ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

HIGH-SPEED TRAJECTORY REPLANNING AND TRAJECTORY TRACKING FOR COLLISION AVOIDANCE

Ph.D. THESIS

Mehmet HASANZADE

Department of Aeronautics and Astronautics Engineering

Aeronautics and Astronautics Engineering Programme

MARCH 2021

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

HIGH-SPEED TRAJECTORY REPLANNING AND TRAJECTORY TRACKING FOR COLLISION AVOIDANCE

Ph.D. THESIS

Mehmet HASANZADE
(511162110)

Department of Aeronautics and Astronautics Engineering

Aeronautics and Astronautics Engineering Programme

Thesis Advisor: Asst. Prof. Dr. Emre KOYUNCU

MARCH 2021

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ÇARPIŞMA ÖNLEMEK İÇİN YÜKSEK HIZLI ROTA PLANLAMA VE ROTA TAKİBİ

DOKTORA TEZİ

Mehmet HASANZADE
(511162110)

Uçak ve Uzay Mühendisligi Anabilim Dalı

Uçak ve Uzay Mühendisligi Programı

Tez Danışmanı: Asst. Prof. Dr. Emre KOYUNCU

MART 2021

Mehmet HASANZADE, a Ph.D. student of ITU Graduate School student ID 511162110, successfully defended the dissertation entitled "HIGH-SPEED TRAJECTORY RE-PLANNING AND TRAJECTORY TRACKING FOR COLLISION AVOIDANCE", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Asst. Prof. Dr. Emre KOYUNCU** Istanbul Technical University	.....
Jury Members :	**Asst. Prof. Dr. Ramazan YENİÇERİ** Istanbul Technical University	.....
	Prof. Dr. İbrahim ÖZKOL Istanbul Technical University	.....
	Dr. M. Umut DEMİREZEN Roketsan	.....
	Asst. Prof. Dr. Hakan AKÇA Ege University	.....

Date of Submission : 13 Feb 2021
Date of Defense : 10 March 2021

To my family,

FOREWORD

I would like to begin with expressing my gratitude to my thesis advisor Prof. Emre KOYUNCU for his support and guidance throughout this study. I am thankful for his invaluable advice. It is a precious experience and a milestone to work with him during my Ph.D. progress in ITU Aerospace Research Center.

I would like to thank Prof. Ramazan YENİÇERİ for his encouragement and support in many projects that we conducted.

I would like to thank Prof. Gökhan İNALHAN for expanding my vision through my academic life.

I can not forget my labmates, without them my Ph.D. progress would have been much more overwhelming and stressful. I would like to thank Aykut ÇETİN, Emre SALDIRAN, and Ömer HEREKOĞLU for being perfect teammates in many of our projects. I hope we will have the opportunity to work together again in the future. I would like to thank Emre SALDIRAN (again) and Omar SHADEED for their contribution and support to my thesis. Also I would like to thank ITU Aerospace Research Center family, especially Arınç Tutku ALTUN, Yunus BİÇER, Mevlüt UZUN, Güney GUNER, Barış BAŞPINAR who have been always there to help and encourage me.

I would like to thank and apologize to all my friends, instead of sharing my time with them, I had to devote all my time to work. Thanks for your patient, understanding, and the most important thing, your friendship during this long and hard path.

Last but definitely not least, I am really grateful and thankful to my lovely, irreplaceable parents Bayzid Ahmet HASANZADE, Mehtap HASANZADE, and my lovely sister Mine HASANZADE for their priceless understanding, patience, and endless support. Thanks to my father's teachings and the vision he brought me, I was able to reach this point.

March 2021 Mehmet HASANZADE

TABLE OF CONTENTS

Page

ABBREVIATIONS

CL-RRT*	**:** Closed Loop - Rapidly-exploring Random Tree
CPU	**:** Central Processing Unit
DDPG	**:** Deep Deterministic Policy Gradient
DRL	**:** Deep Reinforcement learning
ESC	**:** Electronic Speed Control
FOV	**:** Field Of View
GPS	**:** Global Positioning System
GNSS	**:** Global Navigation Satellite System
GPU	**:** Graphical Processing Unit
IMU	**:** Inertial Measurement Unit
INS	**:** Inertial Navigation System
MDP	**:** Markov Decision Process
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
RC	**:** Radio Control
RF	**:** Radio Frequency
RMS	**:** Root Mean Square
SLAM	**:** Simultaneous Localization And Mapping
TRPO	**:** Trust Region Policy Optimization

LIST OF TABLES

Page

LIST OF FIGURES

Page

HIGH-SPEED TRAJECTORY REPLANNING AND TRAJECTORY TRACKING FOR COLLISION AVOIDANCE

SUMMARY

Not long ago, the operations or the applications requiring high-performance guided and navigation would have required the use of tactical-size unmanned aerial vehicles. The main reason for this was that high-performance algorithms required bigger or heavier avionics with high computing capabilities or reliable communication buses linked with the ground systems. However, with the development of technology, these capabilities can now be achieved in smaller avionics, making it possible on-board for small-size unmanned aerial vehicles. New lightweight sensory systems enabled small unmanned systems to have advanced "situational awareness" and allow them to be capable of performing complex missions. Yet, guidance, navigation, and motion planning methodologies are still mostly "conservative" and "use-case-specific," render the UAVs incapable of performing multipurpose-operations.

There are many studies on route planning algorithms for situations where the map of the environment is known. Since these studies can operate in the initial phase of the operation, where a response is not expected to be very fast, it can guide the vehicle from the starting point to the end point safely, which is feasible and safe for the vehicle. However, in cases where unknown obstacles occur which can be sensed by any sensor, replanning of the trajectory is necessary in order to avoid obstacles. It is expected that the algorithms will be able to generate the replanned trajectory since the vehicle has less time to avoid it. Therefore, The time efficiency of the replanning phase is directly related to the speed and the aggressiveness of the trajectory followed by the vehicle. it is crucial to utilize an algorithm that generates an evasive maneuver in real time and ensures safety and dynamical feasibility.

In this thesis, studies were carried out on two topics, trajectory replanning and trajectory tracking. The first study, this thesis proposes a fast re-planning strategy based on deep reinforcement learning for highly agile aerial vehicles. First, the differential flatness model of an air vehicle is utilized, allowing us to directly map the desired output trajectory, which is parameterized with b-spline curves, into required input states to track trajectory. Moreover, perception model is used with fixed range and FOV on the vehicle, and as soon as the vehicle detects the obstacle, it performs the real-time evasive action through repetitive re-planning over an infinite trajectory. Specifically, the algorithm is initialized with a flight trajectory plan, then performs optimal control point vector update and knot insertion to generate a dynamically feasible conflict-free trajectory. Through this modification, the regenerated trajectory provides feasible evasive maneuvers for the vehicle, where the location and the number of the added control points form the "agility" of this evasive maneuver. The control

point insertion considering dynamic constraints and the defined agility metrics is transformed into a trajectory optimization problem, which is solved through deep reinforcement learning (DRL). The proximal policy optimization (PPO) method is utilized to train the re-planner with the random forest generation environment. The agent produces re-planned dynamically feasible conflict-free trajectories with modified control points approximately in 400us, which enables the real-time flight trajectory generation for highly agile aerial vehicles.

The second study proposes 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. PI controller is utilized for the attitude controller and PID for the attitude rate controller as the aerial vehicle's low-level controllers. 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 show the root mean square error of the trajectory tracking performance. Also, DRL agent performance is compared with LQR and LQI based trajectory tracking controllers.

ÇARPIŞMA ÖNLEMEK İÇİN YÜKSEK HIZLI ROTA PLANLAMA VE ROTA TAKİBİ

ÖZET

Son yıllarda artan teknolojik gelişmeler nedeniyle, insansız hava araçlarının yüksek performanslı güdüm ve navigasyon özelliklerine sahip olması beklenmektedir. Bu özelliklere sahip olan araçların, özellikle taktiksel boyutlarda olması istenmektedir. Elbette bu beklenti, hafif aviyonik komponentlerin araç üzerinde bulunmasını zorunlu kılmakta ve bu gibi aviyoniklerin beklenen algoritmik hesaplama ağırlığı kaldırması beklentisi doğurmaktadır. Bunun yanında da bu özellikleri barındıran araçların emir ve kontrolünün yer istasyonuyla sağlanması ve operasyon sürecinde haberleşmenin kopmadan ve istenilen müdahalelere olanak sağlar şekilde olmalıdır.

Günümüz teknolojisi ile bu beklentileri karşılamanın mümkün olduğunu söyleyebiliriz. Gelişmiş teknoloji ile yüksek hesaplama gücüne sahip aviyoniklerin artık daha hafif şekilde geliştirilebilmesi, taktiksel boyuttaki araçların istenilen özellikleri kazanmasına olanak sağlamıştır. Elbette, hesaplama gücünün bu denli ihtiyaç duyulduğu alanlarınından agresif rota takibi ve rotanın tekrar planlanması uygulamaları gösterilebilir. Agresif bir rotanın takibi esnasında karşısına çıkan bir engelden kaçınabilmesi için, araç çok agresif ve hızlı olduğundan, yine mikrosaniyeler mertebesinde rotanın tekrar planlanım engelden kaçması istenmektedir. Bunun yanında planlanan rotanın lokal olarak optimaliteyi sağlaması, üretilen rotanın kullanılan aracın dinamiklerini göz önüne olarak oluşturulması, rota planlama konusundaki önemli unsurlardan birkaçıdır. Bunun yanında da rotanın tekrar planlanması durumunda engele çarpmamak için üretilecek yeni rotanın arama alanının yeteri kadar büyük olması ve lokal optimaliteye ulaşabilmesi için, kullanılan rota takip algoritmasının da yüksek hassasiyet ile takibi gerçekleştirebilmesi gerekmektedir.

Ortamın haritasının bilindiği durumlar için rota planlama algoritmaları üzerine bir çok çalışma ortaya konulmuştur. Bu çalışmalar, operasyonun başlangıç fazında çalışabileceği için, çok hızlı bir şekilde cevap verilmesi beklenmediğinden, araç için uygun olan ve hiç bir yere çarpmadan güvenli şekilde başlangıç noktasından bitiş noktasına yönlendirebilmektedir. Ama bilinmeyen engellerin oluştuğu durumlarda engellerden kaçabilmek için tekrar planlama yapmak gerekmektedir. Bu hesabın yapılması da aracın takip ettiği rotanın agresifliğiyle direk olarak ilgilidir. Aracın yüksek hızlı olduğu durumlarda karşısına engel çıktığına önleyebilmesi için daha az zamana sahip olduğundan, uygulanması gereken algoritmaların bu özelliği sağlayabilmesi beklenmektedir.

Bu tez içerisinde engel çıktığı zaman rotanın tekrar planlanması problemine ve rota takip problemine çözümler önerilmiştir. Rotanın tekrar planlanması

problemine derin pekiştirmeli öğrenme metodu ile oluşturulan ajan kullanılarak çözüm önerilmiştir. Öncelikle kullanılan insansız hava aracının diferansiyel düzlük (differential flatness) özelliği kullanılarak, aracın pozisyon ve türevleri üzerinden rotanın tanımlaması yapılması sağlanmıştır. Bu özellik sayesinde rotanın yine üzerindeki bir noktanın konumu ve türevleri elde edildiğinde, aracın bu rotayı takip edebilmesi beklenmektedir. Rota planlama için de B-spline denilen rota planlama algoritması kullanılmıştır. B-spline tipinde bir rota, içerisinde bulunan kontrol noktaları sayesinde geometrik olarak rotanın şeklinin belirlenmesini ve bu noktalar arasındaki hız, ivme vs. gibi pozisyon türevlerinin belirlenmesine olanak sağlamaktadır. Bununla beraber B-spline tanımalaması, rota üzerindeki her noktanın türevlerinde de sürekli olmasını garanti etmektedir. Kontrol noktalarının konumaları değiştirilebildği için de istenilen geometrik şekil elde edilebilir ve bu özellik engellerden kaçınmak için algoritmaya olanak sağlayacaktır. Ek olarak insansız hava aracı üzerinde bir kamera sensörünün bulunduğu varsayılmıştır. Bu sensör bir emülatör olarak modellenip, sensörün menzili ve görüş açısı kısıtlanmıştır. İnsansız hava aracı B-spline rotayı takip ederken, önüne bir engel çıktığında, bu engel sensör emülatörü üzerinden algılanabilmektedir. Rotanın sadece algılanan engel etrafında planlanması sağlanabilmesi için yine engel üzerinde bir kontrol noktasının bulunması, daha optimal bir çözüm sağlayabilmektedir. Bu nedenle B-spline tanımlamasının bir özelliği olan düğüm ilavesi (knot insertion) metodu kullanılarak, başlangıçta üretilen B-spline rotasının pozisyon ve türevlerinde hiç bir şekilde değişmeden, fakat kontrol nokta sayısını bir arttırıp, bu yeni kontrol noktasını tam olarak engelin üzerinde veya rotanın algılanan engele en yakın noktasına konumlandırılması sağlanmıştır. Daha sonra derin pekiştirmeli öğrenme metoduyla eğitilen ajanın, bu yeni konumlandırılan kontrol noktası için yeni bir lokasyon belirlenmesi sağlanmıştır. Ajan eğitilirken ödül fonksiyonu, aracın tüm durum değişkenleri sınırları içerisinde kalması sağlandığı gibi, engele çarpmaması için belirlenen bir güvenlik uzaklığını sağlayacak şekilde kontrol noktasının yeni konumunu belirlenmesi sağlanmıştır. Bu şekilde tasarlanan bir simülasyon ortamında ajanın eğitimi sağlanmıştır. Buna ek olarak ajanın ürettiği yeni nokta, aracın dinamik limitlerinde sürekli olarak kalmasını sağlayabilmek için ve üretilen rotanın hali hazırda agresif olduğundan, aracı mümkün olduğunca en az şekilde çevikliğini arttıracak rotayı oluşturmayı amaçlamaktadır. Bu özellik, tüm rota boyunca karşısına çıkacak engeller için yeterli aksiyon setini mümkün olduğunca geniş tutabilmesi için kullanılmıştır. Yapılan eğitim sonucunda simülasyon ortamında testler yapılmış ve bir çok rastgele üretilen senaryo içerisinde üretilen rotanın, istenilen limitler içerisinde kaldığı gösterilmiştir. Aracın gerçek zamanlı olarak bu rotayı yeniden üretebilmesi için de bu problemin formülasyonu bu şekilde basitleştirilmiştir. Bu yaklaşım sayesinde, ajan $400us$'de çözüm üretebilmektedir ve yaptığımız literatür araştırmalarla karşılaştırdığımızda en hızlı çözümü bu metod üretmektedir. Simülasyonda performans kriterleri gösterildikten sonra, iç ortam da VICON sistemi altında Crazyflie aracı ile gerçek testler ve performansları gösterilmiştir. Bu sistemde de diferansiyel düzlük kontrolcüsü olan LQI kontrolcüsü kullanılmıştır. Önceki yapılan çalışmalarda LQR kontrolcüsü ile rota takibi yapıldığında kalıcı durum hatası barındırdığını, bu nedenle bu hatadan kurtulmak için de LQI kontrolcüsü kullanıldı. Daha sonra yarış insansız hava aracı tasarlanmış ve aviyonik şeması paylaşılmıştır. Bu araca da önerilen metodu kullanarak agresif rotalar üretilmiş ve sanal engellerden kaçılması sağlanarak, performansı gösterilmiştir. Bu testlerde araç üzerinde ArduPilot otopilotu kullanılmış ve üzerindek pozisyon ve hız kontrolcüsü kullanılarak testler

gerçekleştirilmiştir. Var olan bu kontorlcünün performansı yeterli görülmediği için de kontrolcü modifiye edilerek, pozisyon, hız ve ivme kontrolcüsü haline getirilmiş ve testler tekrarlanmıştır. Bu kontrolcü ile yapılan uçuşlarda aracın hem açısal hemde ivme değerleri olarak aracın limitlere dayandığını ve bu limitlerde engellerden kaçarak rota takibini gerçekleştirebildiği gösterilmiştir.

Yapılan çalışmalar sonucunda, dış ortam testlerinde araç üzerinde etkiyen rüzgar ve GPS'in sağladığı pozisyon ve hız değerlerindeki belirsizlikten ötürü, iç ortamda yapılan testlerle karşılaştırdığımızda beklenen performansı veremediği görüldü. Bu nedenle rota takip algoritması için derin pekiştirmeli öğrenme tabanlı bir çözüm geliştirilmiştir. Bu çözümde de B-spline ile üretilen agresif manevralara sahip rotanın pozisyon ve türevleri üretilmiş ve derin pekiştirmeli öğrenme ajanının bu referanslara bağlı olarak yunuslama ve yuvarlanma açılarını üretmesi sağlanmıştır. Bu yaklaşımla takip edilen rotalar yine LQR ve LQI diferansiyel düzlük kontrolcüleri ile de takip edilip performans karşılaştırılması yapılmış ve daha yüksek hassasiyet ile rota takibi yapabildiği gösterilmiştir.

1. INTRODUCTION

Especially with the development of battery and engine technology, unmanned aerial vehicle (UAV) have taken their place in our lives for a few decades. We frequently see its use in civil and entertainment areas such as observation, cargo transportation, cinematic shooting, game purposes, drone racing competitions, as well as in military fields such as obtaining information, observing and even using ammunition. With the development of the technology used on these small and large UAVs, it finds a place in new application areas, while it also enables it to be used with more advanced methods.

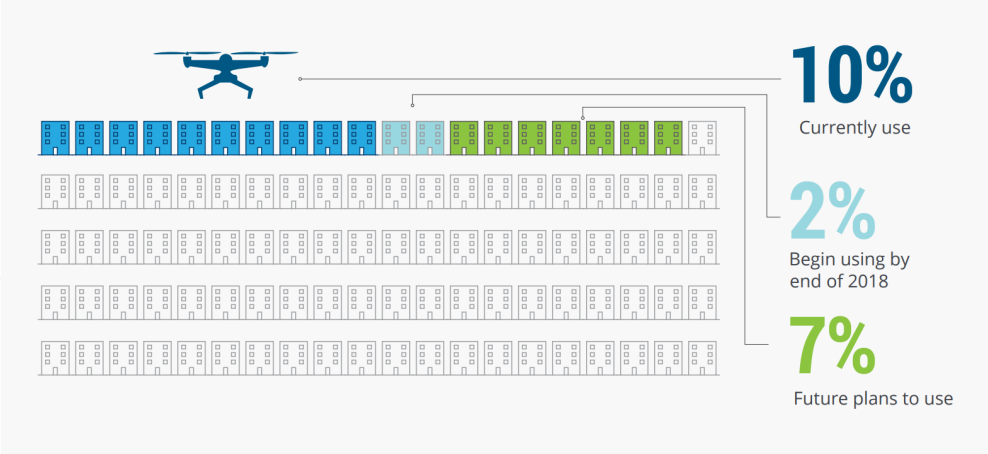


Figure 1.1 : Current and future UAV adaptation among companies with 50M+ revenue [1].

When we look at the UAV market, according to 2018 data, 10% of the companies that earned 50M + income were able to realize the UAV adaptation. 2% of these companies started using drones in late 2018, and 7% planned to use drones in their future plans which can be seen in Fig. 1.1. This shows that 19% of the surveyed companies are using or expecting to use UAVs in the future. This shows that the UAV market increases significantly every year and also creates new needs. [1]

In study conducted in 2018, it was observed that companies using UAVs achieved serious results and therefore increased their usage frequency in the next year. As an example, while there were 41% companies that made +10 flights in 2017, it rose to 54% in 2018, which is shown in Fig. 1.2 . When we look at the total, when the

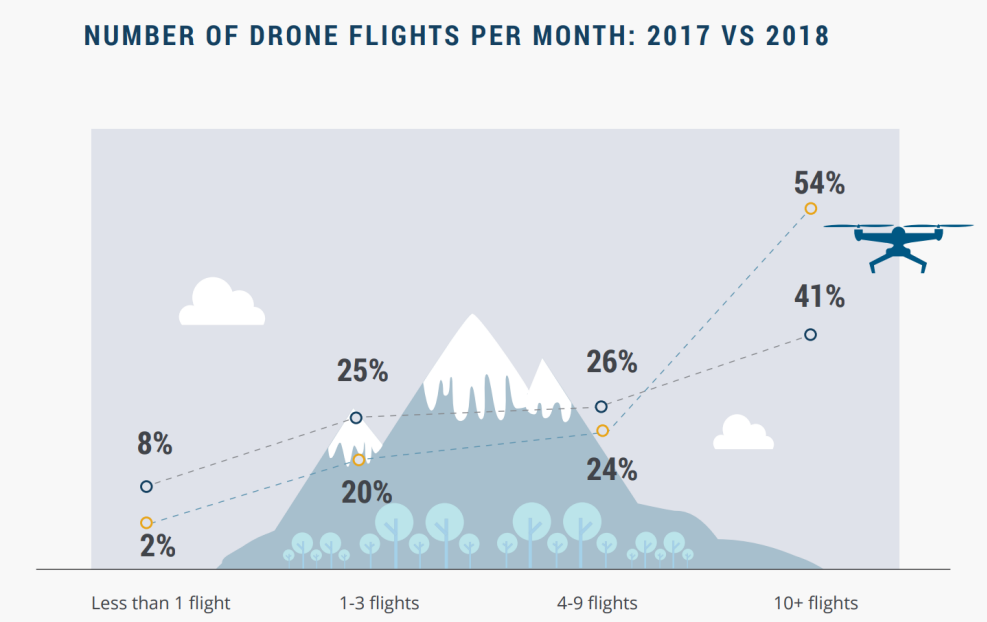


Figure 1.2 : Frequency of drone flights per month: 2017 vs 2018 comparison [1].

comparison is made between 2017-2018, it is seen that the total flight increased by 84%. Considering this result, also the companies that will be adapted, we can say that UAVs will appear in many areas. [1]

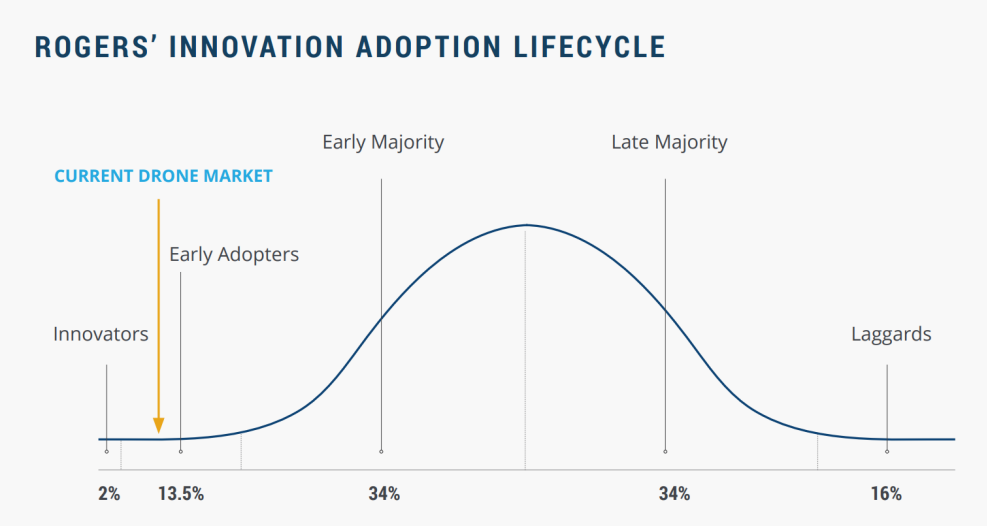


Figure 1.3 : Current UAV market in Rogers' innovation adoption lifecycle.

Also, it is important to mention that this study shows that when we look at the adoption percentage, this indicates that this is not established technology in the market, yet. [1] also shows that according to Rogers' innovation adoption life-cycle, the current drone market is in early adopters stage which is shown in Fig. 1.3 . It is an indication that the drone market will grow a lot in the future. [1]

Looking at the surveys conducted during market research seen in Fig. 2.1 , it is planned to do research and development on light detection ranging (LIDAR) and automation topics on UAVs [1]. Moreover, these two capability are also included in the ideas that attract the most attention of companies. Sense and avoid, collision avoidance and trajectory replanning studies, which have been hot topics recently and they are subtopic of ranging and automation, have been studied and many start-ups have been established in the light of these studies.

UAVs are becoming more and more capable of being used for complicated operations as the technology barriers in perception and computational power are being removed. Novel sensory systems integrated with AI-based perception algorithms allow small size UAVs to be capable of performing the missions requiring high-level situational awareness. Moreover, recent developments in the field of hardware such as GPUs with neural network architectures improve the computational abilities of unmanned systems to run demanding algorithms. Yet, the small UAVs are still incapable of performing such agile operations as the motion planning algorithms mostly built upon and work for specific cases. Therefore, the challenge in developing generalized agile trajectory planning and precise tracking algorithms remains open.

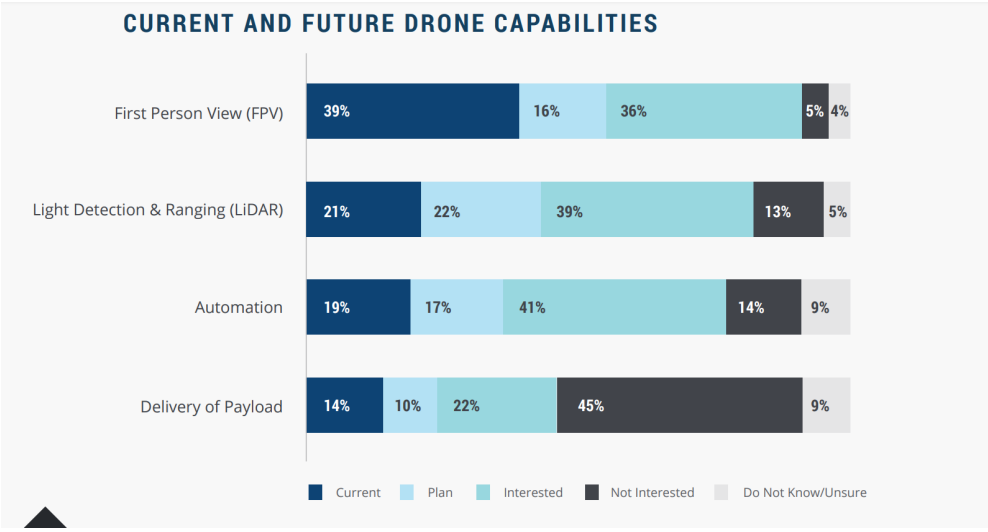


Figure 1.4 : Current and future UAV trends.

1.1 Aims and Objectives

This thesis proposes two different solution for trajectory replanning and trajectory tracking.

The first study, this thesis proposes a fast re-planning strategy based on deep reinforcement learning for highly agile aerial vehicles. First, the differential flatness model of an air vehicle is utilized, allowing us to directly map the desired output trajectory, which is parameterized with b-spline curves, into required input states to track trajectory. Moreover, perception model is used with fixed range and FOV on the vehicle, and as soon as the vehicle detects the obstacle, it performs the real-time evasive action through repetitive re-planning over an infinite trajectory. Specifically, the algorithm is initialized with a flight trajectory plan, then performs optimal control point vector update and knot insertion to generate a dynamically feasible conflict-free trajectory. Through this modification, the regenerated trajectory provides feasible evasive maneuvers for the vehicle, where the location and the number of the added control points form the "agility" of this evasive maneuver. The control point insertion considering dynamic constraints and the defined agility metrics is transformed into a trajectory optimization problem, which is solved through deep reinforcement learning (DRL). The proximal policy optimization (PPO) method is utilized to train the re-planner with the random forest generation environment. The agent produces re-planned dynamically feasible conflict-free trajectories with modified control points approximately in 400us, which enables the real-time flight trajectory generation for highly agile aerial vehicles. This proposed solution is published at Journal of Intelligent and Robotics Systems [11].

The second study proposes 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. PI controller is utilized for the attitude controller and PID for the attitude rate controller as the aerial vehicle's low-level controllers. 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 show the root mean square error of the trajectory tracking performance. Also, DRL agent performance is compared with LQR and LQI based trajectory tracking controllers. This proposed solution is published at AIAA Scitech Forum [12].

1.2 Thesis Outline

The rest of the paper organized as follows: Section-2 mentions about the literature survey for trajectory planning, re-planning and trajectory tracking. In Section-3, deep reinforcement learning based aggressive collision avoidance method is exlained, simulations, indoor tests and outdoor tests results are shared. Section-4 explains the trajectory tracking algorithm based on proximal policy optimization. This proposed architecture is compared with LQR and LQI differential flatness based controller.

1.3 Contributions to Knowledge

Continues Space within dynamical limits:
The proposed method utilizes B-spline trajectory representation which could be generated by using control points. One of the property of the B-spline enables to add new control point without changing trajectory position, velocity, acceleration and other derivatives. By using this property, we can change the curvature of the trajectory by changing the position of the control point. This allows us to find a solution in continues space. When we compare this with motion primitive selection from trajectory library, even these methods only ensures local optimality because of non prior knowledge of all obstacles, our method can generate more optimal trajectory because of searching in continues space.

Computation Time:
Computation time is one of the significant feature of trajectory replanning algorithms, which can allow to be used in more faster or more agile situations. The computational speed of the algorithm used may, in some cases, put in situations where the collision cannot be avoided. Because of this, some of the algorithms also propose safety time which also depends on the situation. This safety time allows to compare the time of the collision going to be occur and the time how much the computation may take. Of course how much the computation going to take have a major effect on the safety time. Therefore, the less computation time is more appropriate for agile and fast cases, where it also allows to find collision avoidance maneuver even in very limited space. It is also important to mention about the relation between sensors limited field of view and computation time. If the safety time is big, in agile cases, the obstacle may occur

in front of the vehicle very fast and not from the end of the view, but maybe in the middle. These cases only could be solved by having very little computation time that allows to generate new trajectory where the vehicle still have enough action space to act. Our proposed method takes 1.2 ms to prepare the header part and generate replanned trajectory which is fastest solution within my knowledge.

Conservative Planning:

Limited FOV only ensures the obstacles that could be sensed by sensor. The vehicle could not know if there are any other obstacles outside of this FOV. It would be dangerous to assume or believe that the outside of FOV is safe. This is why the proposed solution allows the generated replanned trajectory to avoid the obstacles inside the FOV by avoiding as close as possible and ensures the safety distance to the obstacles.

Computation Power:

The proposed solution can produce solutions quickly in time, and it can produce solutions at these speeds without the need for high-performance CPU. This property also allows the solution to be used for very light weighted vehicles where it is not needed to mount heavy electronics or computers.

2. LITERATURE REVIEW

2.1 Trajectory Planning and Replanning Literature

Optimization-based trajectory planning and re-planning are one of the approaches to generate safe non-collision trajectories, and some of these approaches are using differential flatness to formulate the output trajectory. This approach provides convenience to optimize dynamically feasible trajectories [13].

Optimizing polynomial path segments is also one of the path planning algorithm that allows to automatically tune the aggressiveness on each segment by changing time allocation for the segment [2]. The problem formulation modified as a unconstrained quadratic program which allows efficient computation. Polynomial trajectories are represented as differentially flat which allows to get rid of computationally comprehensive sampling and reduce the high dimensional state space of a quadcopter. The algorithm take samples in position only and use straight trajectory steering, then to compute polynomial trajectory, these samples become support points to generate desired trajectory. That approach allow the algorithm to compute faster, but it sacrifices the theoretical optimality. Also, meanwhile finding these support points, algorithm also checks if there are any collisions between these straight trajectories. [14] improves [2]

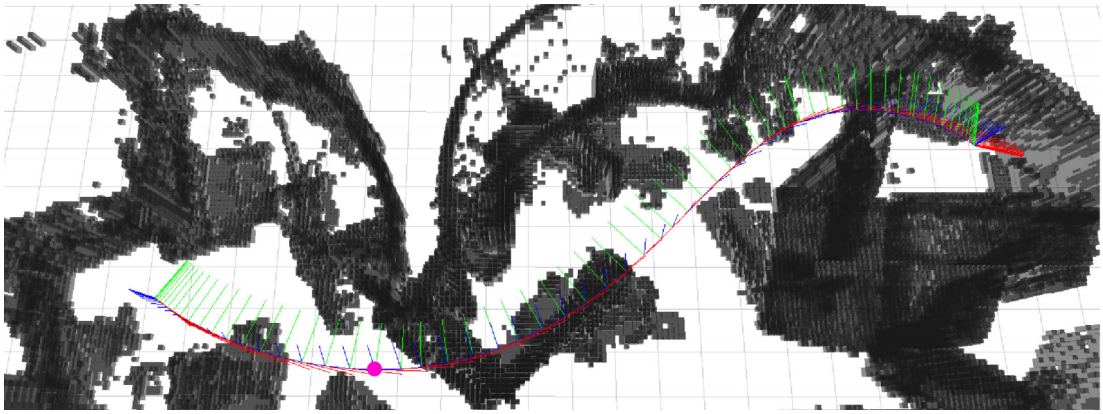


Figure 2.1 : Generated trajectory via optimizing polynomial path segments through a map of a laboratory environment [2].

algorithm by handling the derivatives of the position. Moreover, they use probabilistic approach to exceed uncertainties. They also decrease the computational time, which allows them to use this algorithm in more aggressive situations.

[15] generates time optimal trajectory for each degree of freedom. The trajectory feasibility for the vehicle is also constructed that includes the dynamical limits of the vehicle and input constraints. For computational efficiency, feasibility of the trajectory checked after the time-optimal jerk trajectories generated. Then, the trajectory replanned by tuning jerk constraints if any infeasible part is found.

Traditional planning approaches have combined exploration and SLAM [16] to allow the robot plan collision-free paths with a temporary goal and an accumulated map. [3] proposes a short range policy that utilizes similar approach. The difference is that the algorithm do not need the entire environment information, instead it uses the goal point, it generates nearby points as a temporary goal in known map. Convex decomposition of the local map is used to find these temporary goal and a safe trajectory shown in Fig. 2.2. Because of the temporary goals, this algorithm also allows the vehicle to plan or replan the trajectory when new map boundary data achieved by the sensor data. The algorithm also ensures the dynamical feasibility of the trajectory by using state constraints. [17] uses octree-based environment representation to define

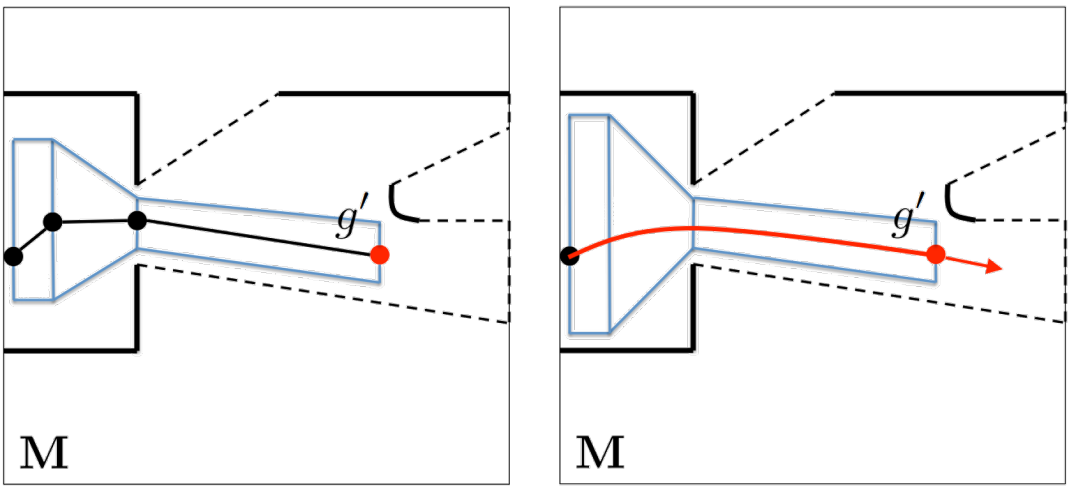


Figure 2.2 : Convex decomposition representation to find temporary goal in known map [3].

the environment. This environment incrementally built by using onboard sensors. Through known and perceived environment, polyhedral decomposition is used to define collision free space for convexification. These spaces are used the find free

corridors and generate minimum snap trajectory. This trajectory satisfies the safety of the path and the dynamical constraints of the vehicle.

[4] developed real-time optimal trajectory generation algorithm that generates flat outputs through the trajectory. The generated minimum snap trajectory also guarantees the position and its derivative constraints and ensures the trajectory to pass through desired corridors which are priori-known safe passages. On the other hand, [5] utilizes

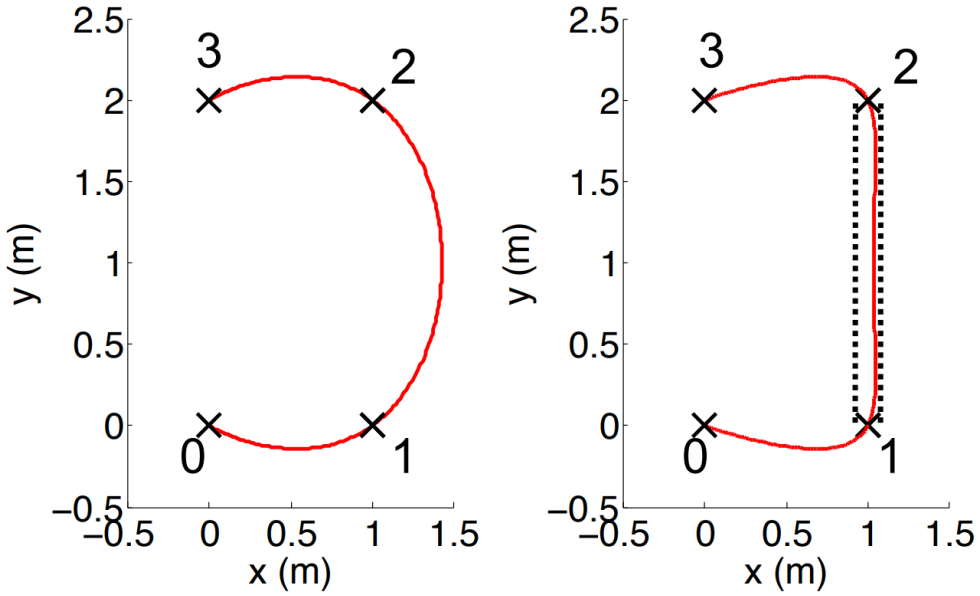


Figure 2.3 : Minimum snap trajectory generation. (left) ensures to pass through desired waypoints. (right) ensures to pass thorugh safe corridors [4].

kd-tree representation of the environment. [5] uses safety flight corridors to guarantee safety in 3D space. To generate these paths, sampling based path finding method is utilized. To generate this, first, RRG method is used to construct a randomly sampled graph where the centers of the ball shape sage regions are used as vertices. Then, these safe regions defined as ball shaped are used to present the safety corridors. Then, the intersection of these safe ball-shaped regions are used as a waypoint constraint shown in Fig. 2.4. To find these waypoints and generate the safe corridors, sampling-based path finding algorithm is used. A quadratically constrained quadratic programming (QCQP) are utilized to generate desired minimum jerk polynomial trajectory. This solution runs onboard within 100 milliseconds, which enables to be used in real-time flights. Another solution for the local replanning is the changing the B-spline control points to avoid the obstacles [18]. The obstacles in the environment are modelled as a occupancy grid and stored in a three dimentional circular buffer. The begining phase,

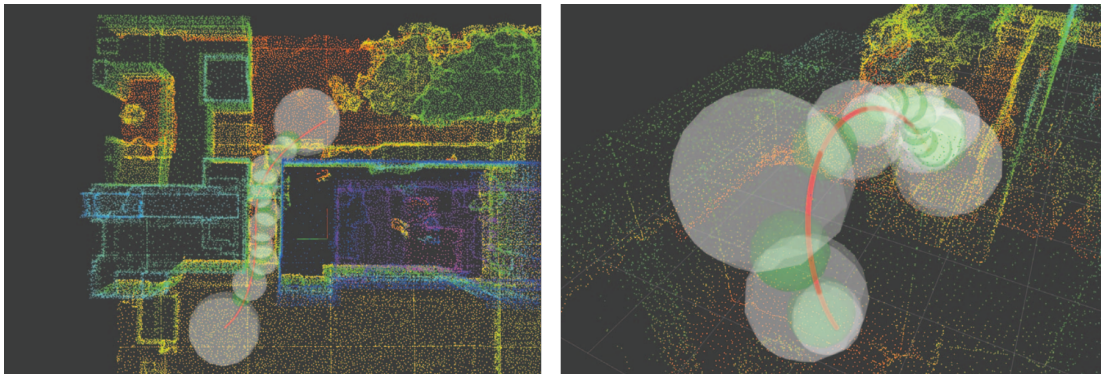


Figure 2.4 : Using ball shaped safe regions to find safety corridors [5].

straight uniform B-spline is defined from starting point to goal point. In flight, If any obstacles intersect with the trajectory, the control point positions are remodified where the number of the modified control points depend on the size of the obstacle. Also, this number directly dependent on the computation time of the proposed algorithm. Replacement of the control points are formulized as a unconstraint optimization which allows the algorithm work faster.

The Bézier curves are also another trajectory representation, and it can be utilized to generate a collision avoidance solutions. [19] integrated this representation for time coordination of the multiple vehicles. First, Pythagorean Hodograph Bézier curves are used to generate the desired trajectory that ensures the dynamical limits of the vehicles and allows the spatial separation between the trajectories. Collision avoidance with dynamic obstacles feature is formulated by adjusting the speed profiles of the vehicles. To improve this, [6] extends this approach with modifying the trajectory by adding an feasible detour where it ensures the safe passage which can be seen in Fig. 2.5. This generation of the detour also considers the constraints of the vehicles position, velocity and acceleration. Short range receding horizon control policy (SRRHCP) is utilized where the policy relies on a sampling based strategy that allows faster computation [20]. With the idea of short trajectories, it plans from the current state to the next waypoint which is also inside the field of view of the sensor. Because of the limited field of view, receding horizon control policy is decided to generate a trajectory in finite time horizon. The collision safety is ensured by using polyhedral decomposition of the visible free space. This method provides safe gaps in the visible map, and the next waypoint is determined within this safe free space. Also, in situations where there is no feasible trajectory to avoid the collision, stopping trajectories are also utilized to

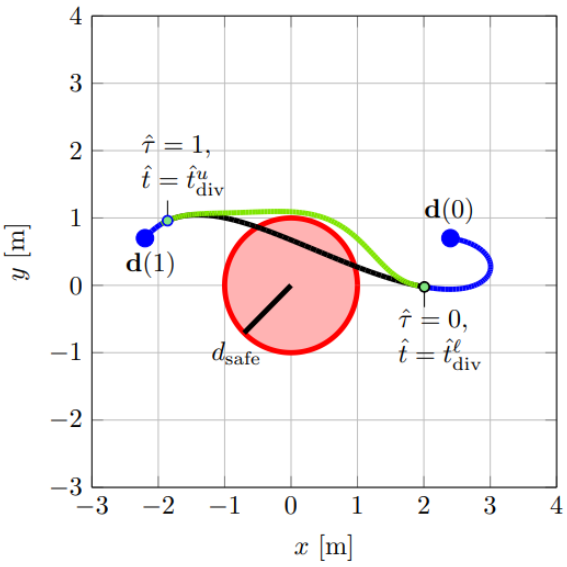


Figure 2.5 : Modifying trajectory by adding feasible detour [6].

allow the vehicle to shift into zero velocity state. Generated trajectories also allows the flight to be within the dynamical limits of the vehicle.

Another critic factor is the effect of the uncertainty on the trajectory. Even though this uncertainty might occur because of the performance of the trajectory tracking controller, attitude controller or the state feedback algorithms, it is also important to consider these situations where the uncertainty of the system may allow the vehicle to have a collision. Therefore, robust high-speed system is proposed [7] where the primitives are utilized to evaluate the paths probabilistically for possible collision which can be seen in Fig. 2.6. The motion primitives are applied in visible depth information, and each primitives are also calculated by adding the uncertainty of the state of the vehicle in corresponding time. These primitive calculations are also provided by triple-double integrator maneuver library. To allow real-time control, approximation methods are implemented with spatial partitioning data structures. These motion primitives can also generated by sampling the vehicle's control space that also allows the dynamically feasibility of the primitives [21].

One of the study focuses on the computationally light-weight motion primitive generation [22]. To overcome the computation weight, instead of including constraints in the planning phase, the constraint check is conducted after having calculated motion primitives. This approach allows faster computation and easy to implement since being modular. These primitives also ensured to be the minimum jerk solutions for

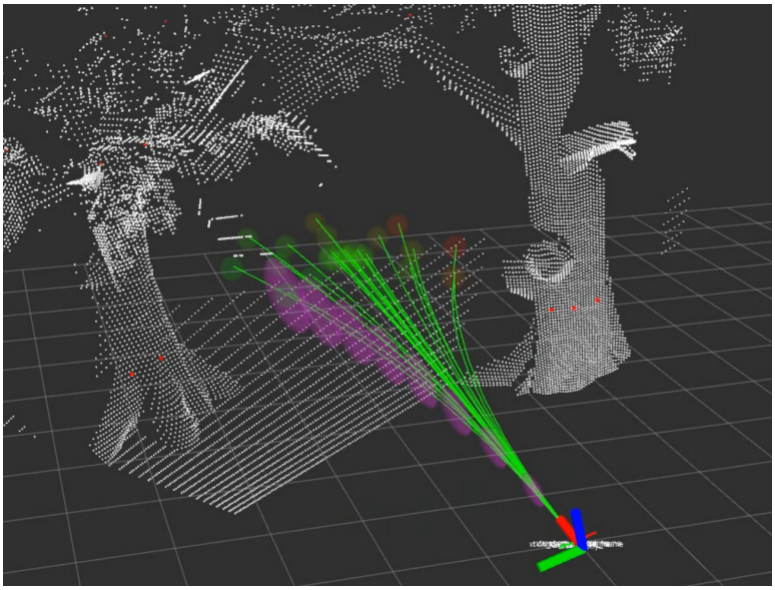


Figure 2.6 : Motion Primitives are utilized to evaluate the paths probabilistically for possible collision [7].

the current state to desired state. After achieving the primitives, dynamical feasibility is checked from each of the primitives.

Some of the work focus on not only how to choose a trajectory from the trajectory library, but how to create a search space which consist of a set of candidate paths [23,24]. This work especially shows that the mutual separation of the set of trajectories search has an effect on prior probability of the relative completeness. This problem formulated as a maximum k-facility dispersion problem, and a greedy search algorithm is used to produce a good set of paths.

[25] proposes incremental replanning algorithm that using previous computation. The motion primitives are generated off-line where the robot motion model is utilized. Differential flatness feature is used the decrease the dimensionality of the problem and the complexity of the dynamics of the vehicle. To generate motion primitives, greedy algorithm is proposed. This method also ensures the continuity of the shifting phase of the motion primitives where all primitives guarantees to be dynamically feasible for the vehicle. This work [8] shown in Fig. 2.7, also addresses a collision avoidance method where there are no priori information about the cluttered environment. By sampling terminal possible states, the generated motion primitive meets the minimum-time, input and state constraints. The proposed algorithm called triple integrator planner (TIP) allows to compute really fast solutions which allows to perform in real-time

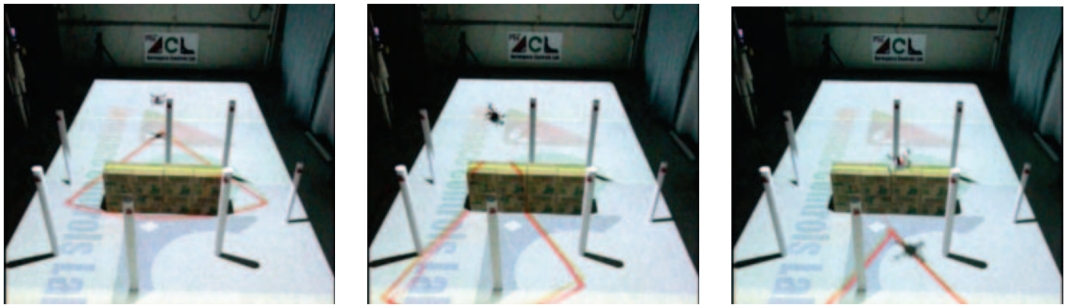


Figure 2.7 : Triple integrator planner generates aggressive manuevers [8].

flights. Also, this algorithm focuses on generating aggressive maneuvers where the vehicle is desired to finish the mission as soon as possible. Therefore, the fast solution generation also contribute to this sort of missions. Because, the expected motion primitives should ensure the minimum time intention, the motion primitives are generated by ensuring to reach the desired point in maximum feasible jerk or maximum feasible acceleration. Still, this approach has limitations because of the sensor field of view. [26] develop this new work by improving triple integrator planner which is called relaxed-contraint triple integrator planner. This method overcomes the limiting vehicle speed problem because of the perception sensor.

2.2 Trajectory Tracking Literature

After planning and replanning of the trajectory, to ensure the desired optimality of the results, trajectory tracking methods become crucial, especially in aggressive collision avoidance cases. The vehicle is expected to track the desired trajectory. Even though, the used path replanning methods may guarantee the local optimality, to ensure this optimality in flight, the trajectory tracking method should provide expected performance. To solve this problem, the trajectory tracking literature approaches are mostly based on Lyapunov/backstepping techniques [27–34] and geometric control [35–37].

In [38], the author implemented a LQR differential flatness based controller to track multiple trajectories with multiple UAVs. However, we saw that as the trajectories get more agile, the tracking performance drops significantly which showed the urgent need for a more agile trajectory tracking controller.

As steady state errors are often encountered in [38], LQI method is used to overcome this error [11]. Even though the steady state error is eliminated, the expected trajectory tracking performance could not be achieved.

Before the vehicles are expected to engage in aggressive behavior, classical or quadratic controller were meeting the expectation. The most important reason for this is that the vehicles were not facing with significant drag force, therefore without even computing this, suggested solutions in the past were providing expected results. However, with the expectation of the aggressive manuevers from the quadcopter, studies on this subject have appeared in the literature recently.

The proposed architecture in [39] utilizes LQR differential flatness based controller for the quadcopter platform. Because of the yaw rate capability of the quadcopters, arbitrary changes of the yaw is accepted, but small pitch and roll angles are not limited.

One of the study [9] focuses on linear rotor drag effects on the vehicle and proposes a solution based on differentially flat formulation of these drag effects via its position and heading. This feature is used to calculate the feed forward control terms from desired trajectory references. Nonlinear feedback control system is proposed to include these control terms to conduct a accurate aggressive flight with quadcopter shown in Fig. 2.8. This solution also shows results that the root mean squared tracking error is improved by 50% when compared with the results found in [39].

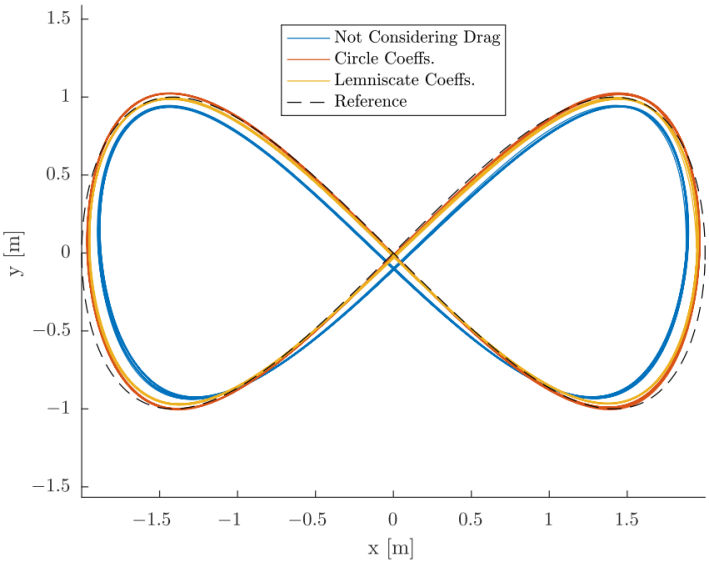


Figure 2.8 : Trajectory tracking performance considering rotor drag effects on the vehicle [9].

The proposed solution in [40] examine a solution for guarantee applicable zero thrust actuation that still converges to the desired trajectory, ensure that the actuation does not grow unbounded and guarantee global asymtotic convergence where the trajectory tracking error becomes zero. To provide these features, a nonlinear state feedback controller is proposed in [40].

Another focus in trajectory tracking controller is a single motor failure. The proposed method in [41] allows the quadcopter to continue to track the trajectory even a single rotor loss situation occurs. To overcome this problem, dynamic feedback control law is presented to maintain the desired velocity references along the trajectory.

[42] proposes an architecture for continuous time predictive control for limited output trajectory tracking problems by designing continuous-time sampled-data NMPC framework. This framework ensures the constraints of the states and inputs. The formulation allows to guarantee the convergence by utilizing end penalties and conditions based on terminal regions. Further, geometric nature of trajectory tracking problems are also studied by utilizing the analysis of transverse normal forms to compute the end penalties.

Recent progresses on emerging field of deep reinforcement learning allows the agent to learn to solve high dimensional complex problems. Because of the risks for damaging vehicle, the utilizing reinforcement learning techniques in vehicles are still restraint, especially in UAVs because of the possible crashes. Since many trials are required for the learning of an agent, the trainings are usually carried out in a simulation environment, and then tested by implementing the agent. Although the theoretical evidence of the reinforcement learning techniques studied is not clear, it can be seen that the agent created with Monte Carlo simulations meets the requirements. Of course, while performing these tests, all factors that may be experienced in real flights should also be found in the Monte Carlo simulation. In literature, we see that the solutions proposed for trajectory tracking problem with the backstepping approach and geometric control are dominant, but the number of solutions with reinforcement learning also increases over time.

The work in [43] used the Deep Deterministic Policy Grandient (DDPG) reinforcement learning algorithm for solving a path following problem in a UAV. The agent was

in charge of only determining the yaw angle command while the desired altitude and forward velocity are specified by the user. The agent generates delta yaw angle which is added to current yaw command, and used in attitude controller. In aggressive trajectories, only generating yaw angle commands would not be sufficient to track.

Figure 2.9 : Control performance of the RL agent [10].

[10] proposes new approach to train the deep reinforcement learning agent end-to-end. The presented method uses raw sensor data and generates motors thrust directly. When we consider classical approaches for controlling UAV, the agent includes sensor fusion, high level and low level controllers as a concept. The agent models the whole process that allows dynamic stabilization even in upside-down throw situations. The agent trained by deterministic on-policy method, where zero-bias and zero variance samples are used in training. Also for precision, they used the vehicle model that they used in real flight tests in training. One of the tests are shown in Fig. 2.9 It is shown that their approach outperformed DDPG and Trust Region Policy Optimization (TRPO) algorithms.

One of the studies focuses on designing controller for autonomous inverted flight on a helicopter. The proposed approach [44] includes using real flight data to learn helicopter's dynamics and utilizing reinforcement learning algorithm for learning a controller.

[45] points out that using PID controller for attitude could provide desired performance under normal conditions, but more intelligent controller architecture

is still needed for harsh, and unpredictable environments. Therefore, [45] proposes reinforcement learning based solution for inner controller by utilizing Trust Region Policy Optimization (TRPO), Proximal Policy Optimization (PPO), and Deep Deterministic Gradiend Policy (DDGP). It was found that PPO gave the best results among the three RL algorithms and it also outperformed the proportional-integral-derivative (PID) controller.

3. A DYNAMICALLY FEASIBLE FAST REPLANNING STRATEGY WITH DEEP REINFORCEMENT LEARNING

3.1 Problem Formulation and Contribution

3.1.1 Problem formulation

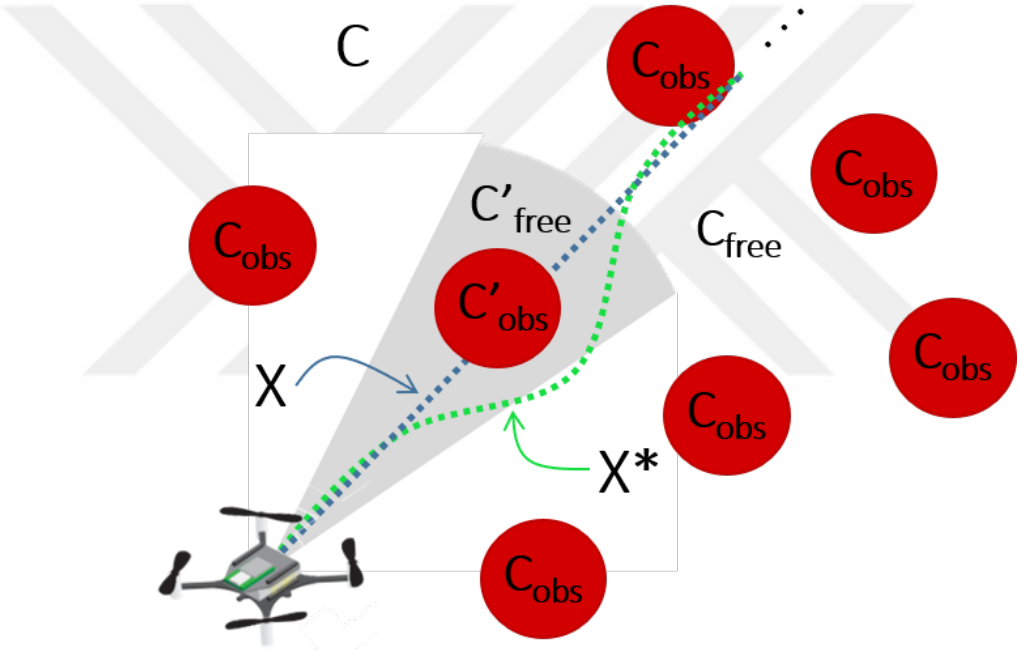


Figure 3.1 : Illustration of the agile trajectory replanning problem.

Let C be the navigation map where $C = C_{free} \cup C_{obs}$, C_{obs} is an obstacle and C_{free} is free space, which is depicted in Fig. 3.1. X is defined as pre-calculated trajectory, which is formulated as a B-spline. Before flight, C_{obs} are unknown, but only $C'_{obs} \subseteq C_{obs}$ are known because of sensor on-board. Moreover, X^* is dynamically feasible and collision-free trajectory only in sensor field of view (FOV) where $X^* \cap C'_{obs} = \{\emptyset\}$. The objective is to navigate the vehicle to use optimal trajectory X^* by using B-spline knot insertion method and changing this new control point via deep reinforcement learning. The test environment is shown in Fig. 3.2 It is important to note that the collision avoidance can only be performed locally because of the limited sensor field of view

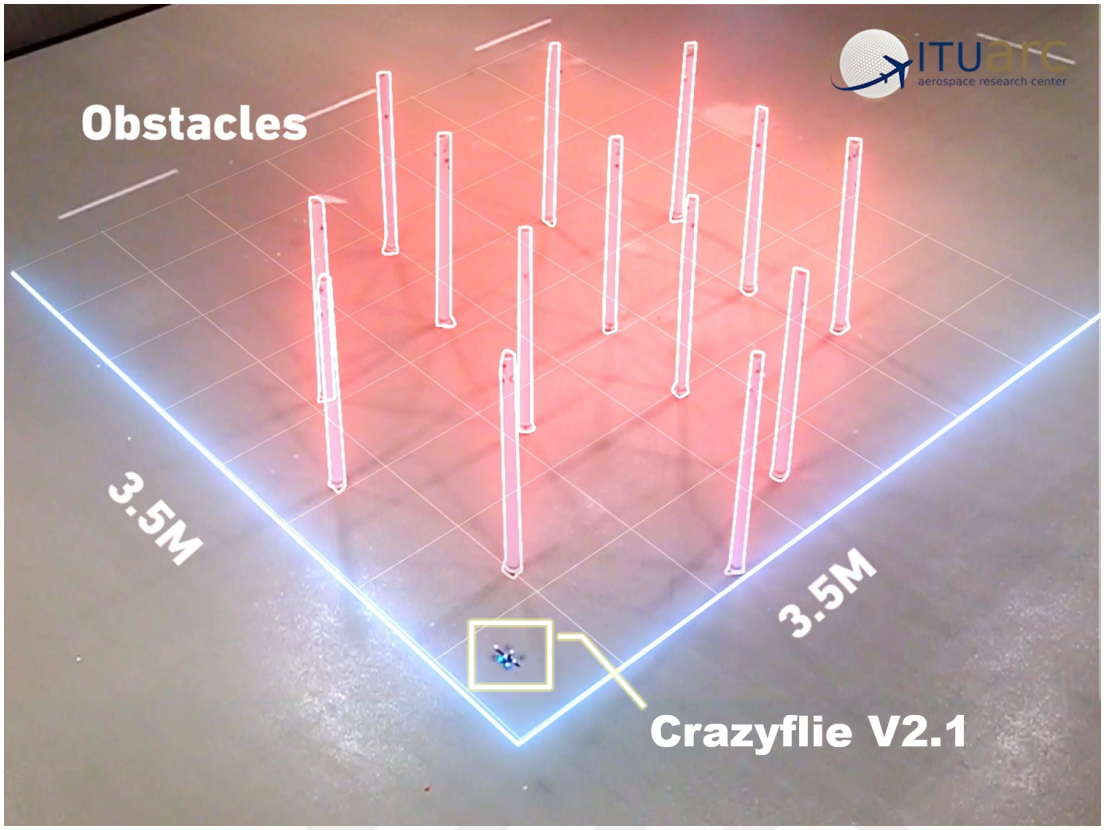


Figure 3.2 : Test environment with Crazyflie, which is used as the platform to navigate in $\rho = 1.2$ obstacle/m^2 dense environment.

and there is no prior information about the position of the obstacles. Therefore, the generated solution is not ensures global optimality, but it guarantees local optimality. Also, in aggressive maneuver envelope, proposed algorithm replans the trajectory in 1.2 ms.

3.2 Differential Flatness Based Dynamic Model

Considering the aerial vehicles, the trajectory planning problem with dynamical constraints on the vehicle might become extremely challenging due to their high dimensional dynamics. However, for most of the dynamical systems, it is generally possible to parameterize a part of the state regarding a given output trajectory and its time derivatives. This phenomenon called differentially flatness enabling an effective dimension reduction when the whole state and the input can be parameterized with one output. Let the state of the system is $x \in \mathbb{R}^n$ and let the input of the system is $u \in \mathbb{R}^m$. A nonlinear system $\dot{x} = f(x,u)$, $y = h(x)$ is differentially flat, if one can write the system

equation in the following form:

$$z = \zeta(x, u, \dot{u}, ..., u^{(p)}) \tag{3.1}$$

where,

$$x = x(z, \dot{z}, ..., z^{(q)}) u = u(z, \dot{z}, ..., z^{(q)}). \tag{3.2}$$

Through differential flatness formulation, all of the feasible trajectories for the system can be written as functions of a flat output $z \in \mathbb{R}^m$ and its derivatives [13].

3.2.1 Aerial vehicle model

In this work, the differential flatness principle is applied to real-time flight trajectory generation problem of an aerial vehicle. We have formulated the desired output trajectory through b-spline curves, enabling a map into the required input states to track the given trajectory. For the implementation purposes, we utilized dynamical model of a quadcopter, which can be derived by the Lagrangian approach as given below:

$$\ddot{x} = U_1 \frac{(\cos\psi)\cos\phi\sin\theta + \sin\psi\sin\phi}{m} \tag{3.3}$$

$$\ddot{y} = U_1 \frac{(\sin\psi)\cos\phi\sin\theta - \sin\phi\cos\psi}{m} \tag{3.4}$$

$$\ddot{z} = U_1 \frac{(\cos\theta)\cos\phi}{m} - g \tag{3.5}$$

$$\dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{1}{I_x} U_2 - \frac{J_m}{I_x} q\Omega_r \tag{3.6}$$

$$\dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{1}{I_y} U_3 + \frac{J_m}{I_y} p\Omega_r \tag{3.7}$$

$$\dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{d}{I_z} U_4 \tag{3.8}$$

$$\dot{\phi} = p + \sin\phi\tan\theta q + \cos\phi\tan\theta r \tag{3.9}$$

$$\dot{\theta} = \cos\phi q - \sin\phi r \tag{3.10}$$

$$\dot{\psi} = \frac{\sin\phi}{\cos\theta} q + \frac{\cos\phi}{\cos\theta} r \tag{3.11}$$

Where m is the aircraft mass, p, q and r are the angular rates in the body-frame, I_x, I_y and I_z are the moments of inertia. J_m is the motor inertia, and Ω_R is defined as $\Omega_R = -\Omega_1 - \Omega_3 + \Omega_2 + \Omega_4$. The input vector with U_1, U_2, U_3, U_4 is expressed in terms

of the motor angular rates Ω_i, where $i = \{1,2,3,4\}$ and given as follows:

$$U_1 = b\sum_{i=1}^{4}\Omega_i^2, \tag{3.12}$$

$$U_2 = bl(\Omega_4^2 - \Omega_2^2), \tag{3.13}$$

$$U_3 = bl(\Omega_3^2 - \Omega_1^2), \tag{3.14}$$

$$U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \tag{3.15}$$

An aerial vehicle state can be represented by the flat variables $\boldsymbol{x} = [x, y, z, \psi]$, where x, y, z are the Cartesian positions and ψ is the vehicle's yaw angle. With these four inputs, the dynamics of the quadcopter can be expressed as differentially flat. To obtain these flat outputs, we utilize the b-spline curves to formulate the Cartesian position vector $[x, y, z]$ and derivatives. Yaw will be considered as a constant for simplification. Therefore, the state of the aerial vehicle $\boldsymbol{x}$ is given as follows:

$$\boldsymbol{x} = [x^T \dot{x}^T \ddot{x}^T]^T = [x^T v^T a^T]^T \tag{3.16}$$

Through B-spline representation, which will be explained in Section-3, one can define position of the vehicle and its derivatives by using the B-spline continuously differentiable property. The total thrust U_1 can be represented by the flat outputs as follows:

$$U_1 = m\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2} \tag{3.17}$$

The pitch θ and roll ϕ angle equations is given as follows:

$$\phi = \arcsin m\ddot{x}\sin\psi - m\ddot{y}\cos\psi / U_1 \tag{3.18}$$

$$\theta = \arcsin \frac{m\ddot{x}\cos\psi - m\ddot{y}\sin\psi}{U_1\cos\phi} \tag{3.19}$$

Finally, U_2, U_3 and U_4 can be derived as in Eq. (3.17) defined through flat outputs and Lagrangian model of quadcopter.

3.2.2 Perception model

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, there is no way that air vehicle can sense the obstacles around itself [26]. The generating infinite collision-free trajectory for a flight in clutter environments with

a limited field of view, regardless of the planning algorithm, resembles the flying in an environment with a random obstacle generating process. In [46], it is shown that when this process is ergodic, the existence of an infinite collision-free trajectory exhibits a phase transition with certain critical speed. Sensing with random obstacle process is depicted in Fig.3.3.

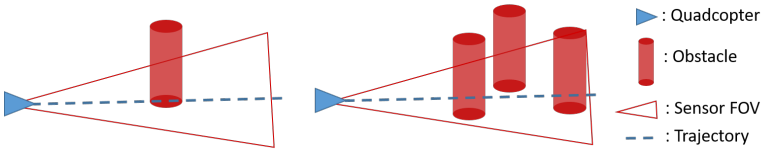


Figure 3.3 : Sensing with limited field of view (FOV) and range for the aerial vehicle. The local interest of environment is perceived as random.

3.3 Trajectory Characterization and Modification

In this section, we provide the details about output trajectory parameterization for the aerial vehicle, and how to provide modification over the generated trajectory.

3.3.1 Trajectory parameterization with B-Spline

The B-Spline representation allows one to describe any flight trajectory with their derivatives, which enables parameterizing of a flat output through the generation of several joined polynomials. Generally, a *pth*-degree B-Spline curve is defined as follows:

$$p(t) = \sum_{i=0}^{n} P_i B_{i,p}(t) a \leq t \leq b \tag{3.20}$$

Where $p(t)$ denotes the curve at 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 [47–49].

$$B_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{3.21}$$

$$B_{i,p}(t) = \frac{u - u_i}{u_{i+p} - u_i} B_{i,p-1}(t) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} B_{i+1,p-1}(t) \tag{3.22}$$

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

$$\tau = [u_0, ..., u_m] \tag{3.23}$$

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

$$\tau = [a, ..., a, t_{p+1}, ..., t_{m-p-1}, b, ..., b] \tag{3.24}$$

The length of the knot vector is $m+1$ where $m = n+p+1$ [49]. We assume that $a = 0$ and $b = 1$ for the unity knot vector, and the first and last knots have multiplicity $p+1$. We define a knot vector $\tau = [t_0, ..., t_m]$ is uniform if all interior knots are equally spaced such that $d = t_{i+1} - t_i$ for all $p \leq i \leq m-p-1$. The algorithm of the B-Spline [49] curve is represented in 3.

Algorithm 1: FindSpan

Input: n, p, u, τ
Output: mid

if $u == \tau[n+1]$ **then**
  return n
end if
$low = p$
$high = n+1$
$mid = (low + high)/2$
while $u < U[mid] \,||\, u \geq U[mid+1]$ **do**
  if $u < U[mid]$ **then**
    $high = mid$
  else
    $low = mid$
  end if
end while
return mid

Algorithm 2: BasisFunc

Input: i, u, p, U
Output: N

$N[0] = 1.0$
for $j = 0$, $j \leq p$, $j++$ **do**
  $left[j] = u - U[i+1-j]$
  $right[j] = U[i+j] - u$
  $saved = 0.0$
  for $r = 0$, $r \leq j$, $r++$ **do**
    $temp = N[r]/(right[r+1] + left[j-r])$
    $N[r] = saved + right[r+1] * temp$
    $saved = left[j-r] * temp$
  end for
  $N[j] = saved$
end for
return N

Property 1. ***Endpoint interpolation***

The trajectory $p(t)$ with control point array $[P_0, ..., P_n]$ consisting of $n+1$ control

Algorithm 3: B-Spline Curve Point

Input: n, p, τ, P, u
Output: pos: Position in time t

```
span = FindSpan(n, p, u)
N = BasisFunc(span, u, p, τ)
pos = 0
for i = 0, i ≤ p, i++ do
    pos = pos + N[i]P[span − p + i]
end for
return pos
```

points, and by assuming first and last knots have multiplicity $p+1$ where $a=0$ and $b=1$, then it holds *endpoint interpolation* property such that $P_0 = p(0)$ and $P_n = p(1)$.

The $p(t)$ curve has a strong relationship between the instantaneous positions of the vehicle as the $p(t)$ curve is the normalized form of the generated trajectory. The *endpoint interpolation* ensures that the first and last control points are the initial and final states of the aerial vehicle.

Let $p_i^{(k)}$ denote the kth-derivative of p_i. Then $p(t)$ said to be C^j continuous at the break-point t_i if $p_i^{(k)}(t_i) = p_{i+1}^{(k)}(t_i)$ for all $0 \leq k \leq j$.

Property 2. ***Continuity***

The trajectory $p(t)$ is infinitely differentiable in the interior of knot intervals, and it is at least $p-k$ times continuously differentiable at a knot multiplicity k. Then, typically a pth-degree B-Spline curve includes piecewise polynomials of degree p and have C^{p-1} continuity. The derivatives of $p(t)$ curve enables to define the velocity, acceleration, jerk and snap of the trajectory, and can be expressed as follows:

$$p^{(k)}(t) = \sum_{i=0}^{n} P_i B_{i,p}^{(k)}(t) t \in [0,1] \tag{3.25}$$

where k represents the order of derivatives. B-spline curve is a linear combination of the basis function $B_{i,p}(t)$,therefore the differentiability and continuity of the B-spline depends on their basis functions $B_{i,p}(t)$. The Eq. (4.12), allow us to obtain velocity and acceleration vectors of the trajectory through first and second derivatives respectively, and an example trajectory with velocity and acceleration vectors is given in Fig. 3.4. Considering the agility in generated flight trajectories, in addition to velocity and acceleration continuity, we have included jerk and snap continuity as well while

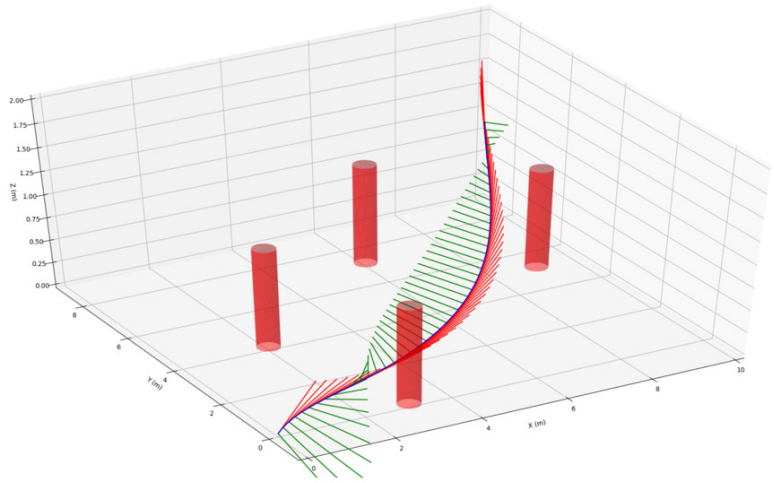


Figure 3.4 : An example B-spline trajectory agility with instantaneous velocities(red) and accelerations (green), where the velocity vectors shows the motion direction of the aerial vehicle.

defining flat outputs to track; therefore, we have chosen to represent flat output trajectories with $p = 6$-degree B-splines to ensure at least C^4 continuity.

The B-spline curves have a strong *convex hull property* that the curve is constrained in the convex hull of its control polygon.

Property 3. ***Strong convex hull***

The curve is contained in the convex hull of its control polygon. If $t \in [t_i, t_{i+1}]$ where $p \leq i \leq m-p-1$, then $p(t)$ is in the convex hull of the control points $P_{i-p}, \ldots, P_i$. In other words, the generated trajectory remains within certain limits, which are formed by control points P_i and as a result of this known beforehand. Knot insertion or breaking uniformity, therefore, does not change these limits of this convex form. This property is depicted in Fig. 3.5 with an example.

The other property that we exploit is *local support*.

Property 4. ***Local support***

Relocating P_i changes $p(t)$ trajectory only interval of $[t_i, t_{i+p+1})$ due to the fact that $B_{i,p} = 0$ for $t \notin [t_i, t_{i+p+1})$. In other words, relocating one of control points changes the curve's position and derivatives only locally [49]. This property allows us to relocate the control points, without repetitively checking the collision and dynamical feasibility of the whole trajectory with its positions and derivatives.

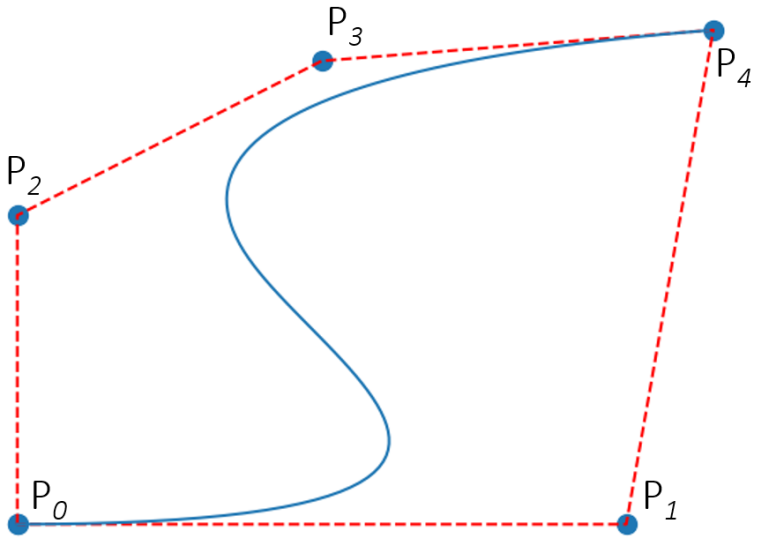


Figure 3.5 : An example of B-spline curve defined by five control points $P_{0,..,4}$ which are shown as blue dots. The curve is completely enclosed within the convex hull created by its control points.

One can note that there is a strong relationship between the knots t and the time allocations over the generated trajectory as $t_i \in [0,1] \quad \forall i \in \mathbb{R}$ is the normalized time parameter. In the other words, considering the knot vector $\eta\tau = \eta[t_1,...,t_k]$, the η factor does not change the geometry of generated trajectory while scaling the derivatives of it.

3.3.2 Trajectory modification with control point relocation and knot insertion

The trajectory replanning strategy is based on control point relocation and knot insertion to the generated B-spline curve. Through exploiting the following properties of the B-splines, we can achieve this replanning with knowing the limits and the local interest of the modification over the trajectory.

After defining the trajectory with a uniform B-spline, through control point and knot insertion, one can replan the trajectory only changing the geometry of local interest for certain intents such as dynamic collision avoidance, re-optimization, etc. Let us assume that the collision over the trajectory around $\bar{t}$ sensed, meaning that immediate replanning is required, which is depicted in Fig. 3.6.

Control Point Relocation

To express the relocation of the control point, Eq. (4.7) can be written in the following form :

$$p(t) = P_0 B_{0,p}(t) + ... + P_i B_{i,p}(t) + ... + P_n B_{n,p}(t) \tag{3.26}$$

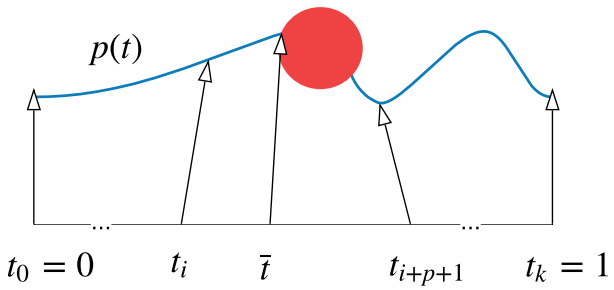


Figure 3.6 : Collision sensing over trajectory around $\bar{t}$ knot point.

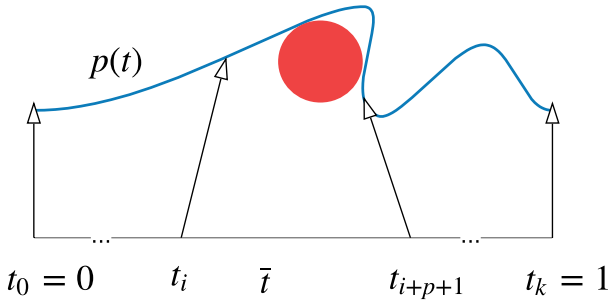


Figure 3.7 : Control point relocation over the trajectory.

Where $0 < i \leq n$. Let us assume to relocate the P_i, then the new control point and location become $\bar{P}_i$:

$$\bar{P}_i = P_i + V \tag{3.27}$$

Where V presents the relocation vector. After relocation, the B-spline curve equation becomes:

$$\bar{p}(t) = P_0 B_{0,p}(t) + \ldots + (P_i + V) B_{i,p}(t) + \ldots + P_n B_{n,p}(t) \tag{3.28}$$

or

$$\bar{p}(t) = p(t) + V B_{i,p}(t) \tag{3.29}$$

The relocation process, due to the *local support*, only effects the curve within the $t \in [t_i, t_{i+p+1})$ interval where $B_{i,p}(t) \neq 0$ and this is depicted in Fig. 3.7.

Knot Insertion

Relocating existing control points gives us a limited behavior for replanning and in some cases, might cause additional agility, almost breaking dynamic feasibility. By addressing this problem, the knot insertion methodology shown in Fig. 3.8, enables an increasing number of control points; in other words, more flexibility, without changing the geometry of the trajectory.

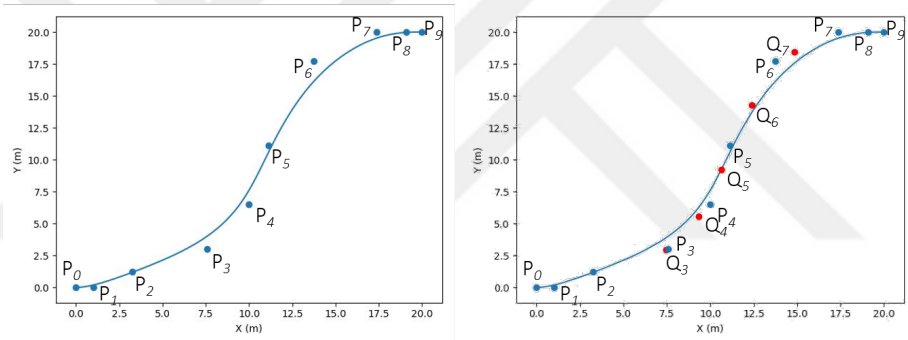
Let us assume that $p(\bar{t})$ is the closest point to the sensed obstacle or mean of the sensed obstacles p_{so} on the trajectory. $\bar{t}$ is starting from the current time t_0 when the aerial vehicle sense any object with its sensor. Then, it uses the knot insertion method to add this new P_{new} control point.

$$P_{new} = p(\bar{t})\bar{t} = min(\|p(\bar{t}) - p_{so}\|_2)\bar{t} \in [t_0, 1]$$

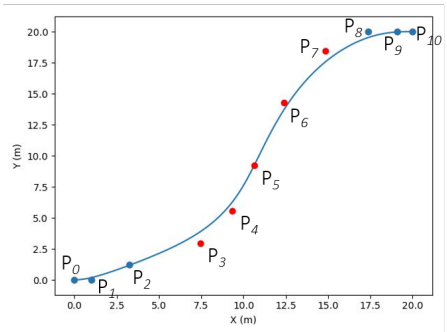
As we already know the location of P_{new} from (3.30), we need to insert $\bar{t}$ inside the knot interval from the knot vector. We assign $l = t_i$, where $\bar{t} \in [t_i, t_{i+1}]$. Then, we utilize following Eq. (3.30) and Eq. (3.31):

$$\alpha_i = \begin{cases} 1 & i \leq l-p+1 \\ 0 & i \geq l+1 \\ \frac{\bar{t}-t_i}{t_{l+p+1}-t_i} & l-p+2 \leq i \leq l \end{cases} \tag{3.30}$$

$$\hat{P}_i = (1-\alpha_i)P_{i-1} + \alpha_i P_i \tag{3.31}$$



(a) Initial Trajectory (b) Knot insertion method



(c) New control points replaced

Figure 3.8 : Replacing control points with the new ones through knot insertion method is presented. (a) is a simple trajectory defined by $P_0, ..., P_9$ control points. (b) shows knot insertion method and the output of the method is shown as $Q_3, ..., Q_7$ control points. (c) presents the result of the knot insertion method where $P_3, ..., P_6$ are replaced with $Q_3, ..., Q_7$ control points.

Through the knot insertion, we find the knots correspond to the required control point positions. We replace control points $P_{l-k+1}, ..., P_{l-1}$ with $Q_{l-k+1}, ..., Q_l$; hence, the

new control point vector is defined as $\hat{P}_i$, and the new B-spline curve can be found with following Eq. (3.32).

$$p(t) = \sum_{i=0}^{n+1} \hat{P}_i \hat{B}_{i,p}(t) t = [0,1] \tag{3.32}$$

An example control point vector modification through knot insertion is depicted in Fig. 3.8. It should be noted that an increasing number of control points does not cause geometric change over the trajectory.

3.3.3 Replan decision point

In trajectory planning problems, mostly global and local planning/replanning runs together to achieve safe navigation in a highly dynamic environment. In addition to geometric and dynamic feasibility, local planners require computational feasibility that considers computational and time complexity of the utilized algorithms. To achieve real-time fast replanning, the algorithm should consider calculation time as the vehicle flies over the trajectory. Through the *local support* property as we know already, which segment of the trajectory to be reformed, it is essential to consider the backpropagation of this modification over the planned trajectory.

Now suppose that $\bar{t}$ is the knot vector value with $\bar{t} \in [t_k, t_{k+p+1})$ and $p(\bar{t}) = P_i$ is the control point where a modification requires and the obstacle detection occurred at $p(t_0)$. Recall that knot values over the trajectory has a direct scaling relationship between the time as the knot vector $t \in [0,1]$ is the normalized time value. It is possible to define an algorithm run-time and transform into a knot value, let it be δ_t. Due to the translation of P_i to $\bar{P}_i$, all trajectory points outside the $\bar{t} \in [t_k, t_{k+p+1})$ interval are to be unaffected. Therefore following safety rule should be satisfied always while replanning;

$$t_k \geq t_0 + \delta_t, \quad \bar{t} \in [t_k, t_{k+p+1}) \tag{3.33}$$

This rule guarantees that any possible replanning trajectory has a chance to provide an evasive maneuver. As the replanning $[t_0, t_{p+1})$ span over the trajectory depends on the size of the knot vector, the scope of the replanning segment shows a difference with the number of control points. There is a trade-off here between the trajectory description resolution and the number of fails in replanning algorithm run: the more control points providing more flexibility in trajectory modification means possibly more fails in dynamic feasibility check; while fewer control points give smoother

trajectory modifications, unlikely breaking the dynamic feasibility over the trajectory. Fig.3.9 demonstrates these effects with different trajectory description resolutions.

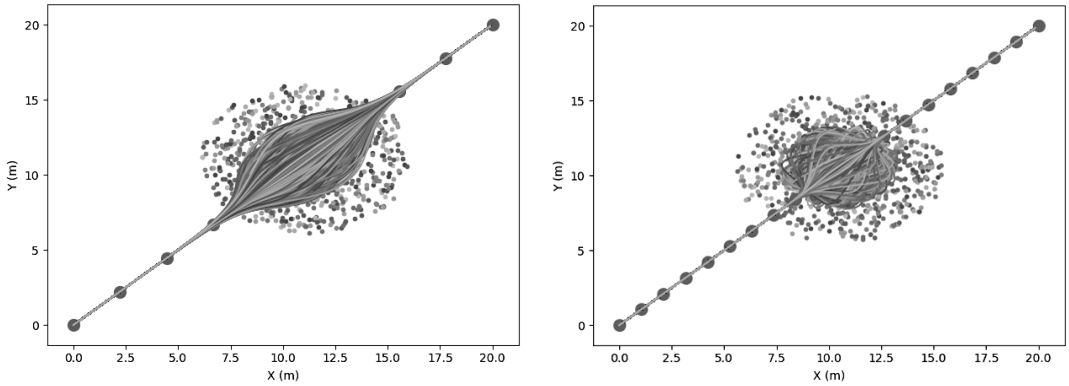


Figure 3.9 : Replanning scopes over the trajectories with different number of control point representations.

3.4 Optimal Replanning with Deep Reinforcement Learning

Since the air vehicle tracks the trajectory at high speeds, the collision avoidance algorithm should be able to respond immediately. An algorithm that does not work with sufficient performance causes the high-speed aircraft to get into the inevitable collision state. In addition, global optimality is not expected from the new generated trajectory, since the environment is partially known by the air vehicle. However, it is desired to produce the local optimum escape maneuver.

As a solution to these requirements which is shown in Alg. 4, first, a new control point is added to the closest point to the obstacle by using one of the features of B-spline called knot insertion method. This method provides more flexibility without breaking dynamic feasibility or changing geometry and progression of the trajectory. Then, the position of this control point is shifted according to the output of the pre-trained deep reinforcement learning agent. The output of the algorithm is a new route that avoids the obstacle detected by local knowledge with the re-positioned control point. Deep Reinforcement Learning (DRL) is a tool that can provide an optimal decision based on what it has learned in certain situations. The purpose of this tool, which uses Markov decision processes (MDP), is to produce an action based on the state of the agent in a particular environment. After taking any action, the agent receives a reward for the action and the next state. Through estimating/learning the rewards corresponding to the state, the agent develops its policy.

Algorithm 4: Aggressive Collision Avoidance Algorithm

Input: C'_{obs}: Sensed obstacle position
Output: $p^*(t)$: Locally optimal, collision-free trajectory

if $C'_{obs} \neq \{\emptyset\}$ **then**
    $\hat{P}$, $\hat{\tau}$ = KnotInsertionMethod(P, τ, C'_{obs}, k)
    Δx, Δy = RLAgent(ObservationMatrix)
    $\hat{P}_k = \hat{P}_k + [\Delta x, \Delta y]$
end if
return $\hat{P}$, $\hat{\tau}$

In reinforcement learning, the maximization of the accumulated reward is the main goal of the agent. Therefore, In each episode, the agent learns to self-adjust policies to maximize the accumulated reward. The agent learns these policies through interaction with the unknown environment, which is formulated as an MDP by a tuple $M = (S, A, T, R, \gamma)$. S is a set of agent states, and A is a set of actions. This representation assumes that the next state s_{t+1} is only conditional on the current state s_t and action a_t which is derived from Markov property. T is a transition probability function $T : SxA \in [0, 1]$, which maps the transition to probability.$R : SxA \in R$ is the reward function represents the amount of reward or punishment that the environment will pass in for a state transition. γ is the discount factor $\gamma \in (0, 1]$, which is used to receive the return from the process as a sum of the discount rewards. Further thought, the agent is at state s_t, and it takes action a_t and receives a reward r_t based on predefined reward function. Then, the environment transitions to state s_{t+1} according to the T.

Reinforcement learning applies this MDP formulation and modifies it to use for learning. The technique that we use is defined under policy-based reinforcement learning. This policy represents a function mapping a state to action, and the agent aims to optimize the policy to maximize the accumulated reward.

3.4.1 Proximal policy optimization

The Proximal Policy Optimization (PPO) is one of the policy gradient methods for reinforcement learning. It uses the first-order algorithm to utilize the benefits of trust region policy optimization (TRPO) such as reliable performance and data efficiency [50]. The TRPO aims to maximize the surrogate objective function L^{CPI} (conservative policy iteration) [51]. Let $r_t(\theta)$ is the probability ratio shown as follows:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} r(\theta_{old}) = 1 \tag{3.34}$$

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right] \tag{3.35}$$

CPI refers to conservative policy iteration. [50] proposes new objective function, which is presented in below:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta)), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right] \tag{3.36}$$

where epsilon ε is a hyperparameter that we have chosen as 0.2. The difference between TRPO and PPO based on the $clip(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t$ term, which allows us to adjust the surrogate objective by saturating the probability ratio. The reason for this clip function is to avoid extensive updates by the maximizing L^{CLIP}. Clipping function also depends on the interval $[1-\varepsilon, 1+\varepsilon]$ term, where we can change the size of the update rate by changing the ε.

For the training part of the DRL, first, we need to define observation, reward, and action vectors. In order to start the learning process, random scenarios where the agent to be trained are generated. In Fig.3.10, some possible scenarios are depicted. In each scenario, Poisson distributed random numbers are generated to create the number of obstacles encountered by air vehicles through the distribution equation given by Eq. (3.37), and the position of the obstacles is randomly assigned according to a uniform distribution.

$$Pois(x;\mu) = (e^{-\mu})(\mu^x)/x! \tag{3.37}$$

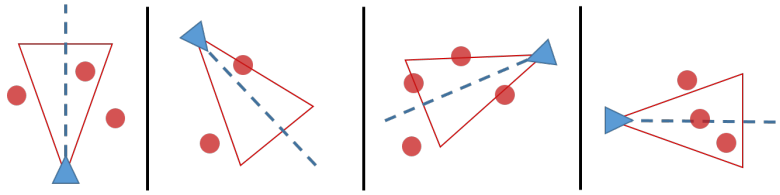


Figure 3.10 : Depiction of scenarios with random obstacle generation and limited FOV.

Considering the small region of interest due to limited FOV, we only produce one collision situation in each scenario during the training. Instead of running the training algorithm depending on the cumulative rewards where there are too many obstacles, we have created specific scenarios that only locates one or more obstacles once, and tries to generate solution locally. This modification simplifies the collision avoidance in the cluttered environment and allows the DRL agent to learn faster.

In DRL, we defined observation, reward and action as follows:

Observation Space:

We assume that the air vehicle can obtain the state measurements via inertial measurement unit (IMU) sensor and GPS. The trajectory plan, defined through B-spline curve, involves initial and future states and their derivatives. For describing a collision state, the observation state also includes the states of the obstacles that are sensed, and the new inserted control point location. Observation space tuple is given as follows:

$$\Omega = \{p(t), \dot{p(t)}, \ddot{p(t)}, p_{obs}, P_{new}\} \tag{3.38}$$

where $\{p(t), \dot{p(t)}, \ddot{p(t)}\}$ are the current state and its derivatives of the aerial vehicle over the generated trajectory; p_{obs} is the closest point to collision with the obstacles; and P_{new} is the newly inserted control point.

Action Space:

Action space consists of the relocation position with respect to the newly inserted control point. After the DRL agent generates a new action based on the observation space, this action values are added to this new control point, and the B-spline curve is updated. Action space tuple is given with relocation of newly inserted control point $\mathscr{A} = \{P_{new}\}$.

Reward Function:

In this local re-planning problem, DRL agent's learning is directly depends on how to define the reward function.

Considering agile flight enabling fast collision avoidance, we have chosen to use the agility metric for the reward function. Given the definition, agility can be a positive reward or penalty to obtain aggressive or smooth flight under the dynamical constraints enabling a feasible trajectory. These constraints limit the search space for the training of the agent for re-planning.

The definition of the agility metric is a well-studied topic on many different vehicles. In [52, 53] many of the agility metrics are summarised. [53] especially focuses on maneuverability and agility for rotary unmanned aerial vehicles, and defines the attitude quickness as $Q = \dot{\alpha_{peak}}/\Delta\alpha$ where $\dot{\alpha_{peak}} = \{p_{peak}, q_{peak}, r_{peak}\}$ and $\Delta\alpha = \{\theta, \phi, \psi\}$. Considering these definitions, we have chosen the agility metric based-on

Instantaneous-rates, which are also proposed in [54–56]. Hence, the reward function R can be defined with the agility metric as a second derivative of the generalized state variables, as given below:

$$R = \mp \sum_{t=t_i}^{t_{i+p+1}} \sum_{j=1}^{m} \omega_j \ddot{\mathbf{x}}_{\mathbf{j_t}} \tag{3.39}$$

where $\mathbf{x} \in \mathbb{R}^m$ represents the generalized state variables and ω_j is the weight value for each state. By setting these weight values, we have chosen to use the agility definition with high dominance at angular velocities.

Dynamic Feasibility:

To make sure the generated trajectory is feasible, we must ensure that velocity, acceleration, jerk, and Euler angles at each state of the trajectory remain bounded, and guarantee that the newly generated trajectory remains within the collision-free space with small safety volume, as depicted in Fig.3.11. Note that relocating one of control point P_i changes $p(t)$ trajectory only interval of $[t_i, t_{i+p+1})$, and requires collision check over this segment of the trajectory only. To meet the dynamical feasibility, we included constraints inside the reward function. If at any point, the generated trajectory respect to the action breaks the following constraints, then the reward function takes a big negative constant value.

$$\|p(t) - p_{so}\|_2 \leq d_{obs,r} + d_{safe} \tag{3.40}$$

$$v_{min} \leq \dot{p}(t) \leq v_{max} \quad t \in [t_i, t_{i+p+1}) \tag{3.41}$$

$$a_{min} \leq \ddot{p}(t) \leq a_{max} \quad t \in [t_i, t_{i+p+1}) \tag{3.42}$$

$$j_{min} \leq \dddot{p}(t) \leq j_{max} \quad t \in [t_i, t_{i+p+1}) \tag{3.43}$$

$$\theta_{min} \leq \theta(t) \leq \theta_{max} \quad t \in [t_i, t_{i+p+1}) \tag{3.44}$$

$$\phi_{min} \leq \phi(t) \leq \phi_{max} \quad t \in [t_i, t_{i+p+1}) \tag{3.45}$$

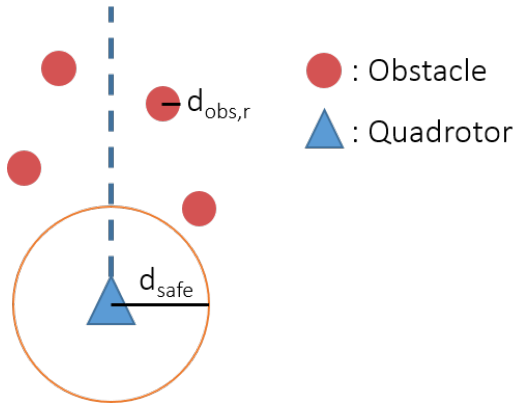


Figure 3.11 : Safety volume around the vehicle.

3.5 Fast Replanning Software Implementation Results

To generate a DRL agent, we used OpenAI gym environment [57] with proximal policy optimization. After defining the required spaces, we trained the DRL agent approximately 1200 episodes. Because of the number of CPU and scenario number for each episodes, the DRL agent trained with approximately 30 million scenarios. The reward results can be seen in Fig.3.12. The initial point G_{init} and the goal point G_{goal}

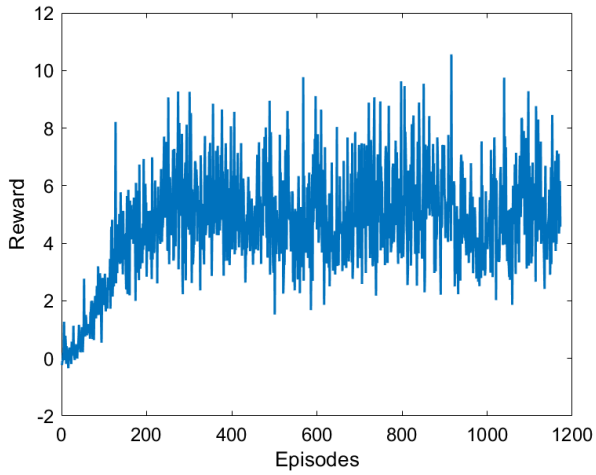


Figure 3.12 : Reward performance, each episode includes 5120 randomly generated scenarios.

are randomly generated with a specific distance between them shown in Fig.3.13. From very start, because there are not any known-obstacles, the minimum distance trajectory between G_{init} and G_{goal} is a simple straight line. Because of randomly generation of these points, the lines heading also changes. This is very important for the scenario generation allows the generalization of the DRL agent. Another important point to mention is encountering obstacles situations.

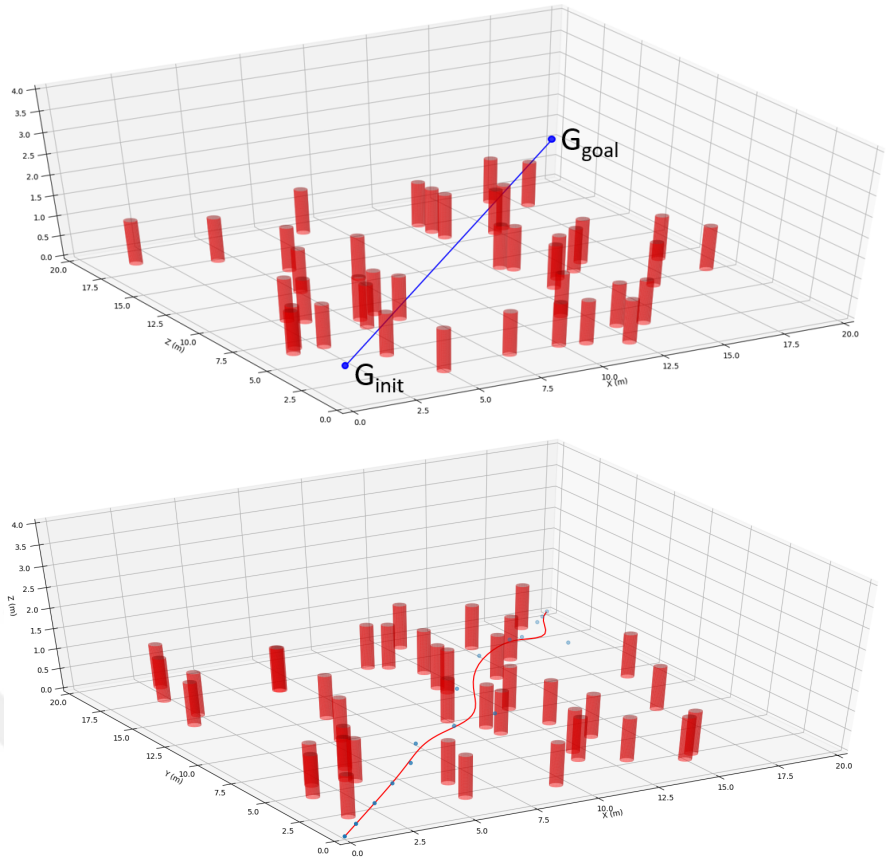


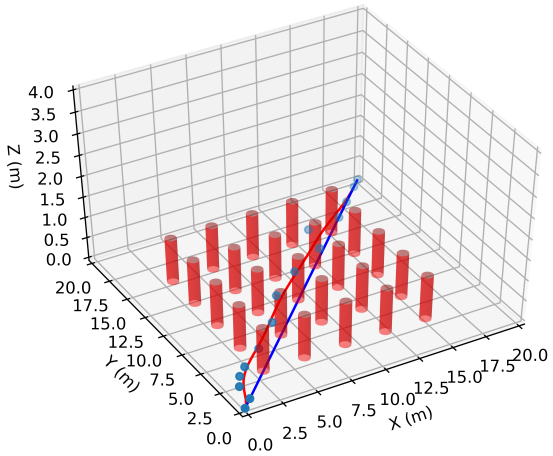
Figure 3.13 : Replanning algorithm in $20m \times 20m$ environment with an obstacle density of 0.1 $obstacles/m^2$.

The on-board sensor model [26] is used to detect obstacles and find their relative positions respect to the air vehicle's location. We used a sensor model on the air vehicle with a 20° field of view (FOV) and 4 meters range. Whenever sensor finds any obstacle, the local re-planning algorithm starts, which consists of adding new control point with knot insertion algorithm, generating observation for DRL agent, action generated by DRL agent and with new location for the new control point is used to update the trajectory. The schematics for the scenario is shown in Fig.3.13.

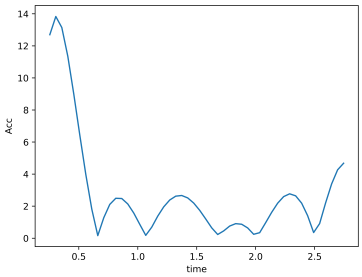
In the cluttered environment simulations, the scenarios are designed through randomly cluttered forests with different obstacle density where the vehicle must fly from G_{init} to G_{goal} in $20m \times 20m$ environment. Before the flight, the air vehicle generates a B-spline, which is an almost straight flight trajectory between G_{init} and G_{goal}, which can be seen in Fig.3.13.

We have conducted several scenarios at different complexities for testing our DRL agent. The results from three randomly generated scenarios can be seen in Fig.3.14,

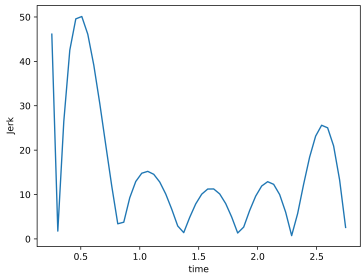
3.15 and 3.16 such that the obstacle densities are higher than 0.05 $obstacles/m^2$ and 0.1 $obstacles/m^2$.



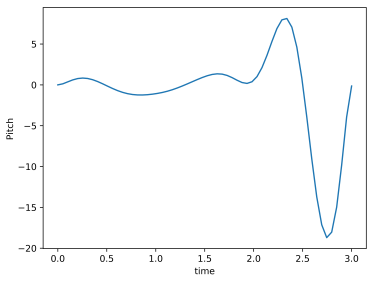
(a) Scenario



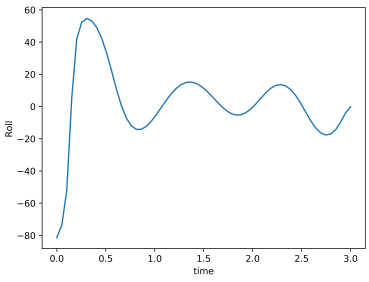
(b) Acceleration



(c) Jerk



(d) Pitch

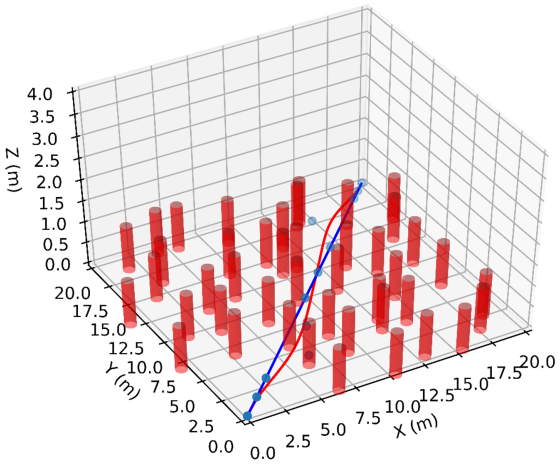


(e) Roll

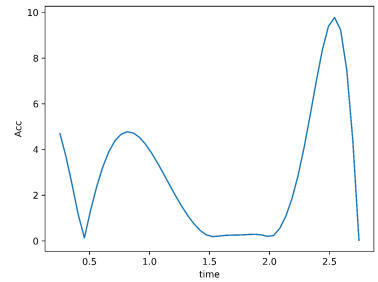
Figure 3.14 : The cluttered environment simulation, randomly generated obstacles in $20m \times 20m$ environment with an obstacle density of $> 0.05 obstacles/m^2$.

3.5.1 Batch simulation results

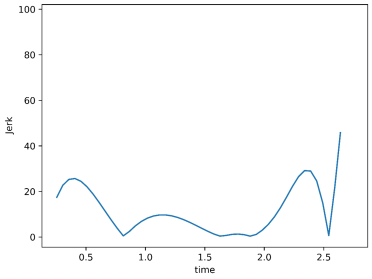
Considering the dynamic feasibility of replanning, through the proposed methodology, we have full control authority over the defined dynamical constraints. The



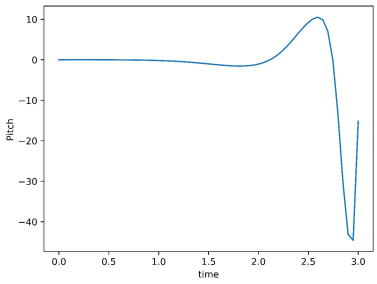
(a) Scenario



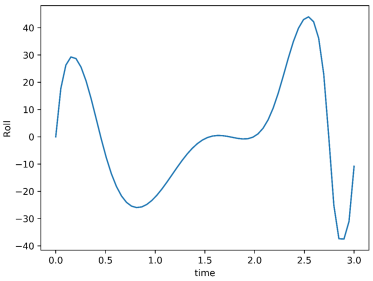
(b) Acceleration



(c) Jerk



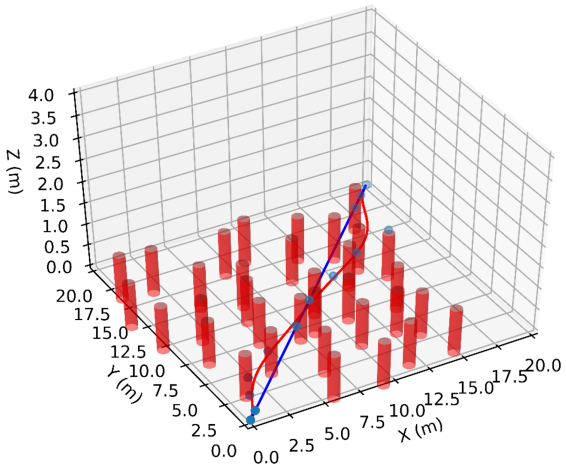
(d) Pitch



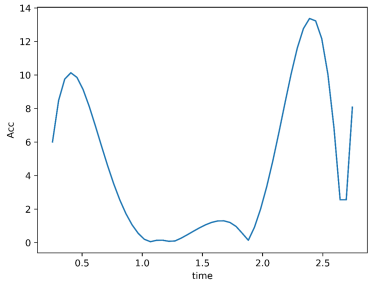
(e) Roll

Figure 3.15 : The cluttered environment simulation, randomly generated obstacles in $20m \times 20m$ environment with an obstacle density of $> 0.1 obstacles/m^2$.

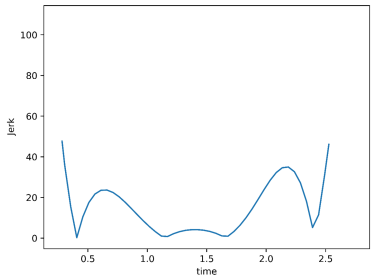
methodology allows us to define the constraints and generate the trajectories to comply with them. To test the ability of the algorithm, e.g., in the training phase of DRL agent, we have defined upper bound constraint at $11m/s$ and lower bound constraint at $8.5m/s$ in velocity; upper bound at $15m/s^2$ in acceleration, and upper bound $50m/s^3$ in jerk. This set of constraints has been chosen for a small UAV to track a trajectory at high speeds based on the strict dynamic limits. Through a batch run with 500 randomly



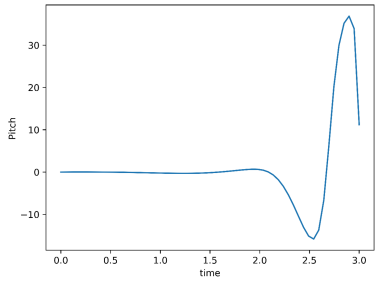
(a) Scenario



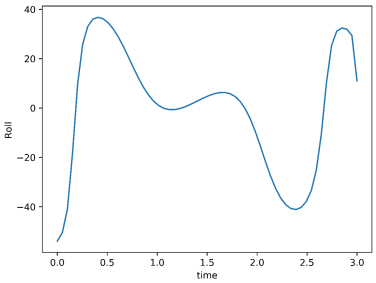
(b) Acceleration



(c) Jerk



(d) Pitch



(e) Roll

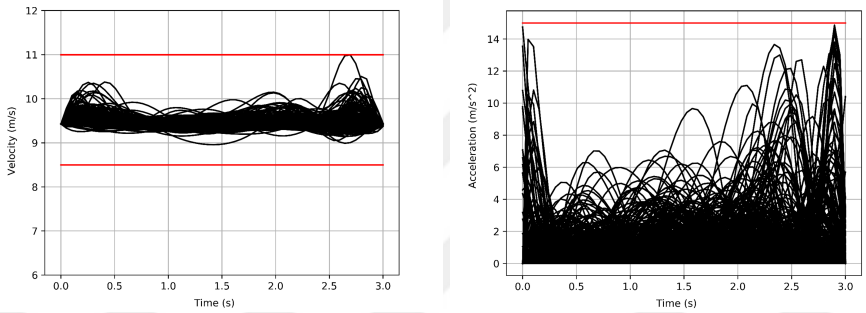
Figure 3.16 : The cluttered environment simulation, randomly generated obstacles in $20m \times 20m$ environment with an obstacle density of $> 0.1 obstacles/m^2$.

generated scenarios, the velocity, acceleration, and jerk profiles generated by the algorithm, are shown in Fig. 15. As seen in the figure, the algorithm does not break the constraints, even though hits the limits in some cases. In addition to these results, there were only three scenarios terminated by pre-defined safety rules, demonstrating that the RL agent can not generate a dynamically feasible collision-free trajectory because of the deficiency of enough time to maneuver. Furthermore, another 500 randomly

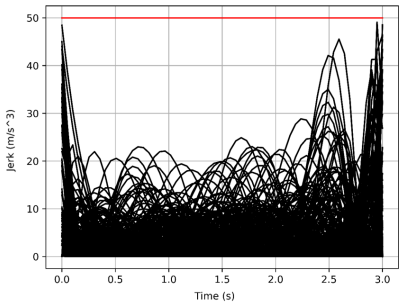
Table 3.1 : Comparisons with other similar trajectory replanning methodologies.

Liu et al. [3]	$160ms$	3.4 GHz dual-core i7 Intel NUC
Burri et al. [14]	$> 40ms$	AscTec Firefly 2.4 GHz Controller
Chen et al. [17]	$> 34ms$	3.20 GHz Intel Core i5-4570 CPU
Lopez et al. [8]	$\leq 5.06ms$	2.70GHz Intel Core i7-2620M
CL-RRT* [58]	$\geq 650ms$	Intel Xeon 2.4GHz
Our method	$1.2ms$	Intel Xeon 2.4GHz

generated scenarios are generated, with increased control point numbers to define a B-spline trajectory. In this case, five scenarios were terminated due to no solution found under dynamical limitations. From a practical point of view, It is possible to foresee these situations and hold the action to avoid possible crash situations.



(a) Velocity (b) Acceleration

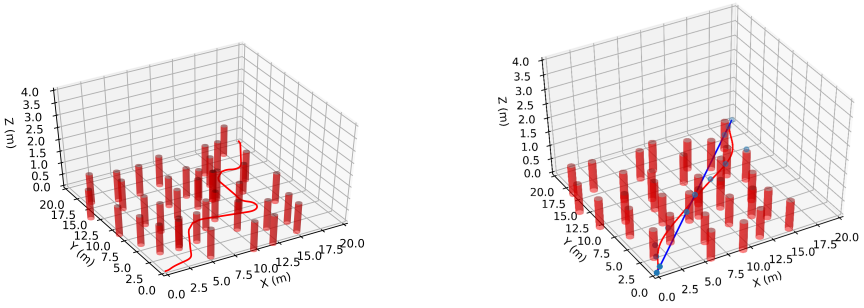


(c) Jerk

Figure 3.17 : 500 different flight scenarios where all the obstacles are positioned randomly on the map are tested. The result of the flight tests are shown for the following metrics (a) velocity m/s, (b) acceleration m/s^2,and (c) jerk m/s^3.

3.5.2 Performance comparison with other algorithms

Considering the real-time applicability, we have measured the computation times, which are conducted on Intel Xeon 2.4 GHz. For the worst case scenario, we have observed that it takes 0.512ms to produce a new location for the control point of the



(a) CL-RRT* solution at 650ms (b) Our methodology at 1.2 ms

Figure 3.18 : Comparison with Closed-loop RRT* solution for trajectory replanning.

DRL agent. And with the knot insertion method, generating and updating the trajectory takes $1.2ms$. The computation time table with other reference methodologies with their computational power is given below, and it is worth noting that $1.2ms$ computation time starts from after obstacle detection to the end of the trajectory updating. The reported computation times in the table might include the point cloud processing to detect the obstacles in the environment, which typically has utilization around $\%5-10$.

Then we have concentrated on the RRT* algorithm, as it is a well-known and well-studied sampling-based trajectory planning methodology assuring asymptotic optimality [58]. Considering the dynamical feasibility, we applied the closed-loop modification of RRT* (CL-RRT*) algorithm to compare with the presented method.

It should be noted that, in the agile collision avoidance problem, the goal is to generate a solution as fast as possible; therefore, we have chosen to use the first feasible trajectory of CL-RRT*. In sampling-based algorithms, while the number of samples goes to infinity, the algorithm's solution converges to the optimal solution [59]. Hence, they need to use large samples to generate nearly optimal solutions.

When we compare our solution with the CL-RRT* algorithm, while our algorithm provides a re-planned trajectory in 1.2 ms, the CL-RRT* algorithm generates a feasible trajectory after 650 ms. This result is expected, as the CL-RRT* has no ability to use existing (but compromised) flight trajectory, while our methodology exploits it, in addition to offline training. As seen in Fig. 3.18, the first product of the closed-loop RRT* algorithm, which presents after $50-75$ samplings on average, is far from the optimal trajectory.

3.6 Fast Replanning Hardware Implementation Results: Under VICON System

We have applied the methodology integrating with agile trajectory tracking controller to a small UAV hardware platform flying under motion capture system. A video including explanation of the algorithm and a couple of random scenarios can bee seen in the following link: https://youtu.be/8IiLQFQ3V0E.

The air vehicle used in this work is Crazyflie 2.1 [60]. Fig. 3.19 shows the architecture used in the implementation. The fastest route from start point to end point with no obstacles assumption is a straight line. Therefore, first we generate straight differentially flat trajectory from starting point to end point. The initial desired B-spine trajectory reference is sent to the trajectory tracking controller by the navigation system. This system run inside one of the desktop computer with Intel i7 CPU. Also inside, sensor emulator which is explained in Section 3.2.2 is used as a on-board sensor which has 1 m range and 20 deg FOV. Sensor emulator also obtain position data of the Crazyflie and the obstacles from VICON in $100Hz$. After any situation appears where any obstacles inside the FOV, sensor emulator sends obstacles position to the algorithm. Knot insertion method is used to add new control point on the sensed obstacle and DRL agent generates delta distances for this new control point. After this, trajectory is generated in that knot interval. Even sensor emulator finds any obstacle or not, trajectory references are independently sent to the trajectory tracking controller, where this controller also get position and orientation data from the VICON. Trajectory tracking controller generates desired roll, pitch and yaw references and these values are sent to the Crazyflie by RF radio. These processes are running inside the desktop computer.

Crazyflie obtains these reference values and also its own angular states from sensor fusion which only uses gyro and accelerometer data. The desired PWM values are generated inside Crazyflie by using attitude controller, attitude rate controller and control mixer.

Trajectory Tracking Controller:

Aggressive trajectories is really hard to track with classic controller methods. With using LQR, because of the steady state error, the tracking performance was very poor. To the state-space representation to include the integration of the position states in

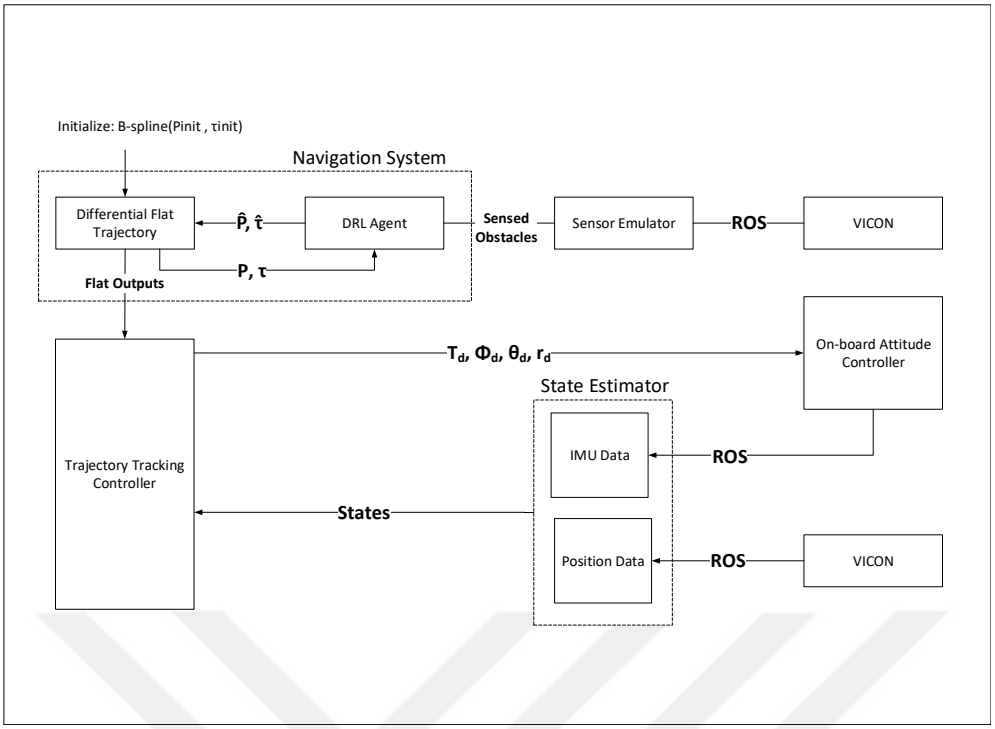


Figure 3.19 : The system architecture for hardware implementations.

order to minimize the tracking error in agile maneuvers, as follows:

$$X = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \psi & \int x & \int y & \int z \end{bmatrix}^T \tag{3.46}$$

And the state-space model becomes:

$$\dot{x} = Ax + Bu - bg \tag{3.47}$$

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

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

The Fig.3.20 shows the controller architecture. The error in the quad-rotor position is integrated in the LQI block and combined with the position and the velocity errors. This error is then driven to zero by finding the control input $u = -K * error$ that

minimizes the quadratic cost function,

$$J = \int_0^\infty X^T QX + u^T Ru \, dt \tag{3.48}$$

where Q and R are the symmetric positive-definite weighting matrices.

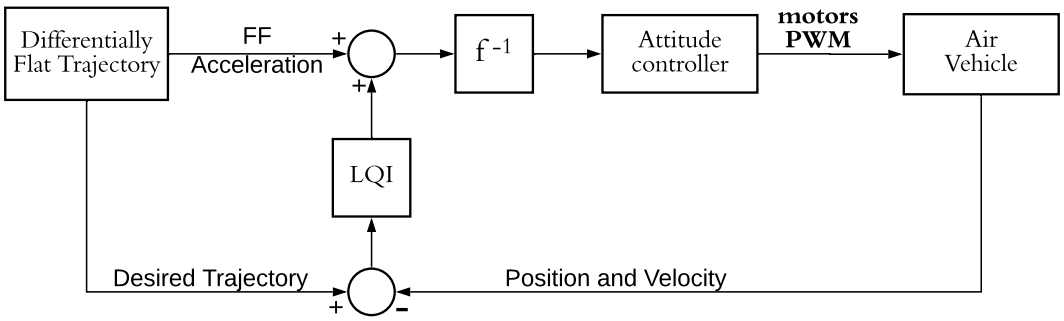


Figure 3.20 : Differentially flat trajectory tracking architecture.

Differential Flat Trajectory is corresponds to the B-spline trajectory generation. Before the flight, straight trajectory is defined which starts from $[0,0]$ point to $[3.5,3.5]$ point for testing. Also, the defined trajectory is ensured to be aggressive that even it is straight line, the forward acceleration is at nearly its limits. The output of this block is the desired position, velocity and acceleration values.

The reason not to be exactly at its limits is that to leave enough control space to avoid the obstacles. Another solution might be changing the know vector coefficient to earn enough time to avoid. This topic will be discussed at the last section.

LQI and inverse function is used to generate the inputs for attitude controller. First, the error of position and velocity values are found and used in LQI. Acceleration values are feed forwarded to the output of the LQI block. The output of adding block are corresponds to the desired acceleration values. Attitude controller inputs are $\upsilon = [T_d, \phi_d, \theta_d, \psi_d]$. To achieve these values, we need to use this inverse function for getting attitude controller inputs. The input of the inverse function is $u = [u_x, u_y, u_z]$, where these are in inertial frame. Constant yaw angle command is used for this case.

$$\begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} m = R^T(\psi)R^T(\theta)R^T(\phi) \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{3.49}$$

$$T_d = m\sqrt{u_x^2 + u_y^2 + u_z^2} \tag{3.50}$$

By solving for ϕ and θ [61], we get,

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = R(\psi) \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \frac{m}{T} \tag{3.51}$$

$$\phi_d = \arcsin(-z_2) \tag{3.52}$$

$$\theta_d = \arctan(\frac{z_1}{z_3}) \tag{3.53}$$

Attitude Controller is already implemented on-board Crazyflie 2.1 [60], which is shown in Fig. 3.23. A two cascaded proportional-integral-derivative (PID) control architecture is used for low level controller. The attitude controller frequency is $250Hz$ and the attitude rate controller frequency is $500Hz$. For common missions such as way point tracking, there is no need to use high frequency controllers. But for aggressive trajectory tracking, the frequency of the low level controller has a significant effect upon trajectory tracking performance. Even though these frequencies are already set, these frequencies also would be efficient enough for achieving desired tracking performance.

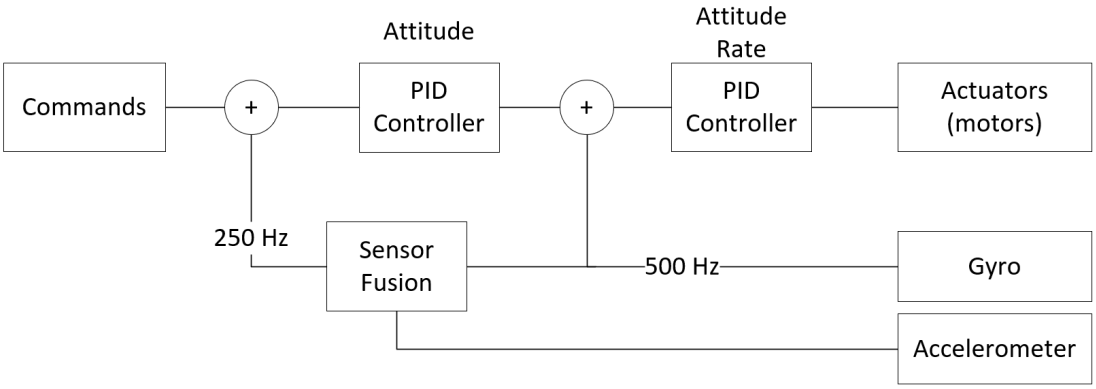


Figure 3.21 : Attitude controller inputs and outputs.

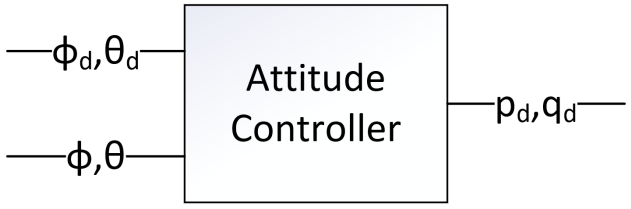


Figure 3.22 : Attitude rate controller inputs and outputs.

Inside attitude controller which is shown in Fig. 3.22, PI controllers are used for roll and pitch control. The desired pitch and roll values are obtained after calculation of

Eq. 3.53,3.52. As feedback, on-board gyro and accelerometer sensors are used to find angle states. Even the tests are performed under VICON motion capture system, the orientation outputs of the VICON are not used inside the sensor fusion. In summary, roll ϕ and pitch θ estimates compared with desired roll ϕ_d and pitch θ_d values, and proportional-integral (PI) controllers are used to find desired attitude rate values p_d, q_d, as it is shown in Eq. 3.54, 3.55.

$$p_d(t) = K_{P,\phi}(\phi_d(t) - \phi(t)) + K_{I,\phi} \int_{\tau=0}^{\tau=t} (\phi_d(\tau) - \phi(\tau))d\tau \tag{3.54}$$

$$q_d(t) = K_{P,\theta}(\theta(t) - \theta(t)) + K_{I,\theta} \int_{\tau=0}^{\tau=t} (\theta(\tau) - \theta(\tau))d\tau \tag{3.55}$$

For both roll and pitch attitude controllers, The controller coefficients are assigned as $K_{P,\phi} = K_{P,\theta} = 3.5$, $K_{I,\phi} = K_{I,\theta} = 2.0$ [60].

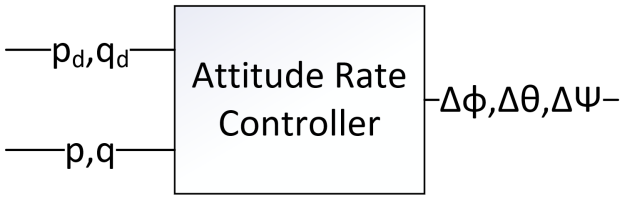


Figure 3.23 : Low level controller architecture.

After obtaining desired attitude rate commands, attitude rate controller is used to find required angular momentum. Attitude rate controller inputs and outputs are shown in Fig. 3.22. The inputs of the controller are desired roll rate p_d, desired pitch rate q_d, roll rate state p and pitch rate state q. Inside attitude rate controller, proportional controller Eq. 3.56,3.57 are used for roll and pitch control, and PI controller 3.58 is used for yaw control.

$$\Delta_\phi(t) = K_{P,p}(p_d(t) - p(t)) \tag{3.56}$$

$$\Delta_\theta(t) = K_{P,q}(q_d(t) - q(t)) \tag{3.57}$$

$$\Delta_\psi(t) = K_{P,\psi}(r(t) - r(t)) + K_{I,r} \int_{\tau=0}^{\tau=t} (r(\tau) - r(\tau))d\tau \tag{3.58}$$

For attitude rate controllers, The controller coefficients are assigned as $K_{P,p} = K_{P,q} = K_{P,r} = 70$, $K_{I,r} = 16.7$ [60].

Control mixer is used after obtaining desired input variation of the motors. These values corresponds to the desired torque values in the desired direction of movement.

To describe the movement relative to the motors, these $\Delta_\phi(t), \Delta_\theta(t), \Delta_\psi(t)$ values should be distributed to the motor outputs by using control mixer. Crazyflie is designed as "X" frame, therefore, control mixer is used as Eq. 3.59 [60].

$$\begin{aligned} PWM_{m1} &= \omega_{base} - \Delta_\phi(t)/2 - \Delta_\theta(t)/2 - \Delta_\psi(t) \\ PWM_{m2} &= \omega_{base} + \Delta_\phi(t)/2 - \Delta_\theta(t)/2 + \Delta_\psi(t) \\ PWM_{m3} &= \omega_{base} + \Delta_\phi(t)/2 + \Delta_\theta(t)/2 - \Delta_\psi(t) \\ PWM_{m4} &= \omega_{base} - \Delta_\phi(t)/2 + \Delta_\theta(t)/2 + \Delta_\psi(t) \end{aligned} \tag{3.59}$$

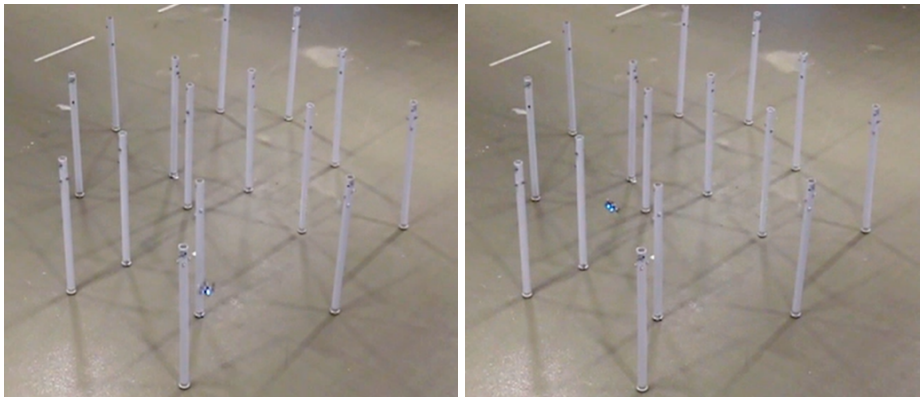
Where ω_{base} is the base PWM signal value for hovering.

In test environment, pipes are positioned randomly to provide $\rho = 1.2$ obstacle/m^2 obstacle density in Fig. 3.24. Two randomly created flight test results are shown in Fig. 3.25,3.26, and we can see the margin of error between the commanded and the actual. This performance metrics are also used in decision of safety distance from the obstacles and the Crazyflie. The trajectory tracking performance RMS error is approximately $20cm$. If the error caused by tracking performance is not added to the safety distance, even though the system has actually produced a trajectory that will avoid the obstacles, collisions still may occur due to the error in the tracking performance. Therefore, this RMS error is added to the reward function for DRL agent to come up with proper raplanned trajectory where it also compensate the trajectory tracking errors.

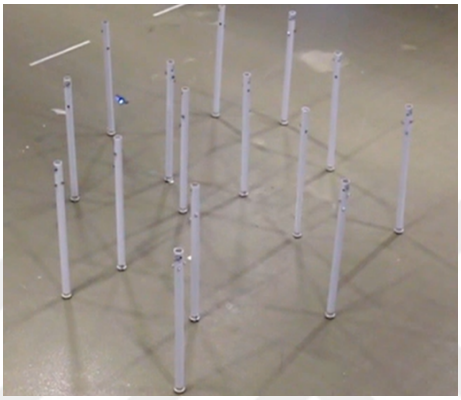
The velocity, acceleration, roll angle, pitch angle, roll rate angle and pitch rate angle graphs of these two scenarios are also shared in Fig. 3.27, 3.28. The velocity and acceleration data are obtained from VICON system, and other attitude and attitude rate data are gathered from on-board sensor fusion.

In Scenario 1, the achieved maximum velocity is $3.74m/s$, and the maximum acceleration is $11.08m/s^2$. A peak roll angle of $30.5deg$ and a peak pitch angle of $38.1deg$ was achieved during flight. A peak roll rate of $248.23deg/s$ and a peak pitch rate of $409.8deg/s$ was achieved during flight.

In Scenario 2, The achieved maximum velocity is $4.21m/s$, and the maximum acceleration is $10.19m/s^2$. A peak roll angle of $17.1deg$ and a peak pitch angle of $32.2deg$ was achieved during flight. A peak roll rate of $185.84deg/s$ and a peak pitch rate of $309.23deg/s$ was achieved during flight.



(a) First Scene (b) Second Scene



(c) Third Scene

Figure 3.24 : Frames from the flight test. The obstacles density in the environment is $\rho = 1.2$ obstacle/m^2.

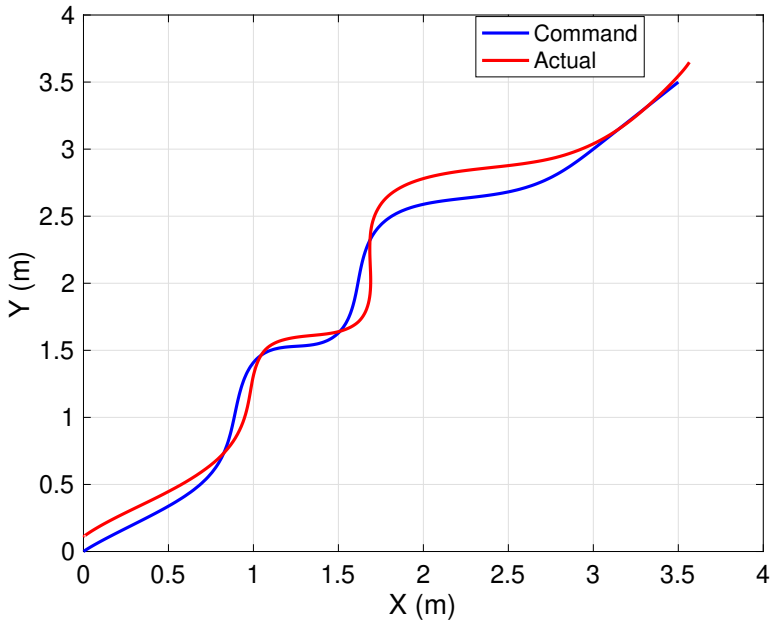


Figure 3.25 : Scenario 1: The reference trajectory is given in blue and the actual trajectory flown is given in red.

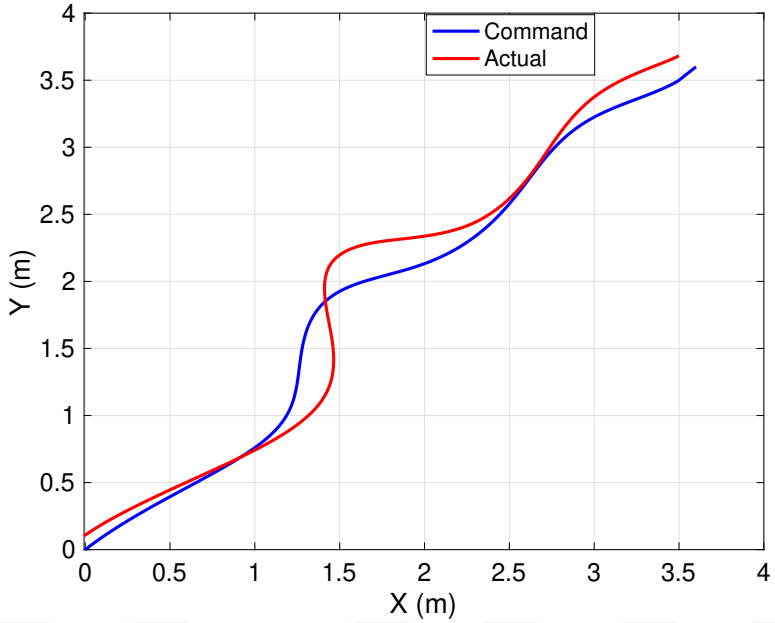


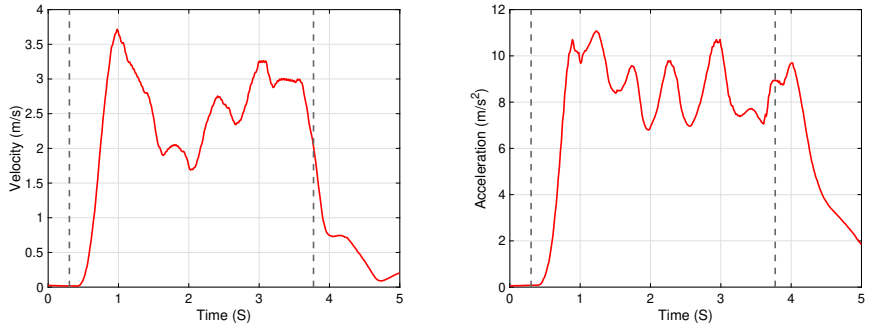
Figure 3.26 : Scenario 2: The reference trajectory is given in blue and the actual trajectory flown is given in red.

Table 3.2 : The aerial vehicle specifications.

Weight without any payload	$620g$
Maximum Speed	$30m/s$
Propeller radius	$5inch$
Nominal motor power	$530W$
Battery Cell	4
Battery Capacity	$1550mAh$
Battery discharge	$95C$
ESC nominal current	$40A$
ESC peak current	$60A$

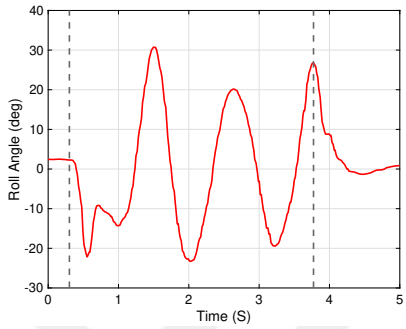
3.7 Fast Replanning Hardware Implementation Results: Outdoor Test

We also wanted to investigate the performance of the system with outdoor tests. We built a drone with 1:7.65 thrust weight ratio, because we are expecting to see our system performance when the aerial vehicle while performing aggressive maneuvers. This ratio is not as high as the racer drones, because of the additional avionics placed on the aerial vehicle. But still, the motor, the ESC and the battery performances are the same that has been used in racer drones. The specifications for the designed aerial vehicle is shown in Table 3.2.

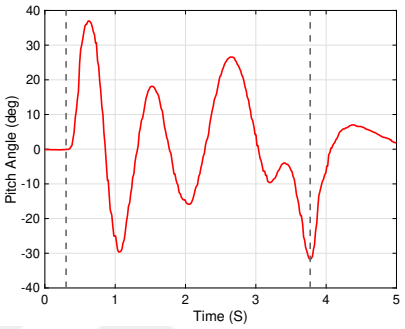


(a) Velocity

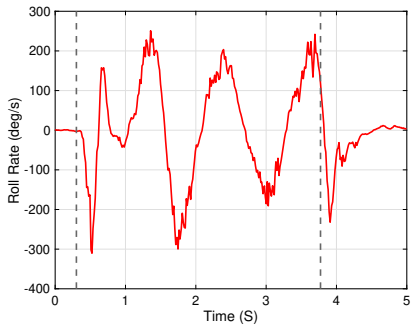
(b) Acceleration



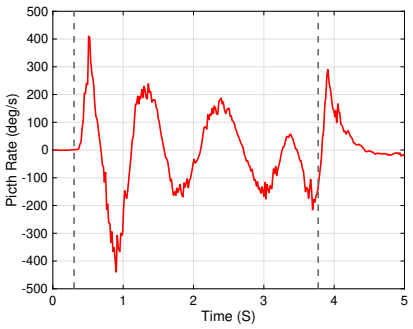
(c) Roll Angle



(d) Pitch Angle



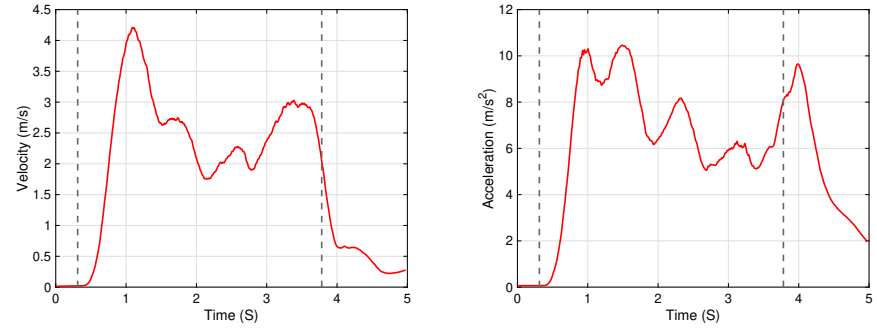
(e) Roll Rate



(f) Pitch Rate

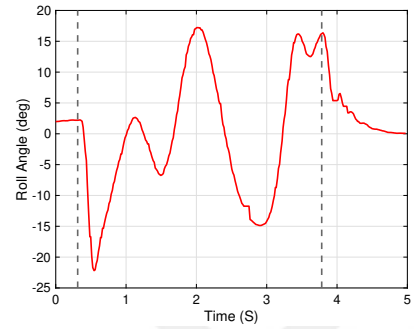
Figure 3.27 : Scenario 1: The dashes indicate the start and the end of the scenario.(a) The achieved maximum velocity is $3.74m/s$, and (b) the maximum acceleration is $11.08m/s^2$. (c) A peak roll angle of $30.5deg$ was achieved during flight. (d) A peak pitch angle of $38.1deg$ was achieved during flight.(e) A peak roll rate of $248.23deg/s$ was achieved during flight. (f) A peak pitch rate of $409.8deg/s$ was achieved during flight.

We used Orange Pi Zero Plus2 as a computer on the aerial vehicle. It has Armbian based operating system that allows us to use our Python based codes to run and generate desired referances from trajectory and send them to the autopilot. It has Quad-core Cortex-A7 microcontroller and Mali400MP2 GPU. For autopilot system, we used Omnibus F4 Pro, which is light-weighted and very small controller card. It allows us to run ArduCopter autopilot system to control the aerial vehicle. It

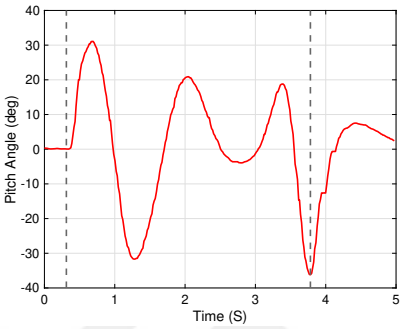


(a) Velocity

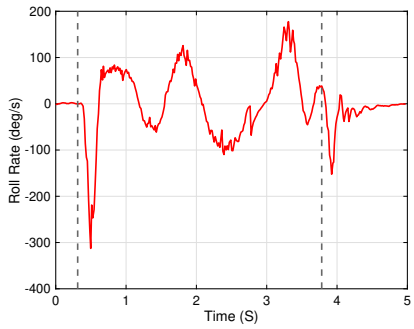
(b) Acceleration



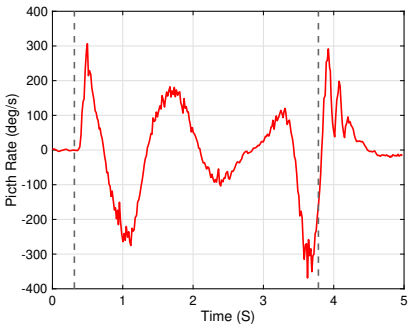
(c) Roll Angle



(d) Pitch Angle



(e) Roll Rate



(f) Pitch Rate

Figure 3.28 : Scenario 2: The dashes indicate the start and the end of the scenario.(a) The achieved maximum velocity is $4.21 m/s$, and (b) the maximum acceleration is $10.19 m/s^2$. (c) A peak roll angle of $17.1 deg$ was achieved during flight. (d) A peak pitch angle of $32.2 deg$ was achieved during flight.(e) A peak roll rate of $185.84 deg/s$ was achieved during flight. (f) A peak pitch rate of $309.23 deg/s$ was achieved during flight.

has MPU6000 IMU and BMP280 barometer sensors. As a GNSS receiver, we used mRo SAM-M8Q which includes u-Blox SAM-M8Q module. ALso inside the mRo SAM-M8Q, there is IST8308 magnetometer sensor. This sensor especially placed higher level then the motors the power cable level. This ensures that the magnetic interference from the motors and power cable are as low as possible. And for RC receiver, we used FrSky FS-A8S product for controlling the aerial vehicle with the

remote control. It is placed on the vehicle to instantly intervene in possible problems that may occur. The avionics schematics are shown in Fig. 3.29, and the power distribution schematics are presented in Fig. 3.30.

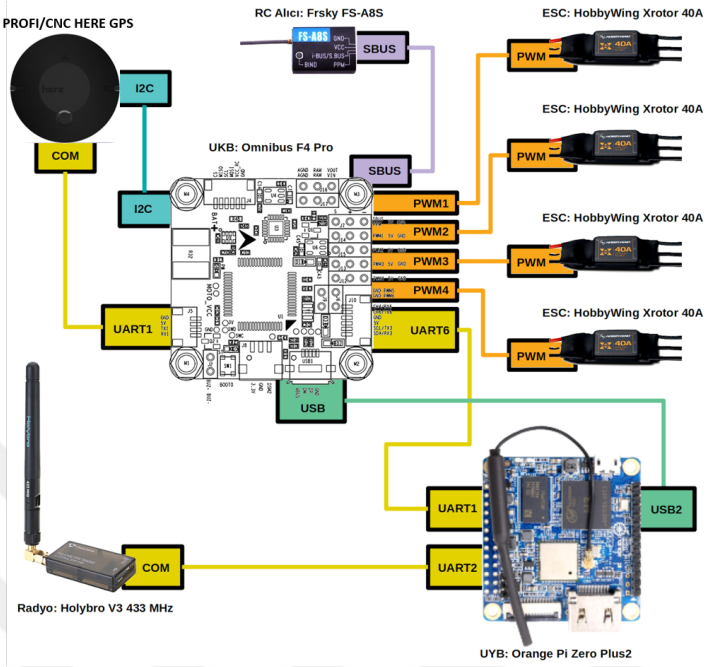


Figure 3.29 : Avionics and the communication interfaces on the aerial vehicle.

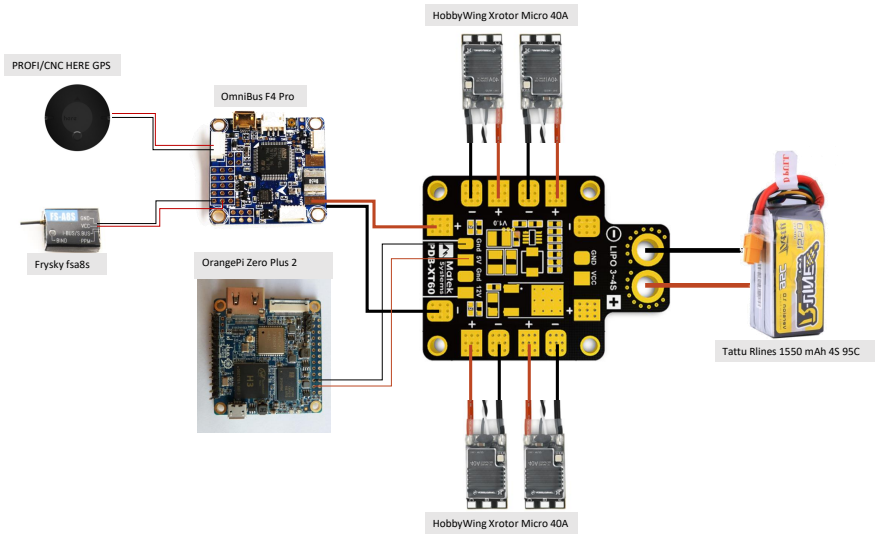


Figure 3.30 : Power distribution schematics.

The same procedure is used as in the tests conducted under VICON system. The differences between each tests are, the trajectory references are sent in $40Hz$. Because of the expected trajectory tracking performance, GNSS accuracy, wind and gust effects, the emulated sensor range is modified as $6m$ and $30\,\text{deg}$, which still narrower than the used sensors in the literature. This modification allows us to perform our tests

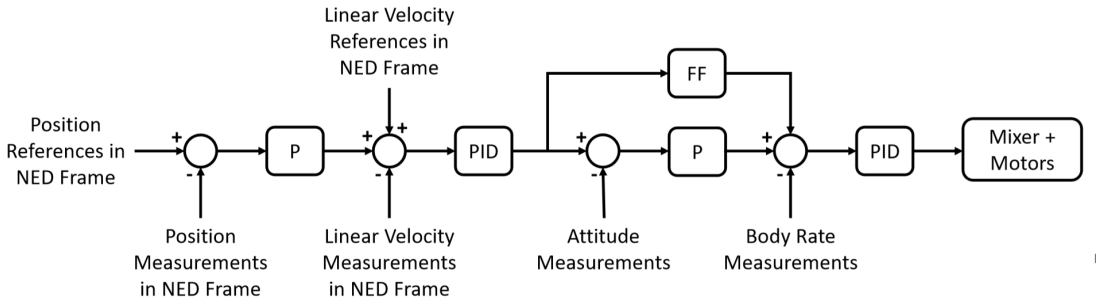


Figure 3.31 : Ardupilot position control system schematics.

safely. Also, two different high level controller is used to compare the performances of the trajectory tracking controllers and also monitor the overall system performance.

Differential Flat Trajectory is corresponds to the B-spline trajectory generation. Before the flight, straight trajectory is defined which starts from $[0,0]$ point to $[40.0,40.0]$ point for testing.

Ardupilot Controller Architecture is a legacy control system that consists of nested-loop structure [62]. The simplified version of the controller architecture is shown in Fig. 3.31. At the beginning, position references in NED frame are sent to the controller. The error of the position is pass through proportional controller. Then, this reference and the error of the velocity in NED frame which is calculated by extracting the reference velocity and the measured velocity are summed. PID controller is used for velocity controller. The output of the velocity controller is used for the input of the attitude controller and the input for the attitude rate controller as feed forward element. Proportional controller for attitude control and PID controller for attitude rate control is utilized, then the output of this inner-loop rate controller is sent to the mixer to obtain the required PWM signals. To indicate this outer loop control, we will call "pos+vel controller".

In this control architecture, position and velocity references generated from the trajectories can be utilized. The outer loop controller works in $100Hz$, and the inner loop controller works in $400Hz$. We tried two different scenarios to analyze the performans of the overall system.

The velocity, acceleration, roll angle, pitch angle, roll rate angle and pitch rate angle graphs of the scenario-3 are also shared in Fig. 3.33. The position and velocity data are

obtained from Extended Kalman Filter running on-board which utilizes the on-board sensors data system. Also for outer loop control, pos+vel controller is used.

In Scenario-3 with pos+vel controller in Fig. 3.33, the trajectory tracking and obstacle avoidance performance can be seen in Fig. 3.32. The RMS error of the trajectory tracking performance is $3.55m$. The achieved maximum velocity in X-Y axis is $12.06m/s$, and the maximum acceleration in X-Y axis is $12.36m/s^2$. A peak roll angle of $51.46deg$ was achieved during flight. A peak pitch angle of $45.77deg$ was achieved during flight. A peak roll rate of $451.27deg/s$ was achieved during flight. A peak pitch rate of $328.26deg/s$ was achieved during flight.

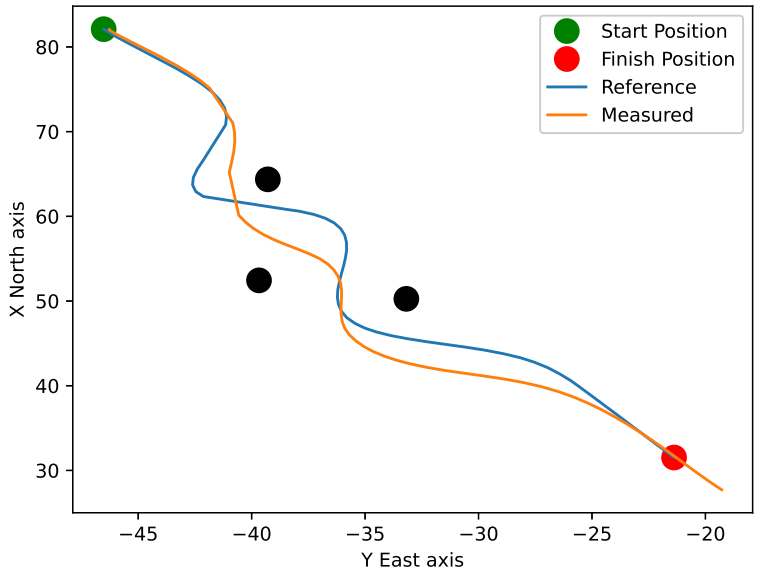
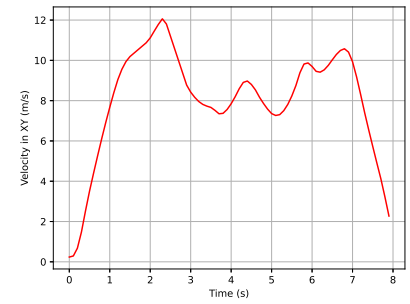
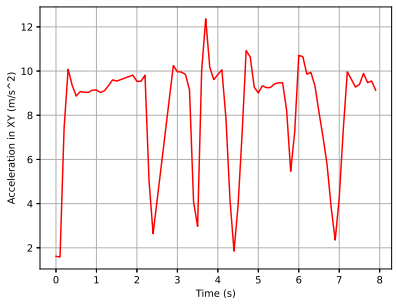


Figure 3.32 : Scenario 3: pos+vel controller. The results show the performance of the controllers for trajectory tracking and obstacle avoidance.

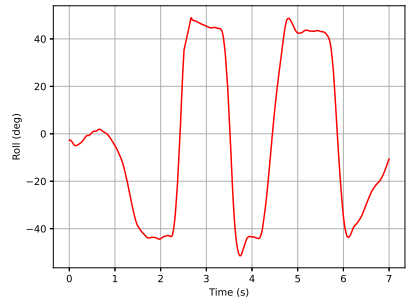
In Fig. 3.32, even though the generated trajectory with RL agent is differentially flat and dynamically feasbile trajectory, because of the tracking performance, the aerial vehicle responds late to the references. This phenomenon might cause a collision, even the RL agent is generating evasive maneuver. Also, the RMS error of this trajectory tracking is not providing the expected performance. But, to overcome this RMS error performance, the maximum error faced in other random scenarios is added to the safety distance that is used in reward function of the RL agent. Therefore, even this lack of a precision of the tracking is unwanted, the generated trajectory still can avoid the sensed obstacles. Although this approach guarantees safety, the solution diverges from the local optimality.



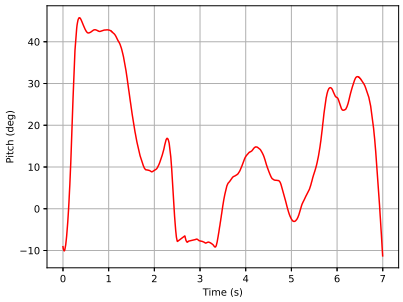
(a) Velocity



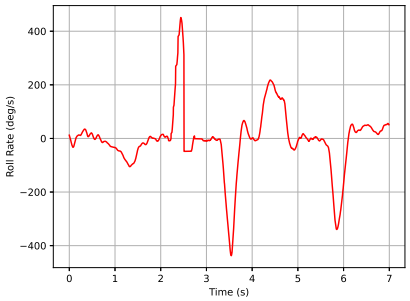
(b) Acceleration



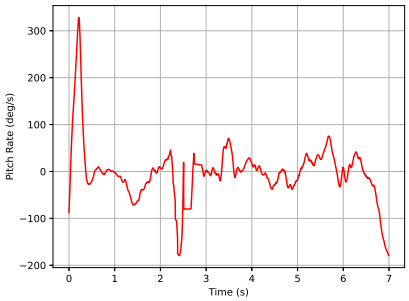
(c) Roll Angle



(d) Pitch Angle



(e) Roll Rate



(f) Pitch Rate

Figure 3.33 : Scenario 3: the performance of the flight tests where for outer loop control, pos+vel controller is utilized.(a) The achieved maximum velocity in X-Y axis is 12.06m/s, and (b) the maximum acceleration in X-Y axis is 12.36m/s^2. (c) A peak roll angle of 51.46deg was achieved during flight. (d) A peak pitch angle of 45.77deg was achieved during flight.(e) A peak roll rate of 451.27deg/s was achieved during flight. (f) A peak pitch rate of 328.26deg/s was achieved during flight.

To overcome this case, we modified the outer loop control of the ArduPilot, and also added acceleration references of the trajectory as a feed forward input to the output of the velocity controller. For this controller, we will call "pos+vel+acc controller". The modified outer loop control schematics can be seen in Fig. 3.41.

The velocity, acceleration, roll angle, pitch angle, roll rate angle and pitch rate angle graphs of the scenario-4 are also shared as for outler loop control, in Fig. 3.36, pos+vel

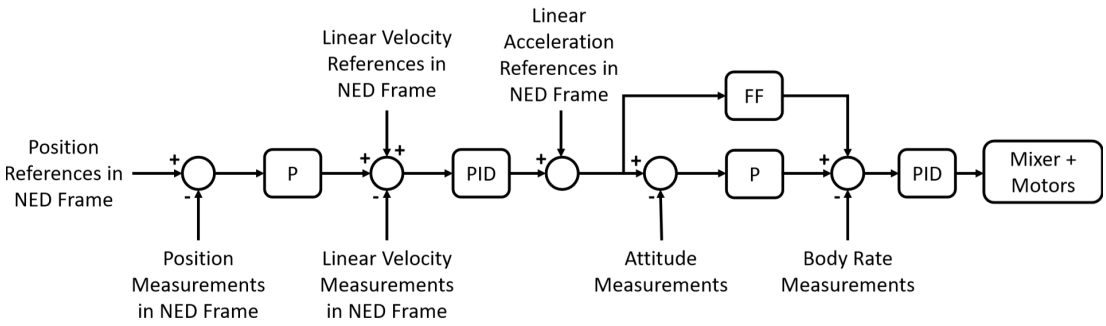


Figure 3.34 : Ardupilot modified position control system schematics.

controller is used, and in Fig. 3.37 pos+vel+acc controller is used. For each figure, the same obstacle positions, start point and finish point is used. Also, the generated evasive trajectory is used for both controller to compare the performances. The trajectory tracking performance can be seen in Fig. 3.35

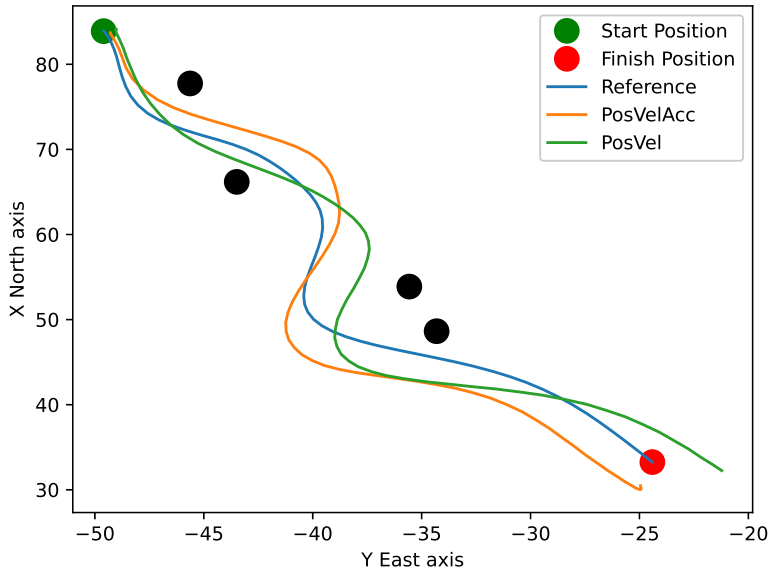
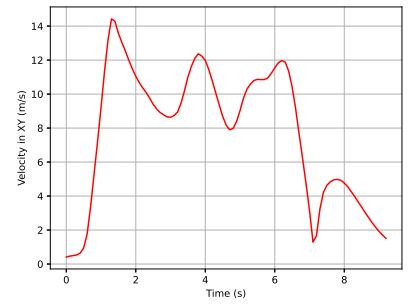
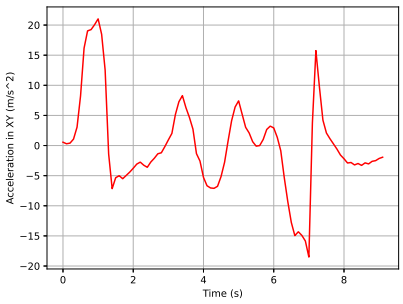


Figure 3.35 : Scenario 3: pos+vel controller vs pos+vel+acc controller. The results show the peformance of the controllers for trajectory tracking and obstacle avoidance.

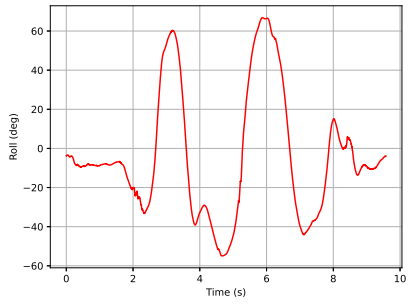
In Scenario-4 with pos+vel controller in Fig. 3.36, The RMS error of the trajectory tracking is 3.65m. The achieved maximum velocity in X-Y axis is 14.42m/s, and the maximum acceleration in X-Y axis is 21.03m/s^2. A peak roll angle of 66.83deg was achieved during flight. A peak pitch angle of 68.36deg was achieved during flight. A peak roll rate of 412.23deg/s was achieved during flight. A peak pitch rate of 320.14deg/s was achieved during flight.



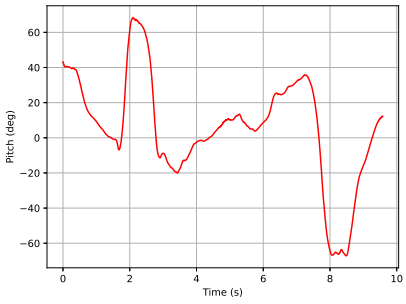
(a) Velocity



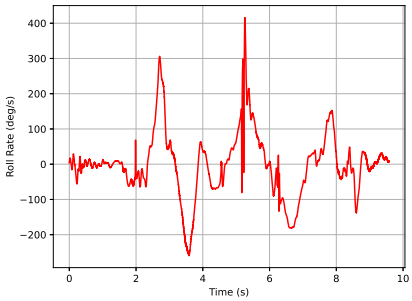
(b) Acceleration



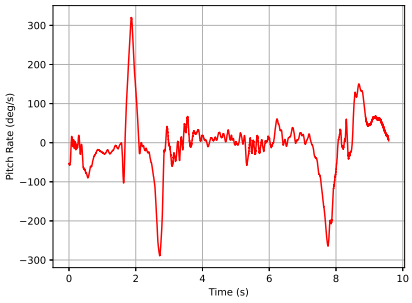
(c) Roll Angle



(d) Pitch Angle



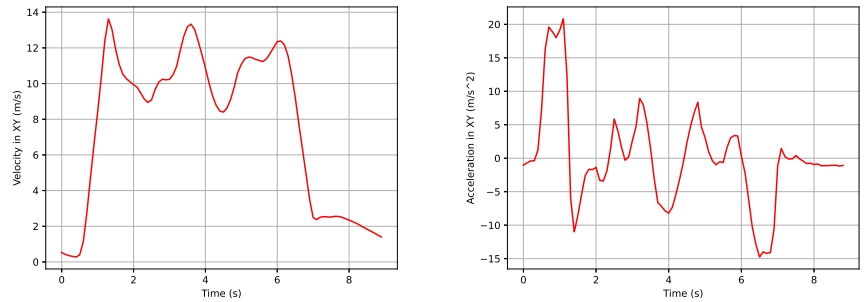
(e) Roll Rate



(f) Pitch Rate

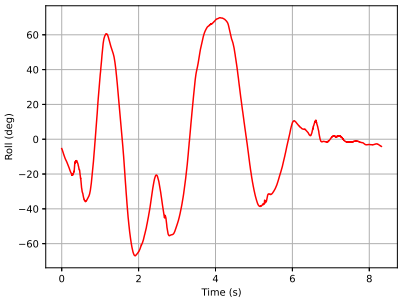
Figure 3.36 : Scenario 4: the performance of the flight tests where for outer loop control, pos+vel controller are utilized.(a) The achieved maximum velocity in X-Y axis is 14.42m/s, and (b) the maximum acceleration in X-Y axis is 21.03m/s^2. (c) A peak roll angle of 66.83deg was achieved during flight. (d) A peak pitch angle of 68.36deg was achieved during flight.(e) A peak roll rate of 412.23deg/s was achieved during flight. (f) A peak pitch rate of 320.14deg/s was achieved during flight.

In Scenario-4 with pos+vel+acc controller in Fig. 3.36, The RMS error of the trajectory tracking is 2.93m. The achieved maximum velocity in X-Y axis is 13.62m/s, and the maximum acceleration in X-Y axis is 20.82m/s^2. A peak roll angle of 69.84deg was achieved during flight. A peak pitch angle of 70.29deg was achieved during flight. A peak roll rate of 386.80deg/s was achieved during flight. A peak pitch rate of 346.22deg/s was achieved during flight.

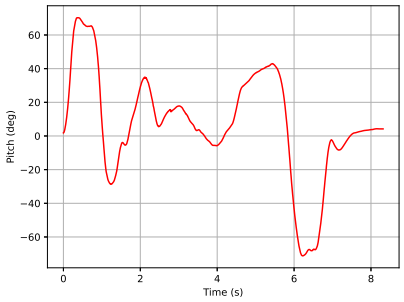


(a) Velocity

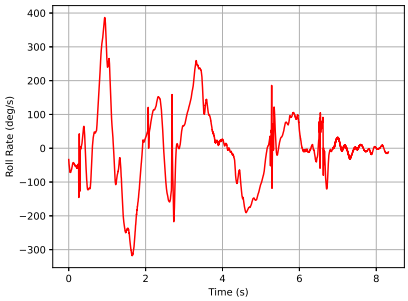
(b) Acceleration



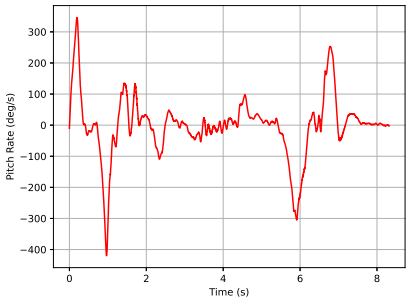
(c) Roll Angle



(d) Pitch Angle



(e) Roll Rate



(f) Pitch Rate

Figure 3.37 : Scenario 4: the performance of the flight tests where for outer loop control, pos+vel+acc controller is utilized.(a) The achieved maximum velocity in X-Y axis is $13.62 m/s$, and (b) the maximum acceleration in X-Y axis is $20.82 m/s^2$. (c) A peak roll angle of $69.84 deg$ was achieved during flight. (d) A peak pitch angle of $70.29 deg$ was achieved during flight.(e) A peak roll rate of $386.80 deg/s$ was achieved during flight. (f) A peak pitch rate of $346.22 deg/s$ was achieved during flight.

When we compare these two controllers, the significant response time difference draws the attention. Also, pos+vel+acc controller performance is better when we compare the controllers with RMS errors.

We conducted another scenario with pos+vel+acc controller. This trajectory however far more aggressive than the previous scenarios. Because of capturing desired

responses with pos+vel+acc controller, we wanted to force the aerial vehicle at its limits. The flight named as scneario-5 which can be seen in Fig. 3.38.

The RMS error of the trajectory tracking performance is $3.41m$. The achieved maximum velocity in X-Y axis is $18.43m/s$, and the maximum acceleration in X-Y axis is $19.10m/s^2$. A peak roll angle of $77.02deg$ was achieved during flight. A peak pitch angle of $70.22deg$ was achieved during flight. A peak roll rate of $522.01deg/s$ was achieved during flight. A peak pitch rate of $365.59deg/s$ was achieved during flight. The results can be seen in Fig. 3.39.

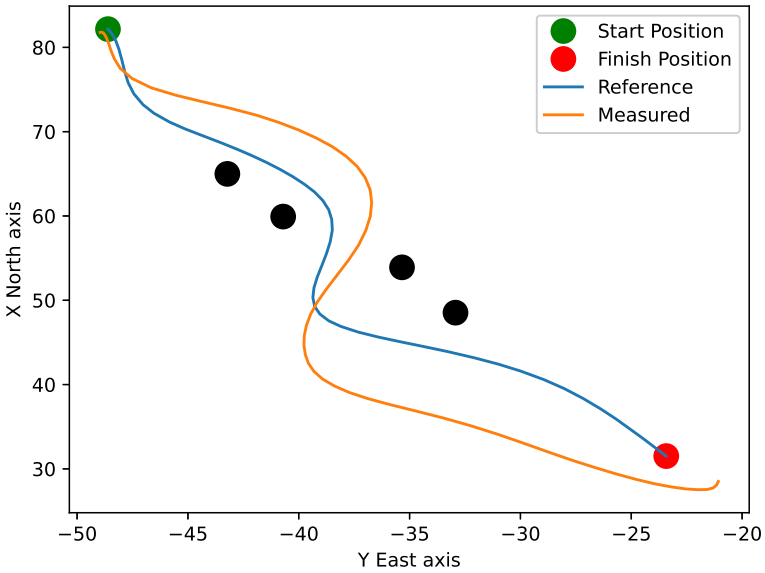
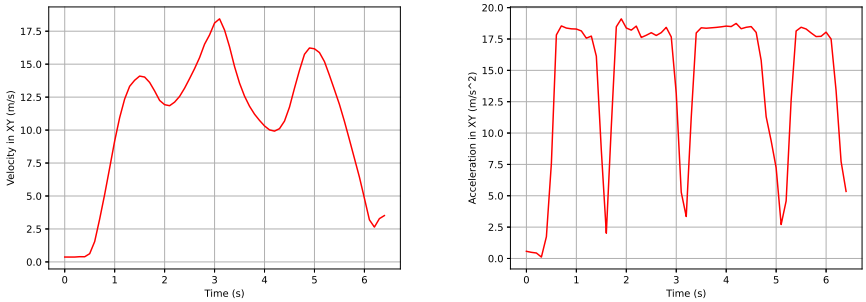


Figure 3.38 : Scenario 5: pos+vel+acc controller. The results show the performance of the controllers for trajectory tracking and obstacle avoidance.

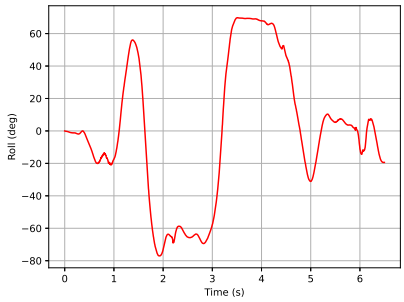
In this scenario, It can be seen that acceleration values through to trajectory is almost at its limits. This also indicates the aggressiveness of the vehicle, where it also can be observed from the roll angle plot, where the aerial vehicles reaches to the 70 deg roll angle and stays there almost more than 1 second. After exceeding 70 deg roll angle, the aerial vehicle can not maintain its altitude level, which is also observed from the previous flights. Therefore, the limits for the attitude and also the limits for the acceleration in X-Y axis are hard coded before the flight tests. So, the results indicates that the vehicle during the flight time, it flies almost around its dynamical limits.

In the proposed collision avoidance system, with the utilization of the B-Spline trajectory representation and using its knot insertion properties to only change the

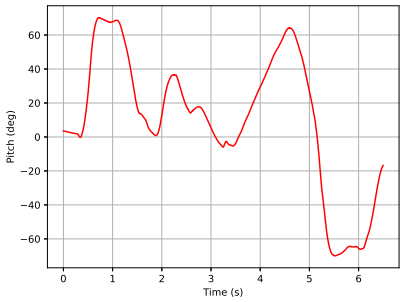


(a) Velocity

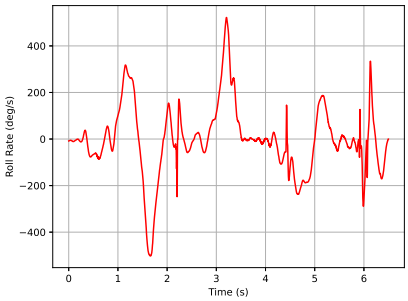
(b) Acceleration



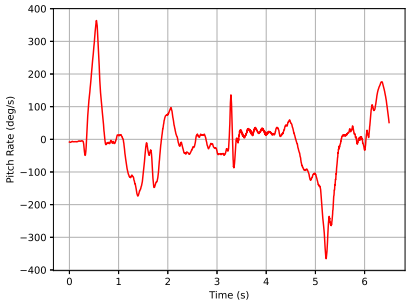
(c) Roll Angle



(d) Pitch Angle



(e) Roll Rate



(f) Pitch Rate

Figure 3.39 : Scenario 5: the performance of the flight tests where for outer loop control, pos+vel+acc controller is utilized. (a) The achieved maximum velocity in X-Y axis is $18.43 m/s$, and (b) the maximum acceleration in X-Y axis is $19.10 m/s^2$. (c) A peak roll angle of $77.02 deg$ was achieved during flight. (d) A peak pitch angle of $70.22 deg$ was achieved during flight.(e) A peak roll rate of $522.01 deg/s$ was achieved during flight. (f) A peak pitch rate of $365.59 deg/s$ was achieved during flight.

trajectory around the sensed obstacle, we formulated a new collision avoidance approach based on deep reinforcement learning. By only changing the control point which is located on the obstacle with knot insertion property, this proposed method replans the trajectory. Because of the search space for the control point is continues, the generated new trajectory ensures the local optimality. Moreover, B-spline trajectory representation allows us to take the derivative of each point to obtain the position

Figure 3.40 : One of the outdoor flight tests scene with using pos+vel controller.

Figure 3.41 : One of the outdoor flight tests scene with using pos+vel+acc controller.

derivatives of each point on curve, which allows us to use differential flat controllers to track the trajectory. Since the problem is reduced to just changing the location of the control point, the trained RL agent can generates new location in 400us, which is the fastest solution in the literature to the best of our knowledge and research.

However, the trajectory tracking performance also has significant effect on optimality. In VICON system, we used LQI differential flatness controller to track the trajectory. By using Crazyflie aerial vehicle, we achived 20cm RMS error where we used VICON system outputs for feedback which has millimeter accuracy and there is no wind or gust effects on the aerial vehicle. To capture these effects, we designed a aerial vehicle, which has 1:7.6 thrust-to-weight ratio that indicates the level of agility. We utilized

ArduPilot autopilot on the aerial vehicle, where it already has a trajectory tracking controller, which only utilizes position and velocity references. Because it could not give the desired performance, the controller on the ArduPilot is modified by adding feed forward acceleration references. This controller shows better performance, when we compare it with the existing controller on the ArduPilot. The trajectory tracking controller precision has a significant effect on all high level operational missions. The preciseness of the controller could increase action set for the search space, and this could provide finding more optimal solutions for collision avoidance. Therefore, this thesis also focused on the trajectory tracking controller with proposing a new approach, which is mentioned in Section 4.

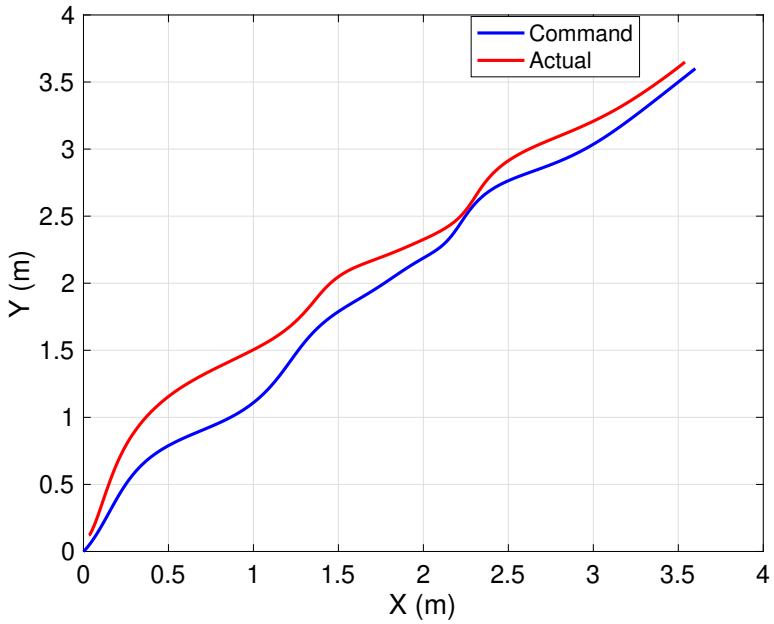


Figure 4.1 : The reference trajectory is given in blue and the actual trajectory flown is given in red.

After conducting trajectory replanning flight tests, trajectory tracking performance has proved to be a crucial element for obstacle avoidance. Because if it is assumed that there is no trajectory tracking error and the safety distance is only defined by adding the radius of the aerial vehicle and the radius of the obstacle, then avoidance maneuver is expected to be very close to the obstacle. If this approach is followed, then as it can be seen in Fig. 4.1, there is $21cm$ RMS error that will definitely cause a collision. To overcome this problem, this RMS error is also added to the safety distance to guarantee the avoidance.

In the training phase of the RL agent, trajectory tracking error is added to the safety distance which ensures the evasive distance from the sensed obstacle. This solution guarantees collision avoidance, on the other hand, the performance takes us away from the locally optimal solution. Also, this phenomenon increase the safety time that is decided to ensure computational feasibility that considers computational and time

complexity of the utilized algorithms. Therefore, It is imperative to find a solution to achieve a precise trajectory tracking performance.

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 Fig. 4.2. The trajectory generation block generates many random agile trajectories consisting of position and velocity references that meet the aerial vehicle dynamical constraints given in section 4.3.2. The trajectory reference is then given to the agent along with the observation vector and the reward as defined in Eq. 4.20-4.22 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. 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. 4.2-4.6 are then used to simulate the aerial vehicle dynamics and outputs the reward and the observation vector to close the training loop.

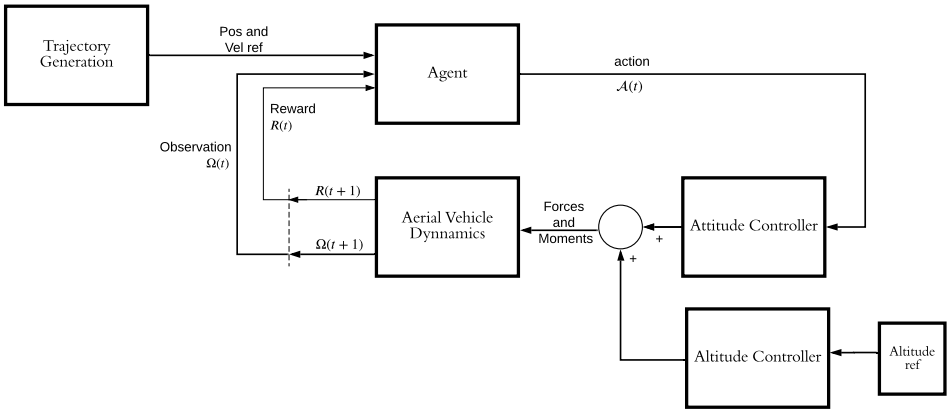


Figure 4.2 : Proposed system architecture.

4.2 Aerial Vehicle Model

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:

$$R_o^b = \begin{bmatrix} cos\theta cos\psi & cos\theta sin\psi & -sin\theta \\ sin\phi sin\theta cos\psi - cos\phi sin\psi & sin\phi sin\theta sin\psi + cos\phi cos\psi & sin\phi cos\theta \\ cos\phi sin\theta cos\psi + sin\phi sin\psi & cos\phi sin\theta sin\psi - sin\phi cos\psi & cos\phi cos\theta \end{bmatrix} \quad (4.1)$$

The acceleration of the air-vehicle is then defined in the body-fixed frame as follows [63]:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ F_z/m \end{bmatrix} - R_o^b \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{4.2}$$

And the position of the air-vehicle in the inertial frame is expressed by integrating the output of the following equation [63]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_b^o \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{4.3}$$

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

From the theorem of angular momentum, the angular velocities are expressed as [63]:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = (J)^{-1} \left(\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times J \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \tag{4.4}$$

where J is the inertia matrix given as:

$$J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{4.5}$$

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.

Finally, the Euler angles are given as [63]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sin\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi/cos\theta & cos\phi/cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \text{ for } \theta \neq \frac{\pi}{2} \tag{4.6}$$

4.3 Trajectory Generation

4.3.1 Trajectory representation

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:

$$P(t) = \sum_{i=0}^{n} P_i B_{i,p}(t) a \leq t \leq b \tag{4.7}$$

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 [47, 48, 64].

$$B_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

$$B_{i,p}(t) = \frac{u - u_i}{u_{i+p} - u_i} B_{i,p-1}(t) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} B_{i+1,p-1}(t) \tag{4.9}$$

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

$$\tau = [u_0, ..., u_m] \tag{4.10}$$

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

$$\tau = [a, ..., a, t_{p+1}, ..., t_{m-p-1}, b, ..., b] \tag{4.11}$$

The length of the knot vector is $m+1$ where $m = n+p+1$ [64]. 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:

$$P^{(k)}(t) = \sum_{i=0}^{n} P_i B_{i,p}^{(k)}(t) a \leq t \leq b \tag{4.12}$$

where k represents the order of derivatives.

4.3.2 Aerial vehicle dynamical constraints

Before giving a trajectory to the agent to track, we had to make sure it is dynamically feasible. Therefore, we optimized all provided B-spline trajectories to ensure not exceeding the dynamical limits of the aerial vehicle. The vehicle used for this work is the Crazyflie 2.1 [60]. Based on Newton's second law of motion, we get the following classical mechanics equation:

$$F_{thrust} - F_{gravity} - F_{AirDrag} = m\dot{v} \tag{4.13}$$

In the vertical climbing case, Eq. 4.13 can be written as:

$$T - mg - \frac{\rho}{2} C_D A_{eff} v^2 = m\dot{v} \tag{4.14}$$

And in the forward flight case, Eq. 4.13 can be written as:

$$\sqrt{1-(\frac{mg}{T})^2}\,T-\frac{\rho}{2}C_DA_{eff}v^2=m\dot{v} \tag{4.15}$$

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:

$$\frac{T_{max}}{m}-g-\frac{\rho}{2m}C_DA_{eff}v_{max}^2=0 \tag{4.16}$$

$$\sqrt{1-(\frac{mg}{T_{max}})^2}\,\frac{T_{max}}{m}-\frac{\rho}{2m}C_DA_{eff}v_{max}^2=0 \tag{4.17}$$

Table 4.1 shows the parameters used in solving Eq. 4.16 to find the maximum rate of climb and the maximum forward flight speed. By solving Eq. 4.16, the maximum climb rate is found as $v_{climb,max}=2.96\ m/s$. And by solving Eq. 4.17, the maximum forward velocity is found as $v_{forward,max}=7.36\ m/s$. Fig. 4.3 shows the maximum forward acceleration as a function of the Crazyflie's velocity.

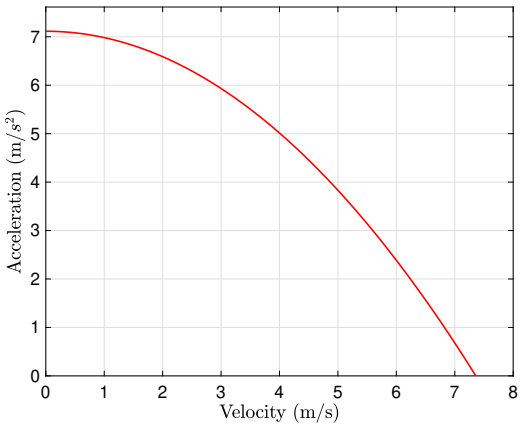


Figure 4.3 : Forward acceleration limits.

Table 4.1 : Crazyflie aerial vehicle parameters.

Parameter	Description	Value
m	Mass of the drone	0.034 [Kg]
T_{max}	Maximum available thrust	0.41202 [N] [60]
g	Gravitational acceleration	9.81 [m/s^2]
ρ	Density of air	1.225 [kg/m^3]
C_D	Drag coefficient	0.9 [65]
A_{eff}	Effective area of the Crazyflie	8.1×10^{-3} [m^2] [60]

4.4 Deep Reinforcement Learning - 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 [50]. 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 [50]. Define the probability ratio, $r_t(\theta)$, as follows:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad r(\theta_{old}) = 1 \tag{4.18}$$

The work in [50] proposes a new objective function defined as follows:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)\right] \tag{4.19}$$

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:

$$\Omega_1 = \{P_x(t) - x(t),\ P_y(t) - y(t),\ \dot{P}_x(t) - u(t),\ \dot{P}_y(t) - v(t)\} \tag{4.20}$$

$$\Omega = \{\Omega_1, \mathscr{A}(t-1) - \mathscr{A}(t-2),\ x(t),\ y(t),\ z(t)\} \tag{4.21}$$

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:

$$R = 2\sum_{i=1}^{5}(\ e^{-\alpha_i \Omega(i)} - 1\ ) \tag{4.22}$$

Where Ω is the observation vector defined in Eq. 4.21 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 4.2 shows the selected α values and the corresponding error value that determines when the agent gets a high positive reward.

Table 4.2 : 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.

4.5 Simulation Experiments

For training a DRL agent, we used OpenAI gym and stable-baseline environment [57] [66]. 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 to compare performances with our solution.

4.5.1 LQR differential flatness based controller

The Crazyflie comes equipped with a PID attitude rate controller and a PI attitude controller that accepts the inputs as the total desired thrust T_d, roll angle ϕ_d, pitch angle θ_d, and body yaw rate r_d and the control input is defined as:

$$v = \begin{bmatrix} T_d \\ \phi_d \\ \theta_d \\ r_d \end{bmatrix} \tag{4.23}$$

The work in [38] implemented differential flatness based controller in order to minimize the error in tracking agile trajectories. Assuming that the given trajectory is a flat output [67] and can be written as:

$$P(t) = \begin{bmatrix} P_x & P_y & P_z & \dot{P}_x & \dot{P}_y & \dot{P}_z & \ddot{P}_x & \ddot{P}_y & \ddot{P}_z & \psi_{ref} & \dot{\psi}_{ref} \end{bmatrix}^T \tag{4.24}$$

The state and control inputs are then defined as:

$$X^r = \begin{bmatrix} P_x & P_y & P_z & \dot{P}_x & \dot{P}_y & \dot{P}_z & \psi_{ref} \end{bmatrix}^T \tag{4.25}$$

and

$$u^r = \begin{bmatrix} \ddot{P}_x & \ddot{P}_z & \ddot{P}_z + g & \dot{\psi}_{ref} \end{bmatrix}^T \tag{4.26}$$

Figure 4.4 shows the architecture of the LQR based differentially flat controller as implemented in [38] where the LQR controller block 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 becomes:

$$\dot{\tilde{X}} = A\tilde{X} + B\tilde{u} \tag{4.27}$$

A full state feedback controller is used where $\tilde{u} = -K\tilde{X}$. By using Bryson's rule [68] and *lqr* command in Matlab, the K matrix that minimizes [68]

$$J = \int_0^\infty X^T QX + u^T Ru \, dt \tag{4.28}$$

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

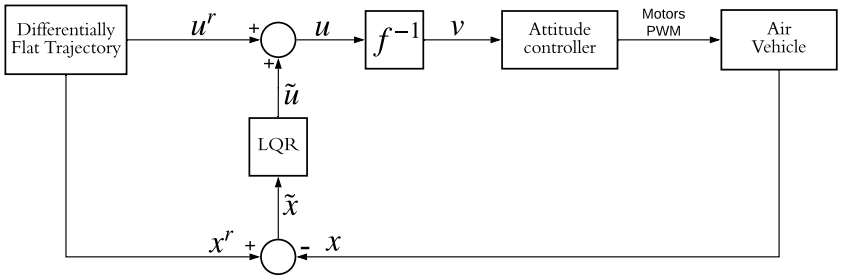


Figure 4.4 : Architecture of LQR Differentially Flat based Controller.

4.5.2 LQI differential flatness based controller

In order to further improve the tracking performance of the high-level controller, the state-space representation can be modified to include the integration of the position states to further minimize the tracking error in agile maneuvers, as follows:

$$X = \begin{bmatrix} x & y & z & u & v & w & \psi & \int x & \int y & \int z \end{bmatrix}^T \tag{4.29}$$

And the state-space model becomes:

$$\dot{X} = AX + Bu - bg \tag{4.30}$$

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

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

The LQR block shown in Figure 4.4 is replaced with an LQI block. In the LQI block, the integral of the position error is combined with the position and the velocity errors. This error is then driven to zero by finding the control input $u = -K * error$ that minimizes the cost function shown in Eq. 4.28.

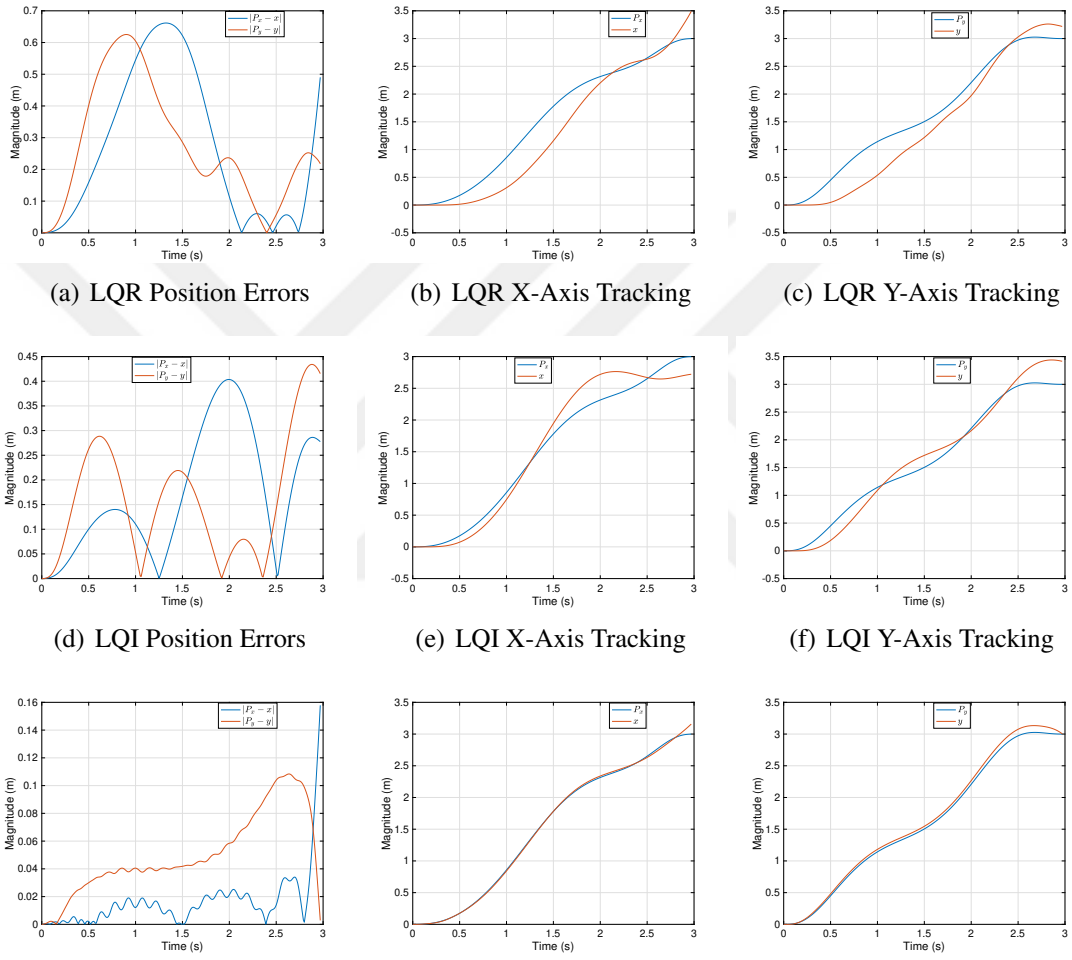
4.5.3 Performance comparison

We modeled the aerial vehicle using Eq.4.2 - 4.6 and by using the physical parameters defined in [63], 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 build a separate model for the LQR/LQI differential flatness based controllers in order to compare the tracking performance between these two approaches and our approach.

Fig. 4.5 shows the simulation results and the controllers position tracking performance, and Fig. 4.6 shows the results for velocity tracking performance. The trajectory shown in Fig.4.5 and 4.6 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 Fig. 4.3, 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 4.3.

Table 4.3 : Error RMS values as a result of 500 Monte-Carlo analysis.

RMS Value	LQR based Diff Flatness	LQI based Diff Flatness	Our 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

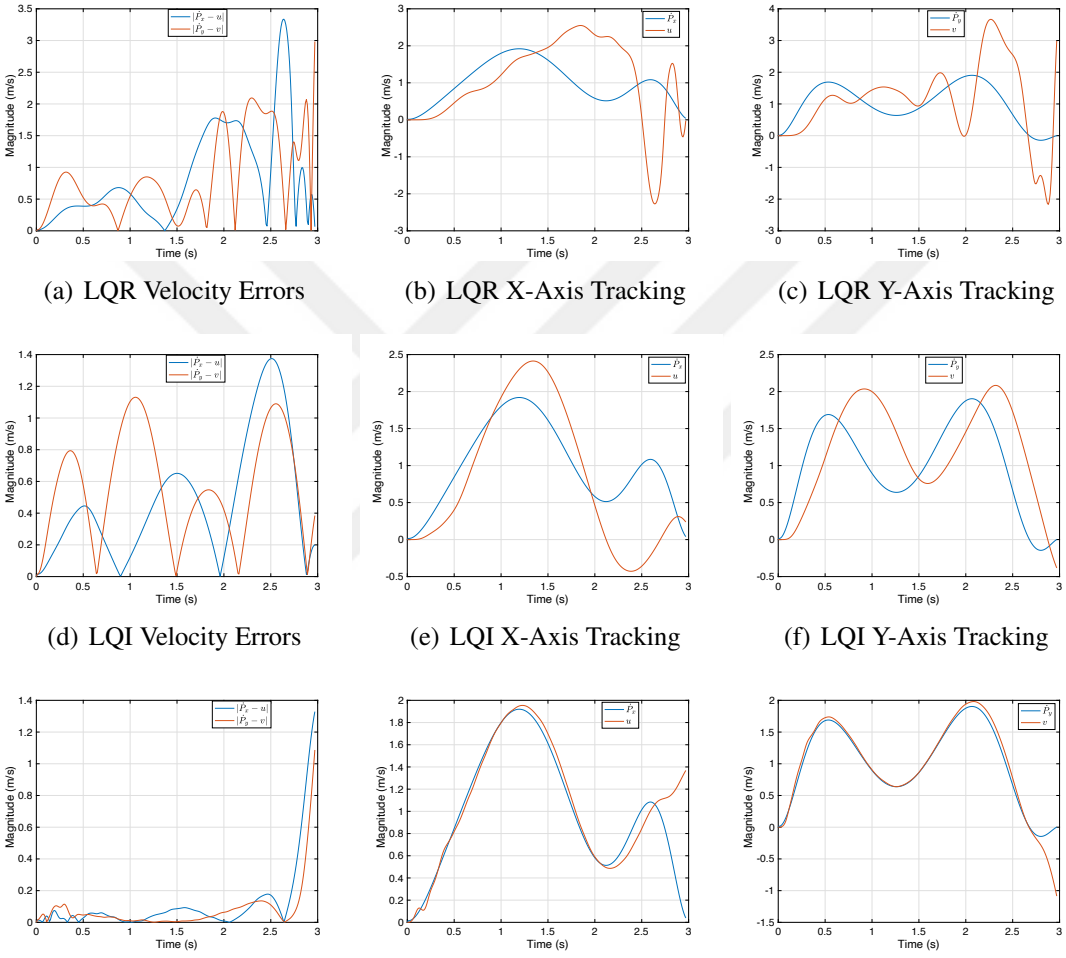


(a) LQR Position Errors (b) LQR X-Axis Tracking (c) LQR Y-Axis Tracking

(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 4.5 : Comparison between three different 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 4.6 : Comparison between three different trajectory tracking controllers (velocity).

5. CONCLUSIONS AND FUTURE RECOMMENDATIONS

In this thesis, fast trajectory replanning methodology based on B-spline regeneration with deep reinforcement learning is proposed. The aim was to simplify and accelerate the replanning problem by providing a rigorous local solution without breaking continuity over the trajectory. B-spline theory is applied for local support property and apply it for defining the agile flight trajectory. For regenerating the local trajectory segments, knot insertion methodology with control point reallocation has been used. As the geometric and dynamic form of the trajectory based on the location of the control points and the knot intervals, the control point reallocation problem is turned into a constrained optimization problem and solved through Deep Reinforcement Learning (DRL). With Principal Proxy Optimization (PPO), constrained optimization problem have been solved by enabling to generate dynamically feasible (considering dynamical constraints) local trajectory segment providing fast collision avoidance. DRL agent is trained with different environmental complexities, as it is defined through obstacle number per m^2. Through the batch simulations, the proposed methodology shows that it is enabling to solve fast trajectory replanning problem under given or hard dynamical constraints, and providing real-time applicability for such fast collision avoidance applications in agile unmanned aerial vehicles.

Also, to overcome trajectory tracking problem, trajectory tracking algorithm is proposed based on deep reinforcement learning, using the PPO to train the agent with dynamically feasible randomly generated trajectories. The error of the position and velocity, also the previous action differences, are the goal to minimize to achieve high performance in agile trajectory tracking and generate applicable outputs for real flight practices. The proposed solution is tested in simulation by comparing it with other trajectory tracking algorithms through tracking errors.

For future work, in trajectory replanning part, adaptive increment in the number of control points will be utilized regarding the defined complexities such as obstacle density, if known/estimated. In practice, this evaluation might be obtained through

the algorithm's number of trials while searching for a dynamically feasible trajectory, and the knot number might be increased at certain thresholds. Considering the strong functional relationship between the number of control points and the knot vector's dimension, algorithm will have limited control over the number of knots. Yet, the nonuniform knot vector utilization, enabling to morph the focused region of interest's length, will provide additional authority and robustness to generate a replanned trajectory.

Another natural advancement of the work will be the extension of its implementation to the 3D environments. On the replanning side, the theory behind the proposed methodology is intrinsically generic and independent of the work-space dimension. Obvious requirements will be integrating the control point search over the additional dimensions and applying higher dimensional Euclidian distances. On the implementation side, attitude tracking over the generated trajectory at trustworthy levels will be imperative. This will be further enhance our methodology by supporting with precise trajectory tracking for an extension to 3D environment implementations.

Also, to utilizing the B-spline representation from start point to end point may become restrictive to add additional control points and change their positions. In every relocation of the new control point is adding more action load on the aerial vehicle. For very long distances and very dense clustered environments, increasing number of avoidance action may cause the vehicle to exceed its limits. To solve this issue, in future work, the B-spline trajectory representation will bi utilized not from start point to stop point, but as a receding horizon definition. This will allow to the vehicle to change its agility by having more flexible modification. This also allows to not depend on the past followed trajectory.

Moreover, the algorithm will be extended to use in the environments with dynamic obstacles through approximate motion estimation.

In trajectory tracking approach, to increase the precision of the tracking performance, higher derivatives of the positions will also considered to be used in RL agent. Also, in recent studied shows that the linear rotor drag effects on the vehicle has a significant impact on tracking performance, where this force becomes crucial when the aerial

vehicle is forced to fly in its dynamical limits. So, this effect will also be studied and conduct real flight tests to confirm the performance effect of the solution.

On the other hand, the robustness of the model uncertainties and integrate iterative controller training will be studied through online learn-the-limits and train-the-controller loops. Hence, such a controller will enable us to provide on-policy learning without giving onset limits for agility as the controller will explore the dynamics on the fly.

REFERENCES

[1] **Skyward** (2018). Industry Report: The State of Drones in Big Business, `http://go.skyward.io/rs/902-SIU-382/images/2018%20State%20of%20Drones.pdf`.

[2] **Richter, C.**, **Bry, A. and Roy, N.**, (2016). Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments, Robotics Research, Springer, pp.649–666.

[3] **Sikang Liu**, **Watterson, M.**, **Tang, S. and Kumar, V.** (2016). High speed navigation for quadrotors with limited onboard sensing, *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp.1484–1491.

[4] **Mellinger, D. and Kumar, V.** (2011). Minimum snap trajectory generation and control for quadrotors, *2011 IEEE international conference on robotics and automation*, IEEE, pp.2520–2525.

[5] **Gao, F. and Shen, S.** (2016). Online quadrotor trajectory generation and autonomous navigation on point clouds, *in Safety, Security, and Rescue Robotics (SSRR)*, IEEE International Symposium on., pp.139–146.

[6] **Mehdi, S.B.**, **Choe, R.**, **Cichella, V. and Hovakimyan, N.** (2015). Collision avoidance through path replanning using Bézier curves, *AIAA Guidance, Navigation, and Control Conference*, p.0598.

[7] **Florence, P.**, **Carter, J. and Tedrake, R.** (2016). Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps, *Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

[8] **Lopez, B.T. and How, J.P.** (2017). Aggressive 3-D collision avoidance for high-speed navigation, *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp.5759–5765.

[9] **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*.

[10] **Hwangbo, J.**, **Sa, I.**, **Siegwart, R. and Hutter, M.** (2017). Control of a quadrotor with reinforcement learning, *IEEE Robotics and Automation Letters*, *2*(4), 2096–2103.

[11] **Hasanzade, M. and Koyuncu, E.** (2021). A Dynamically Feasible Fast Replanning Strategy with Deep Reinforcement Learning, *Journal of Intelligent & Robotic Systems*, *101*(1), 1–17.

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

[13] **Van Nieuwstadt, M. and Murray, R.M.** (1996). Real time trajectory generation for differentially flat systems, *IFAC Proceedings Volumes*, *29*(1), 2301–2306.

[14] **Burri, M.**, **Oleynikova, H.**, **Achtelik, M.W. and Siegwart, R.** (2015). Real-time visual-inertial mapping, re-localization and planning onboard MAVs in unknown environments, *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp.1872–1878.

[15] **Hehn, M. and DAndrea, R.** (2011). Quadrocopter trajectory generation and control, *in World Congress*, *18*(1), 1485–1491.

[16] **Charrow, B.**, **Liu, S.**, **Kumar, V. and Michael, N.** (2015). Information-theoretic mapping using cauchy-schwarz quadratic mutual information, *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp.4791–4798.

[17] **Jing Chen**, **Tianbo Liu and Shaojie Shen** (2016). Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments, *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp.1476–1483.

[18] **Usenko, V.**, **von Stumberg, L.**, **Pangercic, A. and Cremers, D.** (2017). Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer, *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, `http://dx.doi.org/10.1109/IROS.2017.8202160`.

[19] **Cichella, V.**, **Choe, R.**, **Mehdi, B.S.**, **Xargay, E.**, **Hovakimyan, N.**, **Trujillo, A.C. and Kaminer, I.** (2014). Trajectory generation and collision avoidance for safe operation of cooperating UAVs, *AIAA Guidance, Navigation, and Control Conference*, p.0972.

[20] **Watterson, M. and Kumar, V.** (2015). Safe receding horizon control for aggressive mav flight with limited range sensing, *in Intelligent Robots and Systems (IROS)*, 2015 IEEE/RSJ International Conference on. IEEE, pp.3235–3240.

[21] **Howard, T.M.**, **Green, C.J.**, **Kelly, A. and Ferguson, D.** (2008). State space sampling of feasible motions for high-performance mobile robot navigation in complex environments, *Journal of Field Robotics*, *25*(6-7), 325–345.

[22] **Mueller, M.W.**, **Hehn, M. and D'Andrea, R.** (2015). A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation, *IEEE Transactions on Robotics*, *31*(6), 1294–1310.

[23] **Green, C.J. and Kelly, A.** (2007). Toward optimal sampling in the space of paths, *In Proceedings of the International Symposium of Robotics Research*, Citeseer.

[24] **Daftry, S.**, **Zeng, S.**, **Khan, A.**, **Dey, D.**, **Melik-Barkhudarov, N.**, **Bagnell, J.A. and Hebert, M.** (2016). Robust Monocular Flight in Cluttered Outdoor Environments, *CoRR*, *abs/1604.04779*.

[25] **Pivtoraiko, M.**, **Mellinger, D. and Kumar, V.** (2013). Incremental micro-UAV motion replanning for exploring unknown environments, *2013 IEEE International Conference on Robotics and Automation*, pp.2452–2458.

[26] **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.

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

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

[29] **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.

[30] **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.

[31] **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.

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

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

[34] **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.

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

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

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

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

[39] **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.

[40] **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.

[41] **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)*.

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

[43] **Rubí, B.**, **Morcego, B. and Pérez, R.** (2020). A Deep Reinforcement Learning Approach for Path Following on a Quadrotor, *2020 European Control Conference (ECC)*, IEEE, pp.1092–1098.

[44] **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*.

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

[46] **Karaman, S. and Frazzoli, E.** (2012). High-speed Flight in an Ergodic Forest, *CoRR*, *abs/1202.0253*, `http://arxiv.org/abs/1202.0253`, `1202.0253`.

[47] **Cox, M.G.** (1972). The numerical evaluation of B-splines, *IMA Journal of Applied Mathematics*, *10*(2), 134–149.

[48] **De Boor, C.** (1972). On calculating with B-splines, *Journal of Approximation theory*, *6*(1), 50–62.

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

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

[51] **Kakade, S. and Langford, J.** (2002). Approximately optimal approximate reinforcement learning, *ICML*, volume 2, pp.267–274.

[52] **Dorn, M.** (1989). Aircraft Agility: The Science and the Opportunities, *Systems and Operations Conference*, AIAA.

[53] **Verbeke, J. and Schutter, J.D.** (2018). Experimental maneuverability and agility quantification for rotary unmanned aerial vehicle, *International Journal of Micro Air Vehicles*, *10*(1), 3–11, https://doi.org/10.1177/1756829317736204, https://doi.org/10.1177/1756829317736204.

[54] **Herbst, W.** (8-10 Mar 1988). Agility, *Briefing Presented at the Workshop on Agility Metrics Held at the AF Flight Test Center*, Edwards AFB.

[55] **Skow, A.** (Jan 1989). Transient Agility Enhancements for Tactical Aircraft, *Eidetics International Report (TR89-001) Prapered Under USAF Contracts F33615-85-C-0120 and F33657-87-C-2045 for ASD/XRM*.

[56] **Skow, A.** (Jan 1989). Innovative Performance and Maneuverability Measures of Merit for Air Combat, *Eidetics International Report (TR-210) USAF Contracts F33615-85-C-0120 for ASD/XRM*.

[57] **Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M. and Radford, Alec... Zhokhov, P.**, (2017), OpenAI Baselines, https://github.com/openai/baselines.

[58] **Karaman, S. and Frazzoli, E.** (2011). Sampling-based algorithms for optimal motion planning, *The International Journal of Robotics Research*, *30*(7), 846–894, https://doi.org/10.1177/0278364911406761, https://doi.org/10.1177/0278364911406761.

[59] **Karaman, S. and Frazzoli, E.** (2010). Optimal kinodynamic motion planning using incremental sampling-based methods, *49th IEEE Conference on Decision and Control (CDC)*, pp.7681–7687.

[60] **Url-1**, <https://www.bitcraze.io/crazyflie-2-1/>, date retrieved 01.06.2019.

[61] **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*, Ieee, pp.2688–2693.

[62] **Url-2**, <http://ardupilot.org/copter/>, date retrieved 05.04.2019.

[63] **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*.

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

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

[66] **Hill, A.**, **Raffin, A.**, **Ernestus, M.**, **Gleave, A.**, **Kanervisto, A. and Traore, Rene... Wu, Y.**, (2018), Stable Baselines, `https://github.com/hill-a/stable-baselines`.

[67] **Fliess, M.**, **Levine, J.**, **Martin, P. and Rouchon, P.**, (January 1994). Flatness and defect of nonlinear systems: Introductory theory and examples, CAS, Tech. Rep. A-284.

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

CURRICULUM VITAE

Name Surname: Mehmet Hasanzade

EDUCATION:

- **B.Sc.:** 2014, Istanbul Technical University, Electrical and Electronics Faculty, Telecommunication Engineering
- **M.Sc.:** 2016, Istanbul Technical University, Electrical and Electronics Faculty, Automation and Control Engineering

Professional Experience:

- Leading a team of 14 engineers, developed agile swarm unmanned aerial vehicle technology - Havelsan
- Managed a team of 4 engineers, developed agile unmanned aerial vehicle technology - STM
- Supervised unmanned aerial systems team with multi-agent flight project - Aselsan
- Led a team of 9 engineers, developed a localization system with multi-agent unmanned aerial vehicles - HUGEM

Grants and Rewards:

- Best Presentation Award - International Conference on Mechanical and Aerospace Engineering (ICMAE)
- Best Presentation Award - Integrated Communications, Navigation and Surveillance Conference (ICNS)
- 25.000 TL grant and a year office award in ITU-Seed incubation - Floradem Startup
- Best solar car design award in 2012 Tubitak-G Solar car racing competition
- Third place award in 2012 Tubitak-G Solar car racing competition

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Hasanzade M.**, Shadeed, O., Koyuncu, E. Deep Reinforcement Learning based Aggressive Collision Avoidance with Limited FOV under Dynamic Constraints, 5th IEEE Conference on Control Technologies and Applications (IEEE CCTA 2021), August 8-11, 2021 San Diego, USA. (submitted)

- Shadeed, O., **Hasanzade M.**, Koyuncu, E. Deep Reinforcement Learning based Aggressive Flight Trajectory Tracker, AIAA Scitech 2021 Forum, January 11-15, 2021 Virtual, USA

- **Hasanzade M.**, Koyuncu, E. A Dynamically Feasible Fast Replanning Strategy with Deep Reinforcement Learning. J Intell Robot Syst 101, 13 (2021). https://doi.org/10.1007/s10846-020-01274-1

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS:

- Herekoglu, Ö., **Hasanzade M.**, Saldiran, E., Cetin, A., Ozgur, I., Kucukoglu, A., Ustun, M. B., Yuksek, B., Yeniceri, R., Koyuncu, E., Inalhan, G., 2019. Flight Testing of a Multiple UAV RF Emission and Vision Based Target Localization Method, AIAA Scitech 2019 Forum, January 7-11, 2019 San Diego, California, USA.

- **Hasanzade M.**, Herekoglu, O., Yeniceri, R., Koyuncu, E., Inalhan, G., 2018. RF Source Localization using Unmanned Aerial Vehicle with Particle Filter, 2018 9th International Conference on Mechanical and Aerospace Engineering (ICMAE), pp. 284–289, July 10-13, 2018 Budapest, Hungary.

- **Hasanzade M.**, Herekoglu, O., Ure, N. K., Koyuncu, E., Yeniceri, R., Inalhan, G., 2017. Localization and tracking of RF emitting targets with multiple unmanned aerial vehicles in large scale environments with uncertain transmitter power, 2017 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1058-1065, June 13-16, 2017 Miami, Florida, USA.

- Yeniçeri, R., **Hasanzade M.**, Koyuncu, E., İnalhan, G. (2017, April). Enabling Centralized UTM services through cellular network for VLL UAVs. In 2017 Integrated Communications, Navigation and Surveillance Conference (ICNS) (pp. 2E1-1). IEEE.

- Tarhan F., **Hasanzade M.**, Çetin A., Biçer Y., Üre N.K., Koyuncu E., Yeniçeri R., İnalhan G., "Kampüs İHA: 4G Şebeke Destekli Kampüs Güvenliği Artırma Projesi," Otomatik Kontrol Ulusal Toplantısı 2017 (TOK 2017) Bildiri Kitabı, sayfa 655-660, İstanbul, 21-23 Eylül 2017. (Turkish)

- **Hasanzade M.**, Herekoğlu Ö., Biçer Y., Üre N.K., Koyuncu E., Yeniçeri R., İnalhan G., "Belirsiz Verici Gücünde RF Sinyal Yayan Kaynakların İnsansız Hava Araçları ile Geniş Ölçekli Ortamda Konumunun Tespiti," Otomatik Kontrol Ulusal Toplantısı 2017 (TOK 2017) Bildiri Kitabı, sayfa 714-719, İstanbul, 21-23 Eylül 2017. (Turkish)

- **Hasanzade M.**, Herekoğlu, Ö., Yeniçeri, R., Koyuncu, E., İnalhan, G., "İnsansız Hava Aracı ve Parçacık Filtresi ile RF Sinyal Kaynağının Lokalizasyonu," 7. Ulusal Havacılık ve Uzay Konferansı (UHUK'2018), Samsun, 12-14 Eylül 2018. Accepted. (Turkish)

- Tarhan, A. F., Koyuncu, E., **Hasanzade M.**, Ozdemir, U., Inalhan, G. (2014, May). Formal intent based flight management system design for unmanned aerial vehicles. In 2014 International Conference on Unmanned Aircraft Systems (ICUAS) (pp. 984-992). IEEE.