# Stability Analysis and Formally Guaranteed Tracking Control of Quadrotors

by

Haocheng Chang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Applied Mathematics

Waterloo, Ontario, Canada, 2024

## Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

## Statement of Contributions

This thesis contains research works in submission to a scientific journal. The author lists for the paper and contributions made by myself and various coauthors are as follows:

**Reach-Avoid Control Synthesis for a Quadrotor UAV with Safety Guarantees**

Mohamed Serry[1], Haocheng Chang[1], and Jun Liu[1]
[1]Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, Canada

M. Serry contributed to the derivation of analytical results, the design of the computer code, and the writing and editing of manuscipt. H. Chang contributed to the derivation of analytical results, running of simulations, and the writing and editing of manuscipt. J. Liu contributed to the derivation of analytical results and the editing of the manuscript. The first two authors contributed equally to the work.

**Abstract**

Reach-avoid tasks are among the most common challenges in autonomous aerial vehicle (UAV) applications. Despite the significant progress made in the research of aerial vehicle control during recent decades, the task of efficiently generating feasible trajectories amidst complex surroundings while ensuring formal safety guarantees during trajectory tracking remains an ongoing challenge. In response to this challenge, we propose a comprehensive control framework specifically for quadrotor UAVs reach-avoid tasks with robust formal safety guarantees. Our approach integrates geometric control theory with advanced trajectory generation techniques, enabling the consideration of tracking errors during the trajectory planning phase.

Our framework leverages the well-established geometric tracking controller, analyzing its stability to demonstrate the local exponential stability of tracking error dynamics with any positive control gains. Additionally, we derive precise and tight uniform bounds for tracking errors, ensuring guaranteed safety of the system's behavior under certain conditions. In the trajectory generation phase, our approach incorporates these bounds into the planning process, employing sophisticated sampling-based planning algorithms and safe hyper-rectangular set computations to define robust safe tubes within the environment. These safe tubes serve as corridors within which trajectories can be constructed, with piecewise continuous Bezier curves employed to ensure smooth and continuous motion. Furthermore, to enhance the performance and adaptability of our framework, we formulate an optimization problem aimed at determining optimal control gains, thereby enabling the quadrotor UAV to navigate with optimal safety guarantees.

To demonstrate the validation of the proposed framework, we conduct comprehensive numerical simulations as well as real experiments, demonstrating its ability to successfully plan and execute reach-avoid maneuvers while maintaining a high degree of safety and precision. Through these simulations, we illustrate the practical effectiveness and versatility of our framework in addressing real-world challenges encountered in UAV navigation and trajectory planning.

**Acknowledgments**

Many thanks to my super knowledgeable, inspiring and supportive supervisor, Jun Liu. Your spirit of exploring new knowledge will be my life-time wealth. I would like to thank my co-author, Mohamed Serry, for the significant contributions to this thesis. Your collaboration and expertise were vital to the successful completion of this research project. Thank you also to all of the members of the Hybrid Systems Lab for all the insightful conversation and interesting presentations. Lastly, to Giang Tran and Kirsten Morris, I am very grateful for the time you dedicated and the comments you provided as members of my defence committee.

## Dedication

This thesis is dedicated to my family and Anna, who have always supported me.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

Over the years, many traditional controllers like Proportional-Integral-Derivative (PID) [20] and Linear Quadratic Regulator (LQR) [10] have been widely applied to quadrotor systems. Despite their acceptable performance in many scenarios, these conventional control strategies exhibit notable limitations. Specifically, they neither have a systematic way of tuning parameters nor have formal safety guarantees. As a result, it will take considerable effort to tune for optimal parameters. The controller may exhibit unpredictable or unstable behavior under certain operating conditions as well. Furthermore, most control algorithms use the linearized model [1], which not only ignores the geometric structure of quadrotor rotation but also cause singularities [28]. In this case, the controller does not have the optimal performance, and will fail in certain scenarios. On the other hand, while it is true that path planning can be decoupled from quadrotor dynamics due to the system's differential flatness property, the needs for practical constraints are not yet fullfilled. Factors such as bounded control inputs and the physical dimensions of the quadrotor must still be accounted for to ensure the feasibility and safety of planned trajectories.

When facing these challenges, the geometric controller in [18] with Lyapunov stability guarantee presents a promising solution for ensuring the safety and robustness of quadrotor control systems. By exploiting the geometric structure of the system, geometric controllers offer the potential to achieve stable and agile performance [8]. Moreover, Lyapunov stability analysis provides a rigorous framework for tuning the control gains. Another advantage of such controllers with safety guarantee is their

ability to construct invariant sets within the state space, effectively identifying safe regions of operation for the quadrotor. These invariant sets serve as "safe tunnels", guiding the quadrotor along trajectories that guarantee stability and avoid dangerous regions of the state space. Hence, such geometric controllers offer a principled approach to ensuring the reliability of quadrotor systems in diverse operational environments.

## 1.2 Background

Unmanned aerial vehicles (UAVs) were invented and developed in the 1900s, when aircraft needed to be operated without a crew for complex and dangerous military tasks, such as dropping bombs. In the following centuries, UAV technology underwent substantial evolution, driven by advancements in electronic systems. They soon became available for civilian and commercial aviation activities. Among all UAV classifications, the quadrotor design stands out for its simplified manufacturability. Over the past decades, with the increasing use of quadrotors, extensive research and development have been conducted to enable the performance of complex tasks robustly. Due to their low cost and agility, quadrotors are becoming relevant in various fields of application (e.g., rescue, firefighting, surveillance), which necessitates the ability to handle complex tasks (e.g., navigating to a goal region while avoiding obstacles or remaining within a specific zone for a designated period).

Quadrotors, designed with a cross-shaped structure featuring four pairs of rotors and propellers as control inputs, have been a widely researched topic in control theory. The simple structure and design of the quadrotor is a double-edged sword. It has gained both popularity and novelty, but, as a tradeoff, it is challenging to control due to underactuation. The quadrotor model has six degrees of freedom (DOFs): three in attitude and three in translation, but only four control inputs. The thrust (force) of a quadrotor is constrained to the z-axis within the body frame, resulting in a coupling between translation and attitude control. Fortunately, the quadrotor system was proven to be differentially flat in [21], which simplifies the control design process by enabling the direct calculation of control inputs necessary to achieve desired trajectories or maneuvers, thus facilitating more efficient and effective control strategies.

Another property of the quadrotor model is that it can be considered a rigid body due to the way the rotors are attached to the arms. Since the quadrotor attitude system is fully actuated, the attitude control of a rigid body, which has already been

developed in [2], can be directly applied. Being a rigid body means that we can consider its attitude as elements within the Special Orthogonal Group $\mathsf{SO}(3)$, so it is natural to adopt concepts from differential geometry to develop a control law for attitude. The authors of [18] were the first to use differential geometry to develop a singularity-free controller with Lyapunov stability guarantees. Later, [29, 19, 4, 8, 16] expanded upon this groundwork by extending the system to a more complex configuration where the quadrotor is connected to a mass payload. Notably, they demonstrated that the extended system retains the property of differential flatness while also developing a controller with Lyapunov stability guarantees. Such research illustrates quadrotor applications in scenarios involving the transportation of mobile cargo.

Before the research work in [21, 6], quadrotor dynamics had to be considered when performing path planning. However, the differential flatness property of the quadrotor system illustrates the possibility of separating planning and tracking. Path planning itself includes trajectory generation and trajectory optimization. Typically, trajectory generation is performed by first generating waypoints and then applying a smoothing algorithm. Common and mature waypoint generation methods include Dijkstra's algorithm [5], A* [9], and rapidly-exploring random trees (RRT*) [11]. Once a sequence of waypoints that satisfies some given specification, such as ego reach-avoid, is generated, the path can be smoothed using curves like Clothoid curves, polynomials, and Bezier curves. Constraints such as velocity and acceleration bounds can be considered during the smoothing step. Finally, since trajectory generation provides various possible smooth paths, optimization can be applied to choose the best trajectory according to a specific objective function. For instance, methods to achieve minimum jerk and minimum snap are discussed in [14] and [21]. Graph-Search-Based Algorithms (GSBAs) for achieving minimum path length are discussed in [3].

However, while differential flatness simplifies path planning by abstracting away the quadrotor dynamics, it does not ensure safe tracking performance. Motivated by this, we integrate this trajectory generation method with Lyapunov stability analysis as described in [18]. This combination results in an algorithm that provides safety guarantees for reach-avoid specifications. Specifically, we derive an upper bound on the position error from a Lyapunov function and use this upper bound to define the radius of a safe tube along the generated trajectory, ensuring that the quadrotor remains within this tube during tracking.

## 1.3 Problem Statement

Given an operating domain, a target set, and an unsafe set, assume the quadrotor dynamics are known. Design a synthesis procedure that generates a smooth trajectory and a tracking controller. The trajectory should start outside the unsafe set, remain outside the unsafe set, and eventually reach the target set. The tracking controller should constrain the tracking errors to be within certain bounds, ensuring the quadrotor operates in a "safe tunnel" while successfully executing the reach-avoid task.

### 1.3.1 Main Contributions

First, we correctly formulate the differential flatness of the quadrotor system by addressing the confusion between the body coordinate and the world coordinate. This is accomplished by combining the proofs from [21] and [6], as detailed in Section 2.5 and Section 4.2. Next, we properly formulate the Lyapunov stability analysis of the geometric tracking controller by correcting a cubic term in the Lyapunov function. Specifically, we derive a position error bound based on the stability analysis in Subsection 4.2.4. Optimal control gains are selected to minimize the position error bound, as described in Section 4.4. This optimal error bound is then integrated into the trajectory generation procedure to create a framework that ensures the safety of the quadrotor during reach-avoid tasks.

## 1.4 Overview

The structure of this thesis is summarized as follows. In Chapter 2, we first derive the ordinary differential equations (ODEs) that describe quadrotor dynamics from the structure of the quadrotor and the laws of physics. We then define the errors for position, velocity, attitude, and angular velocity, enabling the formulation of a PD feedback controller based on these errors. The quadrotor dynamics are then transformed into error dynamics for Lyapunov analysis. Finally, a geometric controller is defined based on these errors.

Chapter 3 introduces a trajectory generation method based on RRT and Bezier curves. A waypoint trajectory is first generated using RRT, a sampling-based method.

The discrete trajectory is then smoothed using piecewise Bezier curves. Under certain constraints at waypoints, this method will generate a class $C^4$ trajectory, which fulfills the requirements of quadrotor tracking tasks.

Chapter 4 provides the full proof of Lyapunov stability for the quadrotor error dynamics under the geometric controller. The proof is divided into attitude stability and translation stability. The Lyapunov function, treated as a quadratic function of the errors, is used to derive the error bounds. These error bounds are then utilized for formally guaranteed safe planning. Finally, an autotuning algorithm is proposed based on the Lyapunov function, allowing the control gains in the controller to be automatically selected to ensure the largest safe funnel.

Finally, Chapter 5 provides documentation-like instructions on how to conduct Webots simulations and real experiments using the Optitrack motion capture (mocap) system. The components of the mocap system and ROS are introduced to ensure the accuracy of the experimental performance. Various methods of conducting experiments are then described for future reference.

In Appendix A, various methods for parameterizing the $\mathsf{SO}(3)$ group is introduced as tools for designing the controller and deriving error bounds. Several properties of $\mathsf{SO}(3)$ and the corresponding proofs are also given for reference.

# Chapter 2

# Quadrotor Modelling and Control

In this chapter, we introduce the quadrotor system and the geometric controller. First, we start by describing the system with ODEs. Then, the errors for position, velocity, attitude, and angular velocity are defined. The attitude error is defined on the special orthogonal group $\mathsf{SO}(3)$, and hence is not an intuitive definition. A PID-style geometric controller is then designed based on the errors. With such a controller, the quadrotor dynamics is transformed into an error dynamics which is easier to analyze. Finally, an important property of the quadrotor system called differential flatness is introduced and proven. It will later be used to decouple the planning and tracking.

## 2.1 Coordinate System

There are two commonly used definitions for the coordinate systems of quadrotors. In both definitions, the $z$-axis points upwards. In one, the $x$-axis and $y$-axis align with two of the arms, while in the other, two axes are angled 45° away from the arms, as illustrated in Figure 2.1. The four forces generated by the rotors are denoted as $f_1, f_2, f_3, f_4$. A linear transformation exists between these four forces and the control inputs $f, \tau_x, \tau_y, \tau_z$, where $f$ represents the total thrust aligned with the $z$-axis in the body frame, and $\tau_x, \tau_y, \tau_z$ represent torques that induce rotation about the center of mass of the quadrotor. The linear transformation of coordinate system 1 (left of

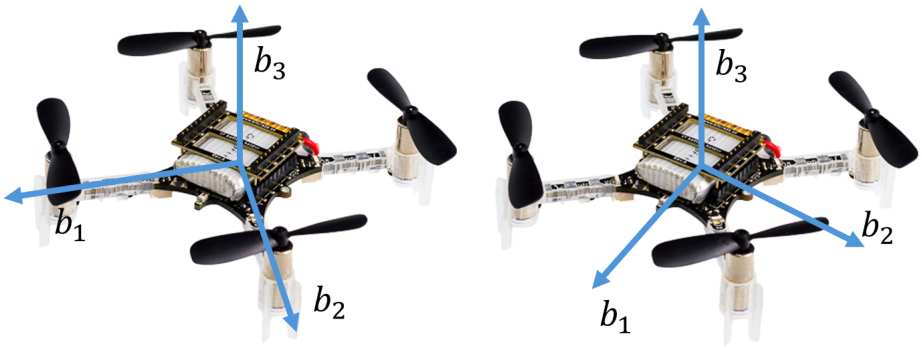


Figure 2.1: Two different body coordinate system.

Figure 2.1) is

$$\begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ d & 0 & -d & 0 \\ -c_{\tau f} & c_{\tau f} & -c_{\tau f} & c_{\tau f} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}. \tag{2.1}$$

The inverse is

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & 0 & \frac{1}{2d} & -\frac{1}{4c_{\tau f}} \\ \frac{1}{4} & -\frac{1}{2d} & 0 & \frac{1}{4c_{\tau f}} \\ \frac{1}{4} & 0 & -\frac{1}{2d} & -\frac{1}{4c_{\tau f}} \\ \frac{1}{4} & \frac{1}{2d} & 0 & \frac{1}{4c_{\tau f}} \end{bmatrix} \begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}. \tag{2.2}$$

The linear transformation of coordinate system 2 (right of Figure 2.1) is

$$\begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\frac{d}{\sqrt{2}} & -\frac{d}{\sqrt{2}} & \frac{d}{\sqrt{2}} & \frac{d}{\sqrt{2}} \\ \frac{d}{\sqrt{2}} & -\frac{d}{\sqrt{2}} & -\frac{d}{\sqrt{2}} & \frac{d}{\sqrt{2}} \\ c_{\tau f} & -c_{\tau f} & c_{\tau f} & -c_{\tau f} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}. \tag{2.3}$$

The inverse is

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & -\frac{\sqrt{2}}{4d} & \frac{\sqrt{2}}{4d} & \frac{1}{4c_{\tau f}} \\ \frac{1}{4} & -\frac{\sqrt{2}}{4d} & -\frac{\sqrt{2}}{4d} & -\frac{1}{4c_{\tau f}} \\ \frac{1}{4} & \frac{\sqrt{2}}{4d} & -\frac{\sqrt{2}}{4d} & \frac{1}{4c_{\tau f}} \\ \frac{1}{4} & \frac{\sqrt{2}}{4d} & \frac{\sqrt{2}}{4d} & -\frac{1}{4c_{\tau f}} \end{bmatrix} \begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}. \tag{2.4}$$

We adopt coordinate system 1 as the body frame in the first four chapters as well as in MATLAB simulations. However, coordinate system 2 will be used in Chapter 5 due to the protocol of the mocap system. We represent the body frame as $\{b_1, b_2, b_3\}$, with three orthogonal unit vectors attached to the quadrotor body. The world frame is denoted as $\{e_1, e_2, e_3\}$, consisting of three constant orthogonal unit vectors that maintain consistent position and direction at all times. Both frames in terms of coordinate system 2 are depicted in Figure 2.2.

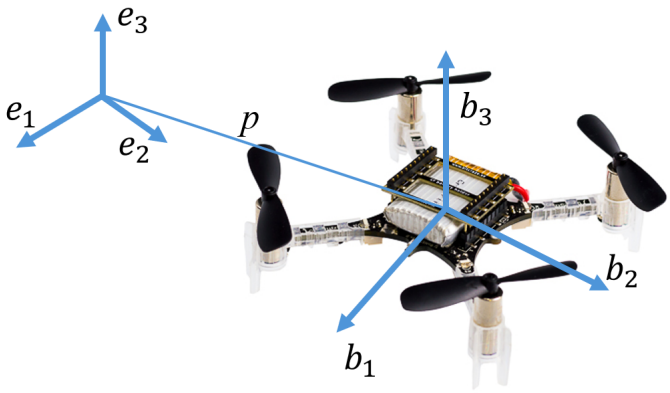


Figure 2.2: The world frame $\{e_1, e_2, e_3\}$ and the body frame $\{b_1, b_2, b_3\}$.

## 2.2 Dynamics

In this section, we introduce a set of equations that govern the motion of the quadrotor. A detailed derivation is then provided for a better understanding of the equations. The quadrotor dynamics can be described by a set of ODEs as follows:

$$\dot{p} = v, \tag{2.5}$$
$$\dot{v} = -ge_3 + m^{-1} f R e_3, \tag{2.6}$$
$$\dot{R} = R\hat{\omega}, \tag{2.7}$$
$$\dot{\omega} = -J^{-1}\omega \times (J\omega) + J^{-1}\tau, \tag{2.8}$$

where $e_3 = [0; 0; 1] \in \mathbb{R}^3$, and the *hat map* $\hat{\cdot} : \mathbb{R}^3 \to \mathrm{SO}(3)$ is defined by

$$\widehat{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}} = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix},$$

$m$ is the mass, $g$ is the gravitational constant, $J$ is the moment of inertia, $p = (x, y, z) \in \mathbb{R}^3$, $v = (v_x, v_y, v_z) \in \mathbb{R}^3$, $R = (b_1, b_2, b_3) \in \{\mathbb{R}^{3\times3} \mid R^T R = I, \det(R) = 1\}$, $\omega = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$, $\tau = (\tau_x, \tau_y, \tau_z) \in \mathbb{R}^3$. In the definition of $R$, $b_1, b_2, b_3$ are axes in the body coordinate, as shown in Figure 2.2. $R$ is also referred to as the "rotation matrix" in Subsection A.1.1. Since we use the rotation matrix $R$ to denote the attitude of the quadrotor, our state space is 18-dimensional. Some materials describe the quadrotor system as a 12-dimensional system, since $R$ can be converted to Euler angles $(\phi, \theta, \psi)$ (introduced in Subsection A.1.2) using the formula (A.2). The 12-dimensional state is $[p_x, p_y, p_z, \phi, \theta, \psi, \dot{p}_x, \dot{p}_y, \dot{p}_z, \omega_x, \omega_y, \omega_z] \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3$. We use the 18-dimensional state space to avoid singularities due to gimbal lock.

For a better understanding of the quadrotor dynamics, we explain the translation dynamics as follows. Eq. (2.5) represents a simple first-order derivative. Eq. (2.6) highlights the presence of underactuation. The quadrotor is influenced not only by gravity but also by the control force $f$. However, the quadrotor's structural constraints limit the direction of $f$ to be aligned with the $z$-axis in the body frame (i.e., $b_3 = Re_3$). Thus, the force generated by the quadrotor can be expressed as $fRe_3$. Consequently, the quadrotor system has four control inputs $(f, \tau) \in \mathbb{R} \times \mathbb{R}^3$, but six degrees of freedom (DOFs). This discrepancy between the number of control inputs and the number of DOFs makes the quadrotor system underactuated.

Attitude dynamics, on the other hand, are governed by (2.7) and (2.8). Eq. (2.7) is derived from the chain rule and a physics principle known as the "tangential velocity formula." From the definition of the rotation matrix, $R$, which is an orthogonal matrix, we have

$$RR^T = I_{3\times3}. \tag{2.9}$$

Taking the derivative with respect to time, we obtain

$$\dot{R}R^T + R\dot{R}^T = \mathbf{0}_{3\times3}. \tag{2.10}$$

Let $S = \dot{R}R^T$. Then we have

$$S + S^T = \mathbf{0}_{3\times3}. \tag{2.11}$$

Thus, $S$ is skew-symmetric. From the tangential velocity formula, the relationship

between the speed of a point in the world frame $p^w$ and the same point in the body frame $p^b$ is

$$\dot{p}^w = \omega \times p^w = \omega \times Rp^b = \hat{\omega}Rp^b. \tag{2.12}$$

An alternative expression for $\dot{p}^w$ can be derived. Assume the point in the body frame is fixed. Then it also holds that

$$\dot{p}^w = \dot{R}p^b = SRp^b. \tag{2.13}$$

By comparing (2.12) and (2.13), we find $S = \hat{\omega}$. Hence, $\dot{R} = \hat{\omega}R$. The angular velocity $\omega$ can be transformed to the body frame by Lemma A.2.5 as follows:

$$\hat{\omega} = (R\omega^b)^\wedge = R\hat{\omega}^b R^T. \tag{2.14}$$

Substituting (2.14) into (2.12), we get

$$\dot{p}^w = R\hat{\omega}^b p^b. \tag{2.15}$$

Therefore, $\dot{R} = R\hat{\omega}^b$, which proves that (2.7) is correct.

Eq. (2.8) describes how the torque affects the rotation of the quadrotor. The torque generated by the quadrotor's rotation is given by the cross product of the angular velocity and the angular momentum, i.e., $\tau_{in} = \omega \times (J\omega)$. According to Newton's second law for rotation, we have $\tau_{out} - \tau_{in} = J\dot{\omega}$. Assuming $\tau_{out} = \tau$, we obtain

$$\tau - \omega \times (J\omega) = J\dot{\omega}. \tag{2.16}$$

Left-multiplying both sides by $J^{-1}$, we derive (2.8).

## 2.3 Error Definitions

In this section, we introduce the error definitions for position, velocity, attitude, and angular velocity, along with explanations for these definitions. These errors will later be used as feedback for the stabilizing controller.

Define the errors as follows:

$$e_p = p - p_d, \tag{2.17}$$
$$e_v = v - v_d, \tag{2.18}$$
$$e_R = \frac{1}{2}(R_d^T R - R^T R_d)^\vee, \tag{2.19}$$
$$e_\omega = \omega - R^T R_d \omega_d, \tag{2.20}$$

where $e_p, e_v, e_R, e_\omega$ are the errors of the position, velocity, attitude, and attitude velocity, respectively. the *vee map* $(\cdot)^\vee$ is the inverse map of the *hat map* $\hat{\cdot}$, i.e. $(\cdot)^\vee : \mathsf{SO(3)} \to \mathbb{R}^3$ is defined by $\begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}^\vee = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$.

The least intuitive definition is the attitude error $e_R$. To grasp this definition, we view the rotation matrices $R$ and $R_d$ as mappings between different frames. The rotation matrix $R$ maps from the body frame to the world frame, while $R_d$ maps from the desired body frame to the world frame. Since $R_d$ satisfies $R_d^T = R_d^{-1}$, we have $R_d^T R = R_d^{-1} R$, which represents a mapping from the body frame to the desired body frame using the world frame as an intermediary. If we consider the initial frame as the body frame instead of the world frame, then $R_d^T R$ represents a rotation matrix relative to the body frame. We denote this as $R_{rel} = R_d^T R$, which is a frame relative to the body frame.

To understand the appearance of the vee map in the definition of $e_R$, we first explore the axis-angle representation of SO(3). According to (A.3), we have

$$\begin{aligned} R_{ref}(3,2) - R_{ref}(2,3) &= 2a_1 \sin\theta, \\ R_{ref}(1,3) - R_{ref}(3,1) &= 2a_2 \sin\theta, \\ R_{ref}(2,1) - R_{ref}(1,2) &= 2a_3 \sin\theta. \end{aligned} \tag{2.21}$$

Therefore, if $(\boldsymbol{a}, \theta)$ is the axis-angle representation of $R_d^T R$, we have

$$e_R = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \sin\theta. \tag{2.22}$$

Taking the norm on both sides of (2.22), it is evident that $\|e_R\| = |\sin\theta|$. Thus, the definition of $e_R$ measures the angle $\theta$ in the axis-angle representation. Note that $\|e_R\| \leq 1$.

## 2.4 Geometric Control

Geometric control treats attitude error on a smooth manifold, rather than merely using the difference in Euler angles. This approach exemplifies the successful application of differential geometry to control problems. In this section, we introduce the geometric tracking controller proposed in [18]. We also present the error system that results from incorporating this controller into the system described by (2.5)–(2.8).

### 2.4.1 Geometric Controller

A geometric controller is defined as:

$$\begin{cases} f = F_d \cdot Re_3, \\ \tau = -k_R e_R - k_\omega e_\omega + \omega \times J\omega - J\hat{\omega} R^T R_d \omega_d + J R^T R_d \dot{\omega}_d, \end{cases} \tag{2.23}$$

where $k_p, k_v, k_R, k_\omega \in \mathbb{R}$ are the control gains, $F_d = -k_p e_p - k_v e_v + mge_3 + m\ddot{p}_d$, $\ddot{p}_d$ is the desired acceleration which is obtained from planning. Apparently, $(f, \tau) \in \mathbb{R} \times \mathbb{R}^3$ is a PD type controller since $k_p e_p$ and $k_R e_R$ are proportional error and $k_v e_v$ and $k_\omega e_\omega$ are errors of first derivatives. $F_d$ is the desired force, but the real force is the projection of $F_d$ on $z_B$ due to the quadrotor structure. Furthermore, $v_d$, $R_d$, $\omega_d$, and $\dot{\omega}_d$ are generated as follows:

Assume $p_d \in C^4$. In other words, $p_d^{(4)}$ is continuous. Then

$$v_d = \dot{p}_d, \tag{2.24}$$
$$R_d = [b_{1,d}, b_{2,d}, b_{3,d}], \tag{2.25}$$
$$\omega_d = (R_d^T \dot{R}_d)^\vee, \tag{2.26}$$
$$\dot{\omega}_d = (\dot{R}_d^T \dot{R}_d + R_d^T \ddot{R}_d)^\vee, \tag{2.27}$$

where

$$
\begin{aligned}
b_{1,d} &= \frac{1}{\|F_d\|}\begin{pmatrix} F_{d,3} + \frac{F_{d,2}^2}{\|F_d\|+F_{d,3}} \\ -\frac{F_{d,1}F_{d,2}}{\|F_d\|+F_{d,3}} \\ -F_{d,1} \end{pmatrix}, \\
b_{2,d} &= \frac{1}{\|F_d\|}\begin{pmatrix} -\frac{F_{d,1}F_{d,2}}{\|F_d\|+F_{d,3}} \\ F_{d,3} + \frac{F_{d,1}^2}{\|F_d\|+F_{d,3}} \\ -F_{d,2} \end{pmatrix}, \\
b_{3,d} &= \frac{F_d}{\|F_d\|}.
\end{aligned} \tag{2.28}
$$

### 2.4.2 Error Dynamics

With the proposed controller (2.23) and the error definitions (2.17)–(2.20), we transform the quadrotor dynamical system (2.5)–(2.8) into the following error system.

**Proposition 2.4.1.** The error system is governed by

$$\dot{e}_p = e_v, \tag{2.29}$$

$$\dot{e}_v = -\frac{1}{m}(k_p e_p + k_v e_v - \Delta_f), \tag{2.30}$$

$$\dot{e}_R = C(R, R_d)e_\omega, \tag{2.31}$$

$$\dot{e}_\omega = J^{-1}(-k_R e_R - k_\omega e_\omega), \tag{2.32}$$

where

$$C(R, R_d) = \frac{1}{2}(\mathrm{tr}[R^\intercal R_d]\, I - R^\intercal R_d), \tag{2.33}$$

$$\Delta_f = \|F_d\|\,((b_{3,d}\cdot b_3)b_3 - b_{3,d}). \tag{2.34}$$

*Proof.* Obviously, $\dot{e}_p = e_v$. Then, the derivative of $e_v$ is given by

$$
\begin{aligned}
m\dot{e}_v &= m\ddot{p} - m\ddot{p}_d \\
&= -mge_3 + fRe_3 - m\ddot{p}_d \\
&= -mge_3 - m\ddot{p}_d + F_d + (fRe_3 - F_d),
\end{aligned}
$$

where, using the definition of $F_d$ in (2.23),

$$-mge_3 - m\ddot{p}_d + F_d = -k_p e_p - k_v e_v,$$

and

$$\begin{aligned} fRe_3 - F_d &= (F_d \cdot Re_3) - F_d(\|F_d\| \, b_{3,d} \cdot b_3)b_3 - \|F_d\| \, b_{3,d} \\ &= \|F_d\| \, ((b_{3,d} \cdot b_3)b_3 - b_{3,d}) = \Delta_f. \end{aligned}$$

To derive the derivative of $e_R$, we have

$$\begin{aligned} \dot{e}_R &= \frac{d}{dt}(\frac{1}{2}(R_d^\intercal R - R^\intercal R_d)^\vee) \\ &= \frac{1}{2}(\frac{d}{dt}(R_d^\intercal R) - \frac{d}{dt}(R^\intercal R_d))^\vee \\ &= \frac{1}{2}(R_d^\intercal R\hat{e}_\omega - \hat{e}_\omega^\intercal R^\intercal R_d)^\vee \\ &= \frac{1}{2}(R_d^\intercal R\hat{e}_\omega + \hat{e}_\omega R^\intercal R_d)^\vee. \end{aligned} \tag{2.35}$$

Using Lemma A.2.4 of the hat map, we obtain

$$\dot{e}_R = \frac{1}{2}(\mathrm{tr}[R^\intercal R_d] \, I - R^T R_d)e_\omega. \tag{2.36}$$

For the derivative of angular velocity error, we have

$$\begin{aligned} \dot{e}_\omega &= \dot{\omega} - \frac{d}{dt}(R^\intercal R_d \omega_d) \\ &= \dot{\omega} - e_\omega^\intercal R^\intercal R_d \omega_d - R^\intercal R_d \dot{\omega}_d. \end{aligned} \tag{2.37}$$

Substitute (2.8) and (2.23) into $\dot{e}_\omega$, we get

$$\dot{e}_\omega = J^{-1}(-k_R e_R - k_\omega e_\omega). \tag{2.38}$$

The configuration error function of $\mathsf{SO(3)}$ is define in [18] as:

$$\Psi(t) = \frac{1}{2}\mathrm{tr}(\mathrm{I}_3 - R_d^\intercal(t)R(t)). \tag{2.39}$$

For the time derivative of $\Psi$, we have

$$\begin{aligned}\dot{\Psi}(R, R_d) &= \frac{d}{dt}(\frac{1}{2}\text{tr}[I - R_d^T]) \\ &= -\frac{1}{2}\text{tr}[R_d^T R \hat{e}_\omega] \,.\end{aligned} \tag{2.40}$$

By Lemma A.2.3, we have

$$\begin{aligned}\dot{\Psi}(R, R_d) &= \frac{1}{2} e_\omega^T (R_d^T R - R^T R_d)^\wedge \\ &= e_\omega^T e_R \\ &= e_\omega \cdot e_R.\end{aligned} \tag{2.41}$$

□

## 2.5 Differential Flatness

Differential flatness was first studied as a property of nonlinear systems in [7], extending the concept of controllability from linear to nonlinear systems. In a differentially flat system, the states and control inputs can be expressed as explicit functions of flat outputs and a finite number of their derivatives. This property ensures the existence of a unique open-loop control $\mathbf{u}_{df}$ for any flat system, provided that the flat outputs are designed as class $C^k$ functions, where $k$ is finite. Thus, the goal is to design a controller that eventually converges to $\mathbf{u}_{df}$. The authors of [21] first proved the differential flatness of the quadrotor system, but their proof contained minor errors due to the use of an ambiguous coordinate system. Later, [6] corrected the proof with carefully defined vectors in world and body coordinates. In this section, we prove the differential flatness of the quadrotor using the corrected approach.

Consider an autonomous system described by the nonlinear differential equation:

$$\dot{X} = f(X, U), \quad X \in \mathbb{R}^n, \quad U \in \mathbb{R}^m. \tag{2.42}$$

The system is differentially flat [30] if there exist functions $\Pi$, $\Lambda$, and an invertible

function $\Gamma$ such that

$$\begin{aligned} \sigma &= \Pi(X, U, \dot{U}, \ldots, U^{(p)}), \\ X &= \Lambda(\sigma, \dot{\sigma}, \ldots, \sigma^{(q)}), \\ U &= \Gamma^{-1}(\sigma, \dot{\sigma}, \ldots, \sigma^{(q)}), \end{aligned} \tag{2.43}$$

where $\sigma$ is called the flat output. The invertible function $\Gamma$ ensures the uniqueness of the control inputs given the flat outputs and their derivatives.

The flat outputs for the quadrotor system are chosen as $\sigma = [p_x, p_y, p_z, \psi]$, where $p_x, p_y, p_z$ represent the positions along the $x$-, $y$-, and $z$-axes, and $\psi$ is the yaw angle. In the remainder of this section, we seek explicit functions that describe the states

$$[p_x, p_y, p_z, R, \dot{p}_x, \dot{p}_y, \dot{p}_z, \omega_x, \omega_y, \omega_z]$$

and the control inputs

$$[f, \tau_x, \tau_y, \tau_z]$$

using $\sigma$ and its derivatives as inputs.

### 2.5.1 Translation

Because the first three flat outputs are simply chosen as the position in $x$-axis, $y$-axis, and $z$-axis, the first and second derivatives of the first three flat outputs correspond to velocity and acceleration respectively. In other words,

$$\begin{aligned} \dot{p}_x, \dot{p}_y, \dot{p}_z &= \dot{\sigma}_{1:3}. \\ \ddot{p}_x, \ddot{p}_y, \ddot{p}_z &= \ddot{\sigma}_{1:3}. \end{aligned} \tag{2.44}$$

### 2.5.2 Attitude

The rotation matrix $R = [x_B, y_B, z_B]$ can be used to denote the attitude of quadrotor, where $x_B, y_B, z_B$ are vectors that denote the directions of $x$-axis, $y$-axis, and $z$-axis

of the body frame. Then $x_B, y_B, z_B$ can be expressed as follows:

$$\begin{aligned} x_B &= \frac{y_C \times z_C}{\|y_C \times z_C\|}, \\ y_B &= \frac{z_C \times x_B}{\|z_C \times x_B\|}, \\ z_B &= x_B \times y_B. \end{aligned} \tag{2.45}$$

where

$$\begin{aligned} x_C &= \left(\cos\sigma_4, \sin\sigma_4, 0\right)^T, \\ y_C &= \left(-\sin\sigma_4, \cos\sigma_4, 0\right)^T, \\ z_C &= \left(\ddot{\sigma}_1, \ddot{\sigma}_2, \ddot{\sigma}_3 + g\right)^T, \end{aligned} \tag{2.46}$$

where $x_C$ and $y_C$ are collinear to the projection of $x_B$ and $y_B$ into the $x_W - y_W$ plane. The rotation matrix $R$ can be converted to Euler angles $(\phi, \theta, \psi)$ using (A.1).

### 2.5.3 Control Input - Thrust $f$

The expression for $f$ is derived here because it will be used later in the expressions for angular velocity. By (2.6), the thrust can be expressed as

$$f = m(\ddot{p} + ge_3) \cdot z_B. \tag{2.47}$$

The first derivative and second derivative are

$$\dot{f} = m(p^{(3)} \cdot z_B + \ddot{p} \cdot R\hat{\omega}e_3), \tag{2.48}$$

$$\ddot{f} = m(p^{(4)} \cdot z_B + 2p^{(3)} \cdot R\hat{\omega}e_3 + p^{(2)} \cdot (R\hat{\omega}^2 e_3 + R\dot{\hat{\omega}}e_3)). \tag{2.49}$$

### 2.5.4 Angular Velocity

For angular velocity denoted as

$$\omega_{BW} = \omega_x x_B + \omega_y y_B + \omega_z z_B,$$

we have the following proposition.

**Proposition 2.5.1.** We have

$$\begin{aligned}\omega_x &= -\frac{m}{f}p^{(3)} \cdot y_B, \\ \omega_y &= \frac{m}{f}p^{(3)} \cdot x_B, \\ \omega_z &= \frac{\dot{\psi}{x_C}^T x_B + \frac{m}{f}{y_C}^T z_B({x_B}^T p^{(3)})}{\|y_C \times z_B\|}.\end{aligned} \tag{2.50}$$

*Proof.* By differentiating (2.6), we get

$$\begin{aligned}p^{(3)} &= \frac{1}{m}(\dot{f}z_B + f\dot{z}_B) \\ &= \frac{1}{m}(\dot{f}z_B + fR\hat{\omega}e_3).\end{aligned} \tag{2.51}$$

Substituting (2.48) into (2.51), we have

$$h_\omega = R\hat{\omega}e_3 = \frac{m}{f}(p^{(3)} - (p^{(3)} \cdot z_B)z_B), \tag{2.52}$$

where $h_\omega$ is the projection of $\frac{m}{f}p^{(3)}$ onto the $x_B - y_B$ plane. Then we have the components $\omega_x$ and $\omega_y$ expressed as

$$\begin{aligned}\omega_x &= -h_\omega \cdot y_B \\ &= -\frac{m}{f}(p^{(3)} - (p^{(3)} \cdot z_B)z_B) \cdot y_B \\ &= -\frac{m}{f}p^{(3)} \cdot y_B,\end{aligned} \tag{2.53}$$

$$\begin{aligned}\omega_y &= h_\omega \cdot x_B \\ &= \frac{m}{f}(p^{(3)} - (p^{(3)} \cdot z_B)z_B) \cdot x_B \\ &= \frac{m}{f}p^{(3)} \cdot x_B.\end{aligned} \tag{2.54}$$

As for $\omega_z$, we multiply both sides of (2.7) by ${y_B}^T$ and obtain

$$\begin{aligned}
{y_B}^T \dot{R} &= {y_B}^T R\hat{\omega}, \\
{y_B}^T \begin{pmatrix} \dot{x}_B & \dot{y}_B & \dot{z}_B \end{pmatrix} &= {y_B}^T \begin{pmatrix} x_B & y_B & z_B \end{pmatrix} \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}, \\
\begin{pmatrix} {y_B}^T \dot{x}_B & {y_B}^T \dot{y}_B & {y_B}^T \dot{z}_B \end{pmatrix} &= \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}, \\
\omega_z &= {y_B}^T \dot{x}_B.
\end{aligned} \tag{2.55}$$

Since $x_B$ is perpendicular to $y_C$ and $z_B$, we have

$$x_B = \frac{\tilde{x}_B}{\|\tilde{x}_B\|}, \tag{2.56}$$

where $\tilde{x}_B = y_C \times z_B$. The derivative is then

$$\dot{x}_B = \frac{\dot{\tilde{x}}_B}{\|\tilde{x}_B\|} - \tilde{x}_B \frac{\tilde{x}_B^T \dot{\tilde{x}}_B}{\|\tilde{x}_B\|^3}. \tag{2.57}$$

Since $\tilde{x}_B$ is collinear to $x_B$, it is perpendicular to $y_B$. Then we have

$$\omega_z = {y_B}^T \frac{\dot{\tilde{x}}_B}{\|\tilde{x}_B\|}. \tag{2.58}$$

The derivative of $\dot{\tilde{x}}_B$ is

$$\dot{\tilde{x}}_B = \dot{y}_C \times z_B + y_C \times \dot{z}_B, \tag{2.59}$$

where

$$\begin{aligned}
\dot{y}_C &= R_{WC}(\dot{\psi} e_3 \times e_2) \\
&= \begin{pmatrix} x_C & y_C & z_C \end{pmatrix} (-\dot{\psi} e_1) \\
&= -\dot{\psi} x_C,
\end{aligned} \tag{2.60}$$

and

$$\begin{aligned}\dot{z}_B &= R\hat{\omega}e_3 \\ &= R\begin{pmatrix}\omega_y & -\omega_x & 0\end{pmatrix}^T \\ &= \omega_y x_B - \omega_x y_B,\end{aligned} \tag{2.61}$$

and so the derivative of $\dot{\tilde{x}}_B$ is

$$\dot{\tilde{x}}_B = -\dot{\psi}x_C \times z_B + \omega_y y_C \times x_B - \omega_x y_C \times y_B. \tag{2.62}$$

Plug this back into (2.58), we get

$$\begin{aligned}\omega_z &= {y_B}^T \frac{\dot{\tilde{x}}_B}{\|\tilde{x}_B\|} \\ &= \frac{1}{\|\tilde{x}_B\|}(-\dot{\psi}{y_B}^T(x_C \times z_B) + \omega_y {y_B}^T(y_C \times x_B)) \\ &= \frac{1}{\|\tilde{x}_B\|}(\dot{\psi}{x_C}^T(y_B \times z_B) - \omega_y {y_C}^T(y_B \times x_B)) \\ &= \frac{1}{\|\tilde{x}_B\|}(\dot{\psi}{x_C}^T x_B + \omega_y {y_C}^T z_B).\end{aligned} \tag{2.63}$$

Substitute $\omega_y$ from (2.54) and we have an explicit expression $\omega_z$ as in (2.50).

□

### 2.5.5 Angular Acceleration

Next, we derive equations for the angular accelerations.

**Proposition 2.5.2.** The angular accelerations can be expressed as follows:

$$\begin{aligned}\dot{\omega}_x &= -\frac{m}{f}{y_B}^T p^{(4)} - 2\frac{\dot{f}}{f}\omega_x + \omega_y\omega_z, \\ \dot{\omega}_y &= \frac{m}{f}{x_B}^T p^{(4)} - 2\frac{\dot{f}}{f}\omega_y + \omega_x\omega_z, \\ \dot{\omega}_z &= 1 + \frac{{y_B}^T z_B({x_B}^T p^{(4)} - 2\frac{\dot{f}}{m}\omega_y + \frac{f}{m}\omega_x\omega_z)}{\frac{f}{m}\|y_C \times z_B\|}.\end{aligned} \tag{2.64}$$

*Proof.* We start by obtaining the snap. Taking the derivative of (2.51), we have

$$p^{(4)} = \frac{1}{m}(\ddot{f} z_B + 2\dot{f} R\hat{\omega} e_3 + f R\hat{\omega}^2 e_3 + f R\hat{\dot{\omega}} e_3) \tag{2.65}$$

Multiplying both sides of (2.65) with ${x_B}^T$ on the left, we have

$${x_B}^T p^{(4)} = \frac{1}{m}(f\dot{\omega}_y + 2\dot{f}\omega_y + f\omega_x\omega_z). \tag{2.66}$$

Multiplying both sides of (2.65) with ${y_B}^T$ on the left, we have

$${y_B}^T p^{(4)} = \frac{1}{m}(-f\dot{\omega}_x - 2\dot{f}\omega_x + f\omega_y\omega_z). \tag{2.67}$$

Differentiating (2.58) and using (2.57), we have

$$\begin{aligned} -\dot{\omega}_y {y_C}^T z_B - \omega_y \dot{y}_C^T z_B - \omega_y {y_C}^T \dot{z}_B + \dot{\omega}_z \|y_C \times z_B\| + \omega_z \frac{\tilde{x}_B^T \dot{\tilde{x}}_B}{\|\tilde{x}_B\|} \\ = \ddot{\psi} {x_C}^T x_B + \dot{\psi} \dot{x}_C^T x_B + \dot{\psi} {x_C}^T \dot{x}_B. \end{aligned} \tag{2.68}$$

To further simplify (2.68), we use the following derivations:

From (2.60) we have

$$-\omega_y \dot{y}_C^T z_B = \dot{\psi}\omega_y {x_C}^T z_B. \tag{2.69}$$

By (2.61) and ${y_C}^T z_B = 0$, we get

$$-\omega_y {y_C}^T \dot{z}_B = \omega_x\omega_y {y_C}^T y_B. \tag{2.70}$$

By (2.56) and (2.62), we have

$$\begin{aligned} \omega_z \frac{\tilde{x}_B^T \dot{\tilde{x}}_B}{\|\tilde{x}_B\|} &= \omega_z {x_B}^T \dot{\tilde{x}}_B \\ &= \omega_z {x_B}^T (-\dot{\psi} x_C \times z_B + \omega_y y_C \times x_B - \omega_x y_C \times y_B) \\ &= \omega_z (-\dot{\psi} {x_B}^T (x_C \times z_B) - \omega_x {x_B}^T (y_C \times y_B)) \\ &= \omega_z (\dot{\psi} {x_C}^T (x_B \times z_B) + \omega_x {y_C}^T (x_B \times y_B)) \\ &= \omega_z (-\dot{\psi} {x_C}^T y_B + \omega_x {y_C}^T z_B). \end{aligned} \tag{2.71}$$

Similarly to (2.60), we derive

$$\begin{aligned}\dot{x}_C &= R_{WC}(\dot{\psi}e_3 \times e_1) \\ &= \begin{pmatrix} x_C & y_C & z_C \end{pmatrix} (\dot{\psi}e_2) \\ &= \dot{\psi}y_C.\end{aligned} \tag{2.72}$$

With (2.72) and the fact that ${y_C}^T x_B = 0$, we get

$${\dot{\psi}\dot{x_C}}^T x_B = \dot{\psi}^2 {y_C}^T x_B = 0. \tag{2.73}$$

Finally, similarly to (2.61), we have

$$\begin{aligned}\dot{x}_B &= R\hat{\omega}e_1 \\ &= R \begin{pmatrix} 0 & \omega_z & -\omega_y \end{pmatrix}^T \\ &= \omega_z y_B - \omega_y z_B,\end{aligned} \tag{2.74}$$

which can be simplified to

$$\dot{\psi}{x_C}^T \dot{x}_B = \dot{\psi}\omega_z {x_C}^T y_B - \dot{\psi}\omega_y {x_C}^T z_B. \tag{2.75}$$

Substituting (2.69)–(2.75) into (2.68) and we obtain

$$\begin{aligned}-\dot{\omega}_y {y_C}^T z_B + \dot{\omega}_z \left\| y_C \times z_B \right\| = \ddot{\psi}{x_C}^T x_B + 2\dot{\psi}\omega_z {x_C}^T y_B - 2\dot{\psi}\omega_y {x_C}^T z_B \\ - \omega_x\omega_y {y_C}^T y_B - \omega_x\omega_z {y_C}^T z_B.\end{aligned} \tag{2.76}$$

Then the angular accelerations can be obtained by solving the following combination of linear equations consisting of (2.66), (2.67), and (2.76):

$$\begin{aligned}\dot{\omega}_y \frac{f}{m} &= {x_B}^T p^{(4)} - \frac{2}{m}\dot{f}\omega_y - \frac{f}{m}\omega_x\omega_z, \\ \dot{\omega}_x \frac{f}{m} &= -{y_B}^T p^{(4)} - \frac{2}{m}\dot{f}\omega_x - \frac{f}{m}\omega_y\omega_z, \\ \dot{\omega}_y(-{y_C}^T z_B) + \dot{\omega}_z \left\| y_C \times z_B \right\| &= \ddot{\psi}{x_C}^T x_B + 2\dot{\psi}\omega_z {x_C}^T y_B - 2\dot{\psi}\omega_y {x_C}^T z_B \\ &\quad - \omega_x\omega_y {y_C}^T y_B - \omega_x\omega_z {y_C}^T z_B.\end{aligned} \tag{2.77}$$

Solving this, we have (2.64).

□

### 2.5.6 Control Input - Torque $\tau$

From the dynamics (2.8), we have

$$\tau = J\dot{\omega} + \omega \times J\omega, \tag{2.78}$$

where $\omega = \begin{pmatrix} \omega_x & \omega_y & \omega_z \end{pmatrix}^T$ and $\dot{\omega} = \begin{pmatrix} \dot{\omega}_x & \dot{\omega}_y & \dot{\omega}_z \end{pmatrix}^T$ can be found in Subsection 2.5.4 and Subsection 2.5.5.

# Chapter 3

# Trajectory Generation with RRT and Bezier Curve

In this chapter, we develop a trajectory generation approach for quadrotor reach-avoid specifications. We employ the Rapidly-exploring Random Trees (RRT) algorithm for generating waypoints, followed by using Bezier curves to transform the generated waypoint trajectory into a smooth $C^k$ function. While the concept of differential flatness grants flexibility by allowing us to bypass quadrotor dynamics during planning, the resultant state space remains four-dimensional instead of three. Consequently, we set $\sigma_4(t) = \psi(t) = 0$ throughout the trajectory, avoiding the need for planning the attitude trajectory. Given that the attitude trajectory holds lesser significance in reach-avoid tasks, we can confidently disregard $\psi(t)$ and concentrate on planning a three-dimensional trajectory. In other words, we forgo planning the rotation around the $z$-axis in the body coordinate to plan in three-dimensional space $(p_x, p_y, p_z)$ instead of four-dimensional space $(p_x, p_y, p_z, \psi)$.

## 3.1 Waypoints Generation with RRT

### 3.1.1 Problem Formulation

Let $\mathcal{X}_o, \mathcal{X}_g, \mathcal{X}_u \subseteq \mathbb{R}^3 \times \mathbb{R}^3 \times \mathsf{SO(3)} \times \mathbb{R}^3$ be an operating domain, a target set, and an unsafe set, respectively. Define $\mathcal{X}_s = \mathcal{X}_o \backslash \mathcal{X}_u$ to be the safe set. Construct a continuous 3D path consisting of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$, where

vertices are waypoints and edges are the segments between two waypoints. The path should satisfy the following requirements:

- The path starts from any point $x_{init} \in \mathcal{X}_s$ and ends at any point $x_g \in \mathcal{X}_g$.
- The parent of each vertex is contained in a safe hyper-rectangular neighbor of the parent vertex.
- Every edge in $\mathcal{E}$ lies entirely in $\mathcal{X}_s$.

### 3.1.2 The Modified RRT Algorithm

Rapidly-exploring Random Trees (RRT) [15] is a widely-used algorithm in the field of motion planning. It efficiently explores the state space to generate feasible paths for robotic systems by iteratively sampling random configurations and connecting them to the existing tree structure. RRT is particularly suited for high-dimensional and complex environments due to its ability to rapidly expand the search space towards unexplored regions, ultimately leading to the discovery of feasible paths from the start to the goal state. We introduce a modified version of RRT for safety-guaranteed planning. The key modification involves computing a safe hyper-rectangular region around each vertex, which serves as the new sampling space for selecting the next vertex.

Define

$$\tilde{\mathcal{X}}_o = \mathcal{X}_o - p_{safe}\mathbb{B}_3^\infty, \tag{3.1}$$

$$\tilde{\mathcal{X}}_u^{(i)} = [\![\underline{x}_u^{(i)}, \overline{x}_u^{(i)}]\!] + p_{safe}\mathbb{B}_3^\infty,\ i \in [1; N_u], \tag{3.2}$$

$$\tilde{\mathcal{X}}_u = \bigcup_{i=1}^{N_u} \tilde{\mathcal{X}}_u^{(i)}, \tag{3.3}$$

$$\tilde{\mathcal{X}}_t = \mathcal{X}_t - p_{safe}\mathbb{B}_3^\infty, \tag{3.4}$$

where $\mathbb{B}_3^\infty$ denotes the 3-dimensional closed unit ball w.r.t. $\|\cdot\|_\infty$. $p_{safe}$ is the safe margin guaranteed by the controller. It will be computed in Chapter 4. $\tilde{\mathcal{X}}_u$ can be treated as an inflated version of the unsafe set $\mathcal{X}_u$. The hyper-rectangle $[\![a, b]\!]$ denotes the set $\{x \in \mathbb{R}^n : a \le x \le b\}$. For such hyper-rectangle, we define

$$\text{center}([\![a, b]\!]) = (a + b)/2, \tag{3.5}$$

$$\text{radius}([\![a, b]\!]) = (b - a)/2, \tag{3.6}$$

where $a$ and $b$ are assumed to be finite.

The safe tube is obtained by generating $N_s + 1$ waypoints $\mathtt{p}_0, \mathtt{p}_1, \ldots, \mathtt{p}_{N_s} \in \mathbb{R}^3$, and associated vector radii $\mathtt{r}_0, \mathtt{r}_1, \mathtt{r}_2, \ldots, \mathtt{r}_{N_s} \in \mathbb{R}^3_+$, satisfying the following:

$$\begin{aligned} \mathtt{p}_0 &= p_0, \\ \mathtt{p}_{i+1} &\in \mathtt{p}_i + [\![-\mathtt{r}_i, \mathtt{r}_i]\!], \ i \in [0; N_s - 1], \\ \mathtt{p}_i + [\![-\mathtt{r}_i, \mathtt{r}_i]\!] &\subseteq \tilde{\mathcal{X}}_o \setminus \tilde{\mathcal{X}}_u, \ i \in [0; N_s - 1], \\ \mathtt{p}_{N_s} + [\![-\mathtt{r}_{N_s}, \mathtt{r}_{N_s}]\!] &\in \tilde{\mathcal{X}}_t. \end{aligned} \tag{3.7}$$

The waypoints and the associated safe radii can be estimated by integrating sampling-based approaches [12] with safe and efficient hyper-rectangular set-based computations [27]. Let $N_{\text{sample}}$ be the maximum number of vertices and $C_{\text{sample}} \in (0, 1)$ be a parameter, and let the function sample be a sampling function such that sample$(S)$ randomly generates a point from the set $S$. The random tree is then computed according to Algorithm 1.

Define

$$(\text{ClosestPoint}(x, [\![a, b]\!]))_i = \begin{cases} x_i, \ x_i \in [a_i, b_i], \\ c_i + r_i \text{sgn}(x_i - c_i), \ \text{otherwise}, \end{cases} \tag{3.8}$$

where $\text{sgn}(\cdot)$ is the signum function, $r = \text{radius}([\![a, b]\!])$ and $c = \text{center}([\![a, b]\!])$. Then the algorithm is as follows:

**Algorithm 1** The modified RRT algorithm

$i \leftarrow 1$, $x_i \leftarrow p_0$, $\mathcal{V} \leftarrow \{x_i\}$, $\mathcal{E} \leftarrow \emptyset$
**while** $i \leq N_{\text{sample}}$ **do**
  **if** $i \leq C_{\text{sample}} N_{\text{sample}}$ **then**
    $x_s \leftarrow \mathtt{sample}(\tilde{\mathcal{X}}_{\text{s}} \setminus \tilde{\mathcal{X}}_u)$
  **else**
    $x_s \leftarrow \mathtt{sample}(\tilde{\mathcal{X}}_t)$
  **end if**
  $d \leftarrow \infty$
  **for** $x_j$ in $\mathcal{V}$ **do**
    $d' \leftarrow \left\| x_s - \text{ClosestPoint}(x_s, \mathfrak{R}(x_j, \tilde{\mathcal{X}}_o, \tilde{\mathcal{X}}_u, \alpha)) \right\|_\infty$
    **if** $d' < d$ **then**
      $d \leftarrow d'$, $x_{i\text{near}} \leftarrow x_j$

**end if**
**end for**
$i \leftarrow i + 1$
$x_i \leftarrow \text{ClosestPoint}(x_{\text{s}}, \mathfrak{R}(x_{i_{\text{near}}}, \tilde{\mathcal{X}}_o, \tilde{\mathcal{X}}_u, \alpha))$
$\mathcal{V} \leftarrow \mathcal{V} \cup \{x_i\}$, $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{i_{\text{near}}}, x_i)\}$
**if** $x_i \in \tilde{X}_t$ **then** break
**end if**
**end while**
**return** $\mathcal{T} = (\mathcal{V}, \mathcal{E})$

In Algorithm 1,

$$\mathfrak{R}(y, \tilde{\mathcal{X}}_{\text{o}}, \tilde{\mathcal{X}}_{\text{u}}, \alpha) = \mathcal{H}(y, \tilde{\mathcal{X}}_{\text{o}}) \bigcap \left( \cap_{i=1}^{N_{\text{u}}} \mathcal{S}(y, \tilde{\mathcal{X}}_{\text{u}}^{(i)}, \alpha) \right),$$

$$\mathcal{H}(v, [\![a, b]\!]) = v + [\![-r_{\mathcal{H}}, r_{\mathcal{H}}]\!],$$

$$r_{\mathcal{H}} = \text{radius}([\![a, b]\!]) - |\text{center}([\![a, b]\!]) - v|,$$

$$\mathcal{S}(x, [\![a, b]\!], \alpha) = \{z \in \mathbb{R}^3 : \|z - x\| \leq \alpha \|x - \text{ClosestPoint}(x, [\![a, b]\!])\|_\infty, \alpha \in [0, 1)\}.$$

## 3.2 Piecewise Bezier Curves

Once the waypoints are generated, the desired trajectory is smoothed by piecewise Bezier curves. A Bezier curve is a linear combination of Bernstein polynomial functions. It is parameterized by some number $\eta$ over $[0, 1]$ as follows:

$$p(\eta) = \sum_{i=0}^{N} c_i b_i^N(\eta), \tag{3.9}$$

where the Bernstein polynomial function is defined as

$$b_i^N(\eta) = \binom{N}{i} (1 - \eta)^{N-i} \eta^i, \tag{3.10}$$

$\eta \in [0, 1]$, $N$ is the order of the polynomial, and $c_i$ are control points that determine the shape of the polynomial. If we treat every waypoint generated by RRT as an anchor point of two Bezier curves, then we can use $N - 1$ Bezier curves to connect $N$

waypoints and get one piecewise trajectory. Under some constraints on every anchor point, the smoothed trajectory is of class $C^k$.

We assume $p_d$ to be a piecewise Bezier curve with $N_s$ segments, where each segment has $N_p$ control points. Let $\delta_i,\ i \in [1; N_s]$ be the duration of each segment, i.e. $\delta_i = t_i - t_{i-1}$, the initial time be $t_0 = 0$, and the total time be $T = \sum_{i=1}^{N_s}$. Then $p_d \colon [0, T] \to \mathbb{R}^3$ has the form:

$$p_d(t) = \begin{cases} \sum_{i=0}^{N_p} c_1^i b_i^{N_p}(\frac{t-t_0}{\delta_1}),\ t \in [t_0, t_1], \\ \sum_{i=0}^{N_p} c_2^i b_i^{N_p}(\frac{t-t_1}{\delta_2}),\ t \in [t_1, t_2], \\ \vdots \\ \sum_{i=0}^{N_p} c_j^i b_i^{N_p}(\frac{t-t_{j-1}}{\delta_j}),\ t \in [t_{j-1}, t_j], \\ \vdots \\ \sum_{i=0}^{N_p} c_{N_s}^i b_i^{N_p}(\frac{t-t_{N_s-1}}{\delta_{N_s}}),\ t \in [t_{N_s-1}, t_{N_s}]. \end{cases} \tag{3.11}$$

The control points $c_i^j,\ i \in [1; N_s],\ j \in [0; N_p]$, are required to satisfy the following set of constraints:

- the initial value of $(p_d,\, \dot{p}_d,\, \ddot{p}_d,\, p_d^{(3)}) = (p_0,\, v_0,\, 0_3,\, 0_3)$;
- safety of the generated trajectory in the sense that $p_d(t) \in \mathtt{p}_{i-1} + [\![-\mathtt{r}_{i-1}, \mathtt{r}_{i-1}]\!],\ t \in [t_{i-1}, t_i],\ i \in [1; N_s]$;
- continuity of $p_d$, $\dot{p}_d$, $\ddot{p}_d$, $p_d^{(3)}$, and $p_d^{(4)}$ at the junction points;
- satisfying the bound $|ge_3 + \ddot{p}_d(t)| \leq a_{\max},\ \forall t \in [0, T]$, $a_{\max} \in \mathbb{R}^3$ is a constant vector;
- the value of $p_d$ at final time is in $\mathtt{p}_{N_s} + [\![-\mathtt{r}_{N_s}, \mathtt{r}_{N_s}]\!]$;
- the value of $\dot{p}_d$ at final time is $0_3$.

In Algorithm 2, we present a heuristic approach to determine the desired trajectory $p_d$ by means of iterative linear programming. The heuristic approach relies on initially guessing the time $T$ and incrementally increasing it until the constraints become feasible.

**Algorithm 2** Computing the desired trajectory.

1: Define $l_i = \|\mathtt{p}_i - \mathtt{p}_{i-1}\|$, $i \in [1; N_s]$, $L = \sum_{i=1}^{N_s} l_i$, $q_i = \frac{l_i}{L}$, $i \in [1; N_s]$.
2: Let $T_0$ be a positive parameter specifying an initial guess for the full time horizon and $\alpha_t > 1$ be a design parameter.
3: Define $\delta_i = q_i T_0$, $i \in [1; N_s]$.
4: With the values of $\delta_i$, $i \in [1; N_s]$, defined in step 3, solve a linear program involving the control points $c_i^j$, while considering the constraints.
5: If the linear program in step 4 is feasible, the resulting control points in addition to the time durations $\delta_i$ can then be used to synthesize the desired trajectory according to equation (3.11).
6: If the linear program in step 4 is infeasible, redefine $T_0$ as $T_0 = \alpha_T T_0$, and repeat steps 3 and 4.

# Chapter 4

# Lyapunov-based Safety Guaranteed Synthesis

In this chapter, we first present several definitions of equilibrium point stability within the context of Lyapunov theory. Subsequently, we analyze the quadrotor error system to establish exponential stability criteria. Using the Lyapunov function, we derive an upper bound on the position error to ensure safety-guaranteed tracking performance. However, recognizing the conservative nature of this analysis, we provide an alternative approach that uses Bernoulli's inequality to achieve a more precise error bound while exploring asymptotic stability.

## 4.1 Lyapunov Stability Theorem

Consider the nonlinear system described by an ODE as follows:

$$\dot{x} = f(x), \tag{4.1}$$

where $f : X \to \mathbb{R}^n$ is a locally Lipschitz function on an open set $X \subset \mathbb{R}^n$. Suppose $X$ contains an equilibrium point $x^*$ of (4.1). It is convenient to assume that $x^* = 0$ without loss of generality. Furthermore, if (4.1) describes an error system, it is natural to analyze if the solution $x(t, x_0)$ for some $x(0) = x_0 \in X$ would eventually go to the equilibrium point $x^* = 0$.

**Definition 4.1.1.** (Five Types of Stability of Equilibrium Point) The equilibrium point $x^* = 0$ of (4.1) is

1. **stable**, if for any $\epsilon > 0$, there exists $\delta > 0$ such that

$$\|x_0\| \leq \delta \Rightarrow \|x(t, x_0)\| \leq \epsilon, \forall t \geq 0; \tag{4.2}$$

2. **asymptotically stable**, if it is stable and there exists $\xi > 0$ such that

$$\|x_0\| \leq \xi \Rightarrow \lim_{t\to\infty} x(t, x_0) = 0; \tag{4.3}$$

3. **globally asymptotically stable**, if it is stable and

$$\lim_{t\to\infty} x(t, x_0) = 0, \quad \forall x_0 \in \mathbb{R}^n; \tag{4.4}$$

4. **exponentially stable**, if there exist $m, \gamma, \beta > 0$, such that

$$\|x(t, x_0)\| \leq m\|x_0\|e^{-\beta t}, \quad \forall \|x_0\| \leq \gamma, \forall t \geq 0; \tag{4.5}$$

5. **globally exponentially stable**, if there exist $m, \beta > 0$, such that

$$\|x(t, x_0)\| \leq m\|x_0\|e^{-\beta t}, \quad \forall x_0 \in \mathbb{R}^n, \forall t \geq 0. \tag{4.6}$$

**Definition 4.1.2.** (Lyapunov Stability Theorem) Let $X \subset \mathbb{R}^n$ be an open set. Suppose the origin $x^* = 0$ is an equilibrium point of (4.1) and is contained in $X$. Let $\mathcal{V} : X \to \mathbb{R}$ be continuously differentiable and positive definite on $X$. Define $\dot{\mathcal{V}} = \frac{d\mathcal{V}}{dx} \cdot f$ as the derivative of $\mathcal{V}$ along solutions of (4.1). The following statements hold:

1. if $\dot{\mathcal{V}}$ is negative semidefinite, then $x^* = 0$ is stable;

2. if $\dot{\mathcal{V}}$ is negative definite, then $x^* = 0$ is asymptotically stable;

3. if $\dot{\mathcal{V}}$ is negative definite and $\mathcal{V} \to 0$ as $|x| \to \infty$ with $X = \mathbb{R}^n$, then $x^* = 0$ is globally asymptotically stable;

4. if there exist positive constants $c_1$, $c_2$, and $c_3$ such that

$$c_1\|x\|^2 \leq \mathcal{V} \leq c_2\|x\|^2$$

and

$$\dot{\mathcal{V}} \leq -c_3\|x\|^2$$

for all $x \in X$, then $x^* = 0$ is exponentially stable;

5. if $X = \mathbb{R}^n$ and there exist positive constants $c_1$, $c_2$, and $c_3$ such that

$$c_1\|x\|^2 \leq \mathcal{V} \leq c_2\|x\|^2$$

and

$$\dot{\mathcal{V}} \leq -c_3\|x\|^2,$$

then $x^* = 0$ is globally exponentially stable.

## 4.2 Lyapunov Stability Analysis

In this section, we show the Lyapunov stability analysis of the error system (2.29)–(2.32). The original analysis is in [18], but the use of Lie derivative is incorrect in their analysis and hence we introduce the corrected version here.

**Theorem 4.2.1.** Consider the controller (2.23). Given any $C^4$ trajectory that satisfies

$$\|mge_3 + m\ddot{p}_d\| \leq B, \tag{4.7}$$

if, the initial conditions of $\Psi$ (defined by (2.39)) and $\|e_\omega\|$ satisfy:

$$\Psi(R(0), R_d(0)) \leq \overline{\Psi} \leq 1,$$
$$\|e_\omega(0)\|^2 \leq \frac{2}{\lambda_{min}(J)} k_R(1 - \Psi(R(0), R_d(0))),$$

for some constant $\overline{\Psi}$, we can find positive constants ($k_p$, $k_v$, $k_R$, $k_\omega$, $c_1, c_2$) such that

$$c_1 < \min\{\sqrt{k_p m}, \frac{(k_p - \alpha k_p)(k_v - \alpha k_v)}{k_p - \alpha k_p + \frac{{k_v}^2}{4m}}\}, \tag{4.8}$$

$$c_2 < \min\{\sqrt{\frac{2k_R}{2 - \overline{\Psi}}\lambda_{max}(J)}, \frac{\frac{k_R k_\omega}{\lambda_{max}(J)}}{(\frac{k_R}{\lambda_{max}(J)} + \frac{{k_\omega}^2}{4\lambda_{min}(J)})}\}, \tag{4.9}$$

$$\lambda_{min}(W_1) > \frac{\|W_2\|^2}{4\lambda_{min}(W_3)}, \tag{4.10}$$

where

$$\alpha = \sqrt{\overline{\Psi}(2-\overline{\Psi})}, \tag{4.11}$$

$$W_1 = \begin{pmatrix} (1-\alpha)\frac{c_1}{m}k_p & -\frac{1}{2}(\frac{c_1}{m}k_v + \frac{c_1}{m}\alpha k_v + \alpha k_p) \\ -\frac{1}{2}(\frac{c_1}{m}k_v + \frac{c_1}{m}\alpha k_v + \alpha k_p) & (1-\alpha)k_v - c_1 \end{pmatrix}, \tag{4.12}$$

$$W_2 = \begin{pmatrix} \frac{c_1}{m}B & 0 \\ B & 0 \end{pmatrix}, W_3 = \begin{pmatrix} \frac{c_2 k_R}{\lambda_{max}(J)} & \frac{c_2 k_\omega}{2\lambda_{min}(J)} \\ \frac{c_2 k_\omega}{2\lambda_{min}(J)} & c_3 k_\omega - c_2 \end{pmatrix}, \tag{4.13}$$

then the error system (2.29)–(2.32) is exponentially stable. Moreover, the position error is bounded as follows:

$$\|e_p\|_{max} \leq \sqrt{\frac{V(0)}{\lambda_{min}(M_1)}}, \tag{4.14}$$

where

$$\begin{aligned} V(0) =& \frac{1}{2}k_p\|e_p(0)\|^2 + \frac{1}{2}m\|e_v(0)\|^2 + c_1 e_p(0)\cdot e_v(0) \\ &+ \frac{1}{2}e_\omega(0)\cdot Je_\omega(0) + k_R\Psi(R(0), R_d(0)) + c_2 e_R(0)\cdot e_\omega(0), \end{aligned} \tag{4.15}$$

$$M_1 = \frac{1}{2}\begin{pmatrix} k_p & -c_1 \\ -c_1 & m \end{pmatrix}. \tag{4.16}$$

The rest of this section is the full proof of Theorem 4.2.1. The stability of attitude and angular velocity is presented first, as it is easier to prove due to the attitude control system being fully actuated. Position and velocity stability is harder to prove since it is coupled with attitude dynamics. This is addressed by bounding the attitude error. Finally, the stability of the complete dynamics is proved by combining the two.

### 4.2.1 Altitude and Angular Velocity Stability

For altitude and angular velocity stability, we consider the following Lyapunov candidate:

$$V_2 = \frac{1}{2}e_\omega \cdot Je_\omega + k_R\Psi(R, R_d) + c_2 e_R \cdot e_\omega.$$

Then, $V_2$ is bounded by

$$z_2^T M_3 z_2 \leq V_2 \leq z_2^T M_4 z_2, \tag{4.17}$$

where $z_2 = [\|e_R\|, \|e_\omega\|]^T$, $M_3 = \frac{1}{2}\begin{pmatrix} k_R & -c_2 \\ -c_2 & \lambda_{min}(J) \end{pmatrix}$ and $M_4 = \frac{1}{2}\begin{pmatrix} \frac{2k_R}{2-\bar{\Psi}} & c_2 \\ c_2 & \lambda_{max}(J) \end{pmatrix}$.

The derivative $\dot{V}_2$ is

$$\begin{aligned}
\dot{V}_2(t) =& e_\omega(t) \cdot (-k_R e_R - k_\omega e_\omega(t)) + k_R e_R(t) \cdot e_\omega(t) \\
&+ c_2 C(R(t), R_d(t)) e_\omega(t) \cdot e_\omega(t) \\
&+ c_2 e_R(t) \cdot J^{-1}(-k_R e_R(t) - k_\omega e_\omega(t)) \\
\leq& - e_R(t) \cdot (c_2 k_R J^{-1}) e_R(t) - (k_\omega - c_2) e_\omega(t) \cdot e_\omega(t) - e_R(t) \cdot (c_2 k_\omega J^{-1}) e_\omega(t) \\
\leq& - \frac{c_2 k_R}{\lambda_{\max}(J)} \|e_R(t)\|^2 - (k_w - c_2) \|e_\omega(t)\|^2 + \frac{c_2 k_\omega}{\lambda_{\min}(J)} \|e_R(t)\| \|e_\omega(t)\| \\
\leq& - z_2^T \begin{pmatrix} \frac{c_2 k_R}{\lambda_{\max}(J)} & -\frac{c_2 k_\omega}{2\lambda_{\min}(J)} \\ -\frac{c_2 k_\omega}{2\lambda_{\min}(J)} & k_w - c_2 \end{pmatrix} z_2 \\
=& -z_2^T W_3 z_2.
\end{aligned}$$

As long as the matrices $M_3$, $M_4$, and $W_3$ are positive definite, we have $V_2$ to be positive definite and $\dot{V}_2$ to be negative definite. By Theorem 4.1.2, the attitude and angular velocity system is exponentially stable.

Note that we analyze the norms of errors instead of the errors themselves to simplify the process. However, we will adopt a different stability analysis as an improvement over the current analysis, involving the errors themselves.

### 4.2.2 Position and Velocity Stability

For position and velocity stability, we consider the following Lyapunov candidate:

$$V_1 = \frac{1}{2} k_p \|e_p\|^2 + \frac{1}{2} m \|e_v\|^2 + c_1 e_p \cdot e_v. \tag{4.18}$$

Apparently, $V_1$ can be bounded as

$$z_1^T M_1 z_1 \leq V_1 \leq z_1^T M_2 z_1, \tag{4.19}$$

where $z_1 = [\|e_p\|, \|e_v\|]^T$, $M_1 = \frac{1}{2}\begin{pmatrix} k_p & -c_1 \\ -c_1 & m \end{pmatrix}$ and $M_2 = \frac{1}{2}\begin{pmatrix} k_p & c_1 \\ c_1 & m \end{pmatrix}$.

The derivative is $\dot{V}_1 = k_p e_p \cdot \dot{e}_p + m e_v \cdot \dot{e}_v + c_1 \dot{e}_p \cdot e_v + c_1 e_p \cdot \dot{e}_v$. Since $\dot{V}_1$ is a function of $\dot{e}_v$, we use the following two remarks to bound $\dot{e}_v$ in order to bound $\dot{V}_1$.

**Lemma 4.2.2.** *$m\dot{e}_v = -mge_3 - m\ddot{p}_d + F_d + \Delta_f$, where $\Delta_f = fRe_3 - F_d$.*

*Proof.* The derivative of $e_v$ is given by

$$\begin{aligned} m\dot{e}_v &= m\ddot{p} - m\ddot{p}_d \\ &= -mge_3 + fRe_3 - m\ddot{p}_d \\ &= -mge_3 - m\ddot{p}_d + F_d + (fRe_3 - F_d), \end{aligned}$$

where, using the definition of $F_d$ in (2.23),

$$-mge_3 - m\ddot{p}_d + F_d = -k_p e_p - k_v e_v,$$

and

$$\begin{aligned} fRe_3 - F_d &= (F_d \cdot Re_3)Re_3 - F_d \\ &= (\|F_d\| \, b_{3,d} \cdot b_3)b_3 - \|F_d\| \, b_{3,d} \\ &= \|F_d\| \, ((b_{3,d} \cdot b_3)b_3 - b_{3,d}) = \Delta_f. \end{aligned}$$

where $F_d$, $b_{3,d}$ are defined in Section 2.4. □

Lemma 4.2.2 indicates the necessity of bounding $\Delta_f$ in order to constrain $\dot{e}_v$. Consequently, we present the following lemma to bound $\Delta_f$ in terms of 2-norm.

**Lemma 4.2.3.** *Assume $\|-mge_3 + m\ddot{p}_d\| \leq B$. Then $\|\Delta_f\| \leq (k_p\|e_p\| + k_v\|e_v\| + B)\alpha$ for some $\alpha \in (0, 1]$.*

*Proof.* For $f = F_d \cdot Re_3$, we have

$$F_d = \frac{f}{e_3^T R_d^T R e_3} R_d e_3.$$

Then,

$$\begin{aligned} \Delta_f &= fRe_3 - F_d \\ &= \frac{f}{e_3^T R_d^T R e_3}((e_3^T R_d^T Re_3)Re_3 - R_d e_3). \end{aligned}$$

Let $A = -k_p e_p - k_v e_v - \Delta_f$. Notice that $\|A\| \leq k_p\|e_p\| + k_v\|e_v\| + B$, where $\| - mge_3 + m\ddot{p}_d\| \leq B$. Define

$$f = -(-k_p e_p - k_v e_v - mge_3 + m\ddot{p}_d) \cdot Re_3 = (\|A\|R_d e_3) \cdot Re_3.$$

Then,

$$-\frac{f}{e_3^T R_d^T R e_3} R_d e_3 = -\frac{(\|A\|R_d e_3) \cdot Re_3}{e_3^T R_d^T R e_3} \cdot -\frac{A}{\|A\|} = A.$$

Therefore,

$$\|\frac{f}{e_3^T R_d^T R e_3}\| = \|\frac{f}{e_3^T R_d^T R e_3} R_d e_3\| = \|A\|.$$

Then,

$$\begin{aligned}
\|\Delta_f\| &\leq \|\frac{f}{e_3^T R_d^T R e_3}\|\|(e_3^T R_d^T R e_3)Re_3 - R_d e_3\| \\
&\leq \|A\|\|(e_3^T R_d^T R e_3)Re_3 - R_d e_3\| \\
&\leq (k_p\|e_p\| + k_v\|e_v\| + B)\|(e_3^T R_d^T R e_3)Re_3 - R_d e_3\| \\
&\leq (k_p\|e_p\| + k_v\|e_v\| + B)\|e_R\| \\
&\leq (k_p\|e_p\| + k_v\|e_v\| + B)\alpha,
\end{aligned}$$

where $\alpha \in (0, 1]$ is the bound of $\|e_R\|$, as shown in Section 2.3. □

Then, by using Lemma 4.2.2 and Lemma 4.2.3, $\dot{V}_1$ can be bounded as follows:

$$\begin{aligned}
\dot{V}_1 &= k_p e_p \cdot e_v + m e_v \cdot \frac{1}{m}(-k_p e_p - k_v e_v - \Delta_f) \\
&\quad + c_1 e_v \cdot e_v + c_1 e_p \cdot \frac{1}{m}(-k_p e_p - k_v e_v - \Delta_f) \\
&\leq (c_1 - k_v + \alpha k_v)\|e_v\|^2 + (\alpha - 1)\frac{c_1}{m} k_p \|e_p\|^2 \\
&\quad + (\frac{c_1}{m} k_v + \frac{c_1}{m}\alpha k_v + \alpha k_p)\|e_p\|\|e_v\| \\
&\quad + \frac{c_1}{m} B\|e_p\|\|e_R\| + B\|e_v\|\|e_R\| \\
&\leq -[\|e_p\|\|e_v\|] \begin{pmatrix} (1-\alpha)\frac{c_1}{m}k_p & -\frac{1}{2}(\frac{c_1}{m}k_v + \frac{c_1}{m}\alpha k_v + \alpha k_p) \\ -\frac{1}{2}(\frac{c_1}{m}k_v + \frac{c_1}{m}\alpha k_v + \alpha k_p) & (1-\alpha)k_v - c_1 \end{pmatrix} \begin{pmatrix} \|e_p\| \\ \|e_v\| \end{pmatrix} \\
&\quad + [\|e_p\|\|e_v\|] \begin{pmatrix} \frac{c_1}{m}B & 0 \\ B & 0 \end{pmatrix} \begin{pmatrix} \|e_R\| \\ \|e_\omega\| \end{pmatrix}.
\end{aligned} \tag{4.20}$$

### 4.2.3 Complete Dynamics Stability

Consider, $V = V_1 + V_2$ for a Lyapunov candidate function of complete dynamics. Then, let $z_1 = [||e_p||, ||e_v||]^T$, $z_2 = [||e_R||, ||e_\omega||]^T$, and

$$
\begin{aligned}
M_1 &= \frac{1}{2}\begin{pmatrix} k_p & -c_1 \\ -c_1 & m \end{pmatrix}, M_2 = \frac{1}{2}\begin{pmatrix} k_R & -c_2 \\ -c_2 & \lambda_{min}(J) \end{pmatrix}, \\
M_3 &= \frac{1}{2}\begin{pmatrix} k_p & c_1 \\ c_1 & m \end{pmatrix}, M_4 = \frac{1}{2}\begin{pmatrix} \frac{2k_R}{2-\psi} & c_2 \\ c_2 & c_3\lambda_{max}(J) \end{pmatrix}, \\
W_1 &= \begin{pmatrix} (1-\alpha)\frac{c_1}{m}k_p & -\frac{1}{2}(\frac{c_1}{m}k_v + \frac{c_1}{m}\alpha k_v + \alpha k_p) \\ -\frac{1}{2}(\frac{c_1}{m}k_v + \frac{c_1}{m}\alpha k_v + \alpha k_p) & (1-\alpha)k_v - c_1 \end{pmatrix}, \\
W_2 &= \begin{pmatrix} \frac{c_1}{m}B & 0 \\ B & 0 \end{pmatrix}, W_3 = \begin{pmatrix} \frac{c_2 k_R}{\lambda_{max}(J)} & \frac{c_2 k_\omega}{2\lambda_{min}(J)} \\ \frac{c_2 k_\omega}{2\lambda_{min}(J)} & c_3 k_\omega - c_2 \end{pmatrix}.
\end{aligned}
$$

Let $M_{12} = \begin{pmatrix} M_1 & \mathbf{0}_{2\times 2} \\ \mathbf{0}_{2\times 2} & M_2 \end{pmatrix}$, $M_{34} = \begin{pmatrix} M_3 & \mathbf{0}_{2\times 2} \\ \mathbf{0}_{2\times 2} & M_4 \end{pmatrix}$, $W = \begin{pmatrix} \lambda_{min}(W_1) & -\frac{1}{2}\|W_2\| \\ -\frac{1}{2}\|W_2\| & \lambda_{min}(W_3) \end{pmatrix}$, $z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$. Then

$$
z_1^T M_1 z_1 + z_2^T M_2 z_2 \leq V \leq z_1^T M_3 z_1 + z_2^T M_4 z_2, \tag{4.21}
$$

$$
z^T M_{12} z \leq V \leq z^T M_{34} z, \tag{4.22}
$$

$$
\begin{aligned}
\dot{V} &\leq -z_1^T W_1 z_1 + z_1^T W_2 z_2 - z_2^T W_3 z_2 \\
&\leq -\|z_1^T\|\|W_1 z_1\| + \|z_1^T\|\|W_2 z_2\| - \|z_2^T\|\|W_3 z_2\| \\
&\leq -\lambda_{min}(W_1)\|z_1\|^2 + \|z_1\|\|W_2\|\|z_2\| - \lambda_{min}(W_3)\|z_2\|^2 \\
&\leq -[\|z_1\|, \|z_2\|]\begin{pmatrix} \lambda_{min}(W_1) & -\frac{1}{2}\|W_2\| \\ -\frac{1}{2}\|W_2\| & \lambda_{min}(W_3) \end{pmatrix}\begin{pmatrix} \|z_1\| \\ \|z_2\| \end{pmatrix}.
\end{aligned} \tag{4.23}
$$

Furthermore,

$$
\dot{V} \leq -z^T W z \leq -\lambda_{min}(W)\|z\|^2 \leq -\frac{\lambda_{min}(W)}{\lambda_{min}(M_{12})}V. \tag{4.24}
$$

We seek the conditions for $M_1, M_2, M_3, M_4$ to be positive-definite in order to get exponential stability. It is required that:

$$
\begin{aligned}
\det(M_1) &= \frac{1}{2}(k_p m - c_1^2) > 0, \\
\det(M_2) &= \frac{1}{2}(k_R \lambda_{min}(J) - c_2^2) > 0, \\
\det(M_3) &= \frac{1}{2}(k_p m - c_1^2) > 0, \\
\det(M_4) &= \frac{1}{2}(\frac{2k_R}{2-\overline{\Psi}} \lambda_{max}(J) - c_2^2) > 0,
\end{aligned}
\tag{4.25}
$$

which gives

$$
\begin{aligned}
c_1 &< \sqrt{k_p m}, \\
c_2 &< \sqrt{\frac{2k_R}{2-\overline{\Psi}} \lambda_{max}(J)}.
\end{aligned}
\tag{4.26}
$$

The matrix $W_1$, $W_3$, and $W = \begin{pmatrix} -\lambda_{min}(W_1) & \frac{1}{2}\|W_2\| \\ \frac{1}{2}\|W_2\| & -\lambda_{min}(W_3) \end{pmatrix}$ also need to be positive-definite. Then,

$$
\begin{aligned}
\det(W_1) &= ((1-\alpha)\frac{c_1}{m}k_p)((1-\alpha)k_v - c_1) - \frac{1}{4}((1+\alpha)\frac{c_1}{m}k_v + \alpha k_p)^2 > 0, \\
&\quad - ((1-\alpha)\frac{k_p}{m} + \frac{1}{4}(1+\alpha)^2(\frac{k_v}{m})^2)c_1^2 + (\frac{1}{2}\alpha^2 - \frac{5}{2}\alpha + 1)\frac{k_p k_v}{m}c_1 - \frac{1}{4}\alpha^2 k_p^2 > 0, \\
\det(W_3) &= (\frac{c_2 k_R}{\lambda_{max}(J)})(k_\omega - c_2) - (\frac{c_2 k_\omega}{2\lambda_{min}(J)})^2 > 0, \\
\det(W) &= \lambda_{min}(W_1)\lambda_{min}(W_3) - \frac{1}{4}\|W_2\|^2 > 0,
\end{aligned}
\tag{4.27}
$$

which gives

$$
\begin{aligned}
c_1 &< \frac{(k_p - \alpha k_p)(k_v - \alpha k_v)}{k_p - \alpha k_p + \frac{{k_v}^2}{4m}}, \\
c_2 &< \frac{\frac{k_R k_\omega}{\lambda_{max}(J)}}{(\frac{k_R}{\lambda_{max}(J)} + \frac{{k_\omega}^2}{4\lambda_{min}(J)})}.
\end{aligned}
\tag{4.28}
$$

### 4.2.4 Error Bound for $\|e_p\|$

By assuring $M_2$ is positive-definite, we have

$$\begin{aligned} z_1^T M_1 z_1 + z_2^T M_2 z_2 &\geq z_1^T M_1 z_1 \\ &\geq \lambda_{min}(M_1)\|z_1\|^2 \\ &\geq \lambda_{min}(M_1)\|e_p\|^2. \end{aligned} \tag{4.29}$$

Since $\dot{V}(t)$ is negative-definite for all $t \geq 0$, then we know $V(t) \leq V(0) \forall t \geq 0$. Eq. (4.21) gives us $z_1^T M_1 z_1 + z_2^T M_2 z_2 \leq V \leq V(0)$. Hence,

$$\begin{aligned} \lambda_{min}(M_1)\|e_p\|^2 &\leq V(0), \\ \|e_p\| &\leq \sqrt{\frac{V(0)}{\lambda_{min}(M_1)}}, \end{aligned} \tag{4.30}$$

where

$$\begin{aligned} V(0) =& \frac{1}{2}k_p\|e_p(0)\|^2 + \frac{1}{2}m\|e_v(0)\|^2 + c_1 e_p(0)\cdot e_v(0) \\ &+ \frac{1}{2}e_\omega(0)\cdot J e_\omega(0) + k_R\Psi(R(0), R_d(0)) + c_2 e_R(0)\cdot e_\omega(0), \end{aligned} \tag{4.31}$$

$$M_1 = \frac{1}{2}\begin{pmatrix} k_p & -c_1 \\ -c_1 & m \end{pmatrix}. \tag{4.32}$$

Then we have the following bound for position error:

$$\|e_p\|_{max} = \sqrt{\frac{V(0)}{\lambda_{min}(M_1)}}. \tag{4.33}$$

## 4.3 Asymptotic Stability Analysis

The following is the improved stability analysis, proposed in [26].

**Theorem 4.3.1.** Consider the error system (2.29)–(2.32). Let $I \subset \mathbb{R}_+$ be an interval with 0 as the left endpoint, and let $t \in I$. Assume

$$|ge_3 + \ddot{p}_d(t)| \leq a_{\max}, \tag{4.34}$$

for all $t \in I$ and some specified $a_{\max} \in \mathbb{R}^3_+$. Let $\alpha_\psi \in ]0,1[$ and $\overline{\Psi} \in ]0,2[$ be specified parameters and assume the following conditions hold:

$$\Psi(0) \leq \alpha_\psi \overline{\Psi}, \tag{4.35}$$

$$\frac{1}{2} e_\omega^T(0) J e_\omega(0) \leq k_R(1-\alpha_\psi)\overline{\Psi}. \tag{4.36}$$

Define

$$M_1' = \frac{1}{2}\begin{pmatrix} k_p \mathrm{I}_3 & c_1 \mathrm{I}_3 \\ c_1 \mathrm{I}_3 & m\mathrm{I}_3 \end{pmatrix}, \tag{4.37}$$

$$W_1' = \begin{pmatrix} \frac{c_1 k_p}{m}\mathrm{I}_3 & \frac{c_1 k_v}{2m}\mathrm{I}_3 \\ \frac{c_1 k_v}{2m}\mathrm{I}_3 & (k_v - c_1)\mathrm{I}_3 \end{pmatrix}, \tag{4.38}$$

$$M_{2,1}' = \frac{1}{2}\begin{pmatrix} k_R \mathrm{I}_3 & c_2 \mathrm{I}_3 \\ c_2 \mathrm{I}_3 & J \end{pmatrix}, \; M_{2,2}' = \frac{1}{2}\begin{pmatrix} \frac{2k_R}{2-\overline{\Psi}}\mathrm{I}_3 & c_2 \mathrm{I}_3 \\ c_2 \mathrm{I}_3 & J \end{pmatrix}, \tag{4.39}$$

$$W_2' = \begin{pmatrix} c_2 k_R J^{-1} & \frac{c_2 k_\omega}{2} J^{-1} \\ \frac{c_2 k_\omega}{2} J^{-1} & (k_w - c_2)\mathrm{I}_3 \end{pmatrix}, \tag{4.40}$$

where $c_1$ and $c_2$ are constants, satisfying

$$0 < c_1 < \min\left(\sqrt{k_p m}, \frac{4 m k_p k_v}{k_v^2 + 4 m k_p}\right), \tag{4.41}$$

$$0 < c_2 < \min\left(\sqrt{k_R \lambda_{\min}(J)}, \frac{4\lambda_{\min}(J) k_R k_\omega}{k_\omega^2 + 4\lambda_{\min}(J) k_R}\right). \tag{4.42}$$

Moreover, define

$$z_1'(t) = \begin{pmatrix} e_p(t) \\ e_v(t) \end{pmatrix}, \; z_2'(t) = \begin{pmatrix} e_R(t) \\ e_\omega(t) \end{pmatrix}, \; t \in I,$$

and

$$V_1'(t) = z'^T_1(t) M_1' z_1'(t), \tag{4.43}$$

$$V_2'(t) = \frac{1}{2} e_\omega^T(t) J e_\omega(t) + k_R \Psi(t) + c_2 e_R(t) \cdot e_\omega(t), \tag{4.44}$$

$$V'(t) = V_1'(t) + V_2'(t), \; t \in I. \tag{4.45}$$

Then, $M_1', W_1', M_{2,1}', M_{2,2}', W_2'$ are positive definite. Furthermore, for all $t \in I$,

$$z_2'^T(t) M_{2,1}' z_2'(t) \leq V_2'(t) \leq z_2'^T(t) M_{2,2}' z_2'(t), \tag{4.46}$$

$$V_2'(t) \leq V_2'(0) e^{-2\beta t}, \tag{4.47}$$

$$\sqrt{V'(t)} \leq \mathcal{L}(V_1'(0), V_2'(0), t), \tag{4.48}$$

where, for $x, y, t \in \mathbb{R}_+$,

$$\mathcal{L}(x, y, t) = \mathcal{L}_1(x, y, t) + \mathcal{L}_2(y, t), \tag{4.49}$$

$$\mathcal{L}_1(x, y, t) = e^{\frac{\alpha_1 \sqrt{y}}{2\beta}} \sqrt{x + y} e^{-\frac{\alpha_0}{2} t}, \tag{4.50}$$

$$\mathcal{L}_2(y, t) = e^{\frac{\alpha_1 \sqrt{y}}{2\beta}} \frac{\alpha_2 \sqrt{y}}{2} e^{-\frac{\alpha_0}{2} t} \int_0^t e^{(\frac{\alpha_0}{2} - \beta)s} \mathrm{d}s, \tag{4.51}$$

and $\beta$, $\alpha_0$, $\alpha_1$, and $\alpha_2$, are given by

$$\beta = \frac{1}{2} \lambda_{\min}(M_{2,2}'^{-\frac{1}{2}} W_2' M_{2,2}'^{-\frac{1}{2}}), \tag{4.52}$$

$$\alpha_0 = \min\left(\lambda_{\min}(M_1'^{-\frac{1}{2}} W_1' M_1'^{-\frac{1}{2}}), 2\beta\right), \tag{4.53}$$

$$\alpha_1 = \left\| [\frac{c_1}{m} \mathrm{I}_3, \mathrm{I}_3] M_1'^{-\frac{1}{2}} \right\| \left\| [k_p \mathrm{I}_3, k_v \mathrm{I}_3] M_1'^{-\frac{1}{2}} \right\| \left\| [\mathrm{I}_3, 0_{3\times 3}] M_{2,1}'^{-\frac{1}{2}} \right\| \sqrt{\frac{2}{2 - \overline{\Psi}}}, \tag{4.54}$$

$$\alpha_2 = m \left\| a_{\max} \right\| \left\| [\frac{c_1}{m} \mathrm{I}_3, \mathrm{I}_3] M_1'^{-\frac{1}{2}} \right\| \left\| [\mathrm{I}_3, 0_{3\times 3}] M_{2,1}'^{-\frac{1}{2}} \right\| \sqrt{\frac{2}{2 - \overline{\Psi}}}. \tag{4.55}$$

*Proof.* See [26]. □

A direct consequence of the results above is as follows:

**Corollary 1.** *Consider the error system (2.29)–(2.32). Assume (4.34), (4.35), and (4.36) hold. Let the functions $V_1$, $V_2$, and $V$ be defined as in (4.43)–(4.45), where conditions (4.41) and (4.42) hold. Define, for $x, y \in \mathbb{R}_+$,*

$$\mathcal{L}_u(x, y) = \max_{t \in \mathbb{R}_+} \mathcal{L}(x, y, t) = \mathcal{L}(x, y, t_m(x, y)), \tag{4.56}$$

*where $t_m$ is given by*

$$t_m(x,y) = \begin{cases} \max\left(\frac{1}{\frac{\alpha_0}{2}-\beta}\ln\left(\frac{\frac{\alpha_2\beta\sqrt{y}}{2\beta-\alpha_0}}{\frac{\alpha_0}{2}\sqrt{x+y}+\frac{\alpha_0\alpha_2\sqrt{y}}{2(2\beta-\alpha_0)}}\right),0\right), & \frac{\alpha_0}{2}\neq\beta,\ y>0, \\ \max\left(\frac{2(\alpha_2\sqrt{y}-\alpha_0\sqrt{x+y})}{\alpha_0\alpha_2\sqrt{y}},0\right), & \frac{\alpha_0}{2}=\beta,\ y>0, \\ 0,\ \textit{otherwise.} \end{cases} \tag{4.57}$$

*Then,*

$$\sqrt{V(t)} \leq \mathcal{L}_u(V_1(0),V_2(0)), t\in I.$$

The uniform bound derived above can be used to estimate the deviation of the position and velocity from their desired values as follows:

**Corollary 2.** *Consider the error system (2.29)–(2.32). Assume (4.34), (4.35), and (4.36) hold. Let the functions $V_1$, $V_2$, and $V$ be defined as in (4.43)–(4.45), where conditions (4.41) and (4.42) hold. Define, for $x,y\in\mathbb{R}_+$,*

$$\mathcal{L}_p(x,y) = \left\|[\mathrm{I}_3,0_{3\times3}]{M'_1}^{-\frac{1}{2}}\right\|\mathcal{L}_u(x,y), \tag{4.58}$$

$$\mathcal{L}_v(x,y) = \left\|[0_{3\times3},\mathrm{I}_3]{M'_1}^{-\frac{1}{2}}\right\|\mathcal{L}_u(x,y), \tag{4.59}$$

$$\mathcal{L}_f(x,y) = \left\|[k_p\mathrm{I}_3,k_v\mathrm{I}_3]{M'_1}^{-\frac{1}{2}}\right\|\mathcal{L}_u(x,y), \tag{4.60}$$

*then, for all $t\in I$,*[1]

$$\begin{aligned} \|e_p(t)\| &\leq \mathcal{L}_p(V_1(0),V_2(0)), \\ \|e_v(t)\| &\leq \mathcal{L}_v(V_1(0),V_2(0)), \\ \|k_pe_p(t)+k_ve_v(t)\| &\leq \mathcal{L}_f(V_1(0),V_2(0)). \end{aligned}$$

**Corollary 3.** *Let $\alpha_\psi\in]0,1[$, $\overline{\Psi}\in]0,2[$, and $\overline{\mathcal{V}}_1\in]0,\infty[$ be given and $\bar{\mathcal{V}}_2$ is computed by*

$$\overline{\mathcal{V}}_2 = \left(k_R + 2c_2\sqrt{\frac{k_R}{\lambda_{\max}(J)}\alpha_\psi(1-\alpha_\psi)}\right)\overline{\Psi}. \tag{4.61}$$

*Assume there exists a finite time $T>0$ and a four-times continuously differentiable*

[1] As $\mathcal{L}$, given in equation (4.49), is monotonically increasing with respect to its first two arguments, then it follows, using the definition of $\mathcal{L}_u$, given by equation (4.56), that $\mathcal{L}_u$ is monotonically increasing with respect to its two arguments. Therefore, $\mathcal{L}_p$, $\mathcal{L}_u$, and $\mathcal{L}_f$, given by (4.58), (4.59), and (4.60), respectively, are also monotonically increasing with respect to their arguments.

$p_d\colon [0,T] \to \mathbb{R}^3$ *satisfying* (4.34)*, and* $m\ddot{p}_{d,3}(t) \geq \mathcal{L}_f(\overline{\mathcal{V}}_1, \overline{\mathcal{V}}_2) - mg + \varepsilon$*,* $t \in [0,T]$*, for some* $\varepsilon > 0$*. Then, for any initial condition* $(p(0), v(0), R(0), \omega(0))$ *satisfying*

$$\begin{aligned} \Psi(0) &\leq \alpha_\psi \overline{\Psi}, \\ \frac{1}{2} e_\omega^{\intercal}(0) J e_\omega(0) &\leq k_R(1-\alpha_\psi)\overline{\Psi}, \\ V_1(0) &\leq \overline{\mathcal{V}}_1, \end{aligned} \tag{4.62}$$

*Then, for all* $t \in [0,T]$*,*

$$\begin{aligned} \|e_p(t)\| &\leq \mathcal{L}_p(\overline{\mathcal{V}}_1, \overline{\mathcal{V}}_2), \\ \|e_v(t)\| &\leq \mathcal{L}_v(\overline{\mathcal{V}}_1, \overline{\mathcal{V}}_2), \\ \|k_p e_p(t) + k_v e_v(t)\| &\leq \mathcal{L}_f(\overline{\mathcal{V}}_1, \overline{\mathcal{V}}_2), \end{aligned}$$

*where* $\mathcal{L}_p$*,* $\mathcal{L}_v$*, and* $\mathcal{L}_f$ *are given by* (4.58)*,* (4.59)*, and* (4.60)*, respectively.*

## 4.4 Autotuning Algorithm

In this section, we formulate gain tuning into a bound-constrained optimization problem and adopt simulated annealing method to find the optimal solution. This approach systematically searches for the most effective control gains for any quadrotors assume the mass and inertia matrix are known.

### 4.4.1 Problem Formulation

Eq.(4.58) establishes a theoretical bound on position error $e_p$. Then, since

$$V_1(0) \leq \overline{\mathcal{V}}_1, V_2(0) \leq \overline{\mathcal{V}}_2,$$

we have

$$\|e_p(t)\| \leq \mathcal{L}_p(\overline{\mathcal{V}}_1, \overline{\mathcal{V}}_2),\ t \in I.$$

While our theoretical result implies local exponential stability of the closed-loop quadrotor dynamics for any choice of positive control gains, that choice should ensure that the theoretical uniform bounds, which depend implicitly on the control gains, are not too conservative. In particular, we want the gains choice to result in small values for the uniform bounds $\mathcal{L}_p(\overline{\mathcal{V}}_1, \overline{\mathcal{V}}_2)$, where $\overline{\mathcal{V}}_2$ is computed according to (4.61).

Let $\gamma_1, \gamma_2 \in ]0,1[$ be parameters that determine the values of $c_1$ and $c_2$, used in the definitions of the functions $V_1$ and $V_2$ in (4.43) and (4.44), respectively, through the relations

$$c_1 = \gamma_1 \min\left(\sqrt{k_p m}, \frac{4mk_pk_v}{k_v^2 + 4mk_p}\right),$$
$$c_2 = \gamma_2 \min\left(\sqrt{k_R \lambda_{min}(J)}, \frac{4\lambda_{min}(J)k_Rk_\omega}{k_\omega^2 + 4\lambda_{min}(J)k_R}\right).$$

The above relations ensure that conditions (4.41) and (4.42) hold. Let $\underline{k}, \overline{k} \in \mathbb{R}_+ \setminus \{0\}$ be user-defined positive lower and upper bounds on the control gains, respectively, and $w_1$, $w_2$, and $w_3$ be positive weights to be assigned to the unifrom bounds during the optimization process. The control gains, and the parameters $\gamma_1$ and $\gamma_2$ are then determined by solving the following nonlinear optimization problem, where the uniform bounds $\mathcal{L}_p(\overline{\mathcal{V}}_1, \overline{\mathcal{V}}_2)$ are functions of the gains through the relations (4.56), (4.58), and (4.47) and the arguments $\overline{\mathcal{V}}_1$ and $\overline{\mathcal{V}}_2$ are dropped:

$$\begin{aligned} \min_{k_p,k_v,k_R,k_\omega,\gamma_1,\gamma_2} \quad & \mathcal{L}_p(\overline{\mathcal{V}}_1, \overline{\mathcal{V}}_2), \\ \text{s.t.} \quad & \underline{k} \leq k_p, k_v, k_R, k_\omega \leq \overline{k}, \\ & 0 < \gamma_1, \gamma_2 < 1. \end{aligned} \tag{4.63}$$

### 4.4.2 Simulated Annealing

For the optimization problem (4.63), usually the domain for searching the control gains, i.e. $[\underline{k}, \overline{k}]$, is small. Notice that (4.63) is bound-constrained; it is natural to adopt simulated annealing as our global optimization method because it has a high probability of finding an optimal solution in a short period of time when the domain is small. However, other optimization methods like multi-start are also suitable for solving such problems.

Simulated annealing [13] is a stochastic global search optimization algorithm proposed by Kirkpatrick, Gelatt, and Vecchi in 1983. Although it is not guaranteed to find the global minimum, it increases the likelihood of finding a near-optimal solution. The basic idea of simulated annealing is to mimic the process of annealing in metallurgy, where a material is heated and then gradually cooled to reach a low-energy crystalline state. Similarly, in simulated annealing, the algorithm starts with a high "temperature" for random exploration of solutions and gradually cools, becoming more selective. It iteratively explores neighboring solutions, probabilistically accept-

ing or rejecting them based on quality and temperature. This process continues until either a maximum number of iterations is reached or no significant improvement in the objective function is observed over several iterations.

Our objective function, i.e., $\mathcal{L}p(\overline{\mathcal{V}}1, \overline{\mathcal{V}}_2)$, is a highly nonlinear function with many local minima. Since the only constraints are bound constraints, and the search domain is not very large, simulated annealing can find the global minimum in a short time with high probability. The proposed algorithm is as follows:

**Algorithm 3** Simulated annealing for gain tuning

$k_{\text{init}} \leftarrow (k_{p_{\text{init}}}, k_{v\text{init}}, k_{R\text{init}}, k_{\omega\text{init}}, \gamma_{1_{\text{init}}}, \gamma_{2_{\text{init}}})$
Initialize current solution $k_{\text{current}} \leftarrow k_{\text{init}}$
$i \leftarrow 0$, $\Delta L \leftarrow \mathcal{L}_p(k_{\text{init}})$
**while** $i \leq N_{\max}$ or $\Delta L > \delta$ **do**
    $k_{\text{new}} = \text{Neighbor}(k_{\text{current}})$
    $L_{\text{current}} = \mathcal{L}_p(k_{\text{current}})$
    $L_{\text{new}} = \mathcal{L}_p(k_{\text{new}})$
    $\Delta L = L_{\text{new}} - k_{\text{current}}$
    **if** $\Delta L < 0$ or $e^{-\Delta L/T} \geq \text{random}(0, 1)$ **then**
        $k_{\text{current}} \leftarrow k_{\text{new}}$
    **end if**
    $k \leftarrow k + 1$
**end while**
**return** $k_{\text{current}}$

# Chapter 5

# Experiments

In this chapter, we present the results from MATLAB and Webots simulations, followed by detailed instructions for real experiments with the Optitrack mocap system and the micro-quadrotor Crazyflie.

## 5.1 MATLAB Simulations

Numerical quadrotor simulations using MATLAB[1], incorporating the proposed control synthesis approach, are conducted to demonstrate its performance and effectiveness. For the purpose of consistency, all calculations and computations are done on a computer with an i7-12700 CPU. To ensure the simulations are reproducible, we configure the MATLAB control random number generator (rng) to its "default" setting. This ensures consistent results each time the simulations are executed. The hyper-rectangular plots presented in this section were obtained using the MATLAB command `plotcube` [23]. The parameters of the quadrotor match those of the Crazyflie 2.1 and are as follows:

$$J = \text{diag}([2.31, 2.31, 4.125]) \times 10^{-5} \text{ kg} \cdot \text{m}^2, \ m = 0.033 \text{ kg}.$$

[1]The code for all MATLAB simulations described in Section 5.1 is available on GitHub: https://github.com/BerenChang/quadrotor-safe-synthesis

### 5.1.1 Reach-Avoid Task Setup

We consider a reach-avoid control scenario, where the operating domain is defined as a cube with edges measuring five meters each. The initial nominal position of the quadrotor is $p_0 = [0.5, 0.5, 1]^T$, and the target set is given by $\mathcal{X}\mathrm{t} = [\![[4, 4, 4]^T, [5, 5, 5]^T]\!]$. The unsafe set is given as a union of ten hyper-rectangles. The operating domain, the target, and the unsafe sets are depicted in Figure 5.1. We let $v_{\max} = [1, 1, 1]^T$, $f_{\max} = 2mg = 0.6468$ N, and $a_{\max} = [0.1, 0.1, 9.9]^T$. The control synthesis process starts by setting $\overline{\Psi} = 0.005$, $\alpha_\psi = 0.4$, and $\overline{\mathcal{V}}_1 = 10^{-4}$.

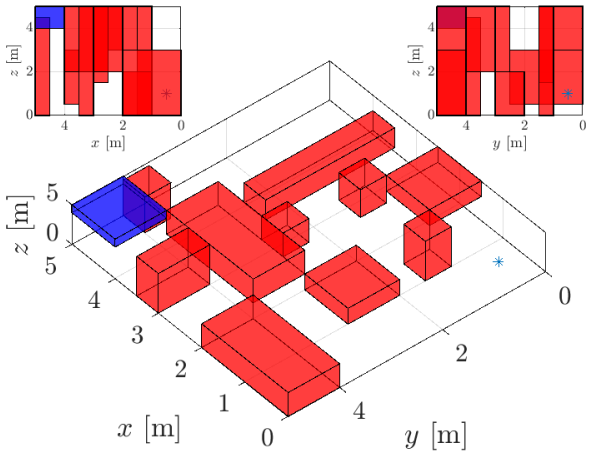


Figure 5.1: The environment with ten obstacles as red boxes, one target set as blue box, and the starting point as blue asterisk. The top right corner shows the top view of the environment.

### 5.1.2 Gain Tuning Through Optimization

To solve the optimization problem (4.63), we set the gain bounds to be $\underline{k} = 0.001$, $\overline{k} = 0.05$. The problem is solved with Simulated Annealing. The MATLAB built-in function `simulannealbnd` is adopted. With the initial guess

$$k_p = 0.05,\ k_v = 0.05,\ k_R = 0.05,\ k_\omega = 0.05,\ \gamma_1 = 0.5,\ \gamma_2 = 0.5,$$

the optimized control gains and coefficients are obtained as

$$k_p = 0.05,\ k_v = 0.03,\ k_R = 0.005,\ k_\omega = 0.001,$$

$$\gamma_1 = 0.5468,\ \gamma_2 = 0.6124.$$

The average computing time for obtaining the gains via optimization, over one hundred runs, is 0.8934 seconds. The resulting value of $\overline{\mathcal{V}}_2 = 7.9976 \times 10^{-5}$ and the resulting uniform bound is $\mathcal{L}_p(\overline{\mathcal{V}}_1, \overline{\mathcal{V}}_2) = 0.2589$ m.

### 5.1.3 Safe Tube and Trajectory Synthesis

Next, we construct a safe tube composed of connected hyper-rectangles, through which the desired trajectory is synthesized, where an RRT is synthesized according Algorithm 1 and a shortest path algorithm is conducted over the RRT. The parameters in Algorithm 1 are chosen to be

$$\alpha = 0.9,\ N_v = 400,\ C_{\text{sampling}} = 0.9.$$

The resulting safe tube consists of fifteen hyper-rectangles depicted in Figure 5.2, where the associated computational time is 0.06 seconds.

Next, we construct the desired trajectory given as piecewise Bezier curve with fifteen segments, where each segment is parameterized by fifteen control points. We implement Algorithm 2 for the trajectory synthesis, where we use the following parameters:

$$\alpha_T = 1.1,\ T_0 = 100.$$

The produced trajectory, with time duration $T = 146.1$ seconds, is depicted in Figure 5.3. Note that the synthesized trajectory lies within the safe tube. The computation of trajectory based on Algorithm 2 required 0.27 seconds of CPU time.

### 5.1.4 Initial Points Generation

Safety of the closed-loop quadrotor system is guaranteed if the initial conditions satisfy (4.62). To visualize the set given in (4.62), initial points are generated as follows. First, we randomly generated initial points $(p(0), v(0), R(0), \omega(0))$ as follows. Each point is computed via means of sampling through the relations $p(0) = \texttt{sample}(0.2[\![-1_3, 1_3]\!])$, $v(0) = \texttt{sample}(0.05[\![-1_3, 1_3]\!])$, $R(0) = e^{(\texttt{sample}(0.05[\![-1_3,1_3]\!]))^\wedge}$,

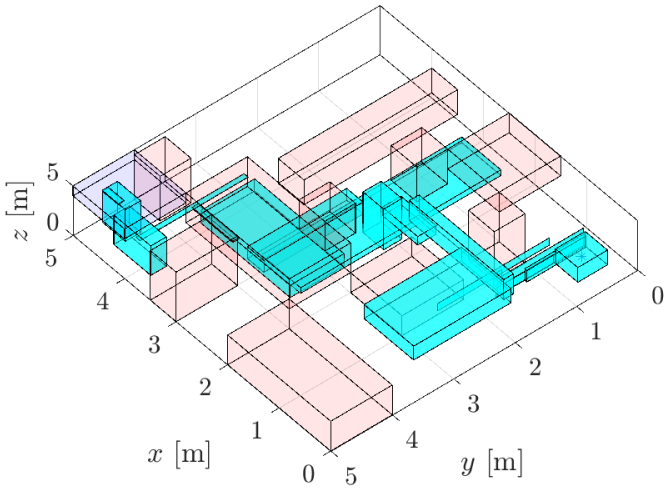


Figure 5.2: The safe hyper-rectangles are shown as blue boxes.

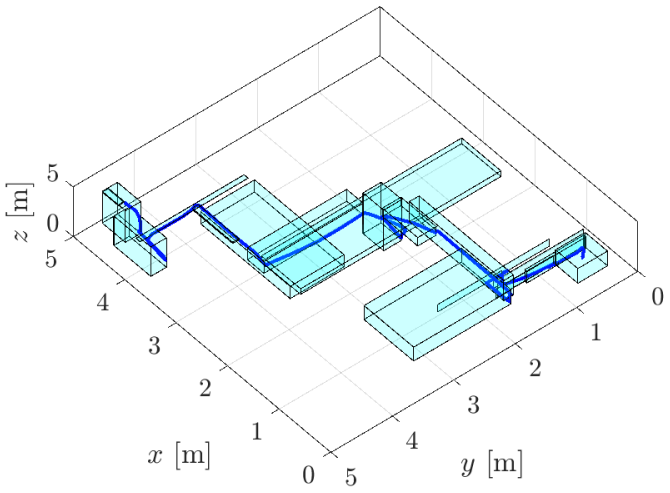


Figure 5.3: The generated trajectory is shown as the blue curve.

$\omega(0) = \texttt{sample}(0.05[\![-1_3, 1_3]\!])$, where only the generated points satisfying (4.62) are considered. The two hundred points are illustrated in Figure 5.4. The positions of first fifty points are shown in Figure 5.5 with attitude represented as axis-angle representation. The first twenty initial points are adopted to compute trajectories

using the Runge-Kutta method (`RK45` in MATLAB)[2]. The integrated trajectories are depicted in Figure 5.6. All position profiles associated with the generated trajectories from the chosen twenty points, represented by blue lines, originate from the bottom right corner and terminate at the blue box located at the top left corner, remaining within the safety tube throughout, thus validating the safety assurances of the tracking controller.

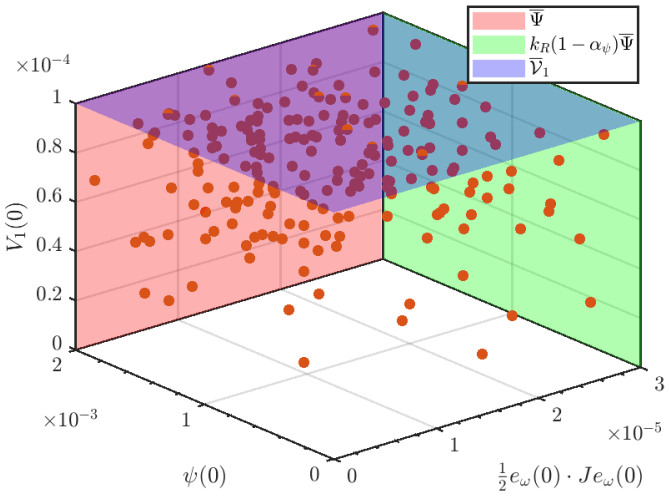


Figure 5.4: The values of $\psi(0)$, $e_\omega(0)$, and $V_1(0)$ of the generated two hundreds initial points are shown to stay within theoretical bounds.

To better visualize the shape of the initial set satisfying (4.62), the initial position error is sampled through the relation $e_p(0) = \texttt{sample}(0.08[\![-1_3, 1_3]\!])$, with the initial velocity error $e_v(0)$ attitude error $e_R(0)$, and angular velocity error $e_\omega(0)$ all set to zero. One million points are sampled, with safe points marked red and unsafe points marked blue, as illustrated in Figure 5.7. The cross-sectional figures shows the boundary between safe region and unsafe region, as represented by the sample points.

[2]The Runge-Kutta method does not guarantee the preservation of the $\mathsf{SO}(3)$ structure of the attitude $R$ during numerical integration. However, the Runge-Kutta method provides convergence guarantees which motivates using it for the purpose of simulations in this section.

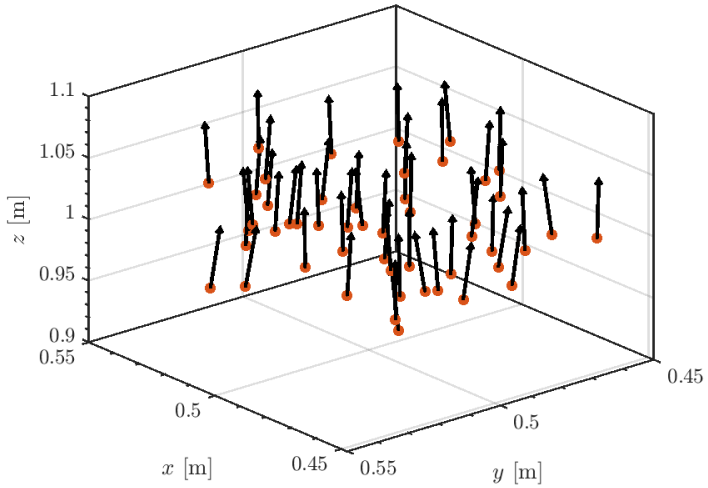


Figure 5.5: The position and attitude of fifty initial points. The arrows represent the directions of the z-axis with respect to the body frame (i.e., $b_3(0)$).

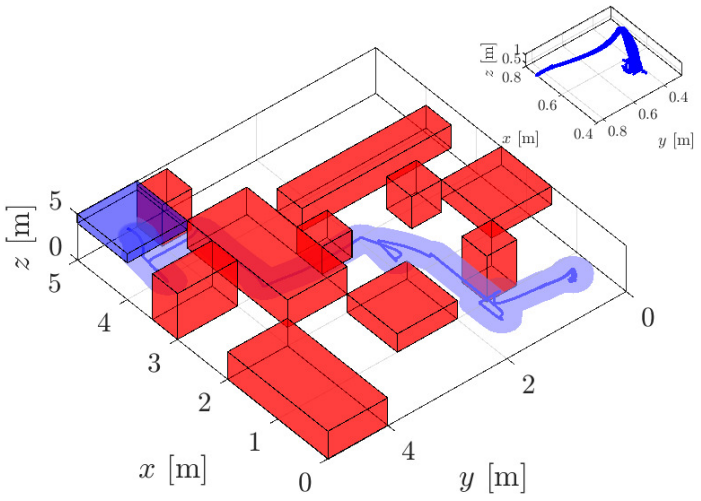


Figure 5.6: Tracking trajectories going through obstacles. The red boxes are obstacles. The blue box is the target set. The blue tube is the region with guarantees.

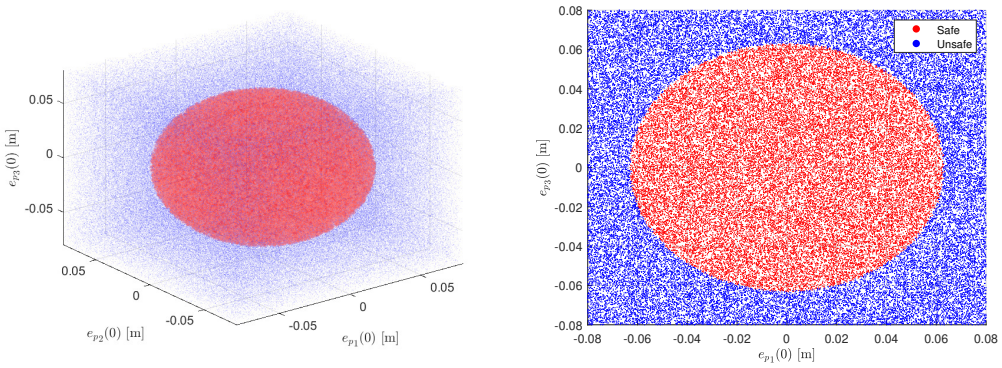


Figure 5.7: The shape of initial set of position, approximated by one million points. The red points are safe points that satisfy (4.62). On the left is the 3D plot and on the right is the cross section at $e_{p_2} = 0$.

### 5.1.5 Validation of the Tracking Performance

To demonstrate the effectiveness of the proposed framework and the safety guarantees for the generated twenty trajectories, we use Figure 5.8 to show that the position error and the velocity error stay within the theoretical bounds $\mathcal{L}p(\overline{\mathcal{V}}1, \overline{\mathcal{V}}2)$ and $\mathcal{L}v(\overline{\mathcal{V}}1, \overline{\mathcal{V}}2)$. The simulations demonstrate that when (4.62) is fulfilled, the quadrotor performs as expected and the position errors remain perfectly within the theoretical threshold, validating the safety guarantees of the proposed framework.

## 5.2 Webots Simulations

Webots[3] is an open-source 3D robot simulator. Simulations conducted in the Webots environment on an Ubuntu platform enable the creation of a "digital twin" of the Crazyflie 2.1, achieved through the use of a mesh file supplied by Bitcraze. To ensure methodological consistency and facilitate comparative analysis with MATLAB simulations, an environment mirroring that of the MATLAB simulation setup is employed.

Within the Webots simulations, the quadrotor is initially tasked with navigating to and maintaining a hover at a predetermined starting point, i.e., $[0.5, 0.5, 1]^T$ in our

[3] https://cyberbotics.com

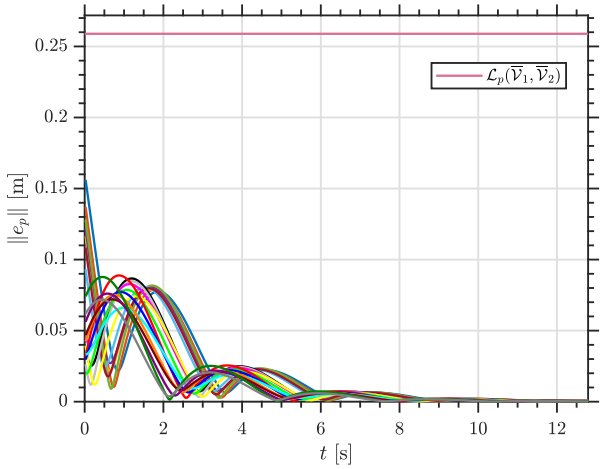


Figure 5.8: For all $t \in [0, T]$, $\|e_p\|$ remains within the theoretical bound. Data is only shown for the first 13 seconds to demonstrate the details of convergence, however the bounds are still respected for all $t \in [0, T]$.

task. Subsequently, tracking maneuvers are initiated upon the fulfillment of specific initial conditions:

$$V_1(0) \leq 1.0 \times 10^{-4},$$
$$\psi(0) \leq 2.0 \times 10^{-3},$$
$$\frac{1}{2} e_\omega(0)^T J e_\omega(0) \leq 3.0 \times 10^{-5}.$$

The norm of position error is in Figure 5.10. It is very similar to the position error from MATLAB simulations plotted in Figure 5.8. However, due to stochastic perturbations in Webots (from Crazyflie mesh file assembly, delay of propeller rotations, etc.), the tracking performance still has small position errors. Nevertheless, the position error remains within the theoretical bounds, demonstrating the robustness of the proposed framework. Figure 5.9 is a combination of fifteen sequential snapshots capturing the tracking performance of the Crazyflie quadrotor. It illustrates how the quadrotor flies through gaps and finally reaches the target safely.

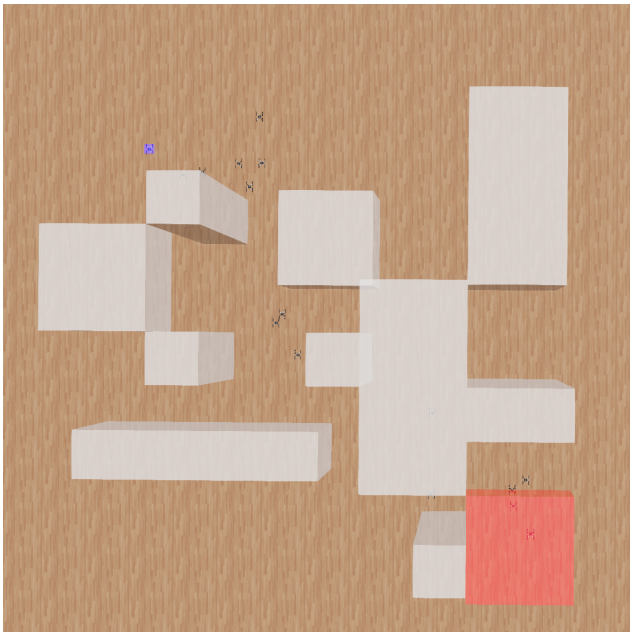

Figure 5.9: Combination of fifteen screenshots of Crazyflie in Webots (view from top), showing the valid reach-avoid tracking performance. The blue box is the initial position and the red box is the target set.

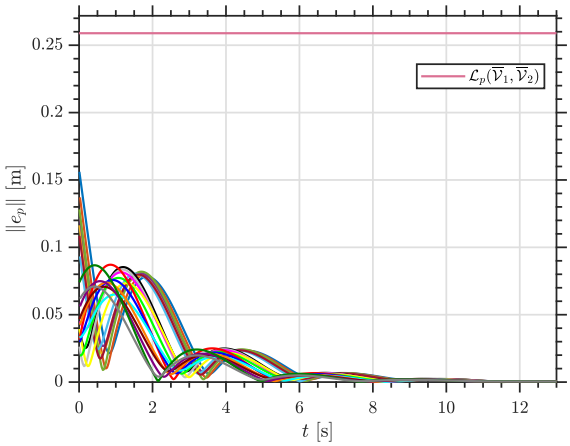


Figure 5.10: The norm of position error of twenty tracking trajectories in Webots simulations. The error remains within the theoretical bounds. Data is only shown for the first 13 seconds, however the bound is still respected for all $t \in [0, T]$.

## 5.3 Real Experiments

Some real experiments are done in this section. The results will be shown first for a better connection with previous sections, followed by detailed instructions on experimental settings and programming.

### 5.3.1 Experimental Results

Based on the lab environment, we designed a small version of a reach-avoid task. The operating domain is $[-1, 1] \times [-1, 1] \times [0, 1.5]$. Initially, the quadrotor will be commanded to hover at $[-0.5, -0.5, 1]^T$. It will then try to avoid an obstacle occupying $[-0.2, 0] \times [-0.2, 0] \times [0, 1.5]$, and eventually reach the region $[0.2, 0.8] \times [0.2, 0.8] \times [0.9, 1.5]$. Under the same parameters mentioned in Section 5.1.2, we run ten tracking trajectories using Crazyswarm[4] with Optitrack[5] and Python commands with Bezier curves (introduced in the following subsections). The position errors are recorded and illustrated in Figure 5.11, validating the effectiveness of the proposed control synthesis. One of the ten tracking performances is recorded and available on YouTube[6] for future reference.

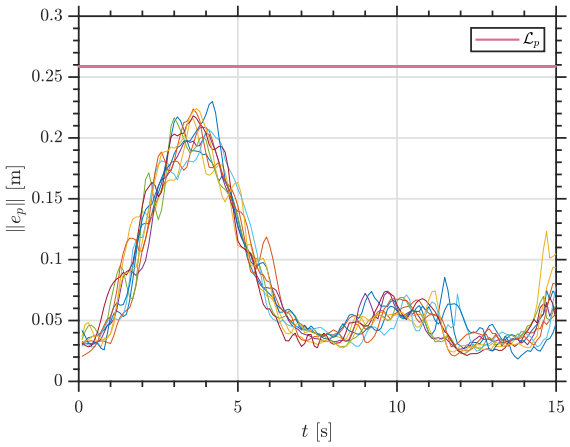


Figure 5.11: The norm of position error for ten tracking trajectories in real experiments.

[4] https://crazyswarm.readthedocs.io/en/latest/
[5] https://optitrack.com/
[6] https://youtu.be/wMMqXJsCATg

### 5.3.2 Experimental Settings

For experiments in the lab, we typically need a feedback controller, which requires estimates of the quadrotor states. Although the onboard sensors of the Crazyflie perform well in measuring velocity and rotational velocity, their estimation of position and attitude cannot be fully trusted because the real-time numerical integration has cumulative errors that can cause the quadrotor to drift. Instead, we use the Optitrack positioning system, which consists of several cameras and the mocap software Motive. By capturing the position of the attached markers on the quadrotor, the system can provide position and attitude estimates within milliseconds. The mocap system can be treated as a small prototype of the global positioning system (GPS). Thus, the same control loop can be extended to quadrotors equipped with GPS modules.

For the micro-quadrotor Crazyflie, there are two Crazyflie 2.0 and one Crazyflie 2.1 in the lab, but the Crazyflie 2.1 is more usable than the 2.0. To control the Crazyflie, we use Ubuntu 20.04 and ROS Noetic. The Crazyradio is attached to the computer to ensure better communication with the Crazyflie. The GitHub repository Crazyswarm, an open-source ROS package, is chosen as the platform to execute our controller. Both Python and C++ are supported in Crazyswarm, but C++ is recommended for better performance.

Since Motive is only available for Windows and ROS is better supported on Linux, the lab has two computers: one running Motive to receive state estimates from the Optitrack cameras, and the other running ROS to calculate and send real-time control commands. Changes to the versions of Motive, Ubuntu, and ROS are not recommended due to compatibility issues. The two computers communicate via an Ethernet cable.

### 5.3.3 Gap between Simulations and Real Experiments

In simulations, we assume the linear transformation in Section 2.1 holds. However, since the Crazyflie uses brushed motors, the relationship between the PWM signal and RPM is nonlinear. Consequently, the PWM signal, which is actually used to drive the brushed motors, and thrust also have a nonlinear relationship. Fortunately, Bitcraze has measured the thrust generated under different PWM signals as follows.

We approximate the relationship between PWM and thrust by imposing a linear constraint that passes through the origin. By minimizing the mean square error, we establish the relationship: PWM (digital) $= 1.2 \times 10^5$ thrust (N). The linear

| PWM (%) | 0 | 6.25 | 12.5 | 18.75 | 25 | 31.25 | 37.5 | 43.25 |
|---|---|---|---|---|---|---|---|---|
| Thrust (gram) | 0.0 | 1.6 | 4.8 | 7.9 | 10.9 | 13.9 | 17.3 | 21.0 |
| PWM (%) | 50 | 56.25 | 62.5 | 68.75 | 75 | 81.25 | 87.5 | 93.75 |
| Thrust (gram) | 24.4 | 28.6 | 32.8 | 37.3 | 41.7 | 46.0 | 51.9 | 57.9 |

Table 5.1: Data points collected by Bitcraze for estimating the thrust under different PWM signal.

regression is visualized in 5.12. This approximation is effective within the range of thrust 0 - 0.5 N and PWM signal 0 - 60000.

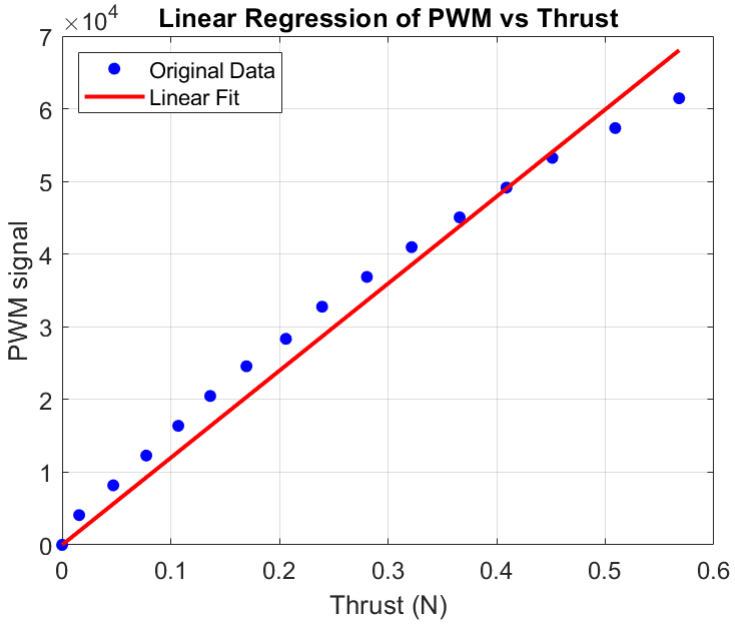


Figure 5.12: Linear regression between PWM signal and thrust. PWM is converted from percent (0% to 100%) to binary (0 to $2^{16}$-1).

Let $\kappa = 1.2\times10^5$. Let $(f, \tau)$ be the control inputs in MATLAB simulations, $(f', \tau')$ in PWM signal be the control inputs in real experiments. Then the relationship between $k_1$ and $k_2$ is

$$
\begin{aligned}
\begin{bmatrix} f' \\ \tau'_x \\ \tau'_y \\ \tau'_z \end{bmatrix} &= \kappa \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ d & 0 & -d & 0 \\ -c_{\tau f} & c_{\tau f} & -c_{\tau f} & c_{\tau f} \end{bmatrix} \begin{bmatrix} \frac{1}{4} & -\frac{\sqrt{2}}{4d} & \frac{\sqrt{2}}{4d} & \frac{1}{4c_{\tau f}} \\ \frac{1}{4} & -\frac{\sqrt{2}}{4d} & -\frac{\sqrt{2}}{4d} & -\frac{1}{4c_{\tau f}} \\ \frac{1}{4} & \frac{\sqrt{2}}{4d} & -\frac{\sqrt{2}}{4d} & \frac{1}{4c_{\tau f}} \\ \frac{1}{4} & \frac{\sqrt{2}}{4d} & \frac{\sqrt{2}}{4d} & -\frac{1}{4c_{\tau f}} \end{bmatrix} \begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \\
&= \kappa \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} f \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}.
\end{aligned}
\tag{5.1}
$$

That relationship is reflected in the code of the Crazyflie firmware[7].

### 5.3.4 Optitrack Mocap System

The mocap system consists of two parts: Optitrack cameras with Optihub (shown in Figure 5.13) and a Windows computer with Motive software. Since the accuracy of the system drifts over time (due to cameras being accidentally moved, etc.), it is strongly suggested to calibrate the system with the CWM-250 Calibration Wand and CS-200 Calibration Square in the lab before every experiment. Be careful when using the calibration square, as it sets the $x$-axis and $y$-axis for the world frame. The $x$-axis and $y$-axis of the body frame, i.e., the Crazyflie, need to be aligned with the world frame before conducting the experiment.

The sampling frequency of the mocap system can be adjusted to 30 Hz, 60 Hz, 90 Hz, and 120 Hz in Motive. The Crazyradio, a USB communication device, is used to send the positioning data to the Crazyflie quadrotor. Due to USB hardware constraints, the highest sampling frequency is limited to 100 Hz, and higher sampling frequencies will result in increased communication delay. Therefore, we typically use 30 Hz or 60 Hz to balance between sampling frequency and communication delay.

The Motive streaming settings allow us to choose which IP address we stream data to. Since we currently have no router that allows both internet and local Ethernet to coexist at the same time, we use the local address for local communication. We can choose "192.168.10.4," the address of Ubuntu, as the local interface for streaming.

[7] https://github.com/bitcraze/crazyflie-firmware

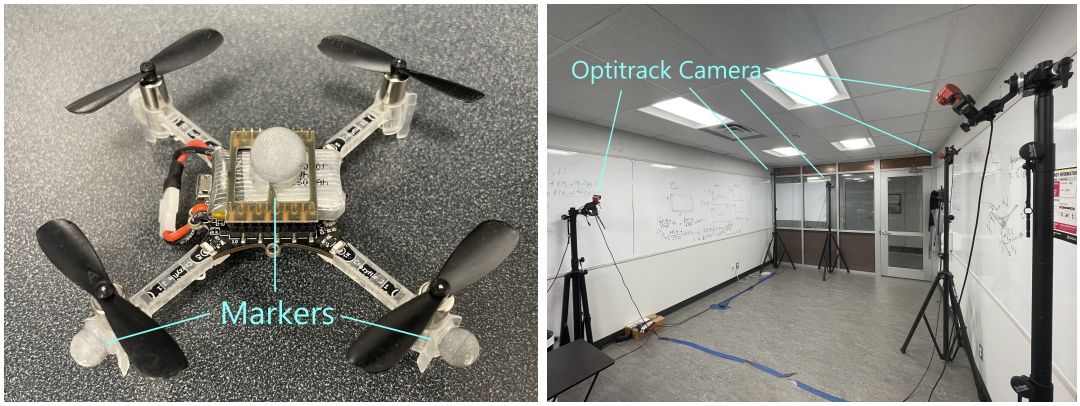


Figure 5.13: The hardware components of Optitrack mocap system. The Optitrack cameras capture the location of markers on the quadrotor and calculate the states in real-time.

Later, we choose "192.168.10.1" (the address of Windows) as the address used by Crazyswarm for accessing the data.

### 5.3.5 Crazyswarm

Crazyswarm [25] is a ROS-based project developed mainly by James A. Preiss and Wolfgang Honig. This project integrates mocap, ROS, and Crazyflie APIs. Since the Motive protocol is opaque to us, Crazyswarm is the best option based on the setup in the lab. In this section, we will introduce various ways to send commands to the Crazyflie using Crazyswarm. The choice of command depends on the needs of each publication.

#### Experiments on Polynomial (Bezier curve) Trajectory

The best way to control the Crazyflie is to upload a polynomial trajectory beforehand and let it execute the trajectory using its onboard controller. In this way, the communication latency is minimized. However, the tradeoff is that the controller is predefined, which means this commanding method only works for simple-structured controllers. The onboard controllers that have already been employed are the PID controller, Mellinger controller, INDI controller, and Brescianini controller. Of course, self-defined controllers also work.

To upload and execute a polynomial trajectory, the following command is needed.

```
swarm = Crazyswarm().allcfs[0]
traj = uav_trajectory.Trajectory()
traj.loadcsv("figure8.csv")
cf.uploadTrajectory(0, 0, traj)  # upload the trajectory
timeHelper.sleep(2.5)
pos = np.array(cf.initialPosition) + np.array([0, 0, 1.0])
cf.goTo(pos, 0, 2.0)  # hover at initial position
cf.startTrajectory(0, timescale=TIMESCALE)  # execute the trajectory
timeHelper.sleep(traj.duration + 2.0)
cf.land(targetHeight=0.06, duration=2.0)
```

Listing 5.1: Python script for executing the polynomial trajectory "figure 8"

### Experiments on Waypoint Trajectory

For testing waypoint trajectories such as the trajectories generated by RRT or signal temporal logic (STL), we use "goTo" command to let Crazyflie reach the waypoints sequentially. A sample Python script is given as follows:

```
for waypoint in waypoints:
    if waypoint.arrival == 0:
        pos = [waypoint.x, waypoint.y, waypoint.z]
        cf.goTo(pos, 0, 2.0)
    elif waypoint.duration > 0:
        timeHelper.sleep(waypoint.arrival - lastTime)
        lastTime = waypoint.arrival
        pos = [waypoint.x, waypoint.y, waypoint.z]
        cf.goTo(pos, 0, waypoint.duration)
cf.land(targetHeight=0.02, duration=2.0)  # land
```

Listing 5.2: Python script for executing a waypoint trajectory

### Experiments on Vector Field

This command can be used to test the effectiveness of a vector field. For example, if a project proposes a method of generating potential field for a reach-avoid task, one can simply take the gradient of the potential field and feed the gradient to Crazyflie to test if it can accomplish the task. Below is an example of Crazyflie following a circular vector field.

```
def goCircle(timeHelper, cf, totalTime, radius, kPosition):
    startTime = timeHelper.time()
    pos = cf.position()
```

```
    startPos = cf.initialPosition + np.array([0, 0, Z])
    center_circle = startPos - np.array([radius, 0, 0])
    while True:
        time = timeHelper.time() - startTime
        omega = 2 * np.pi / totalTime
        vx = -radius * omega * np.sin(omega * time)
        vy = radius * omega * np.cos(omega * time)
        desiredPos = center_circle + radius * np.array(
            [np.cos(omega * time), np.sin(omega * time), 0])
        errorX = desiredPos - cf.position()
        cf.cmdVelocityWorld(np.array([vx, vy, 0] + kPosition *
errorX), yawRate=0)
        timeHelper.sleepForRate(sleepRate)
```

Listing 5.3: Python script for executing a vector field

The command "cf.cmdVelocityWorld" sends the desired velocity to Crazyflie. For any project generating vector fields, even though the vector fields may in continuous form, they have to be saved as a discrete function of position, and the command "cf.cmdVelocityWorld" can be used based on the real-time position of Crazyflie.

### Experiments on Learning-based Control

Learning-based control is the most challenging command as it touches the low level control down to PWM signal for rotors. A modified Crazyswarm package called "crazyswarm_pwm" and a modified Crazyflie firmware called "crazyflie-firmware_pwm" are already in Ubuntu computer. A sample Python script is given below.

```
time_stamp = rospy.Time.now()
time_stamp_secs = time_stamp.secs // 10000
time_stamp_nsecs = time_stamp.nsecs // 100000
cf.cmdPwm(time_stamp_secs, time_stamp_nsecs, pwm1, pwm2, pwm3, pwm4)
```

Listing 5.4: Python script for sending PWM signal

The command "cf.cmdPwm" is self-defined. It sends PWM signal to Crazyflie in a frequency of 30Hz to 50Hz. This command can be used to test learning-based methods such as a neural network based controller or sparse identification of non-linear dynamics (SINDY), where the controller is approximated using a combination of basis functions.

# Chapter 6

# Conclusion

In this thesis, we proposed a control framework for a quadrotor UAV to accomplish reach-avoid tasks with formal safety guarantees, where the standard planning-tracking paradigm is adapted to account for tracking errors. The framework integrates geometric control theory for trajectory tracking and polynomial trajectory generation using Bézier curves, where tracking errors are accounted for during trajectory synthesis. We revisited the stability analysis of the closed-loop quadrotor system under geometric control, where we proved local exponential stability of the tracking error dynamics for any positive control gains and we provided uniform bounds on tracking errors that can be used in planning. We also derived sufficient conditions to be imposed on the desired trajectory to ensure the well-definedness of the closed-loop quadrotor dynamics. The trajectory synthesis involved an efficient algorithm that constructs a safe tube using sampling-based planning and safe hyper-rectangular set computations. The desired trajectory, represented as a piecewise continuous Bézier curve, is computed through the generated safe tube using a heuristic efficient approach, relying on iterative linear programming. We performed extensive MATLAB numerical quadrotor simulations to demonstrate the proposed framework's effectiveness in reach-avoid planning scenarios. Webots simulations are also conducted to demonstrate the robustness of the proposed framework under stochastic perturbations that closely mimic real-life conditions. Instructions for real experiments are also provided as a reference for future studies.

In future work, we plan to integrate the effects of measurement and input noises, as well as disturbances, into the safe planning framework to enhance its applicability in real-world scenarios. This integration will involve developing robust algorithms that can handle uncertainties and ensure reliable performance under varying condi-

tions. Furthermore, we intend to expand the proposed control synthesis to accommodate more complex quadrotor models, such as those including aerodynamic drag and other aerodynamic effects. These enhancements will allow us to more accurately capture the dynamics of quadrotors operating in realistic environments. Moreover, we aim to address more general specifications, including those defined by temporal logics, which will enable the framework to handle a wider range of mission requirements and operational constraints. By incorporating these advanced specifications, we can ensure that the quadrotors not only follow desired trajectories but also adapt to complex temporal and safety constraints. Additionally, we see potential for improving the uniform tracking error bounds by utilizing modified versions of geometric tracking control. Specifically, employing gain matrices instead of scalars will provide more flexibility and precision in the control design. This approach, which involves much more complicated stability analysis, is expected to result in less conservative trajectory synthesis, thereby enhancing the performance and efficiency of the control system. Overall, these advancements will contribute to the development of a more robust and versatile control framework for quadrotors in diverse and challenging environments.

# References


[1] Samir Bouabdallah, Andre Noth, and Roland Siegwart. PID vs LQ control techniques applied to an indoor micro quadrotor. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2451–2456. IEEE, 2004.

[2] Francesco Bullo and Andrew D Lewis. *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*, volume 49. Springer, 2019.

[3] Jing Chen, Kunyue Su, and Shaojie Shen. Real-time safe trajectory generation for quadrotor flight in cluttered environments. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1678–1685. IEEE, 2015.

[4] Shicong Dai, Taeyoung Lee, and Dennis S Bernstein. Adaptive control of a quadrotor UAV transporting a cable-suspended load with unknown mass. In *53rd IEEE Conference on Decision and Control*, pages 6149–6154. IEEE, 2014.

[5] Edsger W Dijkstra. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, pages 287–290. 2022.

[6] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626, 2017.

[7] Michel Fliess, Jean Lévine, Philippe Martin, and Pierre Rouchon. Flatness and defect of non-linear systems: introductory theory and examples. *International journal of control*, 61(6):1327–1361, 1995.

[8] Farhad A Goodarzi, Daewon Lee, and Taeyoung Lee. Geometric control of a quadrotor UAV transporting a payload connected via flexible cable. *International Journal of Control, Automation and Systems*, 13:1486–1498, 2015.

[9] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[10] Xie Heng, David Cabecinhas, Rita Cunha, Carlos Silvestre, and Xu Qingsong. A trajectory tracking LQR controller for a quadrotor: Design and experimental evaluation. In *TENCON 2015-2015 IEEE region 10 conference*, pages 1–7. IEEE, 2015.

[11] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[12] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[13] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[14] Kostas J Kyriakopoulos and George N Saridis. Minimum jerk path generation. In *Proceedings. 1988 IEEE international conference on robotics and automation*, pages 364–369. IEEE, 1988.

[15] Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[16] Taeyoung Lee. Geometric control of quadrotor UAVs transporting a cable-suspended rigid body. *IEEE Transactions on Control Systems Technology*, 26(1):255–264, 2017.

[17] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Control of complex maneuvers for a quadrotor UAV using geometric methods on SE (3). *arXiv preprint arXiv:1003.2005*, 2010.

[18] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Geometric tracking control of a quadrotor UAV on SE (3). In *49th IEEE conference on decision and control (CDC)*, pages 5420–5425. IEEE, 2010.

[19] Taeyoung Lee, Koushil Sreenath, and Vijay Kumar. Geometric control of cooperating multiple quadrotor UAVs with a suspended payload. In *52nd IEEE conference on decision and control*, pages 5510–5515. IEEE, 2013.

[20] Jun Li and Yuntang Li. Dynamic analysis and PID control for a quadrotor. In *2011 IEEE International Conference on Mechatronics and Automation*, pages 573–578. IEEE, 2011.

[21] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011.

[22] Adam Morawiec. *Orientations and rotations*. Springer, 2003.

[23] Oliver. PLOTCUBE. MATLAB Central File Exchange: https://www.mathworks.com/matlabcentral/fileexchange/15161-plotcube, 2024. [Online; accessed May, 2024].

[24] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7):691–699, 2010.

[25] James A Preiss, Wolfgang Honig, Gaurav S Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3299–3304. IEEE, 2017.

[26] Mohamed Serry, Haocheng Chang, and Jun Liu. Reach-Avoid Control Synthesis for a Quadrotor UAV with Safety Guarantees. *arXiv preprint arXiv:2405.20502*, 2024.

[27] Mohamed Serry, Liren Yang, Necmiye Ozay, and Jun Liu. Safe Tracking Control of Discrete-Time Nonlinear Systems Using Backward Reachable Sets. In *2024 American Control Conference (ACC) (Accepted)*. IEEE, 2024.

[28] Zhe Shen and Takeshi Tsuchiya. Singular zone in quadrotor yaw–position feedback linearization. *Drones*, 6(4):84, 2022.

[29] Koushil Sreenath, Taeyoung Lee, and Vijay Kumar. Geometric control and differential flatness of a quadrotor UAV with a cable-suspended load. In *52nd IEEE conference on decision and control*, pages 2269–2274. IEEE, 2013.

[30] Michiel J Van Nieuwstadt and Richard M Murray. Real-time trajectory generation for differentially flat systems. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 8(11):995–1020, 1998.

# APPENDICES

# Appendix A

## A.1 Parameterization of $\mathsf{SO}(3)$

Special Orthogonal Group, denoted as $\mathsf{SO}(3)$, contains all rotations around the origin in 3-dimensional Euclidean space $\mathbb{R}^3$. It can be used to represent attitude of a rigid body. In this section, we introduce various parameterization methods to represent attitude in $\mathsf{SO}(3)$. The relationships among these parameterization methods are also given as tools for analyzing the quadrotor dynamics later.

### A.1.1 Rotation Matrix

Since rotation in 3-dimensional space is a linear transformation, it can be parameterized as a 3-by-3 matrix. For instance, suppose the body frame of a rigid body has three axises $\boldsymbol{x}_b$, $\boldsymbol{y}_b$, $\boldsymbol{z}_b \in \mathbb{R}^3$, then the rotation matrix is $[\boldsymbol{x}_b, \boldsymbol{y}_b, \boldsymbol{z}_b]$. Such so called "rotation matrix" is defined as $R \in \{R \in \mathbb{R}^{3\times 3} | R^T R = I, \det(R) = 1\}$.

### A.1.2 Euler Angles

The rotations in $\mathbb{R}^3$ have three degrees of freedom (DOF), so the minimum number of parameters to represent rotations is 3. One way of parameterization uses three angles—roll, yaw, and pitch, or $(\phi, \theta, \psi)$—as the parameters. They are often called Euler angles, named after Leonhard Euler, who first developed this parameterization method. $(\phi, \theta, \psi)$ are respectively the angles of rotation around $\boldsymbol{x}_b$, $\boldsymbol{y}_b$, and $\boldsymbol{z}_b$. However, there is a specific order in which the rotations occur. Euler angles use the "ZYX" representation, which means the rotation around $\boldsymbol{x}_b$ happens first, followed by $\boldsymbol{y}_b$, and lastly, $\boldsymbol{z}_b$. Different orders of rotation may result in different attitudes.

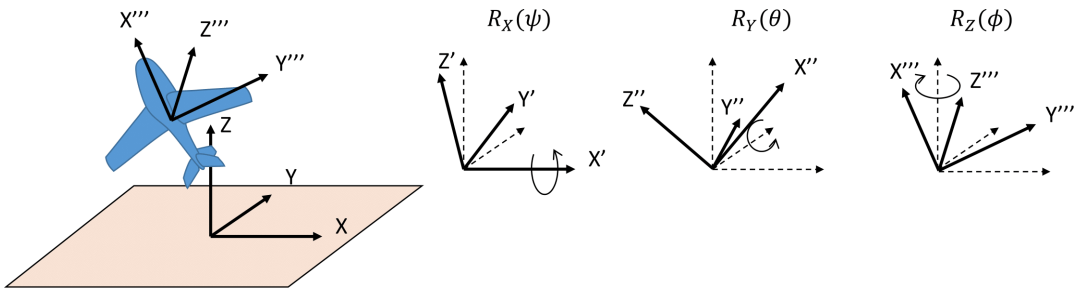


Figure A.1: Euler angles

### A.1.3 Angle-axis Representation

The angle-axis representation is used less frequently compared to the rotation matrix or Euler angles, but it helps in understanding the quadrotor error dynamics defined later. This representation uses an angle $\theta \in \mathbb{R}$ and a vector $\boldsymbol{a} \in \mathbb{R}^3$, where $|\boldsymbol{a}| = 1$. Although it appears that four numbers should account for four DOFs, one DOF is lost due to the requirement that $|\boldsymbol{a}| = 1$.

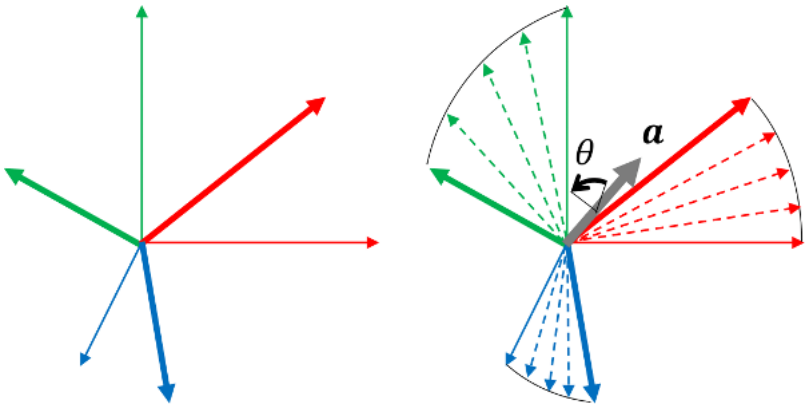


Figure A.2: Rotation matrix represented by a rotation of an angle about an axis.

### A.1.4 Invertible Transformation between Rotation Matrix and Euler Angles

In "ZYX" transformation, rotate angles in radians about x-axis, y-axis, z-axis are $\phi$, $\theta$, $\psi$, we have

$$\begin{aligned} R &= R_z(\psi)R_y(\theta)R_x(\phi) \\ &= \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \\ &= \begin{pmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{pmatrix} \end{aligned} \tag{A.1}$$

where $R_x(\phi)$, $R_y(\theta)$, and $R_z(\psi)$ respectively represents the rotation around x-axis, y-axis, and z-axis. The reason why the $-\sin\theta$ term switches to below the diagonal?

$$\begin{aligned} \theta &= -\sin^{-1}(R_{31}) \\ \phi &= \arctan\frac{R_{32}}{R_{33}} \\ \psi &= \arctan\frac{R_{21}/\cos\theta}{R_{11}/\cos\theta} \end{aligned} \tag{A.2}$$

### A.1.5 Conversion between Rotation Matrix and Angle-axis Representation

The conversion between a rotation matrix $R$ and an axis-angle representation $(\boldsymbol{a}, \theta)$ [22] is

$$R = \begin{pmatrix} \cos\theta + a_1^2(1-\cos\theta) & a_1a_2(1-\cos\theta) - a_3\sin\theta & a_1a_3(1-\cos\theta) + a_2\sin\theta \\ a_1a_2(1-\cos\theta) + a_3\sin\theta & \cos\theta + a_2^2(1-\cos\theta) & a_2a_3(1-\cos\theta) - a_1\sin\theta \\ a_1a_3(1-\cos\theta) - a_2\sin\theta & a_2a_3(1-\cos\theta) + a_1\sin\theta & \cos\theta + a_3^2(1-\cos\theta) \end{pmatrix} \tag{A.3}$$

And we can convert $R$ back to angle-axis as follows:

$$\begin{aligned} \text{tr}[R] &= 3\cos\theta + (a_1^2 + a_2^2 + a_3^2)(1 - \cos\theta) \\ &= 3\cos\theta + (1 - \cos\theta) \\ &= 2\cos\theta + 1 \end{aligned} \tag{A.4}$$

So $\theta$ in terms of $R$ is

$$\theta = \arccos(\frac{\text{tr}[R] - 1}{2}) \tag{A.5}$$

## A.2 Properties of SO(3)

In this section, we prove several lemmas that will be used later for analyzing the error system.

**Definition A.2.1.** (Frobenius Norm of a Matrix)

$$\|A\|_F = (\sum_{i,j=1}^{n} |A(i,j)|^2)^{\frac{1}{2}} = \sqrt{\text{tr}[A^T A]} = \sqrt{\text{tr}[AA^T]}$$

By Definition (A.2.1), we have

$$\begin{aligned} \|R - R_d\|_F &= (\text{tr}\big[(R - R_d)(R - R_d)^T\big])^{\frac{1}{2}} \\ &= (\text{tr}\big[RR^T - R_d R^T - RR_d^T + R_d R_d^T\big])^{\frac{1}{2}} \\ &= (\text{tr}[2I] - \text{tr}\big[2R_d R^T\big])^{\frac{1}{2}} \\ &= (2\text{tr}\big[I - R_d R^T\big])^{\frac{1}{2}} \\ &= (\text{tr}\big[(I - R_d R^T)(I - RR_d^T)\big])^{\frac{1}{2}} \\ &= \|I - R_d R^T\|_F \end{aligned} \tag{A.6}$$

**Definition A.2.2.** (Error function on SO(3))

$$\Psi(R, R_d) = \tfrac{1}{2} tr[I - R_d^T R] = \tfrac{1}{4}\|R - R_d\|_F^2 = \tfrac{1}{4}\|I - R_d R^T\|_F$$

**Lemma A.2.3.** $tr[A\hat{x}] = -(A - A^T)^\vee \cdot x$

*Proof.* Clearly, $\text{tr}[A\hat{x}] = \frac{1}{2}(\text{tr}[A\hat{x}] + \text{tr}[\hat{x}A]) = \frac{1}{2}(\text{tr}[A\hat{x}] + (-\text{tr}\big[A^T\hat{x}\big])) = \frac{1}{2}\text{tr}\big[(A - A^T)\hat{x}\big]$,

$(A - A^T)$ is skew-symmetric. Let $(A - A^T)^\vee = [a_1, a_2, a_3]^T$, then

$$\begin{aligned}
\mathrm{tr}\big[(A - A^T)\hat{x}\big] &= \mathrm{tr}\left[\begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}\right] \\
&= \mathrm{tr}\left[\begin{pmatrix} -a_3x_3 - a_2x_2 & .. & .. \\ .. & -a_3x_3 - a_1x_1 & .. \\ .. & .. & -a_2x_2 - a_1x_1 \end{pmatrix}\right] \qquad \text{(A.7)} \\
&= -2(a_1x_1 + a_2x_2 + a_3x_3) \\
&= -2(A - A^T)^\vee \cdot x
\end{aligned}$$

$\therefore \mathrm{tr}[A\hat{x}] = \frac{1}{2}\mathrm{tr}\big[(A - A^T)\hat{x}\big] = -(A - A^T)^\vee \cdot x$ ■

**Lemma A.2.4.** $\hat{x}A + A^T\hat{x} = ((tr[A]\, I - A)x)^\wedge$

*Proof.* Take

$$x = (x_1, x_2, x_3)^T$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Then,

$$
\begin{aligned}
&(\hat{x}A + A^T\hat{x})^\vee \\
=&(\begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \\
&\quad + \begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}\begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix})^\vee \\
=&(\begin{pmatrix} .. & x_1a_{31} + x_2a_{32} - x_3(a_{22} + a_{11}) & -x_1a_{21} + x_2(a_{11} + a_{33}) - x_3a_{23} \\ .. & .. & -x_1(a_{22} + a_{33}) + x_2a_{12} + x_3a_{13} \\ .. & .. & .. \end{pmatrix})^\vee \qquad \text{(A.8)} \\
=&\begin{pmatrix} x_1(a_{22} + a_{33}) - x_2a_{12} - x_3a_{13} \\ -x_1a_{21} + x_2(a_{11} + a_{33}) - x_3a_{23} \\ -x_1a_{31} - x_2a_{32} + x_3(a_{22} + a_{11}) \end{pmatrix} \\
=&\begin{pmatrix} a_{22} + a_{33} & -a_{12} & -a_{13} \\ -a_{21} & a_{11} + a_{33} & -a_{23} \\ -a_{31} & -a_{32} & a_{11} + a_{22} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \\
=&(\mathrm{tr}[A]\, I - A)x
\end{aligned}
$$

■

**Lemma A.2.5.** $\widehat{R\omega} = R\hat{\omega}R^T$

*Proof.*

$$
\begin{aligned}
&\widehat{R\omega} = R\hat{\omega}R^T \\
\Longleftrightarrow &\widehat{R\omega} = R^{-T}\hat{\omega}R^{-1} \\
\Longleftrightarrow &R^T\widehat{R\omega}R = \hat{\omega} \\
\Longleftrightarrow &y^TR^T\widehat{R\omega}Rx = y^T\hat{\omega}x \qquad \forall x, y \in R^3 \\
\Longleftrightarrow &y^TR^T(R\omega \times Rx) = y^T(\omega \times x) \qquad \forall x, y \in R^3 \\
\Longleftrightarrow &(Ry) \cdot (R\omega \times Rx) = y \cdot (\omega \times x) \qquad \forall x, y \in R^3 \\
\Longleftrightarrow &\det(Ry, R\omega, Rx) = \det(y, \omega, x) \qquad \forall x, y \in R^3 \\
\Longleftrightarrow &\det(R)det(y, \omega, x) = \det(y, \omega, x) \qquad \forall x, y \in R^3 \\
\Longleftrightarrow &\det(R) = 1
\end{aligned}
\qquad \text{(A.9)}
$$

Since $\det(R) = 1$ is always true for all $x, y \in R^3$ and $R \in \mathsf{SO(3)}$, Lemma(A.2.5) holds. ■

Consider the difference of tangent vectors $\dot{R} \in T_R\mathsf{SO(3)}$ and $\dot{R}_d \in T_{R_d}\mathsf{SO(3)}$ as follows.

$$\begin{aligned}
\dot{R} - \dot{R}_d(R_d^T R) &= R\hat{\omega} - R_d\hat{\omega}_d R_d^T R \\
&= R\hat{\omega}R^T R - R_d\hat{\omega}_d R_d^T R \\
&= (R\hat{\omega}R^T - R_d\hat{\omega}_d R_d^T)R \\
&= (\widehat{R\omega} - \widehat{R_d\omega_d})R \qquad Lemma(A.2.5) \\
&= (R\omega - R_d\omega_d)^\wedge R \\
&= (R\omega - RR^T R_d\omega_d)^\wedge R \\
&= (R(\omega - R^T R_d\omega_d))^\wedge R \\
&= R(\omega - R^T R_d\omega_d)^\wedge R^T R \qquad Lemma(A.2.5) \\
&= R(\omega - R^T R_d\omega_d)^\wedge
\end{aligned} \tag{A.10}$$

**Lemma A.2.6.** $\frac{d}{dt}(R_d^T R) = (R_d^T R)\hat{e}_\omega$

*Proof.*

$$\begin{aligned}
\frac{d}{dt}(R_d^T R) &= \dot{R}_d^T R + R_d^T \dot{R} \\
&= (R_d\hat{\omega}_d)^T R + R_d^T R\hat{\omega} \\
&= -\hat{\omega}_d R_d R + R_d^T R\hat{\omega} \\
&= -R_d^T RR^T R_d\hat{\omega}_d R_d R + R_d^T R\hat{\omega} \\
&= R_d^T R(\hat{\omega} - R^T R_d\hat{\omega}_d R_d R) \\
&= R_d^T R(\hat{\omega} - (R^T R_d\omega_d)^\wedge) \qquad Lemma(A.2.5) \\
&= R_d^T R(\omega - R^T R_d\omega_d)^\wedge \\
&= (R_d^T R)\hat{e}_\omega
\end{aligned} \tag{A.11}$$

■