engineering, Kluyverweg 1, Delft, The Netherlands..
†PhD Student, Control & Simulation, Faculty of Aerospace Engineering, Kluyverweg 1, Delft, The Netherlands..
‡Assistant Professor, Control & Simulation, Faculty of Aerospace Engineering, Kluyverweg 1, Delft, The Netherlands. AIAA Member.

approach is adopted to remotely control the drones' positions. Several disadvantages can be identified with such approaches, such as shorter range and low flexibility. A higher level of autonomy is required to further exploit the potential of UAVs.

Advanced autonomy relies on precise information of the drone's states and its surrounding environment. Usually, the IMU yields the UAV attitude, while position is provided by an external localization system such as GPS, VICON[a] or OptiTrack[b]. However, GPS signals may not be available at all times, especially in indoor environments. Several investigations have been performed to tackle the challenge of UAV navigation in absence of external localization system. For instance in Ref. 5, the problem of motion planning for autonomous quadrotor flight without GPS is concerned. A laser range-finder is equipped to estimate the drone position and attitude. On-board camera is also a commonly used sensor for real-time feedback of the environment. Recent research efforts at TUD have developed such an autonomous drone which can sense the gates and navigate itself through the gates successfully with the help of down-facing cameras.

However the use of on-board camera suffers from several drawbacks. Firstly, the velocity of the quadrotor is estimated by analysing the texture on floor. This constraints the flight condition, i.e., agile manoeuvres cannot be achieved due to the high computational effort required by the algorithm; Moreover, the approach cannot be applied to environments with little texture on the floor. Secondly, after detecting the gate, the drone takes significant amount of time to position itself to the center of the gate.

This work aims to verify the feasibility of a novel guidance method, *based on a set of pre-stored trajectories that are dynamically optimal,* for UAVs operating in unknown environment with no external position sensors. These trajectories are first tested with position feedback provided by the OptiTrack. For each trajectory, the control inputs are recorded and saved. In fact, the control inputs to the quadrotor system are the rotor rotational speeds. However in this work, the control inputs are selected to be the Euler angles of the UAV. This is because open-loop flight with rotor speeds as inputs does not guarantee stability of the drone. Moreover, the Euler angles can be more easily interpreted compared to rotor speed. The Euler angles are then "replayed" in the second flight test where the position feedback is not available. Feasibility of the approach can be addressed by analysing the difference between the closed-loop and open-loop flight. The proposed guidance method is illustrated in figure 1.

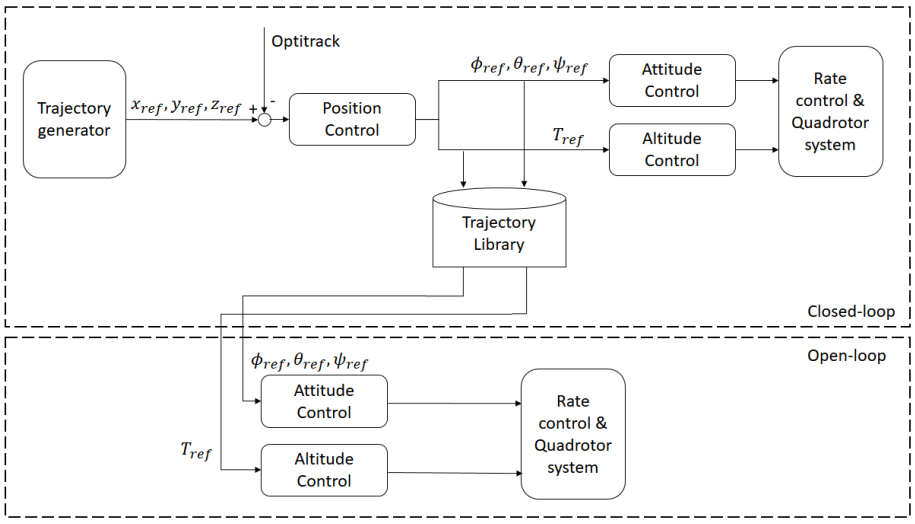


Figure 1. Block diagram illustrating the proposed guidance method.

The paper is structured as follows. In section II an overview of the research done in the related field is provided. Section III introduces the trajectory optimization based on minimum snap. Multiple test flights are performed and the experimental set-up is presented in section IV, followed by the test results in section V. Finally conclusions are drawn in section VI.

[a]VICON. www.vicon.com
[b]OptiTrack. http://optitrack.com

II. Related Work

One major challenge in the automation of UAVs is the guidance, navigation and control. Inspired by the honeybees, a natural approach is to utilize visual information such as optical flow. In Ref. 6, a wall collision avoidance navigation algorithm based on optical flow is presented. Combined with the IMU data, safe navigation through indoor corridors can be achieved. A visual odometer, which performs state estimation by visually tracking objects in the environment, is illustrated in Ref. 7. Similarly, Ref. 8 combines the visual information with a target tracker to navigate a UAV in GPS-denied environment. However these solutions are restricted by the richness of features in the surroundings, or the computational speed of the image processing algorithm. To allow aggressive quadrotor manoeuvres, a planning algorithm to generate collision-free and optimized trajectories is proposed in Ref. 9. Experimental flights with small quadrotor in obstacle-dense environment validated the algorithm.

Various methods and techniques exist for trajectory optimization. The difficulty lies in the imposed system dynamic constraints, which are represented in the form of differential equations. Direct and indirect methods can be distinguished in the field of trajectory optimization. Indirect method is based on calculus of variations. With this approach, the necessary optimality conditions need to be derived and constrained optimization problem is converted into the equivalent unconstrained Hamiltonian function of the problem. Several investigations have been conducted on UAV trajectory optimization using the indirect method.[10,11]

Contrary to the indirect method, the direct method transforms the original optimization problem into a NLP problem. This is done by parametrization of the controls or/and state variables. The decision variables are thus converted from functions to real numbers. In other words, the direct method discretizes the problem before optimization. Direct method has been widely applied in UAV related trajectory optimization. Ref. 12 considers the three dimensional time-optimal trajectory optimization problem. Direct method is also used in Ref. 13 to determine the optimal energy-efficient trajectories of a UAV flying through a thermal cell. The direct method can be further categorized into two approaches, the shooting and collocation method. However, either method requires numerical integration of system equation to derive the next state. This puts a heavy burden on the computational effort.

The quadrotor system has been proved to be *differentially flat*.[14] Differential flatness was firstly proposed in Ref. 15. It states that a system is said to be differentially flat if all states and inputs can be expressed in terms of a set of outputs and their derivatives. This means that the inputs to the system can be derived by simply differentiating the flat outputs. Differential flatness eliminates the need for numerical integration of system equations and guarantees the dynamic feasibility of the designed trajectory, provided that the input bounds are satisfied. Extending the previous work, authors in Ref. 16 showed that by reformulating the problem as an unconstrained Quadratic Programming(QP) problem, the minimum snap trajectory can be solved with numerical stability for long-range trajectories with many segments.

Another approach to reduce the solution space of trajectory optimization problem is to use a finite set of feasible trajectories or motion primitives. It was shown in Ref. 17 that the concatenated trajectory is feasible if the individual element in the library is feasible. In Ref. 18, a motion primitive library was used to enable fast flight through regions with dense obstacles. The motion primitives consist of sequential control inputs(e.g. control surface settings) that are required to perform certain maneuvers. Trajectory library can also be used to account for model uncertainties and disturbances. More recently in Ref. 19, open-loop trajectories are computed in the offline stage, and a "funnel" is created around each trajectory to represent the reachable set. With this method, the uncertainties are explicitly taken into account during the trajectory selection.

III. Minimum-Snap Trajectory Generation

A. Cost Function for Minimum-Snap Trajectory

In this section, method for generating minimum-snap trajectory is derived. The minimum-snap trajectory was originally applied in Ref. 14. It was found that the moments around x and y axis appear to be functions of the snap, and therefore by minimizing the snap, the control effort is minimized. Since the quadrotor system is differentially flat and the flat outputs are selected to be $\boldsymbol{z} = [x_E, y_E, z_E]^T$, we can represent the trajectory, described by the flat outputs, between any grid point and the center of the gate with polynomial of order n as,

$$P(\boldsymbol{p},t) = p_0 + p_1 t + \cdots + p_{n-1}t^{n-1} + p_n t^n = \sum_{n=0}^{N} p_n t^n \tag{1}$$

The optimization objective is the total snap along a trajectory $P(t)$ between $t = 0$ and $t = t_f$, which defines the cost function J.

$$J(\boldsymbol{p},t) = \int_0^{t_f} (\frac{d^4\boldsymbol{z}}{dt^4})^2 dt \tag{2}$$

First, the derivatives of $P(t)$ are taken. The r^{th} derivative of $P(\boldsymbol{p},t)$ can be determined from the following equations.

$$P^{(1)}(\boldsymbol{p},t) = p_1 + 2p_2 t + \cdots + np_n t^{n-1} \tag{3}$$
$$P^{(2)}(\boldsymbol{p},t) = 2p_2 + 3 \times 2p_3 t \cdots + n(n-1)t^{t-2} \tag{4}$$
$$\vdots \tag{5}$$
$$P^{(r)}(\boldsymbol{p},t) = \sum_{n=r}^{N} \left(\prod_{m=0}^{r-1} (n-m) \right) p_n t^{n-r} \tag{6}$$

The coefficient C_n of the n^{th} term of the square of any polynomial, can be found using equation (7).

$$C_n = \sum_{j=0}^{N} p_j p_{n-j} \tag{7}$$

Substituting the square of $P^{(r)}(\boldsymbol{p},t)$ into the cost function yields,

$$J = \int_0^{t_f} \left(P^{(r)}(\boldsymbol{p},t) \right)^2 dt \tag{8}$$
$$= \int_0^T \sum_{j=0}^{2N} \sum_{n=r}^{N} \left(\prod_{m=0}^{r-1} (n-m) p_j p_{n-j} t^{n-2r} \right) dt \tag{9}$$
$$= \sum_{n=2r}^{2N} \left[\sum_{j=r}^{n-r} \left(\prod_{m=0}^{r-1} (j-m)(n-j-m) \right) p_j p_{n-j} \right] \frac{t_f^{n-2r+1}}{n-2r+1} \tag{10}$$

Equation (8) can be expressed in matrix form as,

$$J = \boldsymbol{p}^T Q \boldsymbol{p} \tag{11}$$

where Q is the Hessian matrix defined as,

$$Q = \begin{bmatrix} \frac{\partial^2 J}{\partial p_1^2} & \frac{\partial^2 J}{\partial p_1 \partial p_2} & \cdots & \frac{\partial^2 J}{\partial p_1 \partial p_n} \\ \frac{\partial^2 J}{\partial p_2 \partial p_1} & \frac{\partial^2 J}{\partial p_2^2} & \cdots & \frac{\partial^2 J}{\partial p_2 \partial p_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial p_n \partial p_1} & \frac{\partial^2 J}{\partial p_n \partial p_2} & \cdots & \frac{\partial^2 J}{\partial p_n^2} \end{bmatrix} \tag{12}$$

To construct Q, the cost function J is differentiated with respect to p_i.

$$\frac{\partial J}{\partial p_i} = 2 \sum_{n=2r}^{2N} \prod_{m=0}^{r-1} (i-m)(n-i-m) p_{n-i} \frac{t_f^{n-2r+1}}{n-2r+1} \tag{13}$$

Then equation (13) is differentiated again with respect to each of the polynomial coefficients p_l. Note that now $n = i + l$.

$$\frac{\partial^2 J}{\partial p_i \partial p_l} = 2 \prod_{m=0}^{r-1} (i-m)(l-m) \frac{t_f^{i+l-2r+1}}{i+l-2r+1} \tag{14}$$

B. Boundary Constraints

The constraints at the start and end points of the trajectory need to be specified. They can be imposed either to satisfy constraints on states, or to ensure continuity of the derivatives. Equation (15) formulates the linear equality constraints.

$$\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b} \tag{15}$$

$$\boldsymbol{A} = \begin{bmatrix} A_0 \\ A_f \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} b_0 \\ b_f \end{bmatrix} \tag{16}$$

The quadratic program (11) can be solved with standard quadratic programming solvers. The solver returns an optimal set of polynomial coefficients $\boldsymbol{p}$ that minimizes the cost function while satisfying the boundary constraints.

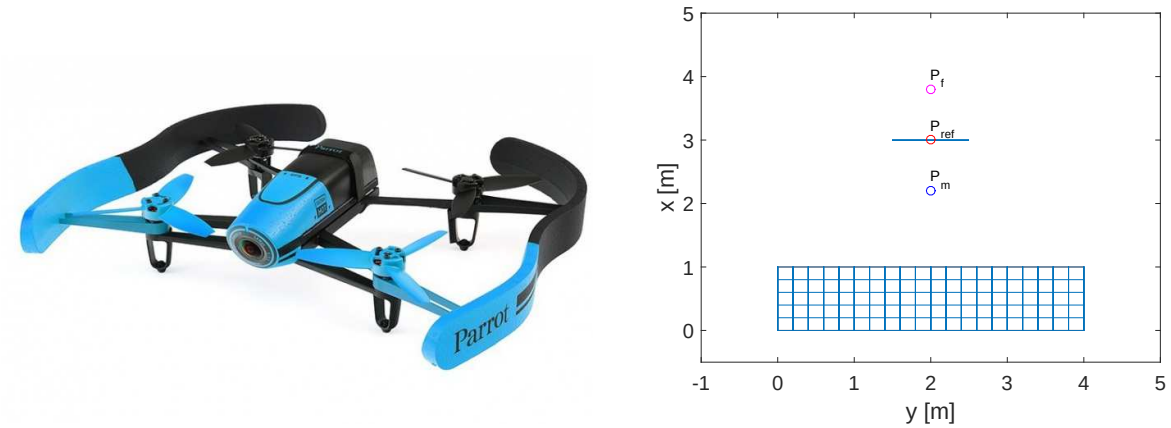
IV. Experimental Set-up

This section describes the facilities used for the flight tests, as well as the software details for the approach.

A. Hardware

The flight tests were performed in the Cyberzoo at TU Delft. The Cyberzoo is a ten by ten meter space equipped with a motion capture system, the OptiTrack, using twelve cameras to trace the 3D coordinates and attitudes of the drone. For all test flights, the 3D coordinates are recorded with the system and they are also used by position controller in closed-loop tests. The attitudes are obtained with on-board AHRS and are used for attitude controller in both closed-loop and open-loop flights. Notice that since both closed-loop and open-loop flight use the same sensors and controllers, the bias in sensor measurements can be assumed consistent throughout the experiment.

The Parrot Bebop drone, shown in figure 2(a), is chosen for the experiments. Two drones, are used to test the sensitivity of the approach with respect to hardware.



(a) Parrot Bebop Drone [c]

(b) 2D representation of the experimental set-up

Figure 2. Parrot Bebop drone used for flight tests and experimental set-up.

B. Software

The on-board controls are programmed with Paparazzi[d]. For this experiment, we only generate and test the two-dimensional trajectory since the drone is also equipped with sonar for altitude detection. The yaw

[c] Parrot Bebop. https://www.parrot.com/us/sites/default/files/styles/product_teaser_display/public/ps/2691-large-parrot-2691jpg.jpg?itok=RNsHN9d7

[d] Paparazzi UAV. http://wiki.paparazziuav.org

angle, ψ, is also not actively controlled, but is kept the same as the original yaw angle throughout the entire trajectory. The trajectory generator minimizes the total snap of the trajectory and outputs the reference coordinates x_{ref} and y_{ref}. The reference positions are controlled by the PID position controller which runs at a frequency of 512 Hz. This is the guidance loop in figure 3. The calculated commands in earth frame are transformed to body frame using the following equations.

$$\begin{aligned} \phi_{ref} &= -sin(\psi) \cdot x_{cmd} + cos(\psi) \cdot y_{cmd} \\ \theta_{ref} &= -cos(\psi) \cdot x_{cmd} - sin(\psi) \cdot y_{cmd} \end{aligned} \tag{17}$$

The reference Euler angles are then tracked with the attitude loop. They are also simultaneously logged

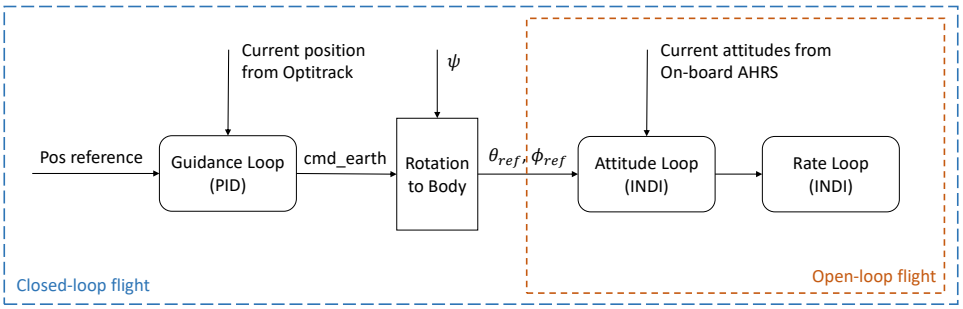


Figure 3. Illustration of control loops.

into a `.csv` file at a rate of 512 Hz and are read at the same frequency during the open-loop flight. The attitude controller and the inner INDI rate controller in closed-loop flights are identical to the open-loop ones.

The set-up used during the experiments, is shown in figure 2(b). It is assumed that the drone needs to fly through a one by one meter gate, depicted with the blue line. This artificial gate will also be used for the feasibility analysis. If the drone successfully flies through the gate with no external position information, and the standard deviation of multiple open-loop flights is smaller than the half of the gate width(0.5 meter), the approach can be concluded feasible. To construct the trajectory library, we generate a grid area in front of the gate. Trajectories are generated between each grid point and point P_m. At P_m, the velocity in x peaks while the velocity in y reaches zero. This defines the boundary conditions in equation (15), and ensures that the drone flies through the gate. P_{ref} is the reference point for visual detection and P_f is the point for next stage in the future.

V. Results

A. Open-loop flight with one closed-loop input data

First, a closed-loop flight is conducted to follow the designed trajectory and the inputs to the system, namely, ϕ, θ and thrust force T, are recorded. It is expected that the open-loop flights exhibit some deviations due to disturbances and therefore, 15 open-loop flights are performed with the logged data to draw statistical conclusions. Figure 4 shows the open-loop trajectories.

It is unsurprising that although with the same control inputs, the open-loop trajectories differ. However, as can be seen in figure 4, the open-loop flights appear to follow a different trajectory which deviates to the right of the gate. To ensure that the open-loop flights have executed the given inputs, the differences between closed-loop and (one of the) open-loop attitudes and thrusts are shown in figure 5.

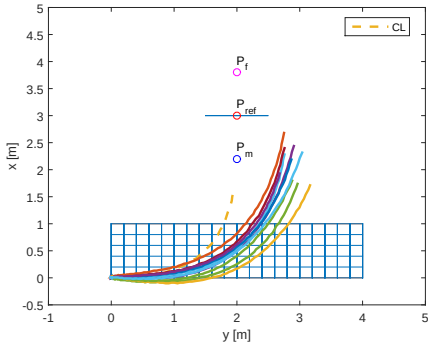
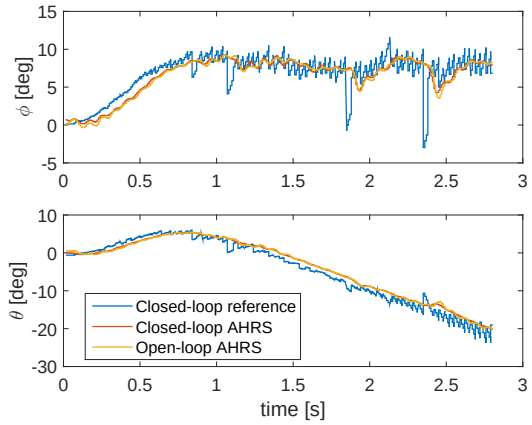
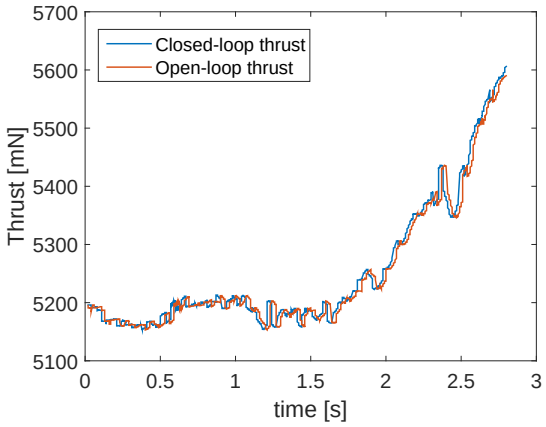


Figure 4. 2D trajectory plot of 15 open-loop flights, with control inputs from one closed-loop flight.

In figure 5(a), the blue lines show the calculated reference angles from the position controller. The reference angles are processed by the attitude controller and the attitudes measured by AHRS are depicted with the red lines. The yellow lines show the same parameters from one of the open-loop flights. The differences between thrust forces can also be seen in figure 5(b). It can be concluded that the open-loop flight did follow the given inputs. The reason for the deviations therefore requires further investigation.



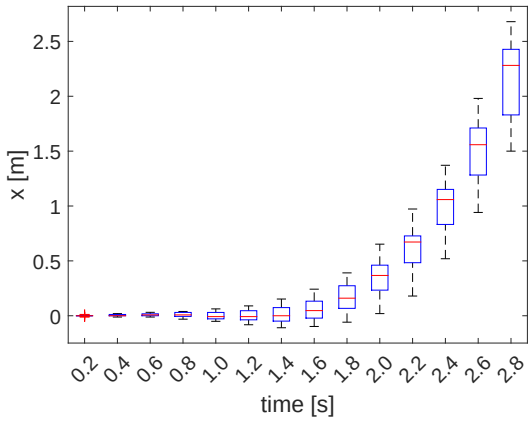
(a) The command angles from closed-loop(Blue), the angles measured by the on-board AHRS in closed-loop(Red) and open-loop(Yellow) flight.



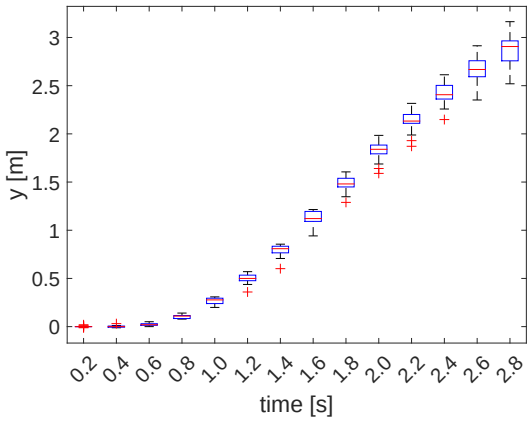
(b) The thrust force from closed-loop and open-loop flight.

Figure 5. Comparison of system inputs from closed-loop and (one) open-loop flight.

Figure 6 indicates the deviations between the open-loop flights at 14 sample points. The points are taken with an interval of approximately 0.2 seconds(100 data points). The deviations in both x and y directions increase with time. The maximum deviation reaches around one meter in x direction at the end of the trajectory.



(a) Box plot of 15 open-loop flights with control inputs from one closed-loop flight, coordinate x.

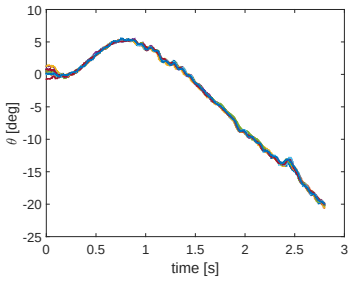


(b) Box plot of 15 open-loop flights with control inputs from one closed-loop flight, coordinate y.

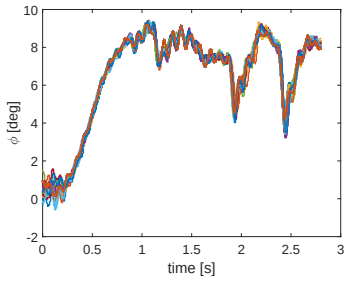
Figure 6. Box plot of 15 open-loop flights with control inputs from one closed-loop flight.

The time history of the open-loop control inputs are shown in figure 7. Differences in thrust force are fairly small and can be assumed to be negligible. A difference of approximately two degrees in both θ and ϕ can be identified in the first 0.2 second of the trajectory. This is due to the discrepancy in the initial conditions. This could be one of the reasons for the deviations between open-loop flights.

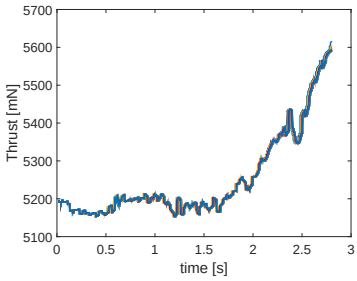
However external disturbances and the discrepancy in initial conditions can only explain the differences between the open-loop flights. The reason why all open-loop flights deviate to the right of the gate remains



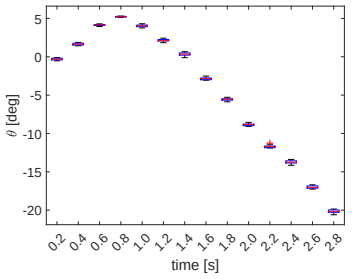
(a) Pitch angle of 15 open-loop flights, control inputs from one closed-loop flight.



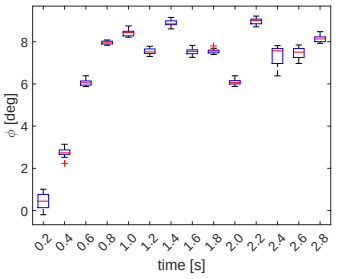
(b) Roll angle of 15 open-loop flights, control inputs from one closed-loop flight.



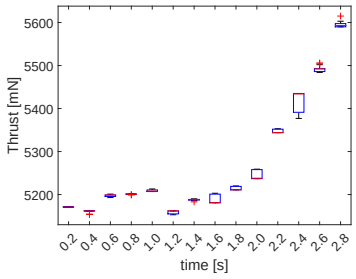
(c) Thrust force of 15 open-loop flights, control inputs from one closed-loop flight.



(d) Box plot of pitch angle from 15 open-loop flights, control inputs from one closed-loop flight.



(e) Box plot of roll angle from 15 open-loop flights, control inputs from one closed-loop flight.



(f) Box plot of thrust force from 15 open-loop flights, control inputs from one closed-loop flight.

Figure 7. Comparison between system inputs from 15 open-loop flights, control inputs from one closed-loop flight

unknown. It is suspected that an external disturbance was present during this particular closed-loop flight, and the controllers compensated the disturbance with for example, larger roll angle and thrust force. The expected direction of the disturbance is shown in figure 8. This corresponds to the reference ϕ angle shown in figure 7(b). The roll angle is always positive, meaning that the drone rolls to the right to overcome the disturbance in the direction of the arrow.

The obtained results indicates that the flight conditions differ from time to time. The corresponding control inputs are calculated to compensate for the very particular flight condition. Simply replaying the control inputs from one closed-loop flight may lead to a different trajectory than expected. Thus more closed-loop flights need to be performed and analysed.

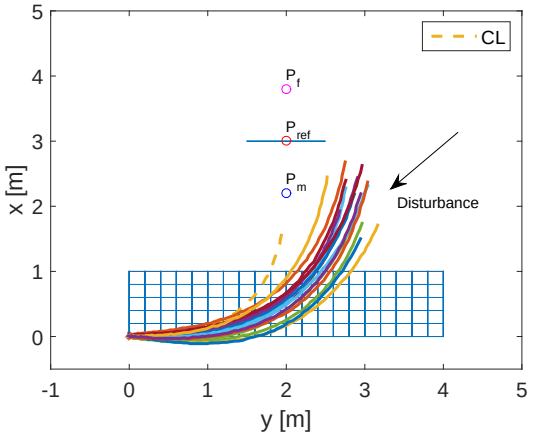


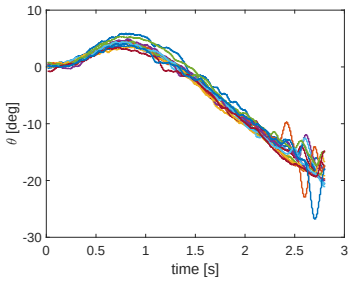
Figure 8. 2D trajectory plot of 15 open-loop flights, with the direction of expected disturbance depicted.

B. Analysis on Closed-loop Flights

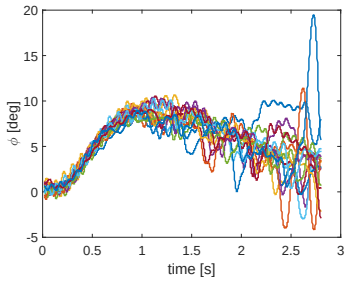
In this section, a further investigation of the closed-loop flight is carried out. Fifteen closed-loop flights were performed here to follow the same trajectory and the control inputs were recorded and analysed. Figure 9 provides a comparison of the fifteen closed-loop inputs θ, ϕ and thrust T.

As shown in the figures, there exists a significant difference between the closed-loop control inputs, especially in the roll angle ϕ. Multiple outliers can be seen in the box plot, and the largest difference between the maximum and minimum roll angle reaches more

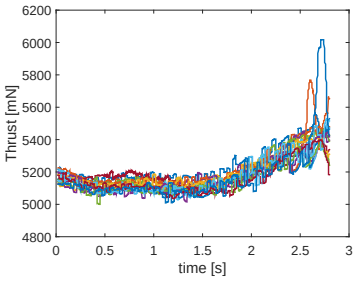
than 12 degrees at $t = 2.6s$. The calculated θ and T follow more or less a consistent trend, despite of some large deviations at the end of the trajectory. The reason behind the deviation in closed-loop flight can be



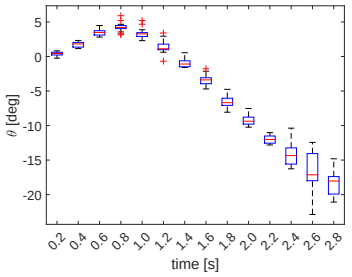
(a) Pitch angle of 15 closed-loop flights.



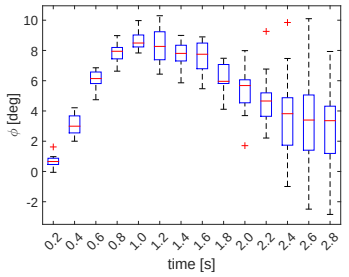
(b) Roll angle of 15 closed-loop flights.



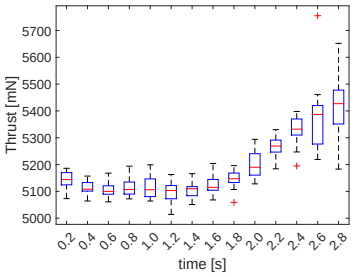
(c) Thrust force of 15 closed-loop flights.



(d) Box plot of pitch angle from 15 closed-loop flights.



(e) Box plot of roll angle from 15 closed-loop flights.



(f) Box plot of thrust force from 15 closed-loop flights.

Figure 9. Comparison between system inputs from 15 closed-loop flights.

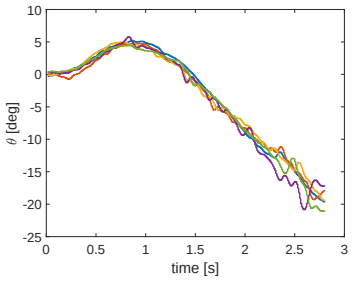
complex. Some possible reasons are listed below.

- The hardware of this particular drone is defect and provides unreliable data.
- The battery level influences the control inputs.
- The aerodynamic forces differ from flight to flight. Thus the corresponding control inputs differ. This includes the external aerodynamic disturbances, rotor-rotor interaction, etc.
- The control inputs deviate due to ground effect. Flights with lower altitude are aerodynamically influenced more by ground effect compared to the ones with higher altitude.

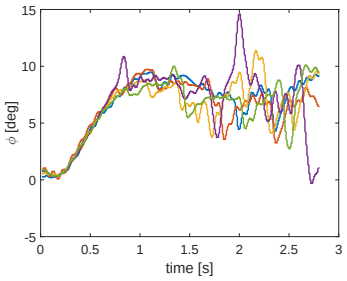
Unfortunately it is difficult to confirm what the exact problem is with the experiments. It perhaps requires a wind-tunnel test to quantitatively model the effect of aerodynamic forces on the drone during the flight. However, the hardware defection, battery level and ground effect can be easily tested.

1. Hardware

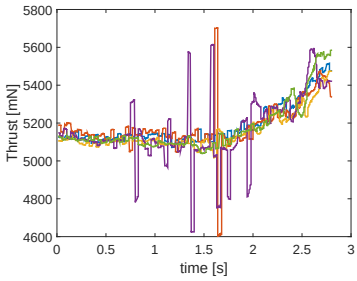
The proposed approach heavily relies on the sensor readings. As seen before, the largest deviation appeared in the roll angle ϕ, which is measured by the on-board AHRS. To verify whether the AHRS on the tested drone measures reliable attitude, we repeat the same trajectory on a different Bebop drone. In figure 10, control inputs θ, ϕ and T from five closed-loop flights are presented. Similar to the previous results, control inputs from only one closed-loop flight cannot be directly used, since different behaviors are seen from the five flights. Again, roll angle shows the largest inconsistency. Since large fluctuations still appear with the second or even a third drone, it can be concluded that either the on-board sensors play a less important role in the closed-loop measurement discrepancies, or the measurement accuracies are limited by the Bebop hardware. Either case the problem can not be solved within the scope of this work. On the other hand, we can identify related behavior in figure 10(b) and 10(c) (peaks in the purple line). Further studies are required to determine the cause of the behavior.



(a) Pitch angle of 5 closed-loop flights, performed with an alternative drone.



(b) Roll angle of 5 closed-loop flights, performed with an alternative drone.



(c) Thrust force of 5 closed-loop flights, performed with an alternative drone.

Figure 10. Comparison of system inputs from 5 closed-loop flights, performed with an alternative drone.

2. *Ground Effect*

Here we would like to analyse whether the flight altitude has an influence on the closed-loop attitudes and thrust. Before the analysis can be done, the performance of closed-loop control inputs needs to be quantified. With 15 closed-loop flights, the averaged ϕ, θ and thrust force can be calculated, and the extent to which each closed-loop control input deviate from the average can be used as a measure of the performance. Mathematically, equation (18) defines the mean squared error for each control variable U_i.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(U_i - \hat{U}_i)^2 \tag{18}$$

$$MSE_{total} = \frac{1}{3}[w_\phi \cdot MSE_\phi + w_\theta \cdot MSE_\theta + w_T \cdot MSE_T] \tag{19}$$

where N is the total number of data points. U_i represents the control input ϕ, θ or thrust at the i^{th} time stamp, and $\hat{U}_i$ denotes the corresponding average value. To use a single metric, the weighted total MSE is calculated with equation (19), where w_ϕ, w_θ and w_T are the weights corresponding to the roll, pitch and thrust, respectively. Table 1 shows the MSE of the three control inputs, the total weighted MSE, together with the mean flight altitude $\bar{z}$. The total MSE is calculated with equal weights, namely, $w_\phi = w_\theta = w_T = 1$. The weights can be further adjusted according to the trajectory. For example, roll angle can be given higher weight if the accuracy in y direction is more crucial.

From Table 1, CL #2 and #13 deviate the least from the average. The altitude $\bar{z}$ does not appear to be correlated to the performance since in CL #1 and CL #13, the drone almost flew at the same altitude while the control inputs, differ drastically. Therefore it is evident that the closed-loop flights, are not significantly influenced by ground effect.

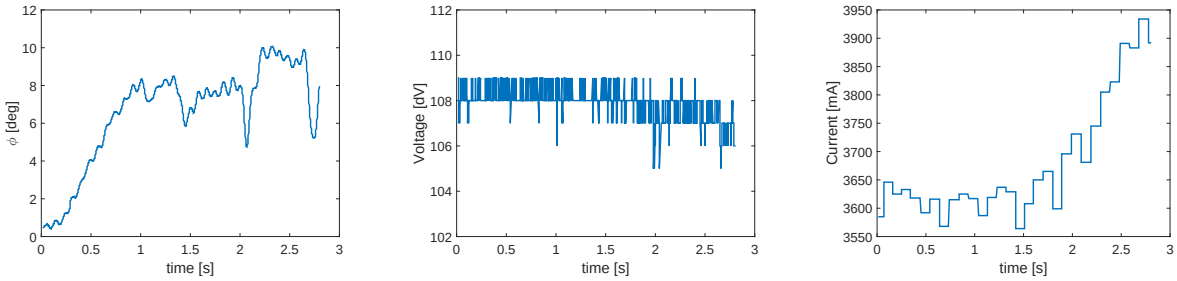
3. *Battery Level*

Two parameters regarding the battery information are recoded during each flight: the voltage and current level. In this subsection, we wish to investigate whether these two parameters have a significant influence on the closed-loop control inputs. Figure 11 shows the control input ϕ and the logged battery voltage and current from the closed-loop flight(CL #1) described in section A. The voltage appears to be stable and no abnormal behavior can be seen in the current(figure 11(c)).

More information is required to draw a conclusion on whether the battery level played a role. Figure 12(a) compares the roll angle ϕ, battery voltage and current of two closed-loop flight. The roll angles start to deviate from each other at around $t = 2s$. For CL #5, the roll angle increases first from zero, then continues to decrease, while for CL #1, ϕ increases again after approximately 2.2 seconds. We however do not see similar patterns in the battery voltage and current level. There exist some small fluctuations in the voltage at $t = 2s$ (CL #1), but it is considered negligible compared to the normal voltage level. Although, the change in voltage and fluctuation in ϕ do appear to be correlated, since similar fluctuation in voltage is seen from CL #5 at around $t = 2.3s$, as well as a small decrease in ϕ at the same time.

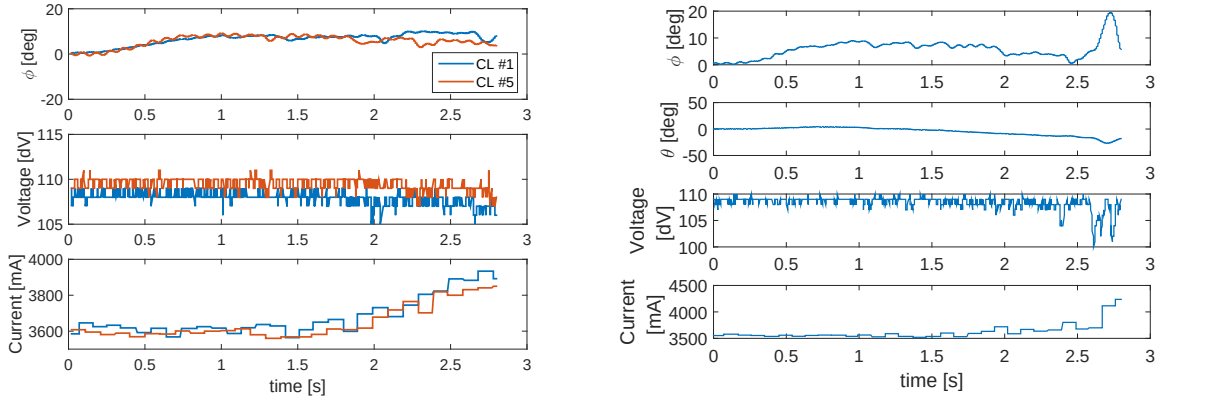
Table 1. Calculated MSE of 15 closed-loop flights, and the corresponding flight altitude

	MSE_ϕ	MSE_θ	MSE_T (normalized)	MSE_{total}	$\bar{z}$
CL #1	6.6157	1.6306	0.4091	2.8851	2.5948
CL #2	0.8607	0.3626	0.1427	0.4553	1.9718
CL #3	1.3810	0.5362	0.0765	0.6646	0.5375
CL #4	2.4802	0.7980	0.2085	1.1622	1.1426
CL #5	1.1773	0.7995	0.1017	0.6928	1.4793
CL #6	2.0692	0.6812	0.1468	0.9657	1.6490
CL #7	1.6320	0.3596	0.2433	0.7449	1.5743
CL #8	2.6996	0.3892	0.1371	1.0753	0.9066
CL #9	5.4107	1.8320	0.4980	2.5802	1.0019
CL #10	1.1015	0.7246	0.1063	0.6441	1.7641
CL #11	1.0757	0.2722	0.1253	0.4910	2.1696
CL #12	0.8725	0.9127	0.2577	0.6809	0.5796
CL #13	0.8958	0.3463	0.1825	0.4748	2.5429
CL #14	1.4641	1.0257	0.1985	0.8961	2.1428
CL #15	7.6439	2.1722	1.1378	3.6513	2.1269



(a) Pitch angle of CL #1. (b) Battery voltage of CL #1. (c) Battery current of CL #1.

Figure 11. Logged control inputs from CL #1 and battery information.



(a) Roll angle, battery voltage and current of CL #1 and CL #5.

(b) Roll angle, pitch angle, battery voltage and current of CL #15.

Figure 12. Closed-loop flight attitude and battery information.

However, the change in voltage is not related to the increase in roll angle at $t = 2.2s$ from CL #1 because similar behavior is not present in CL #5. It can therefore be concluded that for CL #1, the large change in roll angle is not related to the battery level.

Upon further investigation of the rest of the closed-loop flights, a strong correlation between the roll angle was discovered in one particular closed-loop flight. This is depicted in figure 12(b). The voltage starts stable, and so do the control inputs ϕ and θ. At $t = 2.6s$, a sudden change in ϕ, θ and voltage can be seen. The changes start almost simultaneously and it is difficult to identify which variable is the original cause. Nevertheless, the voltage level can be used as an indication of whether the control inputs are stable and reliable.

In this section, it is evident that using the control inputs from only one closed-loop flight does not guarantee the accuracy of open-loop flight. Due to large deviations between closed-loop inputs, it is recommended to use the inputs with the smallest total MSE.

C. Open-loop flight with closed-loop input data with smallest total MSE

As discussed previously, the total MSE serves as a performance index for the closed-loop flights. To perform open-loop flights, the set of control inputs with smallest MSE_{total} should be used. From Table 1, this corresponds to CL #2. Figure 13 shows the trajectories of 15 open-loop flights with the control inputs of CL #2. Compared to the previous test results(with CL #1), large number of open-loop flights follow a more accurate trajectory. To illustrate the results more clearly, figure 14 depicts the box plots of coordinate x and y. Again similar to the results before, the variances in both directions increase with time. At $t = 2.8s$, the difference between the maximum and minimum of all 15 flights, reaches 0.7461 and 0.3564 meter in x and y direction, respectively. In addition, the difference between 25^{th} and 75^{th} percentile is 0.2813 and 0.1211 meter in x and y direction, respectively. This variance is considered reasonable and expected for open-loop flights of 2.5 seconds.

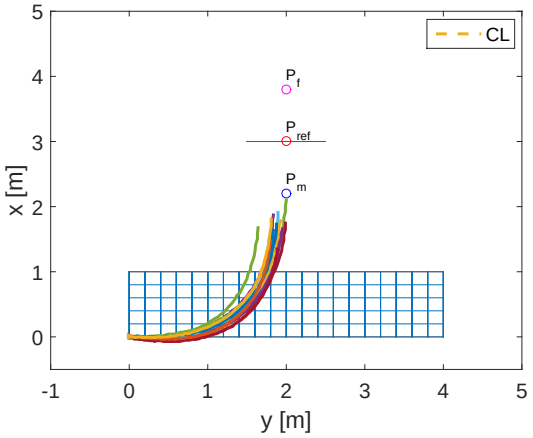
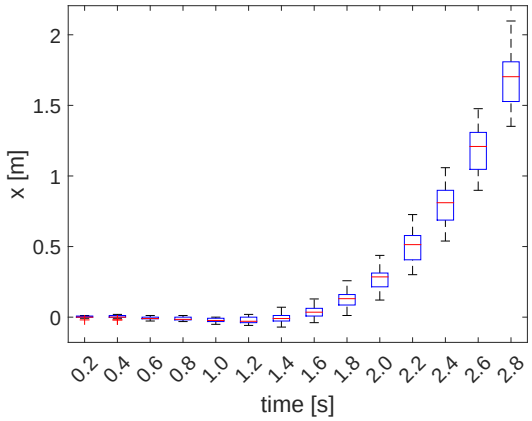
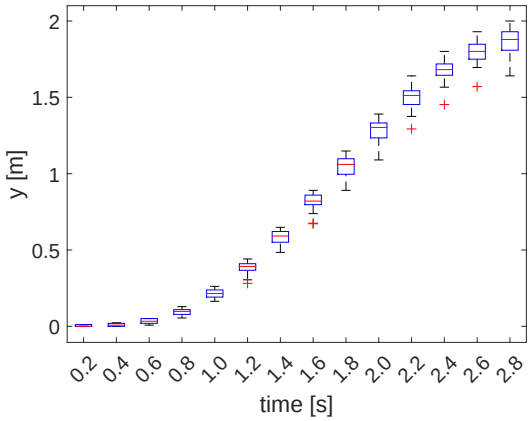


Figure 13. 2D trajectory plot of 15 open-loop flights, with inputs from CL #2.



(a) Box plot of 15 open-loop flights with inputs from CL #2, coordinate x.

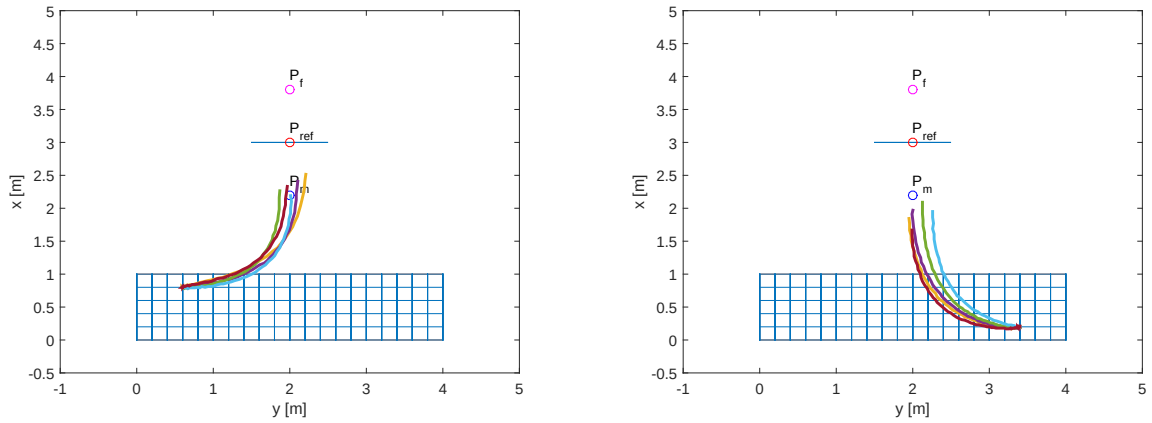


(b) Box plot of 15 open-loop flights with inputs from CL #2, coordinate y.

Figure 14. Box plot of 15 open-loop flights, with inputs from CL #2.

To further verify the results, the experiments are repeated on two other grid points. Five open-loop

flights are performed for each grid point using the closed-loop flight with smallest MSE. Results confirm the feasibility of the approach.



(a) Five open-loop flights from one selected grid point(x5,y4). (b) Five open-loop flights from one selected grid point(x2,y18).

Figure 15. Two additional experiments from selected grid points.

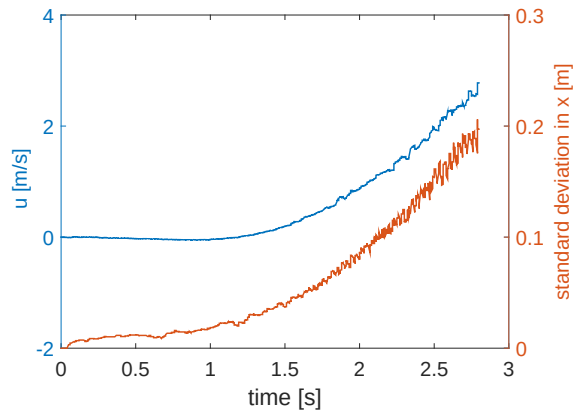
Figure 16 shows the standard deviation in x and y direction with respect to time on the right axis. On the left axis, the velocity u and v, corresponding to x and y directions respectively is plotted. In figure 16(a), the velocity u increases with time and so is the standard deviation. The velocity in y first increases to around 1.5 m/s and then returns to zero at the end of the trajectory. This is defined by the boundary conditions in the minimum snap trajectory generation described in section III. Standard deviation in y continues to increase through out the entire trajectory. It can be noticed around $t = 2s$, that the slope of the standard deviation in figure 16(b) starts to decrease. This means that the decreasing velocity v may have an influence on the standard deviation, and the influence only becomes prominent at the end of the trajectory because of accumulated error.

Since there is no position feedback, errors made in the beginning of the flight cannot be corrected. The errors are accumulated and result in larger deviations at the end of the trajectory. From figure 16, the accumulated error has a larger influence on the standard deviation compared to velocity. This can be seen from figure 16(b), where only the slope of standard deviation decreases at the end, but the standard deviation appears to have a stronger relationship with time. To further verify the observation, two extra experiments with different velocity profiles were performed. Since the durations of the trajectories are different, we will only look at the first 2 seconds of the flights.

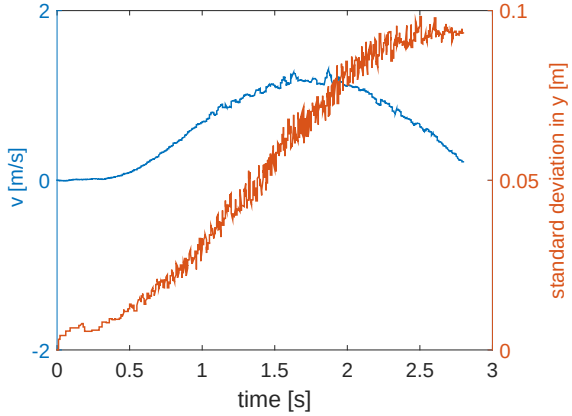
Figure 17(a) and 17(a) show three different velocity profiles where Profile 1 being the fastest and Profile 3 the slowest in order to verify whether the difference in velocity has significant influence on standard deviation. The results are depicted in figure 17(c) and 17(d), where the corresponding standard deviations are plotted. Here the influence of velocity on standard deviation can be clearly seen. In both figure 17(c) and 17(d), the largest standard deviation is seen in the highest velocity profile (Profile 1). This means that higher velocity leads to larger standard deviations. However, notice that time has a more significant influence compared to velocity. Profile 1, with highest velocity, the trajectory is completed in 1.5 seconds, with standard deviation in x of 0.1 m. Although at $t = 1.5s$, profile 3 shows a lower standard deviation, the final value peaks at 0.2 m at the end of the trajectory. Hence a higher velocity profile decreases the total time of the trajectory, which at the end, may lead to more accurate open-loop flight.

VI. Conclusion and Recommendations

From the results shown in previous section, the proposed guidance method shows great potential. It was demonstrated that in a short time intervals, the drone successfully followed the designed trajectory without any form of external position sensor. This means that by using a trajectory library constructed with the closed-loop control inputs stored, the drone can navigate to the destination by replaying the corresponding control inputs of the selected trajectory. The approach has the advantage of allowing agile and aggressive

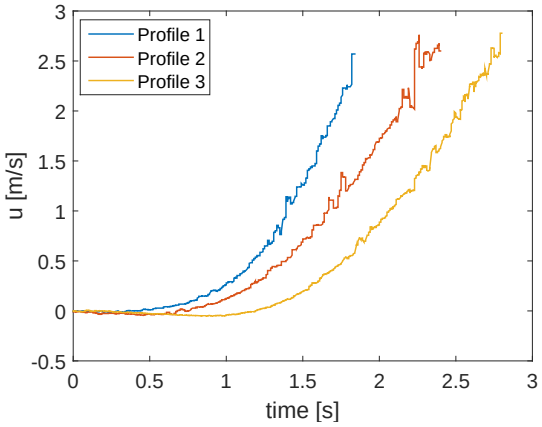


(a) Standard deviation and velocity in x direction.

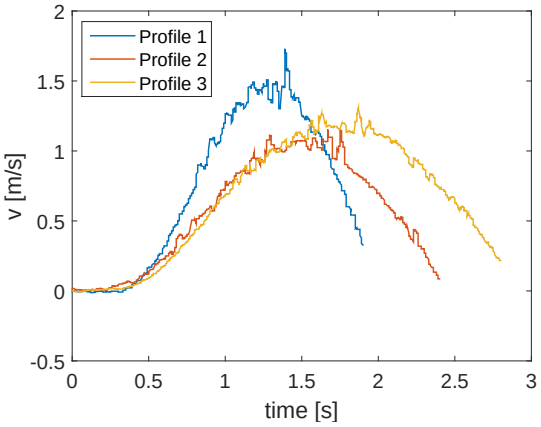


(b) Standard deviation and velocity in y direction.

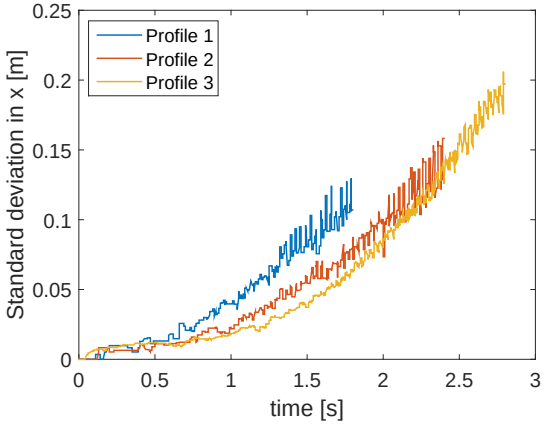
Figure 16. Standard deviation of 15 open-loop flights, with inputs from CL #2, together with the velocities in both x and y directions.



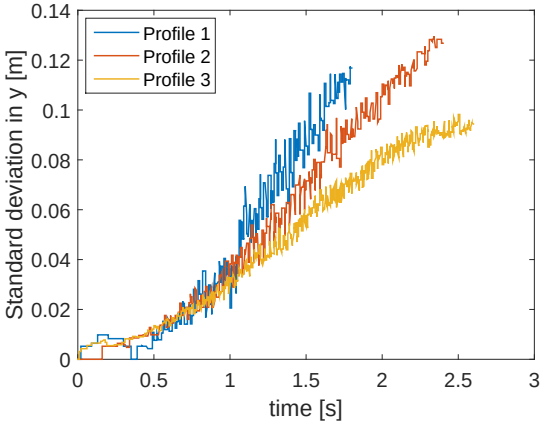
(a) Average velocity in x direction of three velocity profiles.



(b) Average velocity in y direction of three velocity profiles.



(c) Standard deviation in x direction of three velocity profiles.



(d) Standard deviation in y direction of three velocity profiles.

Figure 17. Effect of velocity on standard deviation. Tested with three different velocity profiles.

maneuvers, as well as the wide range of applications. The method can also be extended for obstacle avoidance. Essentially, the approach can be used in any situation where the UAV is required to carry out a trajectory between two fixed points in short intervals.

It was also discovered, that the control inputs of closed-loop flights differ significantly. Hardware, ground effect and battery level have been ruled out from the possible reasons for the difference. It is suspected that aerodynamic effect plays an important role, but further investigation is required to verify the assumption. Repeating the closed-loop flights, control inputs with the smallest MSE can be found. The set of control inputs can then be used for open-loop flight to obtain more accurate results. Experiments show that the open-loop flights followed the designed trajectory with small errors and the although standard deviation increases with time, it remains within the acceptable level at the end of the flight.

One drawback of the approach is that multiple closed-loop flights are required to determine an accurate set of control inputs. This can be somewhat time consuming if a refined grid is used. Hence it might be necessary to investigate the reason for difference between closed-loop flights in order to greatly reduce the cost. Secondly, although the drone is shown to successfully followed the designed trajectory with small errors, outliers still exist, which means that the method does not guarantee success of the open-loop flight. In addition, since the error accumulates during the trajectory, the approach is only applicable in short time. Using the time and standard deviation plot, the appropriate trajectory duration can be selected.

For further studies, the impact of initial conditions needs to be investigated. More specifically, the initial velocity is expected to have a great impact on the results. Due to limitations in time and facilities, the experiments cannot be performed. In addition, a feed-forward controller is recommended for this application. It eliminates the time lag caused by a feed-back only controller.

Another topic of interest for future research is the metric MSE_{total}. Experiments have shown that closed-loop control inputs with the smallest MSE result in successful open-loop flight. However more investigations are required to verify whether the value of MSE_{total} serves as a direct index of the closed-loop flight performance. The following test plan is recommended to investigate, for example, whether $MSE_{total} \leq 0.5$ leads to open-loop flights with high accuracy.

- Select multiple grid points on both sides of the gate.
- Perform 15-20 closed-loop flights from each grid point and calculate MSE_{total} for each flight.
- Identify the CL flight with $MSE_{total} \leq 0.5$ and save the time history of Euler angles.
- Perform open-loop flights with the saved control inputs and plot the 2D trajectory.
- Verify whether the drone successfully flies through the gate.

If the open-loop flights from multiple grid points are successful, a MSE_{total} of less than 0.5 can be used as a standard for the closed-loop flight performance.

References

[1]Chan, B., Guan, H., Jo, J., and Blumenstein, M., "Towards UAV-based bridge inspection systems: a review and an application perspective," *Structural Monitoring and Maintenance*, Vol. 2, No. 3, 2015, pp. 283–300.

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

[3]Colomina, I. and Molina, P., "Unmanned aerial systems for photogrammetry and remote sensing: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 92, 2014, pp. 79–97.

[4]Nex, F. and Remondino, F., "UAV for 3D mapping applications: A review," *Applied Geomatics*, Vol. 6, No. 1, 2014, pp. 1–15.

[5]He, R., Prentice, S., and Roy, N., "Planning in information space for a quadrotor helicopter in a GPS-denied environments," *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, , No. 2007, 2008, pp. 1814–1820.

[6]Zingg, S., Scaramuzza, D., Weiss, S., and Siegwart, R., "MAV navigation through indoor corridors using optical flow," *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 3361–3368.

[7]Amidi, O., Kanade, T., and Fujita, K., "A visual odometer for autonomous helicopter flight," *Robotics and Autonomous Systems*, Vol. 28, No. 2, 1999, pp. 185 – 193.

[8]Watanabe, Y., Fabiani, P., and Le Besnerais, G., "Simultaneous visual target tracking and navigation in a GPS-denied environment," *2009 International Conference on Advanced Robotics*, 2009, pp. 1–6.

[9]Landry, B., Deits, R., Florence, P. R., and Tedrake, R., "Aggressive quadrotor flight through cluttered environments using mixed integer programming," *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1469–1475.

[10]Hehn, M. and Andrea, R. D., "Quadrocopter Trajectory Generation and Control," *IFAC World Congress*, Vol. 18, No. 1, 2011, pp. 1485–1491.

[11]von Stryk, O. and Bulirsch, R., "Direct and indirect methods for trajectory optimization," *Annals of Operations Research*, Vol. 37, No. 1, 1992, pp. 357–373.

[12]Kahale, E., Castillo, P., and Bestaoui, Y., "Minimum time reference trajectory generation for an autonomous quadrotor," *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings*, 2014, pp. 126–133.

[13]Qi, Y. C. and Zhao, Y. J., "Energy-Efficient Trajectories of Unmanned Aerial Vehicles Flying through Thermals," Vol. 18, No. 2, 2005, pp. 84–92.

[14]Mellinger, D. and Kumar, V., "Minimum snap trajectory generation and control for quadrotors," *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[15]Fliess, M., Lévine, J., Martin, P., and Rouchon, P., "Sur les systmes non linaires diffrentiellement plats." *Comptes Rendus de l Acadmie des Sciences - Series I - Mathematics*, 1992.

[16]Richter, C., Bry, A., and Roy, N., "Polynomial trajectory planning for quadrotor flight in dense indoor environments," *International Conference on Robotics and Automation*, , No. Isrr, 2013, pp. 1–16.

[17]Frazzoli, E., Feron, E., and Dahleh, M. A., "Robust hybrid control for autonomous vehicle motion planning," *2000. Proceedings of the 39th IEEE Conference on Decision and Control*, Vol. 1, No. iii, 2000, pp. 821—-826 vol.1.

[18]Paranjape, A. A., Meier, K. C., Shi, X., Chung, S. J., and Hutchinson, S., "Motion primitives and 3-D path planning for fast flight through a forest," *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 2940–2947.

[19]Majumdar, A. and Tedrake, R., "Funnel Libraries for Real-Time Robust Feedback Motion Planning," *arXiv*, 2016, pp. 1–45.

Part II

Appendix

Appendix A

Literature Review on Quadrotor Modeling and Control

A-1 Quadrotor dynamics and modeling

In this section, the dynamics of the quadrotor is illustrated and the nonlinear model is derived. Firstly the reference frames and the basics of quadrotor dynamics are explored, followed by the derivation of equations of motion, from which the state-space model is constructed.

A-1-1 Reference frame

Two reference frames are used to describe the motion of the quadrotor: the Earth-fixed inertial frame $\mathcal{F}_E$, and the body-fixed frame $\mathcal{F}_B$. The Earth-fixed reference frame $\mathcal{F}_E$ is shown in figure A-1, where the North-East-Down (NED) convention is used. The X axis points to the North, Y axis to the East, and the Z axis perpendicular to the XY plane, pointing down to the center of the Earth.

The body reference frame $\mathcal{F}_B$, illustrated in figure A-2, is fixed to the center of mass of the quadrotor. The positive X_B axis points to the front, Y_B to the right of the drone, and Z_B perpendicular to the $X-Y$ plane following the right-hand rule. The four rotors, denoted with W_1, W_2, W_3 and W_4, are shown in the figure as well as their rotating directions. Rotor 1 and 3 rotates counterclockwise, and rotor 2 and 4 rotates clockwise. The quadrotor is in hover mode when all four rotor speeds are the same. By adjusting the rotational speed of different rotors, the six degree of freedom motion can be achieved. Vertical motion is realized by increasing or decreasing all four rotor rotational speeds simultaneously. To achieve roll, pitch and yaw motion, the four rotors are divided into the following groups and the corresponding motion is realized if the rotational speeds of each group are changed in opposite direction.

- (W_1, W_4) and (W_2, W_3): Roll, rotation in X_B direction.

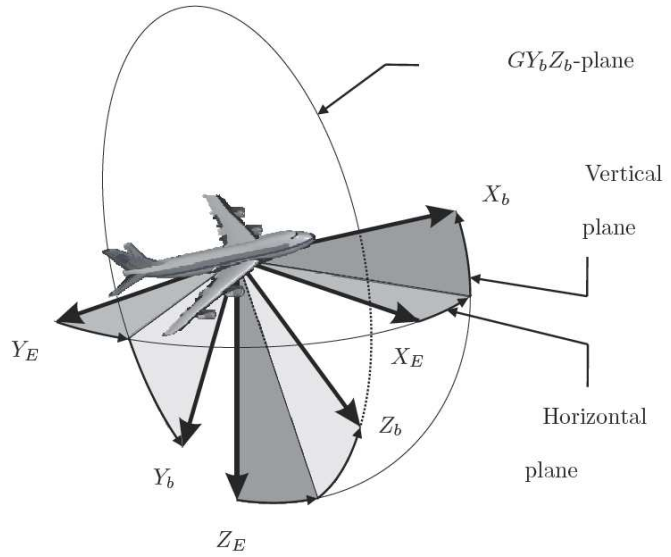


Figure A-1: The earth reference frame $\mathcal{F}_E$, defined with the NED convention (Mulder et al., 2013)

- (W_1, W_2) and (W_3, W_4): Pitch, rotation in Y_B direction.
- (W_1, W_3) and (W_2, W_4): Yaw, rotation in Z_B direction.

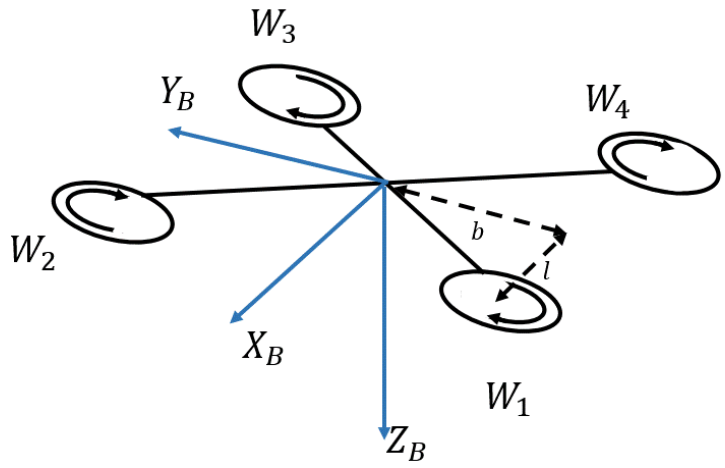


Figure A-2: The body reference frame $\mathcal{F}_B$

The angles between the two reference frames, known as the Euler angles, define the orientation of the quadrotor. Rotation angle about the X_B, Y_B, Z_B axes are denoted as ϕ, θ, ψ,

respectively. Equation (A-1) - (A-3) depict the rotational matrix R used to describe the rotation of quadrotor about an axis.

Rotation about X_B axis by ϕ,

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi) & c(\phi) \end{bmatrix} \tag{A-1}$$

Rotation about Y_B axis by θ,

$$R_y(\phi) = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix} \tag{A-2}$$

Rotation about Z_B axis by ψ,

$$R_z(\psi) = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A-3}$$

where $c. = cos(\cdot)$ and $s. = sin(\cdot)$. The rotation matrix from $\mathcal{F}_B$ to $\mathcal{F}_E$ is given by equation (A-4).

$$R_B^E = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi) \tag{A-4}$$

$$= \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \tag{A-5}$$

Orthogonality of this matrix implies that $R_E^B = (R_B^E)^{-1} = (R_B^E)^T$.

The transformation matrix for angular velocities from $\mathcal{F}_B$ to $\mathcal{F}_E$ is given by equation (A-6) (Cai, Chen, & Lee, 2011).

$$T_B^E = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix} \tag{A-6}$$

where $t(\cdot) = tan(\cdot)$.

A-1-2 Equations of motion

Here the mathematical model will be presented and the equations of motion will be derived with Newton's law and Euler's rotation equations. Vector $\begin{bmatrix} x_E & y_E & z_E & \phi & \theta & \psi \end{bmatrix}^T$ contains the 3D position and orientation of the quadrotor in $\mathcal{F}_E$, while the linear and angular velocities in $\mathcal{F}_B$ can be described with vector $\begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^T$. The variables in $\mathcal{F}_E$ and $\mathcal{F}_B$ can be related with equation (A-7) and (A-8).

$$\begin{bmatrix} \dot{x_E} \\ \dot{y_E} \\ \dot{z_E} \end{bmatrix} = R_B^E \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{A-7}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T_B^E \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{A-8}$$

We denote the linear and angular velocities in $\mathcal{F}_B$ with $\boldsymbol{V_B} = \begin{bmatrix} u & v & w \end{bmatrix}^T$ and $\boldsymbol{\omega_B} = \begin{bmatrix} p & q & r \end{bmatrix}^T$. Following Newton's second law,

$$m\dot{\boldsymbol{V}}_{\boldsymbol{B}} + m\boldsymbol{\omega_B} \times \boldsymbol{V_B} = \sum \boldsymbol{F_B} \tag{A-9}$$

where $\boldsymbol{F_B}$ represents the total external force on the quadrotor in $\mathcal{F}_B$. The external forces consist of gravitational force $\boldsymbol{F_G}$, thrust force from the rotors $\boldsymbol{F_T}$ and drag $\boldsymbol{F_D}$. Notice that the gravitational force needs to be transformed to $\mathcal{F}_B$ since the angular rate, thrust and drag forces can be more easily expressed in the body frame.

$$\sum \boldsymbol{F_B} = R_E^B \cdot \boldsymbol{F_G} + \boldsymbol{F_T} + \boldsymbol{F_D} \tag{A-10}$$

$$= R_E^B \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -T \end{bmatrix} + \begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} \tag{A-11}$$

Where, the drag forces D_x, D_y and D_z are functions of rotor speeds and velocity in x, y, z axis respectively, namely,

$$D_{x,y,z} = f(u, v, w, \Omega_i) \tag{A-12}$$

Combining and rearranging equation (A-9) and (A-11), the linear motion of the quadrotor can be formulated with equation (A-13).

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = R_E^B \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -T/m \end{bmatrix} + \begin{bmatrix} D_x/m \\ D_y/m \\ D_z/m \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{A-13}$$

Angular motion of the quadrotor can be derived from the Euler's rotation equation (A-14).

$$\boldsymbol{I}\dot{\boldsymbol{\omega}}_{\boldsymbol{B}} + \boldsymbol{\omega_B} \times (\boldsymbol{I}\boldsymbol{\omega_B}) = \sum \boldsymbol{M_B} \tag{A-14}$$

Rearrange the terms, (A-15) is obtained.

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \boldsymbol{I}^{-1}\left(\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \boldsymbol{I} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \tag{A-15}$$

where $\boldsymbol{I}$ represents the inertial matrix and is defined in equation (A-16).

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & 0 & I_{xz} \\ 0 & I_{yy} & 0 \\ I_{zx} & 0 & I_{zz} \end{bmatrix} \tag{A-16}$$

A-1-3 Actuator dynamics

Now we consider the system inputs and how they are used to control the behavior of the quadrotor. Let Ω_i be the rotational speed of the i^{th} rotor in rad/s , the thrust force expressed in $\mathcal{F}_B$ is given in equation (A-17)

$$T = \tau_0 \sum_i^4 \Omega_i^2 + (\tau_{11} w + \tau_{12} w^2 + \tau_{13} w^3) \sum_i^4 \Omega_i + \tau_2 (u^2 + v^2) \tag{A-17}$$

From equation (A-17), the force is nonlinear with respect to rotor speeds Ω_i. When the Euler angles are small, the quadrotor can be assumed to be in hover state, where the nonlinearity can be neglected. As explained in previous sections, the difference in rotor speed results in torque about an axis. Thus three degrees of freedom motion can be realized.

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} b\tau_0(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ l\tau_0(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2) \\ C_r(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{A-18}$$

where τ's are the thrust factors and C_r is the coefficient refers to the aerodynamic torque generated by rotor. b and l denotes the distances between rotors and quadrotor center of mass, see figure A-2.

A-1-4 State-space model

The dynamics of quadrotor can be described with the twelve state variables, given in equation (A-19).

$$\boldsymbol{x} = \begin{bmatrix} x_E & y_E & z_E & u & v & w & \phi & \theta & \psi & p & q & r \end{bmatrix}^T \tag{A-19}$$

Combining equation (B-19), (A-8), (A-13) and (A-15), the quadrotor state-space model can be obtained.

$$\begin{bmatrix} \dot{x_E} \\ \dot{y_E} \\ \dot{z_E} \end{bmatrix} = R_B^E \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{A-20}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = R_E^B \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -T/m \end{bmatrix} + \begin{bmatrix} D_x/m \\ D_y/m \\ D_z/m \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{A-21}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T_B^E \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{A-22}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \boldsymbol{I}^{-1} \left(\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \boldsymbol{I} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \tag{A-23}$$

In conclusion, the quadrotor is a highly nonlinear, MIMO and under-actuated system with 6 degrees of freedom. The model will be used further for control system design and simulations.

A-2 Quadrotor control techniques

In this section, the control techniques implemented to ensure that the quadrotor follows the designed trajectory is elaborated. The most commonly applied control algorithm for quadrotors is PID control. The PID controller is preferred for its simplicity and wide application. However, when handling systems with high nonlinearity or large disturbances, problems may occur with PID controller.

An alternative approach, Nonlinear Dynamic Inversion (NDI), has been developed for nonlinear systems. If the model of the vehicle is known, the exact dynamic equations can be used to linearize the system. One major drawback of NDI is its sensitivity to model uncertainties (Sieberling, Chu, & Mulder, 2010). A more model-independent control technique is the Incremental Nonlinear Dynamic Inversion (INDI)where the angular acceleration is measured instead of derived from the model in NDI. The measured angular acceleration removes the need to have an accurate model, and therefore the INDI algorithm is relatively more robust. An adaptive INDI attitude controller is presented in (Smeur, Chu, & de Croon, 2016), as well as a method to filter gyroscope measurements.

Consider Euler rotational equation,

$$\boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} = \sum \boldsymbol{M_B} \tag{A-24}$$

Here, the total moment is related to various parameters. The most general case is applied here, namely, equation (A-25).

$$\sum \boldsymbol{M_B} = \boldsymbol{M}(\boldsymbol{\omega}, \boldsymbol{v}, \dot{\boldsymbol{\Omega}}, \boldsymbol{\Omega}) \tag{A-25}$$

Solve equation (A-24) for $\dot{\boldsymbol{\omega}}$, we can obtain $\dot{\boldsymbol{\omega}}$ as a function of $\boldsymbol{\Omega}, \boldsymbol{v}, \dot{\boldsymbol{\Omega}}$ and $\boldsymbol{\Omega}$.

$$\dot{\boldsymbol{\omega}} = f(\boldsymbol{\omega}, \boldsymbol{v}, \dot{\boldsymbol{\Omega}}, \boldsymbol{\Omega}) \tag{A-26}$$

We take the Taylor expansion to equation (A-26),

$$\begin{aligned}\dot{\boldsymbol{\omega}} \approx\ & \dot{\boldsymbol{\omega}}_0 \\ & + \frac{\partial}{\partial \boldsymbol{\omega}} f|_{\boldsymbol{\omega}=\boldsymbol{\omega}_0}(\boldsymbol{\omega} - \boldsymbol{\omega}_0) \\ & + \frac{\partial}{\partial \boldsymbol{v}} f|_{\boldsymbol{v}=\boldsymbol{v}_0}(\boldsymbol{v} - \boldsymbol{v}_0) \\ & + \frac{\partial}{\partial \boldsymbol{\Omega}} f|_{\boldsymbol{\Omega}=\boldsymbol{\Omega}_0}(\boldsymbol{\Omega} - \boldsymbol{\Omega}_0) \\ & + \frac{\partial}{\partial \dot{\boldsymbol{\Omega}}} f|_{\dot{\boldsymbol{\Omega}}=\dot{\boldsymbol{\Omega}}_0}(\dot{\boldsymbol{\Omega}} - \dot{\boldsymbol{\Omega}}_0)\end{aligned} \tag{A-27}$$

The second and third term is considered negligible compared to the fourth and fifth term (Smeur et al., 2016). Equation (A-27) is then reduced to (A-28).

$$\dot{\boldsymbol{\omega}} = \dot{\boldsymbol{\omega}}_0 + \frac{\partial}{\partial \boldsymbol{\Omega}} f|_{\boldsymbol{\Omega}=\boldsymbol{\Omega}_0}(\boldsymbol{\Omega} - \boldsymbol{\Omega}_0) + \frac{\partial}{\partial \dot{\boldsymbol{\Omega}}} f|_{\dot{\boldsymbol{\Omega}}=\dot{\boldsymbol{\Omega}}_0}(\dot{\boldsymbol{\Omega}} - \dot{\boldsymbol{\Omega}}_0) \tag{A-28}$$

$$= \dot{\boldsymbol{\omega}}_0 + H_1(\boldsymbol{\Omega} - \boldsymbol{\Omega}_0) + H_2(\dot{\boldsymbol{\Omega}} - \dot{\boldsymbol{\Omega}}_0) \tag{A-29}$$

Where all the quantities with subscript zero are measured at current time. The $\dot{\boldsymbol{\Omega}}$ term can be approximated with numerical derivative of $\boldsymbol{\Omega}$, i.e. $\dot{\boldsymbol{\Omega}} = T_s^{-1}(\boldsymbol{\Omega} - \boldsymbol{\Omega} z^{-1})$.

$$\dot{\boldsymbol{\omega}} - \dot{\boldsymbol{\omega}}_0 = H_1(\boldsymbol{\Omega} - \boldsymbol{\Omega}_0) + H_2 T_s^{-1}(\boldsymbol{\Omega} - \boldsymbol{\Omega} z^{-1} - \boldsymbol{\Omega}_0 + \boldsymbol{\Omega}_0 z^{-1}) \tag{A-30}$$

Now we define the incremental control command $\Delta\boldsymbol{\Omega} = \boldsymbol{\Omega} - \boldsymbol{\Omega}_0$. The relationship between $\Delta\boldsymbol{\Omega}$ and the error in $\dot{\boldsymbol{\omega}}$ can be obtained. By choosing $\dot{\boldsymbol{\omega}}$ as a virtual control and solving equation (A-28) for $\boldsymbol{\Omega}$, we can determine the required control command $\boldsymbol{\Omega}_c$. The input to the INDI controller is then the virtual control signal, and the controller outputs desired control command $\boldsymbol{\Omega}_c$. The INDI control scheme can be used in the inner-loop for angular rate control, and a simple PID controller can be used for the outer attitude control loop.

Appendix B

Literature Review on Optimization Methods and Trajectory Library

B-1 Trajectory optimization

In this section, a literature study on trajectory optimization is be performed. Firstly the numerical methods based on optimal control are elaborated, followed by the differential flatness based approach. The latter only applies to differentially flat dynamic systems, however greatly simplifies the optimization problem. In addition, the choice of optimization objective is briefly discussed. Figure B-1 provides an overview of the field of trajectory optimization. Notice that the section will mainly focus on the UAV specific research, for more detailed analysis, reference to (Betts, 1998; Rao, 2009; Geiger, 2009).

The trajectory optimization problem, or sometimes referred as the optimal control problem, concerns the design of a trajectory that optimizes some performance index while satisfying various constraints. Mathematically the optimal control problem aims to find the trajectory which minimizes the following cost function.

$$\min_{\boldsymbol{x}(t),\boldsymbol{u}(t)} J = G[t_0, \boldsymbol{x}(t_0), t_f, \boldsymbol{x}(t_f)] + \int_{t_0}^{t_f} L[\boldsymbol{x}(t), \boldsymbol{u}(t), t]dt \tag{B-1}$$

Subject to the system dynamics equations (B-2)

$$\dot{\boldsymbol{x}} = f[\boldsymbol{x}(t), \boldsymbol{u}(t), t] \tag{B-2}$$

and other constraints in control input, time, states or path

$$\boldsymbol{c_l} \leq \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{u}, t) \leq \boldsymbol{c_u} \tag{B-3}$$

Notice that equation (B-1) is minimized over *funcions* $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$. Thus the problem is also known as functional optimization.

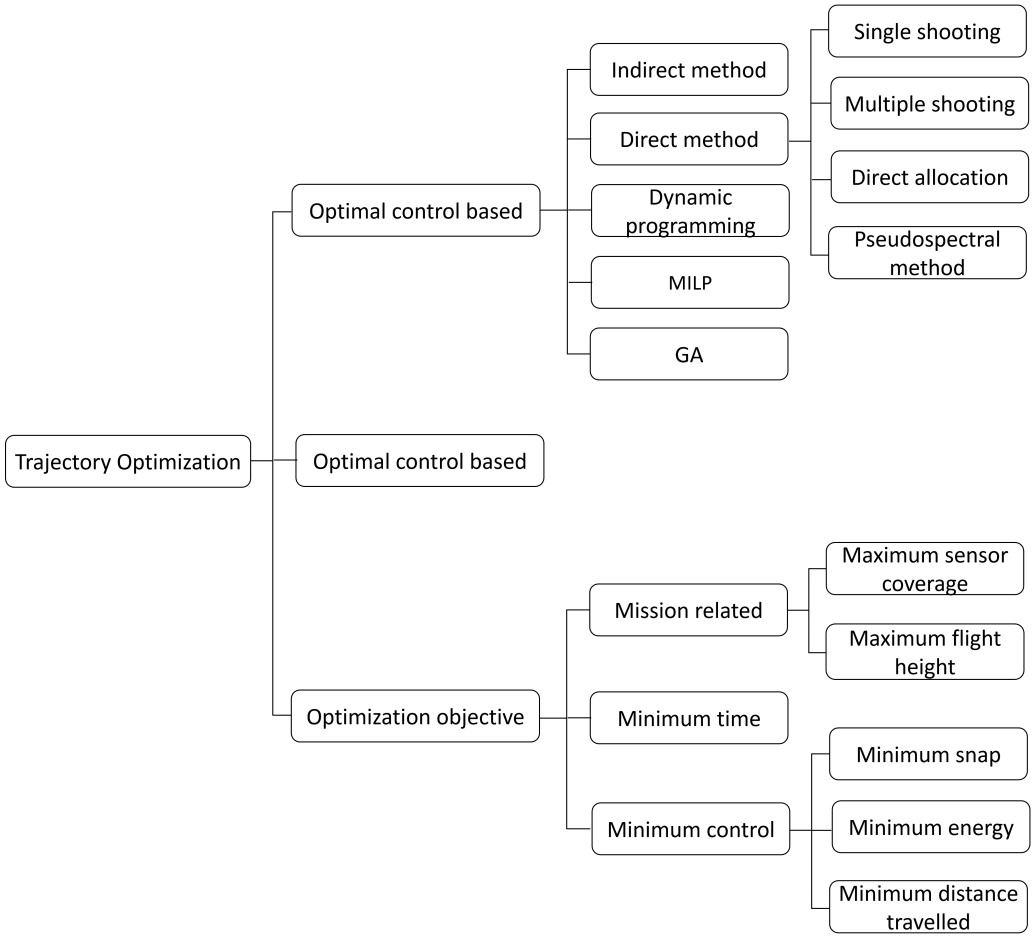


Figure B-1: Overview of the field of trajectory optimization. Two aspects are commonly considered, the optimization method and the optimization objective. The former refers to the numerical method to solving the optimization problem, and the latter defines the cost function.

B-1-1 Indirect method

The indirect method is based on calculus of variations. It aims to find the functions that optimize a functional, or in another word, function of a function. With this approach, the necessary optimality conditions need to be derived. The augmented Hamiltonian function, defined with equation (B-4), is typically used (Rao, 2009). In this way, the constrained optimization problem is converted into an unconstrained one.

$$H = L + \boldsymbol{\lambda}^T \boldsymbol{a} - \boldsymbol{\mu}^T \boldsymbol{b} \tag{B-4}$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are coefficient vectors related to$\boldsymbol{\lambda}$, the adjoint and $\boldsymbol{\mu}$, the Lagrange multiplier associated with the path constraint. Optimality requires that the following equation is satisfied.

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H^T}{\partial \boldsymbol{x}} \tag{B-5}$$

Minimizing equation (B-1) with constraints can be transformed to the optimization problem of Hamiltonion function.

$$\min_{\boldsymbol{u}} H(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}, \boldsymbol{\mu}, t) \tag{B-6}$$

Solving for $\boldsymbol{u}$, the optimal control can be determined. Equation (B-6) is also known as Pontryagin's Minimum Principle (Pontryagin, Boltyanskii, Gamkrelidze, & Mishchenko, 1962).

Several researches have been conducted on UAV trajectory optimization using indirect method. In (Hehn & Andrea, 2011), a time-optimal trajectory generation method is developed. The final time t_f is minimized with constraints on the system dynamics, input and states. To ensure dynamic feasibility, the control input is checked after the trajectories have been generated. If the trajectory is found to be infeasible, the maximum input u is reduced and the algorithm re-calculates the trajectory. (Stryk & Bulirsch, 1992) illustrates a more general form of indirect method used in the field of trajectory optimization, as well as the comparison of direct and indirect method. Moreover, a hybrid approach is proposed to overcome the disadvantages of both methods.

B-1-2 Direct method

Contrary to the indirect method, the direct method transforms the original optimization problem into a Nonlinear Program (NLP) problem. This is done by parametrization of the controls or/and state variables. The decision variables are thus converted from functions to real numbers. In other words, the direct method discretizes the problem before optimization. The direct method can be further categorized into two approaches, the *shooting* and *collocation* method.

Direct shooting method

The shooting method only discritizes the controls $u(t)$ to be piecewise linear or piecewise constant, and the state variables $x(t)$ are obtained by numerical integration. After integration, the terminal conditions and errors are computed. The initial conditions are adjusted accordingly at each iteration. Since the shooting method can be sensitive to the changes in boundary conditions, integration can be performed at intermediate points. This effectively breaks the trajectory into small segments. Integration is restarted at the beginning of each segment, and continuity is enforced at the interface of segments. This variation of shooting method is known as the *direct multiple shooting method*.

Direct collocation method

The *direct collocation method*, on the other hand, discritizes both controls and state variables. The state equations and controls are approximated with polynomials $\hat{x}$ and $\hat{u}$ respectively. At midpoint of a segment, system dynamics and constraints are enforced. These midpoints are called the collocation points. The method can be better illustrated with equations below.

Consider a nonlinear system described with differential equation (B-2), the time is discritized into $N-1$ intervals,

$$t_0 = t_1 < t_2 < \ldots < t_N = t_f \tag{B-7}$$

The polynomials approximating x and u can be written as

$$\hat{x} = \hat{x}(x(t_1), x(t_2), \ldots, x(t_N)) \tag{B-8}$$
$$\hat{u} = \hat{u}(u(t_1), u(t_2), \ldots, u(t_N)) \tag{B-9}$$

The discrete variables $x(t_1), x(t_2), \ldots, x(t_N)$ and $u(t_1), u(t_2), \ldots, u(t_N)$ are to be optimized. For ease of notation, we use parameter y defined with equation (B-10) to represent the decision variables.

$$y = (x(t_1), x(t_2), \ldots, x(t_N), u(t_1), u(t_2), \ldots, u(t_N)) \tag{B-10}$$

The search space of the optimization problem is now restricted to the finite dimension. The original optimization problem can be reformulated as followings

$$\min_y \; J(\hat{x}(y), \hat{u}(y)) \tag{B-11}$$

subject to,

$$\dot{\hat{x}} - f(\hat{x}(y), \hat{u}(y)) = 0 \tag{B-12}$$
$$c_l \leq c(\hat{x}, \hat{u}) \leq c_u \tag{B-13}$$

Equation (B-12) and (D-14) enforce system dynamics and constraints at collocation points $t = \frac{t_j + t_{j+1}}{2}, j = 1, 2, \ldots, N$, respectively.

Direct method has been widely applied in UAV related trajectory optimization. (Kahale, Castillo, & Bestaoui, 2014) considers the three dimensional trajectory generation problem. Direct collocation approach is used to transform the problem and the NLP solver is used to determine the time-optimal trajectory. In addition to minimal time, other performance indexes have also been studied. Due to the limited power of UAV, energy efficiency or control effort has been considered as one of the most important parameters to optimize. Direct collocation method is used in (Qi & Zhao, 2005) to determine the optimal energy-efficient trajectories of a UAV flying through a thermal cell. Four parameters are identified to describe the UAV flight. The resulting NLP problem is solved with the software NPSOL [1].

Control effort is another parameter often optimized in UAV trajectory optimization. For a UAV surveillance problem described in (Geiger, 2009), the sensor coverage is maximized while the control effort is minimized. Both direct collocation and pseudospectral methods are implemented, as well as a new method using neural network. It is shown that the neural network method generates equivalent result when compared to the direct collocation and pseudospectral methods, while greatly reduces the computational time.

[1] Nonlinear Programming and the Systems Optimization Laboratory. `http://www.sbsi-sol-optimize.com/asp/sol_product_npsol.htm`

B-1-3 Dynamic programming

Dynamic programming was introduced in 1957 by Richard Bellman (Bellman, 1957). The algorithm divides the problem into stages, with states associated to the beginning of the stage. After the stage is optimized, the result is used to transform the states at the next stage (Hillier, 2014). In (Flint, Polycarpou, & Fernandez-Gaucherand, 2002), a dynamic programming algorithm is implemented to find the optimal trajectories for multiple UAV performing cooperatively search.

B-1-4 Mixed-integer programming

A Mixed-integer Programming (MIP) refers to optimization problems with restrictions on some decision variables having integer values. If both the objective function and constraints are linear, the problem is known as a Mixed-integer Linear Programming (MILP). MIP is commonly applied to optimal trajectory generation involving multiple vehicles. In (Richards & How, 2002), the collision avoidance problem of multiple aircraft is considered. The aircraft is assumed to move in 2D with constant velocity. Dynamic constraints are formulated as the magnitude limit on velocity and force acting aircraft. Collision is avoided by imposing a rectangular exclusion region on each vehicle. A binary variable is assigned to each aircraft, indicating whether the vehicle has reached its destination at each timestep. The minimum time trajectory problem is solved with MILP constraints. Similarly, in (Mellinger, Kushleyev, & Kumar, 2012) the formation flight of multiple quadrotors is solved with MIP algorithm. The individual flight path is planned with Quadratic Program (QP), and integer constraints are used for obstacle avoidance. This makes the quadratic program into a mixed-integer quadratic program.

B-1-5 Genetic algorithm

Genetic Algorithm (GA) has gained its popularity in UAV trajectory optimization. The algorithm was originally proposed in (Holland, 1992) and was inspired by the idea of natural selection and genetics. Selection, crossover and mutation are the three operators used in genetic algorithm. Selection evaluates the fitness of a solution and allows better genes to pass on to the next generation. Crossover simulates the mating process and recombines the individuals selected from the previous step. Mutation introduces small changes to the process. The algorithm is iterated until a solution with good fitness is found. Reference to (Konak, Coit, & Smith, 2006) for a tutorial in GA for multi-objective optimization.

(Pellazar, 1994) developed a route planner to calculate energy efficient trajectories, while incorporating constraints on terrain, obstacles and vehicle dynamics. It is concluded from simulation that GA produces near-optimal 3-D trajectories, although the computation time is relatively high.

B-2 Differential flatness and trajectory optimization

This section describes the definition of differentially flat system and its influence on trajectory optimization.

Differential flatness was firstly proposed by (Fliess, Lévine, Martin, & Rouchon, 1992). It states that a system is said to be differentially flat if all states and inputs can be expressed in terms of a set of outputs and their derivatives. Mathematically, a nonlinear system (B-14)

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \qquad \boldsymbol{x} \in \mathbb{R}^n, \quad \boldsymbol{u} \in \mathbb{R}^m \tag{B-14}$$

is differentially flat if there exists outputs $\boldsymbol{z} \in \mathbb{R}^m$, in the form of equation (B-15)

$$\boldsymbol{z} = h(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}, \ldots, \boldsymbol{u}^{(p)}) \tag{B-15}$$

such that the states and inputs can be written as

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{z}, \dot{\boldsymbol{z}}, \ldots, \boldsymbol{z}^{(\boldsymbol{l})}) \tag{B-16}$$

$$\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{z}, \dot{\boldsymbol{z}}, \ldots, \boldsymbol{z}^{(\boldsymbol{l})}) \tag{B-17}$$

z is then called the flat outputs. For further details about differentially flat systems and examples, reference to (Fliess et al., 1992; Rouchon, Fliess, Lévine, & Martin, 1993).

One of the challenges in optimal control is to perform the optimization while satisfying the system dynamics in the form of *differential equations*. As discussed in section B-1, this can be achieved with various methods. For example, indirect method reformulates the original optimization problem as an unconstrained one, and direct method either uses simulation(direct shooting) or polynomial approximation(direct collocation) to account for the system dynamics. Differential flatness of a system eliminates the need to solve the two-point boundary value problem in state space or numerical integration of system equations. The problem is transformed from a *dynamic* problem to an *algebraic* one. Therefore, trajectory planning space is reduced since the behavior of a differentially flat system can be determined by the flat outputs. More specifically, this means that the trajectories can be planned in flat space, and then convert them to system inputs. Figure B-2 illustrates the relationship between state space and flat space.

(Murray et al., 2002) elaborates the difference between direct collocation method and optimization with differentially flat system. The authors also illustrated the solution to partially flat systems. B-splines are chosen as basis functions to define the trajectories between waypoints. Both system dynamics and constraints are enforced at the collocation points.

Researches have been performed on trajectory optimization of flat system. In (Mellinger & Kumar, 2011), the authors proved that the quadrotor system is indeed differentially flat, and the full state space can be reduced to flat outputs x, y, z, ψ and their derivatives. Piecewise polynomials are used to represent the trajectories between waypoints. Since the moments around x and y axis appear to be functions of the fourth derivatives of the position(also known as the snap), the trajectories are designed to minimize the square of the norm of the snap. Moment around z axis, on the other hand, appear to be functions of the second derivative of yaw angle ψ. Therefore the second derivative of ψ is minimized. The problem can be formulated as a QP and solved with standard method. In (Turpin, Michael, & Kumar, 2012), the problem of quadrotor formation flight in three-dimensional environment is tackled. The relative positions and bearings between quadrotors is described using a shape matrix. The trajectory of each quadrotor is planned independently, aiming to minimize the snap over the trajectory. From experimental results, it is concluded that the performance can be

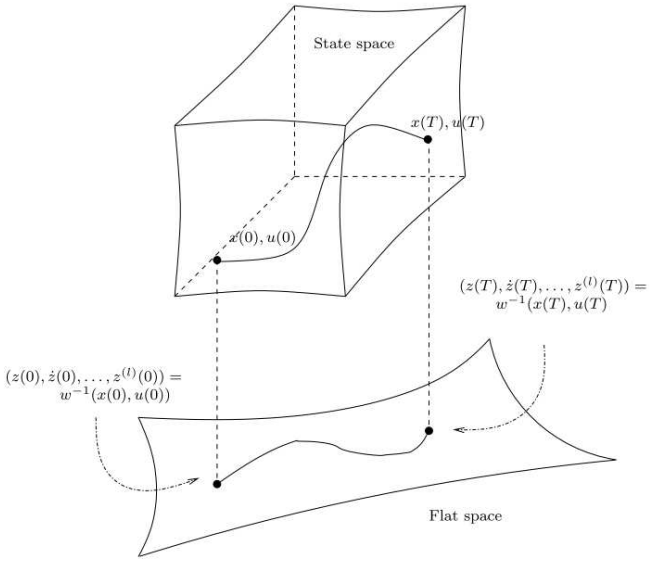


Figure B-2: Relationship between state space and flat space (Milam, 2003)

successfully maintained and with degrading communications, the motions become increasingly aggressive.

Extending the work of (Mellinger & Kumar, 2011), (Richter, Bry, & Roy, 2013) showed that by reformulating the problem as an unconstrained QP, the minimum snap trajectory can be solved with numerical stability for long-range trajectories with many segments. In addition, the time allocated to each segment is incorporated into the optimization. Since the system inputs can be written as functions of flat outputs $\boldsymbol{z}$ and its derivatives, the designed trajectory is guaranteed to be dynamically feasible, provided that the boundary conditions and actuator limits are satisfied. The actuator limits can be formulated as inequality constraints and (Faiz, Agrawal, & Murray, 2001) attempted to solve trajectory planning of flat systems subject to inequality constraints, by approximating the feasibility set with linear inequality constraints. However because of the non-convex and nonlinear constraints, the approach requires high computational effort.

A different approach is described in (Richter et al., 2013). The optimal ratio of time allocation for each segment is firstly determined, followed by assigning the total trajectory time. The control inputs are computed algebraically to verify if the actuator constraints are violated. The process is iterated until the minimal cost function is found while the actuator constraints are satisfied.

B-2-1 Differential flatness of quadrotor

The following derivation follows from (Mellinger & Kumar, 2011). To prove that the quadrotor model is differentially flat, we need to show that the state variables and control inputs are functions of the flat outputs in equation (B-18) and their derivatives.

$$\boldsymbol{z} = \begin{bmatrix} x_E & y_E & z_E & \psi \end{bmatrix}^T \tag{B-18}$$

In case the yaw angle ψ is defined in the direction of the velocity, i.e. $\psi = atan(y_E/x_E)$, ψ is already a function of the flat outputs. The velocity in body frame $\mathcal{F}_B$ is related to the flat outputs with equation (B-19).

$$\begin{bmatrix} \dot{x_E} \\ \dot{y_E} \\ \dot{z_E} \end{bmatrix} = R_B^E \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{B-19}$$

To show that R_B^E is a function of flat outputs and their derivatives, we rewrite R_B^E in terms of the unit vectors $\mathbf{x}_B, \mathbf{y}_B$ and $\mathbf{z}_B$ that define the body frame,

$$R_B^E = \begin{bmatrix} \mathbf{x}_E \\ \mathbf{y}_E \\ \mathbf{z}_E \end{bmatrix} = \begin{bmatrix} \mathbf{x}_B & \mathbf{y}_B & \mathbf{z}_B \end{bmatrix} \tag{B-20}$$

Newton's equation (A-9) can be formulated in $\mathcal{F}_E$,

$$m\dot{\boldsymbol{V}}_{\boldsymbol{E}} = mg\mathbf{z}_E + (-T)\mathbf{z}_B \tag{B-21}$$

The unit vector $\mathbf{z}_B$ is then related to the second derivative of flat outputs.

$$\mathbf{z}_B = \frac{\begin{bmatrix} \ddot{z}_1 & \ddot{z}_2 & \ddot{z}_3 - g \end{bmatrix}^T}{|| \begin{bmatrix} \ddot{z}_1 & \ddot{z}_2 & \ddot{z}_3 - g \end{bmatrix}^T ||} \tag{B-22}$$

The rotation from $\mathcal{F}_B$ to $\mathcal{F}_E$ can be decomposed into two steps, i.e. $R_B^E = R_C^E \cdot R_B^C$. Pitch and roll are represented by R_B^C, while yaw is described by R_C^E. The unit vector $\mathbf{x}_C$ of $\mathcal{F}_C$ can be defined in equation (B-23). $\mathbf{y}_B$ and $\mathbf{x}_B$ follow equation (B-24) and (B-25).

$$\mathbf{x}_C = \begin{bmatrix} cos\psi & sin\psi & 0 \end{bmatrix}^T \tag{B-23}$$

$$\mathbf{y}_B = \frac{\mathbf{z}_B \times \mathbf{x}_C}{||\mathbf{z}_B \times \mathbf{x}_C||} \tag{B-24}$$

$$\mathbf{x}_B = \mathbf{y}_B \times \mathbf{z}_B \tag{B-25}$$

Here it can be concluded that the unit vectors $\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B$ are functions of flat outputs and their derivatives, so is the rotational matrix R_B^E. To further investigate the relationship between acceleration and flat outputs, the derivative of equation (B-21) is computed,

$$m\dot{\boldsymbol{a}} = -\dot{T}\ \mathbf{z}_B - \omega_B^E \times T\ \mathbf{z}_B \tag{B-26}$$

Projecting this equation on $\mathbf{z}_B$, we take the dot product of equation (B-26) with $\mathbf{z}_B$,

$$m\dot{\boldsymbol{a}} \cdot \mathbf{z}_B = -\dot{T}\ \mathbf{z}_B \cdot \mathbf{z}_B - \omega_B^E \times T\ \mathbf{z}_B \cdot \mathbf{z}_B \tag{B-27}$$

$$= -\dot{T} \tag{B-28}$$

Notice that equation (B-28) is true because the rotational rate of unit vector $\mathbf{z}_B$ is always orthogonal to $\mathbf{z}_B$. Substitute equation (B-28) into equation (B-26), we can determine the projection of $\frac{m}{T}\dot{\boldsymbol{a}}$ onto the $\mathbf{x}_B - \mathbf{y}_B$ plane, denoted with $\boldsymbol{j}_{XY}$,

$$\boldsymbol{j}_{XY} := \omega_B^E \times \mathbf{z}_B \tag{B-29}$$

$$= \frac{m}{T}\dot{\boldsymbol{a}} - \frac{m}{T}(\dot{\boldsymbol{a}} \cdot \mathbf{z}_B)\mathbf{z}_B \tag{B-30}$$

$$= \frac{m}{T}(\dot{\boldsymbol{a}} - (\dot{\boldsymbol{a}} \cdot \mathbf{z}_B)\mathbf{z}_B) \tag{B-31}$$

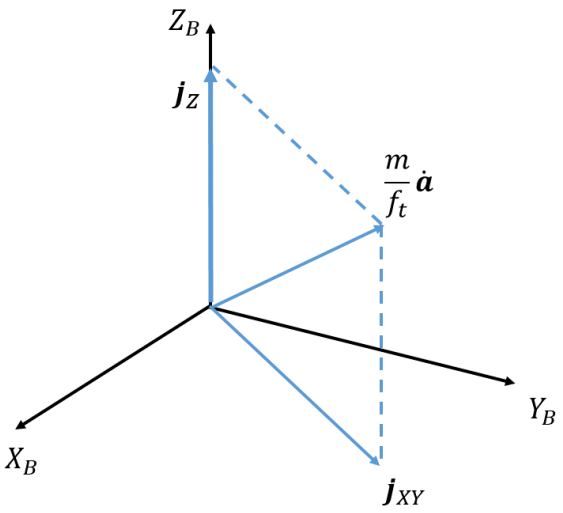


Figure B-3: Illustration of j_{XY} and j_Z

Vector $\boldsymbol{j}_{XY}$, $\boldsymbol{j}_Z$, $\frac{m}{T}\dot{\boldsymbol{a}}$ and their relationship with unit vectors of $\mathcal{F}_B$ are illustrated with figure B-3.

The angular velocities in $\mathcal{F}_E$ can be expressed with equation (B-32).

$$\omega_B^E = p\mathbf{x}_B + q\mathbf{y}_B + r\mathbf{z}_B \tag{B-32}$$

We can relate the angular velocities to $\boldsymbol{j}_{XY}$ by taking the cross product of equation (B-32) and $\mathbf{z_B}$.

$$\begin{aligned}\boldsymbol{j}_{XY} &= \omega_B^E \times \mathbf{z}_B && \text{(B-33)}\\ &= p\mathbf{x}_B \times \mathbf{z}_B + q\mathbf{y}_B \times \mathbf{z}_B + r\mathbf{z}_B \times \mathbf{z}_B && \text{(B-34)}\\ &= -p\mathbf{y}_B + q\mathbf{x}_B && \text{(B-35)}\end{aligned}$$

By projecting equation (B-35) onto $\mathbf{y}_B$ and $\mathbf{x}_B$, roll and pitch rate can be written as functions of the derivatives of flat outputs.

$$p = -\boldsymbol{j}_{XY} \cdot \mathbf{y}_B, \quad q = \boldsymbol{j}_{XY} \cdot \mathbf{x}_B \tag{B-36}$$

Lastly, from equation (B-21), the thrust force T is directly related to the flat outputs and their derivatives,

$$T = -m|| \begin{bmatrix} \ddot{z}_1 & \ddot{z}_2 & \ddot{z}_3 - g \end{bmatrix}^T || \tag{B-37}$$

It can therefore be concluded that the state variables and control inputs are functions of the flat outputs (B-18) and their derivatives. This proves that the quadrotor system is indeed differentially flat, and the trajectory can be planned in the flat space instead of state space.

B-3 Trajectory library

In this section, a summary of exiting literature on navigation with trajectory library and/or motion primitives is provided. As discussed in section B-1, finding the optimal trajectory in the high-dimensional planning space is a complex problem. One solution is to reduce the solution space to a finite set of feasible trajectories or motion primitives. A trajectory or maneuver library can be constructed with these primitives. The optimal trajectory between two points can be then quickly generated by interconnection of the primitives. (Frazzoli, Feron, & Dahleh, 2000) showed that the concatenated trajectory is feasible if the individual element in the library is feasible.

An attempt has been made by (Dever et al., 2006) to reduce nonlinear trajectory planning dimensionality with parameterized maneuver classes defined with a family of trajectories and family-specified parameters describing the boundary conditions. These trajectories can be obtained with human pilot maneuvers or motion capture. This allows the user to select the desired vehicle behavior, for example, if an agile or a smooth maneuver is required. A interpolation algorithm is used to ensure continuity between maneuver classes. With this algorithm, online trajectory generation with stored motion samples can be achieved significantly faster than the common optimal control approach.

In (Paranjape, Meier, Shi, Chung, & Hutchinson, 2013), motion primitive library is used to enable fast flight through regions with dense obstacles. The research focuses on the design of two families of motion primitives: steady 3D turns and aggressive turn-around maneuvers. The motion primitives consist of sequential control inputs(e.g. control surface settings) that are required to perform the aforementioned maneuvers. Since the change of state cannot be completed instantaneously, a time delay is introduced before the switch of two successive primitives.

Trajectory library can also be used to account for model uncertainties and disturbances. More recently in (Majumdar & Tedrake, 2016), open-loop trajectories are computed in the offline stage, and a “funnel” is created around each trajectory to represent the reachable set. With this method, the uncertainties are explicitly taken into account during the trajectory selection.

Appendix C

Literature Summary and Proposed Method

From the literature covered in previous sections, the following conclusions can be drawn.

- The mathematical model of Bebop dynamic system is elaborated. The quadrotor system is nonlinear and under-actuated.
- An INDI controller is chosen for its capability to cope with nonlinearities. The outer attitude loop will be controlled by a simple PD controller while the inner INDI controller stabilizes the angular accelerations.
- Classic optimal control based approach tackles the optimization problem subjected to system dynamic constraints in the form of differential equations. Various methods have been discussed, as well as their advantages and disadvantages. In brief, though these methods are more general compared to differential flatness based approach, they either lead to large sparse NLP(e.g. direct collocation) or require expensive computational effort(e.g. GA).
- Differential flatness based approach, on the other hand, is only restricted to differentially flat systems. However this property transforms the dynamic constraints from differential equations to algebraic equations. Dynamic feasibility is also ensured, provided that the actuator constraints are satisfied. Thus, by parameterization of the flat outputs, cost function can be formulated and optimized.
- In order to ease the computation efforts involved in trajectory optimization, attempts have been made to use pre-computed trajectory libraries or motion primitives. It is shown that dynamic feasibility can be guaranteed if individual elements in the library is dynamically feasible. Although the exact same approach cannot be adapted in the content of this thesis, the idea of using a trajectory library comprises of sequential control inputs required to perform a certain maneuver is inspirational.

For this thesis, a combination of the aforementioned approaches will be chosen. Figure C-1 illustrates the idea of trajectory library used in this thesis. Two stages can be identified and the approaches consists of the following components.

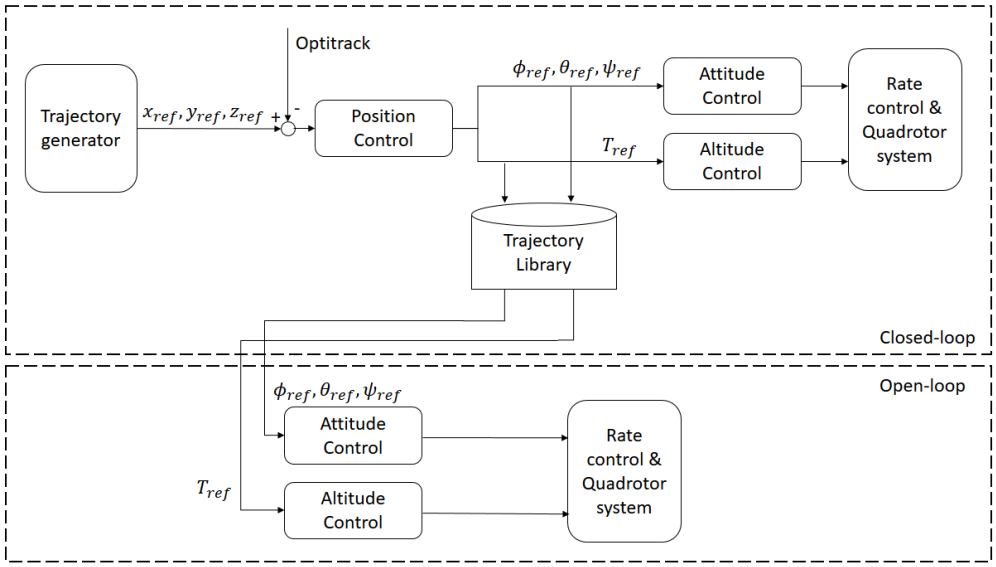


Figure C-1: Block diagram illustrating the proposed guidance method

- Trajectory generator: The trajectory generator outputs the position reference $x_{ref}, y_{ref}, z_{ref}$ between two points based on minimum snap.
- Position controller: A feedback PID controller that computes the reference attitude $\phi_{ref}, \theta_{ref}, \psi_{ref}$ and thrust force T_{ref}. The control loop is closed with position feedback provided by the external motion capture system, Optitrack.
- Trajectory library: An on-board trajectory library that stores the reference attitude $\phi_{ref}, \theta_{ref}, \psi_{ref}$ and thrust force T_{ref} from the closed-loop flight.
- Attitude controller, altitude controller, rate controller and quadrotor system: These are the inner loops of the control system. Together they calculate the required rotor speeds Ω_i to track the reference attitude and thrust. This part of the control loop is identical in closed-loop and open-loop flight.

Feasibility of the approach will be addressed in this thesis with the difference between the closed-loop and open-loop flight, as well as the variance of the open-loop flights.

Appendix D

Preliminary Results

D-1 Trajectory library

The quadrotor is required to fly through a gate with dimension of one by one meter, without information of the quadrotor's global position. The only knowledge of position comes from visual detection of the relative distance with respect to the gate. Moreover, the velocity of the drone is estimated by the down-facing camera, with optical flow based algorithm. The maximum distances from the gate that can be detected by the front camera define a three-dimensional"visible zone". To construct the trajectory library, we create a grid area in front of the gate. The global position of the quadrotor can be approximated to the nearest grid point. Figure D-1 depicts the aforementioned setup. Between each grid point and the center of the gate, a trajectory is generated. In the experimental stage, the trajectories will be uploaded to Bebop and closed-loop flights will be performed with position feedback from Optitrack. The closed-loop flight control inputs are logged and stored for each grid point in the library.

D-2 Minimum snap trajectory optimization

In this section, we derive the method for generating minimum-snap trajectory. The minimum-snap trajectory is originally proposed in (Mellinger & Kumar, 2011). It is found that moments around x and y axis appear to be functions of the snap, and therefore by minimizing the snap, the control effort is minimized. Since the quadrotor system is differentially flat and the flat outputs are selected to be $\boldsymbol{z} = [x_E, y_E, z_E]^T$, we can represent the trajectory, described by the flat outputs, between any grid point and the center of the gate with polynomial of order n, see equation D-1.

$$P(\boldsymbol{p}, t) = p_0 + p_1 t + \cdots + p_{n-1} t^{n-1} + p_n t^n = \sum_{n=0}^{N} p_n t^n \tag{D-1}$$

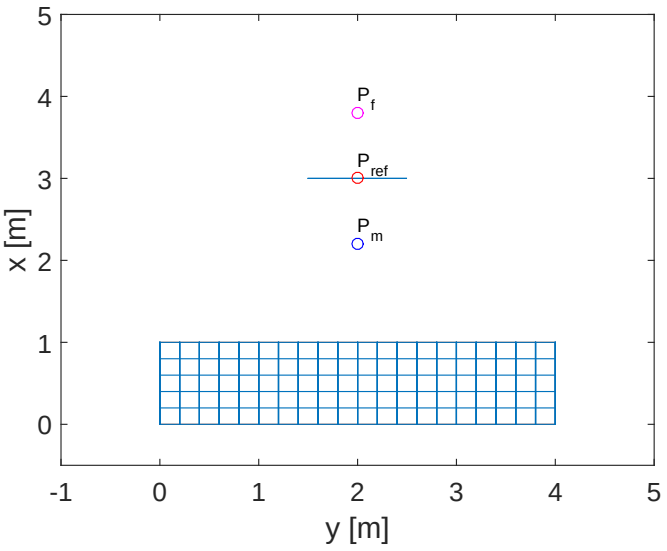


Figure D-1: Illustration of the trajectory library. P_{ref} is the reference point to which the distance is measured by the front camera. Between each grid point and P_m, a trajectory will be generated.

The optimization objective is the total snap along a trajectory $P(t)$, which defines the cost function J.

$$J(\boldsymbol{p}, t) = \int_0^T (\frac{d^4 \boldsymbol{z}}{dt^4})^2 dt \tag{D-2}$$

Firstly we need to take the derivatives of $P(t)$. The r^{th} derivative of $P(\boldsymbol{p}, t)$ can be determined with the following equations.

$$P^{(1)}(\boldsymbol{p}, t) = p_1 + 2p_2 t + \cdots + np_n t^{n-1} \tag{D-3}$$
$$P^{(2)}(\boldsymbol{p}, t) = 2p_2 + 3 \times 2p_3 t \cdots + n(n-1)t^{t-2} \tag{D-4}$$
$$\vdots \tag{D-5}$$
$$P^{(r)}(\boldsymbol{p}, t) = \sum_{n=r}^{N} \left(\prod_{m=0}^{r-1} (n-m) \right) p_n t^{n-r} \tag{D-6}$$

The coefficient C_n of the n^{th} term of the square of any polynomial, can be found with equation (D-7).

$$C_n = \sum_{j=0}^{N} p_j p_{n-j} \tag{D-7}$$

Substitute the square of $P^{(r)}(\boldsymbol{p},t)$ into the cost function,

$$J = \int_0^T \left(P^{(r)}(\boldsymbol{p},t)\right)^2 dt \tag{D-8}$$

$$= \int_0^T \sum_{j=0}^{2N} \sum_{n=r}^{N} \left(\prod_{m=0}^{r-1} (n-m) p_j p_{n-j} t^{n-2r} \right) dt \tag{D-9}$$

$$= \sum_{n=2r}^{2N} \left[\sum_{j=r}^{n-r} \left(\prod_{m=0}^{r-1} (j-m)(n-j-m) \right) p_j p_{n-j} \right] \frac{T^{n-2r+1}}{n-2r+1} \tag{D-10}$$

The Hessian matrix Q is defined by:

$$Q = \begin{bmatrix} \frac{\partial^2 J}{\partial p_1^2} & \frac{\partial^2 J}{\partial p_1 \partial p_2} & \cdots & \frac{\partial^2 J}{\partial p_1 \partial p_n} \\ \frac{\partial^2 J}{\partial p_2 \partial p_1} & \frac{\partial^2 J}{\partial p_2^2} & \cdots & \frac{\partial^2 J}{\partial p_2 \partial p_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial p_n \partial p_1} & \frac{\partial^2 J}{\partial p_n \partial p_2} & \cdots & \frac{\partial^2 J}{\partial p_n^2} \end{bmatrix} \tag{D-11}$$

To construct the Hessian matrix, the cost function J is differentiated with respect to p_i.

$$\frac{\partial J}{\partial p_i} = 2 \sum_{n=2r}^{2N} \prod_{m=0}^{r-1} (i-m)(n-i-m) p_{n-i} \frac{T^{n-2r+1}}{n-2r+1} \tag{D-12}$$

Then equation (D-12) is differentiated again with respect to each of the polynomial coefficients p_l. Note that now $n = i + l$.

$$\frac{\partial^2 J}{\partial p_i \partial p_l} = 2 \prod_{m=0}^{r-1} (i-m)(l-m) \frac{T^{i+l-2r+1}}{i+l-2r+1} \tag{D-13}$$

The next step is to define the boundary conditions. The constraints at start and end point of the trajectory need to be specified. They can be imposed either to satisfy constraints on states, or to ensure continuity of the derivatives. Equation (D-14) formulates the linear equality constraints.

$$\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b} \tag{D-14}$$

$$\boldsymbol{A} = \begin{bmatrix} A_0 \\ A_f \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} b_0 \\ b_f \end{bmatrix} \tag{D-15}$$

The quadratic program can be formulated as follows,

$$\min \quad J = \boldsymbol{p}^T Q \boldsymbol{p} \tag{D-16}$$

$$s.t. \quad \boldsymbol{A}\boldsymbol{p} = \boldsymbol{b} \tag{D-17}$$

Problem (D-16) can be solved with standard MATLAB function `quadprog.m`. The function returns the optimal set of polynomial coefficients $\boldsymbol{p}$ that minimizes the cost function while satisfying the boundary constraints. In figure D-2, the generated trajectory from one grid point in shown.

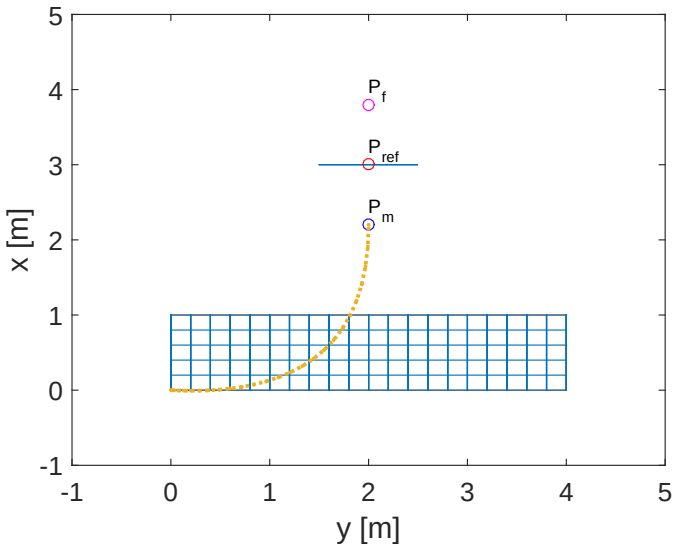


Figure D-2: An example of minimum-snap trajectory

D-3 Effect of polynomial order

Up till now the polynomial representing the trajectory has a randomly chosen order of 7. Here we would like to determine if the order of polynomial plays an important role in the feasibility of the approach. The minimum order is 4 since the cost function is related to the 4^{th} derivative of the polynomial. Figure D-3 depicts the reference trajectory with polynomial order 4, 7 and 10. In all three directions, the differences between the responses from three polynomial orders are negligibly small. It can be concluded that the order of the polynomial does not play an important role.

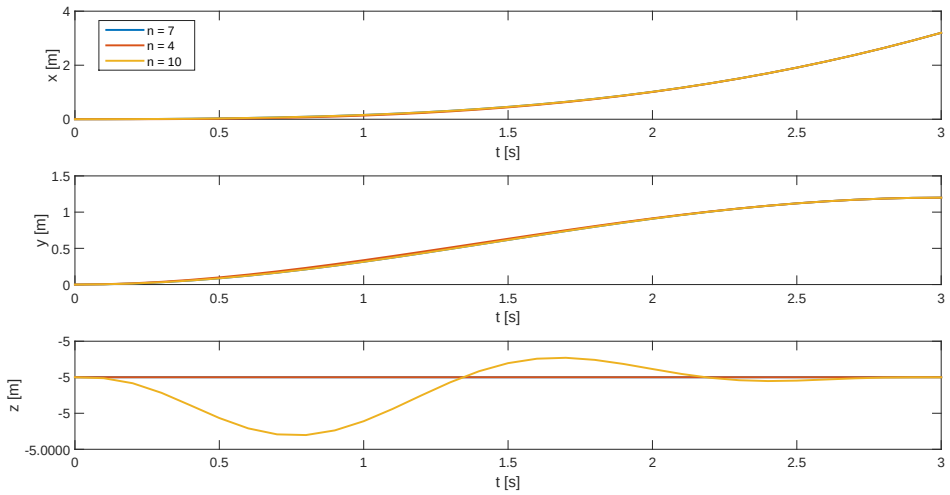


Figure D-3: Time response of the position(simulation time = 3.0s), with different polynomial order

D-4 Time allocation

Time allocation for each trajectory is also important. This can be seen in figure D-4. Two trajectories are generated with equal time allocation of 3 seconds. Since the distances between the starting point and end point are different, it is unreasonable to assign the same time to these trajectories. One method to find the optimal time allocation is to use the maximum pitch angle of 20 degrees to determine the minimum time t_{min} for a straight flight and scale the time allocation for each trajectory with the distance to the end point. From simulation, it is found that $t_{min} = 1.6s$.

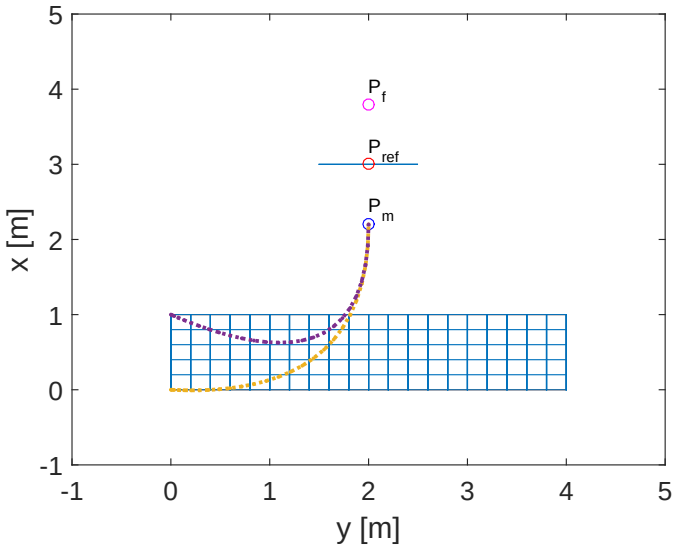


Figure D-4: Two trajectories generated with equal time allocation of 3 seconds

D-5 Simulation result with PID controller

Simulations are performed with PID controller. The simulation result is shown in figure D-5. It can be seen that the quadrotor is capable of following the trajectory. However, the drone does not reach the destination within the simulation time. Figure D-6 shows the time response of 3D position x, y and z. Discrepancies between the reference value and response increase with respect to time. This is due to the inherent disadvantage of using only feedback controller in tracking a moving target. A time lag will always be present since the feedback controller only reacts to current error. The non-zero steady state error can be proved with Final Value Theorem(FVT). From FVT, the steady-state error $e(t = \infty)$ is defined as follows,

$$e(t = \infty) = \lim_{s \to 0} E(s) = \lim_{s \to 0} \frac{sU(s)}{1 + G(s)} \tag{D-18}$$

with $G(s) = N(s)/D(s)$ the transfer function on the forward path of a unity feedback system. The designed trajectory, is related to the 7^{th} order of time. The Laplace transform of such

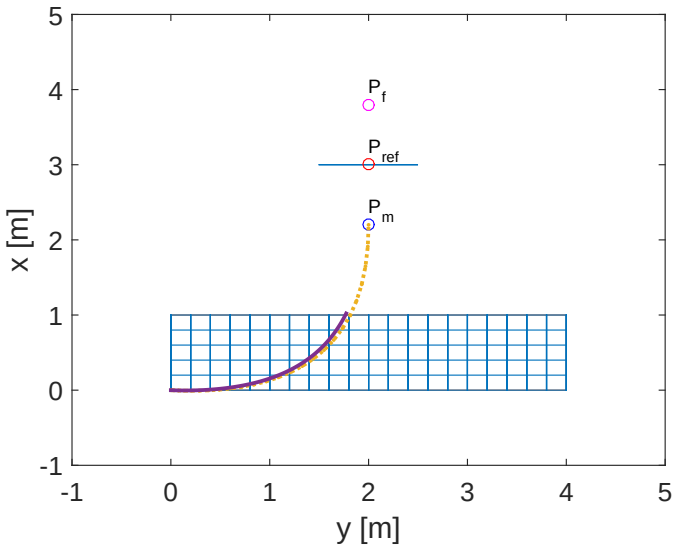


Figure D-5: Simulation result of one designed trajectory. The yellow dotted line represents the minimum-snap trajectory, and the purple solid line shows the simulation result.(simulation time = 3.0 s)

polynomial can be written as,

$$U(s) = \mathcal{L}\{p_0 + p_1 t + \cdots + p_7 t^7\} = p_0 \frac{1}{s} + p_1 \frac{1!}{s^2} + \cdots + p_7 \frac{7!}{s^8} \tag{D-19}$$

Collecting all the coefficients,

$$U(s) = c_0 \frac{1}{s} + c_1 \frac{1}{s^2} + \cdots + c_7 \frac{1}{s^8} = \frac{c_0 s^7 + c_1 s^6 + \cdots + c_7}{s^8} \tag{D-20}$$

Substituting equation D-20 into D-18,

$$e(t = \infty) = \lim_{s \to 0} \frac{c_0 s^7 + c_1 s^6 + \cdots + c_7}{s^7} \cdot \frac{1}{1 + \frac{N(s)}{D(s)}} \tag{D-21}$$

$$= \lim_{s \to 0} \frac{c_0 s^7 + c_1 s^6 + \cdots + c_7}{s^7} \cdot \frac{D(s)}{N(S) + D(s)} \tag{D-22}$$

In order to have zero steady state error, the closed-loop system is required to have at least 8 integrators. This is difficult to achieve with feedback controllers. The worst case time lag, which occurs when the start point is the farthest away from the gate, is about $0.5s$. This can be seen in figure D-7 and D-8, where the simulation runs 0.5 seconds longer than the designed time. At $t = 3.5s$, the quadrotor reaches destination P_m and the time responses of x and y also reach the reference value.

The purpose of this thesis is to verify the feasibility of using a off-line trajectory library to guide the quadrotor through the gate when position information is unavailable. The time lag has no significant impact on the feasibility of the proposed guidance approach. Multiple solutions are available and can be implemented if time permits.

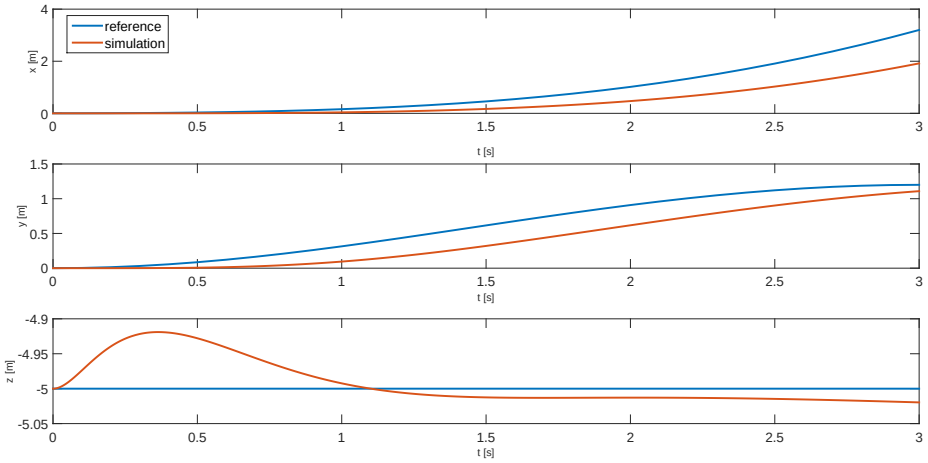


Figure D-6: Time response of the position(simulation time = 3.0s)

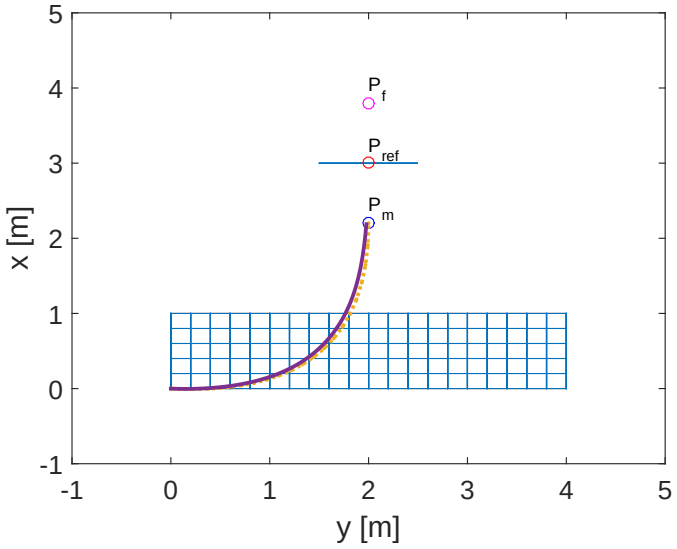


Figure D-7: Simulation result of one designed trajectory (simulation time = 3.5s)

- Replace the feedback only controller with a combination of feedforward and feedback controller. The feedforward controller calculates the required control input from the designed trajectory, while the feedback controller compensates for the model uncertainties and unmeasured disturbances. With the feedforward controller, rapid response can be provided. One example of such controller is developed in (Lee, 2011) and implemented on quadrotors in (Mellinger & Kumar, 2011). The principle of the controller is elaborated as follows.

  Let the reference 3D coordinates of the quadrotor be denoted with $\boldsymbol{r}_d = \begin{bmatrix} x_d & y_d & z_d \end{bmatrix}^T$, and the reference attitude R_d and angular velocity ω_d can be com-

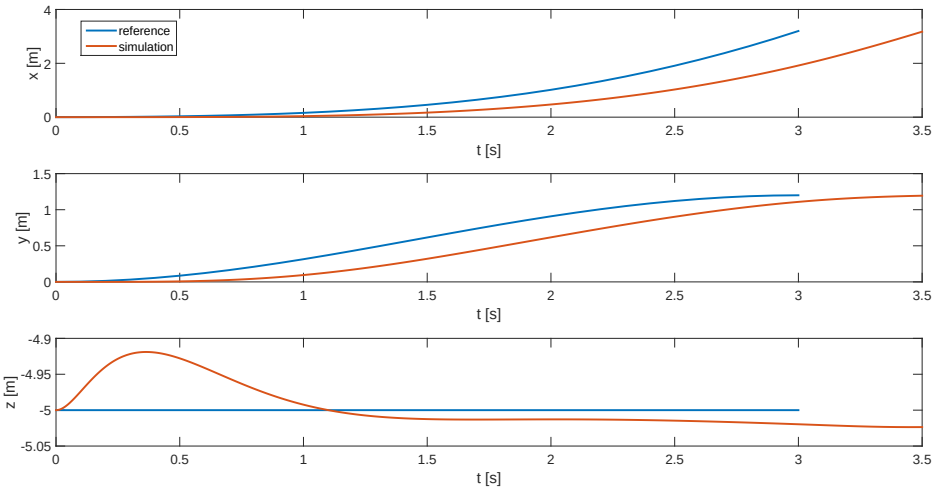


Figure D-8: Time response of the position(simulation time = 3.5s)

puted from the trajectory.

$$f = (-k_x e_x - k_v e_v + mg\mathbf{z_E} + m\ddot{r}_d)R\mathbf{z_E} \tag{D-23}$$

$$M = -k_R e_R - k_\omega e_\omega + \omega \times I\omega - I(\hat{\omega}R^T R_d \omega_d - R^T R_d \dot{\omega}_d) \tag{D-24}$$

Where, e_x, e_v, e_R and e_ω are the errors in position, velocity, attitude and angular velocity respectively, and the k's are the corresponding control gains. The hat operator $\hat{\cdot}$ maps a 3D vector onto $\mathbb{R}^3$, see (Lee, 2011) and (Jan Tommy Gravdahl & Johan, 2013) for a detailed explanation.

- An easier way to solve the problem is to explicitly take the time lag into account when running the simulation. The trajectory generation algorithm outputs the time series of x, y and z coordinates, which are then used as reference values. Since the initial position of the quadrotor coincide with the designed trajectory, the initial error in position is zero and builds with increasing speed. In order to overcome the time lag, we can give a non-zero initial error, which makes the controller react immediately. This can be done by inputting the further reference positions into the system. In figure D-9, the reference coordinates at $t = 0.6s$ is given to the system at $t = 0$ and the simulation is run for 3 seconds as designed. The non-zero initial error can be more clearly seen the $y(t)$ plot in figure D-10. The reference y coordinates starts at a non-zero value compared to figure D-5. At $t = 3s$, the quadrotor reaches P_m.

D-6 Effect of (model) uncertainty

- Drag
  The effect of uncertainty in drag is examined here. The trajectory is generated with the modeled drag. In figure D-11, this uncertainty is taken into account. It can be

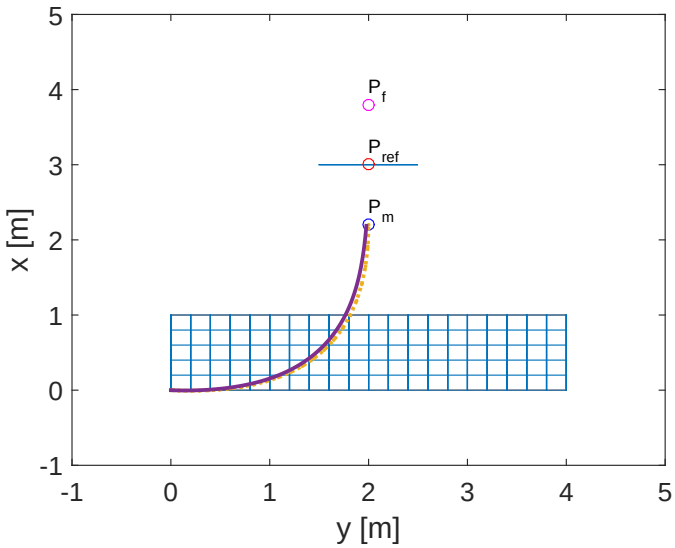


Figure D-9: Simulation result of one designed trajectory (simulation time = 3.0s), with 5 time steps ahead

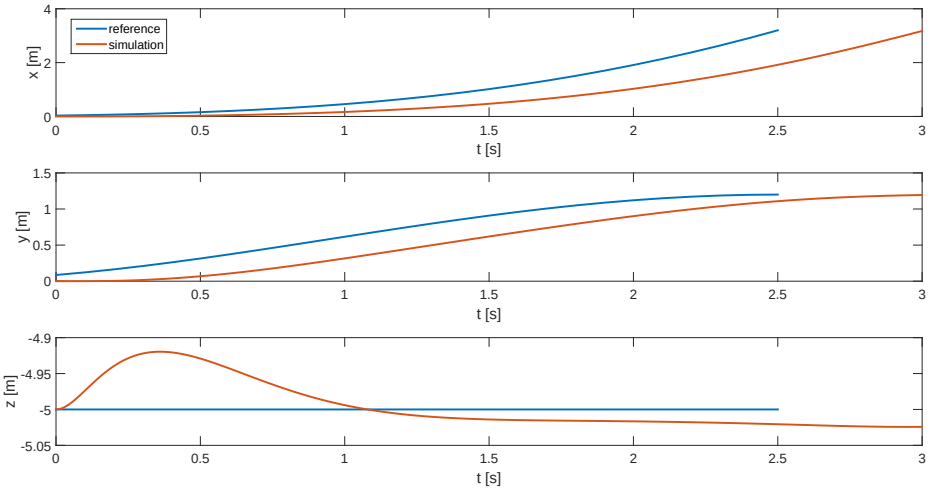
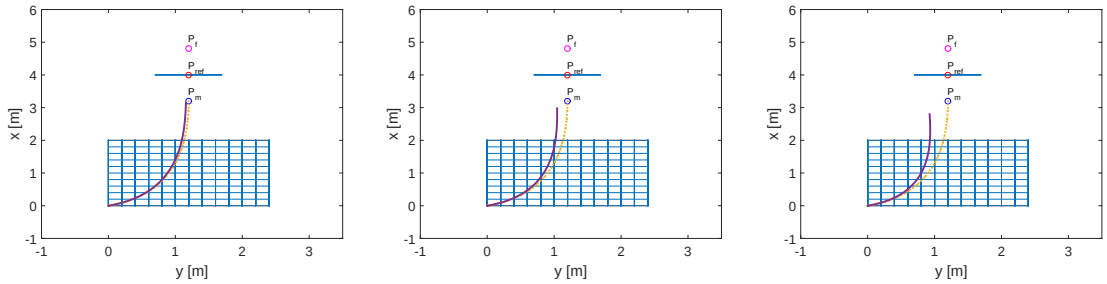


Figure D-10: Time response of the position(simulation time = 3.0s), with 5 time steps ahead

concluded that even with 100% error in the drag estimation, the quadrotor is still capable of performing the task.

- Initial position
  It can be expected that the simulated flight trajectory will be parallel to the designed one, but have an offset of the initial position error. If the initial position error is smaller than half of the gate width, the quadrotor will still be able to fly through the gate.

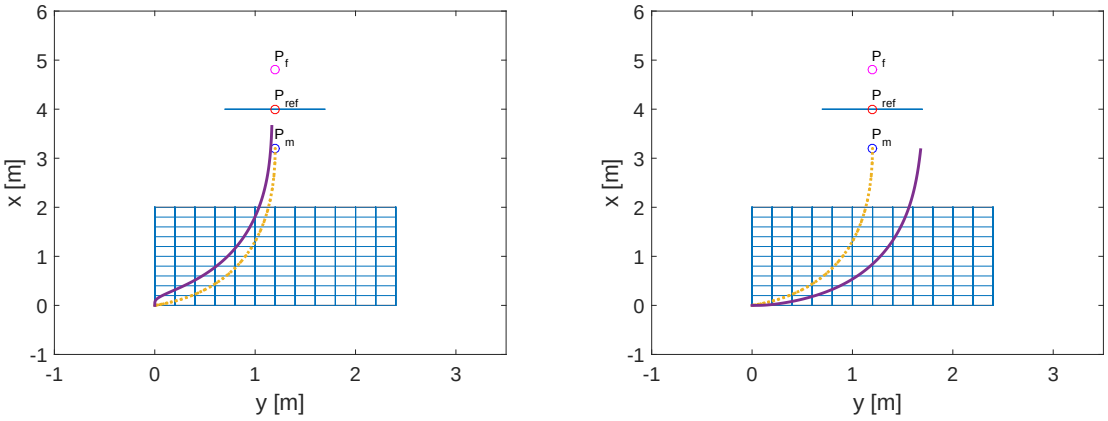
- Initial velocity
  Figure D-12 shows the effect of initial velocity. Here only the initial velocity error in x



(a) 10% error in drag **(b)** 50% error in drag **(c)** 100% error in drag

Figure D-11: Effect of uncertainty in drag

and y directions are considered. The trajectory is generated for one grid point assuming zero initial velocity in all directions. It can be seen that the trajectory is more sensitive to initial velocity error, especially in y direction. With an error in v_0 of 0.2 m/s, the quadrotor cannot pass the gate (Figure D-12b). Although 0.2 m/s seems to be a relatively large error, it has to verified with the performance of optic flow algorithm. The maximum allowable initial velocity error still has to be tested for different grid points.



(a) $V_0 = \begin{bmatrix} 0.2 & 0 & 0 \end{bmatrix}^T$ **(b)** $V_0 = \begin{bmatrix} 0 & 0.2 & 0 \end{bmatrix}^T$

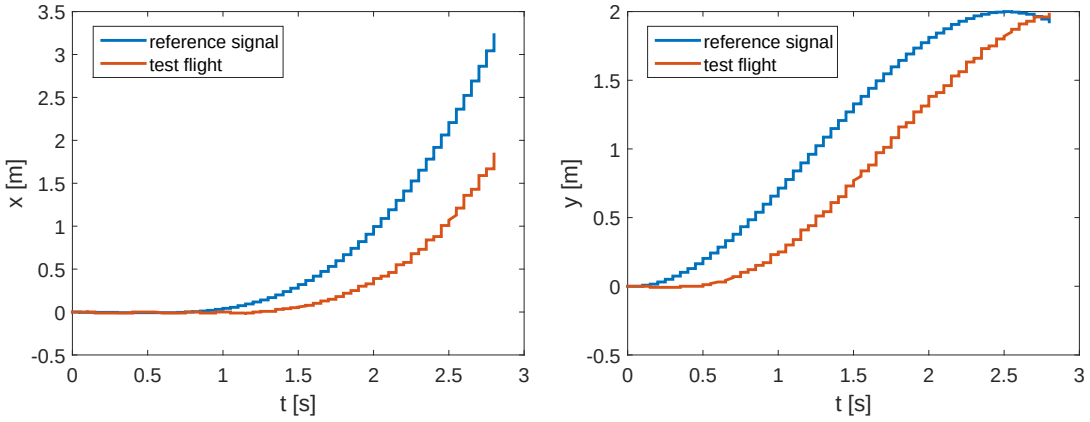
Figure D-12: Effect of initial velocity on trajectory generation

D-7 Experimental flight results

MATLAB simulations in the previous sections have shown the feasibility of the approach. Here, the generated trajectory will be uploaded to the Bebop drone. The on-board controls are programmed with Paparazzi [1]. A PID feedback controller tracks the position, and the inner loops consist of INDI controllers. The previously generated trajectory is tested here.

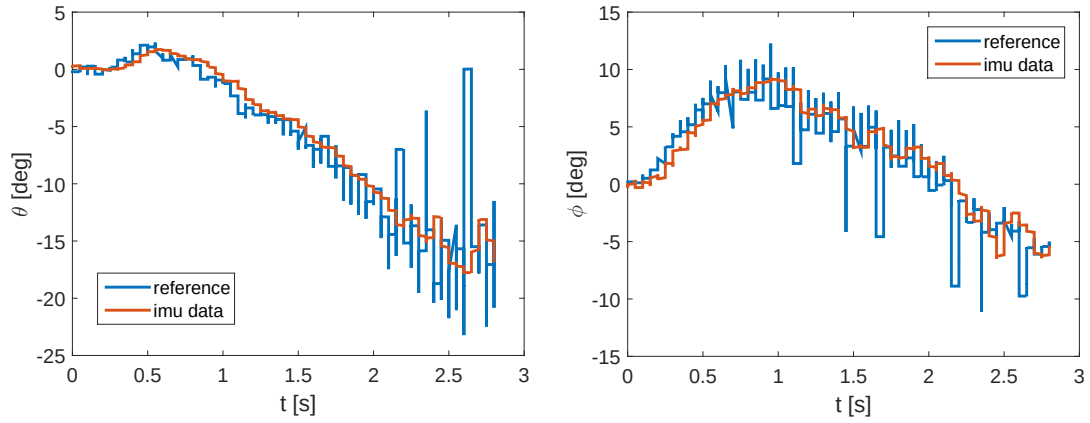
[1] Paparazzi UAV. `http://wiki.paparazziuav.org`

The responses in x, y directions and the calculated Euler angles can be seen in figure E-2 and D-14.



(a) Reference signal x and response **(b)** Reference signal y and response

Figure D-13: Quadrotor response to input in x and y



(a) Calculated inner loop reference angle θ and response **(b)** Calculated inner loop reference signal ϕ and response

Figure D-14: Calculated inner loop reference angles and quadrotor responses

Similar to the simulation results, the closed-loop flight lags behind the designed trajectory in time. The reason for this has already been elaborated in the previous section. Due to limitations in time and facilities, the current feedback controllers will continue to be used. It is recommended to investigate in the design of a feed-forward controller combined with feedback in the future.

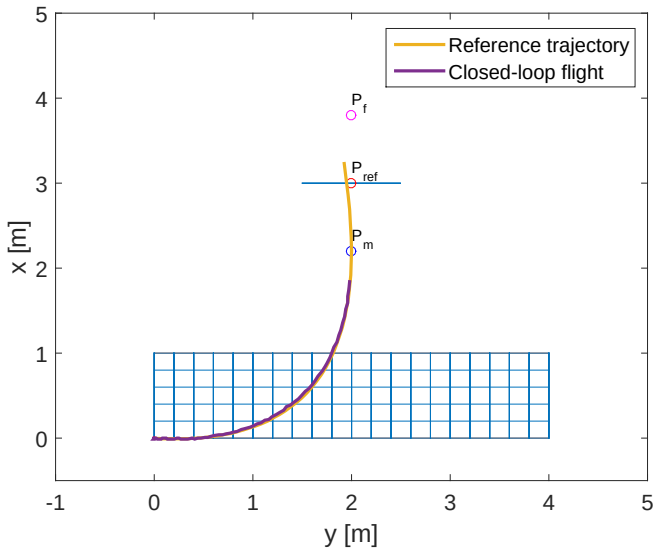


Figure D-15: Comparison between the designed trajectory and closed-loop flight

Appendix E

Additional Results

In this chapter, additional results obtained in the experimental phase are provided. They are not included in the scientific paper, but can be useful for future studies.

E-1 Averaged control inputs from closed-loop flights

In order to determine how many closed-loop flights are required to produce a reliable mean, we perform 20 closed-loop flights and study the average control input. Figure E-1 shows the averaged roll angle of 10, 15 and 20 Closed-loop (CL) flights. In the first 1.5 seconds,

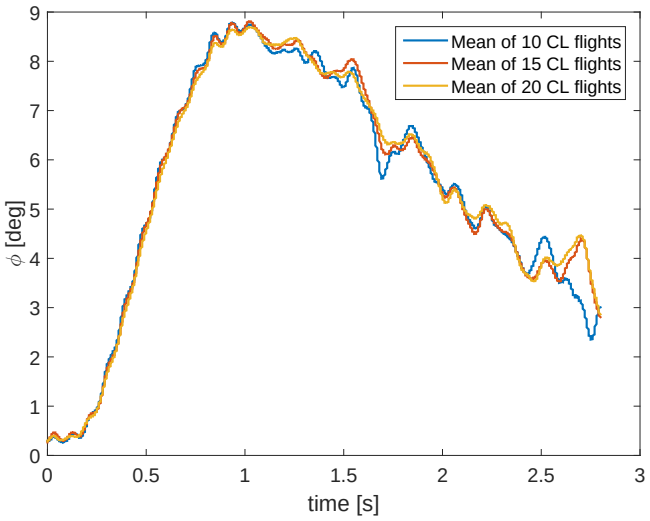


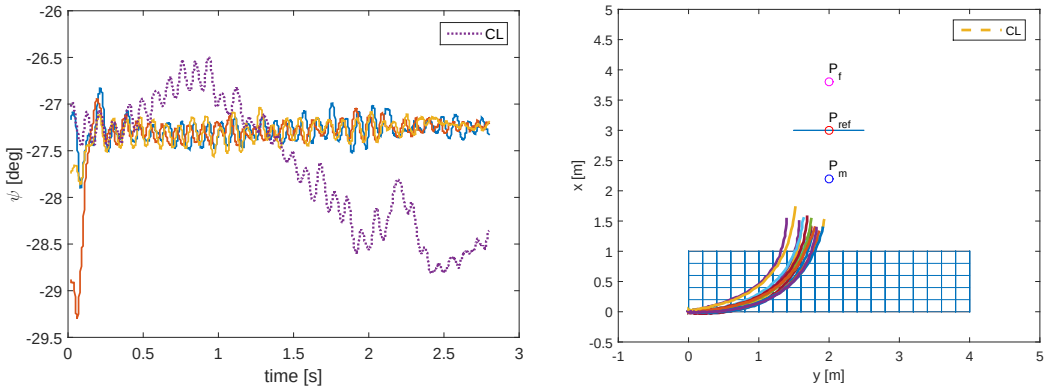
Figure E-1: Averaged roll angle ϕ of 10, 15 and 20 CL flights

three lines overlap and differences start to appear from $t = 1.5s$. Larger discrepancies can be identified between the blue and red line. This means that outliers are compensated by taking

more data sets into account. The mean is not much improved by averaging more than 15 CL flights, since the differences between the red and yellow lines are relatively small.

E-2 Effect of yaw angle

The yaw angle ψ, is not actively controlled in this project. Instead, the yaw angle at the beginning of the flight is maintained by the attitude controller. It has been discovered that in the closed-loop flight, the yaw angle experiences a small change of about 3 degrees. This is depicted by the purple dotted line in figure E-2a. The rest five lines show the yaw angles in (five of the) open-loop flights. Comparison of the corresponding trajectories are shown in figure E-2b.

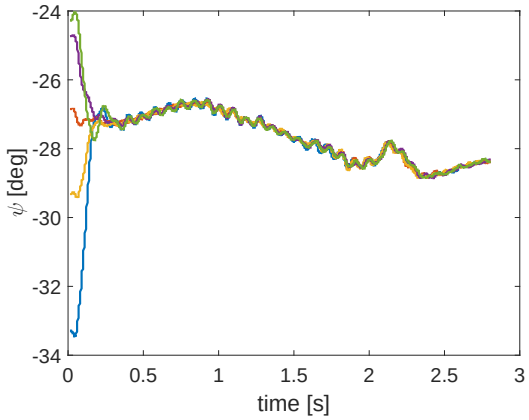


(a) Comparison of yaw angles from closed-loop and open-loop flights.

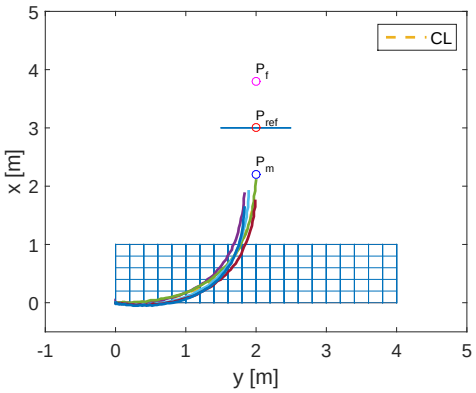
(b) 2D trajectory plot of the open-loop flights with uncontrolled yaw angle.

Figure E-2: Effect of yaw angle on the accuracy of open-loop flight.

The initial correction for yaw angle can be clearly seen from figure E-2a. However the yaw angle did not follow the small change in the reference closed-loop angle. This appears to cause the error in figure E-2b, where all open-loop flights deviate to the left of the reference trajectory. In figure E-3a, the fully controlled yaw angle is shown, and the corresponding trajectories are presented in figure E-3b.



(a) Corrected yaw angle from five open-loop flights.



(b) 2D trajectory plot of the open-loop flights with fully controlled yaw angle.

Figure E-3: Open-loop flights with corrected yaw control.

Chapter 6

Conclusion of the Project

Guidance, navigation and control of UAVs remains a major challenge due to the absence of accurate and stable position information. GPS signals can not reach sufficient accuracy in indoor environment, and inertial navigation accumulates errors with time. Common approaches such as vision-based navigation are restricted by the richness of features in the surrounding environment. Other method utilizes external sensors such as laser scanner to provide position information of the UAV.

In this study, a novel guidance method, aiming to allow agile maneuvers without external position sensors, is proposed and tested. A trajectory library that stores a set of control inputs from trajectories that are dynamically optimal is constructed. Open-loop flights without position sensors are then performed by "replaying" the stored control inputs. Both MATLAB simulations and flight tests are conducted. Results confirm the feasibility of the approach and the standard deviations are analyzed.

One major discoveries of the study is the significant difference in control inputs between closed-loop flights. The exact reason for this is undetermined, and more investigation is required. The solution to this problem is to use a single metric, the total mean squared error, to quantify the performance of the closed-loop flight. The set of control inputs with smallest MSE should be used for open-loop flights. Whether the value of MSE can be used as a performance index remains a question for future research.

The relationships between standard deviation, trajectory duration and velocity are also explored in the project. As expected, time plays a more important role. Although higher velocity leads to larger standard deviations, it reduces the total duration of the trajectory. This in return, helps improving the accuracy of the open-loop flight.

This project, shows the feasibility of an alternative guidance method for UAV in GPS-denied environment. The approach has the advantage of allowing agile and aggressive maneuvers, as well as the wide range of applications. The method can also be extended for obstacle avoidance. Essentially, the approach can be used in any situation where the UAV is required to carry out a trajectory between two fixed points in short intervals. Many recommendations are also provided in the thesis for further exploit the full potential of the approach.

Bibliography

Bellman, R. (1957). *Dynamic programming* (1st ed.). Princeton, NJ, USA: Princeton University Press.

Betts, J. T. (1998). Survey of Numerical Methods for Trajectory Optimization. *Journal of Guidance, Control, and Dynamics*, *21*(2), 193–207.

Cai, G., Chen, B. M., & Lee, T. H. (2011). *Unmanned rotorcraft systems*. Springer.

Chan, B., Guan, H., Jo, J., & Blumenstein, M. (2015). Towards UAV-based bridge inspection systems: a review and an application perspective. *Structural Monitoring and Maintenance*, *2*(3), 283–300.

Colomina, I., & Molina, P. (2014). Unmanned aerial systems for photogrammetry and remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, *92*, 79–97.

Dever, C., Mettler, B., Feron, E., Popovic, J., McConley, M., Mettler, B. B., et al. (2006). Nonlinear Trajectory Generation for Autonomous Vehicles via Parametrized Maneuver Classes. *Journal of Guidance, Control, and Dynamics*, *29*(2), 289–302.

Faiz, N., Agrawal, S. K., & Murray, R. M. (2001). Trajectory Planning of Differentially Flat Systems with Dynamics and Inequalities. *Journal of Guidance, Control, and Dynamics*, *24*(2), 219–227.

Fliess, M., Lévine, J., Martin, P., & Rouchon, P. (1992). Sur les systmes non linaires diffrentiellement plats. *Comptes Rendus de l Acadmie des Sciences - Series I - Mathematics*.

Flint, M., Polycarpou, M., & Fernandez-Gaucherand, E. (2002). Cooperative control for multiple autonomous UAV's searching for targets. *41st IEEE Conference on Decision and Control*(December), 2823–2828.

Frazzoli, E., Feron, E., & Dahleh, M. A. (2000). Robust hybrid control for autonomous vehicle motion planning. *2000. Proceedings of the 39th IEEE Conference on Decision and Control*, *1*(iii), 821—-826 vol.1.

Geiger, B. (2009). Unmanned aerial vehicle trajectory planning with direct methods. (August).

He, R., Prentice, S., & Roy, N. (2008). Planning in information space for a quadrotor helicopter in a GPS-denied environments. *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*(2007), 1814–1820.

Hehn, M., & Andrea, R. D. (2011). Quadrocopter Trajectory Generation and Control. *IFAC World Congress*, *18*(1), 1485–1491.

Hillier, F. (2014). *Introduction to operations research* (10th ed.). McGraw-Hill Education.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence*. Cambridge, MA, USA: MIT Press.

Jan Tommy Gravdahl, K. Y. P., & Johan, P. (2013). *Vehicle-manipulator systems, modeling for simulation, analysis, and control*. Springer.

Kahale, E., Castillo, P., & Bestaoui, Y. (2014). Minimum time reference trajectory generation for an autonomous quadrotor. *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings*, 126–133.

Konak, A., Coit, D. W., & Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering and System Safety*, *91*(9). (Special Issue - Genetic Algorithms and ReliabilitySpecial Issue - Genetic Algorithms and Reliability)

Lee, T. (2011). Geometric tracking control of the attitude dynamics of a rigid body on SO(3). *Proceedings of the 2011 American Control Conference*(1), 1200–1205.

Majumdar, A., & Tedrake, R. (2016). Funnel Libraries for Real-Time Robust Feedback Motion Planning. *arXiv*, 1–45.

Mellinger, D., & Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. *2011 IEEE International Conference on Robotics and Automation*, 2520–2525.

Mellinger, D., Kushleyev, A., & Kumar, V. (2012). Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. *Proceedings - IEEE International Conference on Robotics and Automation*, 477–483.

Milam, M. B. (2003). Real-time optimal trajectory generation for constrained dynamical systems. , *2003*, 161.

Mulder, J., van Staveren, W., van der Vaart, J., de Weerdt, E., de Visser, C., in t Veld, A., et al. (2013). *Lecture notes ae3202 flight dynamics*. Delft University of Technology.

Murray, R. M., Hauser, J., Jadbabaie, A., Milam, M. B., Petit, N., Dunbar, W. B., et al. (2002). Online Control Customization via Optimization-Based Control. , 1-21.

Nex, F., & Remondino, F. (2014). UAV for 3D mapping applications: A review. *Applied Geomatics*, *6*(1), 1–15.

Paranjape, A. A., Meier, K. C., Shi, X., Chung, S. J., & Hutchinson, S. (2013). Motion primitives and 3-D path planning for fast flight through a forest. *IEEE International Conference on Intelligent Robots and Systems*, 2940–2947.

Pellazar, M. (1994). Vehicle Route Planning with Constraints using Genetic Algorithms. *Proceedings of National Aerospace and Electronics Conference (NAECON'94)*(4), 111–118.

Pontryagin, L. S., Boltyanskii, V. G., Gamkrelidze, R. V., & Mishchenko, E. F. (1962). *The mathematical theory of optimal processes*. Interscience Publishers.

Qi, Y. C., & Zhao, Y. J. (2005). Energy-Efficient Trajectories of Unmanned Aerial Vehicles Flying through Thermals. , *18*(2), 84–92.

Rao, A. V. (2009). A survey of numerical methods for optimal control. *Advances in the Astronautical Sciences*, *135*(1), 497–528.

Richards, A., & How, J. (2002). Aircraft trajectory planning with collision avoidance using mixed integer linear programming. *American Control Conference*, *3*(2), 1936–1941 vol.3.

Richter, C., Bry, A., & Roy, N. (2013). Polynomial trajectory planning for quadrotor flight in dense indoor environments. *International Conference on Robotics and Automation*(Isrr), 1–16.

Rouchon, P., Fliess, M., Lévine, J., & Martin, P. (1993). Flatness , motion planning and trailer systems. *Proceedings of IEEE Control and Decision Conference*.

Sieberling, S., Chu, Q. P., & Mulder, J. a. (2010). Robust Flight Control Using Incremental Nonlinear Dynamic Inversion and Angular Acceleration Prediction. *Journal of Guidance, Control, and Dynamics*, *33*(6), 1732–1742.

Smeur, E. J. J., Chu, Q. P., & de Croon, G. C. H. E. (2016). Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Aerial Vehicles. *Journal of Guidance, Control, and Dynamics*, *39*(3), 450–461.

Stryk, O. von, & Bulirsch, R. (1992). Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, *37*(1), 357–373.

Turpin, M., Michael, N., & Kumar, V. (2012). Trajectory design and control for aggressive formation flight with quadrotors. *Autonomous Robots*, *33*(1-2), 143–156.

Zhang, C., & Kovacs, J. M. (2012). The application of small unmanned aerial systems for precision agriculture: A review. *Precision Agriculture*, *13*(6), 693–712.