BACKSTEPPING APPROACH FOR DESIGN OF CASCADED PID CONTROLLER WITH GUARANTEED TRAJECTORY TRACKING PERFORMANCE FOR MICRO-AIR UAV

by

YUSUF KARTAL

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON
December 2019

To my parents, big brother and my lovely wife.

ACKNOWLEDGEMENTS

Starting from my academic advisors Dr. Frank Lewis and Dr. Atilla Dogan, I would like to thank people who had an important role in my academic career. I would like to thank Dr. Nicholas Gans for his interest in my research and invaluable comments. I would like to state my special thanks to Dr. Kamesh Subbarao for being such a great professor in the area of control theory and guidance through each stage of my M.S. degree. I would also like to thank Dr. Dogan for providing me opportunity to have M.S. degree in UTA. I especially would like to express my deepest gratitude to Dr. Lewis for inspiring my interest in the development of innovative technologies and continuous encouragement. Thanks to his immense knowledge, our frequent meetings and fruitful discussions, I had a fun throughout writing this thesis. Without his continuous support, it would have been very difficult for me to maintain my motivation through my M.S. degree.

I would also like to extend my appreciation to Turkish Aerospace for providing me financial support for my research studies. I am extremely grateful to Dr. Ugur Zengin for his unconditional support. I would like to thank Dr. Animesh Chakravarthy for being my committee member. I wish to thank University of Texas at Arlington Research Institute family for being very helpful and kind to me.

I am also very grateful to all teachers who have taught me from the elementary school to college in Turkey. I would like to express special thanks to Ibrahim Tasran for his encouragement to an eleven years old boy.

Finally, I would like to express my deep gratitude to my friends, family and wife. I am extremely fortunate to have them in my life.

December 03, 2019

ABSTRACT

BACKSTEPPING APPROACH FOR DESIGN OF CASCADED PID CONTROLLER WITH GUARANTEED TRAJECTORY TRACKING PERFORMANCE FOR MICRO-AIR UAV

YUSUF KARTAL, M.S.

The University of Texas at Arlington, 2019

Academic Advisors: Dr. Frank Lewis, Dr. Atilla Dogan

Flight controllers for micro-air UAVs are generally designed using Proportional-Integral-Derivative (PID) methods, where the tuning of gains is difficult and time-consuming, and performance is not guaranteed. In this thesis, we develop a rigorous method based on the sliding mode analysis and nonlinear backstepping to design a PID controller with guaranteed performance. This technique provides the structure and gains for the PID controller, such that a robust and fast response of the UAV for trajectory tracking is achieved. First, the second-order sliding variable errors are used in a rigorous nonlinear backstepping design to obtain guaranteed performance for the nonlinear UAV dynamics. Then, using a small angle approximation and rigorous geometric manipulations, this nonlinear design is converted into a PID controller whose structure is naturally determined through the backstepping procedure. PID gains that guarantee robust UAV performance are finally computed from the sliding mode gains and from stabilizing gains for tracking error dynamics. We prove that the desired Euler angles of the inner attitude controller loop are related to the dynamics of the outer backstepping tracker loop by inverse kinematics, which provides a seamless

connection with existing built-in UAV attitude controllers. We implement the proposed method on actual UAV, and experimental flight tests prove the validity of these algorithms. It is seen that our PID design procedure yields tighter UAV performance than an existing popular PID control technique.

The publication resulted from this thesis is listed below:

Kartal, Y., Kolaric, P., Lopez, V. et al. Control Theory Technol. (2019). https://doi.org/10.1007/s11768-020-9145-y

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Quadrotors and other unmanned aerial vehicles (UAVs) have drawn great attention over the last decade because of their high mobility, simplicity of dynamics and capability to perform certain tasks like transportation [1], reconnaissance and monitoring [2]. The fact that quadrotors can be described as a simplified form of the helicopter dynamics [3], leads to similar control design challenges such as under-actuation, strong coupling and unmodelled disturbances. One of the most important challenges of designing a quadrotor controller is providing robust capabilities that provide the ability to follow prescribed trajectories in a stable and reliable manner.

Flight controllers for small micro-air UAVs are normally implemented using PID control, since a PID outer position tracking loop interfaces seamlessly with existing built-in attitude control systems in commercial UAVs. However, the structure and gains of these PID tracking controllers are designed ad hoc, require lengthy flight tests for tuning, and have no performance or robustness guarantees.

On the other hand, a great amount of research has been conducted to design robust control for quadrotors that primarily deals with a linearized model of the quadrotor around a hover condition, which is stable when the small angle approximation is valid for the pitch and roll angles [3–5]. Several works [3, 6–12] deal with nonlinear modeling of the quadrotor, which is preferred to overcome the limitations of the linear model. The Six Degree-of-Freedom (6-DOF) dynamics of a quadrotor result in the typical translational and rotational kinematics equations as in [13] and [14]. The fact that the number of DOF

are less than the number of input commands, which are thrust commands for each of four rotors in our case, rises the problem of the under-actuation [15–17].

Popular controller design methods for quadrotors include linear quadratic regulator (LQR), H-infinity state-space design, model predictive control (MPC), sliding mode techniques [3,8,9] and backstepping [3,8,11,18]. The LQR method is a common technique for controlling aircrafts as it decouples the dynamics and works well with linearized models of the aircraft [4, 5, 19]. In [7, 12, 20], H-infinity controllers are used to perform a robust controller design. In particular, [20] demonstrates that the H-infinity method can be used as a robust attitude controller to handle parameter uncertainties with respect to their nominal values. [1] proposes MPC as a good strategy to handle the effects of the atmospheric turbulence, modeling them as additive disturbances. [21] shows how to use a self-tuning fuzzy PID controller to decrease the tracking error by using AR.Drone 2.0; the authors, however, do not consider the fact that the desired Euler angles of the inner controller loop are related to the dynamics of the outer tracker loop by inverse kinematics, affecting the tracking performance. Sliding variable design in [3, 8] has been used to address this issue by establishing convergence relations between position and velocity errors.

A differential flatness approach is used to design linear output feedback control for quadrotor controllers in [22–24]. The authors deal with estimation of the input-output system model nonlinearities and the unmeasured phase variables to accomplish the trajectory following task by treating the quadrotor as a differentially flat system. [10] proposes an adaptive dynamic controller which improves navigation performance to control UAVs when accomplishing trajectory tracking tasks whereas [9] adopts a super twisting sliding mode control algorithm to study the same task. Uncertainty of the system parameters is examined in [25]. The authors find an adaptive technique based on feedback linearization to prove asymptotic convergence of the tracking errors.

Moreover, the last decade has witnessed rapid progress in swarm of micro-UAVs, that are smaller than 1 meter in scale and 1 kilogram or less in mass [26]. Authors of [27] propose visual relative localization technique for steering swarms of UAVs. In addition, a range of image-based visual servo control algorithms for regulation of the position of UAV is examined in [28]. Also, in [29], authors use the method of visual feedback with multiple cameras using image-processing to solve the positioning problem. In our swarm of UAVs implementation, we use industrial motion capture system that provides the position of multi-agent quadrotor swarm. To achieve the goal of swarming UAVs, we use predetermined trajectories for each agent of the swarm.

Formation is a type of swarm that relies on relative motion of agents [32]. In the literature, numerous researches carried out to establish formation control behavior for swarming systems. Therefore, these formation behaviors are categorized. Three of these subcategories are behavior-based [33], virtual structure-based [34] and leader-follower-based categories [35]. In the behavior-based approaches, each agent of the formation acts according to predetermined scenario. This approach is behaviorally inflexible since the motion is predefined. Alternately, in virtual structure-based approaches, cooperative control of the swarm is ensured by establishing a virtual leader agent. However, since the virtual leader is not exposed to any type of disturbance in the environment, there is a high chance that the followers break formation in the event of unexpected environmental disturbances. On the other hand, the leader-follower approach is based on the real structure in the group, hence all agents of the multi-agent system react the same to any environmental change.

Numerous work deals with linear dynamics of swarm systems [36–40]. One of them, [37] reveals some of the necessary and sufficient conditions to achieve predefined time-varying formation control with switching interaction topologies based on the algebraic Riccati equation. The authors of [40] propose a distributed adaptive control technique that uses adaptive gain scheduling to tune the coupling weights between the individuals of the

swarming group. Furthermore, [41] uses the similar technique to satisfy prescribed H-∞ like performance and to manage the side effects of uncertainties in the system dynamics.

Practical controllers for quadrotors are invariably implemented using PID controllers that interface with the existing built-in UAV attitude controller. PID gains are generally manually tuned.

In this thesis we develop a rigorous nonlinear backstepping method based on the second-order sliding variable to design a PID controller for micro-air UAVs with guaranteed performance. This technique provides the structure for the PID controller, as well as deriving PID gains that result in desired damping ratio and natural frequency, and hence robust and fast response of the UAV for trajectory tracking. It is shown how to select the PID gains to obtain desired transient responses. This PID structure is naturally implemented using the built-in attitude controllers available in commercial UAVs. We propose a novel approach that shows how the desired Euler angles of the inner control loop are related to the dynamics of the outer position tracking loop. Instead of solving the nonlinear inverse kinematic problem, which can cause singularity issues in the controller, we use full nonlinear backstepping design, followed by a small angle approximation to find a controller structure in the form of proportional, integral and derivative terms of outer loop tracking errors. This approach addresses the challenge of choosing proper tracker PID gains in the outer loop of the backstepping controller, and results in a PID controller with guaranteed performance. In flight tests on a real UAV, our PID tracking control loop is seen to exhibit performance similar to the full nonlinear backstepping controller, yet with a far simpler structure.

1.2 Thesis Outline

A brief overview of chapters is given below.

In Chapter 2, we develop a dynamic model based on Newtonian dynamics of the quadrotor. In Chapter 3, we show analysis of the control structures proposed, which involves the classical nonlinear backstepping and PID position control loop from backstepping control methods for the task of trajectory tracking. In addition, we illustrate the attitude controller design procedure for the quadrotors and then we reveal the stability analysis of the proposed control algorithms. In Chapter 4, we show model design and simulation results whereas Chapter 5 and 6 shows the flight tests on a real UAV, where we compare the trajectories followed by AR.Drone 2.0 quadrotor with a full nonlinear backstepping controller and PID controller design with a guaranteed performance.

CHAPTER 2

MATHEMATICAL MODEL

In this section we introduce the standard nonlinear model of the quadrotor dynamics. To localize the quadrotor position, we use the fixed Earth frame. The origin of the three-dimensional (3D) axis system of the Body frame is assumed to be at the center of mass of the quadrotor.

The kinematics of Euler rates can be expressed as

$$\boldsymbol{w_B} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\varphi & c\theta s\varphi \\ 0 & -s\varphi & c\theta c\varphi \end{bmatrix} \dot{\boldsymbol{\eta}} \tag{2.1}$$

where $\boldsymbol{w_B}$ is the vector of the Euler rates and $\boldsymbol{\eta}$ is the Euler Angle vector in φ (phi), θ (theta) and ψ (yaw) order.

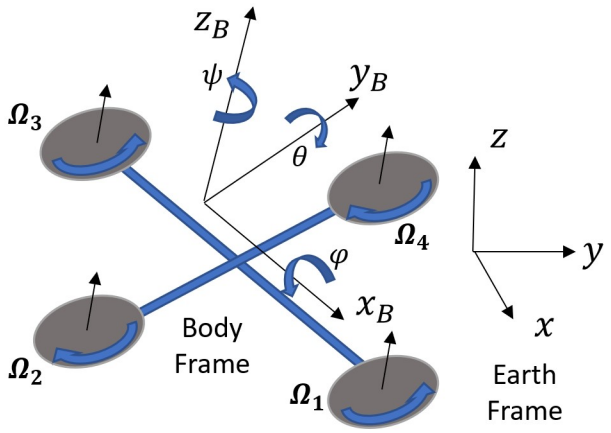

Figure 2.1. Coordinate systems of the quadrotor.

The rotational dynamics are given by

$$\boldsymbol{I_B}\dot{\boldsymbol{w}}_{\boldsymbol{B}} = \boldsymbol{S}\left(\boldsymbol{w_B}\right)\boldsymbol{I_B}\boldsymbol{w_B} + \boldsymbol{\tau_B} \tag{2.2}$$

where $\boldsymbol{S}\left(\boldsymbol{w_B}\right)$ is the skew-symmetric matrix [16], $\boldsymbol{\tau_B}$ is the torque vector and $\boldsymbol{I_B}$ is the inertia matrix defined in Body frame. Then we express the translational dynamics of the quadrotor in the Body frame as

$$m\dot{\boldsymbol{U}} = \begin{bmatrix} 0 \\ 0 \\ \mu \end{bmatrix} - \boldsymbol{R}\boldsymbol{F_g} \tag{2.3}$$

where $\boldsymbol{U} = [u\ v\ w]^T$ is the velocity vector defined in the Body frame, μ is the input total thrust produced by rotors in the Body frame z_B-axis, $\boldsymbol{F_g} = [0\ 0\ mg]^T$ is the gravitational force vector and $\boldsymbol{R}$ is the rotation matrix from the Earth frame to the Body frame. We obtain this rotation matrix using the yaw-pitch-roll (3-2-1) sequence such that [30]

- Right-handed rotation about the z_B-axis, or positive ψ (compass heading).
- Right-handed rotation about the new y_B-axis, or positive θ (pitch).
- Right-handed rotation about the new x_B-axis, or positive φ (roll).

Then, it is given by

$$\boldsymbol{R} = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ -c\varphi s\psi + s\varphi s\theta c\psi & c\varphi c\psi + s\varphi s\theta s\psi & s\varphi c\theta \\ s\varphi s\psi + c\varphi s\theta c\psi & -s\varphi c\psi + c\varphi s\theta s\psi & c\varphi c\theta \end{bmatrix} \tag{2.4}$$

where c and s refers to cosine and sine respectively. Notice that $\boldsymbol{R}$ is a Special Orthogonal matrix with rank 3, or SO(3), whose determinant is equal to 1 [16].

The translational dynamics of the quadrotor in the Earth frame is formulated as

$$m\ddot{\boldsymbol{\xi}} = \boldsymbol{F} - \boldsymbol{F_g} \tag{2.5}$$

where $\boldsymbol{\xi}$ denotes the position vector in the Earth frame and $\boldsymbol{F}$ is the input force vector. Then (2.3) and (2.5) gives the relation $\boldsymbol{F} = \boldsymbol{R}^T \left[0\ 0\ \mu\right]^T$ such that,

$$\boldsymbol{F} = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} \mu(s\varphi s\psi + c\varphi s\theta c\psi) \\ \mu(-s\varphi c\psi + c\varphi s\theta s\psi) \\ \mu(c\varphi c\theta) \end{bmatrix} \tag{2.6}$$

CHAPTER 3

CONTROLLER DESIGN

3.1 Outer position backstepping controller

This section explains the full nonlinear backstepping control design for the quadrotor. The backstepping control structure derived here is shown in 3.1. Standard backstepping is then modified in Section 3.2 to generate inner loop attitude and altitude commands that are commensurate with existing built-in UAV controllers. Then in Section 3.3. we design an inner attitute controller to complete the overall control structure. Moreover, Section 3.4 reveals rigorous stability analysis of the proposed control algorithm.

3.1.1 Exact nonlinear backstepping tracker

To apply the classical backstepping control method to the system defined in (2.5), we begin by adding and removing $\boldsymbol{F_d}$, ideal virtual force input, and obtain the Newtonian model in terms of the desired forces

$$m\ddot{\boldsymbol{\xi}} = \boldsymbol{F_d} - \boldsymbol{F_g} + \tilde{\boldsymbol{F}}_{\boldsymbol{d}} \tag{3.1}$$

where $\tilde{\boldsymbol{F}}_{\boldsymbol{d}} = \boldsymbol{F} - \boldsymbol{F_d}$. $\boldsymbol{F_d}$ is now selected to get $\boldsymbol{\xi} \rightarrow \boldsymbol{\xi_d}$, where $\boldsymbol{\xi_d} = [x_d\ y_d\ z_d]^T$ is the given desired position vector in the Earth frame. Then, $\boldsymbol{\tau_B}$ in (2.2) is designed using desired Euler angle information matrix $\boldsymbol{\eta_d} = [\varphi_d\ \theta_d\ \psi_d]^T$ and the time rate of change of desired vertical speed information, $\dot{\boldsymbol{w}}_{\boldsymbol{d}}$, as explained in Section 3.3.

Define the position error

$$\boldsymbol{e} = \boldsymbol{\xi_d} - \boldsymbol{\xi} \tag{3.2}$$

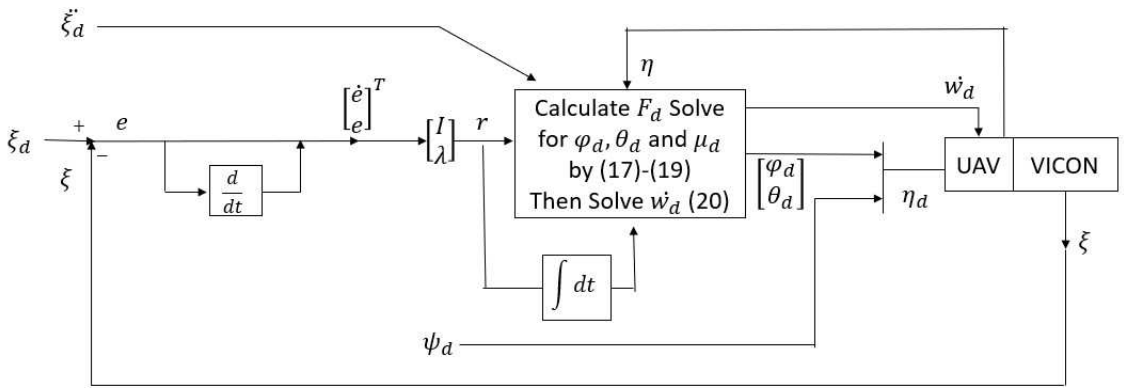

Figure 3.1. Tracker design of backstepping controller using sliding variable.

and express the sliding variable in terms of weighted sum of the position and velocity errors [31]

$$\boldsymbol{r} = \dot{\boldsymbol{e}} + \boldsymbol{\lambda e} \tag{3.3}$$

where $\boldsymbol{\lambda}$ is a diagonal matrix constructed from positive elements. The error dynamics can be obtained as

$$\begin{aligned} m\dot{\boldsymbol{r}} &= m\ddot{\boldsymbol{e}} + m\boldsymbol{\lambda}\dot{\boldsymbol{e}} \\ &= m\ddot{\boldsymbol{\xi}}_{\boldsymbol{d}} - m\ddot{\boldsymbol{\xi}} + m\boldsymbol{\lambda}\left(\boldsymbol{r} - \boldsymbol{\lambda e}\right) \\ &= m\ddot{\boldsymbol{\xi}}_{\boldsymbol{d}} - \boldsymbol{F_d} + \boldsymbol{F_g} - \tilde{\boldsymbol{F}}_{\boldsymbol{d}} + m\boldsymbol{\lambda}(\boldsymbol{r} - \boldsymbol{\lambda e}). \end{aligned} \tag{3.4}$$

Then we select the desired force vector $\boldsymbol{F_d}$ as

$$\boldsymbol{F_d} = m\ddot{\boldsymbol{\xi}}_{\boldsymbol{d}} + \boldsymbol{F_g} + m\boldsymbol{\lambda r} - m\boldsymbol{\lambda^2 e} + \boldsymbol{K_r r} + \boldsymbol{K_i}\int \boldsymbol{r}dt \tag{3.5}$$

where $\boldsymbol{K_r}$ and $\boldsymbol{K_i}$ are diagonal matrices constructed from positive elements. Then the closed loop error dynamics become

$$m\dot{\boldsymbol{r}} = -\boldsymbol{K_r r} - \boldsymbol{K_i}\int \boldsymbol{r}dt + \tilde{\boldsymbol{F}}_{\boldsymbol{d}} \tag{3.6}$$

$\tilde{\boldsymbol{F}}_{\boldsymbol{d}}$ must be kept at zero by the attitude controller proposed in Section 3.3, making (3.6) stable for all $\boldsymbol{K_r}>\mathbf{0}$ and $\boldsymbol{K_i}>\mathbf{0}$.

3.1.2 Gain design for desired transient response

Herein we show how to select gains $\boldsymbol{K_r}$ and $\boldsymbol{K_i}$ to achieve desired transient response and error magnitude. It can be shown that $\boldsymbol{e}$ is bounded if the sliding variable $\boldsymbol{r}$ is bounded [7] such that

$$||\boldsymbol{e}|| \leq \frac{||\boldsymbol{r}||}{\sigma_{min}\left(\boldsymbol{\lambda}\right)}, \, |\,|\dot{\boldsymbol{e}}|\,| \leq |\,|\boldsymbol{r}|\,| \tag{3.7}$$

After differentiating (3.6) and normalizing with respect to the highest order coefficient, we can write

$$\ddot{\boldsymbol{r}} + \frac{\boldsymbol{K_r}}{m}\dot{r} + \frac{\boldsymbol{K_i}}{m}\boldsymbol{r} = \mathbf{0_{3\times 1}}. \tag{3.8}$$

We define $\boldsymbol{K_r}$ and $\boldsymbol{K_i}$ in the form of diagonal matrices with k_{rj} and k_{ij} diagonal entries, respectively.

Recognizing that the general form of the second-order characteristic polynomial of (3.8) is in the form of $s^2 + 2\zeta\omega_n s + {\omega_n}^2$, the desired performance of closed loop error dynamics can be achieved by setting

$$k_{ij} = {\omega_n}^2 m, \quad k_{rj} = 2m\zeta\omega_n \tag{3.9}$$

where ω_n is the natural frequency and ζ is the damping ratio of the quadrotor's transfer function.

3.1.3 Desired Euler angles

We show here how to complete the position tracker desired force (3.5) to the desired Euler angles in the rotational dynamics. We couple the position tracker loop in Fig. 2 to the

rotation dynamics (2.2) by selecting prescribed attitude angles that yield the desired force vector (3.5). This is accomplished by using the inverse kinematics approach.

Then by the mathematical model of the quadrotor given in (2.3) and (2.5), and the relation (2.6), it can be deduced that $\boldsymbol{F_d} = \boldsymbol{R}^T \left[0\ 0\ \mu_d\right]^T = \left[f_{x_d}\ f_{y_d}\ f_{z_d}\right]^T$. Then u_d, φ_d and θ_d can be expressed in terms of known quantities f_{x_d}, f_{y_d} and f_{z_d} as

$$\begin{bmatrix} f_{x_d} \\ f_{y_d} \\ f_{z_d} \end{bmatrix} = \begin{bmatrix} \mu_d(s\varphi_d s\psi_d + c\varphi_d s\theta_d c\psi_d) \\ \mu_d(-s\varphi_d c\psi_d + c\varphi_d s\theta_d s\psi_d) \\ \mu_d(c\varphi_d c\theta_d) \end{bmatrix} \tag{3.10}$$

which implies

$$\begin{aligned} \tan\theta_d &= \frac{f_{x_d}\cos\psi_d + f_{y_d}\sin\psi_d}{f_{z_d}} \\ \theta_d &= \tan^{-1}\frac{f_{x_d}\cos\psi_d + f_{y_d}\sin\psi_d}{f_{z_d}} \end{aligned} \tag{3.11}$$

$$\begin{aligned} \tan\varphi_d &= \frac{\cos\theta_d\,(f_{x_d}\sin\psi_d - f_{y_d}\cos\psi_d)}{f_{z_d}} \\ \varphi_d &= \tan^{-1}\frac{\cos\theta_d\,(f_{x_d}\sin\psi_d - f_{y_d}\cos\psi_d)}{f_{z_d}} \end{aligned} \tag{3.12}$$

$$\mu_d = \frac{f_{z_d}}{\cos\varphi_d\,\cos\theta_d}. \tag{3.13}$$

Notice that ψ_d can be arbitrarily prescribed, and only the variables θ_d, φ_d and μ_d must be found. Fig. 3.1 shows the outer control loop design of the backstepping controller.

The inner loop control design of the backstepping method requires the time rate of change of the desired vertical speed information, $\dot{w}_d$, calculated by using (2.3) such that

$$\dot{w}_d = \frac{\mu_d}{m} - g\cos\varphi_d\,\cos\theta_d \tag{3.14}$$

3.2 PID position control loop from backstepping controller

This section shows that, with a proper formulation, a suitably designed PID controller can in fact be rigorously developed from the backstepping controller in Fig. 2. We show how the method analyzed in Section 3.1 can be used to design a PID controller with guaranteed performance and transient responses. The PID controller structure developed here is shown in Fig. 3. Selection of the final PID gains in equations (3.31)-(3.32) depends on $\boldsymbol{K_r}$ and $\boldsymbol{K_i}$ in (3.8) through (34)-(36). As such these PID gains can guarantee the prescribed damping ratio and the natural frequency.

The procedure for solving (3.10)-(3.13) to obtain θ_d, φ_d and μ_d can be simplified by adopting a small angle approximation on (3.10) with respect to θ_d and φ_d to obtain following

$$\begin{bmatrix} f_{xd} \\ f_{yd} \\ f_{zd} \end{bmatrix} = \begin{bmatrix} \mu_d(\varphi_d s\psi_d + \theta_d c\psi_d) \\ \mu_d(-\varphi_d c\psi_d + \theta_d s\psi_d) \\ \mu_d \end{bmatrix} \tag{3.15}$$

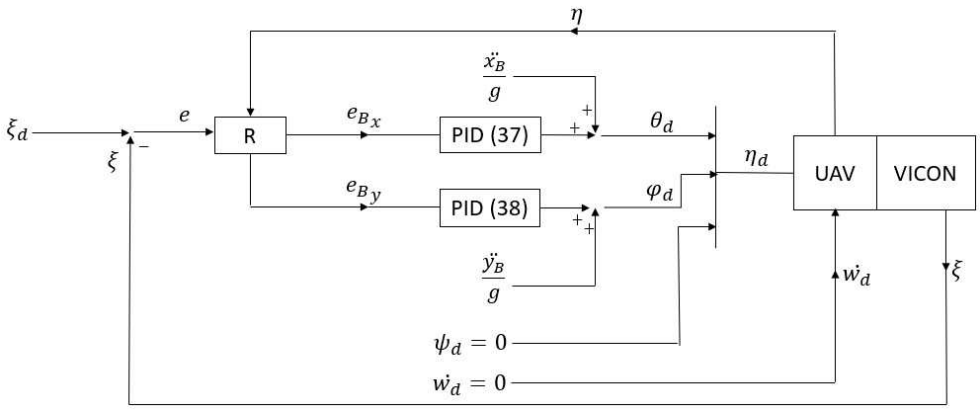

Figure 3.2. Tracker design of backstepping controller using PID control method.

which leads to

$$\theta_d = \frac{f_{x_d}\cos\psi_d + f_{y_d}\sin\psi_d}{f_{z_d}} \tag{3.16}$$

$$\varphi_d = \frac{f_{x_d}\sin\psi_d - f_{y_d}\cos\psi_d}{f_{z_d}} \tag{3.17}$$

$$\mu_d = f_{z_d} \tag{3.18}$$

$$\dot{w}_d = \frac{\mu_d}{m} - g \tag{3.19}$$

The trackers for quadrotors are often designed as a PID controller. Tracking is achieved by setting desired pitch angle to control position in x-axis, setting desired roll angle and vertical speed to control position in y-axis and z-axis respectively. Moreover, we set the desired yaw angle as zero not to be exposed of considerable drag force. Controlling height separately means that the change in pitch and roll angles is assumed to affect only the motion in the x-y plane.

Equation (3.15) shows that the z-axis is naturally decoupled from x-y plane motion. As a result of this decoupling, the UAV needs to hover in z-axis, which is guaranteed by equating f_{z_d} to mg, in (3.1). This implies that in the backstepping controller with PID position control loop (3.15), we equate μ_d to mg. Then

$$\theta_d = \frac{f_{x_d}}{mg} \tag{3.20}$$

$$\varphi_d = \frac{-f_{y_d}}{mg} \tag{3.21}$$

$$\mu_d = mg \tag{3.22}$$

Furthermore, by (3.19), $\dot{w}_d$ becomes zero as the quadrotor keeps its altitude at a certain level or the x-y plane is decoupled from z-axis, also we use (3.5) and (3.20)-(3.22) to obtain the following

$$\theta_d = [1\ 0\ 0][\frac{\ddot{\boldsymbol{\xi}}_{\boldsymbol{d}}}{g} + \frac{\boldsymbol{K_r\lambda + K_i}}{mg}\boldsymbol{e} + \frac{m\boldsymbol{\lambda} + \boldsymbol{K_r}}{mg}\dot{\boldsymbol{e}} + \frac{\boldsymbol{K_i\lambda}}{mg}\int \boldsymbol{e}dt] \tag{3.23}$$

$$\varphi_d = -[0\ 1\ 0][\frac{\ddot{\boldsymbol{\xi}}_{\boldsymbol{d}}}{g} + \frac{\boldsymbol{K_r\lambda + K_i}}{mg}\boldsymbol{e} + \frac{m\boldsymbol{\lambda} + \boldsymbol{K_r}}{mg}\dot{\boldsymbol{e}} + \frac{\boldsymbol{K_i\lambda}}{mg}\int \boldsymbol{e}dt] \tag{3.24}$$

In (3.23) and (3.24), $\frac{\boldsymbol{K_r\lambda + K_i}}{mg}$ stands for the proportional gain term, $\frac{\boldsymbol{K_i\lambda}}{mg}$ stands for the integral gain term, and $\frac{m\boldsymbol{\lambda} + \boldsymbol{K_r}}{mg}$ is the derivative gain term of the PID controller.
Note that these gains are given naturally in terms of the gains used in the backstepping controller, which are easy to find for good performance as given in (3.8). The term $\ddot{\boldsymbol{\xi}}_{\boldsymbol{d}}$ in (3.23) and (3.24) is a feed-forward term of the controller, which improves its performance in terms of decreasing tracking error, particularly in high accelerations. Furthermore, ψ_d and $\dot{w}_d$ can be set to zero.

To get the quadrotor to follow predetermined trajectory at a certain altitude, we designed a finite state machine (FSM) whose details are given in Chapter 5. Keeping in mind that natural frequency and zeta values for pitch and roll angles are very close to each other from the assumption of symmetric quadrotor, we can write $\boldsymbol{\lambda}$, $\boldsymbol{K_r}$ and $\boldsymbol{K_i}$ in the form of $c_j * \boldsymbol{I}$ where c_j, $j = 1,\ 2, 3$, are positive constants and $\boldsymbol{I}$ is a 3×3 identity matrix, (3.23) and (3.24) can be further simplified such that the position error can be evaluated in Body frame. Define the position error in Body frame $\boldsymbol{e_B}$ as

$$\boldsymbol{e_B} = \boldsymbol{Re} \tag{3.25}$$

Let the position error in x-axis of the Body frame and y-axis of the Body frame be e_{Bx} and e_{By} respectively, such that

$$e_{Bx} = [1\ 0\ 0]\boldsymbol{e_B} \tag{3.26}$$

$$e_{By} = [0\ 1\ 0]\boldsymbol{e_B} \tag{3.27}$$

and we write the assumptions for $\boldsymbol{\lambda}$, $\boldsymbol{K_r}$ and $\boldsymbol{K_i}$ matrices

$$\boldsymbol{\lambda}=c_1\boldsymbol{I} \tag{3.28}$$

$$\boldsymbol{K_r}=c_2\boldsymbol{I} \tag{3.29}$$

$$\boldsymbol{K_i}=c_3\boldsymbol{I} \tag{3.30}$$

Substituting (3.26)-(3.30) in (3.23) and (3.24) yields the following PID controller equations

$$\theta_d = \frac{\ddot{x_{Bd}}}{g}+\frac{c_2c_1+c3}{mg}e_{Bx} + \frac{mc_1+c_2}{mg}\dot{e_{Bx}}+\frac{c_1c_3}{mg}\int e_{Bx}dt \tag{3.31}$$

$$\varphi_d = -\frac{\ddot{y_{Bd}}}{g}-\frac{c_2c_1+c3}{mg}e_{By}-\frac{mc_1+c_2}{mg}\dot{e_{By}}-\frac{c_1c_3}{mg}\int e_{By}dt \tag{3.32}$$

which is simplified solution of $\boldsymbol{F_d}$. Fig. 3.2 shows the simplified design of the PID position control loop of the backstepping controller.

Selection of the final PID gains in equations (3.31)-(3.32) depends on $\boldsymbol{K_r}$ and $\boldsymbol{K_i}$ in (3.8) through (3.8)-(3.30). As such these PID gains can guarantee prescribed damping ratio and natural frequency.

3.3 Inner attitude controller

In this section, we explain the second step of the backstepping method for the quadrotor, which is attitude control.

It is beneficial to develop a motor model to analyze a quadrotor's ability to produce linear and angular velocities from a nominal state. To establish the relation between thrust produced by the rotors and angular velocity of the rotor, it is reasonable to assume that $f_i = t\Omega_i^2 \ \ \forall i = 1, 4$ where f_i is the thrust produced by each rotor.

The attitude controller is generally built in to the UAV and cannot be modified. Consequently, the interface between the outer-loop PID controller just designed and the attitude controller must be properly crafted.

This attitude controller must take the desired values of φ_d, θ_d and ψ_d (3.23)-(3.24) as inputs and produce the torque vector τ_B in (2.2) to stabilize the rotational dynamics according to the expression

$$\boldsymbol{\tau_B} = \begin{bmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lt({\Omega_4}^2 - {\Omega_2}^2) \\ lt({\Omega_3}^2 - {\Omega_1}^2) \\ d({\Omega_1}^2 + {\Omega_3}^2 - {\Omega_2}^2 - {\Omega_4}^2) \end{bmatrix} \tag{3.33}$$

where l is the lever length, t is the thrust factor, Ω_i, i = $1, 2, 3, 4$, is the angular velocity of the rotor and d is the drag factor. The direction of angular velocities for each rotor is given in Fig.4 as while first and fourth rotor turn anti-clockwise, the other two turn clockwise. This is because of canceling the yawing moments generated while quadrotor is at nominal condition (hover). The total thrust, μ_d, is equal to the sum of thrusts generated by each rotor, that is

$$\mu_d = t({\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2) \tag{3.34}$$

Then the attitude controller to generate changes in angular velocities using PID control is designed as

$$\begin{bmatrix} \Delta\Omega_\varphi \\ \Delta\Omega_\theta \\ \Delta\Omega_\psi \end{bmatrix} = \begin{bmatrix} P_\varphi(\varphi_d - \varphi) + D_\varphi(p_d - p) + I_\varphi \int(\varphi_d - \varphi) \\ P_\theta(\theta_d - \theta) + D_\theta(q_d - q) + I_\theta \int(\theta_d - \theta) \\ P_\psi(\psi_d - \psi) + D_\psi(r_d - r) + I_\psi \int(\psi_d - \psi) \end{bmatrix} \tag{3.35}$$

where p_d, q_d and r_d are calculated using kinematics of the Euler rates (2.1) and desired

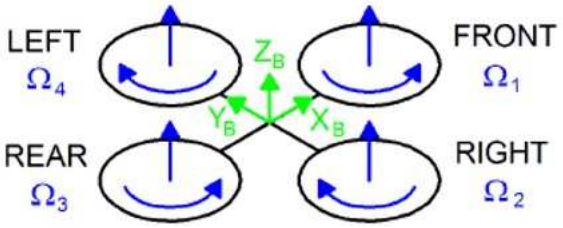

Figure 3.3. Quadrotor configuration frame.

values of Euler angles, such that

$$\begin{bmatrix} p_d \\ q_d \\ r_d \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s\theta_d \\ 0 & c\varphi_d & c\varphi_d s\varphi_d \\ 0 & -s\varphi_d & c\theta_d c\varphi_d \end{bmatrix} \begin{bmatrix} \dot{\varphi}_d \\ \dot{\theta}_d \\ \dot{\psi}_d \end{bmatrix} \tag{3.36}$$

and the values of p, q and r are obtained from (2.1).

Finally, we obtain the desired angular velocity of each rotor as

$$\begin{bmatrix} \Omega_{1d} \\ \Omega_{2d} \\ \Omega_{3d} \\ \Omega_{4d} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \Omega_h + \Delta\Omega_{net} \\ \Delta\Omega_\varphi \\ \Delta\Omega_\theta \\ \Delta\Omega_\psi \end{bmatrix} \tag{3.37}$$

where $\Delta\Omega_\varphi$, $\Delta\Omega_\theta$ and $\Delta\Omega_\psi$ are computed in (3.35), Ω_h is the rotor speed required to hover such that

$$\Omega_h = \sqrt{\frac{mg}{4t}} \tag{3.38}$$

and $\Delta\Omega_{net}$ is the outcome of desired vertical speed, w_d, in the form of

$$\Delta\Omega_{net}=\frac{m}{8\mathrm{t}\Omega_h}\dot{w}_d \tag{3.39}$$

substituting (3.37) in (3.33) yields the desired torque vector.

3.4 Stability analysis

This section reveals the stability analysis of the backstepping control method in Chapter 2, based on Lyapunov analysis. The assumptions are:

Assumption 1 The quadrotor is a rigid body.

Assumption 2 The disturbance and aerodynamic forces that interferes with the fuselage terms are negligible.

Assumption 3 In Section 3.1, we assume pitch and roll angles of the quadrotor do not exceed -π/2 and π/2 bounds whereas in Section 3.2, we accept they do not exceed -π/6 and π/6 bounds.

Assumption 4 Inertial Frame (Earth Frame) is fixed in the 3D space.

The next theorem shows stability of the error dynamics (3.6). It combines the analysis in [6] with our novel design in terms of gains $\boldsymbol{K_r}$ and $\boldsymbol{K_i}$ in (3.6), and the performance of the inner attitude controller in Section 3.3.

Theorem 1. *Given the force* (3.5)*, desired Euler angles in* (3.11),(3.12) *and thrust* (3.13)*. Assume the inner attitude controller* (3.33)*, tracks the Euler angles* (3.11),(3.12) *and vertical speed* (3.14)*. Then the error dynamics* (3.6) *are asymptotically stable.*

Proof. The candidate Lyapunov function is

$$\boldsymbol{L} = \frac{1}{2}\boldsymbol{r}^T\boldsymbol{P_1}\boldsymbol{r} + \frac{1}{2}\left[\int_0^t \boldsymbol{r}^T dt\right]\boldsymbol{P_2}\left[\int_0^t \boldsymbol{r} dt\right] \tag{3.40}$$

where $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ are positive definite matrices. By using the Leibniz Integral Rule, the derivative of $\boldsymbol{L}$ becomes

$$\dot{\boldsymbol{L}} = \boldsymbol{r}^T\boldsymbol{P_1}\dot{\boldsymbol{r}} + \left[\int_0^t \boldsymbol{r}^T dt\right]\boldsymbol{P_2}\boldsymbol{r}. \tag{3.41}$$

and substituting (3.6) in (3.41) results in the following

$$\begin{aligned}\dot{\boldsymbol{L}} = -\boldsymbol{r}^T\frac{\boldsymbol{P_1}\boldsymbol{K_r}}{m}\boldsymbol{r} - \boldsymbol{r}^T\frac{\boldsymbol{P_1}\boldsymbol{K_i}}{m}\left[\int_0^t \boldsymbol{r} dt\right] \\ +\boldsymbol{r}^T\boldsymbol{P_1}\frac{\tilde{\boldsymbol{F}}_{\boldsymbol{d}}}{m} - \left[\int_0^t \boldsymbol{r}^T dt\right]\boldsymbol{P_2}\boldsymbol{r}.\end{aligned} \tag{3.42}$$

Select $\boldsymbol{P_2}$ and note that it is positive definite and diagonal

$$\boldsymbol{P_2} = \frac{\boldsymbol{P_1}\boldsymbol{K_i}}{m}. \tag{3.43}$$

Then (3.42) becomes

$$\dot{\boldsymbol{L}} = -\boldsymbol{r}^T\frac{\boldsymbol{P_1}\boldsymbol{K_r}}{m}\boldsymbol{r} + \boldsymbol{r}^T\boldsymbol{P_1}\frac{\tilde{\boldsymbol{F}}_{\boldsymbol{d}}}{m}. \tag{3.44}$$

Notice that tracking the desired Euler angles in (3.11),(3.12) and vertical velocity (3.14) by the attitude controller in Section 3.3, guarantees μ_d in (3.13) is the same as μ_d in (3.34). This means the force generated in the Earth Frame is actually in the form of the desired force $\boldsymbol{F_d}$, given in (3.5) by the relation $\boldsymbol{F_d} = \boldsymbol{R}^T\left[0\ 0\ \mu_d\right]^T$, hence $\boldsymbol{F} \to \boldsymbol{F_d}$ and $\tilde{\boldsymbol{F}}_{\boldsymbol{d}} \to \boldsymbol{0}$. Then (3.44) becomes

$$\dot{\boldsymbol{L}} = -\boldsymbol{r}^T \frac{\boldsymbol{P_1 K_r}}{m} \boldsymbol{r} \tag{3.45}$$

which is negative definite. Therefore, the closed-loop error dynamics given in (3.6) is asymptotically stable. □

CHAPTER 4

MODEL DESIGN & SIMULATION RESULTS

In this chapter, we share the simulation model design of the AR.Drone 2.0 for both theoretical and identified model. Both of the simulation models are created using MATLAB-Simulink environment.

4.1 Theoretical Model

Herein we show the simulation model based on the mathematical model of the quadrotor helicopter defined in Chapter 2. Figure 4.1 shows the Simulink design of the mathematical model.

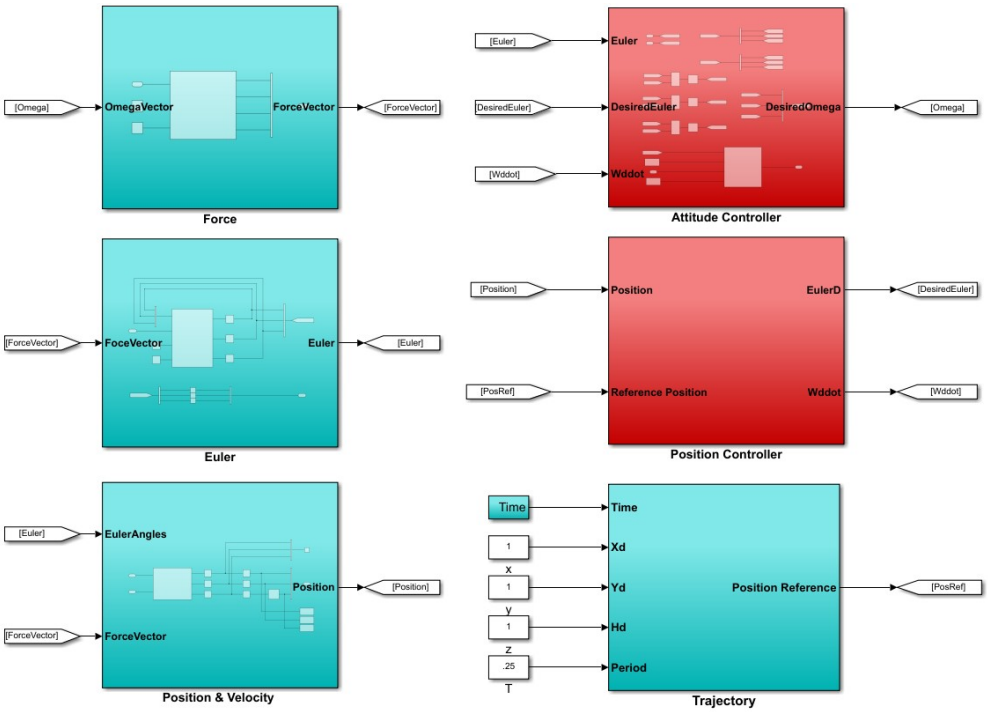

Figure 4.1. Theoretical Model.

4.2 Identified Model

Herein we show the identified model based on the data collected through the flight tests. The identified model is created using System identification toolbox of MATLAB. Figure 4.3 shows the Simulink design of the identified model. In the background of the identifying model process, this toolbox uses recursive least squares method. Graphical User Interface of System Identification Tool is given in the Fig 4.2.

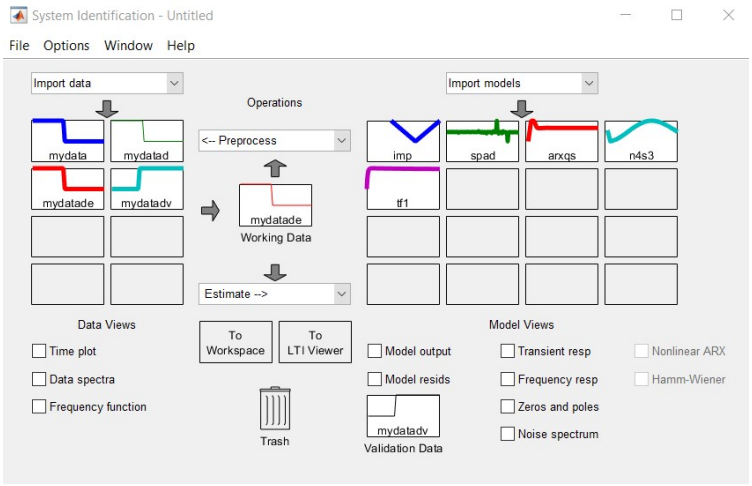

Figure 4.2. GUI of System Identification Tool.

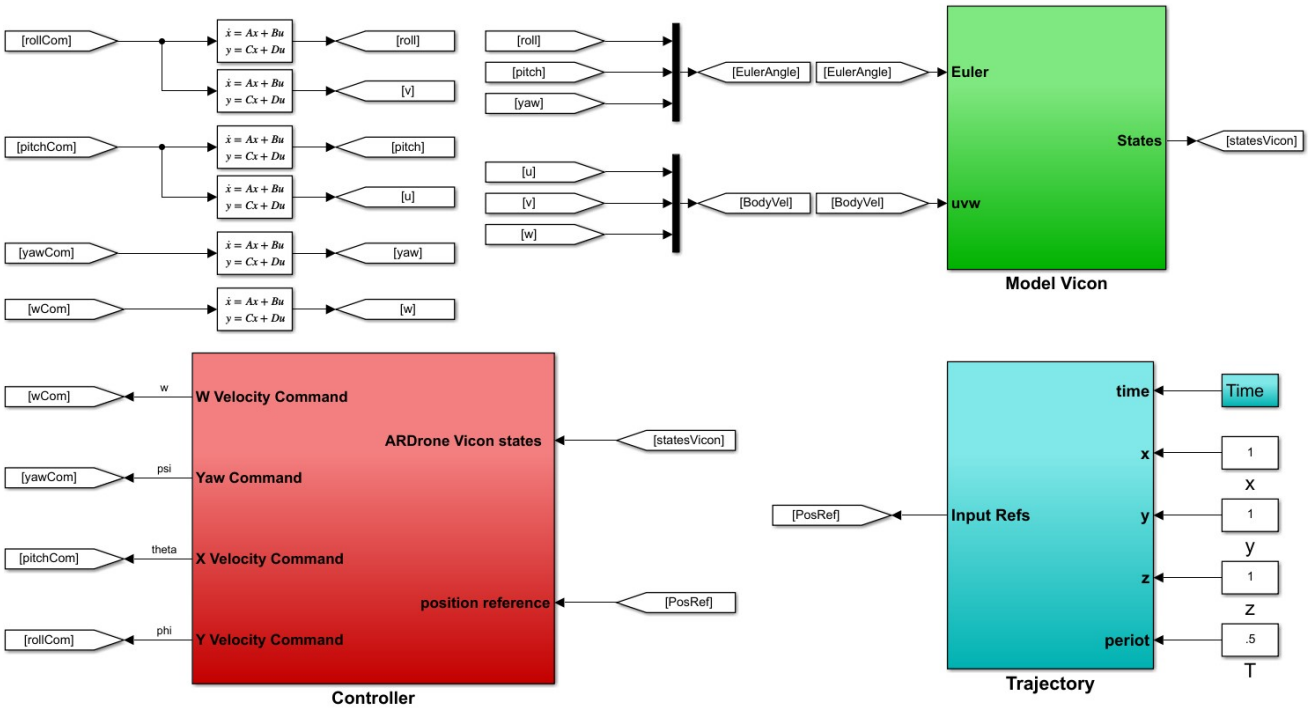

Figure 4.3. Identified Model.

4.3 Simulation Results

This section reveals the simulation results obtained with different trajectories and scenarios. We first show the trajectories followed using the theoretical model. Then we share the graphical results when the identified model is used.

4.3.1 Theoretical Model Results

In this section, we give the show the desired and followed trajectories of the UAV using the theoretical model. For the backstepping-PID controller proposed in Fig. 3.2, Fig. following figures show the desired path and the path tracked by the UAV respectively when both circular and 8-figure are desired trajectories.

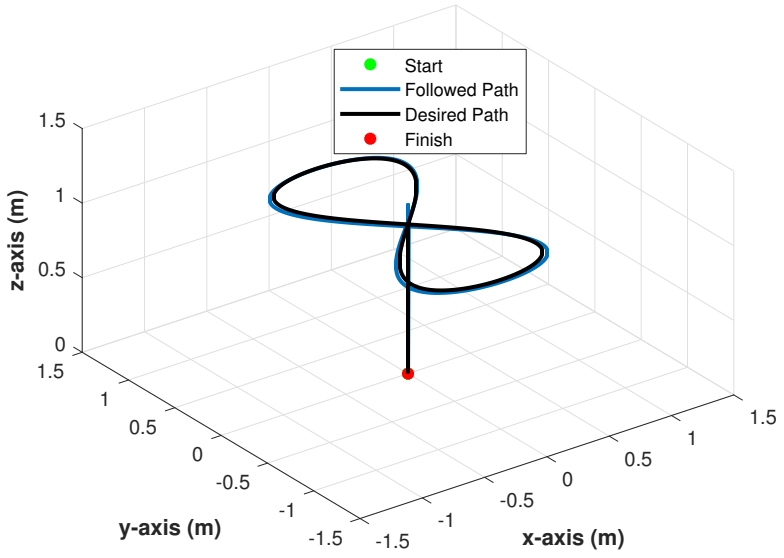

Figure 4.4. Theoretical model eight-figure desired trajectory implementation..

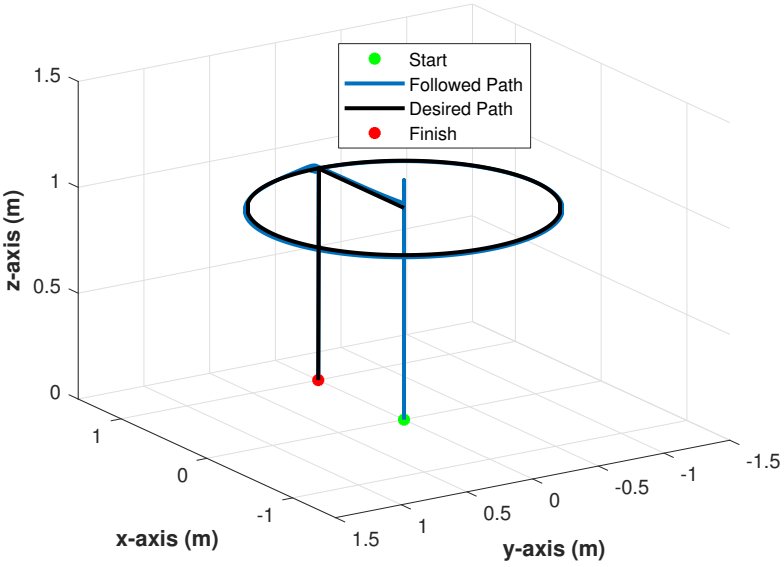

Figure 4.5. Theoretical model circular-figure desired trajectory implementation.

4.3.2 Identified Model Results

In this section, we show the desired and followed trajectories of the UAV using the model. For the backstepping-PID controller proposed in Fig. 3.2, Fig. 4.6 and Fig. 4.7

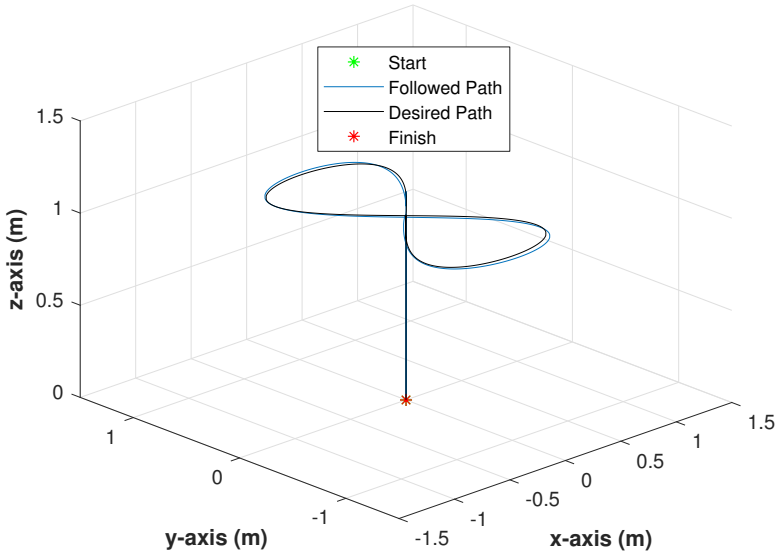

Figure 4.6. Identified model eight-figure desired trajectory implementation..

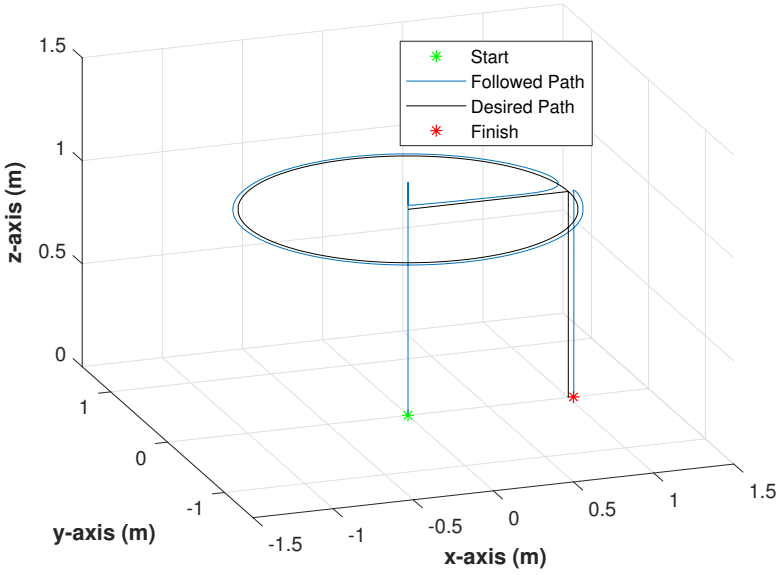

Figure 4.7. Identified model circular-figure desired trajectory implementation..

show the desired path and the path tracked by the UAV respectively when both circular and 8-figure are desired trajectories.

CHAPTER 5

EXPERIMENT DESIGN AND FLIGHT TEST DETAILS

This section addresses the crucial elements of our experiments, which are the lab equipment, the trajectory generation and the flight controller design. We also give details of the problems encountered and solved through implementation of proposed control algorithms.

5.1 Lab environment

Here we illustrate the lab equipment and their specifications. Fig. 5.1 shows the entire system designed at UTA Research Institute for testing the backstepping control approaches on UAVs. Basic three elements are the Vicon, the Parrot AR.Drone 2.0, and the master computer. Vicon is a motion capture system that provides the position of the UAV by sending infrared waves to the markers that are sensitive to these waves. In the application procedure, these markers are placed on the UAV and Vicon is started. In the experiments, we use eight Vicon camera system.

The communication between master computer and Vicon is provided via User Datagram Protocol (UDP). The frequency of the UDP Packets taken from Vicon motion capture system is 100 Hz, meaning that in every 10 milliseconds, the master computers gets the position information of the AR.Drone 2.0.

The AR.Drone 2.0 shown in Fig. 5.2 is a well-known Parrot produced quadrotor that has a built-in gyroscope and the Inertial Measurement Unit (IMU) sensor suite. In the experiments introduced in Chapter 5, we use the cover of AR. Drone 2.0 shown in Fig.

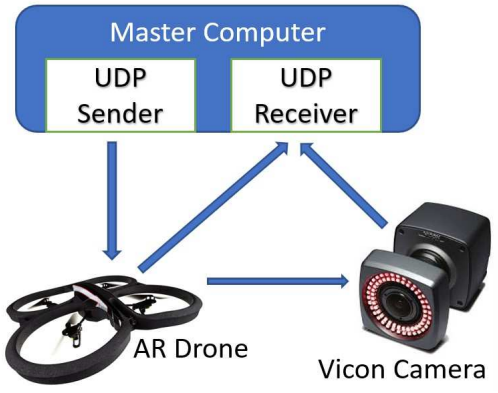

Figure 5.1. Autonomous systems lab experiment.

5.2 regarding safety. The desired pitch, roll and yaw angles and time rate of the vertical velocity commands are sent to the AR.Drone to track the desired position.

Euler angle information of the UAV is obtained via IMU that is embedded in the AR.Drone. In practical applications many quadrotors are designed with a built-in attitude controller and AR.Drone has its own attitude controller. This controller takes the desired values of φ_d, θ_d and ψ_d as inputs. The Parrot quadrotor also has a built-in altitude hold controller, that takes the desired vertical velocity command as input. The communication between the master computer and the AR.Drone is done via UDP. The frequency of UDP packages is determined by the wifi_rate, which is the network communication parameter of the AR.Drone and it is not allowed for developer to change this parameter.

MATLAB-Simulink is used to create UDP nodes that are communicating with the AR.Drone 2.0 and Vicon software. The receiver and the sender UDP nodes are inserted to the Simulink model in the form of S-functions. The controller and the trajectory generation algorithms with the FSM design are implemented in the Simulink model. Simulink-Desktop Real Time Add-on is used to send the real time commands to the AR.Drone, whose details given at Chapter 5. The UDP nodes created are tolerable up to %10 packet loss rate,

Figure 5.2. The Parrot AR.Drone 2.0.

which is necessary to handle communication channel noise created by the lab environment.

5.2 Trajectory generation

In this section, we recall how to generate a suitable trajectory for the quadrotor systems. To begin with, let $\zeta(t)$:$\mathbb{R}\rightarrow\mathbb{R}^3$ be a desired path to follow in analytical form and ζ_d= $(t, x, y, z)\in\mathbb{R}^4$ be a representation of a desired path $\zeta(t)$ in a form of a dense collection of vectors. Notice that $\zeta(t)$ can be constructed even for the sets of points that do not conform to a specific analytical function.

We construct the heuristic function that interpolates $\zeta(t)$ into a set of way-points $(\xi_i, t_i)\rightarrow(\mathbb{R}^3, \mathbb{R}^1)$ based on the local curvature. Each segment (t_i, t_{i+1}) is a third order polynomial with unspecified parameters, which must be determined. One approach to find them includes the use of parametric constraints such that

$$\begin{aligned} \xi(t_i) &= \xi_{t_i} \\ \xi(t_i{+}T) &= \xi_{t_{i+1}}(t_{i+1}) \\ \dot{\xi}(t_i) &= \dot{\xi}_{t_i} \\ \dot{\xi}(t_i{+}T) &= \dot{\xi}_{t_{i+1}}(t_{i+1}) \end{aligned} \tag{5.1}$$

This 4th order system of equations is solved to determine the coefficients of the polynomial for each segment

$$\begin{bmatrix} 1 & t & t^2 & t^3 \\ & 1 & 2t & 3t^2 \\ & & 2 & 6t \\ & & & 6 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \xi_{t_i} \\ \xi_{t_{i+1}} \\ \dot{\xi}_{t_i} \\ \dot{\xi}_{t_{i+1}} \end{bmatrix} \tag{5.2}$$

5.3 Flight controller design

The flight controller is a high-level decision-making mechanism that activates different modes of the operation depending on the state of UAV. We recognize three modes of the operation in our MATLAB-Simulink implementation, which are IDLE, HOVER and TRACK_PATH modes as shown in Fig. 5.3. The quadrotor enters the IDLE mode, when either we turn-on the AR.Drone 2.0 manually or it is landed by receiving Land command.

As we turn the quadrotor on, it directly calibrates itself, which requires its initial attitude to be parallel with respect to the ground. While the UAV is in IDLE mode, it is actively receiving data packets via UDP and is ready to get TakeOff command. If the TakeOff command is sent to the quadrotor, the value of vertical velocity command of the AR.Drone 2.0 is incrementally increased until the desired height value is achieved. This command is to generate the sufficient thrust to take the AR.Drone 2.0 off the ground and to get the quadrotor in hover at a certain altitude.

It is critical that the thrust is balanced over the propellers to ensure that the take-off is vertical. When UAV reaches the desired height, z_d, the flight controller switches to the HOVER mode. The physical shape of the UAV as well as aerodynamical effects make it hard to generate the balanced thrust vector summed over all four propellers. That results in curved take-off trajectory until hover altitude is reached. Notice that AR.Drone 2.0 is not a physically symmetric on its x-y plane To amend for the offset in the initial position,

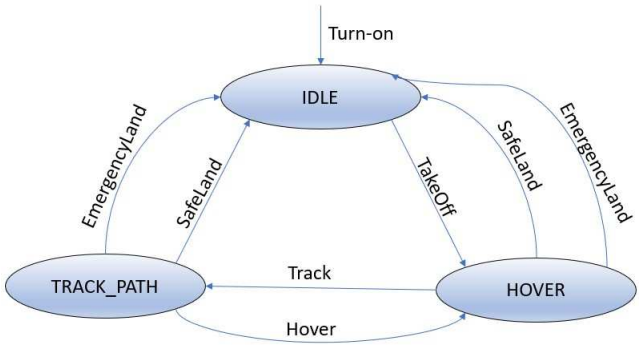

Figure 5.3. The flight controller FSM design.

we generate position command (0,0,0.75) as soon as possible, thus securing the hovering precisely above the origin. This motion is represented clearly in the figures of Chapter 6.

In the HOVER mode, the UAV is at the nominal condition, and controller's task is to keep the attitude parallel to ground. If the Track command is received in the HOVER mode, the flight controller switches to the PATH_TRACK mode. In this mode, the AR.Drone 2.0 tracks the predetermined trajectory. We record both the desired and the followed position data by the quadrotor in MATLAB workspace in the real-time flight tests.

Each iteration of the PATH_TRACK mode first reads the IMU sensor buffers and then Vicon buffer to construct the close loop error dynamics. Since we have two different backstepping control design we choose one of them at this point. If the control algorithm suggested in Chapter 3.1 is used, functions (3.10)-(3.13) is called. Else if the proposed control algorithms in Chapter 3.2 used, functions (3.30), (3.31) used. Notice herein the desired yaw angle and time rate of vertical speed command is set to zero.

To land the quadrotor on the ground, we use either EmergencyLand or SafeLand commands. The difference is the timing of stopping propeller movements. If we send the SafeLand command to the AR.Drone, it reduces the propeller speed till its height is at the range of 0-0.1 meter and shuts down the propellers. Else if we send the EmergencyLand to the quadrotor, it directly stops the propellers and lands on the ground. The appropriate

structure for implementing the flight controller is the finite state machine (FSM) since the mode switching event is driven as shown in Fig. 5.3.

CHAPTER 6

FLIGHT TEST IMPLEMENTATION RESULTS

This section reveals the flight test implementation results obtained with different trajectories and scenarios. In Section 6.1, it is seen the backstepping-based PID controller in Fig. 3.2 gives the trajectory tracking performance as good as the exact backstepping controller in Fig. 3.1 during movement of the quadrotor.

Moreover, in Section 6.3 we compare our PID design in Fig. 3.2 with a standard existing PID controller in the literature. We show that the performance of the backstepping-designed PID controller is better in terms of decreasing both of the path following error and positional overshoot that occurs especially when the sudden position command is received. Therefore, the performance of proposed PID controller in Fig. 3.2 is verified.

6.1 Densely sampled trajectories

In this section, we compare the performance of the full nonlinear backstepping controller in Fig. 3.1 and the PID controller in Fig. 3.2 derived from it by using different desired trajectories. We measure the performance of these algorithms using the desired trajectories that have the sampling rate of 500 Hz. This means the goal point of the trajectory is generated every 2 milliseconds and desired trajectories are constructed smoothly. We present the desired path of the quadrotor and path followed by the quadrotor in following figures.

To model the system dynamics and observe system parameters offline, we use the system identification toolbox of MATLAB, which uses the recursive least squares estimation method as a base algorithm. Then, we get the ω_n as 1.26-1.22 rad/sec and ζ as

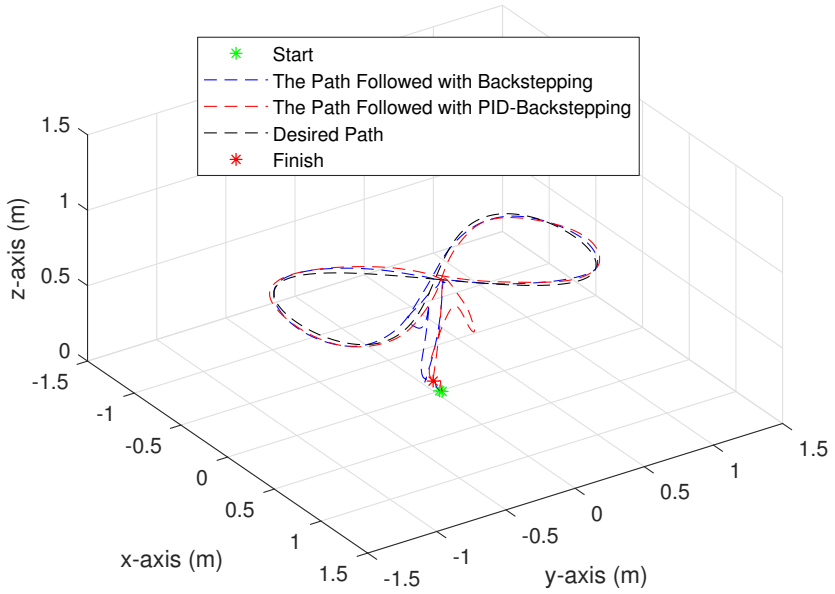

Figure 6.1. The path tracked by the UAV with backstepping controller and backstepping controller with PID tracker using 8-figure trajectory.

0.92-0.96 to control the quadrotor's pitch and roll angles respectively. These values are substituted in (3.9) and (3.28)-(3.30) to calculate the PID gains. Notice that these values highly depend on the firmware version of the AR.Drone, external disturbances and the sampling rate used while identifying the quadrotor system.

For the backstepping controller proposed in Fig. 3.1, Fig. 6.1 and Fig. 6.2 show the desired path and the path tracked by the UAV respectively when both circular and 8-figure are desired trajectories. The performance of proposed PID controller in Fig. 3.2 is seen to be almost the same as the full nonlinear backstepping controller for the outer loop. Notice that the tracking error is maintained inside acceptable bounds. On the other hand, for the backstepping controller with PID position control loop given in Fig. 3.2, these figures reveal the tracking performance of the UAV. Again, the results show that the proposed PID controller performs well in the practical implementation.

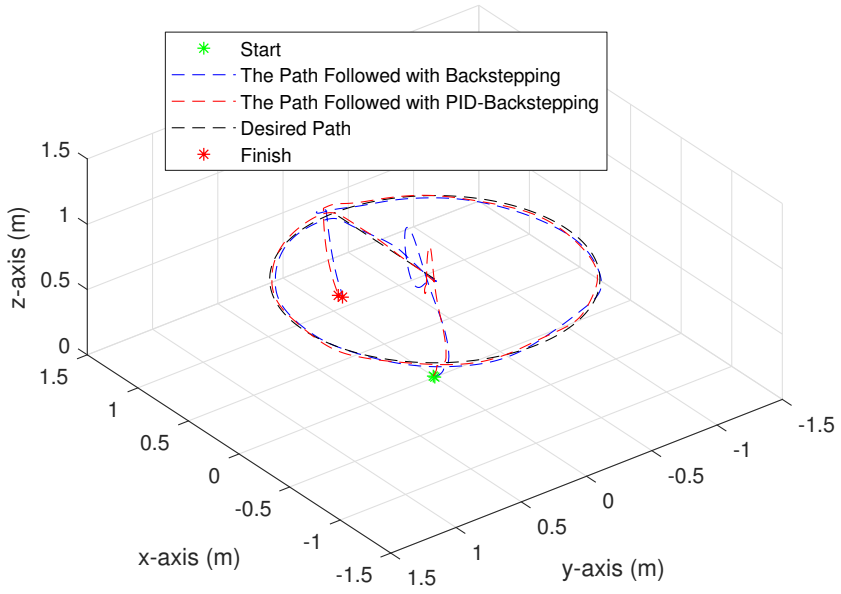

Figure 6.2. The path tracked by the UAV with backstepping controller and backstepping controller with PID tracker using circular trajectory.

To test the limits of the control algorithms, we determine the time of completing the eight-figure and circular trajectory as 4π or 12.6 seconds. Table 1 represents the percentage of error, which is calculated by $\boldsymbol{Err\%} = \frac{||\boldsymbol{\xi_d} - \boldsymbol{\xi}||}{|\boldsymbol{\xi_d}|} \times 100$ with respect to different trajectories.

In these experiments, the quadrotor enters the PATH_TRACK mode after hovering at the position (0,0,0.75). And when UAV enters this mode, the flight time is exactly 10 seconds meaning that the UAV starts to track the desired path 10 seconds after the communication between Master computer, Vicon and AR.Drone 2.0 is established. And since the period is set as 4π seconds, the $\boldsymbol{Err\%}$ is calculated from 10 seconds to 4π+10 seconds. Moreover, the data for tracked path are shifted to overlap on the desired path during calculation of the percentage of error.

Fig. 6.3 and 6.4 indicates the comparison of the proposed algorithms in Fig. 3.1 and 3.2 in terms of positional errors. Here, we illustrate the desired trajectories and experimen-

tal values of x-y-z axis positions. Results justify the validity of small angle approximation for the pitch and roll angles in the inverse kinematics case.

6.2 Behavior with sparsely sampled trajectory

Herein we show how backstepping controller derived PID position control loop eliminates positional overshoot while AR.Drone 2.0 making waypoint navigation. We get the UAV to the goal points, which are 1.4 meter and 2.8 meter away from each other. The first step is getting the quadrotor to the desired point in 3D space. Then, we examine the positional overshoot of the quadrotor.

Table 6.1. The four test cases used to generate the experimental results

Percent of Error	8-figure	Circle
Classical backstepping controller	2.3356	1.8655
Backstepping controller with PID tracker	2.33357	1.8654

Fig. 6.5 shows the start, goal, finish points and the trajectory of the quadrotor. Quadrotor hovers at the first goal point (0, 0, 0.75), then we set the desired position as (-1, -1, 0.75), which is the second target position. The last goal point is commanded as (1, 1, 0.75). As can be seen from the Fig. 12, the positional overshoot is small, which proves that the backstepping derived PID position control loop handles the overshoot well.

6.3 Comparison with existing PID controller

In this section, we compare the backstepping controller with PID position control loop performance with the work presented in [21], which proposes self-tuning fuzzy PID controller. The robustness of the proposed PID controller in Fig. 3.2 is verified. In this

scenario, it is desired for the quadrotor to follow a square trajectory constructed with goal points, which are 0.5 meters apart from each other. This is expected to result in aggressive movement of the quadrotor since the trajectory is infrequently sampled. According to results of [21], maximum error of the fuzzy logic PID controller is 0.18 meters. This upper bound of the error is highly convincing for a flight control system designer.

The infrequent sampled trajectory is simply the low rates of commands updating given in [10]. We conduct this experiment case to measure the navigational performance of our control methods under the effect of low rates of command updating. We notice with this effect, the quadrotor should have quick reactions, high stability and maneuverability capabilities to follow the desired trajectory.

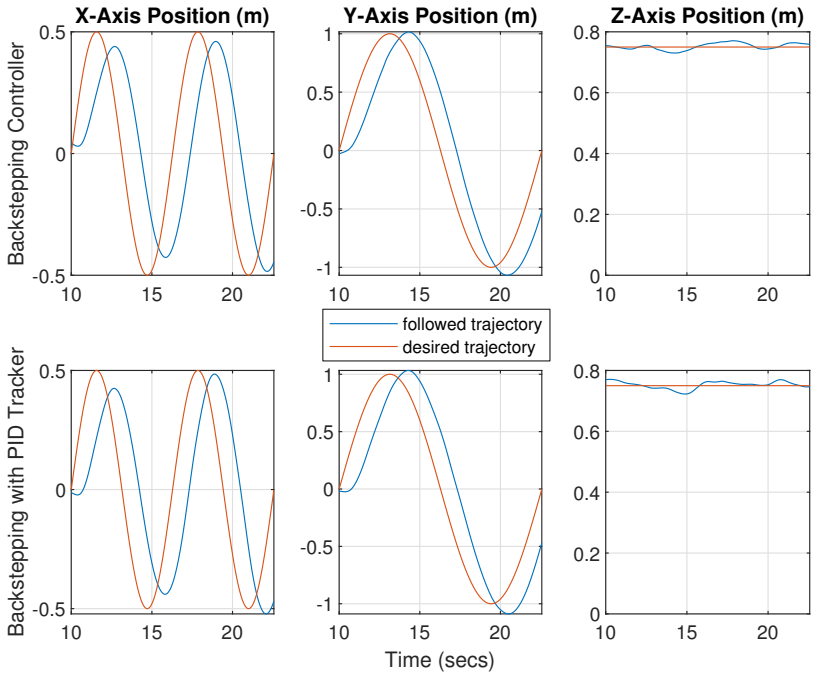

Figure 6.3. Comparison of proposed algorithms in Fig.2 and Fig. 3 with 8-figure desired trajectory.

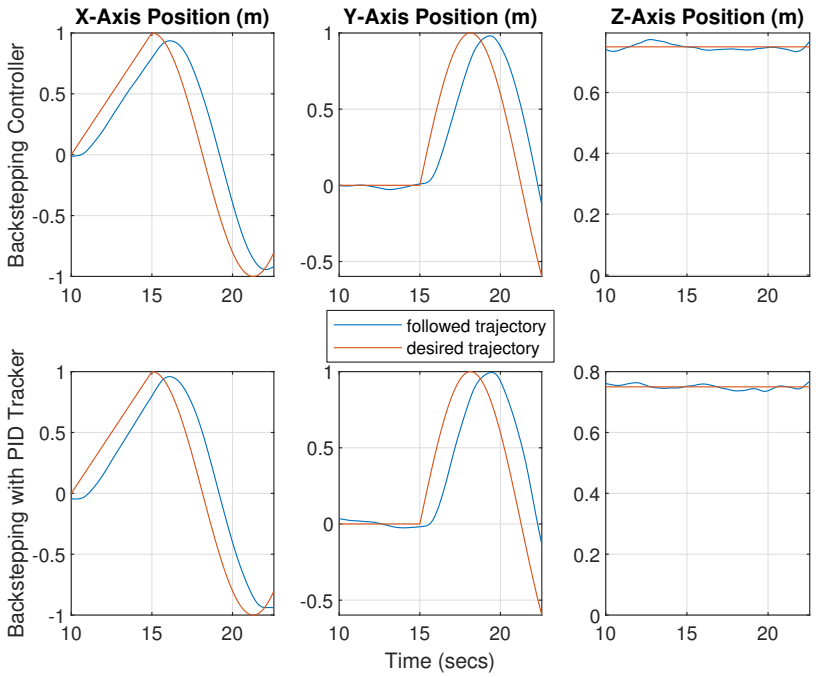

Figure 6.4. Comparison of proposed algorithms in Fig.2 and Fig. 3 with circular desired trajectory.

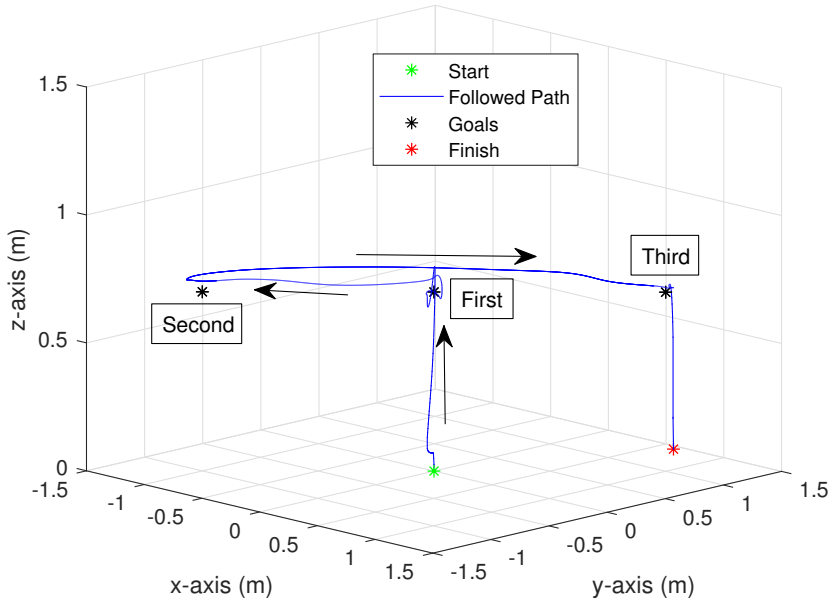

Figure 6.5. Behavior of the UAV with sudden position command.

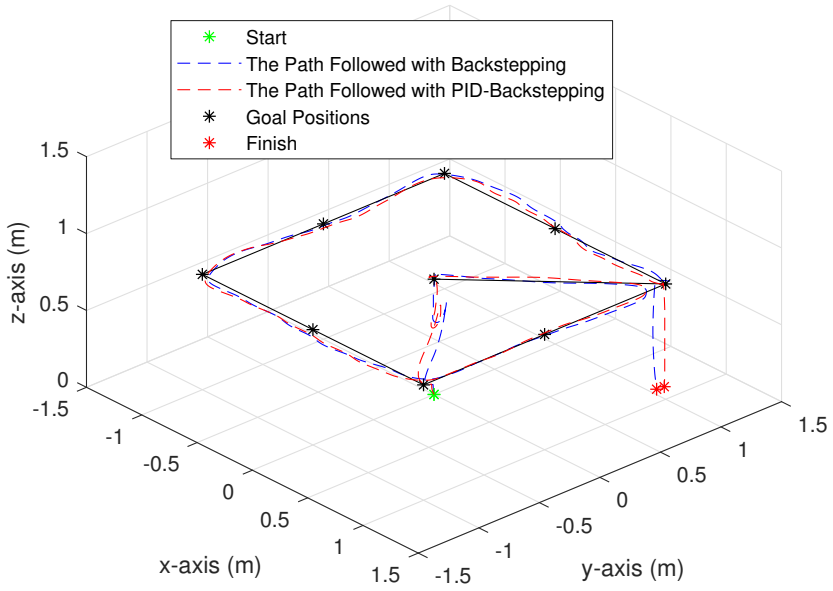

Figure 6.6. Behavior of the UAV with sparsely sampled trajectory.

Fig. 6.6 shows the path tracked by UAV with the goal points placed 0.5 meters away, which creates the square trajectory. In our case, the max error is 0.10 meters, which occurred while quadrotor is passing through the corner of square trajectory.

This test case basically proves that the PID controller derived from nonlinear backstepping design has a guaranteed performance even with infrequent sampled desired trajectory.

Finally, we record a video of the experiments described, interested reader can use the link 'https://www.youtube.com/watch?v=i4qpmmnqFso' to have visual understanding of this section. Notice that we also shot the formation control of three quadrotors using backstepping approach. These UAVs interact to make a very strong wind effect. Nevertheless, the formation control is effective and accurate. This proves the robustness of proposed control method.

6.4 Swarm Behavior

The aim of this section is to demonstrate robustness of the PID-backstepping control method as the wind produced by quadrotors affecting flight performance of each agent of the swarm. In following figure, we share the trajectories followed for multiple AR.Drone 2.0 swarm, while each of them is supposed to follow the circular trajectories. Interested reader can use 'https://www.youtube.com/watch?v=BmDtooVMc0M' to understand the swarm behavior visually.

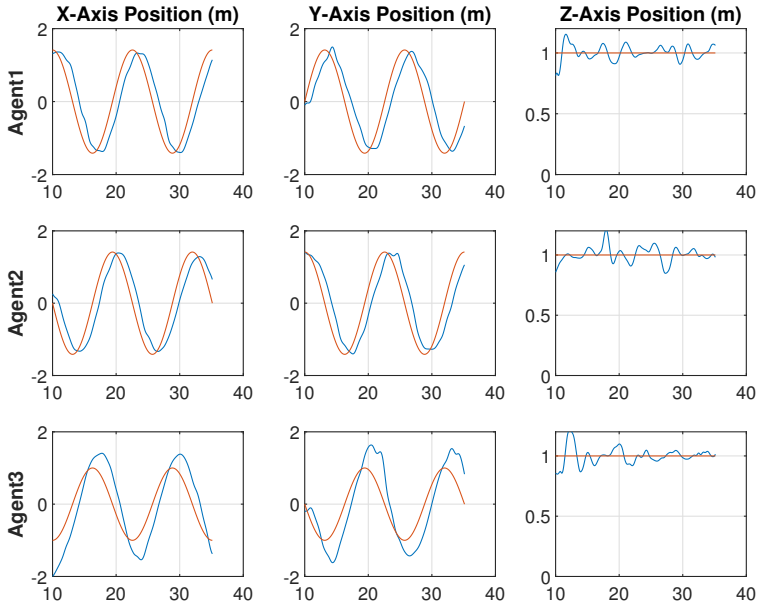

Figure 6.7. Behavior of the swarm of multiple UAVs.

CHAPTER 7

CONCLUSION

In this thesis, a method of backstepping control based on the second-order sliding variable is discussed. The proposed algorithms are validated by using Vicon Tracker, AR.Drone 2.0 and a master computer. Through rigorous experimentation, we showed that backstepping control with a PID outer position control loop provides a guaranteed performance in terms tracking error. We compare the trajectories followed by AR.Drone 2.0 quadrotor with a full nonlinear backstepping controller and our guaranteed PID controller design. Our PID tracking control loop is seen to exhibit performance similar to the full nonlinear backstepping controller, yet with a far simpler structure that is compatible with the built-in UAV attitude inner loop controller. Furthermore, the PID tracking control loop is proved to eliminate the positional overshoot problem when sudden desired position command received. Further research can be conducted to investigate how the aerodynamic state dependent disturbances affect the proposed controller algorithm in this paper.

REFERENCES

[1] Yang HC., AbouSleiman R., Sababha B., Gjioni E. et al. Implementation of an autonomous surveillance quadrotor system. *AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference* , AIAA, 2009: 2047.

[2] Fraundorfer F., Heng L., Honegger D. et al. Vision-based autonomous mapping and exploration using a quadrotor MAV. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2012: 4557 – 4564.

[3] Das A., Lewis F. and Subbarao K. Backstepping approach for controlling a quadrotor using lagrange form dynamics. *Journal of Intelligent and Robotic Systems*, 2009, 56(1-2): 127 – 151.

[4] Khatoon S., Gupta D. and Das LK. PID & LQR control for a quadrotor: Modeling and simulation. *IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, 2014: 796 – 802.

[5] Reyes-Valeria E., Enriquez-Caldera R., Camacho-Lara S. et al. LQR control for a quadrotor using unit quaternions: Modeling and simulation. *International Conference on Electronics, Communications and Computing (CONIELECOMP)*, IEEE, 2013: 172 – 178.

[6] Das A., Subbarao K. and Lewis F. Dynamic inversion with zero-dynamics stabilization for quadrotor control. *IET control theory & applications*, 2009, 3(3), 303 – 314.

[7] Jeong DY., Kang T. and Dharmayanda HR. H-infinity attitude control system design for a small-scale autonomous helicopter with nonlinear dynamics and uncertainties. *Journal of aerospace engineering*, 2011, 25(4): 501 – 518.

[8] Madani T. and Benallegue A. Backstepping control for a quadrotor helicopter. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2006: 3255 – 3260.

[9] Munoz F., Espinoza ES., Gonzalez-Hernandez I. et al. Robust Trajectory Tracking for Unmanned Aircraft Systems using a Nonsingular Terminal Modified Super-Twisting Sliding Mode Controller. *Journal of Intelligent & Robotic Systems*, 2019, 93(1-2): 1 – 8.

[10] Santos MC., Rosales CD., Sarapura JA.et al. An Adaptive Dynamic Controller for Quadrotor to Perform Trajectory Tracking Tasks.*Journal of Intelligent & Robotic Systems*, 2018, 93(1-2): 5 – 16.

[11] Rosaldo-Serrano MA., Santiaguillo-Salinas J., Aranda-Bricaire E. Observer-based time-varying backstepping control for a quadrotor multi-agent system. *Journal of Intelligent & Robotic Systems*, 2019, 93(1-2): 135 – 150.

[12] Yang CD. and Liu WH. Nonlinear H∞ Decoupling Hover Control with Parameter Uncertainties. *Journal of Aeronautics, Astronautics and Aviation. Series A*, 2008, 40(4): 225 – 236.

[13] Hoffmann G., Huang H. and Tomlin C et al. Quadrotor helicopter flight dynamics and control: Theory and experiment. *AIAA Guidance, Navigation and Control Conference and Exhibit*, AIAA, 2007: 6461.

[14] Mellinger D., Michael N. and Kumar V. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *The International Journal of Robotics Research*, 2012, 31(5): 664 – 674.

[15] Castillo P., Lozano R. and Dzul AE. *Modelling and control of mini-flying machines* Springer Science and Business Media, 2006: 39 – 59.

[16] Dogan A., Sato S. and Blake W. Flight control and simulation for aerial refueling. *AIAA guidance, navigation, and control conference and exhibit*, AIAA, 2005: 6264.

[17] Stevens BL., Lewis FL. and Johnson EN. *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

[18] Zuo Z. Trajectory tracking control design with command-filtered compensation for a quadrotor. *IET control theory applications*, 2011, 4(11): 2343 – 2355.

[19] Cowling ID., Yakimenko OA. and Whidborne JF et al. A prototype of an autonomous controller for a quadrotor UAV. *Europian Control Conference (ECC)*, IEEE, 2007: 4001 – 4008.

[20] Mokhtari A., Benallegue A. and Daachi B. Robust feedback linearization and GH/sub/spl infin//controller for a quadrotor unmanned aerial vehicle. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2005: 1198 – 1203.

[21] Demir BE., Bayir R. and Duran F. Real-time trajectory tracking of an unmanned aerial vehicle using a self-tuning fuzzy proportional integral derivative controller. *International Journal of Micro Air Vehicles*, 2016, 8(4): 252 – 268.

[22] Aguilar-Ibanez, C., Sira-Ramirez, H., Suarez-Castanon, M. S., et al. The trajectory tracking problem for an unmanned four-rotor system: flatness-based approach *International Journal of Control*, 2012, 85(1): 69 – 77.

[23] Aguilar-Ibanez C., Sira-Ramirez H., Suarez-Castanon M. S., Garrido R. Robust Trajectory-Tracking Control of a PVTOL under Crosswind. *Asian Journal of Control*, 2019, 21(3):1293 – 306.

[24] Sira-Ramirez H. On the linear control of the quad-rotor system. *IEEE American Control Conference*, IEEE, 2011: 3178 – 3183.

[25] Santos MC., Rosales CD., Sarapura JA. et al. An adaptive dynamic controller for quadrotor to perform trajectory tracking tasks. *Journal of Intelligent & Robotic Systems*, 2019, 93(1-2): 5 – 16.

[26] Kushleyev, A., Mellinger, D., Powers, C. and Kumar, V. (2013). Towards a swarm of agile micro quadrotors. Autonomous Robots, 35(4), 287-300.

[27] Saska, M., Vakula, J. and Přeućil, L. (2014, May). Swarms of micro aerial vehicles stabilized under a visual relative localization. In 2014 IEEE International Conference on Robotics and Automation (ICRA) (pp. 3570-3575). IEEE.

[28] Bourquardez, O., Mahony, R., Guenard, N., Chaumette, F., Hamel, T. and Eck, L. (2009). Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle. IEEE Transactions on Robotics, 25(3), 743-749.

[29] Recchiuto, C. T., Sgorbissa, A. and Zaccaria, R. (2016). Visual feedback with multiple cameras in a UAVs Human–Swarm Interface. Robotics and Autonomous Systems, 80, 43-54.

[30] Stevens, B. L., Lewis, F. L. and Johnson, E. N. (2015). Aircraft control and simulation: dynamics, controls design, and autonomous systems. John Wiley Sons.

[31] Slotine, J. J. E. and Li, W. (1991). Applied nonlinear control (Vol. 199, No. 1). Englewood Cliffs, NJ: Prentice hall.

[32] Maza, I., Ollero, A., Casado, E., & Scarlatti, D. (2015). Classification of multi-UAV Architectures. *Handbook of unmanned aerial vehicles*, 953-975.

[33] Lee, G., & Chwa, D. (2018). Decentralized behavior-based formation control of multiple robots considering obstacle avoidance. *Inteq51gent Service Robotics*, *11*, 127-138.

[34] Zhou, D., Wang, Z., & Schwager, M. (2018). Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures. *IEEE Transactions on Robotics*, *34*, 916-923.

[35] Duan, H., & Qiu, H. (2018). Unmanned aerial vehicle distributed formation rotation control inspired by leader-follower reciprocation of migrant birds. *IEEE Access*, *6*, 23431-23443.

[36] Xi, J., Cai, N., & Zhong, Y. (2010). Consensus problems for high-order linear time-invariant swarm systems. *Physica A: Statistical Mechanics and its Applications*, *389*, 5619-5627.

[37] Dong, X., Zhou, Y., Ren, Z., & Zhong, Y. (2016). Time-varying formation control for unmanned aerial vehicles with switching interaction topologies. *Control Engineering Practice*, *46*, 26-36.

[38] Carvalho, J. F., Pequito, S., Aguiar, A. P., Kar, S., & Johansson, K. H. (2017). Composability and controllability of structural linear time-invariant systems: Distributed verification. *Automatica*, *78*, 123-134.

[39] Ma, C. Q., & Zhang, J. F. (2010). Necessary and sufficient conditions for consensusability of linear multi-agent systems. *IEEE Transactions on Automatic Control*, *55*, 1263-1268.

[40] Rui, W., Xiwang, D., Qingdong, L., Qilun, Z., & Zhang, R. (2015, July). Adaptive time-varying formation control for high-order LTI multi-agent systems. In *2015 34th Chinese Control Conference (CCC)* (pp. 6998-7003). IEEE.

[41] Baghbani, F., Akbarzadeh-T, M. R., & Sistani, M. B. N. (2017, June). Cooperative adaptive fuzzy tracking control for a class of nonlinear multi-agent systems. In *2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Inteq51gent Systems (IFSA-SCIS)* (pp. 1-6). IEEE.

APPENDIX A

NOMENCLATURE

$\boldsymbol{\xi}$: position vector in Earth frame

$\boldsymbol{U}$: velocity vector in Body frame

$\boldsymbol{\xi_d}$: desired position vector in Earth frame

x_d : desired position in x-axis in Earth frame

y_d : desired position in y-axis in Earth frame

z_d : desired position in z-axis in Earth frame

x : current position in x-axis in Earth frame

y : current position in y-axis in Earth frame

z : current position in z-axis in Earth frame

x_B : desired position in x-axis in Body frame

y_B : desired position in y-axis in Body frame

z_B : desired position in D-axis in Body frame

$\boldsymbol{R}$: rotation matrix from Earth to Body frame

$\boldsymbol{F}$: force vector in the Earth frame

$\boldsymbol{F_d}$: desired force vector in Earth frame

$\boldsymbol{F_g}$: gravitational force vector in Earth frame

g : gravitational acceleration

μ : thrust produced in Body frame z-axis

μ_d : desired total thrust in Body frame z-axis

w_B : angular velocity matrix in Body frame

u : forward velocity in Body frame

v : sideward velocity in Body frame

w : vertical velocity in Body frame

p : roll rate

q : pitch rate

r : yaw rate

e : error term

φ : roll angle

θ : pitch angle

ψ : yaw angle

φ_d : desired roll angle

θ_d : desired pitch angle

ψ_d : desired yaw angle

$\boldsymbol{I_B}$: inertia matrix

m : mass of the quadrotor

l : lever length

t : thrust factor

$\boldsymbol{\tau_B}$: torque vector

Ω : angular velocity of rotor

BIOGRAPHICAL STATEMENT

Yusuf KARTAL studied Electrical and Electronics Engineering at Bilkent University (Turkey) obtaining the degree of Bachelor's in Science in 2015. He is a Master in Science student at Aerospace Engineering Department in University of Texas at Arlington. His main interests include algorithms for multi-robot systems, nonlinear control, optimal path planning, unmanned aerial vehicles, machine learning and distributed control.

E-mail: yusuf.kartal@mavs.uta.edu.