ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

HIGH-SPEED TRAJECTORY TRACKING CONTROLLER DESIGN

M.Sc. THESIS

Omar SHADEED

Department of Aeronautics and Astronautics Engineering

Aeronautics and Astronautics Engineering Programme

FEBRUARY 2024

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

HIGH-SPEED TRAJECTORY TRACKING CONTROLLER DESIGN

M.Sc. THESIS

Omar SHADEED
(511211181)

Department of Aeronautics and Astronautics Engineering

Aeronautics and Astronautics Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Emre KOYUNCU

FEBRUARY 2024

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK HIZLI İZ TAKİP KONTROLÖRÜ TASARIMI

YÜKSEK LİSANS TEZİ

Omar SHADEED
(511211181)

Uçak ve Uzay Mühendisliği Anabilim Dalı

Uçak ve Uzay Mühendisliği Programı

Tez Danışmanı: Doç.Dr. Emre KOYUNCU

ŞUBAT 2024

Omar SHADEED, a M.Sc. student of ITU Graduate School of Science student ID 511211181 successfully defended the thesis entitled “HIGH-SPEED TRAJECTORY TRACKING CONTROLLER DESIGN”, which he prepared after ful-filling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Emre KOYUNCU** Istanbul Technical University	.....
Jury Members :	**Asst. Prof. Dr. Ramazan YENIÇERI** Istanbul Technical University	.....
	Dr. Mehmet HASANZADE	.....

Date of Submission : **16 JANUARY 2024**
Date of Defense : **19 FEBRUARY 2024**

To my family,

FOREWORD

I would like to express my sincerest gratitude to my thesis advisor, Prof. Emre Koyuncu, for his unwavering support and invaluable guidance throughout my MSc program at ITU Aerospace Research Center. Working with him has been a precious experience and a significant milestone in my academic journey.

I am also grateful to Prof. Gökhan İnalhan for broadening my academic horizons and helping me grow as a researcher.

I owe a great debt of gratitude to my labmates, Mehmet Hasanzase and Halise Turkmen, for their contributions and support in my thesis work. Without them, my MSc progress would have been more challenging and stressful. I hope to have the opportunity to collaborate with them again in the future.

I would also like to extend my thanks to the entire ITU Aerospace Research Center family, particularly Burak Yuksek, Arınç Tutku Altun, Yunus Biçer, and Mehmet Zeki Paşaoğlu, for their unwavering support and encouragement.

To my friends, I express my gratitude for their understanding and patience during this demanding phase of my academic journey. I apologize for not being able to spend as much time with them as I would have liked.

Finally, I am deeply grateful to my wife and partner Daria Vasileva for her support, encouragement, and invaluable assistance throughout the journey of writing this thesis. I am indebted to my parents, Abdelaziz Shadeed and Eman Maria, and my brothers, Yahya Shadeed and Abdelrahman Shadeed, for their limitless support, patience, and understanding. I attribute my success to the teachings and vision imparted by my mother and the unwavering support of my family.

FEBRUARY 2024 Omar SHADEED

TABLE OF CONTENTS

Page

ABBREVIATIONS

LQR	**:** Linear Quadratic Regulator
LQI	**:** Linear Quadratic Integrator
UAV	**:** Unmanned Aerial Vehicle
CPU	**:** Central Processing Unit
DDPG	**:** Deep Deterministic Policy Gradient
DRL	**:** Deep Reinforcement learning
FOV	**:** Field Of View
NED	**:** North-East-Down
UAV	**:** Unmanned Aerial Vehicle
PI	**:** Proportional-Integral controller
PID	**:** Proportional-Integral-Derivative controller
PPO	**:** Proximal Policy Optimization
PWM	**:** Pulse-Width Modulation
RMS	**:** Root Mean Square
TRPO	**:** Trust Region Policy Optimization

LIST OF TABLES

Page

LIST OF FIGURES

Page

HIGH-SPEED TRAJECTORY TRACKING CONTROLLER DESIGN

SUMMARY

Until recently, activities or tasks that necessitated the use of sophisticated guidance, navigation, and control systems at high performance levels would have mandated the deployment of unmanned aerial vehicles that were of tactical size. The underlying reason for this was the requirement for larger or heavier avionics with advanced computing capabilities or reliable communication buses that were connected to the ground systems to accommodate high-performance algorithms. However, advancements in technology have made it possible to incorporate these capabilities into smaller avionics, enabling the use of small unmanned aerial vehicles. Furthermore, new lightweight sensory systems have endowed these small unmanned systems with advanced situational awareness, making them capable of executing complex missions. Despite technological advancements, the control, navigation, and motion planning methods for unmanned aerial vehicles remain primarily "conservative" and tailored to specific use cases, resulting in limitations on their ability to carry out versatile operations.

Unmanned Aerial Vehicles (UAVs) have emerged as a highly versatile and efficient tool in a wide range of industries and applications, including but not limited to surveillance, transportation, and environmental monitoring. One of the critical functions of UAVs is high-speed trajectory tracking, which requires precise and reliable control algorithms for optimal performance. Control algorithms play a vital role in determining the dynamic response of a UAV, including its stability, agility, and accuracy in tracking the desired trajectory. The proper selection and implementation of control algorithms are essential for achieving safe, efficient, and effective high-speed trajectory tracking. Conversely, ineffective or inaccurate control algorithms can result in decreased performance, instability, reduced accuracy in trajectory tracking, and even accidents or crashes. Thus, the development of efficient and robust control algorithms is paramount for improving the performance of UAVs in high-speed trajectory tracking tasks, ensuring their reliability, safety, and overall success.

In this thesis, studies were carried out on the topic of trajectory tracking. The study proposes three different types of trajectory tracking controllers and provides simulation and real-time experimental results showing the root mean square error of the trajectory tracking performance. The first controller type is a differential flatness based controller where both LQR and LQI techniques are utilized. The second controller type is a deep reinforcement learning-based trajectory tracking controller, enabling to minimize the positional and velocity track error for aerial vehicles based on a Proximal Policy Optimization (PPO) algorithm where the controller is trained through randomly generated feasible trajectories. The third and last controller type is a cross entropy optimization based controller, enabling to minimize the positional and velocity track error for aerial vehicles. For all the three controllers type, PI was utilized for

the attitude controller and PID for the attitude rate controller as the aerial vehicle's low-level controllers.

YÜKSEK HIZLI İZ TAKİP KONTROLÖRÜ TASARIMI

ÖZET

Yakın zamana kadar, yüksek performans düzeylerinde sofistike rehberlik, navigasyon ve kontrol sistemlerinin kullanılmasını gerektiren aktiviteler veya görevler, taktik boyutta olan insansız hava araçlarının kullanımını gerektirirdi. Bunun temel nedeni, yüksek performanslı algoritmaları barındırmak için daha büyük veya daha ağır aviyoniklere veya ileri hesaplama yeteneklerine sahip güvenilir iletişim hatlarına ihtiyaç duyulmasıydı. Ancak, teknolojideki gelişmeler, bu yeteneklerin daha küçük aviyoniklere entegre edilmesini mümkün kıldı ve küçük insansız hava araçlarının kullanımına imkan sağladı. Ayrıca, yeni hafif algılama sistemleri, bu küçük insansız sistemlere gelişmiş durumsal farkındalık kazandırdı ve karmaşık görevleri yerine getirebilir hale getirdi. Teknolojik gelişmelere rağmen, insansız hava araçlarının kontrol, navigasyon ve hareket planlama yöntemleri, genellikle "muhafazakar" ve belirli kullanım durumlarına özgü olarak tasarlanmıştır ve çok yönlü operasyonlar yapma yeteneklerinde sınırlamalara neden olabilir.

İnsansız Hava Araçları (İHA'lar), gözetim, taşımacılık ve çevre izleme de dahil olmak üzere geniş bir endüstri ve uygulama yelpazesinde son derece çok yönlü ve verimli bir araç olarak ortaya çıkmıştır. İHA'ların kritik fonksiyonlarından biri, optimal performans için hassas ve güvenilir kontrol algoritmaları gerektiren yüksek hızda iz yolu takibidir. Kontrol algoritmaları, İHA'nın dinamik tepkisini, stabilitesini, çevikliğini ve istenen iz yolunu takip etme doğruluğunu belirlemede hayati bir rol oynar. Güvenli, verimli ve etkili bir yüksek hızlı iz yolu takibi için uygun kontrol algoritmalarının seçimi ve uygulanması önemlidir. Bunun tersi, etkisiz veya doğru olmayan kontrol algoritmaları, performansta azalmaya, istikrarsızlığa, iz yolunda azalmış doğruluğa ve hatta kazalara veya çarpışmalara neden olabilir. Bu nedenle, verimli ve sağlam kontrol algoritmalarının geliştirilmesi, İHA'ların yüksek hızda iz yolu takibi görevlerinde performansını artırmak, güvenilirliğini, güvenliğini ve genel başarısını sağlamak için çok önemlidir.

Bu tezde, iz takibi konusunda çalışmalar yapılmıştır. Çalışma, üç farklı türde iz takibi denetleyicisi önermektedir ve iz takibi performansının kök ortalama kare hatası gösteren simülasyon ve gerçek zamanlı deneysel sonuçlar sunmaktadır. İlk denetleyici türü, LQR ve LQI teknikleri her ikisi de kullanılan bir diferansiyel düzlemsellik tabanlı denetleyicidir. İkinci denetleyici türü, bir Proksimal Politika Optimizasyonu (PPO) algoritmasına dayanan derin pekiştirmeli öğrenme tabanlı bir iz takibi denetleyicisidir ve denetleyici rastgele oluşturulan uygun izler üzerinde eğitilir. Üçüncü ve son denetleyici türü, pozisyonel ve hızlı iz hata hatasını azaltmaya olanak sağlayan bir çapraz entropi optimizasyon tabanlı denetleyicidir. Tüm üç denetleyici türü için, düşük seviye denetleyiciler olarak hava aracının durum denetleyicisi için PI ve durum hızı denetleyicisi için PID kullanılmıştır.

1. INTRODUCTION

The advancements in battery and engine technology have paved the way for unmanned aerial vehicles (UAVs) to become an integral part of modern society over the last few decades. These devices are commonly employed in a wide range of civil and entertainment contexts, including observation, cargo transportation, cinematic shooting, gaming, and drone racing competitions. Additionally, they have found a place in various military operations, such as information gathering, observation, and ammunition deployment. With the continued evolution of UAV technology, novel application areas have emerged, and more sophisticated methods of operation have become feasible.

The prevalence of UAVs in diverse sectors has highlighted their potential to provide safer, more cost-effective, and time-efficient alternatives to conventional methods, such as manned aircraft or ground-based transport. Their adaptability has made them increasingly popular in civil and entertainment applications, while their use in military fields has become indispensable. The integration of UAV technology in different contexts has enabled the devices to find new application areas, while also providing more advanced methods of operation.

The global consumer drone market, which was valued at USD 4,120.8 million in 2022, is expected to grow at a CAGR of 13.3% from 2023 to 2030 as shown in Figure 1.1. In addition to traditional hobbyists and flight enthusiasts, consumer drones are becoming increasingly popular among individuals seeking to enhance their leisure activities through aerial photography or exploration, as well as those looking to create a gig or fly drones for fun. The global consumer drone sales have witnessed a significant increase as customers are exploring the benefits of drones for personal interests. Technological innovations in consumer drones are driving market growth, and this trend is expected to continue over the forecast period.

The adoption of drone technology by several organizations across various industries has been spurred by the rapid growth of the drone service market. Companies investing

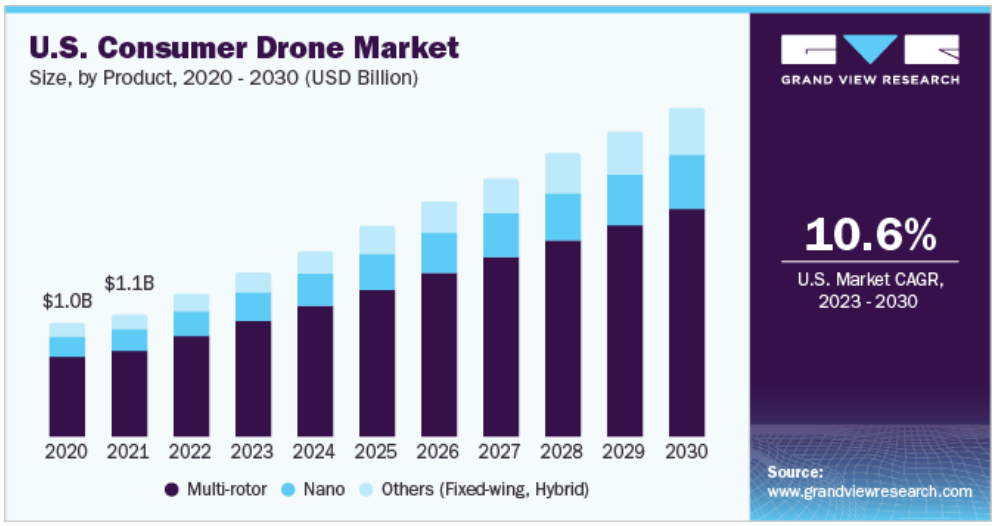


Figure 1.1 : Consumer Drone Market [1]

in drones anticipate cost reductions, revenue growth, and improved decision-making through the analysis of drone-collected data. Aerial photography, cinematography, and special effects are just a few of the applications of drones in the media and entertainment industries. As drones continue to collaborate with other technologies, more changes are likely to emerge in the future.

Notably, there have been recent developments in the consumer drone market, such as Skyfish's technical collaboration with Sony Electronics to deliver drones with high-quality data gathering and photogrammetry capabilities, and DJI's launch of the Mini 3 Pro drone. The Mini 3 Pro drone, which weighs 249 grams, offers additional flight control capabilities and high-quality picture and video recording functionality. The product is designed specifically for consumers and enthusiasts, with a particular emphasis on portability and affordability.

Recent advancements in sensory systems and AI-based perception algorithms have eliminated the technology barriers in perception and computational power, making unmanned aerial vehicles (UAVs) increasingly capable of performing complex operations. This enables small-sized UAVs to perform missions that require high-level situational awareness. Additionally, developments in hardware such as graphic processing units (GPUs) with neural network architectures have enhanced the computational abilities of unmanned systems, enabling them to run demanding algorithms. However, small UAVs are still unable to perform highly agile operations due to trajectory tracking algorithms being built and optimized for specific cases.

As a result, the challenge of developing precise and generalized tracking algorithms remains.

1.1 Aims and Objectives

As discussed previously, accurate trajectory tracking algorithms are of utmost importance for small UAVs, as they enable these unmanned aerial vehicles to precisely navigate through complicated and dynamic environments. These algorithms can help the UAVs to maintain a stable flight trajectory, avoid obstacles, and accomplish mission objectives with greater efficiency and safety. Given the limited payload capacity and energy constraints of small UAVs, precise trajectory tracking can also help to optimize their flight paths and conserve energy. In addition, the ability to accurately track trajectories is essential for applications such as aerial surveying, search and rescue, and environmental monitoring, which require high-level situational awareness and precise control over the UAV's movements. As such, the development of effective trajectory tracking algorithms is crucial to unlocking the full potential of small UAVs in a range of fields. Therefore, studies were carried out on the topic of trajectory tracking in this thesis.

The study proposes three different types of trajectory tracking controllers and provides simulation and real-time experimental results showing the root mean square error of the trajectory tracking performance. The first controller type is a differential flatness based controller where both LQR and LQI techniques are utilized. This study is published in two conference papers at AIAA Scitech Forum [4] [5].

The second controller type is a deep reinforcement learning-based trajectory tracking controller, enabling to minimize the positional and velocity track error for aerial vehicles based on a Proximal Policy Optimization (PPO) algorithm where the controller is trained through randomly generated feasible trajectories. This study is published at the AIAA Scitech Forum [6].

The third and last controller type is a cross entropy optimization based controller, enabling to minimize the positional and velocity track error for aerial vehicles. This study is published at the European Control Conference [7]. For all the three controllers

type, PI was utilized for the attitude controller and PID for the attitude rate controller as the aerial vehicle's low-level controllers.

1.2 Literature Review

In cases where aggressive collision avoidance is required, trajectory tracking methods are essential to ensure the desired optimality of the results. The vehicle needs to follow the desired trajectory, and even though path planning methods can guarantee local optimality, the trajectory tracking method must ensure expected performance during flight to ensure overall optimality. To address this issue, most trajectory tracking approaches in the literature are based on Lyapunov/backstepping techniques [8–15] and geometric control [16–18].

In the past, classical or quadratic controllers were deemed sufficient for vehicles to perform satisfactorily, as they did not encounter significant drag forces. Consequently, the suggested solutions for vehicle control did not factor in the effects of drag force. However, with the growing demand for more aggressive and complex maneuvers from quadcopters, recent studies have emerged in the literature to address this issue.

In [19], an architecture is presented that employs an LQR differential flatness based controller for the quadcopter platform. The quadcopters' ability to rapidly rotate around the vertical axis allows for arbitrary changes in the yaw angle, but the tilt angles in the pitch and roll axes are not restricted, which may be small.

One particular study of [2] investigates the influence of linear rotor drag on quadrotor flight and proposes a solution based on the differentially flat formulation of these drag effects, as expressed in terms of the vehicle's position and heading. This formulation allows for the computation of feedforward control terms from desired trajectory references. A nonlinear feedback control system is then designed to incorporate these control terms, enabling precise and aggressive quadrotor flight, as illustrated in Figure 1.2. The findings indicate that this approach leads to a significant improvement in root mean squared tracking error by up to 50

The solution presented in [20] investigates an approach for ensuring a zero thrust actuation that can be effectively applied and still converges to the intended trajectory. The solution aims to ensure that the actuation does not escalate uncontrollably, while

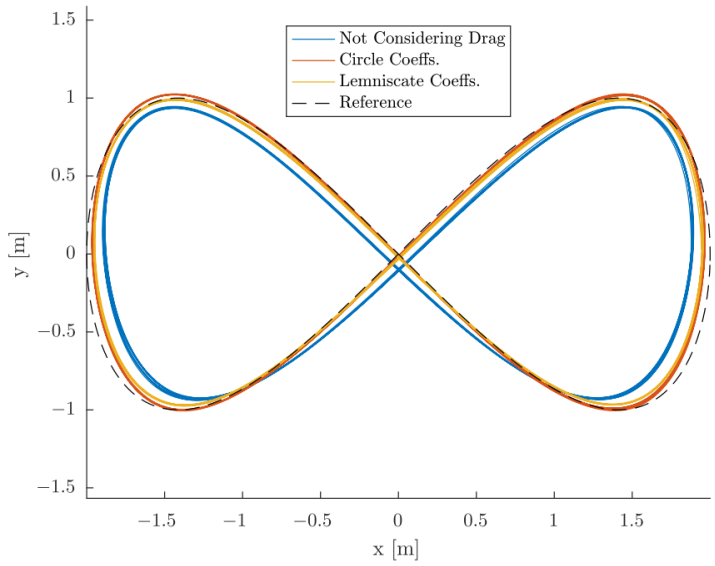


Figure 1.2 : Trajectory tracking performance considering rotor drag effects on the vehicle [2]

also ensuring global asymptotic convergence to a point where the trajectory tracking error is reduced to zero. In order to achieve these objectives, [20] proposes the use of a nonlinear state feedback controller.

One of the areas of interest in trajectory tracking control for quadcopters is the case of a single motor failure. In the study by [21], a proposed method enables the quadcopter to continue tracking the desired trajectory even when one rotor fails. To address this challenge, a dynamic feedback control law is presented to ensure that the desired velocity references are maintained along the trajectory.

The article [22] presents a proposal for a continuous-time predictive control architecture to address limited output trajectory tracking issues. The proposed approach involves designing a continuous-time sampled-data nonlinear model predictive control (NMPC) framework that accounts for the state and input constraints. The framework ensures convergence by employing end penalties and conditions based on terminal regions. In addition, the authors explore the geometric aspects of trajectory tracking problems by utilizing transverse normal form analysis to compute end penalties. The proposed architecture has the potential to achieve better tracking performance compared to existing approaches while ensuring constraints are satisfied.

Recent advances in the field of deep reinforcement learning have enabled agents to tackle complex, high-dimensional problems. However, the application of reinforcement learning techniques in vehicles is still limited due to the potential for vehicle damage, particularly in unmanned aerial vehicles (UAVs) where crashes can be costly. As learning an agent requires multiple trials, training is typically conducted in a simulated environment and then tested by implementing the agent. Although the theoretical evidence supporting reinforcement learning techniques is not yet clear, Monte Carlo simulations have shown that the created agent meets the requirements for trajectory tracking. Nonetheless, all factors that may be encountered during real flights should also be replicated in the Monte Carlo simulation. While back-stepping and geometric control approaches dominate the literature on trajectory tracking solutions, an increasing number of solutions incorporating reinforcement learning are emerging over time.

In [23], a path following problem in a UAV is tackled using the Deep Deterministic Policy Gradient (DDPG) reinforcement learning algorithm. The focus of the study is on the determination of the yaw angle command by the agent, while the desired altitude and forward velocity are predetermined by the user. The generated delta yaw angle is added to the current yaw command and utilized in the attitude controller. However, it is noted that merely producing yaw angle commands would not be adequate to achieve trajectory tracking in aggressive scenarios.

In their work presented in [3], a novel approach for end-to-end training of deep reinforcement learning (DRL) agent is proposed. The suggested method uses raw sensor data to generate motor thrusts directly. The conventional UAV control approaches typically involve sensor fusion, high-level, and low-level controllers as separate modules. In contrast, the proposed DRL agent models the entire process and enables dynamic stabilization even in complex situations such as upside-down throws. To train the agent, deterministic on-policy method is employed using zero-bias and zero-variance samples. Additionally, the authors utilized the vehicle model employed in real flight tests to enhance precision during training. A performance comparison with DDPG and Trust Region Policy Optimization (TRPO) algorithms is presented in Figure 1.3, demonstrating the superiority of the proposed approach.

Figure 1.3 : Control performance of the RL agent [3]

One of the research works concentrates on developing a control system for self-governing inverted flight of a helicopter. The suggested methodology in [24] involves leveraging actual flight data for comprehending the helicopter's dynamics, and employing reinforcement learning algorithms to acquire the controller.

The conventional proportional-integral-derivative (PID) controller for attitude control is effective under normal conditions, but it may not be sufficient in harsh and unpredictable environments. To address this issue, a reinforcement learning (RL) based solution for the inner controller has been proposed in [25]. Trust Region Policy Optimization (TRPO), Proximal Policy Optimization (PPO), and Deep Deterministic Gradient Policy (DDGP) were utilized as the RL algorithms. Among these algorithms, PPO showed the best performance and outperformed the PID controller.

1.3 Thesis Outline

Figure 1.4 shows the general architecture of any autonomous rotary wing vehicle. Each of the systems represented in this architecture is explained in this thesis. The desired trajectory generation is explained in section 2.3. The attitude controller is explained in section 2.2. The motors characteristics are explained in section 2.1.1. The dynamics of the rotary wing vehicle used in this research is explained in section 2.1. The main focus of this thesis is to develop different position, velocity, acceleration controllers aiming

to maximize the tracking performance. These controllers are explained in chapters 3, 4, and 5.

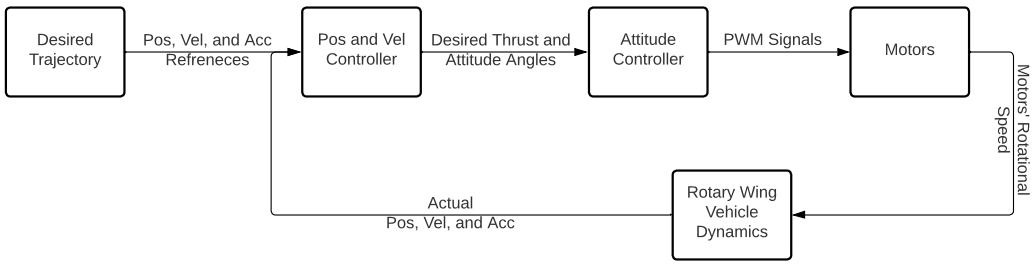


Figure 1.4 : Autonomous rotary wing vehicle architecture

2. UNMANNED AERIAL VEHICLE DYNAMICS AND MODELING

2.1 Aerial Vehicle Model

The air vehicle used in this work is the Crazyflie 2.1 [26]. In this section, the dynamical model of that air vehicle will be given and it is based on a NWU inertial frame system as shown in Figure 2.1.

The 12 states that will be used to derive the equation of motion of the quadrotor are defined as follows:

$$X = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & p & q & r \end{bmatrix}^T \tag{2.1}$$

Where x, y and z are the position of the Crazyflie drone in the inertial coordinate frame. The inertial velocities are $\dot{x}, \dot{y}$ and $\dot{z}$. ϕ, θ and ψ are the Euler angles that describe the orientation and p, q and r are the body frame angular rates.

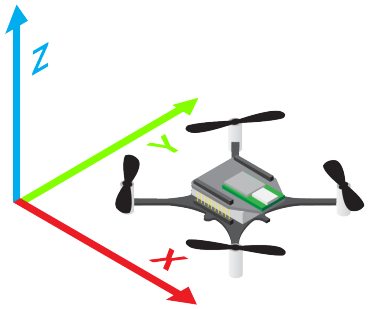


Figure 2.1 : Quadrotor UAV coordinate system

Before giving the non-linear equations governing the motion of the air-vehicle, it is important to define the transformation matrix used to transform from the inertial frame to the body frame and it is given as follows:

$$R_o^b = \begin{bmatrix} cos\theta cos\psi & cos\theta sin\psi & -sin\theta \\ sin\phi sin\theta cos\psi - cos\phi sin\psi & sin\phi sin\theta sin\psi + cos\phi cos\psi & sin\phi cos\theta \\ cos\phi sin\theta cos\psi + sin\phi sin\psi & cos\phi sin\theta sin\psi - sin\phi cos\psi & cos\phi cos\theta \end{bmatrix} \tag{2.2}$$

The acceleration of the air-vehicle is then defined in the body-fixed frame as follows [27]:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ F_z/m \end{bmatrix} - R_o^b \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.3}$$

Where F is the total contact forces acting on the aerial vehicle and is defined in the body-fixed frame as:

$$F^b = \begin{bmatrix} 0 \\ 0 \\ C_T(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \end{bmatrix} - F_{drag} \tag{2.4}$$

And

$$F_{drag} = \frac{1}{2}\rho C_D A_{eff} \begin{bmatrix} u^2 \\ v^2 \\ w^2 \end{bmatrix} \tag{2.5}$$

Where ω_i is the rotation speed of the i-th motor, and C_T is the thrust coefficient. u, v, and w are the velocities in the body-fixed frame in the X, Y, and Z axes respectively. ρ, C_D, and A_{eff} are the parameters defined in Table 2.2.

The position of the air-vehicle in the inertial frame is expressed by integrating the output of the following equation [27]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_b^o \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.6}$$

where $R_b^o = (R_o^b)^{-1} = (R_o^b)^T$.

From the theorem of angular momentum, the angular velocities are expressed as [27]:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = (J)^{-1} \left(\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times J \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \tag{2.7}$$

where J is the inertia matrix given as:

$$J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.8}$$

and M_x, M_y, M_z are the moments acting on the body of the air-vehicle around the x,y,z axes respectively. The total moment acting on the aerial vehicle is defined as:

$$M^b = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} dC_T/\sqrt{2}(-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2) \\ dC_T/\sqrt{2}(-\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2) \\ C_D(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{bmatrix} \tag{2.9}$$

Table 2.1 : Physical parameters of the Carzyflie aerial vehicle.

Parameter	Description	value
m	mass of the drone	0.034 [Kg]
d	Arm length	39.7 x 10^{-3} [m]
I_{xx}	moment of inertial around x axis	1.395 x 10^{-5} [Kg x m^2]
I_{yy}	moment of inertial around y axis	1.436 x 10^{-5} [Kg x m^2]
I_{zz}	moment of inertial around z axis	2.173 x 10^{-5} [Kg x m^2]
C_T	Thrust coefficient of the propeller	3.1582 x 10^{-10} [N/rpm^2]
C_D	Aerodynamic drag coefficient of the propeller	7.9379 x 10^{-12} [Nm/rpm^2]

Where d is the arm length, C_T is the thrust coefficient, ω_i is the rotation speed of the i-th motor, and C_D is the aerodynamic drag coefficient due to the movement of the propeller.

Finally, the Euler angles are given as [27]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi/cos\theta & cos\phi/cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \text{ for } \theta \neq \frac{\pi}{2} \tag{2.10}$$

The physical parameters of the Crazyflie that were used in the simulation are summarized in Table 2.1.

2.1.1 Motors characterization

Upon examining the Crazyflie 2.1 firmware, it became clear that the forces and moments acting on the aerial vehicle's body are not the system's actual input. Instead, the voltage sent to each DC motor is controlled by a PWM signal, which is a 16-bit number ranging from 0 to 65535. This means that the system's actual input can be considered as the PWM signal rather than the voltage sent to the motors. In [28], experimental data was gathered from the motors to establish the relationship between the PWM signal sent to the motors and the generated RPMs. The experiments demonstrated that the angular speed of the motors has a linear correlation with the PWM input of the system, as given by the equation:

$$RPM = 0.2685 \times PWM + 4070.3 \tag{2.11}$$

A DC motor's characterization typically results in a first-order transfer function that specifies a certain response time. Experimental results have shown that the motor's

response can be characterized by the following first-order transfer function:

$$\frac{PWM_{actual}}{PWM_{desired}} = \frac{15.4666}{s+15.4666} \tag{2.12}$$

2.2 Attitude Controller

The attitude controller utilized in all the studies conducted in chapters 3 through 5 is presented in this section. Figure 2.2 displays the architecture of the attitude controller that is implemented on the Crazyflie 2.1. For low-level control, a two cascaded proportional-integral (PI) control architecture is employed. The attitude controller operates at a frequency of $250Hz$, while the attitude rate controller operates at a frequency of $500Hz$. High frequency controllers are unnecessary for common missions such as way point tracking, but they are essential for aggressive trajectory tracking. Despite the fact that these frequencies are already established, they are efficient enough to achieve the desired tracking performance.

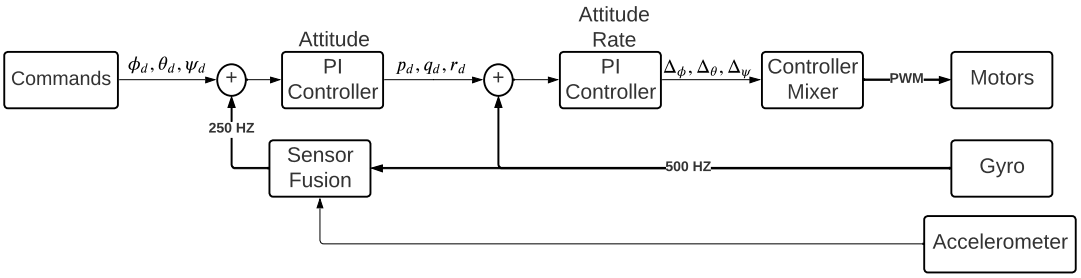


Figure 2.2 : Attitude controller architecture

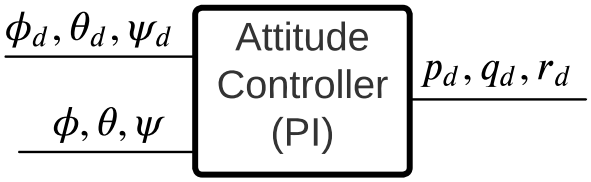


Figure 2.3 : Attitude controller inputs and outputs

The attitude controller depicted in Figure 2.3 employs PI controllers for roll and pitch control. The desired attitude angles values are computed using the position and velocity controllers as will be explained in the following chapters. The angle states are determined using onboard gyro and accelerometer sensors as feedback, and the orientation outputs from the VICON motion capture system are not utilized in the

sensor fusion. To summarize, the estimated roll ϕ and pitch θ values are compared to the desired roll ϕ_d and pitch θ_d values, and proportional-integral (PI) controllers are utilized to determine the desired attitude rate values p_d and q_d, as shown in Equations 2.13 and 2.14.

$$p_d(t) = K_{P,\phi}(\phi_d(t) - \phi(t)) + K_{I,\phi} \int_{\tau=0}^{\tau=t} (\phi_d(\tau) - \phi(\tau))d\tau \tag{2.13}$$

$$q_d(t) = K_{P,\theta}(\theta(t) - \theta(t)) + K_{I,\theta} \int_{\tau=0}^{\tau=t} (\theta(\tau) - \theta(\tau))d\tau \tag{2.14}$$

For both roll and pitch attitude controllers, The controller coefficients are assigned as $K_{P,\phi} = K_{P,\theta} = 3.5$, $K_{I,\phi} = K_{I,\theta} = 2.0$ [26].

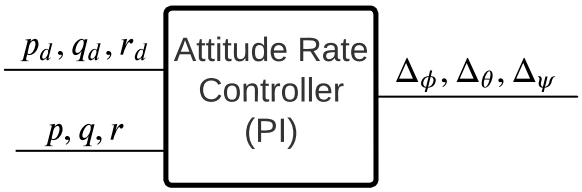


Figure 2.4 : Attitude rate controller inputs and outputs

Once the desired attitude rate commands are obtained, the attitude rate controller is utilized to determine the necessary angular momentum. The inputs and outputs of the attitude rate controller are depicted in Figure 2.4. The inputs to the controller include the desired roll rate p_d, the desired pitch rate q_d, the roll rate state p, and the pitch rate state q. The attitude rate controller utilizes a proportional controller, as shown in Equations 2.15 and 2.16, for roll and pitch control, and a PI controller, as shown in Equation 2.17, for yaw control.

$$\Delta_\phi(t) = K_{P,p}(p_d(t) - p(t)) \tag{2.15}$$

$$\Delta_\theta(t) = K_{P,q}(q_d(t) - q(t)) \tag{2.16}$$

$$\Delta_\psi(t) = K_{P,\psi}(r(t) - r(t)) + K_{I,r} \int_{\tau=0}^{\tau=t} (r(\tau) - r(\tau))d\tau \tag{2.17}$$

For attitude rate controllers, The controller coefficients are assigned as $K_{P,p} = K_{P,q} = K_{P,r} = 70$, $K_{I,r} = 16.7$ [26].

Once the desired input variation of the motors has been determined, a **control mixer** is employed. These values correspond to the desired torque values in the direction of movement. To describe the movement in relation to the motors, the values of $\Delta_\phi(t)$, $\Delta_\theta(t)$, and $\Delta_\psi(t)$ must be allocated to the motor outputs using the control mixer. The Crazyflie has been constructed with an "X" frame, so the control mixer is expressed using Equation 2.18 [26].

$$\begin{aligned} PWM_{m1} &= PWM_{base} - \Delta_\phi(t)/2 - \Delta_\theta(t)/2 - \Delta_\psi(t) \\ PWM_{m2} &= PWM_{base} + \Delta_\phi(t)/2 - \Delta_\theta(t)/2 + \Delta_\psi(t) \\ PWM_{m3} &= PWM_{base} + \Delta_\phi(t)/2 + \Delta_\theta(t)/2 - \Delta_\psi(t) \\ PWM_{m4} &= PWM_{base} - \Delta_\phi(t)/2 + \Delta_\theta(t)/2 + \Delta_\psi(t) \end{aligned} \tag{2.18}$$

Where PWM_{base} is the base PWM signal value needed for hovering.

2.3 Trajectory Generation

2.3.1 Trajectory representation

All the trajectories used throughout this thesis for simulation and experimental tests, are B-Spline based trajectories that provide the desired position, velocity, and acceleration on every time step. The B-Spline representation is used to define a flight trajectory, which enables parameterizing of a flat output through the generation of several joined polynomials and ensures the trajectory to be continuously differentiable. In order to meet with the trajectory requirements, We have chosen to represent flat output trajectories with $p = 5$ degree B-splines to ensure at least C^4 continuity. Generally, a pth-degree B-Spline curve is defined as follows:

$$P(t) = \sum_{i=0}^{n} P_i B_{i,p}(t) \qquad a \leq t \leq b \tag{2.19}$$

Where $P(t)$ denotes the curve at t which also could be represented as $P(t) = [P_x(t), P_y(t), P_z(t)]$ and the P_i are the control points. The $B_{i,k}(t)$ are basis functions that can be computed using the De Boor-Cox recursive formula [29–31].

$$B_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.20}$$

$$B_{i,p}(t) = \frac{u - u_i}{u_{i+p} - u_i} B_{i,p-1}(t) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} B_{i+1,p-1}(t) \tag{2.21}$$

These basis functions are defined as the function of the knot vectors:

$$\tau = [u_0, ..., u_m] \tag{2.22}$$

We used uniform knot vector which could also be presented as:

$$\tau = [a, ..., a, t_{p+1}, ..., t_{m-p-1}, b, ..., b] \tag{2.23}$$

The length of the knot vector is $m+1$ where $m = n + p + 1$ [31]. We assume that $a = 0$ and $b = 1$ for the unity knot vector, and the first and last knots have multiplicity $p+1$. Used knot vector is uniform if all interior knots are equally distanced such that $d = t_{i+1} - t_i$ for all $p \leq i \leq m - p - 1$. The derivatives of $P(t)$ can be used to define the velocity and acceleration of the trajectory, and can be expressed as follows:

$$P^{(k)}(t) = \sum_{i=0}^{n} P_i B_{i,p}^{(k)}(t) \qquad a \leq t \leq b \tag{2.24}$$

where k represents the order of derivatives.

2.3.2 Aerial vehicle dynamical constraints:

Before giving a trajectory to any type of controller to track, the dynamical feasibility needs to be verified. Therefore, all provided B-spline trajectories were optimized to ensure that they are not exceeding the dynamical limits of the aerial vehicle. The vehicle used for this work is the Crazyflie 2.1 [26]. Based on Newton's second law of motion, we get the following classical mechanics equation:

$$\mathbf{F}_{thrust} - \mathbf{F}_{gravity} - \mathbf{F}_{drag} = m^b \dot{\mathbf{V}}_{CG/o} \tag{2.25}$$

In the vertical climbing case, Eq. 2.25 can be written as:

$$T - mg - \frac{\rho}{2} C_D A_{eff} v^2 = m\dot{v} \tag{2.26}$$

And in the forward flight case, Eq. 2.25 can be written as:

$$\sqrt{1 - (\frac{mg}{T})^2}\, T - \frac{\rho}{2} C_D A_{eff} v^2 = m\dot{v} \tag{2.27}$$

The aerial vehicle will reach a limit speed without further acceleration due to air drag. We can then calculate the maximum possible rate of climb and the maximum forward flight speed from the following two equations:

$$\frac{T_{max}}{m} - g - \frac{\rho}{2m} C_D A_{eff} v_{max}^2 = 0 \tag{2.28a}$$

$$\sqrt{1 - (\frac{mg}{T_{max}})^2}\, \frac{T_{max}}{m} - \frac{\rho}{2m} C_D A_{eff} v_{max}^2 = 0 \tag{2.28b}$$

Table 2.2 shows the parameters used in solving Eq. 2.28 to find the maximum rate of climb and the maximum forward flight speed. By solving Eq. 2.28a, the maximum climb rate is found as $v_{climb,max} = 2.96\ m/s$. And by solving Eq. 2.28b, the maximum forward velocity is found as $v_{forward,max} = 7.36\ m/s$. Figure 2.5 shows the maximum forward acceleration as a function of the Crazyflie's velocity.

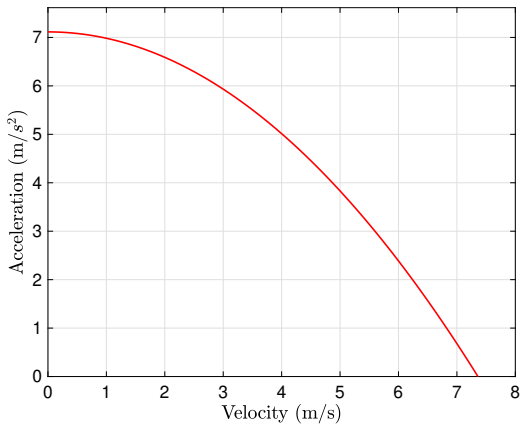


Figure 2.5 : Forward acceleration limits

Table 2.2 : Crazyflie aerial vehicle performance calculation parameters.

parameter	Description	Value
m	Mass of the drone	0.034 [Kg]
T_{max}	Maximum available thrust	0.41202 [N] [26]
g	Gravitational acceleration	9.81 [m/s^2]
ρ	Density of air	1.225 [kg/m^3]
C_D	Drag coefficient	0.9 [32]
A_{eff}	Effective area of the Crazyflie	8.1×10^{-3} [m^2] [26]

3. DIFFERENTIAL FLATNESS-BASED TRAJECTORY TRACKING CONTROLLER

3.1 LQR Differential Flatness Based Controller

The attitude controller explained in section 2.2 accepts the inputs as the total desired thrust T^d, roll angle ϕ^d, pitch angle θ^d, and body yaw rate r^d. The vector of inputs is defined as:

$$v = \begin{bmatrix} T_d \\ \phi_d \\ \theta_d \\ r_d \end{bmatrix} \tag{3.1}$$

The attitude controller achieves the desired input commands and therefore the state vector can be reduced to be

$$X = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \psi \end{bmatrix}^T \tag{3.2}$$

Assuming that the onboard attitude controller will achieve the commanded inputs with no error, the study in [19] suggests a change of variables on the inputs that describes the system using linear equations. The following definition shows the new input vector, a non-linear map of the inputs v and state X:

$$u = \begin{bmatrix} u_{p\ (3x1)} \\ u_{\psi\ (1x1)} \end{bmatrix} \equiv \begin{bmatrix} \hat{f}_p(X,v) \\ \hat{f}_\psi(v,q) \end{bmatrix} \tag{3.3}$$

where

$$\hat{f}_p(X,v) = R_B^i(\psi,\theta,\phi) \begin{bmatrix} 0 \\ 0 \\ \frac{T}{m} \end{bmatrix} \tag{3.4}$$

and

$$\hat{f}_\psi(v,q) = q\frac{sin\ \phi}{cos\ \theta} + r\frac{cos\ \phi}{cos\ \theta} \tag{3.5}$$

The state-space model of the system dynamics is derived by using the change of variables defined above, as follows:

$$\dot{x} = Ax + Bu - bg \tag{3.6}$$

where

$$A = \begin{bmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times1} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & 0_{1\times3} & 0_{1\times1} \end{bmatrix}$$

$$B = \begin{bmatrix} 0_{3\times3} & 0_{3\times1} \\ I_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}$$

and

$$b = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}'$$

3.1.1 Inverse function for control inputs

The state-space model of the system provided by Eq. 3.6 shows the system dynamics in terms of the input u that is defined by Eq. 3.3 and in units of acceleration. However, the Carzyflie attitude controller accepts the input in the form of v defined by Eq. 3.1. Therefore, it is necessary to convert the input u to the input v. The inverse functions $\hat{f}_p^{-1}$ and $\hat{f}_\psi^{-1}$ are used to compute the equivalent v inputs that will achieve the desired accelerations based on the desired spline. To solve for the first 3 inputs of u namely T, ϕ, and θ commands, Eq. 3.4 can be rewritten as:

$$\begin{bmatrix} u_{p1} \\ u_{p2} \\ u_{p3} \end{bmatrix} m = R^T(\psi)R^T(\theta)R^T(\phi) \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}$$

And by taking the norm of both sides, the total thrust is given by

$$T = m\sqrt{u_{p1}^2 + u_{p2}^2 + u_{p3}^2} \tag{3.7}$$

And by solving for the commands ϕ and θ, Eq. 3.4 becomes:

$$R(\psi) \begin{bmatrix} u_{p1} \\ u_{p2} \\ u_{p3} \end{bmatrix} \frac{m}{T} = R^T(\theta)R^T(\phi) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.8}$$

And by solving for ϕ and θ, we get

$$\phi = sin^{-1}(-z_2) \tag{3.9}$$

and

$$\theta = tan^{-1}\left(\frac{z_1}{z_3}\right) \tag{3.10}$$

where

$$z = R(\psi)\begin{bmatrix} u_{p1} \\ u_{p2} \\ u_{p3} \end{bmatrix}\frac{m}{T}$$

Equations 3.7, 3.9 and 3.10 construct the inverse function $\hat{f}_p^{-1}$. The inverse function $\hat{f}_\psi^{-1}$ is equal to r which is solved from Eq. 3.5, as

$$\hat{f}_\psi^{-1} = r = \dot{\psi}\, cos\theta\, cos\phi \; - \; \dot{\theta}\, sin\phi \tag{3.11}$$

3.1.2 System architecture

As shown in Section 2.3.1, the generated B-splines are three times differentiable, therefore, the given spline is a flat output and can be written as follows:

$$p(t) = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \ddot{x} & \ddot{y} & \ddot{z} & \psi & \dot{\psi} \end{bmatrix}^T \tag{3.12}$$

The state and control inputs are defined by

$$X^r = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \psi \end{bmatrix}^T \tag{3.13}$$

and

$$u^r = \begin{bmatrix} \ddot{x} & \ddot{y} & \ddot{z}+g & \dot{\psi} \end{bmatrix}^T \tag{3.14}$$

As shown in Figure 3.1, the input u^r gives the desired inputs in terms of Eq. 3.6, which are converted to the input in the form of Eq. 3.1.

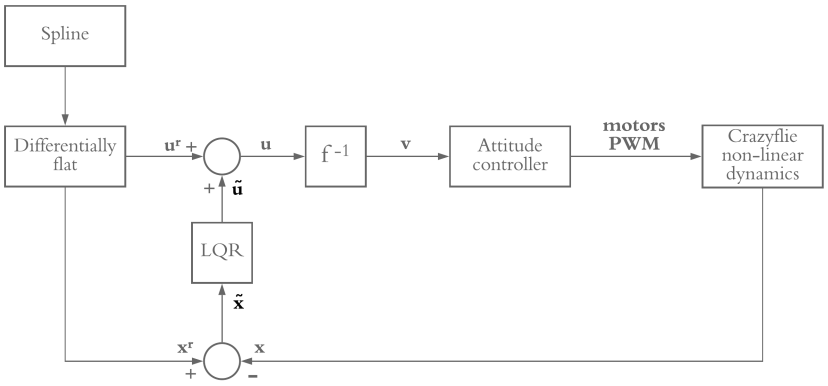


Figure 3.1 : Architecture of the LQR Differentially Flatness based Controller

The input v is then given as commands to the attitude controller which is a PID controller with the parameters defined in the original firmware of the Carzyflie. Equations 2.1 - 2.12 are used to simulate the dynamical behavior of the drone. The physical parameters of the Crazyflie that were used in the simulation are summarized in Table 2.1.

The LQR controller block shown in Figure 3.1 is used as a feedback controller that derives the deviation between the state vector X and the reference state X^r to zero. This deviation is defined as

$$\tilde{X} = X - X^r$$

And the error state equation, similar to Eq. 3.6, becomes

$$\dot{\tilde{X}} = A\tilde{X} + B\tilde{u} \tag{3.15}$$

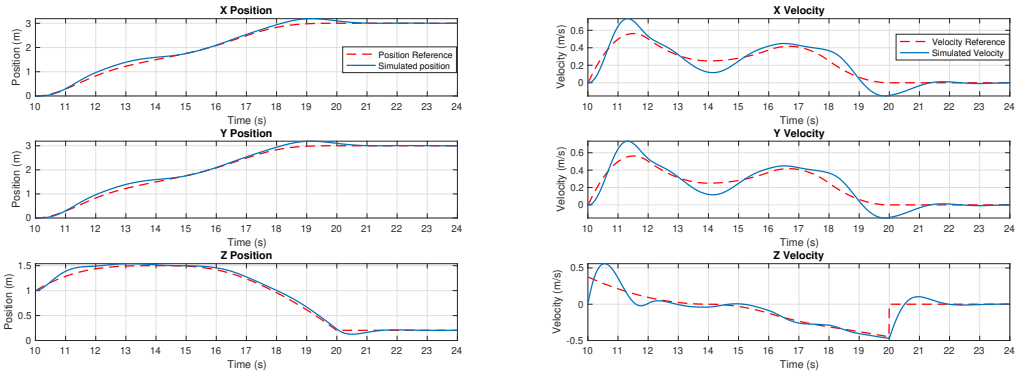
A full state feedback controller is used where $\tilde{u} = -K\tilde{X}$. By using Bryson's rule [33] and lqr command in Matlab, the K matrix that minimizes [33]

$$J = \int_0^\infty \tilde{X}^T Q\tilde{X} + \tilde{u}^T R\tilde{u}\, dt \tag{3.16}$$

is found. Where the Q and R are the symmetric positive-definite weighting matrices.

Applying the architecture shown in Figure 3.1 in SIMULINK, we were able to verify the designed controller and simulate the expected behavior of the Crazyflie as shown

below. Figure 3.2 shows the reference and output position of the Crazyflie and shows the commanded and output velocities for the given spline.



((a)) Positional error for trajectory tracker

((b)) Velocity error for trajectory tracker

Figure 3.2 : Tracking performance of the LQR differential flatness based controller (simulation)

3.1.3 Experimental results

For the validations and real experiments with agile nano-sized quadrotors, a test environment under a VICON Motion Capture System with 8 cameras was set up as seen in Figure 3.3. Crazyflie drones were used for multi-UAV coordination. The test case involves a problem such that the drones need to pass through a virtual 20 *cm* radius circle in the air. We can catch the real flown positional logs through VICON Motion Capture System with a positional error under a centimeter. Figure 3.4 shows the architecture used for the implementation, where the Robot Operating System (ROS) packages were customized after [27] as to be compatible with the schematic of the experimental setup. The splines were started at $t = 10s$ and at a hovering position in order to avoid from the ground effect and to guarantee the stability of the system.

In the given test case, as the merged point is the same for all drones, their paths are expected to converge at some point. The B-Spline control point and absolute knot vectors are shared with the nearby drones, therefore allowing them to calculate the trajectories of other drones in the fleet. This allows them to predict the point of collision if there is one. A priority ID is assigned to the drones by each other based on their closeness to the point that the trajectories intersect. Then, the drones intuitively scale up/down their own knot vectors to pass the intersection point without

collision. We have chosen to use integer values for time, meaning that drones put at least 1-second intervals to pass over the merge points (i.e., the center of the virtual circle) for proper trajectory synchronization.

Figure 3.3 : Test Environment with 8 motion capture cameras (VICON V16) and 3 nano-size drones (Crazyflie)

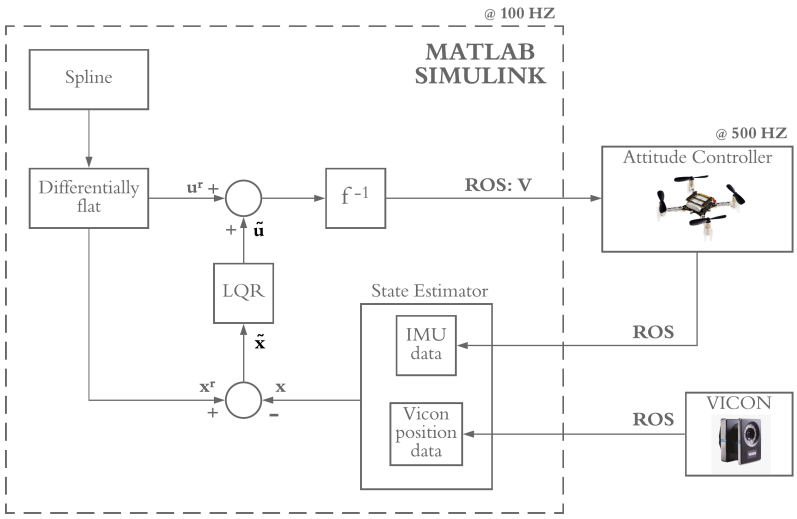


Figure 3.4 : Complete architecture of the system

Figure 3.5 shows the time synchronized splines followed by each Crazyflie where their target was to pass through the center of the 20 *cm* radius circle. In order to evaluate trajectory time synchronization sensitivity, the flight test shown in Figure 3.6 was repeated 4 times and the result trajectories are given in Figure 3.7. The average time error is evaluated based on Figure 3.7 to be roughly 0.15 seconds.

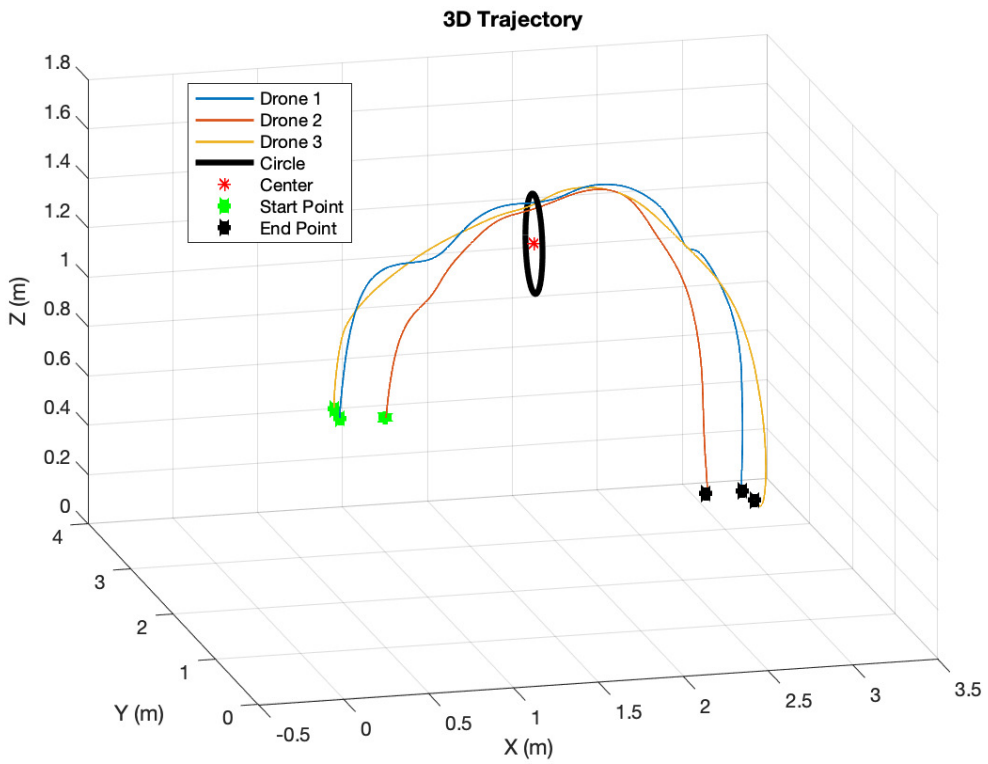
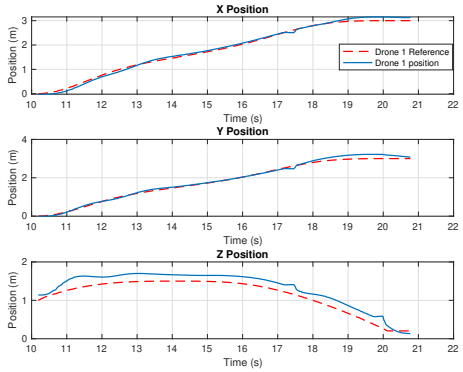
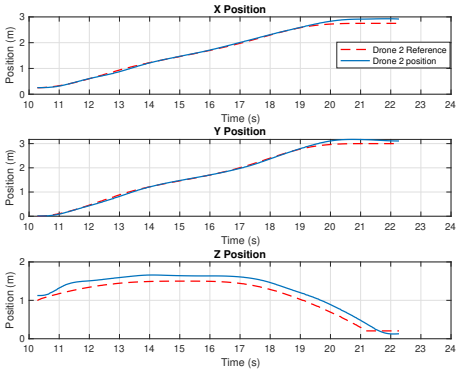


Figure 3.5 : Real flown trajectories of the vehicles plotted through Motion Capture System

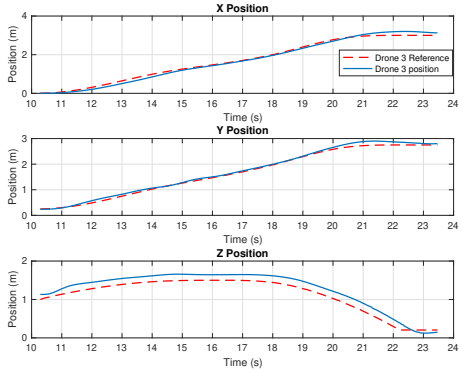
Figure 3.6 show the trajectory results of the first, second, and third drones as well as the three drones flying together in a 2D prospective.



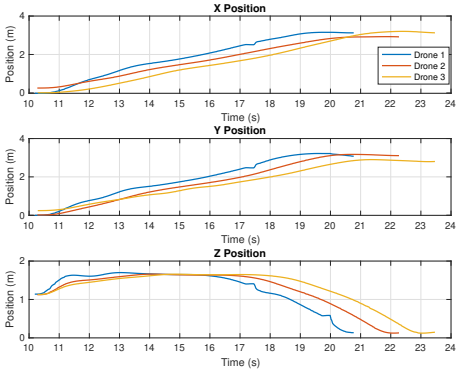
((a)) X-Y-Z trajectory components of Drone 1



((b)) X-Y-Z trajectory components of Drone 2



((c)) X-Y-Z trajectory components of Drone 3



((d)) X-Y-Z trajectory components of all drones together

Figure 3.6 : X-Y-Z trajectory tracking performance of all drones together

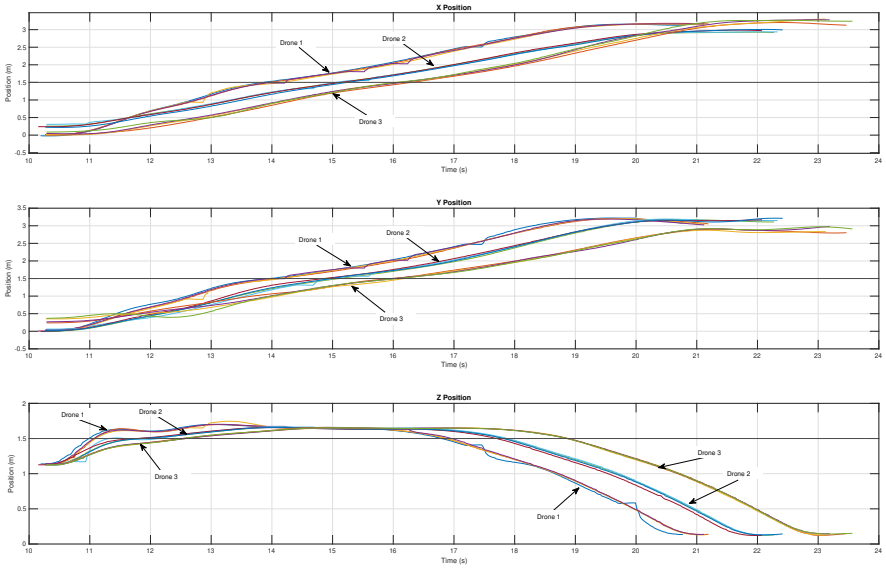


Figure 3.7 : Multiple trials for generated trajectories - black lines shows center of X-Y-Z location of the virtual circle in the air

3.2 LQI Differential Flatness Based Controller

The LQI differential flatness based controller follows the same approach defined in section 3.1, however, the state-space representation is now modified to include the integration of the position states in order to minimize the tracking error in agile maneuvers, as follows:

$$\tilde{X} = \begin{bmatrix} x_{error} & y_{error} & z_{error} & \dot{x}_{error} & \dot{y}_{error} & \dot{z}_{error} & \psi_{error} & \int x_{error} & \int y_{error} & \int z_{error} \end{bmatrix}^T \tag{3.17}$$

And the state-space model becomes:

$$\dot{\tilde{X}} = A\tilde{X} + B\tilde{u} \tag{3.18}$$

Where

$$A = \begin{bmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times1} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times1} & 0_{3\times3} \\ 0_{1\times3} & 0_{1\times3} & 0_{1\times1} & 0_{1\times3} \\ I_{3\times3} & 0_{3\times3} & 0_{3\times1} & 0_{3\times3} \end{bmatrix}$$

$$B = \begin{bmatrix} 0_{3\times3} & 0_{3\times1} \\ I_{3\times3} & 0_{3\times1} \\ 0_{1\times3} & 1 \\ 0_{3\times3} & 0_{3\times1} \end{bmatrix}$$

and

$$u = u^r + \tilde{u} \tag{3.19}$$

Where u^r is the feed-forward acceleration and u is the control input defined in [4]. Finally, $\tilde{u} = -K\tilde{X}$ which is used as full state feedback controller. By using the *lqr* command in Matlab, the K matrix that minimizes

$$J = \int_0^\infty \tilde{X}^T Q \tilde{X} + \tilde{u}^T R \tilde{u}\, dt \tag{3.20}$$

is found. Where the Q and R are the symmetric positive-definite weighting matrices and they were obtained by using Bryson's rule [33].

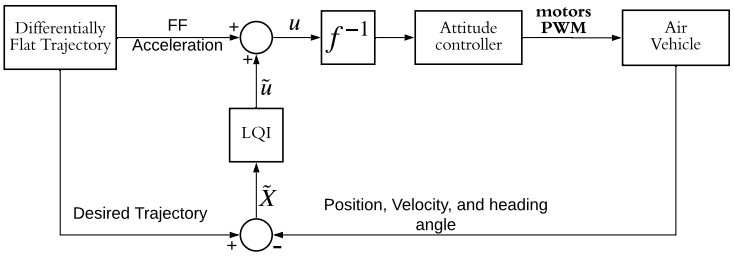


Figure 3.8 : Architecture of the LQI differentially flatness based controller

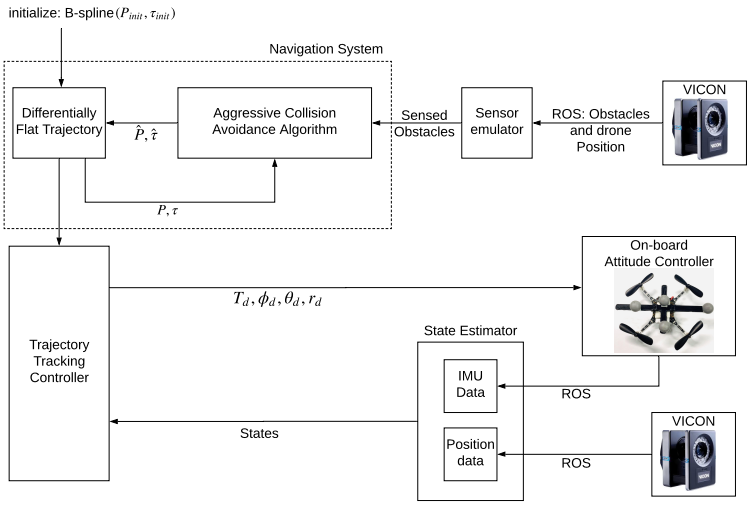


Figure 3.9 : The system architecture for the hardware implementation

3.2.1 Experimental results

The approach defined above was implemented as a trajectory tracking algorithm in a collision avoidance problem published in [5]. Figure 3.9 shows the architecture used in the implementation. The initial desired B-spine trajectory reference is sent to the trajectory tracking controller by the navigation system. The VICON motion capture system is used at a 100 HZ to obtain the drone and the obstacles positions. The off-board trajectory tracking controller, explained in Figure 3.8, takes the desired trajectory and the estimated states of the air vehicle and sends the desired thrust and angles to the on-board attitude controller which runs at 500 Hz.

Typically, aerial vehicles have limited range and field of view (FOV). Because of the forward and horizontal acceleration that the air vehicle can produce, the sensor needs to have more FOV, on the other hand, in that physically maximum acceleration situations,

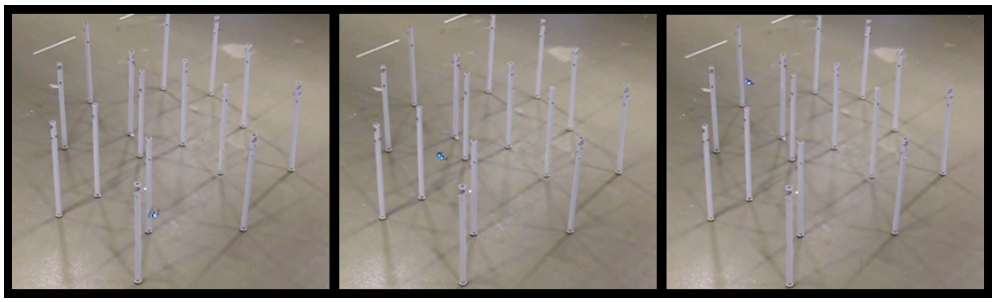


Figure 3.10 : Frames from the flight test. The obstacles density in the environment is $\rho = 1.2$ obstacle/m^2.

there is no way that air vehicle can sense the obstacles around itself [34]. Thus, $1m$ range and 20° FOV is used for the sensor emulator.

The navigation system works at 100 HZ an Intel core i7 CPU. It generates the trajectory references for the trajectory tracking controller. If there are any obstacles in the sensor FOV, the aggressive collision avoidance algorithm is triggered by the sensor emulator. Then, the algorithm generates collision avoidance maneuver by only relocating newly added control point.

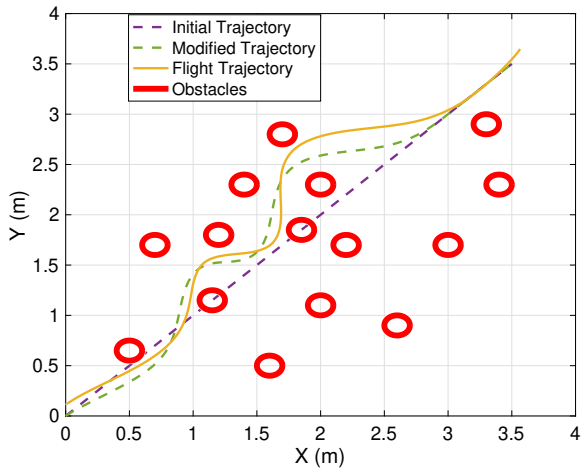
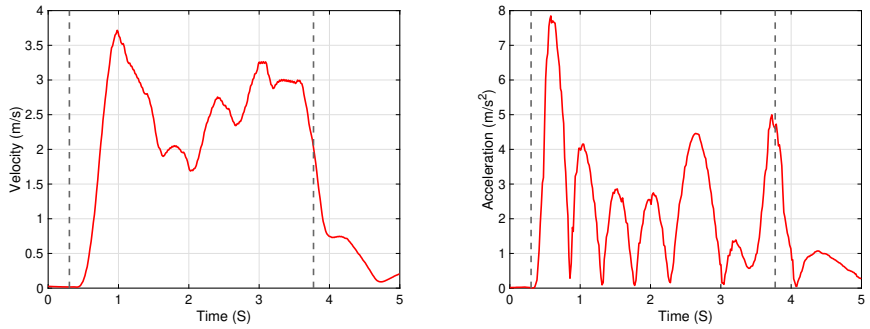
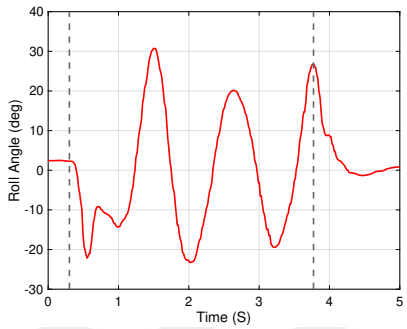


Figure 3.11 : The initial given straight line is shown in purple, the modified trajectory (RL agent output) is shown in green, the air vehicle's flight trajectory is shown in orange, and the obstacles are shown in red.

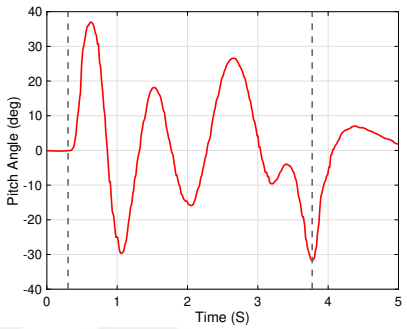


((a)) Velocity

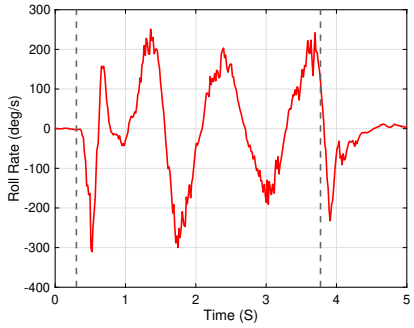
((b)) Acceleration



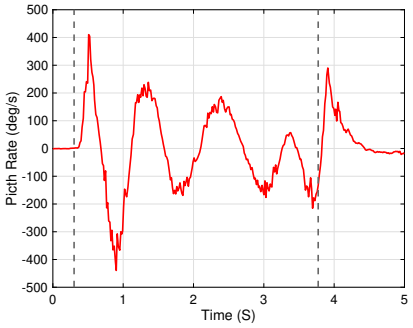
((c)) Roll Angle



((d)) Pitch Angle



((e)) Roll Rate



((f)) Pitch Rate

Figure 3.12 : The dashes indicate the start and the end of the scenario.(a) The achieved maximum velocity is $3.74 m/s$, and (b) the maximum acceleration is $7.8 m/s^2$. (c) A peak roll angle of $30.5 deg$ was achieved during flight. (d) A peak pitch angle of $38.1 deg$ was achieved during flight.(e) A peak roll rate of $248.23 deg/s$ was achieved during flight. (f) A peak pitch rate of $409.8 deg/s$ was achieved during flight.

The aggressive collision avoidance and the trajectory tracking algorithms are tested in a $\rho = 1.2$ obstacle density environment, as shown in Figure 3.10. The starting point was at $[0,0]$ and the goal point was at $[3.5,3.5]$. The initialized B-spline trajectory was a straight line, but after the detecting obstacles several times and running the algorithm, the result reference trajectory is shown in Figure 3.11. Since the problem is solved

locally and only the segment of the trajectory containing the obstacles is re-planned, we see in Figure 3.11 that the modified trajectory converges to the initial trajectory after each obstacle it avoids. Three different scenarios were implemented to test how robust the collision avoidance and the trajectory tracking algorithms are, however, the plots of only one scenario are shown in Figure 3.11 and Figure 3.12. The rest of the scenarios are shown in the video found in `https://youtu.be/8IiLQFQ3V0E`.

4. DEEP REINFORCEMENT LEARNING BASED TRAJECTORY TRACKING CONTROLLER

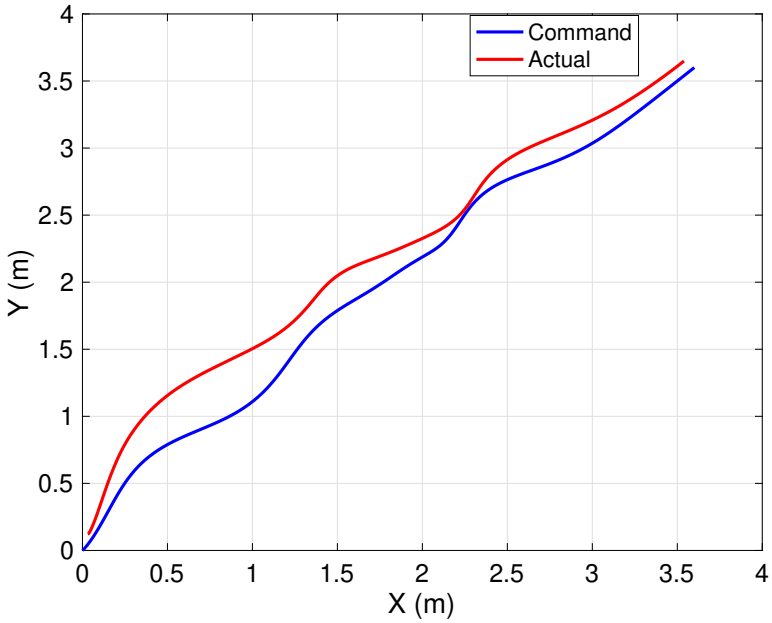


Figure 4.1 : The reference trajectory is given in blue and the actual trajectory flown is given in red

After conducting the experimental results shown in section 3.2.1, it became evident that in the case of very agile trajectories, the LQI based differential flatness algorithm will not produce the optimum results when it comes to trajectory tracking performance. It is shown in Figure 4.1 that there is $21cm$ RMS error which in fast aggressive flights will definitely cause a collision. This poor performance takes us away from the locally optimal solution in the cases when fast collision avoidance is crucial. Therefore it became vital to find a solution to achieve a precise trajectory tracking performance which will be discussed in this chapter.

4.1 Proposed Architecture

Proposed system architecture which utilizes deep reinforcement learning agent to generate pitch and roll references to track the desired trajectory is shown in Figure 4.2.

The trajectory generation block generates many random agile trajectories, following the algorithm defined in 2.3.1, consisting of position and velocity references that meet the aerial vehicle dynamical constraints given in section 2.3.2. The trajectory reference is then given to the agent along with the observation vector and the reward as defined in Eq. 4.3-4.4 respectively. The agent then generates outputs for the desired roll and the pitch angles while the desired yaw angle is kept fixed as zero. The agent's commanded roll and pitch angles are achieved by a PID attitude controller that is defined in 2.2. The aerial vehicle altitude was handled by a separate PID controller which along with the attitude controller outputs the forces and moments acting on the aerial vehicle. Eq. 2.3-2.10 are then used to simulate the aerial vehicle dynamics and outputs the reward and the observation vector to close the training loop.

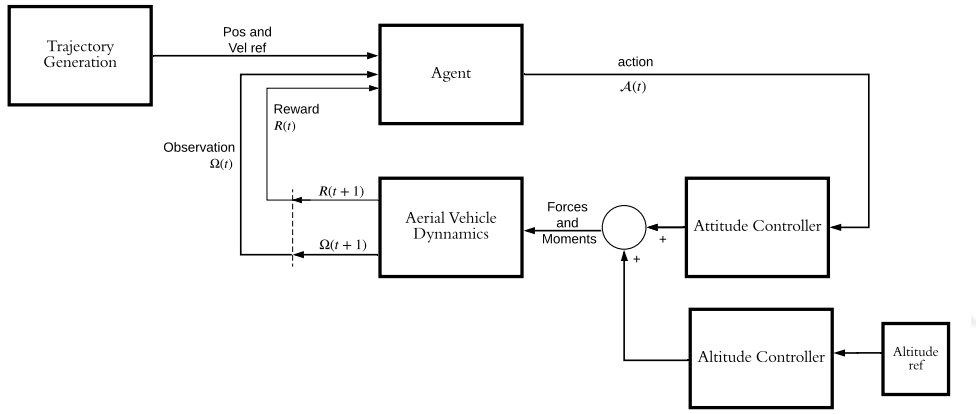


Figure 4.2 : Proposed system architecture

4.2 Proximal Policy Optimization Approach

Reinforcement learning main objective is to maximize the accumulated reward by learning how to adjust the policy after each episode. The problem is formulated as a Markov Decision Process (MDP) and the agent learns the optimal policy by interacting with the unknown environment.

The approach used in this paper is based on generating commands for pitch and roll angles by Proximal Policy Optimization (PPO) algorithm [35]. PPO is one of the policy gradient methods in deep reinforcement learning which is easy to use and has good performance. In order to make use of the benefits of trust region policy

optimization (TRPO) such as reliable performance and data efficiency, PPO uses the first-order algorithm [35]. Define the probability ratio, $r_t(\theta)$, as follows:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad r(\theta_{old}) = 1 \tag{4.1}$$

The work in [35] proposes a new objective function defined as follows:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right] \tag{4.2}$$

where epsilon ε is a hyper free parameter that we chose as 0.2. The term $clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t$ allows us to adjust the surrogate objective by saturating the probability ratio. The reason this clip function is used is to prevent overdoing good actions and prevent giving zero future probability to actions that are considered bad in the current step.

For the training part of the DRL, observation, action, and reward, vectors are defined as follows:

In order to track an agile trajectory, both the position and velocity error were considered in the observation space. The difference between the previous two actions was also added to the observation space in order to minimize the rate of change of the agent's actions. If the rate of change is not considered in the observation vector and in the reward function, the agent will output the commanded pitch and roll angles at a very high angular velocities which cannot be achieved by the attitude controller.

In our solution, altitude is handled by a PID controller, and we are expecting high performance from the agent to track the trajectory in the x and y-axes. The observation space is described as follows:

$$\Omega = \{P_x(t) - x(t),\ P_y(t) - y(t),\ \dot{P}_x(t) - u(t),\ \dot{P}_y(t) - v(t),\ \mathscr{A}(t-1) - \mathscr{A}(t-2),\ x(t),\ y(t),\ z(t)\} \tag{4.3}$$

Where $\mathscr{A}(t-1)$ is the action given by the agent one time step ago and $\mathscr{A}(t-2)$ is the action given by the agent two time step ago.

Action space consists of commands of roll and pitch angles. The agent's action is then given as $\mathscr{A} = \{\phi_{cmd}, \theta_{cmd}\}$.

In this trajectory tracking problem, we need to reward or penalize the agent by a reward function. Each action decision results in a new state, which also needs to depend on these parameters. Therefore, to achieve high performance in trajectory tracking, we defined the reward function as the following:

$$R = 2\sum_{i=1}^{5}(e^{-\alpha_i \Omega(i)} - 1) \tag{4.4}$$

Where Ω is the observation vector defined in Eq. 4.3 and α is a coefficient that determines the curvature of the reward function and helps in deciding when the agent gets a positive reward based on the error between the defined trajectory and the aerial vehicle state. Table 4.1 shows the selected α values and the corresponding error value that determines when the agent gets a high positive reward.

Table 4.1 : α values.

α_i	Value	Positive reward when
α_1	0.027	$\Omega(1) < 4$ cm
α_2	0.027	$\Omega(2) < 4$ cm
α_3	0.0085	$\Omega(3) < 50$ cm/s
α_4	0.0085	$\Omega(4) < 50$ cm/s
α_5	4	$\Omega(5) < 600$ deg/s

In training, generated trajectories are limited by $x = [0.0, 3.5]$ and $y = [0.0, 3.5]$. Therefore, in order to save extra unneeded training time, the environment boundaries were set as follows:

$$0.0 < x(t) < 4.0\ m$$

$$0.0 < y(t) < 4.0\ m$$

$$0.0 < z(t) < 2.0\ m$$

If the aerial vehicle exceeds any of these boundaries, the agent gets a reward of -100.0 in the following step and then the episode is terminated.

4.3 Simulation Experiments

For training a DRL agent, we used OpenAI gym and stable-baseline environment [36] [37]. The training was done for approximately 335 thousands episodes ($\approx$ 100 M steps) and it was done on a 20 core CPU workstation which took around 3 days.

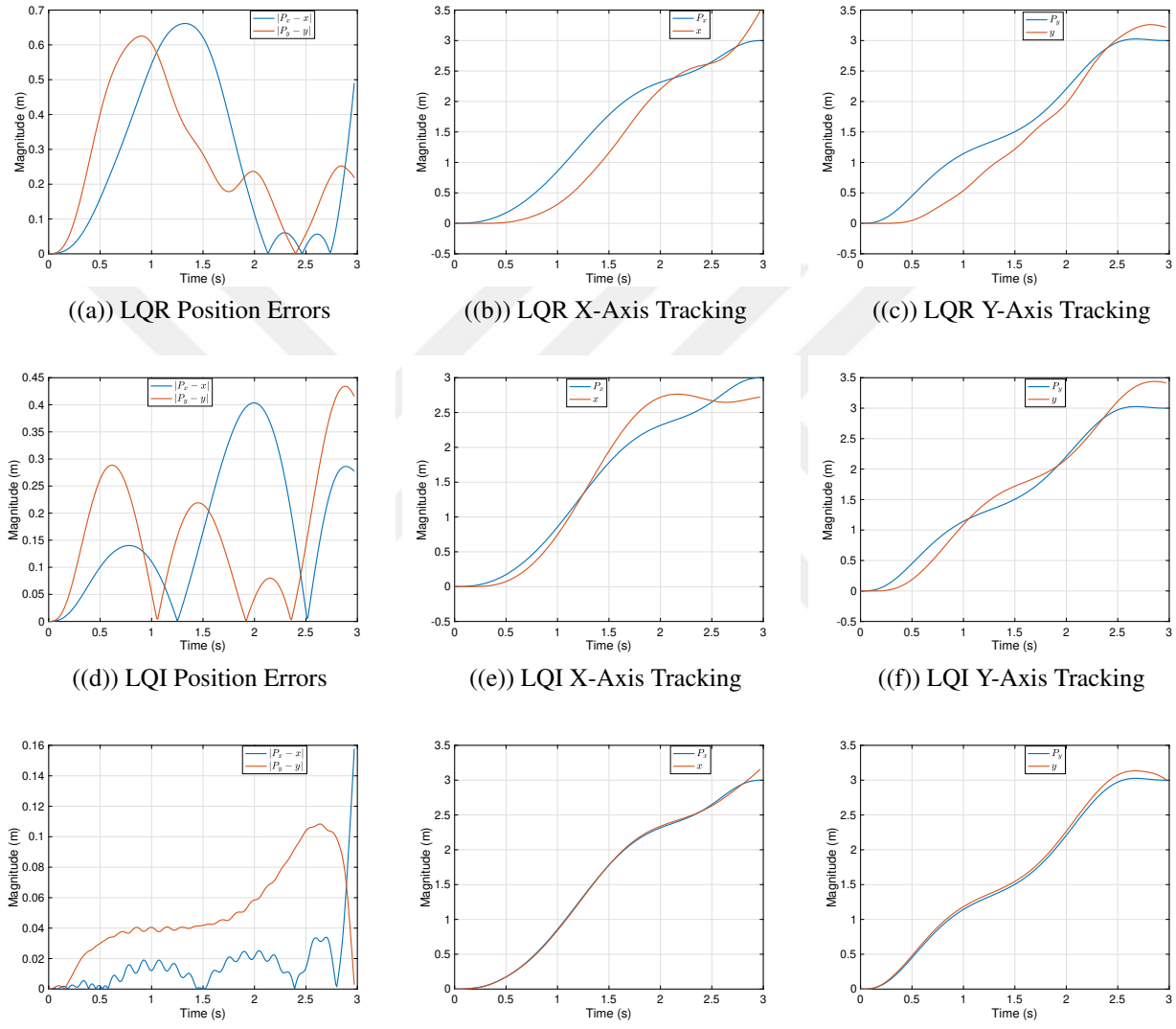
We used LQR and LQI differential flatness based controllers defined in sections 3.1 and 3.2 to compare performances with the DRL based solution.

The aerial vehicle was modeled using Eq.2.3 - 2.10 and by using the physical parameters defined in Table 2.1 in a Matlab/Simulink environment. This model as explained earlier in section 2.1 accepts the forces and moments acting on the Crazyflie in the body-fixed frame, which is calculated by a low-level controller consisting of a PI controller in attitude and a PID controller in attitude rate . The altitude of the Crazyflie was controlled by a PID controller to keep it at a fixed altitude. A separate model was built for the LQR/LQI differential flatness based controllers, explained in sections 3.1 and 3.2 in order to compare the tracking performance between these two approaches and the DRL approach.

Figure 4.3 shows the simulation results and the controllers position tracking performance, and Figure 4.4 shows the results for velocity tracking performance. The trajectory shown in Figure 4.3 and 4.4 is one of the randomly generated trajectories and the maximum achieved acceleration while tracking it is 6.2 m/s^2 which was achieved at a velocity of $0.7m/s$. From Figure 2.5, we see that the maximum feasible acceleration for the Crazyflie at a velocity of $0.7m/s$ is $7m/s^2$. 500 Monte Carlo simulations are conducted to compare the RMS error of the proposed solution with LQR/LQI differential flatness based controllers which is shown in Table 4.2.

Table 4.2 : RMS Error Values Comparison (LQR, LQI, DRL).

RMS Value	LQR based Diff Flatness	LQI based Diff Flatness	DRL approach
$\lvert P_x - x \rvert$ [cm]	35.4	21	2.5
$\lvert P_y - y \rvert$ [cm]	33.1	20	5.8
$\lvert \dot{P}_x - u \rvert$ [m/s]	1.14	0.6	0.25
$\lvert \dot{P}_y - v \rvert$ [m/s]	1.04	0.65	0.16

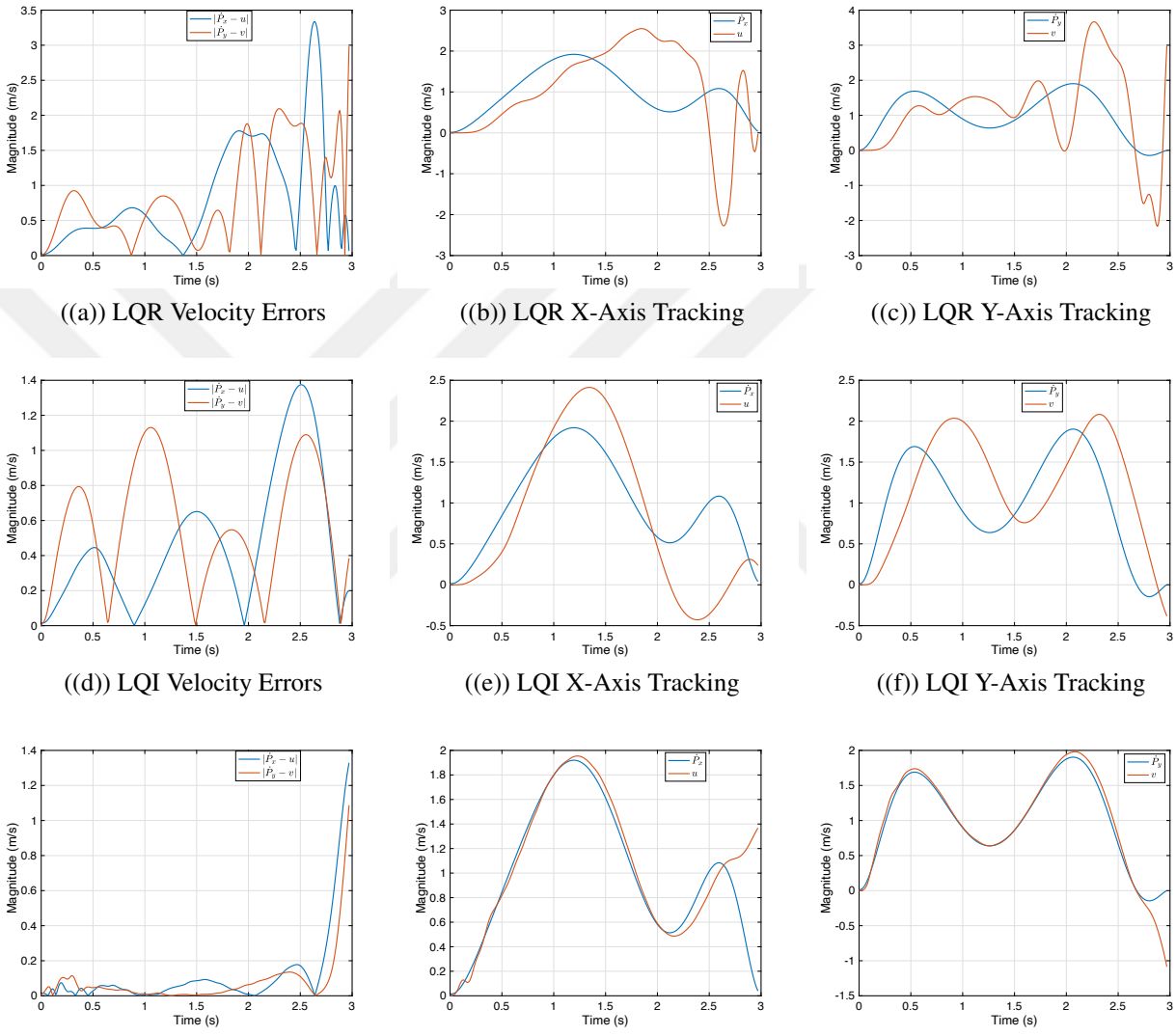


((a)) LQR Position Errors ((b)) LQR X-Axis Tracking ((c)) LQR Y-Axis Tracking

((d)) LQI Position Errors ((e)) LQI X-Axis Tracking ((f)) LQI Y-Axis Tracking

((g)) RL Position Errors ((h)) RL X-Axis Tracking ((i)) RL Y-Axis Tracking

Figure 4.3 : Comparison between LQR, LQI, and DRL based trajectory tracking controllers (position)



((a)) LQR Velocity Errors ((b)) LQR X-Axis Tracking ((c)) LQR Y-Axis Tracking

((d)) LQI Velocity Errors ((e)) LQI X-Axis Tracking ((f)) LQI Y-Axis Tracking

((g)) RL Velocity Errors ((h)) RL X-Axis Tracking ((i)) RL Y-Axis Tracking

Figure 4.4 : Comparison between LQR, LQI, and DRL based trajectory tracking controllers (velocity)

5. CROSS ENTROPY BASED TRAJECTORY TRACKING CONTROLLER

Table 4.2 shows the DRL approach improved the tracking performance dramatically when compared to the LQR and LQI based differential flateness approaches. However there is still a big room for improvements.

This chapter proposes the Cross Entropy optimization based trajectory tracking controller, enabling to minimize the positional and velocity track error for aerial vehicles. The same PI and PID controllers, explained in section 2.3, are utilized for the attitude and attitude rate controllers respectively. The trajectory generator based on the dynamic model guarantees the flat outputs, such that they do not exceed the given dynamical limitations of the vehicle, and produces pitch and roll references to the attitude controller. Simulation results presented in this chapter show the root mean square error of the trajectory tracking performance. Also, this approach is compared with the deep-reinforcement learning based trajectory tracking controller developed in chapter 4.

5.1 Proposed Architecture

The proposed system architecture which utilizes Cross Entropy optimization algorithm to generate pitch and roll references to track the desired trajectory is shown in Figure 5.1. The trajectory generation block, explained in section 2.3.1, generates many random agile trajectories consisting of position and velocity references that meet the aerial vehicle dynamical constraints given in section 2.3.2. The trajectory reference is then given to the Cross Entropy optimization algorithm which generates the optimal desired pitch and roll angles to precisely follow the desired trajectory. A feedback compensator is also utilized and outputs the needed small corrections in the desired roll and pitch angles in order to compensate for un-modeled dynamics and outer disturbances from the environment. The aerial vehicle altitude was handled by a separate PID controller which along with the attitude controller outputs the forces and

moments acting on the aerial vehicle. Eq. 2.3-2.10 are then used to simulate the aerial vehicle dynamics and outputs its position and velocity to close the feedback loop.

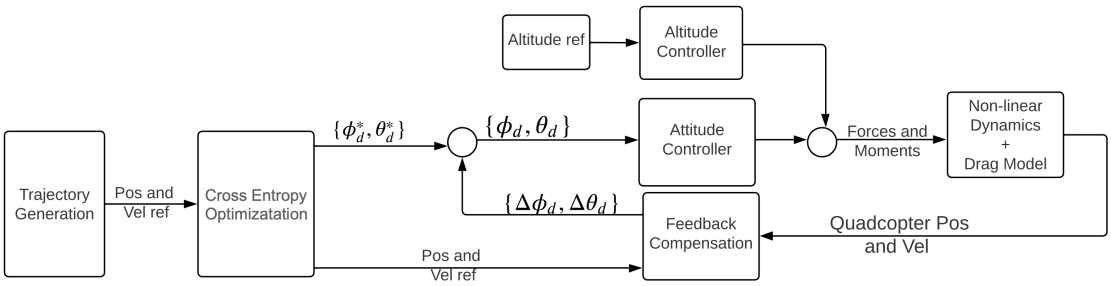


Figure 5.1 : Proposed system architecture

5.2 Cross Entropy Optimization Based Approach

Importance sampling, in which the sample distribution over the state space increasingly focuses on attractive locations, can further increase sampling performance. The sampling issue is transformed into a stochastic optimization issue of selecting an appropriate sample set that guides the algorithm to the trajectory with the lowest cost. We have incorporated the Cross Entropy (CE) technique to do this. The CE adaptive method, first presented in [38], uses variance minimization to predict the probabilities of uncommon events. With the help of an iterative process, CE initially creates a collection of samples from a given distribution before updating the relevant parameters. This process is repeated until the sample set's distribution resembles a delta function.

For our optimization problem, at each iteration t of the Cross Entropy algorithm, we will maintain a small set of solutions $P(t) = \{x_1^t, \ldots, x_n^t\}$, where $x_i^t = \{\phi_i^t, \theta_i^t\}$, t is an iteration number and n is arbitrarily chosen size of the sample. This set evolves naturally afterwards. In each generation, the relatively good solutions will be kept while the relatively bad ones will die out and will be reproduced. In order to determine which solutions are good and which ones are bad, the cost function is defined as

$$J = \frac{1}{2}((x_d - x)^2 + (y_d - y)^2 + (\dot{x}_d - \dot{x})^2 + (\dot{y}_d - \dot{y})^2) \tag{5.1}$$

and will play an important role in the optimization algorithm. Note that the states x, y, $\dot{x}$, and $\dot{y}$ used in the evaluation of the cost function, given in Eq. 5.1, are calculated based on the aerial veicle model described in section 2.1.

Let $Z \subset \mathbb{R}^n$ be a n-dimensional space to generate sample, and $f(.;v)$ be a probability density function. Consider following estimator,

$$\ell = \mathbb{E}[H(Z)] = \int_z H(z) f(z;v) \mathrm{d}z$$

where H is a measurable function. The problem was originally to find a trajectory with minimum cost, such that $J \leqslant \gamma$. Suppose that $\ell \in \mathbb{R}$ is very small real number. Hence, this formalism translates the problem into estimation of rare event probabilities, that is,

$$\ell = \mathbb{P}_v(J \leqslant \gamma) = \mathbb{E}_v \left[I_{\{J \leqslant \gamma\}} \right]$$

where the $I_{\{J \leqslant \gamma\}}$ is 1 if $J \leqslant \gamma, 0$ otherwise.

Algorithm 1: CE Optimization Algorithm

Input: Desired trajectory to be tracked
Output: $p^*(t)$: feasible optimal solution consisting of the desired attitude angles, $\{\phi^*, \theta^*\}$

1: $t = 0$
2: initialize $P(t)$
3: evaluate $P(t)$
4: **while** (**not** termination-condition) **do**
5: t = t+1
6: select $P(t)$ from $P(t-1)$
7: recombine $P(t)$
8: evaluate $P(t)$
9: **end while**
10: **return** $P^*(t)$

The CE optimization algorithm maintains a set $P(t)$ of some solutions $\{x_1^t, \ldots, x_n^t\}$ during iteration t. Note that the size n remains fixed for the duration of the computation. Each solution x_i^t is evaluated by computing $J(x_i^t)$ given by Eq. 5.1, which gives some assessment of the solution. Next, at iteration $t+1$ a new samples are formed by selecting solutions to reproduce on the basis of their relative assessment.

5.3 Simulation Experiments

We modeled the aerial vehicle using Eq.2.3 - 2.10 and by using the physical parameters defined in [27], we modelled the CrazyFlie in Matlab/Simulink environment. This model accepts the forces and moments acting on the Crazyflie in the body-fixed frame,

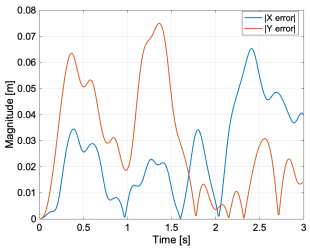
which is calculated by a low-level controller consisting of a PI controller in attitude and a PID controller in attitude rate. The altitude of the Crazyflie was controlled by a PID controller to keep it at a fixed altitude. We also compared our results to the deep reinforcement learning based trajectory tracker that we developed in [6]. Figure 5.2 shows the simulation results and the controllers position tracking performance, and Figure 5.3 shows the results for velocity tracking performance. The trajectory shown in Figure 5.2 and 5.3 is one of the randomly generated trajectories and the maximum achieved acceleration while tracking it is 6.2 m/s^2 which was achieved at a velocity of $0.7m/s$. From Figure 2.5, we see that the maximum feasible acceleration for the Crazyflie at a velocity of $0.7m/s$ is $7m/s^2$.

More Monte Carlo simulations will be conducted for the submission of the full paper to confirm the results found in this abstract.

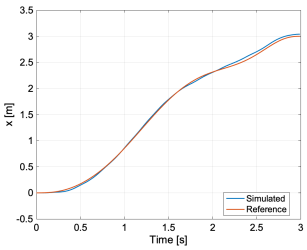
Table 5.1 shows a comparison of the root mean square (RMS) error between the proposed cross entropy based solution and the deep reinforcement based solution proposed in chapter 4.

Table 5.1 : RMS Error Values Comparison (DRL and Cross Entropy).

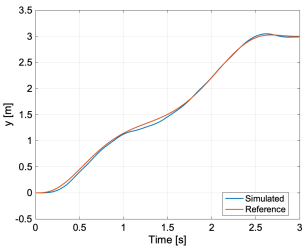
RMS Value	RL Based Trajectory Controller	Cross Entropy approach
$\lvert x_d - x \rvert$ [cm]	2.5	3.1
$\lvert y_d - y \rvert$ [cm]	5.8	3.6
$\lvert \dot{x}_d - \dot{x} \rvert$ [cm/s]	25	10.5
$\lvert \dot{y}_d - \dot{y} \rvert$ [cm/s]	16	15.5



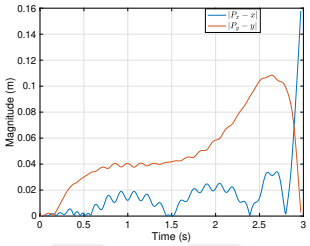
((a)) Proposed Algorithm Position Errors



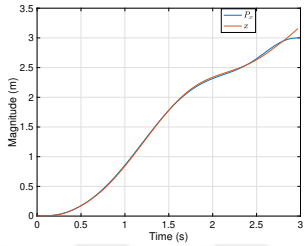
((b)) Proposed Algorithm X-Axis Tracking



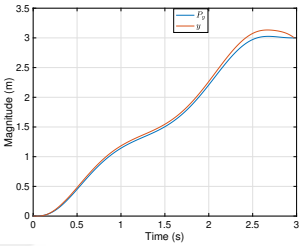
((c)) Proposed Algorithm Y-Axis Tracking



((d)) RL Position Errors

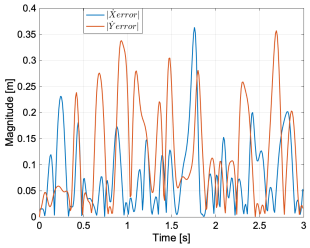


((e)) RL X-Axis Tracking

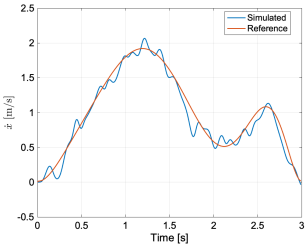


((f)) RL Y-Axis Tracking

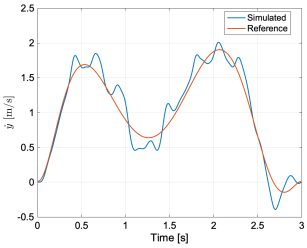
Figure 5.2 : Comparison between RL and cross entropy based trajectory tracking controllers (position)



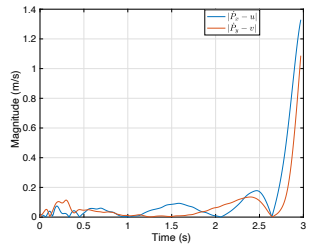
((a)) Proposed Algorithm Velocity Errors



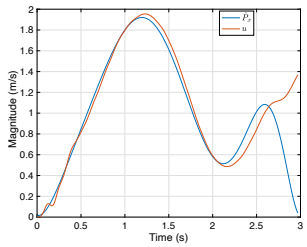
((b)) Proposed Algorithm X-Axis Tracking



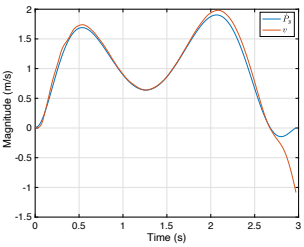
((c)) Proposed Algorithm Y-Axis Tracking



((d)) RL Velocity Errors



((e)) RL X-Axis Tracking



((f)) RL Y-Axis Tracking

Figure 5.3 : Comparison between RL and cross entropy based trajectory tracking controllers (velocity)

6. CONCLUSIONS AND FUTURE RECOMMENDATIONS

In conclusion, this thesis has addressed the critical issue of trajectory tracking for small unmanned aerial vehicles (UAVs) in the context of recent advancements in sensory systems, AI-based perception algorithms, and computational power. As the capabilities of UAVs continue to evolve, the need for precise trajectory tracking algorithms has become increasingly evident. These algorithms play a pivotal role in enabling small UAVs to navigate complex and dynamic environments, maintain stability, avoid obstacles, optimize energy consumption, and execute missions with efficiency and safety.

The research presented in this thesis has contributed to the field by proposing and evaluating three distinct trajectory tracking controllers. The first controller, based on differential flatness, incorporates LQR and LQI techniques. The second controller leverages deep reinforcement learning, specifically the Proximal Policy Optimization (PPO) algorithm, to minimize positional and velocity track error. Importantly, the third controller, which utilizes cross-entropy optimization, has emerged as the most successful in achieving the best results in terms of trajectory tracking performance. In all cases, low-level controllers, namely PI for attitude control and PID for attitude rate control, were employed to enhance the UAV's precision.

The findings of this research, as evidenced by simulation and real-time experimental results, underscore the importance of tailored trajectory tracking solutions for small UAVs. The superior performance of the cross-entropy optimization-based controller demonstrates its potential to outperform other methods and its applicability in various domains, such as aerial surveying, search and rescue operations, and environmental monitoring, where high situational awareness and precise control are imperative.

The development of these trajectory tracking algorithms represents a step towards unlocking the full potential of small UAVs across a wide range of fields. However, it is important to acknowledge that the pursuit of precise and generalized tracking algorithms remains a challenging endeavor. Future research should continue to explore

innovative solutions to further enhance the capabilities of small UAVs, thereby opening up new possibilities for their utilization in critical missions and applications.

REFERENCES

[1] (2022). Consumer Drone Market Size, Share Trends Analysis Report, https://www.grandviewresearch.com/industry-analysis/consumer-drone-market.

[2] **Faessler, M.**, **Franchi, A. and Scaramuzza, D.** (2018). Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories, *IEEE ROBOTICS AND AUTOMATION LETTERS*.

[3] **Hwangbo, J.**, **Sa, I.**, **Siegwart, R. and Hutter, M.** (2017). Control of a Quadrotor with Reinforcement Learning.

[4] **Shadeed, O.**, **Turkmen, H. and Koyuncu, E.** (2020). Trajectory-based Agile Multi UAV Coordination through Time Synchronisation, *AIAA Science and Technology Forum and Exposition*.

[5] **Hasanzade, M.**, **Shadeed, O. and Koyuncu, E.** (2022). Deep Reinforcement Learning based Aggressive Collision Avoidance with Limited FOV for Unmanned Aerial Vehicles, *AIAA Science and Technology Forum and Exposition*.

[6] **Shadeed, O.**, **Hasanzade, M. and Koyuncu, E.** (2021). Deep Reinforcement Learning based Aggressive Flight Trajectory Tracker, *AIAA Scitech 2021 Forum*, p.0777.

[7] **Shadeed, O. and Koyuncu, E.** (2023). Cross Entropy-based Aggressive Flight Trajectory Tracking Optimization, *European Control Conference*.

[8] **Dačić, D.B. and Kokotović, P.V.** (2006). Path-following for linear systems with unstable zero dynamics, *Automatica*, *42*(10), 1673–1683.

[9] **Do, K.D.**, **Jiang, Z.P. and Pan, J.** (2004). Robust adaptive path following of underactuated ships, *Automatica*, *40*(6), 929–944.

[10] **Dacic, D.B.**, **Nesic, D. and Kokotovic, P.V.** (2007). Path-following for nonlinear systems with unstable zero dynamics, *IEEE transactions on automatic control*, *52*(3), 481–487.

[11] **Dačić, D.B.**, **Nešić, D.**, **Teel, A.R. and Wang, W.** (2011). Path following for nonlinear systems with unstable zero dynamics: an averaging solution, *IEEE Transactions on Automatic Control*, *56*(4), 880–886.

[12] **Aguiar, A.P. and Hespanha, J.P.** (2007). Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty, *IEEE transactions on automatic control*, *52*(8), 1362–1379.

[13] **Do, K.D. and Pan, J.** (2006). Global robust adaptive path following of underactuated ships, *Automatica*, *42*(10), 1713–1722.

[14] **Skjetne, R.**, **Fossen, T.I. and Kokotović, P.V.** (2004). Robust output maneuvering for a class of nonlinear systems, *Automatica*, *40*(3), 373–383.

[15] **Skjetne, R.**, **Fossen, T.I. and Kokotović, P.V.** (2005). Adaptive maneuvering, with experiments, for a model ship in a marine control laboratory, *Automatica*, *41*(2), 289–298.

[16] **Banaszuk, A. and Hauser, J.** (1995). Feedback linearization of transverse dynamics for periodic orbits, *Systems & control letters*, *26*(2), 95–105.

[17] **Nielsen, C. and Maggiore, M.** (2006). Output stabilization and maneuver regulation: A geometric approach, *Systems & control letters*, *55*(5), 418–427.

[18] **Nielsen, C. and Maggiore, M.** (2008). On local transverse feedback linearization, *SIAM Journal on Control and Optimization*, *47*(5), 2227–2250.

[19] **Ferrin, J.**, **Leishman, R.**, **Beard, R. and McLain, T.** (2011). Differential flatness based control of a rotorcraft for aggressive maneuvers, *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.2688–2693.

[20] **Cabecinhas, D.**, **Cunha, R. and Silvestre, C.** (2009). Rotorcraft path following control for extended flight envelope coverage, *Proceedings of the 48th IEEE Conference on Decision and Control, held jointly with the 28th Chinese Control Conference (CDC/CCC)*, 3460–3465.

[21] **Akhtar, A.**, **Waslander, S.L. and Nielsen, C.** (2013). Fault Tolerant Path Following for a Quadrotor, *IEEE 52ND ANNUAL CONFERENCE ON DECISION AND CONTROL (CDC)*.

[22] **Faulwasser, T. and Findeisen, R.** (2016). Nonlinear model predictive control for constrained output path following, *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, 1026–1039.

[23] **Rub´ı, B.**, **Morcego, B. and P´erez, R.** (2020). A Deep Reinforcement Learning Approach for Path Following on a Quadrotor.

[24] **Ng, A.**, **Coates, A.**, **Diel, M.**, **Ganapathi, V.**, **Schulte, J.**, **Tse, B.**, **Berger, E. and Liang, E.** (2004). Autonomous inverted helicopter flight via reinforcement learning, *in International Symposium on Experimental Robotics*.

[25] **Koch, W.**, **Mancuso, R.**, **West, R. and Bestavros, A.** (2018). Reinforcement learning for UAV attitude control, *http://arxiv.org/abs/1804.04154*.

[26] Crazyflie 2.1, `https://www.bitcraze.io/crazyflie-2-1/`.

[27] **Luis, C. and Le Ny, J.** (2016). Design of a Trajectory Tracking Controller for a Nanoquadcopter, *Technical report, Mobile Robotics and Autonomous Systems Laboratory, Polytechnique Montreal*.

[28] **Subramanian, G.P.** (2015). Nonlinear control strategies for quadrotors and CubeSats (M.S. Thesis, University of Illinois at Urbana-Champaign).

[29] **Cox, M.G.** (1972). The numerical evaluation of B-splines, *IMA Journal of Applied Mathematics*.

[30] **Boor, C.D.** (1972). On calculating with B-splines, *Journal of Approximation theory*.

[31] **Piegl, L. and Tiller, W.** (2012). *The NURBS book*, Springer Science & Business Media.

[32] **Cano, J.M.** (2013). Quadrotor UAV for wind profile characterization, *Universidad Carlos III deMadrid*.

[33] **Franklin, G.F.**, **Powell, J.D. and Emami-Naeini, A.** (2020). *Feedback control of dynamic systems*, Pearson.

[34] **Lopez, B.T. and How, J.P.** (2017). Aggressive collision avoidance with limited field-of-view sensing, *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp.1358–1365.

[35] **Schulman, J.**, **Wolski, F.**, **Dhariwal, P.**, **Radford, A. and Klimov, O.** (2017). Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347*.

[36] **Dhariwal, P.**, **Hesse, C.**, **Klimov, O.**, **Nichol, A.**, **Plappert, M.**, **Radford, A.**, **Schulman, J.**, **Sidor, S.**, **Wu, Y. and Zhokhov, P.**, (2017), OpenAI Baselines, `https://github.com/openai/baselines`.

[37] **Hill, A.**, **Raffin, A.**, **Ernestus, M.**, **Gleave, A.**, **Kanervisto, A.**, **Traore, R.**, **Dhariwal, P.**, **Hesse, C.**, **Klimov, O.**, **Nichol, A.**, **Plappert, M.**, **Radford, A.**, **Schulman, J.**, **Sidor, S. and Wu, Y.**, (2018), Stable Baselines, `https://github.com/hill-a/stable-baselines`.

[38] **Rubinstein, R.Y. and Kroese, D.P.** (2004). The cross entropy method: A unified approach to combinatorial optimization, Monte-Carlo simulation (Information Science and Statistics), *Secaucus, NJ: Springer-Verla*.

CURRICULUM VITAE

Name Surname: Omar Shadeed

EDUCATION:

- **B.Sc.:** 2019, Istanbul Technical University, Aeronautical and Astronautical Faculty, Astronautical Engineering
- **M.Sc.:** 2024, Istanbul Technical University, Graduate School of Science, Aeronautical and Astronautical Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2018 - 2019 Boeing Undergraduate Fellowship
- 2019 - 2022 Control Systems Research Engineer, ITU-ARC
- 2019 - 2020 Boeing Graduate Fellowship
- 2022 - 2023 Model Based Systems Engineer, Boeing
- 2023 - present Aircraft Systems Architect, Bombardier

PUBLICATIONS:

- **Shadeed, O.**, Koyuncu, E. Cross Entropy-based Aggressive Flight Trajectory Tracking Optimization, European Control Conference, June 13-16, 2023, Romania
- Hasanzade M., **Shadeed, O.**, Koyuncu, E. Deep Reinforcement Learning based Aggressive Flight Trajectory Tracker, AIAA Scitech 2022 Forum, January 3-7, 2022, USA
- **Shadeed, O.**, Hasanzade M., Koyuncu, E. Deep Reinforcement Learning based Aggressive Flight Trajectory Tracker, AIAA Scitech 2021 Forum, January 11-15, 2021, USA
- **Shadeed, O.**, Turkmen H., Koyuncu, E. Trajectory-based Agile Multi UAV Coordination through Time Synchronisation, AIAA Scitech 2020 Forum, January 6-10, 2020, USA