BAŞKENT UNIVERSITY

INSTITUTE OF SCIENCE AND ENGINEERING
DEPARTMENT OF DEFENCE TECHNOLOGİES AND SYSTEMS
MASTER OF SCIENCE IN DEFENCE ELECTRONICS AND SOFTWARE

AUTONOMOUS PURSUIT-EVASION SYSTEM FOR MINI UNMANNED AERIAL VEHICLES

BY

ABDULLAH ENES AKDOĞAN

MASTER OF SCIENCE THESIS

ANKARA - 2021

BAŞKENT UNIVERSITY

INSTITUTE OF SCIENCE AND ENGINEERING
DEPARTMENT OF DEFENCE TECHNOLOGİES AND SYSTEMS
MASTER OF SCIENCE IN DEFENCE ELECTRONICS AND SOFTWARE

AUTONOMOUS PURSUIT-EVASION SYSTEM FOR MINI UNMANNED AERIAL VEHICLES

BY

ABDULLAH ENES AKDOĞAN

MASTER OF SCIENCE THESIS

ADVISOR

ASSISTANT PROF. DR. MURAT ÜÇÜNCÜ

ANKARA - 2021

BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING

This study, which was prepared by Abdullah Enes Akdoğan for the program of Defense Electronics and Software, has been approved in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in Defense Technologies and Systems Department by the following committee.

Date of Thesis Defense: 7 / 12 / 2021.

Thesis Title: Autonomous Pursuit Evasion System For Mini Unmanned Aerial Vehicles

Examining Committee Members	**Signature**
Associate Prof. Dr. Muhammed Fatih DEMİRCİ, TOBB University	……..
Assistant Prof. Dr. Emre SÜMER, Başkent University	……..
Assistant Prof. Dr. Murat ÜÇÜNCÜ, Başkent University	……..

APPROVAL

Prof. Dr. Ömer Faruk ELALDI
Director, Institute of Science and Engineering

Date: 7 / 12 / 2021

…….

BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLER ENSTİTÜSÜ

YÜKSEK LİSANS TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: 7 / 12 / 2021

Öğrencinin Adı, Soyadı : Abdullah Enes Akdoğan

Öğrencinin Numarası : 21810264

Anabilim Dalı : Savunma Teknolojileri ve Sistemleri

Programı : Savunma Elektroniği ve Yazılımı

Danışmanın Unvanı/Adı, Soyadı : Dr. Öğretim Üyesi Murat Üçüncü

Tez Başlığı : Autonomous Pursuit-Evasion System For Mini Unmanned Aerial Vehicles

Yukarıda başlığı belirtilen Yüksek Lisans/Doktora tez çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam 83 sayfalık kısmına ilişkin, 25 / 11 / 2021 tarihinde tez danışmanım tarafından Turnitin adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı %1'dir. Uygulanan filtrelemeler:

1. Kaynakça hariç
2. Alıntılar hariç
3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası:

Abdullah Enes Akdoğan

…….

ONAY

Tarih: 7 /12 /2021

Öğrenci Danışmanı

Dr. Öğr. Üyesi Murat ÜÇÜNCÜ

…….

To all brothers and sisters who contribute to the independence of the Turkish Nation…

Abdullah Enes Akdoğan Ankara-2021

ABSTRACT

Abdullah Enes Akdoğan

AUTONOMOUS PURSUIT-EVASION SYSTEM FOR MINI UNMANNED AERIAL VEHICLES

Başkent University Institute of Science and Technology

The Department of Defence Technologies and Systems

2021

As a result of the widespread use of unmanned aerial vehicles, UAVs are actively used in many different missions in both civilian and military fields. It is anticipated that UAVs, which are primarily used for reconnaissance and surveillance, homeland security, and close air support at the tactical and strategic level in the military field, can also be used in air-to-air combats in the near future. Therefore, UAVs with technological equipment that can engage with other UAVs will be in an advantageous position. The ability to perform air-to-air combats autonomously will provide advantages in terms of both performance and time-criticality. In terms of Mini or Small UAVs, air-to-air combats are mainly focused on evasion and pursuit missions.

There are several pieces of research about path planning missions with avoidance zones in literature. There are also several research about pursuit-evasion problems. However, there is no research that combines the two cases to the best of our knowledge. In this thesis, the problem of path planning in avoidance zones and the problem of pursuit-evasion missions are merged, and a solution is proposed for the integrated problem.

Mission area is divided into green, gray, and red areas according to the level of danger. In this thesis, a mini unmanned aerial vehicle software that can autonomously perform escape and pursuit missions in different types of zones has been developed. Mission Control Software uses Reinforcement Learning and Q Learning to perform autonomous missions.

Keywords: Mini UAV, Air-to-air combat, Avoidance Zone, Danger Zone, Autonomous UAV, Pursuit Evasion System for UAV, Reinforcement Learning, Q Learning

Supervisor: Assistant Prof. Dr. Murat Üçüncü, Başkent University

ÖZET

Abdullah Enes Akdoğan

MİNİ İNSANSIZ HAVA ARAÇLARI İÇİN OTONOM KAÇIŞ TAKİP SİSTEMİ

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Savunma Teknolojileri ve Sistemleri Anabilim Dalı

2021

İnsansız Hava Araçlarının hızla artan yaygın kullanımı sonucunda İHA'lar hem sivil hem askeri alanlarda birçok farklı görevde aktif olarak kullanılmaktadır. Askeri alanda taktik ve stratejik düzeyde temel olarak keşif gözetleme, anayurt güvenliği ve yakın hava desteği amacıyla kullanılan İnsansız Hava Araçlarının yakın gelecekte hava-hava muharebelerinde de kullanılabileceği öngörülmektedir. Dolayısıyla diğer İHA'lar ile angajmana girebilecek teknolojik donanıma sahip İHA'lar avantajlı konumda olacaklardır. Otonom faaliyet gösteren İHA'ların hava-hava muharebelerini de otonom olarak gerçekleştirebilmesi hem performans hem de zaman kritiklik konusunda avantaj sağlayacaktır. Hava-hava muharebeleri döner pervaneli İHA'lar ile temel olarak kaçış ve takip senaryoları odaklı olmaktadır.

Literatürde yasaklı bölgeler gözetilerek rota planlama problemi ve kaçış takip problemleri ayrı ayrı olarak yer almaktadır. Fakat, bu iki problemi bütünleştiren bir uygulama çalışılmamıştır. Bu tez kapsamında rota planlama ve kaçış takip olarak her iki uygulama bütünleşik olarak ele alınmıştır.

Alanlar tehlike düzeyine göre yeşil, gri ve kırmızı alanlar olmak üzere 3 gruba ayrılmıştır. Rotadaki alanların tehlike düzeylerini de dikkate alarak kaçış takip görevlerini otonom olarak gerçekleştirebilen bir mini insansız hava aracı yazılımı geliştirilmiştir. Görevlerin otonom olarak gerçekleştirilmesinde Pekiştirmeli Öğrenme ve Q Öğrenme kullanılmıştır.

Anahtar Kelimeler: Mini İHA, Hava-hava muharebe, Otonom İHA, Kaçış Takip, Pekiştirmeli Öğrenme, Q Öğrenme

Danışman: Dr. Öğretim Üyesi Murat Üçüncü, Başkent Üniversitesi

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ADP	Approximate dynamic programming
AESA	Active electronically scanned array
AI	Artificial intelligence
AMRAAM	Advanced medium-range air-to-air missile
BLDC	Brushless direct current electric motor
BLOS	Beyond line of sight
BVR	Beyond visual range
DEAD	Destruction of enemy air defenses
DP	Dynamic programming
DQN	Deep Q learning
FCC	Flight control card
GNSS	Global navigation satellite system
GUI	Graphical user interface
IMU	Inertial measurement unit
ISR	Intelligence, surveillance, and reconnaissance
JSON	Java script object notation
LIDAR	Light detection and ranging
MDP	Markov decision problem
PCB	Printed circuit board
RL	Reinforcement learning
RPA	Remotely piloted aircraft
SEAD	Suppression of enemy air defenses
SSB	Presidency of defense industries
TAI	Turkish aerospace industries
TCP	Transmission control protocol
UAV	Unmanned aerial vehicle
UCAV	Unmanned combat aerial vehicle
UDP	User datagram protocol

1 INTRODUCTION

UAVs (Unmanned Aerial Vehicles) are currently being used in many missions, such as ISR, Target Practicing, Electronic Warfare, and Attack missions. However, Air superiority and air combat missions are still performed by manned aircraft. However, unmanned platforms may replace manned platforms in air combats missions in the future. In this thesis, UAV's autonomous algorithms for pursuit-evasion problems are analyzed as the main problem of air combats missions with UAVs. To perform air combats missions, two generic problems, path planning in areas including avoidance zones and pursuit-evasion problems, are merged as an integrated problem. This thesis aims to develop autonomous software for Mini UAVs to perform pursuit-evasion missions with different zones such as green, red, and gray zones.

The thesis consists of 5 Chapters, Introduction, Simulation Environment, Software Development, Simulation Studies, and Conclusion.

In the Introduction chapter, there is a short introduction to UAV history, classification, and field of usages of unmanned aerial vehicles. In addition to these backgrounds, air-to-air combat and the associated problem definitions for this thesis are covered.

The simulation Environment chapter is the second chapter, and it gives a brief overview of Turumtay, which is the simulation platform of this thesis. In this chapter, there is essential information about Turumtay, system architecture, and software architecture of Turumtay. Ground Control, Mission Control, Flight Control Software, and their communication structures are presented.

Methodology and algorithms for Software Development are described in the third chapter. The chapter explains Q Learning and Reinforcement Learning algorithms Software Design together with the application in Mission Control Software.

The fourth chapter analyses the Simulation Studies. This chapter also discusses the performance of the algorithm in Patrolling Missions, Evasion Missions, and Pursuit Missions.

The results and evaluation chapter summarizes the results of the algorithms and interprets these outcomes according to gained rewards in 3 different mission types.

In the Conclusion chapter, the result of the study is discussed. Furthermore, possible future works are proposed in this section.

1.1 Unmanned Aerial Vehicles

1.1.1 Short history of UAVs

Contrary to popular belief, the history of UAVs goes back to the beginning of the 20th century. UAV's first can be seen during World War I. Simple unmanned systems such as Kettering Bug were used in World War I. They were like torpedoes or missiles with basic wings. They were remotely controlled by radio frequencies and could make basic maneuvers. UAVs were also used in World War II by Germans to bombard English lands. Buzzbomb was one of the popular UAVs used in World War II. The history of UAVs is examined in two chapters as Early Years (1916- 1958) and Modern Era (1959-Present) [1].

Figure 1.1 UAV used in World War I – Kettering Bug

In the modern era of UAVs, countries started to use UAVs for different purposes. In addition to simple attack missions, UAVs were also used with cameras to perform surveillance missions as well as target/decoy missions. Vietnam War is a milestone for the history of UAVs. The USA used UAVs in Vietnam effectively in strategic missions such as reconnaissance, damage assessments, and intelligence [1]. After the Vietnam Conflict, UAVs continued to take critical roles in combats. In 1982, Israel deceived Lebanese air defense systems by using UAVs as decoys. Lebanese defense systems are locked on decoys, so missiles of Israel eliminate Lebanese targets. Influential usage of UAVs on critical missions has been continuing, and UAVs took part in many conflicts or wars such as Gulf War, Afghanistan, and Israel - Lebanon Conflict in 2006.

In terms of Turkey, UAV history began in the last decade of the 20th century when Turkey preferred direct procurement for UAV acquisition [2]. In 1989, Turkey purchased Banshee Target UAV, and it is the first drone used by the Turkish Armed Forces [3]. GNAT and I-GNAT came after Banshee, and the Turkish Armed Forces have used them by 2005. At the beginning of the 2000s, while direct procurements continued, Turkey also initiated development programs for nationally developed UAVs [2]. As a milestone in Turkish UAV History in 2004, Turkish Undersecretaries for Defense Industries initiated the ANKA project [2]. TAI (Turkish Aerospace Industries) was the prime contractor in this project. ANKA completed its first flight in 2010, and it has been in use by the Turkish Army since 2015. In addition to the ANKA project, Turna, Keklik, Baykus, Pelikan, Marti UAV projects were also developed by TAI. Vestel Defence also developed Arı, Efe, and Karayel UAVs. Karayel UAV can carry bombs and mini-missiles [2]. Baykar is one of the vital game-changer companies of Turkey in terms of UAV development. Baykar first developed Bayraktar Mini UAV. Bayraktar has been used by Turkish Land Forces since 2007 and exceeds 100000 flight hours [2]. In addition to Bayraktar Mini, Baykar also developed Bayraktar TB-2 Tactical UAV, and it is actively used by Turkey in combat zones [4], [5]. Baykar Defence has started to develop AKINCI UAV in 2018. Now, AKINCI completed its first flight in

2019 [6]. Akıncı UAV has the capacity to carry and use air-to-air missiles such as Bozdoğan and Gökdoğan [6].

1.1.2 Classification of UAVs

There are many types of UAVs in terms of their sizes, weight, and mission. The classification of UAVs is essential to understand the mission types of UAVs and their usage areas. UAVs can be classified by different characteristics; NATO classifies UAVs using their flight weights [7]. So, there are 3 UAV classes which are Class 1, Class II, and Class III. Class I is the smallest class, and UAVs lighter than 150 kg, are in this class. Class I is also composed of 3 categories by Micro (< 2 kg), Mini (< 20kg), and Small (< 150kg). Class II is the middle class, and tactical UAVs are under this class, and their weight limit is 600 kg. Class III is the last class, and UAVs heavier than 600 kg belong to this class.

Table 1.1 UAV Classification

Class	Category	Operational Altitude	Mission Radius	Sample Platform
Class I (< 150 kg)	Micro (< 2kg)	200 ft	5 km	Black Hornet
	Mini (< 20kg)	3 000 ft	25 km	Bayraktar Mini IHA
	Small (< 150kg)	5 000 ft	50 km	Hermes 90
Class II (< 600 kg)	Tactical	20 000 ft	200 km	Bayraktar TB2
Class III	MALE	45000 ft	BLOS	ANKA
	HALE	65000 ft	BLOS	Akıncı
	Strike/Combat	65000 ft	BLOS	MQ9-Reaper

1.1.2.1 Micro UAV

Micro UAVs are the smallest members of the UAV Family. They are usually controlled by one soldier and can be controlled by a small tablet or mobile device. Micro UAVs are systems for Personal Reconnaissance, and their primary purpose is to gain real-time information about hostile elements or building plans for possible hostage rescue

missions. They are lighter than 2 kilograms, and they usually operate under 200 ft. Their missions are interested in close-distance operations, so their mission range usually are smaller than 5 kilometers. FLIR Black Hornet is one of the most popular micro UAVs. Many countries use Black Hornets in special operations [8].

Figure 1.2 Black Hornet Micro UAV

1.1.2.2 Mini UAV

Small soldier groups or tactical units use mini UAVs. UAV Operator launches the system with manual launch or hand throw launch. Mini UAV's operational altitude can reach up to 3000 ft and 25 kilometers operational mission range. The weight of these UAVs is generally between 2 kg and 15 kg. Mini UAVs are also a member of the Class I UAV category as Micro UAVs. "Bayraktar Mini IHA" is the most famous Mini UAV in Turkey. It has been actively used by the Turkish Armed Forces since 2007 and reached up to 100,000 flight hours [9]. Bayraktar Mini UAV has 60-80-minute endurance time, 15 km communication range capability, and weights 10 kg [10].

Figure 1.3 Bayraktar Mini UAV

1.1.2.3 Small UAV

The third member of Class I UAVs is the Small UAV family. The Small UAVs have the capability of automatic take-off and landing. They also support manual launches. Their operational altitude is approximately 5000 ft, and their communication radius range is around 50-kilometers. The take-off weight of Small UAVs is larger than 20 kilograms. Hermes-90 is a typical Small UAV, and its flight weight is 115 kilograms, and it can carry payloads up to 25 kg. Hermes-90 is produced by Elbit systems and made its first flight in 2009 [11].

Figure 1.4 Hermes-90 Small UAV

1.1.2.4 Tactical UAV

The only member of Class II is the Tactical UAVs. Land Forces are using tactical UAVs, and they provide Intelligence, Surveillance, and Reconnaissance in operation fields. Soldiers take action according to received ISR from UAVs. Tactical UAVs use peer-to-peer communication data links, and their range reaches up to 200 km in Line of Sight. The operational altitude level of Tactical UAVs can reach up to 20000 ft. The Turkish Armed Forces have been using Bayraktar TB2 since 2014 [4]. Bayraktar TB2 played crucial roles in many operations of the Turkish Armed Forces, such as Operation Euphrates Shield, Operation Olive Branch, and Operation Peace Spring [12]. Bayraktar TB2 is a Combat Proven Armed UAV that can carry four smart munitions and has proved its capabilities in the combat zone [5].

Figure 1.5 Bayraktar TB2

1.1.2.5 MALE UAV

Male UAVs take place in the category of Class III according to NATO UAS Classification [13]. Their operational altitude can reach up to 45000 ft. The weight of these class of UAVs is more than 600 kilograms, so they have the capability of carrying multiple munitions and missiles. Like Tactical UAVs, Male UAVs also use LOS Data Link with approximately 200-250 km range. Many UAVs in the Male UAV Category also have satellite data links. Thanks to satellite links, they have no mission range limits. UAVs of this category are effectively used in strategic missions such as Airborne ISR Collection. ANKA, which TAI developed, is used by the Turkish Armed Forces [14]. ANKA has variations such as ANKA-I, ANKA-S which are separated from each other by their specifications for their missions. ANKA equipped with Electronic Warfare subsystems can be used in Signal and Electronic Intelligence missions.

Figure 1.6 ANKA

1.1.2.6 HALE UAV

High Altitude Long Endurance UAVs can take charge in a highly compelling mission such as Strategic Attacks, Electronic Warfare, and SEAD/DEAD. These UAVs can operate in high altitudes up to 65000 ft. In addition to their operational altitude capability, their payload capacity is higher than the MALE category, and the Payload capacity of HALE UAVs is more than 1000-1500 kilograms. Baykar Defense have been developing Akıncı UAV since 2018. According to Baykar, Akıncı will be more sophisticated version of Bayraktar TB2. Akıncı has two turboprop engines and can carry critical subsystems such as air-to-air missiles and AESA radars [6].

Figure 1.7 Akıncı UAV

1.1.2.7 Strike/Combat UAV

Strike or Combat UAVs are similar to HALE UAVs, and their specifications differ from others with stealth technologies and other advanced avionics. Air and Missile Defense, Counter air missions are suitable for Combat UAVs. Survivability and stealth are critical technologies in UCAVs (Unmanned Combat Aerial Vehicles).

1.1.3 Fields of usage of UAVs

UAV's can be categorized into four main categories, which are ISR, Target Practice, Electronic Warfare, and Attack as shown in Figure 1.8. These main categories also have sub-categories. ISR missions are conducted as Tactical Reconnaissance and Strategical ISR. Target and Decoy usages are two types of Target Practicing. Electronic Warfare is analyzed

under three categories: Electronic Attack, Electronic Protection, and Electronic Warfare Support. Attack missions, which is the fourth main category, cover Homeland Security, Close Air Support, SEAD/DEAD, and Air Superiority.

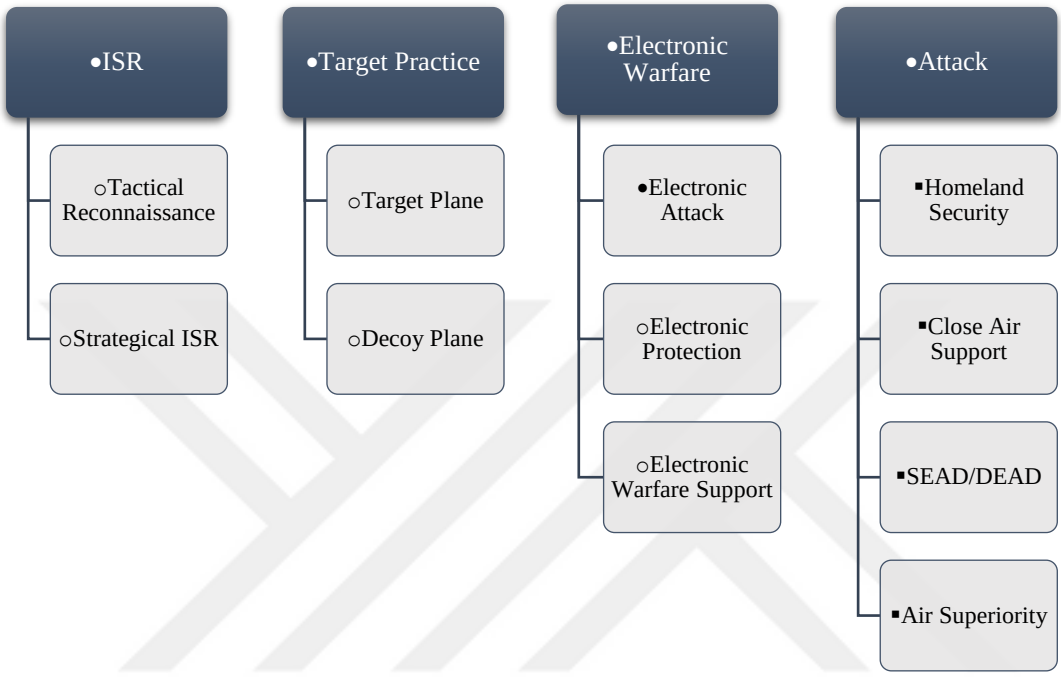

Figure 1.8 Fields of Usages of UAVs

1.1.3.1 ISR

UAVs are primarily used for ISR missions. In order to acquire intelligence and increase awareness about enemy forces or territories, UAVs are used by armed forces. ISR missions are categorized concerning their mission types, such as tactical or strategical.

1.1.3.1.1 Tactical Reconnaissance

Micro or Mini UAVs are generally used in Tactical Reconnaissance missions. The primary purpose of this mission is to acquire real-time intelligence on the operation terrain, such as the positions of enemy forces. Small tactical teams use UAVs to gain an advantage over the enemy, thanks to situational awareness.

1.1.3.1.2 Strategical ISR

Since the first wars, every armed force has tried to gather information about the enemy. Having the knowledge of positions of strategical targets of enemy or knowledge of equipment inventory of enemy soldiers is crucial in preparing for wars. In modern times, countries also use UAVs for this purpose. The critical difference between tactical and strategical reconnaissance is that tactical reconnaissance aims at short-term intelligence, such as positions of enemy soldiers who are preparing for an ambush. On the other hand, in strategical ISR, countries aim to acquire intelligence that helps long-term purposes such as exact positions of enemy air defense systems, power plants, nuclear facilities.

1.1.3.2 Target Practice

In order to imitate real planes or create a simulation environment for testing aerial vehicles, UAVs have been used. UAV's cost-efficiency and no risk of human loss make UAVs advantageous in target practice missions. This usage can be grouped into two categories as Target Planes and Decoy Planes. The main difference between these two types is their operational concepts. Target planes are used for training and testing ally weapons or systems. However, decoy planes are used to mislead enemy systems.

1.1.3.2.1 Target Plane

While developing or testing particular systems and technologies, target planes or target drones are used. For instance, to train air defense systems or radar systems, target planes are being used. Target planes are generally used to imitate real targets with lower costs. Their similarity to real air fighters and their low cost can be used in training or tests.

1.1.3.2.2 Decoy Plane

Like target planes, decoy planes also imitate real targets, but the difference is that target planes are used to train ally weapons or soldiers. On the other hand, the purpose of decoy planes is to deceive enemy forces such as enemy air defense systems. Decoy planes deceive air defense systems, so air defense systems lock on these decoys, and after that, real missiles hit air defense systems. 1982 Lebanese War is an example of using decoy planes in

operations. In 1982, Israel successfully used decoy planes to make Lebanese Air Defense Systems blind and eliminate these targets [1].

1.1.3.3 Electronic Warfare

Electronic Warfare is a set of actions that affect the electromagnetic spectrum, including jamming, detections, and signal intelligence. Electronic warfare missions are divided into three subcategories: Electronic Attack, Electronic Protection, and Electronic Warfare Support [15].

1.1.3.3.1 Electronic Attack

In Electronic Attack missions, electromagnetic energy is used to attack enemies in different ways, such as jamming and deceptions. These missions aim to destroy or neutralize enemy electronic systems such as air defense systems, radars, or communication systems. By using sophisticated EA techniques, enemy radars and sensors can be neutralized, and blind, so friendly forces can operate around these air defense systems without being detected [15]. UAVs used in these missions should have low observability capabilities due to the high potential of enemy engagements.

1.1.3.3.2 Electronic Protection

Protecting friendly forces from enemy attacks, which affect the electromagnetic spectrum, is highly critical. The capability of electronic protection is the capability of defeating enemy electronic attacks. Electronic protection systems can deceive enemy sensors, radars and manipulate them to show friendly forces in different locations. The other example is that using UAVs as decoy planes and deceiving enemy radar-guided anti-air missiles. These decoy planes are equipped with specific electronic protection payloads and transmit signals similar to real friendly aircraft and incoming enemy air-missile targets decoy planes rather than real aircraft [15].

1.1.3.3.3 Electronic Warfare Support

In order to perform electronic attacks or protection for enemy electronic attacks, the characteristics of enemy systems and their electromagnetic spectrum usage should be well known. So, electromagnetic warfare support missions are performed to detect, identify and

locate enemy systems [15]. Signal Intelligence, Electronic Intelligence, and Communication Intelligence are examined in Electronic Warfare Support. UAVs that have low observability and high operational altitude have been used in these missions.

1.1.3.4 Attack

Attack missions are carried out by Combat UAVs, which can carry missile and nutrition bombs. These missions can be grouped into four categories as Homeland Security, Close Air Support, SEAD/DEAD, and Air Superiority.

1.1.3.4.1 Homeland Security

Acquired imagery intelligence with tactical or strategical ISR are analyzed, and possible actions are taken by UAVs accordingly. For example, eliminating terrorist attacks detected by UAVs reconnaissance missions is one of the popular usages of UAVs by countries to assure their homeland security.

1.1.3.4.2 Close Air Support

UAVs can perform close Air Support missions in coordination with ISR information. In these missions, UAVs are used to suppress enemy targets in the area. In order to suppress enemy targets, UAVs used in Close Air Support can carry different types of nutrition and missiles. Enemy tanks or armored personnel carriers can be damaged by Anti-Tank missiles that are launched by UAVs.

1.1.3.4.3 SEAD/DEAD

The capability of performing SEAD (Suppression of Enemy Air Defenses) and DEAD (Destruction of Enemy Air Defenses) missions makes UAVs more promising in combat areas. Potential risks of suppression or destruction of enemy air defense systems make these missions more dangerous for manned aerial vehicles. So, when it is possible to use UAVs, UAVs would be favored rather than manned platforms. Because of this mission's nature, UAVs in SEAD/DEAD operations should have many cutting-edge technologies, and these UAVs must be capable of high speed, high operational altitude, quick maneuverability, and small radar cross-section. In addition to these survivability capabilities, having sophisticated sensors which can detect the exact location of enemy air defense systems is highly critical.

1.1.3.4.4 Air Superiority

Most of the usage fields of UAVs are about asymmetrical warfare, which occurred between regular armies and guerillas or terrorist organizations. Conversely, air superiority missions are the subject of states, and the purpose of these operations is to ensure superiority over enemy or rival states in the air.

Air superiority missions are one of the most compelling aerial missions. In these missions, there are lots of crucial risks and dangers. Similar to the requirements of SEAD/DEAD missions, high speed, maneuverability, and stealth are also crucial in air superiority missions. Moreover, to handle air superiority missions, aerial vehicles need many advanced sensors and weapon systems, including air-to-air missiles and AESA (Active electronically scanned array) radars.

Unmanned Aerial Vehicles can perform various offensive counterair missions such as forcing the enemy to retreat, destroying enemy targets with air-to-air missiles, or disrupting their functions through electronic warfare. In addition to these offensive missions, UAVs can also perform defensive counterair by detecting and neutralizing enemy targets attempting to penetrate friendly zones.

1.2 Air-to-Air Combats

Air Superiority conflicts are mainly based on air-to-air combats and counter-air missions. Air-to-air combats are called dogfights in the literature; techniques used in dogfights were changed in time. Dogfights were started by pistol, rifles and continued with machine guns. After that, air-to-air missiles took place, and developments in the AAM (Air-to-Air Missile) continued. As shown in Figure 1.9, hit types of air-to-air kills have changed significantly over time, and the ratio of AAM increases year by year until the 1990s. After 1990, almost every kill was conducted by AAM, especially with BVR (Beyond Visual Range) AAMs.

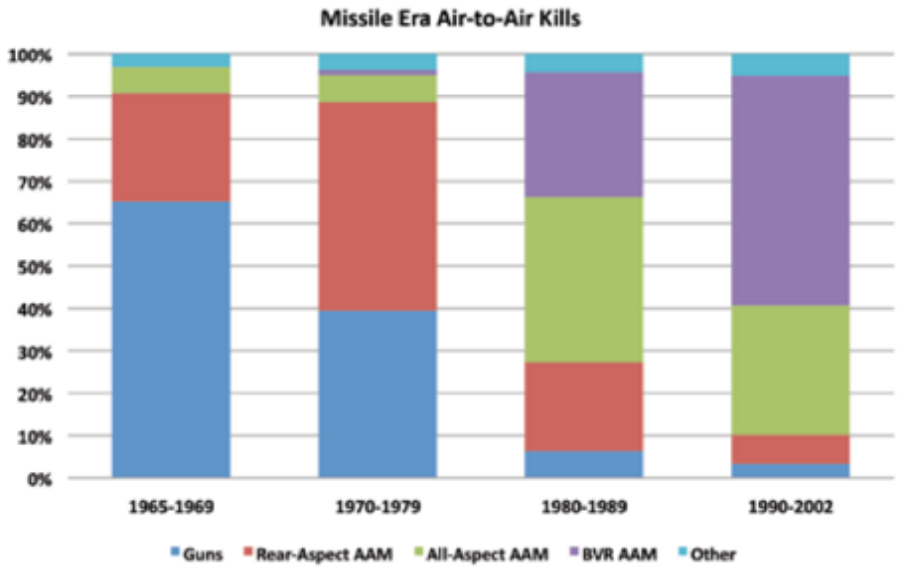

Figure 1.9 Types of Air-to-Air kills (1965-2002) [16]

1.2.1 Development Process of air-to-air combats

The history of Air Superiority and so air-to-air combats goes back a long way; the first air-to-air combat took place in 1913 during Mexican Revolution. According to Dean Ivan Lamb, one of two pilots in this combat, pistols were the primary weapon in the first air-to-air combat of history [17]. During World War I, airplanes were used for aerial reconnaissance and essential observation. Those airplanes performed missions in short-range around combat areas. So, confronting the enemy airplane is not a rare situation during intensive times. Although there were no serial casualties in these confrontations due to the lack of aerial combat techniques and weapons, pilots were using rifles, pistols, and even shotguns, so even if they hit enemy airplanes successfully in short-range, these weapons could not damage the airplane significantly [16].

Usage of machine guns comes after rifles and pistols, and airplanes equipped with machine guns performed many air-to-air combat missions during World War II. However, machine guns also could not generally cause critical damages on enemy airplanes unless the engine or other critical components are hit or damaged [16]. In addition to the low efficiency of machine guns, they also reduce the maneuvering capabilities of pilots and make flights harder to manage. With World War II, Air-to-air missiles were designed, firstly by Germans.

After that United States, the Soviet Union, and England also developed their air-to-air missiles. However, these first missiles could not meet the expectations due to their lack of guiding techniques.

After World War II, working on the development of advanced air-to-air missiles is continued, and these developments were tested in many combats such as the Vietnamese Wars, 1967 Six-Day Wars, and Iran-Iraq War. Afterward, with the developments in the sophisticated air-to-air missiles and radars, air-to-air combats are changed compared to the first era of combats. These advanced missiles and sensors determine the winner of combats.

1.2.2 Critical sub-components used in aerial combats

In early eras of aerial combats, pilots managed all parts of the mission, and they were doing sudden maneuvers while controlling machine guns simultaneously. While they were fighting, they must also monitor their surrounding areas with their eyes to detect potential enemies by acting like advanced radars. They also must control the aim with rifles or machine guns by acting as a complicated fire control system. There were other responsibilities of pilots, which are currently done by sensors or other sub-components.

Without using radars, pilots used their eyes to detect potential threats and identify other airplanes' friendly or enemy. As of today, pilots are using radars instead of their eyes and can detect enemies accurately beyond visual ranges with no extra effort. AESA (Active electronically scanned array) is used in airplanes to detect and track enemy airplanes beyond visual range even after 50 nautical miles [18]. “Detect enemy before enemy detects you” brings significant advantages for pilots thanks to these cutting-edge radar technologies. Long-range sensors, especially which have the capability to detect beyond vision range, are crucial in aerial combats in terms of situational awareness.

Air-to-air Missiles such as AMRAAM, Meteor, or Gökdoğan have been developed in order to destroy enemy aircraft before detected, with agile movement capabilities to track even quick maneuvers of aircraft. Air-to-air missiles have subgroups in terms of their operational ranges, which are within visual range or beyond visual range. Aircrafts equipped with missiles that have BVR capability can engage enemy aircraft from far distances.

Information denial is substantial as information acquisition in aerial combats. In addition to these sensors providing situational awareness and missiles providing first hit chances, stealth and low observability are also crucial for survivability. Stealth capabilities reduce radar cross-sections, making it challenging to detect enemy radars which increase information denial abilities and improve survivability.

1.2.3 Advantages and disadvantages of using UAV in air-to-air combats

Using UAVs in air combats has been discussed for many years, there are numerous advantages and disadvantages of this field of usage.

Fu et al. [19] imply that the technical incapability of current UAV technology is the biggest obstacle for UAVs to perform air combat missions. Fu examined two modes of UAV; one remotely controlled by ground control station and the second one is controlled by the computer inside the UAV. According to Fu, neither of the two modes is unable to perform necessary actions in combat because of the jammer threats or other communication problems in the first mode and insufficient technology to quickly identify targets and perform flexible maneuvers in the second mode [19]. Furthermore, questioning about technological capabilities of autonomous UAVs, ethics, and legal issues are also significant problems for using UAVs in combats. Pashakhanlou et al. [20] Clearly stated these ethical and legal issues in their article. Responsibility and accountability of operations that UAVs have done are one of the issues in that article. If anything goes wrong in the mission, who will be responsible for that mistake, the operator of the UAV or the software engineer of the algorithms running on the UAV, the producer of the UAV? The list of the potentially responsible bodies is not limited to these. Pashakhanlou also indicates that other critics about UAVs are not covered by international law currently [20].

Besides these concerns, there are many arguments for the benefits of using UAVs in air combats. Economic reasons are one of these advantages, according to Pashakhanlou [20]. Educating pilots for fighter jets is a costly process. To educate a fighter pilot, five years are needed, and the cost of this education is between $3 and $11 million approximately [20]. The most important risk of using pilots in air combats is the high potential of loss of pilot life. Instead, using of UAV drastically reduces the risk of loss of the life of UAV compared

to pilots of fighter jets. Pilots have natural needs such as hunger and fatigue, which put some operational limits on time for air combat missions. In UAVs, these limits are not valid [20]. Finally, UAVs and computers on it have sophisticated software technologies, the time of the reaction to threats is crucial in air combats, and this reaction time is very low in machine pilots with comparison to human pilots [20]. Machine pilots can acquire, integrate and fuse incoming information from different sensors more quickly and reliably than human pilots nearly in real-time [21].

1.3 Problem Definition

This thesis analyzes the pursuit and evasion problem in UAV operation, and software is developed according to scenarios. The UAV operation area is divided into green, gray, and red zones. The Mission Control Software developed to guide the UAV autonomously runs on a platform called “Turumtay”. The software is responsible for pursuit and evasion missions while considering the safety of green and red zones. The problem focuses on one-to-one pursuit-evasion combat between Turumtay and Target UAV.

1.3.1 Agents

1.3.1.1 Turumtay

Turumtay is responsible for three main missions: patrolling, escaping to the green zone, and pursuing a UAV. Turumtay performs its mission through software having a layered architecture. These layers are Ground Control, Mission Control, and Flight Control software, and the main component responsible for performing these missions is Mission Control.

1.3.1.2 Target UAV

Target UAV is the other UAV in the environment. This UAV is the opponent of Turumtay. Turumtay and Target UAV pursuit each other or escape from each other’s fields of view in these combat missions. Target UAV runs greedy algorithms while performing missions.

1.3.2 Zones

There are three zones in our environment; Green, Gray, and Red zones, depicted in Figure 1.10. Gray zones are the rest of the points, which are neither green nor red zones. These green, gray and red zones symbolize ally, neutral, and enemy zones.

Zones are 2D shapes like polygons, and a set of points describes them. Sun [22] and You [23] represented the zones by using ellipses. However, in this thesis, zones are represented by polygons like military zones. The advantage of using polygons is that there can be convex and concave shapes which help determine the optimal path for the UAV to follow.

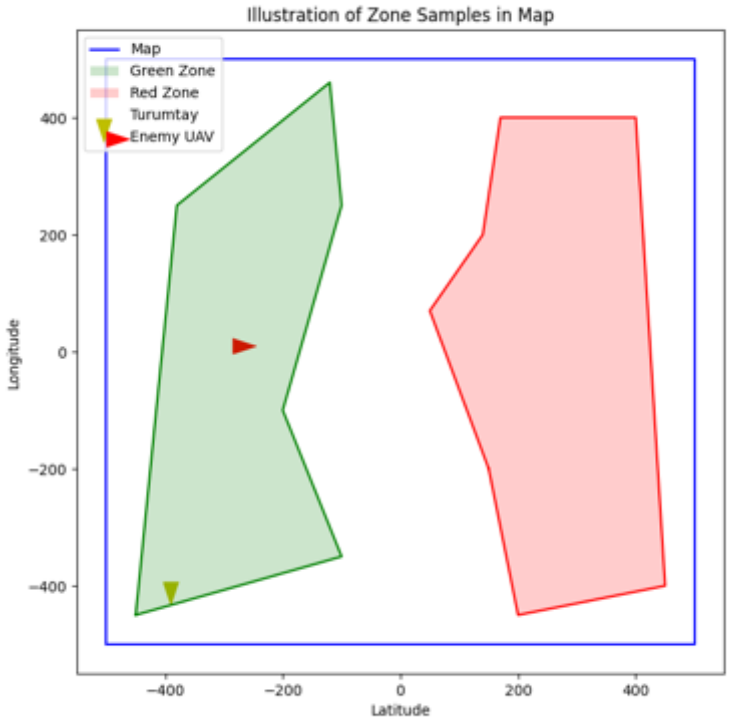

Figure 1.10 Illustration of Zone Samples in Map

1.3.2.1 Green Zone

Green zones are safe for flight. There are no known enemy threats, and the zone is out of range of enemy missiles. Ally forces control these zones, and it symbolizes ally-safe zones.

UAV's fly in green zones rather than gray or red zones due to safety concerns.

1.3.2.2 Red Zone

There are dangerous enemy threats for safe flight in red zones. Moreover, there may be jamming or some other electronic warfare threats. For instance, Yan and Xiang [24] use Radars to express enemy threat zones. In addition to these soft-kill measures, there may also be air defense systems in red zones. The characteristic of these threats falls outside the scope of this thesis. In this thesis, zones are defined by their boundaries, and the coverage is not of concern. Red zones are expressed as 'avoidance zone' [23], [22], or 'threat zone' [25] in some research. There is no significant difference between these expressions; red zones, avoidance zones, and threat zones imply that UAV should not enter these areas in its flight.

1.3.2.3 Gray Zone

Gray zones are controlled by neither ally nor enemy forces.

1.3.3 Missions

1.3.3.1 Patrolling Mission

Patrolling is a usual mission type for aircraft. In these missions, the Aircraft goes between two points in order to detect anomalies in these regions. Patrolling missions are done in gray zones, so aircraft must stay in green or gray zones and not enter red zones.

1.3.3.2 Escape to Green Zone Mission

While escaping to the green zone, UAV should not enter red zones and escape from Enemy Aspect Angle. If Turumtay is in the Field of View of Enemy, it is a potential danger.

Therefore, Red Zones and Enemy Aspect Angles are considered to be critical dangers while escaping.

1.3.3.3 Pursuit Enemy UAV Mission

In Pursuit Mission, Turumtay prefers green zones or gray zones if possible and does not enter red zones. Furthermore, Turumtay tries to lock on Enemy UAVs to keep Enemy UAVs in the Field of View of Turumtay.

1.4 Literature Review

Presidency of Defence Industries, which is affiliated with the Presidency of The Republic of Turkey, prepared a road map for unmanned aerial vehicles in the name of "Türkiye İnsansız Hava Aracı Sistemleri Yol Haritası 2011-2030" in 2012 [7]. This document is the official road map document of Turkey about UAVs. SSB (Presidency of Defence Industries) explained UAV Classifications and Fields of Usages from the point of Turkey. Capabilities of Turkish Defence Companies in terms of UAV Technologies are expounded in this document.

The United States Air Force released the "RPA Vector Vision and Enabling Concepts 2013 – 2038" document in 2014, which supersedes the previous strategical plan document of "USAF Unmanned Aircraft Systems Flight Plan 2009–2047" [26]. In this document, RPA's (Remotely Piloted Aircraft) are analyzed in all aspects. The document examines the usage of RPAs in current operations and also possible future concepts in detail. This document is the official vision of the US Air Force in terms of UAVs and RPAs.

Huy X. Pham, Hung M. La, David Feil-Seifer, and Luan V. Nguyen [27] researched the UAV Navigation system and proposed a technique for solving this navigation problem by using Reinforcement Learning. Pham et al. Transform the continuous space into discrete finite sets by dividing the space into large spheres. So, points represent the center of these spheres, and Q Learning can be implemented with these points. As a result of the algorithm training, UAVs as an agent can successfully reach the goal point by using optimal paths without any environmental knowledge at the start of the flying.

James McGrew et al. [28] analyzed the Strategy of Air Combat by using Approximate Dynamic Programming. McGrew et al. focused on close combats and proposed a technique in order to make optimal maneuvering decisions. This technique uses Dynamic Programming as a basis and implements ADP (Approximate Dynamic Programming) to reduce computational time in complex situations. The proposed technique using ADP focuses on one-to-one air combat scenarios like general pursuer-evader games. McGrew et al. define the problem by using states, actions, and reward-goal functions. System state has been defined by UAV heading, bank angles, and positions. Actions are the possible actions of UAVs in particular states. Additionally, these goal functions are calculated using Aspect Angle, Antenna Train Angle, and Heading Crossing Angle.

Xiaoteng Ma, Li Xia, and Qianchuan Zhao [29] used a different technique for the Air Combat Strategy problem. Ma et al. approach the problem, which is explained in the paper of McGrew et al. by using Deep Q-Learning. Ma et al. uses a similar problem definition as McGrew et al. and analyzes the problem using MDP (Markov Decision Problem). In MDP, there are states, actions, and rewards. Ma et al. define states by velocity, position, heading, and bank angles. Actions are defined by acceleration and turning angles. Ma et al. define five possible actions; maintaining the same speed, turning right, turning left, increasing the speed, or decreasing the speed. As the last item, goal functions are calculated through the help of Aspect Angle and Antenna Train Angle. Ma et al. implement the DQN algorithm according to these states, actions, and rewards.

Chuangchuang Sun, Yen-Chen Liu, and Ran Dai [22] proposed two different approaches for the UAV path planning problem by using avoidance zones. The first approach is the Numerical Optimization Approach, and the second one is Heuristic Search Approach. Sun et al. [22] proposed methods including these approaches to solve optimal path problems using avoidance zones. Figure 1.11 illustrates the UAV path from $[x_0, y_0]$ to $[x_f, y_f]$. Avoidance zones are defined by ellipses in Sun et al. [22], and these zones symbolize the no-fly zones for UAVs.

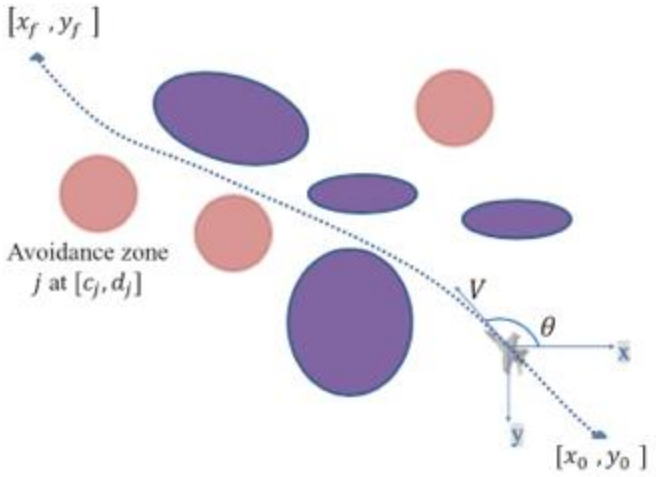

Figure 1.11 Illustration of UAV path planning problem with avoidance zones [22].

There are several pieces of research about pursuit-evasion problems and AI algorithms similar to Q Learning [28], [29], [30]. However, in real scenarios, world maps are not entirely available for safety flights. There are enemy zones over which it is dangerous to fly due to enemy Air Defense Systems and other weapon threats. On the other hand, there are also ally zones that are safe for flights. There are researches [22], [23], [24], [25], [31] where they used avoidance zones or threat zones while creating the optimal path from one point to another point. In this study, these two problems are merged, and the new problem is defined as performing pursuit-evasion missions in an area that includes green, gray, and red zones. To solve this problem, Mission Control Software is developed, and that software is proposed as a concept solution for the pursuit-evasion problems with green, gray, and red zones.

2 SIMULATION ENVIRONMENT

2.1 Turumtay – Mini UAV Platform

Turumtay mini UAV Hardware Platform is a typical non-profit project of 3 colleagues, including the writer of this thesis. Turumtay UAV logo is depicted in Figure 2.1. In this project, Turumtay Flight Control Card is designed to be used in the UAV platform by the other two members, and it is used as a hardware simulation environment in this thesis. So, the design process of this flight control card is beyond the scope of this thesis. All software in Turumtay is completely developed by the author of the thesis. In this thesis, Turumtay is solely used to simulate the ground control station of the UAV to test the mission control software developed in this thesis.

Figure 2.1 Turumtay UAV Logo

Turumtay is a typical Mini UAV in terms of its physical characteristics. Turumtay's length is 0.85 meters, and its height is 0.4 meters with a wingspan of 1.3 meters. Turumtay has a 1300 g electronic payload in total. The total flight weight of Turumtay is around 2000 g. The average flight time is 45 minutes, and Turumtay's maximum speed is approximately 170 km/h.

2.1.1 Flight Control Card

There is 2 IMU (Inertial Measurement Unit), 2 Barometer, 2 Magnetometer, a GNSS Receiver, an STM32 Micro Controller, and several other components in Turumtay Flight Control Card. The Flight Control Card is designed by the other two Turumtay Team Members, and the design process of this card is not included in this thesis. STM32 Micro Controller is used to run the Flight Control software developed in this thesis and simulate the environment. Flight Control Card can be seen in Figure 2.2.

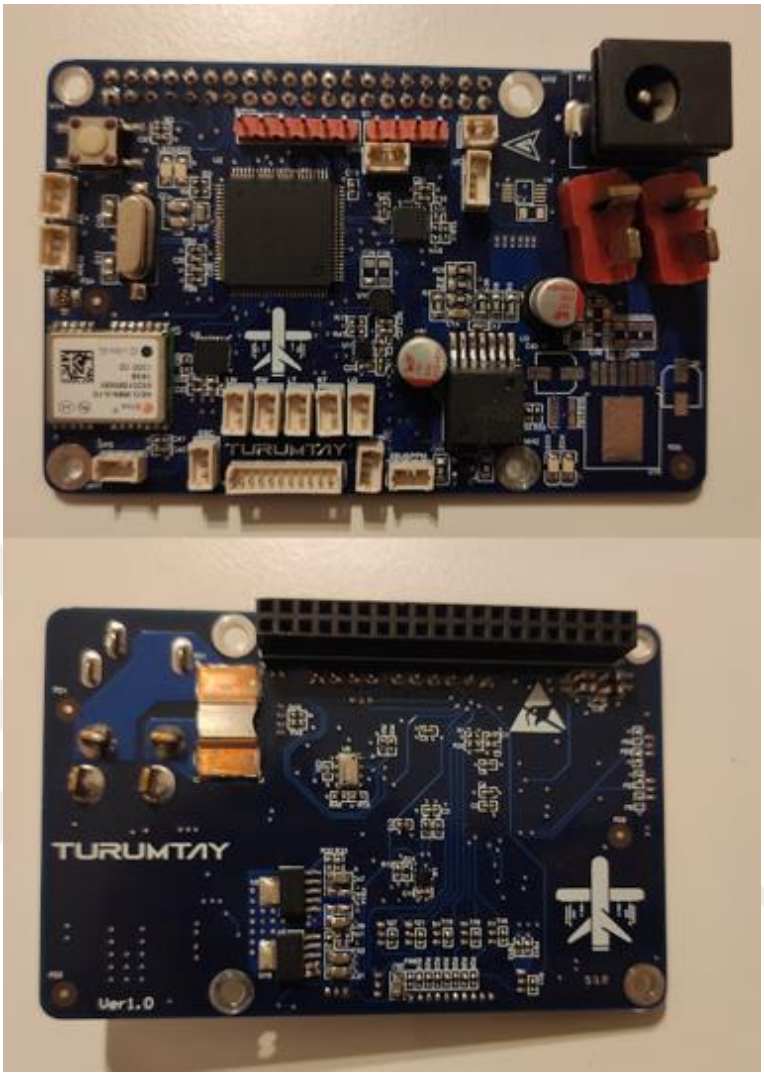

Figure 2.2 Turumtay Flight Control Card

2.2 System Architecture of Turumtay

2.2.1 Turumtay UAV System Architecture

Turumtay UAV System is composed of sensor units, actuator units, power units, and communication units. Figure 2.3 shows the detailed system architecture of Turumtay UAV.

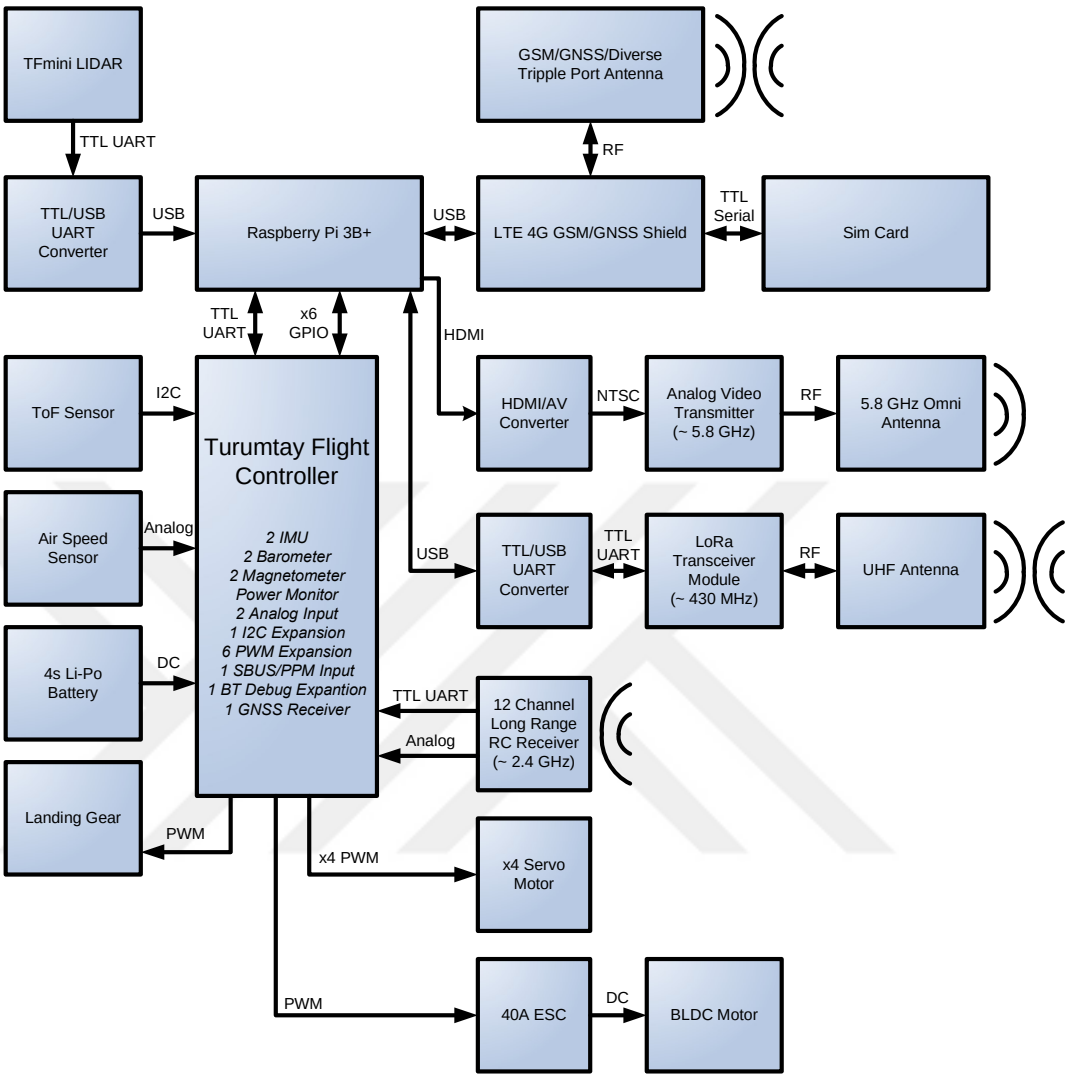

Figure 2.3 Turumtay UAV System Architecture

2.2.1.1 Turumtay UAV Sensor Units

Turumtay UAV Sensor Units are responsible for environmental awareness and acquiring information about UAV systems and the environment. These sensors are camera, GNSS Receiver, altitude sensors, airspeed sensors, IMU, LIDAR, and battery measurement sensors.

Turumtay uses the Raspberry 3 Camera as the main camera at the front of the UAV and provides Field of View of the Turumtay to the pilot on the ground station.

Turumtay gets its location information from GNSS Receiver with its external antenna on the top of Turumtay and uses this location in software algorithms.

GNSS is not sufficient in terms of altitude. However, altitude information is critical for UAVs. An inaccurate altitude level can cause crashes, especially during landing and takeoff. In order to gain a more accurate altitude level, external altitude sensors are used. These external sensors are barometric altitude sensors and LIDAR. Barometric altitude sensors perform well at high altitudes. However, barometric sensors cannot generate sensitive results when a UAV is too close to the ground. LIDAR sensors take place in these situations, and LIDAR outputs provide more accurate altitude results close to the ground. So, Turumtay uses many sensors to have accurate altitude level information.

To gain speed information, Turumtay uses an airspeed sensor. This airspeed sensor calculates the speed of the platform by using atmospheric pressure differentials.

UAVs are needed to have accurate attitude information. Since yaw, pitch, roll angles are crucial in flight, systems must have accurate attitude angles for proper navigation. IMU sensors provide these attitude angles by using gyro meters and magnetometers.

The actual level of the battery is measured by using battery measurement sensors in Turumtay. This sensor calculates the voltage and current which goes out from the battery and measure the state of charge of the battery level.

2.2.1.2 Turumtay UAV Actuator Unit

Turumtay UAV actuator unit comprises one main BLDC motor, four servo motors, and one Electronic Speed Controller. BLDC motor is used in rotating propeller. Two servo motors are used in wing flaps, and the other two servo motors are used in tail flaps.

2.2.1.3 Turumtay UAV Power Unit

Turumtay uses Li-Po batteries as the power resource. Battery with 8000 mAh capacity is enough for an approximately 45-minute Turumtay flight.

2.2.1.4 Turumtay UAV Communication Unit

Turumtay has four communication links: video transmission, telemetry, manual pilot, and LTE 4G.

Video Transmission System uses 5.8 GHz high-frequency link with 5 Mbit high bandwidth. Turumtay can send its field of view video to the ground station by using this link. Raspberry Camera on Raspberry outputs video in HDMI format. HDMI-AV converter converts HDMI to Analog Video. Analog video is sent to the ground station by using an omnidirectional 5.8 GHz antenna.

The telemetry system is used for the main data transfer link between Turumtay UAV and the ground station. All telemetry data are transferred from UAV to Ground Station, and all commands from the Ground Station are transferred to UAV with this communication link. 430 Mhz transmission modules carry out the communication.

In addition to the video transmission and telemetry communication link, there is also another communication link dedicated to the pilot. A pilot who is using an RC controller in Ground Station sends commands through this link. Even if another communication link is not available, the pilot can remotely control the UAV using Manual Pilot RC Communication Link.

Video Transmission, Telemetry, and Manual Pilot Communication Systems are LOS links, all of which depend on line of sight. In emergency situations such as loss of line of sight or error in communications, LTE-4G modules are used to communicate with UAV and Ground Station.

2.2.2 Turumtay Ground Control System Architecture

Ground Control Stations of UAVs are responsible for data monitoring of the flight and commanding the mission by sending new mission commands to the UAV using communication systems [32]. Turumtay Ground Control Station is mainly responsible for UAV communication. As shown in Figure 2.4, there are three communication types: Video Receiver, Telemetry, and Manual Pilot.

Analog Video of Turumtay is received through a 5.8 GHz directional antenna and converted to Digital Video. After this conversion, Digital Video is transferred to Ground Station Computer over USB. The pilot can watch the video of Field of View in Ground Control and send commands according to these views.

Telemetry communicates over 430 MHz UHF Antenna. First, the UHF antenna receives signals and sends these signals to Transceiver Modules, which demodulates signals and converts them to digital signals. Typical UART-USB Converter converts UART output of the module to USB and sends them to Ground Station Computer. Telemetry Information of Turumtay is received via this link, and mission commands are sent to Turumtay by using the telemetry communication link. GUI application which runs on Ground Control Computer, provides received telemetry data to Pilot.

Manual Pilot System is independent of other communication links, and Manual Pilot System is used for remote control of UAVs. The pilot controls Turumtay remotely over this RC link.

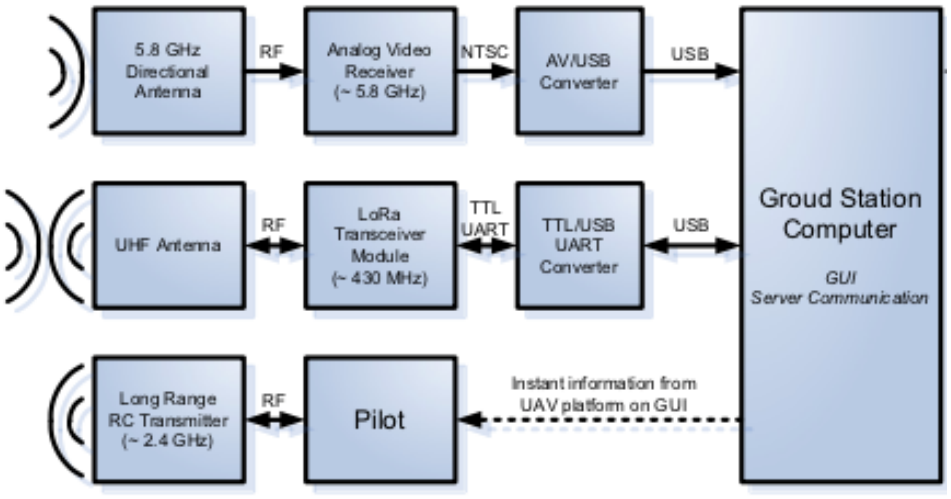

Figure 2.4 Turumtay Ground Control System Architecture

2.3 Software Architecture of Turumtay

Three different software runs on Turumtay; Ground Control, Mission Control, and Flight Control software. The whole software package manages all scenarios and algorithms and controls the flight parameters.

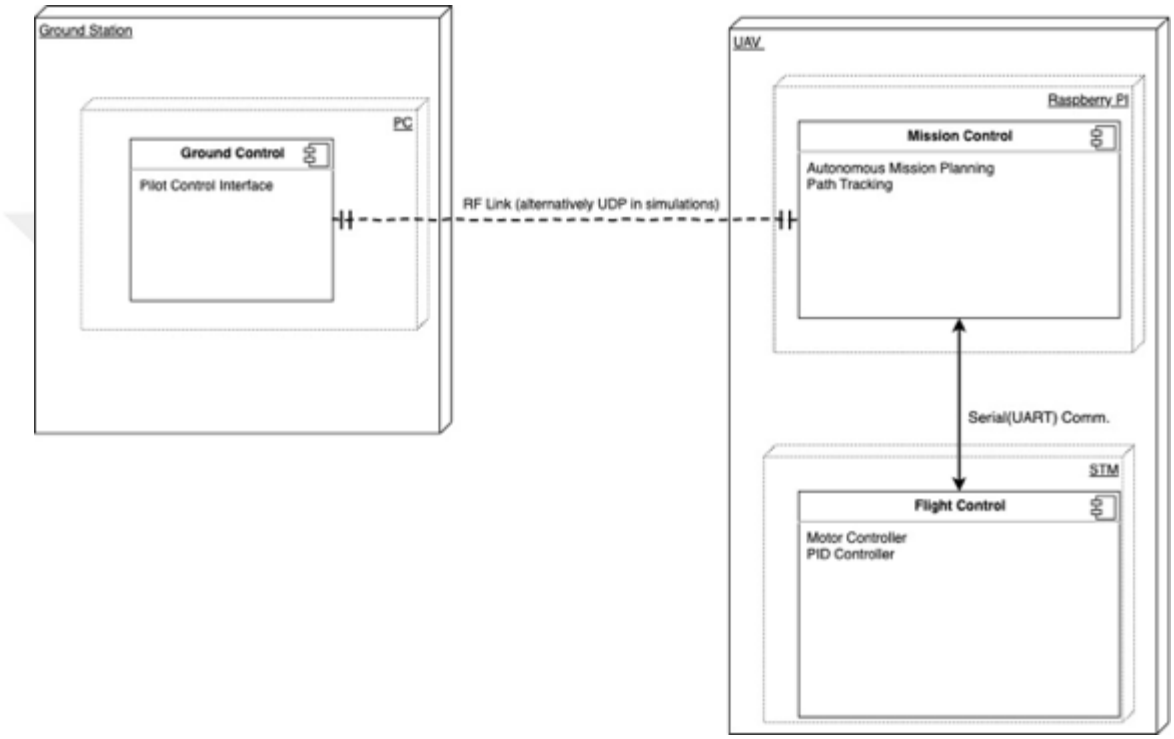

Figure 2.5 Communication Interfaces of Turumtay Software

Ground Control runs on Turumtay Ground Control Station, Mission Control and Flight Control software run on Turumtay UAV. Mission control software runs on Raspberry Pi, and Flight Control software runs on Turumtay FCC. All software needs to communicate with others to transfer necessary information and commands. There are two communication channels; one is between the ground control and mission control software, and the other is between mission control and flight control software. Mission Control is a middleware software, and it communicates with both software.

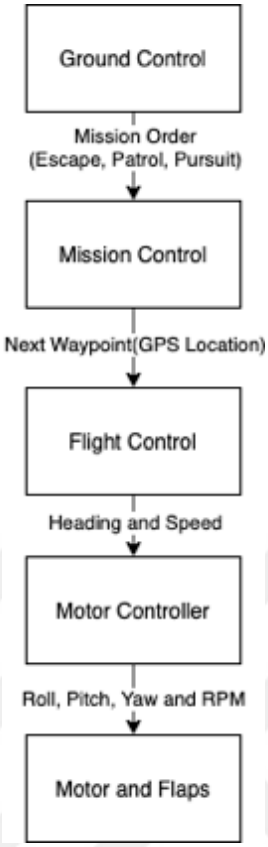

Figure 2.6 Software Architecture of Turumtay

Turumtay software use a layered software architecture design pattern. All layers are responsible for different tasks and do these tasks while communicating with other layers. Ground Control software is a web server that runs on Ground Control Station PC, and the pilot uses the system thanks to this software. Turumtay's telemetry data is shown in this GUI, and the pilot can change the active mission from this software and send new mission order to Mission Control software. Mission Control is responsible for all mission-critical tasks and artificial intelligence. Mission Control finds routes according to actual mission; after Mission Control sends the next waypoint as a GPS Location to Flight Control. Flight Control software runs on STM32 Micro Controller on Turumtay FCC, and it is responsible for all flight-critical controls.

2.3.1 Ground Control

Ground Control is a web server application that runs on a ground control PC. Ground Control Software uses Python and Flask webserver frameworks. Moreover, Three JS 3D WebGL Library are used in the JavaScript frontend design of Ground Control. The software of Ground Control is cross-platform so that it can run on many operating systems such as Windows, Ubuntu, and MacOS. Location, speed, and other status messages of Turumtay can be monitored on Turumtay's website with any mobile phone or PC.

Ground Control Software communicates with Control Software which runs on Turumtay UAV over RF Link or UDP in a simulation environment. Ground control sends new orders such as return to base, patrol around the location, or pursuit enemy UAV with these communication interfaces. Ground Control Software acts like basic mission planner software, and it manages flight remotely.

2.3.1.1 Communication Interface of Ground Control

Ground Control Software consists of two main components: Backend and Frontend, as a typical web server software. The frontend part is implemented with JavaScript, and it runs on the client's web browsers. On the other hand, the backend is a Python project which uses the Flask framework. Websockets are used in communication between Backend and Frontend. Websockets are based on TCP having a typical client-server communication protocol. Backend is a server that supports many Web Browser JavaScript clients. In this communication, JSON (JavaScript Object Notation) is used as a message structure. JSON is one of the popular serialize-deserialize string formats like XML notation.

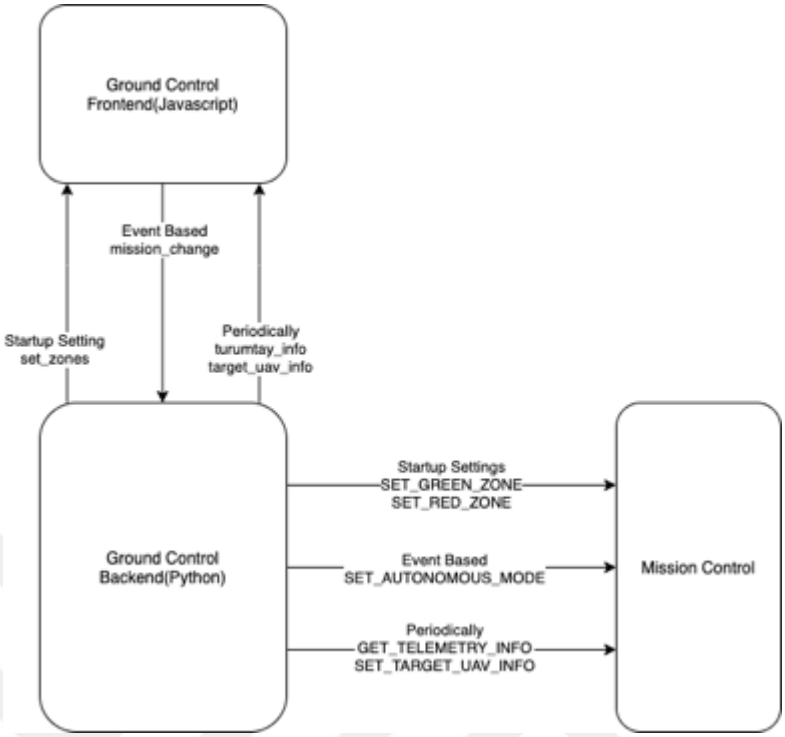

Figure 2.7 Communication Interface of Ground Control

On startup, the backend sends startup settings to connected clients. These settings are boundaries of green and red zones. According to this information, the backend and frontend periodically communicate to transfer Turumtay Info and Target UAV Info to frontend and refresh map in frontend. The operator who controls Turumtay can trigger an event by clicking the button on the webpage, and as a result of this click, a message is sent from frontend to backend to change autonomous mode.

The backend of the Ground Control Software broadcast information about zones to all connected Frontend clients, as shown in Table 2.1.

Table 2.1 Set_Zones Command

set_zones	
Command Code:	set_zones
Command Description:	This command is broadcasted from Ground Control Backend Server to all connected frontend clients on the first connection in order to notify about zones

Command Data:	{
	'enemy': [
	[150, -200], [200, -450],
	[450, -400], [400, 400],
	[170, 400], [140, 200],
	[50, 70]
	],
	'ally': [
	[-450, -450], [-100, -350],
	[-200, -100], [-100, 250],
	[-120, 460], [-380, 250]
	]
	}

Details of the Turumtay Info command are shown in Table 2.2. This command is used to notify clients about the information about Turumtay, such as location and speed.

Table 2.2 Turumtay_Info Command

turumtay_info	
Command Code:	turumtay_info
Command Description:	This command is broadcasted from Ground Control Backend Server to all connected frontend clients periodically in order to notify about the information of Turumtay
Command Data:	{ “id”: “61”, “roll”: “1.2”,

	“pitch”: “0.5”, “yaw”: “10.5”, “lat”: “200”, “lon”: “150”, “alt”: “20”, “autonomous”: “true”, “system_time”: “15:42:54” “speed”: “30”, “waypoints: [{ “lat”: “10”, “lon”: “65”, “heading“: “45”}, { “lat”: “210”, “lon”: “-350”, “heading“: “270”}] }

Target UAV Info Command is similar to Turumtay Info Command. In this command, the server notifies clients about the telemetry data of the Target UAV. Details of these data are shown in Table 2.3. After receiving this data, clients update their graphical user interfaces.

Table 2.3 Target UAV Info Command

target_uav_info	
Command Code:	target_uav_info
Command Description:	This command is broadcasted from Ground Control Backend Server to all connected frontend clients periodically in order to notify about the information of Target UAV

Command Data:	{
	“id”: “1”,
	“roll”: “5.5”,
	“pitch”: “10.2”,
	“yaw”: “250”,
	“lat”: “-100”,
	“lon”: “140”,
	“alt”: “20”,
	“speed”: “20”,
	}

2.3.2 Mission Control

Mission Control is software that is responsible for mission planning and managing scenarios. Python is used in Mission Control software, and it runs on Raspberry PI on Turumtay. According to the active mission, the software creates and follows routes by using Q Learning Techniques.

Mission Control communicates with Ground Control to take mission commands and send a status message and telemetry data. Mission Control also communicates with Flight Control Software over UART serial communication. Mission Control sends the next waypoint GPS location to Flight Control and receives telemetry data from flight control.

2.3.2.1 Mission Control Communication Interface

Mission Control communicates with Ground Control over UDP or RF Link. Mission Control serves UDP Socket, and Ground Control sends messages as a client. The port of the UDP socket is 6161. In this communication, all commands are in the structure. This structure consists of Message Length, Address, Command Code, and Parameters fields.

Table 2.4 Message Structure of Mission Control Software

Message Length(2 Byte)	Address(1 Byte)	Command Code(1 Byte)	Parameters(n byte)
The total length of the message	Address field for the source of the message	Command Codes indicate which message is sent.	Parameters depend on command. Some commands take parameters, and others do not.

Ground Control sends initial settings to Mission Control in the startup. These initial settings are the boundaries of the green zone and red zone. After that, runtime ground control asks some commands periodically at 1-second intervals. These periodic commands are getting Telemetry Data and setting Target UAV info. In addition to these commands, some commands are event-based such as changing autonomous mode commands.

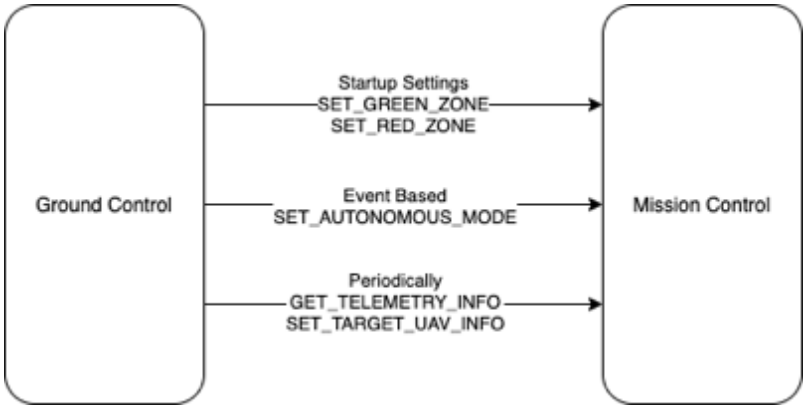

Figure 2.8 Comm. Interface between Ground Control and Mission Control

Ground Control software periodically sends the “get_telemetry” command to mission control to receive flight information of Turumtay. This information includes time, location, heading, roll, pitch, and speed as listed in Table 2.5

Table 2.5 Get Telemetry Command

GET_TELEMETRY = 0x61			
Command Code:	0x61		
Command Name:	GET_TELEMETRY		
Command Description:	This command is sent in order to get telemetry information, including location and speed		
Command Parameters			
Data	Type	Offset	Description
None			
Command Return Parameters			
Data	Type	Offset	Description
ID	UINT16	0x00	Turumtay ID

Hour	UINT8	0x02	Current Hour
Minute	UINT8	0x03	Current Minute
Second	UINT8	0x04	Current Second
Milliseconds	UINT8	0x05	Current Milliseconds divided by 10
Latitude	DOUBLE64	0x06	Latitude in Degrees
Longitude	DOUBLE64	0x0E	Longitude in Degrees
Altitude	FLOAT32	0x16	Altitude in Meters
Pitch	FLOAT32	0x1A	Pitch Degree
Yaw	FLOAT32	0x1E	Yaw Degree
Roll	FLOAT32	0x22	Roll Degree
Speed	FLOAT32	0x26	Speed in m/s
Battery Status	UINT16	0x2A	Battery Charge Status in Percentage
Autonomy Status	UINT8	0x2C	0 -> Manual Control 1-> Autonomous Control
Autonomous Mode	UINT32	0x2D	0 -> Patrol 1 -> Escape 2 -> Pursuit 999 -> Wait and do nothing

In order to change the mission to the autonomous mode of Turumtay, the “set_autonomous” command must be sent. Ground Control software sends this command when the operator clicks the button on the Webserver GUI. There are three autonomous modes of Turumtay; Patrol, Escape, and Pursuit: The details of this command are in Table 2.6.

Table 2.6 Set Autonomous Mode Command

SET_AUTONOMOUS_MODE = 0x85			
Command Code:	0x85		
Command Name:	SET_AUTONOMOUS MODE		
Command Description:	This command is sent in order to set the active autonomous mode of Turumtay		
Command Parameters			
Data	Type	Offset	Description
Autonomous Mode	UINT32	0x00	0 -> Patrol 1 -> Escape 2 -> Pursuit 999 -> Wait and do nothing
Command Return Parameters			
Data	Type	Offset	Description
Status	UINT32	0x00	0 -> Successful Change 1 -> Error

Turumtay calculates the optimal route according to the active mission. The route is composed of waypoints. This ordered waypoint list defines the point where Turumtay is going. When Turumtay reaches the first waypoint, the first waypoint is automatically removed from the list, and the second waypoint becomes the next waypoint. Ground Control periodically sends this command and receives the list of waypoints to show active route on the map.

Table 2.7 Get Waypoint List Command

GET_WAYPOINT_LIST = 0x86			
Command Code:	0x86		
Command Name:	GET_WAYPOINT_LIST		
Command Description:	This command is sent in order to get a waypoint list of the active route of Turumtay		
Command Parameters			
Data	Type	Offset	Description
None			
Command Return Parameters			
Data	Type	Offset	Description
Waypoint Length	UINT32	0x00	
Latitude	DOUBLE64	0x04	Latitude of 1st Waypoint
Longitude	DOUBLE64	0x0C	Longitude of 1st Waypoint
Heading	DOUBLE64	0x14	Heading of 1st Waypoint
Latitude	DOUBLE64	0x1C	Latitude of 2nd Waypoint
Longitude	DOUBLE64	0x24	Longitude of 2nd Waypoint
Heading	DOUBLE64	0x2C	Heading of 2nd Waypoint
…	…	…	…

Ground Control sets the boundaries of green and red zones at startup. These zones indicate the safety level of locations. Green zones are safe for flights, whereas red zones are dangerous for flight. These zones are determined according to the locations of ally and enemy forces. Outside of green and red zones are gray zones. Set zone boundaries command includes both green zone and red zones, as shown in Table 2.8.

Table 2.8 Set Zone Boundaries Command

SET_ ZONE_BOUNDARIES = 0x97			
Command Code:	0x97		
Command Name:	SET_ZONE_BOUNDARIES		
Command Description:	This command is sent in order to set green zone and red zone coordinates.		
Command Parameters			
Data	Type	Offset	Description
Point Count in Green Zone	UINT32	0x00	
Latitude	DOUBLE64	0x04	Latitude of 1st Point of Green Zone
Longitude	DOUBLE64	0x0C	Longitude of 1st Point Green Zone
Latitude	DOUBLE64	0x14	Latitude of 2nd Point Green Zone
Longitude	DOUBLE64	0x1C	Longitude of 2nd Point Green Zone
…	…	…	…
Point Count in Red Zone	UINT32	…	
Latitude	DOUBLE64	…	Latitude of 1st Point of Red Zone
Longitude	DOUBLE64	…	Longitude of 1st Point of Green Zone
Latitude	DOUBLE64		Latitude of 2nd Point of Green Zone
Longitude	DOUBLE64	…	Longitude of 2nd Point of Green Zone
…	…	…	…

Command Return Parameters			
Data	Type	Offset	Description
Status	UINT32	0x00	0 -> Successful 1 -> Error

Mission Control Software calculates the optimal path according to green, red zones, and target UAV. Ground Control sends information of Target UAV to Mission Control. Ground Control has information about target UAVs by using radar sensors or other external information sources. Table 2.9 details target UAV information.

Table 2.9 Set Target UAV Information Command

SET_TARGET_UAV_INFO = 0x95			
Command Code:	0x95		
Command Name:	SET_TARGET_UAV_INFO		
Command Description:	This command is sent in order to set the target UAV's location and heading information.		
Command Parameters			
Data	Type	Offset	Description
Latitude	DOUBLE64	0x00	Latitude of Target UAV
Longitude	DOUBLE64	0x08	Longitude of Target UAV
Altitude	FLOAT32	0x10	The altitude of Target UAV in meters
Heading	FLOAT32	0x14	Heading of Target UAV
Command Return Parameters			
Data	Type	Offset	Description
Status	UINT32	0x00	0 -> Successful, 1 -> Error

2.3.3 Flight Control

Flight Control is embedded software that runs on STM Micro Controller on Turumtay Flight Control Card. It takes information from sensors and manages the flight. According to the received next waypoint location, Flight Control calculates the desired heading vector and speeds.

Flight Control is also responsible for controlling the BLDC motor and four servo motors to adjust flaps according to the desired flight pattern. After the calculation of desired heading vector and speed, Flight Control drives motor controllers with these parameters. The flight parameters as Target roll, pitch, yaw, and rpm values are calculated by the motor controller using PID Controller. This process is called waypoint navigation, and waypoint navigation algorithms include attitude, heading stabilization, altitude stabilization, and speed stabilizations [32].

In the simulation, Flight Control is used in simulation mode, and there is no interaction with motors; it only simulates movements. So, motor controllers are part of Turumtay software architecture, but motor control design is beyond this thesis.

3 SOFTWARE DEVELOPMENT

3.1 Reinforcement Learning and Q-Learning

Reinforcement Learning is mainly based on Markov Decision Process (MDP). The Markov Decision Process defines states (S), actions (A), probabilities of actions (P_a), and rewards (R) in order to handle a problem. The process starts from the initial state. Then, random action is applied, which changes the current state, and a reward of that state is gained. In Reinforcement Learning, the learning process is performed in a cycle of sense-act-learn [33]. On every step, reward according to that state is considered. MDP aims to maximize the final reward.

Reinforcement learning is a model-free learning model, which means that the Reinforcement Learning characteristics of the used model do not affecting the learning process [34]. Thanks to this advantage, developed learning algorithms can be used within different platforms by minor changes [32].

Q Learning is one of the most used techniques in Reinforcement Learning which uses action-value functions [35]. The name of the Q Learning comes from the Q Table, which is used in learning algorithms to calculate the expected utility of actions [35]. Q Learning algorithms can work without prior knowledge about the results of actions, which makes solving real-world problems easier for agents.

In the real world, having all the knowledge about the environment and estimating all possible situations is nearly impossible [36]. One of the most advantages of Reinforcement learning is that it does not require knowing all the information and possible interactions in the environment so it can learn and improve itself while interacting with the environment by randomly chosen actions and calculating the resulting reward of that action in the states [31].

3.2 Agent, Environment, Action, and Reward Model

In the Agent, Environment, Action, and Reward model, the agent performs an action at one state. As the result of this action, the environment causes the state to transition with a reward.

3.2.1 State Set

States are denoted by S, and states are composed of X, Y, H, which are positions in latitude, position in longitude, and heading angle as in Table 3.1. X, Y positions, and H heading angles are continuous elements. Therefore, these continuous spaces need to be divided into discrete spaces to solve this problem and apply algorithms.

Table 3.1 State Set elements

Element	Description
X	X position of the UAV
Y	X position of the UAV
Heading Angle	Heading angle of the UAV

Coordinates are composed of infinite points, but this infinity makes the problem impossible to solve [27]. The map and coordinate system are divided into equal-sized squares. The radius of these spheres is 20 meters, and the center of these spheres is considered one point in the coordinate system. In Equations (3.1) and (3.2), there are formulas for the discretization of positions. While discretizing, latitude and longitude are converted into X, Y positions are the center points of spheres.

$$X = \left(floor\left(\frac{Latitude}{20}\right) * 20\right) \tag{3.1}$$

$$Y = \left(floor\left(\frac{Longitude}{20}\right) * 20\right), \tag{3.2}$$

States also have an H heading angle element for UAVs. Exact heading angles are not important in pursuit or evasion; field of view is enough for these problems. So, heading

angles are put into a process by dividing them by 45 degrees, similar to the discretization of position. This discretization is done by using Equations (3.3) and (3.4).

$$H_1 = \left(floor \left(\frac{Heading\ Angle\ of\ Turumtay}{45} \right) * 45 \right) \tag{3.3}$$

$$H_2 = \left(floor \left(\frac{Heading\ Angle\ of\ Enemy\ UAV}{45} \right) * 45 \right) \tag{3.4}$$

3.2.2 Actions

Actions are denoted by A, and there are three possible actions that Turumtay can do during the transition from one state to another. As it can be seen in Table 3.2, actions are discretized into three movements as moving forward left, forward center, and forward right. These actions are illustrated in Figure 3.1.

Table 3.2 Possible Actions of UAV

Possible Actions	Description
Move Forward Left	Turn left by 45 degrees
Move Forward	Maintain heading vector
Move Forward Right	Turn right by 45 degrees

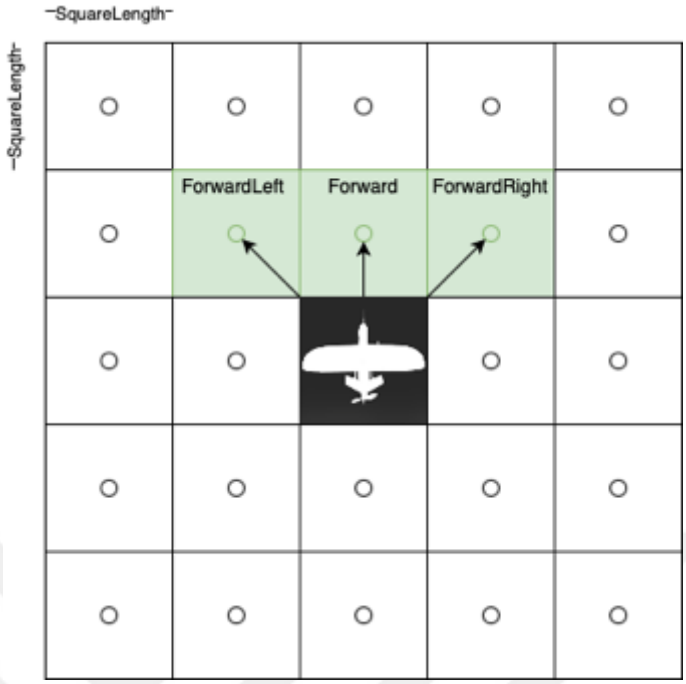

Figure 3.1 Illustration of Actions

In Q Learning, actions are responsible for the transition between states. In order to create transition relations between states, possible actions of each state and their results must be calculated. To calculate the result of the action, location and other elements of the next state must be known. In order to find the location of the next state, Equations (3.5) and (3.6) are used. In these calculations, basic trigonometry is used to find the line's endpoint whose length, angle, and start point are known. So, the line's start point is the current state's position, length is equal to speed, and angle depends on action.

$$\text{NextState.X} = \text{discretizePos}(\text{CurrentState.X} + (\text{speed} * \cos(\text{headingAngle})) \quad (3.5)$$

$$\text{NextState.Y} = \text{discretizePos}(\text{CurrentState.Y} + (\text{speed} * \sin(\text{headingAngle})) \quad (3.6)$$

In Equation (3.5) and Equation (3.6), “headingAngle” depends on the action type. “HeadingAngle” is equal to “CurrentState.heading – 45” while moving forward left and is equal to “CurrentState.heading” while moving forward center and is equal to “CurrentState.heading + 45” while moving forward right.

3.2.3 Function of Reward

In Q Learning problems, the agent aims to gain maximum cumulative reward until it reaches the target state, so constructing reasonable reward functions is critical for the agent's performance in the mission [24]. Rewards depend on the type of mission. Each mission type has different reward functions regarding its goals. While calculating rewards, goal states and the state of the Target UAV are used. There are some simple equations that are used in calculating rewards. These are the function of controlling out of range, distance calculation between states, bearing, and aspect angle calculations.

First, the out of range function is used to check if the states are in the map boundaries or not. In Figure 3.2, there is a simple Out of Range Control function. The software controls the limit of the map with this function. Limits of the map are important in order to penalize states which are out of the limits.

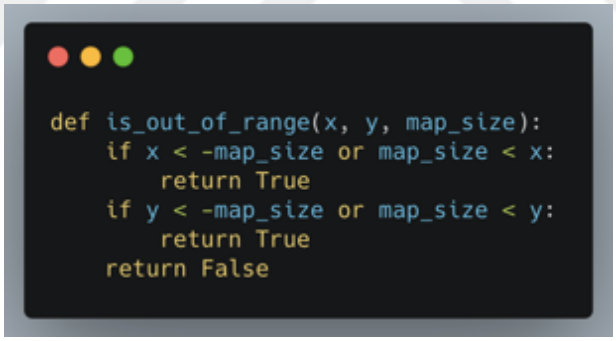

Figure 3.2 Out of Range Control Function

Calculating the distance between states is important in reward functions because reward functions depend on the distance between UAV and its goal state. Calculating distance between two locations is used while initializing rewards. There is the ordinary calculation of distance function in Figure 3.3.

```
def calculate_distance_in_meters(point_x_1, point_y_1, point_x_2, point_y_2):
    return sqrt(pow(point_x_2 - point_x_1, 2) + pow(point_y_2 - point_y_1, 2))
```

Figure 3.3 Calculating Distance in Meters Function

The distance between Turumtay and Enemy UAV is also a continuous variable, and with this continuity, the size of the possible states goes infinite count. So, these distance variables need to be analyzed with groups. Distance is important while choosing the right strategy for pursuit or evasion, but the precision of the distance is not required. Distance groups are sufficient in order to do an action concerning the enemy position. So, these groups consist of "Too Close Target, Close Target, Far Target, and Very Far Target".

Table 3.3 Discretization of Distance

Discretized Distance	Actual Distance
Too Close	$d \leq 50$
Close	$50 < d \leq 200$
Far	$200 < d \leq 750$
Too Far	$750 < d$

There are four distance groups which are Too Close, Close, Far, and Too Far. The Too Close group covers distances closer than 50 meters, and the Close group covers distances between 50 and 200 meters. Distances between 200 meters and 750 meters are included in the Far group. Too Far group contains distances over 750 meters as in Table 3.3

```python
def calc_bearing(self, other_state):
    theta = math.atan2(other_state.lon - self.lon, other_state.lat - self.lat)
    return (math.degrees(theta) + 360) % 360
```

Figure 3.4 Calculation of Bearing Angle

In pursuit-evasion missions, heading angles and distance between Turumtay and Target UAV determine aircraft's targeting zones which is critical in the construction of reward tables [30]. In order to calculate aspect angles, the bearing angle is needed. Bearing angle is a horizontal direction angle between points, and it is calculated by using X, Y positions of two states as in Figure 3.4.

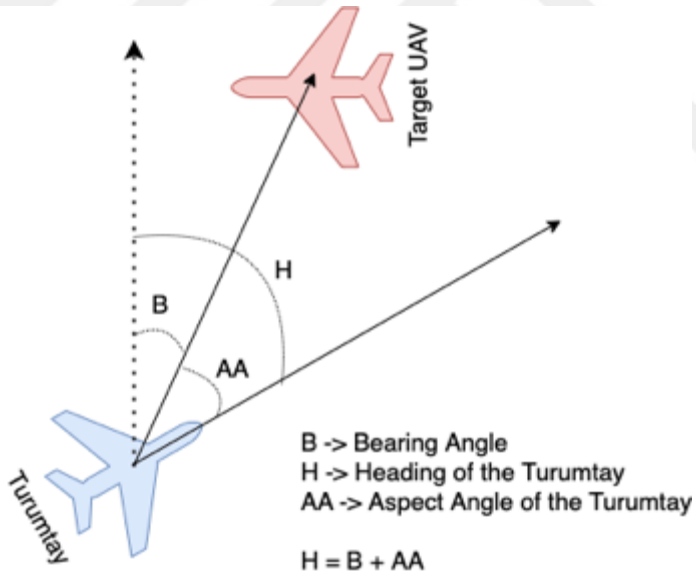

Figure 3.5 Illustration of the Bearing and Aspect Angle

Aspect Angle is similar to bearing angle. However, the aspect angle heading of the state is also taken into account because this angle shows the field of view of the UAV. Aspect Angle is calculated by using bearing angle, and the only difference is that bearing angle is

subtracted from the heading angle of the UAV. There is an illustration of the bearing angle and the aspect angle of the Turumtay Figure 3.5. In Figure 3.6, the calculation of the aspects angle is shown.

```python
def calc_aspect_angle(self, other_state):
    aspect_angle = (self.heading - self.calc_bearing(other_state) + 360) % 360
    if aspect_angle > 180:
        aspect_angle -= 360
    return aspect_angle
```

Figure 3.6 Calculation of the Aspect Angle

There are four Aspect Angle categories; Forward, Left, Right, and Back, as in Table 3.4. Forward Aspect Angle means that Enemy UAV is in the field of view of the Turumtay; this is the goal state in pursuit missions. The Left and Right categories imply the direction of the enemy, according to Turumtay. Finally, in the back category, the enemy is behind the Turumtay, and it is the most dangerous position in pursuit-evasion missions.

Table 3.4 Discretization of Aspect Angle

Aspect Angle Enum	Actual Aspect Angle
FORWARD	$-45 < AA < 45$
LEFT	$-135 \leq AA \leq -45$
RIGHT	$45 \leq AA \leq 135$
BACK	Otherwise

3.2.3.1 Reward Function of Patrol Mission

In Patrolling mission, the main goal is to reach the waypoint. In addition to this, avoiding entering red zones is critical. While creating the path, Turumtay should stay in the green zone rather than gray zones, and it should choose the shortest path. Details of the reward function for patrol missions can be seen in Table 3.5. If Turumtay exits from the limits of the map, it is penalized with a -9999 point. If it reaches the goal state, it receives 3333 points because, in the patrol mission, there are three sub-routes which are from initial position to first patrol point, from the first point to second point, and from the second point to first point. So, in order to make a maximum point to 10000, the rewards of 3 sub-missions are equal to 3333. If it chooses red zones, it takes -1000 points. On the other hand, the penalty is only -1 in the green, zone and the penalty is -10 in gray zones. All points in green, red, and gray zones have negative reward points in order to make the path shorter and prevent unnecessary longer ways.

Table 3.5 Reward Function of Patrol Mission

Reward	Condition
-9999	State out of the map
-1	State in Green Zone
-1000	State in Red Zone
-10	State in Gray Zone
3333	State is closer than 20 meters to Patrol Point

3.2.3.2 Reward Function of Escape Mission

While performing an escape mission, Turumtay tries to escape to a safe zone considering green, red zones information and target UAV's position. Turumtay prefers shorter paths and keeps itself away from the target UAV while escaping. Implementation of the reward function for the escape mission can be seen in Table 3.6. States which are out of the map are penalized with -9999 points, and states in the red zone are penalized with -1000 points. If the distance between Turumtay and Target UAV is Too Close, a reward would be -200. If the distance is Close reward would be -100 points. In addition to these penalties, if Turumtay is in the field of view of the Target UAV and the distance is smaller than Close, the reward would be -400. The reward of entering the green zone while distance is Far or Too Far is 10000, and it is the mission's main goal. In other cases, Turumtay gets a -1 point as a reward to prevent unnecessary movements that cause longer paths.

Table 3.6 Reward Function of Escape Mission

Reward	Condition
-9999	State out of the map
-1	State in Green Zone
-1000	State in Red Zone
-10	State in Gray Zone
-200	Distance between Turumtay and Enemy UAV is smaller than Too Close
-100	Distance between Turumtay and Enemy UAV is smaller than Close
-400	Distance between Turumtay and Enemy UAV is smaller than Close, and Enemy's Aspect Angle is Forward
10000	Turumtay in Green Zone and distance between Turumtay and Enemy UAV is larger than Close

3.2.3.3 Reward Function of Pursuit Mission

Pursuit missions are the opposite of escape missions. Table 3.7**Error! Reference source not found.** summarizes the reward function for pursuit missions. Similar to Patrolling and Escaping missions, states located out of the map are penalized with -9999 points, and states in the red zone are penalized with -1000 points. If Turumtay is Close to Target UAV and Target UAV is in the field of Turumtay and Turumtay is not in the field view of Target UAV, the reward is 10000 points because these states are desired states in the pursuit missions.

In one-to-one air combats, there are four typical categories in terms of headings which are Ally Advantage, Enemy Advantage, Common Safe, and Common Danger [37]. If Turumtay is behind the enemy and its aspect angle is between -45 and 45, Turumtay has the advantage. On the other hand, the enemy has the advantage if Turumtay is in front of the enemy and its aspect angle is not between the (-45,45) range. If both Turumtay and Enemy UAV has aspect angle between -45 and 45, they are in a common danger situation. Other positions are called common safe.

Table 3.7 Reward Function of Pursuit Mission

Reward	Condition
-9999	State out of the map
-1	State in Green Zone
-1000	State in Red Zone
-10	State in Gray Zone
10000	Distance between Turumtay and Enemy UAV is equal to Close, and Turumtay's AA is Forward, and Enemy's AA is Back

3.2.4 Discretization vs Performance Trade-Off

Discretization causes loss of precision; for example, when the heading angle is discretized with 45 degrees, angles between 0 and 45 degrees are not considered while creating the path. Furthermore, routes are composed of straight lines and diagonal lines. On the other hand, discretization makes the problem easy to solve by reducing the number of states and actions. So, there is a trade-off between performance and precision; when precision goes high, performance will decrease. Likewise, when precision goes low, performance will increase. In this study, the mission map is divided into squares, and at one step, UAV can travel to the forward left, forward-center, and forward right. So, each square has eight adjacent squares. To enable all possible travels between these eight squares, 45 degrees of precision will be enough for this simplicity.

With these discretization, 1000m x 1000 m map is divided into 50 x 50 = 2500 squares, and each square has 360 / 45 = 8 different heading angles. So, there are 2500 x 8 = 20000 states on this map. With these sets, preparation of the developed Q Learning algorithm takes approximately 10 seconds until the route is ready, and it takes approximately 80 seconds until Q Table is converged to optimal. These timings are calculated in Raspberry Pi 3B+. The trade-off between performance and precision, which is determined with discretization, affects the timing and performance of the algorithm.

3.3 Software Design for Q Learning Algorithm in Mission Control

While performing autonomous missions, mission control software first initializes state sets and actions. Reward function for mission is initialized after initializing states and convenient actions. When initializations are finished, the process of filling Q Table starts. While filling the table, Turumtay periodically calculates the optimal path and refreshes its waypoint list.

3.3.1 Initializing States

At the startup, mission control must initialize the state set, which is crucial in algorithms. First, the existence of state set in pickles is checked. If the state set pickle is ready to be used, states are read from the pickle, and the creation of the state set part is skipped. States are independent of mission types because the same state set is used in all

missions. So, initializing state sets are done only at startup and is not changed by changing mission type.

3.3.2 Initializing Actions

After initializing states, relations between these states need to be established. These relations are defined by actions. Actions indicate all possible transitions between states at one step.

Pickles are also used in initializing actions. In the process of training algorithms, actions are initialized and saved into a pickle. After that mission, control reads actions from this pickle and continues with initializing reward functions. Actions are not depending on missions and particular rewards. Thanks to this independence, actions are initialized once at startup and are not updated in runtime.

3.3.3 Initializing Rewards

All missions have specific objectives and different goal functions. Rewards are specialized tables for different mission types, and each mission has its own reward table. So, there is not one reward table and initializing different reward tables are needed. When there is a change in active mission, a proper reward table is initialized. In the initialization of the reward table, there are two options which are creating the table and reading from pickles. If there is no ready pickle to be read, mission control creates the proper reward tables and store them in order to use them when necessary.

3.3.4 Filling Q Table

Initialization of states, actions, and rewards are done at startup and are not changed in runtime. However, Q Table is a live table, and it is updated continuously, including real flight scenarios. The filling Q table is independent of the actual state of Turumtay. Therefore, the speed, heading of Turumtay does not take part while filling the table. This process randomly selects a state and updates its Q value by using Bellman Equation [27]. Q Table is defined by 2D-array, the first dimension is state id, and the second dimension is action id.

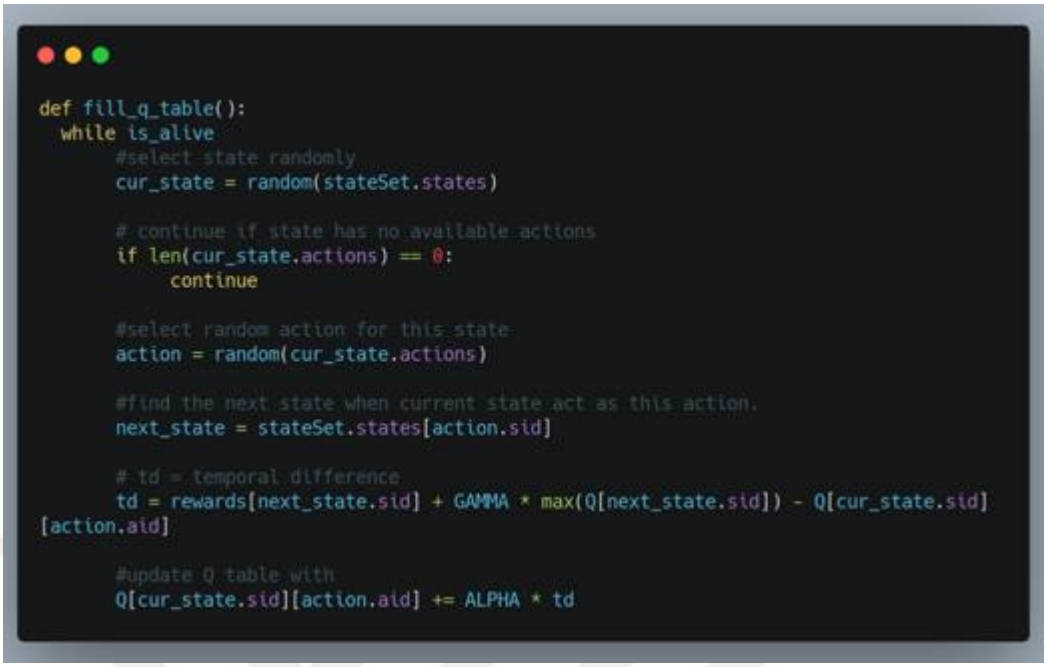

Figure 3.7 Filling Q Table

Pseudocode for filling Q table is visualized in Figure 3.7. The State is randomly selected from state lists in each step, and if there is no available action of that state, the random selection will be repeated. After the selection of the state, action is also picked randomly from available states of that state. When state and action are selected, Bellman Equation becomes ready to be used. First, the temporal difference is calculated by using the reward table, gamma discount rate, and Q table. The maximum Q value of the next state is multiplied by the Gamma discount factor as stated in Bellman Equation [27], and the Q value of the current state and selected action is extracted. After that, the reward of the next state is added to the calculated value, and the temporal difference is determined. Finally, the temporal difference is multiplied by Alpha Learning Rate. Then, we add updates to the Q value of the current state and current action.

3.3.5 Calculating Optimal Path

Filling the Q Table is not a finite process, so the table is updated all the time during the flight. So, creating waypoints cannot wait for the filling to complete. While mission control is filling the Q Table, it is also calculating optimal path asynchronously. The active mission does not affect the calculation algorithm for the optimal path because the algorithm depends on only the Q Table, and Q Table itself is unique for different missions.

```
def get_optimal_path(self, step_count):
  path = []

  # find start state by using Turumtay's current location and heading
  start_state = State(
      self.__turumtay.gps_latitude, self.__turumtay.gps_longitude,
self.__turumtay.yaw_deg)
  cur_state = self.stateSet.get_state(start_state)

  # path will be consist of step_count point
  for i in range(step_count):
      # take current Q value for comparison
      current_q = self.Q[cur_state.sid]

      # initialize max element with negative infinite number
      max_q = -999999999

      next_state = cur_state

      # iterate all actions and find the action with maximum Q Value
      for act in cur_state.actions:

          # update maximum Q Value
          if max_q < current_q[act.aid]:
              max_q = current_q[act.aid]

              # next state is the result state of the action applied on current state
              next_state = self.stateSet.states[act.sid]
      cur_state = next_state

      # append the current state to path
      path.append(cur_state)
  return path
```

Figure 3.8 Function of Get Optimal Path

The function to determine the optimal path is visualized as a pseudo-code in Figure 3.8. In order to determine the optimal path, which starts from Turumtay, current position and heading, Q table and state set are used. In each step, the maximum Q value of the action list of the current state is determined, and the new state is determined by applying that action to the current state. At the end of each step, the current state is appended to the resulting path until the specified count of step paths is found and returned.

4 SIMULATION STUDIES

In the simulation studies, three different cases for three different mission types are analyzed. These missions are selected to examine the results of the developed algorithms in different kinds of scenarios.

4.1 Patrol Missions

The first simulation case is a mission for patrolling. There are two patrol points in patrolling missions, and Turumtay goes them in order. When Turumtay reaches the first point, it continues with the other point, and it repeats.

4.1.1 Case 1

In Case 1, the map is divided into the green zone and red zone in large pieces. The Green Zone polygon is located on the left side of the map, and the red zone is located on the other side. In this case, Turumtay's initial position is behind the red zone, and the first patrol point is in the green zone; the other point is at the top of the red zone, as seen in Figure 4.1.

First, Turumtay escaped from the red zone and directly went to the green zone and reached the first point. After that, it continued to the second patrol point, which is in the top right corner of the map. When Turumtay reaches the second point, it draws an eight shape in order to do a smooth U-turn and goes to the first point again.

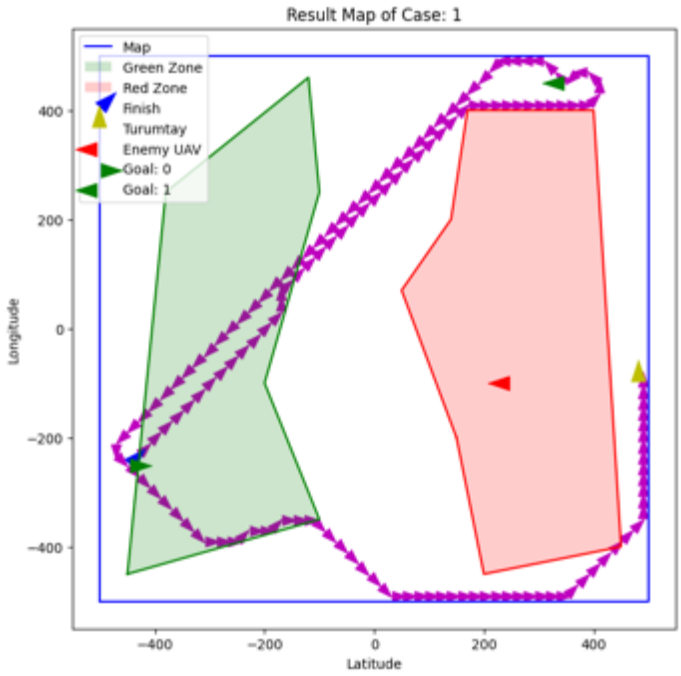

Figure 4.1 Case 1 Result Map

The reward chart of Case 1 is shown in Figure 4.2. This chart is basically formed of three significant steps, and these steps are the moments of reaching patrolling points. The reward is decreased first, especially while Turumtay is going to the green zone. After that, Turumtay reached up to the first point, and the reward is rapidly increased. Changes in the reward table continue like this pattern, in this case, small decrease while going to patrol point and rapid increase when reached the points.

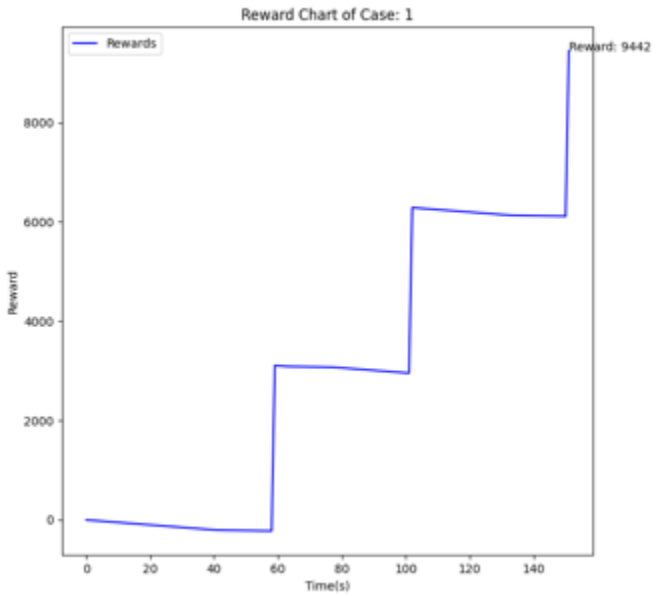

Figure 4.2 Case 1 Reward Chart

4.1.2 Case 2

The path of Case 2 is visualized in Figure 4.3. In this case, the green zone is located on the left side of the map, and the red zone is on the right side of the map. Turumtay is located in the top left corner, and patrol points are in the green zone and near the right edge of the red zone. Turumtay began its move by directly going into the green zone, and it continued in the green zone until reaching the first point. When it reaches the first point, its heading is towards the left because of the heading angle of the first point. So, in order to go to the second point, Turumtay moved as a U-turn and rotated its heading to the right. After that, it reached the second point behind the red zone and turned back again to the first point, and finished the patrol mission.

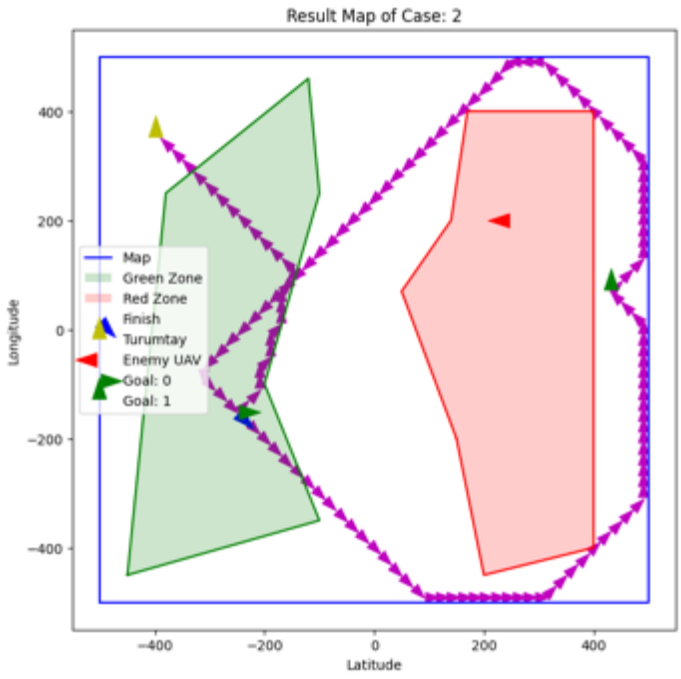

Figure 4.3 Case 2 Result Map

In the reward chart of Case 2 in Figure 4.4, there are three rapid increases in the chart for reaching patrol points. There are slight steady decreases between these rising edges because of the path which Turumtay followed between patrol points. The mission finished in 140th second when Turumtay came close enough to the first point again.

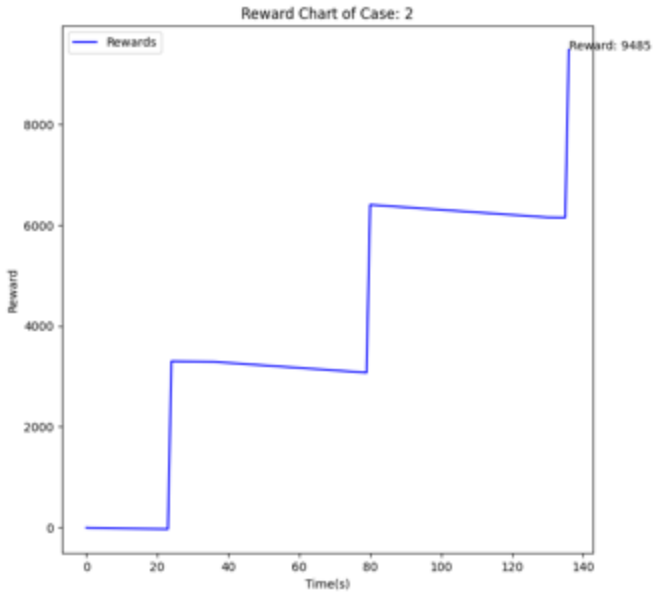

Figure 4.4 Case 2 Reward Chart

4.1.3 Case 3

In Case 3, the last case of the patrolling missions, there is a big rectangle of the red zone at the bottom of the map and a small rectangle of the green zone on top of it. At the initial position, Turumtay is located in the top center of the map, and patrol points are located at the left and right sides of the red zone. Turumtay started its movement by entering the green zone quickly, and after that, it moved into the green zone until the corner of the green zone. Then, it follows the boundaries of the red zone to reach the first patrol point located in the bottom left of the map. After reaching the first point, Turumtay continues to move without changing its direction and go around the bottom side of the red zone and reach the second point on the right side of the red zone. In the final step, Turumtay went the first point again from the second point while using the path which includes green zone points.

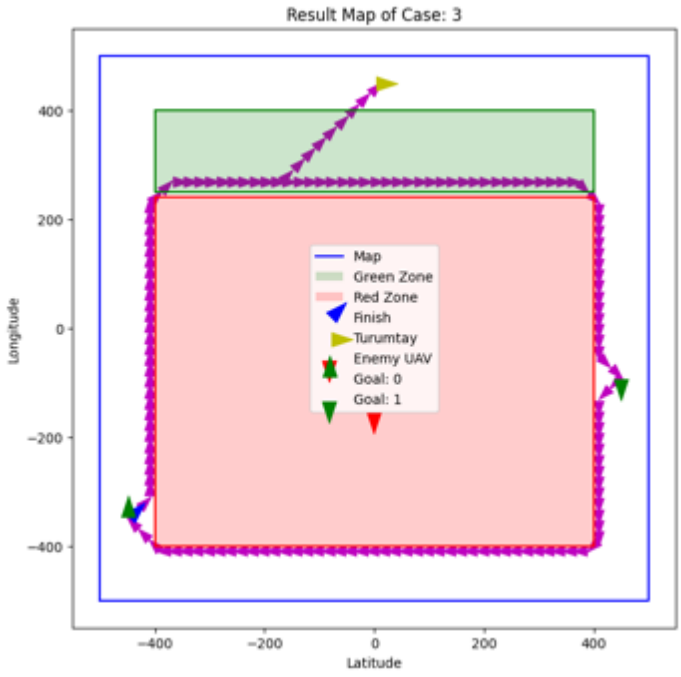

Figure 4.5 Case 3 Result Map

In Figure 4.6, distinct steps in the reward chart of Case 3 are seen. As a typical patrol mission, there are three rapid increases in the reward table orderly reaching the first point, second point, and first point again. Between these rises, there is a comparatively small decrease due to the ordinary penalties to make the path shorter on the map.

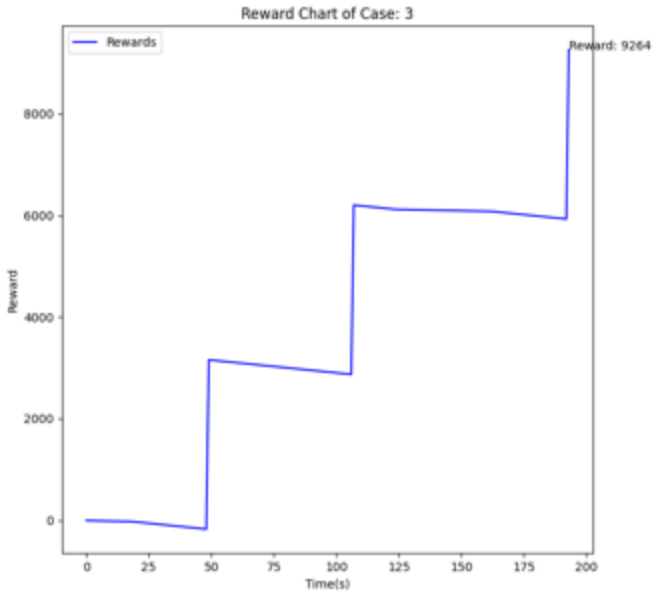

Figure 4.6 Case 3 Reward Chart

4.2 Evasion Missions

4.2.1 Case 4

The optimal route of Turumtay can be seen in Figure 4.7. Turumtay found a way around Red Zone, and without entering the red zones, it goes directly to Green Zone. While calculating this route, Turumtay also escapes from the Enemy UAV's Field of View.

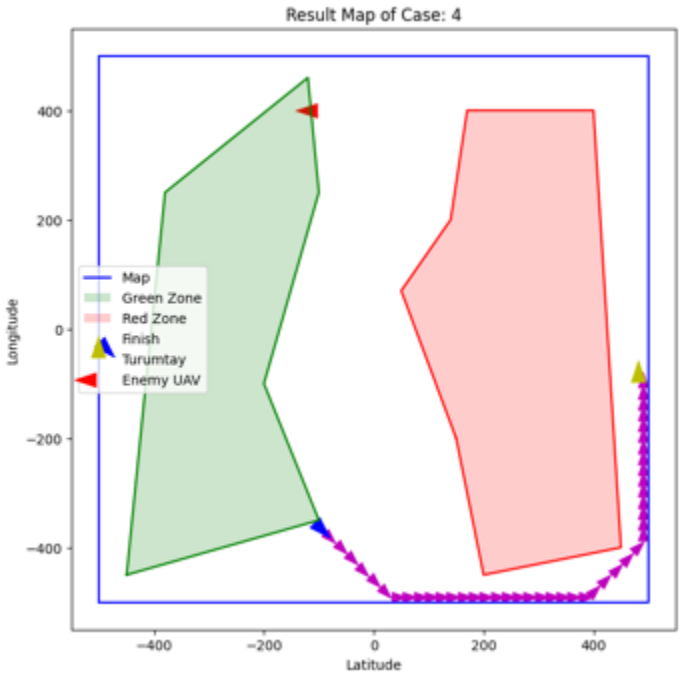

Figure 4.7 Case 4 Result Map

Reward Chart of Case 4 can be seen in Figure 4.8. First, the reward of the Turumtay is decreased because every step on the path decreases the total reward in order to make the route shorter. After a regular decrease, Turumtay entered Green Zone, and its reward was increased by 10000.

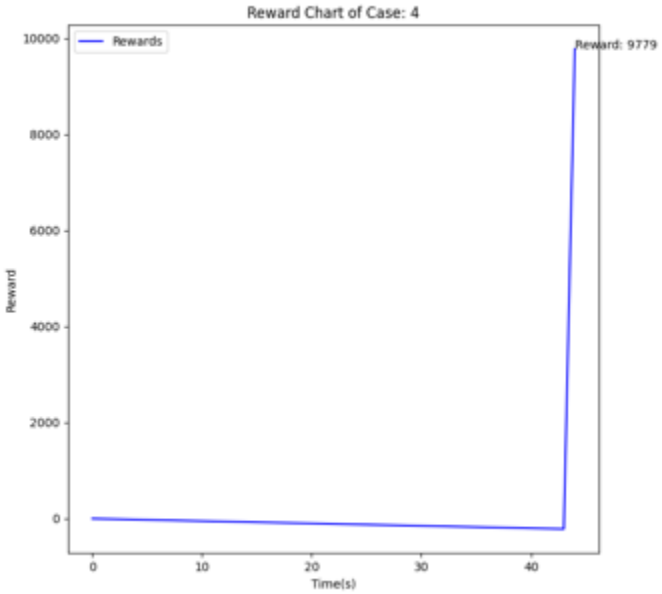

Figure 4.8 Case 4 Reward Chart

4.2.2 Case 5

In Figure 4.9, Turumtay's route is longer than Case 4 because its heading angle is in the opposite direction. According to gained experiences, Turumtay understands that turning down is not possible without entering the red zone, or it must leave the map. So Turumtay prefers a long path that goes from the upside of the red zone.

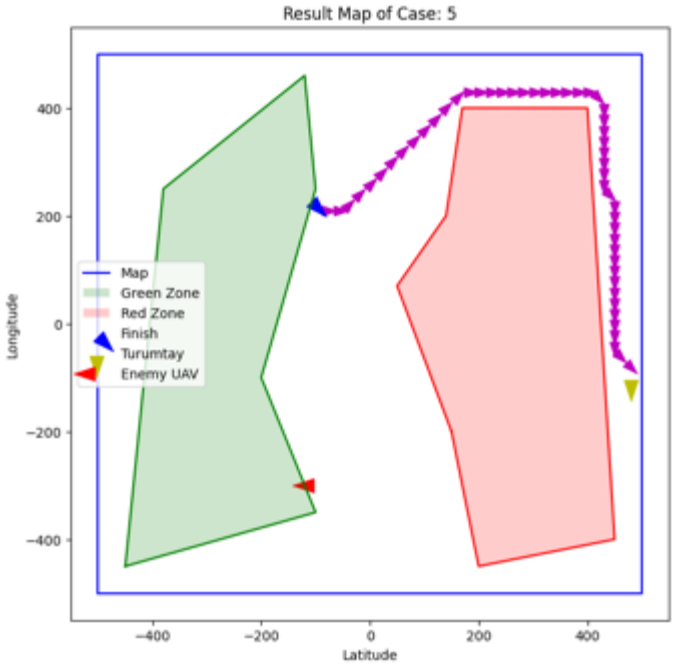

Figure 4.9 Case 5 Result Map

In evasion missions, Turumtay tries to reach Green Zone as soon as possible and prefers the shortest path without engaging with Enemy UAV. So initially, reward decreases lightly because of the longer path penalties. Finally, it is increased by 10000 when Turumtay reaches Green Zone, as depicted in Figure 4.10

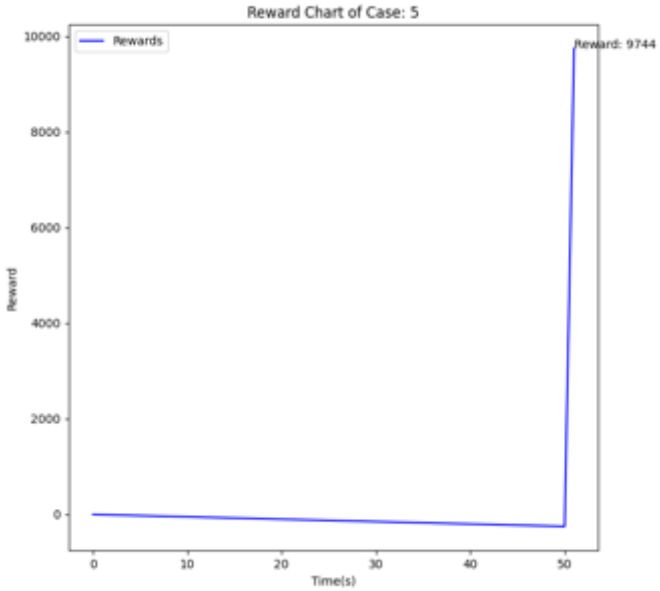

Figure 4.10 Case 5 Reward Chart

4.2.3 Case 6

Case 6 is the last case of evasion missions. In this case, there is a huge triangle of the red zone in the center of the map, and the floor of the triangle is on the bottom boundaries of the map. On the other hand, the green zone is a small shape on the bottom right corner of the map. Turumtay is located on the left side of the red zone, and Enemy UAV is at the top of the Turumtay. In evasion missions, Turumtay tries to go to the green zone as soon as possible while protecting itself from Enemy UAV. The resulting map of Case 6 is shown in Figure 4.11; Turumtay first rotated to the left in order to become distant from the Enemy UAV and its field of view. It rotated to the right after and went around Enemy UAV without being close to it. Then, Turumtay went around of red zone on top of the triangle and directly went to the green zone.

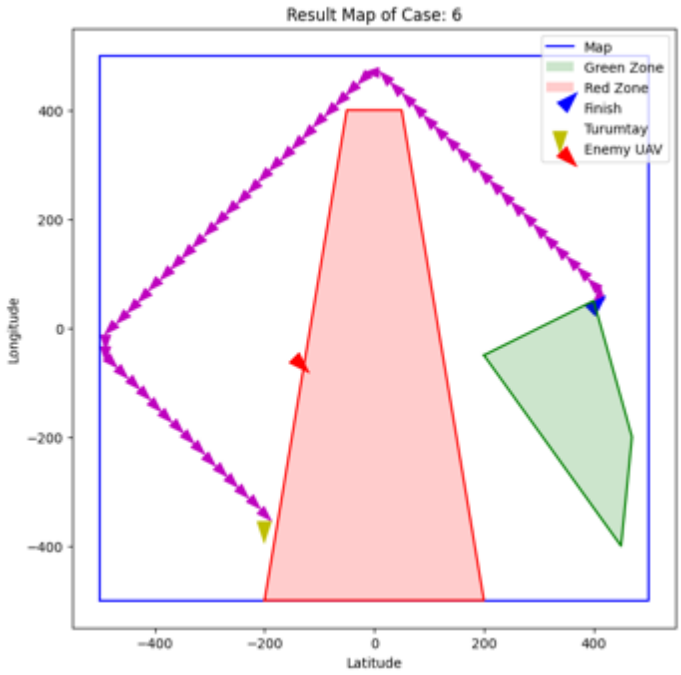

Figure 4.11 Case 6 Result Map

In the reward chart of Case 6, Turumtay is penalized lightly in first movements when it is in gray zones. After a steady decrease in the reward, Turumtay finally reached the green zone and gained a reward. These decreases and rapid increases are visualized in Figure 4.12.

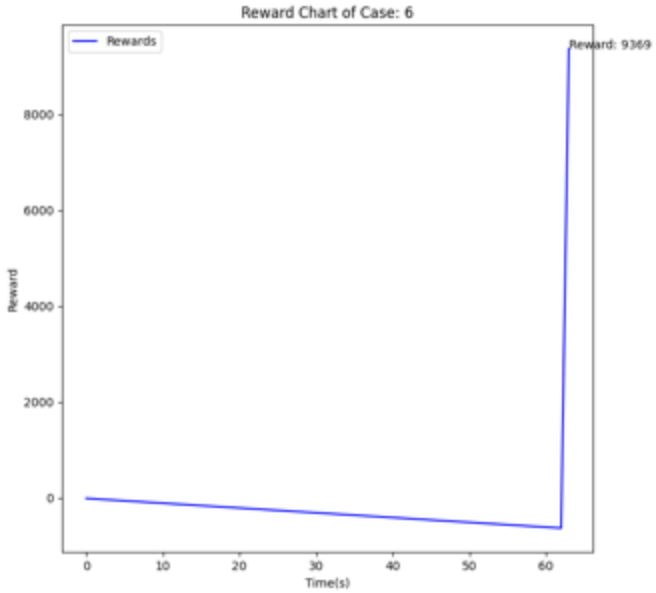

Figure 4.12 Case 6 Reward Chart

4.3 Pursuit Missions

In pursuit missions, Turumtay tries to keep closer to Target UAV as like Close Range Tactics of Red Team [38] and try to keep enemy UAV in Turumtay's field of view.

4.3.1 Case 7

Case 7 is a case that simulates pursuit missions. In this case, the green zone is a triangle located in the upper left corner, and the red zone is located in the bottom right. Turumtay stands at the right side of the green zone, and Enemy UAV stands at the left side of the red zone, as shown in Figure 4.13

In the mission, Turumtay first enters the green zone as soon as possible, and after that, it approaches the Target UAV while standing in the green zone. When it is close to Target sufficiently, which is around 250 meters as described in Table 3.3, Turumtay rotates in order to keep Target UAV in the Turumtay's field of view.

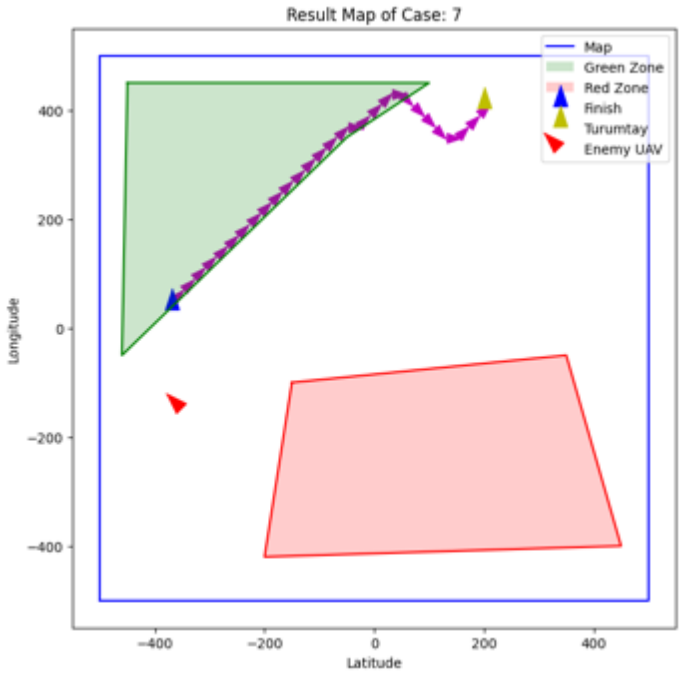

Figure 4.13 Case 7 Result Map

In terms of the reward chart, which is visualized in Figure 4.14, the reward decreases until Turumtay reaches the green zone. In the green zone, the decrease of the reward is slight as compared to the gray zone. Finally, when the distance between Turumtay and Target is close enough, and Target UAV is in the field of view of Turumtay, Turumtay gains a maximum reward.

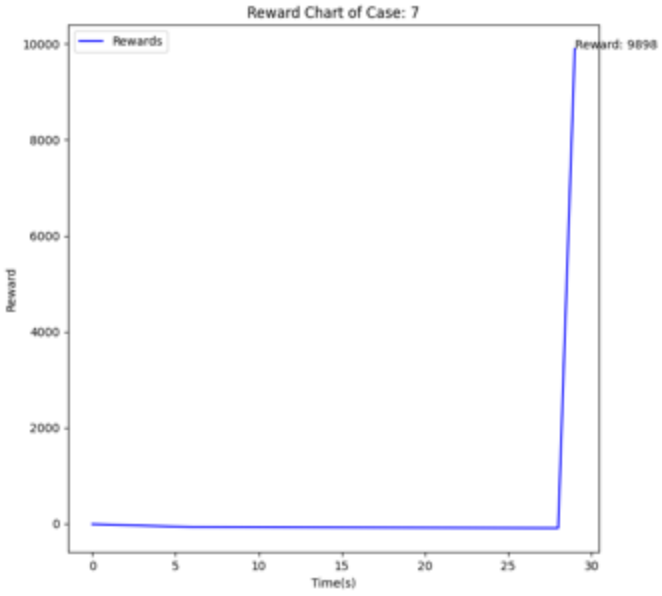

Figure 4.14 Case 7 Reward Chart

4.3.2 Case 8

Case 8 is also a pursuit mission. In Case 8, there is a large red zone triangle located in the center of the map, and a concave green zone polygon surrounding the red zone, which is shown in Figure 4.15

Turumtay goes around the red zone from the upper side of it and enters the green zone. Turumtay continues its move in the green zone, and it passes red zones borders and rotates to Target. In Turumtay's last moves, Turumtay moves diagonally and decreases the distance between Turumtay and Target UAV. In the end, Turumtay rotated again and took a place to satisfy the appropriate field of view by heading up to Target UAV.

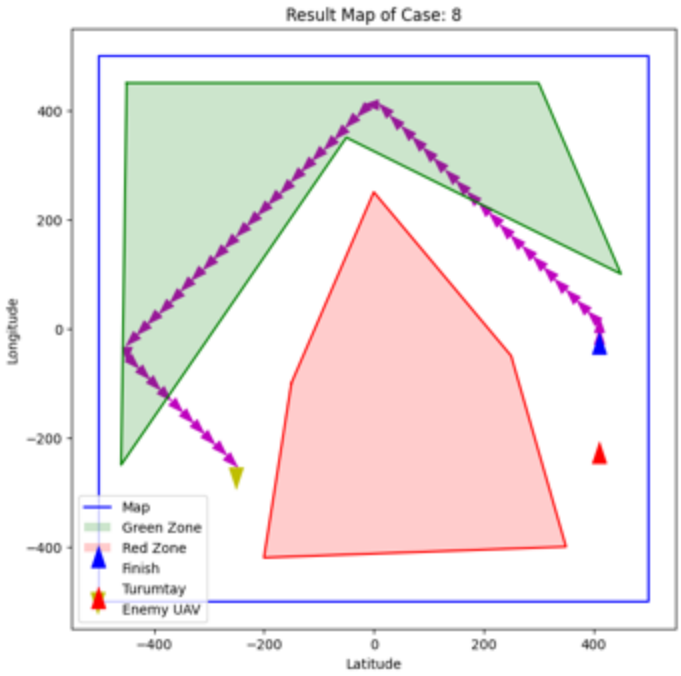

Figure 4.15 Case 8 Result Map

In the reward chart of this case, the path of Turumtay can be interpreted easily. First, the reward decreases while Turumtay is moving in a gray zone. When Turumtay enters the green zone, the decrease rate of the reward falls, and after that, it rises again after exiting from the green zone. Turumtay exits from the green zone in order to approach the enemy. When it is close enough to the enemy with a proper heading angle, Turumtay gains 10000 points, and the reward increases suddenly.

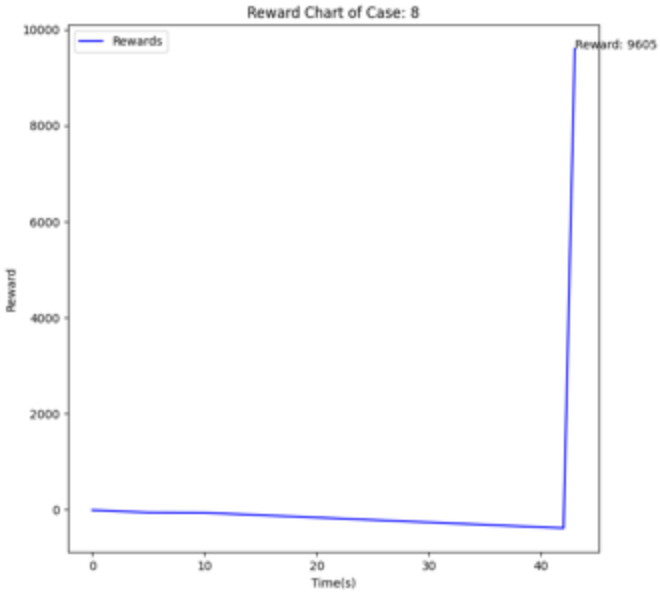

Figure 4.16 Case 8 Reward Chart

4.3.3 Case 9

In Case 9, the map is divided into two equal sides. The red zone is shaped like a rectangle and is located on the left side of the map. On the other hand, the green zone is on the right side of the map as a rectangle. This case is visualized in Figure 4.17. Turumtay and Enemy UAV are standing at the center, in a face-to-face position. When the mission is started, Turumtay first went to the green zone and moved to the top of the map in order to approach the Enemy from the back of it. If Turumtay had gone directly to the enemy initially, the enemy aspect angle would have been equal to forwarding, and Turumtay would have been in the field of view of the Enemy, and this would be a cause of penalty in rewards. After Turumtay goes behind Enemy, it approaches Enemy until the distance between them is close enough for reward gain.

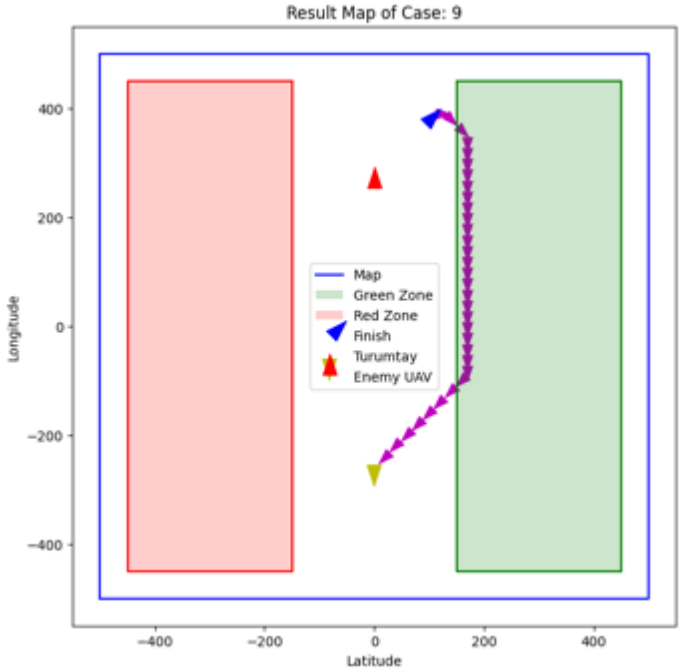

Figure 4.17 Case 9 Result Map

The reward chart of Case 9 is similar to Case 7 and Case 8, as can be seen in Figure 4.18. In the first movements of Turumtay, the reward was decreasing, and the speed of the decrease slowed down after Turumtay entered the green zone. After some steady movements in the green zone, Turumtay approached to Enemy UAV from behind and rotated in order to force the Enemy centered in the Turumtay's field of view. Finally, Turumtay got close enough with proper heading angles, so it gained maximum reward, and the chart is rapidly increased.

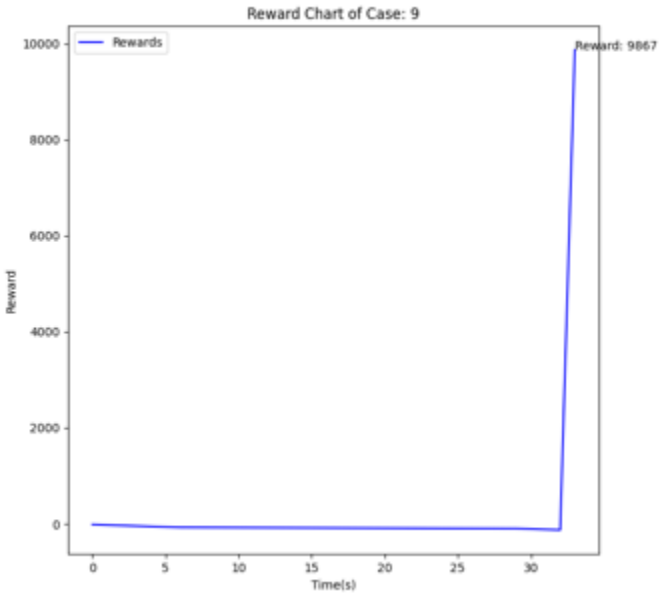

Figure 4.18 Case 9 Reward Chart

5 RESULTS AND EVALUATION

In Reinforcement Learning problems, performance criteria are defined by using reward tables, so each problem has different success metrics according to their reward functions [23], [30], [38]. In Path planning problems, the success of the algorithm is measured by minimal step counts as a reward function [23]. On the other hand, in pursuit-evasion missions, the success of the algorithm is measured by hitting enemy UAVs o being hit by the Enemy as a reward function [28], [30], [38].

In this study, problem definition covers these two types of problems which are path planning regarding zone types and pursuit-evasion missions. So, the result is not distinct as a win or lose because of the different rewards on different missions. In basic pursuit-evasion missions, the definition of success is as clear as hitting an enemy UAV. However, when the problem is getting complex and includes other factors such as green or red zones, the measurement of the success can only be calculated by gained rewards. The mission is successful as gained total reward. In future studies, additional algorithms may be developed to compare their performance results with the algorithm developed in this thesis. The key point is that the newly proposed algorithm should be run with the same state sets and reward functions in this study, and compare its average rewards with the rewards in Table 5.1

Vlahov [30] implemented software that used Reinforcement Learning in pursuit-evasion missions; in that study, 1050 test cases were simulated, and as a result, RL software won 805 cases out of 1050. Like Vlahov, 500 random cases were simulated for each mission type, and the average reward values of these cases are listed in Table 5.1. In patrol missions, the average reward value is 9636, in evasion missions 9939, and in pursuit missions 9739 out of 10000.

Table 5.1 Average Rewards for Each Mission

Mission	Average Reward (out of 10000)
Patrol	9636
Evasion	9939
Pursuit	9739

McGrew [28] and Vlahov [30] demonstrate the performance of their proposed software by using simulation studies and sample case results, similar to cases of this study which are analyzed in the SIMULATION STUDIES chapter. Toubman [38] compares its result by using different tactics for pursuit missions and plots rewards vs. case charts of these tactics. In this study, different mission types are used, and their rewards vs. case charts are shown in Figure 5.2, Figure 5.4 and Figure 5.6.

5.1 Patrol Missions

In order to evaluate patrol missions, 500 cases were randomly created, as in Figure 5.1. In these 500 cases, Turumtay started at random locations and simulated with different patrol points, and in these cases, Enemy UAV is also located randomly.

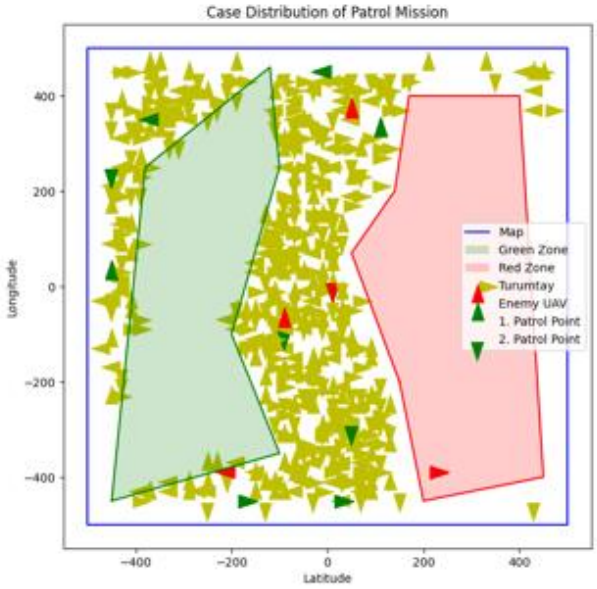

Figure 5.1 Case Distribution of Patrol Missions

On average, Turumtay gained 9636 rewards out of 10000 in patrol missions. The results of gained rewards on 500 patrol mission cases are presented in Figure 5.2. Individual rewards are spread between 9250 and 9750 points, and the mean of 500 cases is 9636. In Patrol Missions, Turumtay firstly goes first patrol point, second patrol point and returns to first patrol point, and at each step on the path, Turumtay penalized with -1. So due to the long path in patrol missions, rewards are smaller than Evasion and Pursuit Missions.

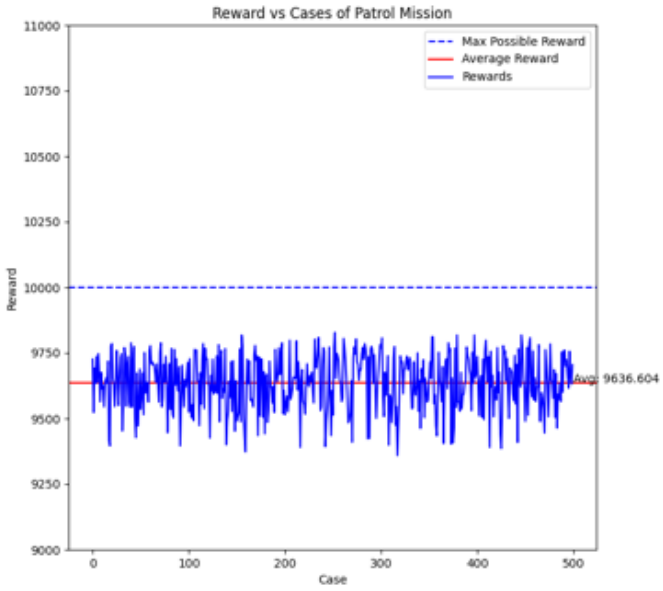

Figure 5.2 Reward vs Cases Chart of Patrol Missions

5.2 Evasion Missions

To assess evasion missions, similar to patrol missions, 500 random cases are used to test the performance of the Mission Control Software of Turumtay. These random cases are distributed as in Figure 5.3. Turumtay and Enemy UAVs are randomly located on the map, and in each case, Turumtay aimed to reach Green Zone as soon as possible while escaping from Enemy UAV.

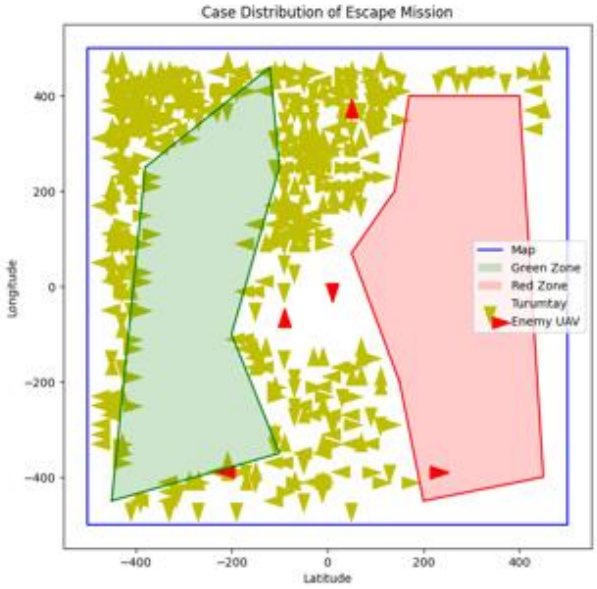

Figure 5.3 Case Distribution of Escape Missions

Turumtay gained 9939 reward points out of 10000 in evasion missions, and rewards are between 9750 and 10000 generally. In these cases, the initial position of the Turumtay effect gained reward. So, it is inevitable that when Turumtay is located too far from Green Zone, the gained reward can be smaller than 9000. However, the number of these cases is negligible, as can be shown in Figure 5.4. The mean of reward in these 500 cases of Escape mission is 9939 out of 1000.

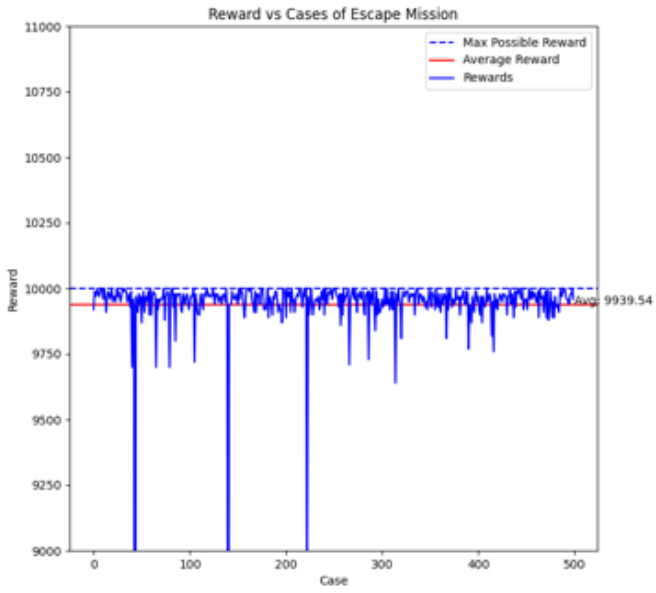

Figure 5.4 Reward vs Cases Chart of Escape Missions

5.3 Pursuit Missions

In pursuit missions, 500 random cases were simulated to analyze the performance of the developed algorithm. Distributions of the location of Turumtay and Enemy UAV are illustrated in Figure 5.5. Turumtay try to close to Enemy UAV and keep Enemy UAV in Turumtay's field of view.

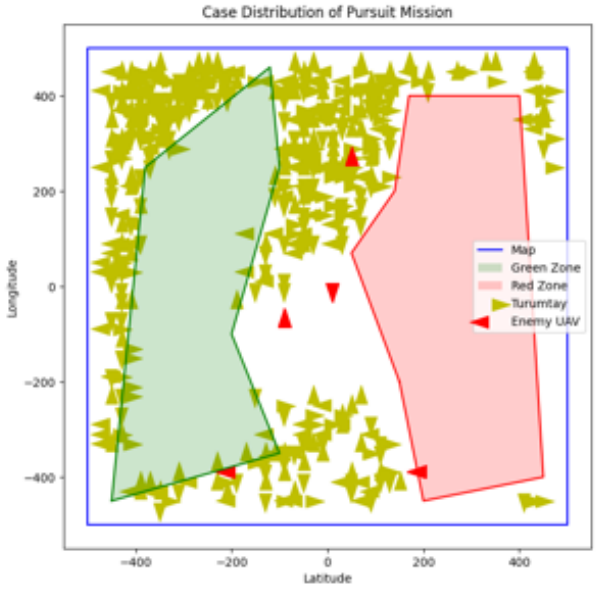

Figure 5.5 Case Distribution of Pursuit Missions

The mean value of reward gained in 500 cases of pursuit mission is 9739. Rewards in these cases are between 9500 and 10000 for the most part of cases, as can be seen from Figure 5.6. 9739 out of 10000 points on average shows that mission control can find a path in order to close Enemy UAV from behind and keep Enemy in Turumtay's field of view in each case of pursuit missions.

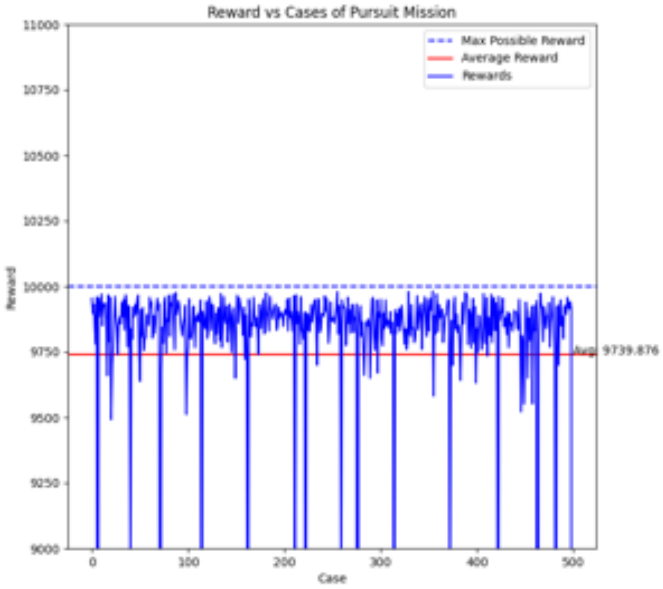

Figure 5.6 Reward vs Cases Chart of Pursuit Missions

6 CONCLUSION AND FUTURE WORKS

In summary, this thesis proposed software as a solution for pursuit-evasion problems of Mini UAVs in areas that include green, gray, and neutral zones. In order to perform this solution, Ground Control, Mission Control, and Flight Control Software are implemented, and the design structure of this software is analyzed in detail. Mission Control Software used Reinforcement Learning methods and Q Learning algorithms to find optimal paths while performing missions. In Q learning, States, Actions, and Reward Tables are created according to the goal mission. Simulation Studies of these algorithms are examined by using example cases of three different cases for each mission type; a total, nine cases are analyzed.

Path planning problems and pursuit-evasion problems are separately popular topics for the Autonomous UAV Software area. However, in this study, path planning with avoidance or threat zones problems and pursuit-evasion problems are taken into account as an integrated problem.

In this thesis, Software Design and Architecture, which can be used in Mini UAVs to perform autonomous pursuit-evasion missions, are implemented. Mission Control which manages all autonomous scenarios for pursuing and escaping while considering green, gray, and red zones, are examined in detail.

This software solution presents a concept design for autonomous missions, thanks to the advantages of the Q Learning algorithms. The developed Mission Control Software can easily be adapted to other problems by initializing proper reward tables and action sets for the desired mission types.

The results of developed software in this thesis are analyzed in SIMULATION STUDIES and RESULTS AND EVALUATION chapters. The findings of this study show that autonomous software to perform pursuit-evasion missions in combat areas, including friendly zones or enemy zones, can be developed by using Reinforcement Learning and Q Learning methods.

In order to improve this algorithm, one-to-many target support is an important milestone. There may be more than one target UAV in real-life scenarios, and all of them need to be counted while calculating optimal paths. For example, while pursuing a target

UAV, positions of other UAVs are also critical because of their potential threats. In this thesis, the mission control algorithm focused on one-to-one combats, and in future works, this can be extended by using multiple target UAVs.

In future studies, movements of the Target UAVs may be estimated, and Q Learning algorithms may be executed by considering the prospective location of the Target UAV rather than the actual location so that action can be taken in advance. Thanks to these estimations, the algorithm's performance may well be increased.

Furthermore, in this thesis, information about Target UAV is simulated in Ground Control software, and it is assumed that this information is gathered from radars or other sensors connected with Ground Control Stations. As future work, gathering information about target UAV may be carried out by friendly UAV by its own means. In order to gather this information, image processing by using cameras on the UAV can be used. This method may be useful in Micro or Mini UAVs because pursuit-evasion missions in these categories are done in small zones, and locations of other UAVs can be gathered from the field of views cameras.

REFERENCES

[1] K. L. B. Cook, "The Silent Force Multiplier: The History and Role of UAVs in Warfare," in *2007 IEEE Aerospace Conference*, 2007.

[2] S. Kahvecioglu and H. Oktal, "Historical Development of UAV Technologies in the World: The Case of Turkey," in *Sustainable Aviation: Energy and Environmental Issues*, Springer, 2016, pp. 323-331.

[3] S. Düz, "Türkiye'nin Gökyüzündeki Yeni Gücü İHA'lar," SETA, 2021.

[4] "History of Baykar," BAYKAR Unmanned Aerial Vehicle Systems, [Online]. Available: https://baykardefence.com/History.html. [Accessed 7 December 2021].

[5] "Bayraktar TB2," BAYKAR Unmanned Aerial Vehicle Systems, [Online]. Available: https://baykardefence.com/uav-15.html. [Accessed 7 December 2021].

[6] "AKINCI UAV," BAYKAR Unmanned Aerial Vehicle Systems, [Online]. Available: https://www.baykarsavunma.com/iha-14.html. [Accessed 7 December 2021].

[7] "IHA Sistemleri Yol Haritası," Presidency of Defence Industries of Turkey, 2012.

[8] "Black Hornet PRS," FLIR Systems, [Online]. Available: https://www.flir.com/products/black-hornet-prs/. [Accessed 7 December 2021].

[9] "Bayraktar Mini UAV," SSB Product Catalog, [Online]. Available: https://www.ssb.gov.tr/urunkatalog/en/146/. [Accessed 7 December 2021].

[10] "Bayraktar Mini UAV," BAYKAR Unmanned Aerial Vehicle Systems, [Online]. Available: https://baykardefence.com/uav-16.html. [Accessed 7 December 2021].

[11] "Elbit Hermes-90," Hermes-90 Elbit Systems, [Online]. Available: https://elbitsystems.com/product/hermes-90/. [Accessed 7 December 2021].

[12] C. Kasapoğlu and B. Kırdemir, "The Rising Drone Power: Turkey On The Eve Of Its Military Breakthrough," *EDAM Foreign Policy and Security,* vol. 4, 2018.

[13] North Atlantic Treaty Organization, "UAS Tactical Pocket Guide," NATO Standardization Agency, 2014.

[14] "ANKA," Turkish Aerospace Industry, [Online]. Available: https://www.tusas.com/urun/anka. [Accessed 7 December 2021].

[15] Department of the Army of The United States, "Electronic Warfare Techniques," Army Techniques Publication, 2019.

[16] J. Stillion, "Trends in Air-to-Air Combat Implications for Future Air Superiority," Center for Strategic and Budgetary Assessments, 2015.

[17] K. B. Ragsdale, Wings over the Mexican Border: Pioneer Military Aviation in the Big Bend, University of Texas Press, 1997.

[18] G. Gaitanakis, G. Limnaios and K. Zikidis, "On the use of AESA (Active Electronically Scanned Array) Radar and IRST (InfraRed Search&Track) System to Detect and Track Low Observable Threats," in *MATEC Web of Conferences*, 2019.

[19] L. Fu, F. Xie, D. Wang and G. Meng, "The overview for UAV Air-Combat Decision method," in *The 26th Chinese Control and Decision Conference (2014 CCDC)*, 2014.

[20] A. H. Pashakhanlou, "AI, autonomy, and airpower: the end of pilots?," *Defence Studies*, vol. 19, pp. 337-352, 2019.

[21] M. W. Byrnes, "Nightfall: Machine Autonomy in Air-to-Air Combat," *Air & Space Power Journal* , Vols. May-June, 2014.

[22] C. Sun, Y.-C. Liu, R. Dai and D. Grymin, "Two Approaches for Path Planning of Unmanned Aerial Vehicles with Avoidance Zones," in *Journal of Guidance, Control, and Dynamics*, 2017.

[23] S. You, C. Wan and R. Dai, "Iterative Learning Optimization for UAV Path Planning with Avoidance Zones," in *2019 American Control Conference (ACC)*, 2019.

[24] C. Yan and X. Xiang, "A Path Planning Algorithm for UAV Based on Improved Q-Learning," in *2018 2nd International Conference on Robotics and Automation Sciences (ICRAS)*, 2018.

[25] A. Gupta, A. K. Gupta, C. Bocaniala and V. V. S. S. Sastry, "Avoidance of threat zone by UAV for automated navigation," in *2008 Annual IEEE India Conference*, 2008.

[26] "RPA Vector: Vision and Enabling Concepts 2013-2038," United States Air Force, 2014.

[27] H. X. Pham, H. M. La, D. Feil-Seifer and L. V. Nguyen, "Reinforcement Learning for Autonomous UAV Navigation Using Function Approximation,"

2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pp. 1-6, 2018.

[28] J. McGrew, J. How, L. Bush, B. Williams and N. Roy, "Air-Combat Strategy Using Approximate Dynamic Programming," *Journal of Guidance, Control, and Dynamics*, 2010.

[29] X. Ma, L. Xia and Q. Zhao, "Air-Combat Strategy Using Deep Q-Learning," *2018 Chinese Automation Congress (CAC)*, pp. 3952-3957, 2018.

[30] B. Vlahov, E. Squires, L. Strickland and C. Pippin, "On Developing a UAV Pursuit-Evasion Policy Using Reinforcement Learning," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018.

[31] B. Zhou, W. Wang, Z. Wang and B. Ding, "Neural Q Learning Algorithm based UAV Obstacle Avoidance," in *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, 2018.

[32] S. Tosunoğlu, "Autopilot System And Ground Station Software For UAV's," Istanbul Technical University Department Of Aeronautics And Astronautics, 2013.

[33] N. Imanberdiyev, C. Fu, E. Kayacan and I. Chen, "Autonomous navigation of UAV by using real-time model-based reinforcement learning," in *14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2016.

[34] T. Zhou, M. Chen and J. Zou, "Data Fusion of Air Combat Based on Reinforcement Learning," in *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2019.

[35] S.Manju and M.Punithavalli, "An Analysis of Q-Learning Algorithms with Strategies of Reward Function," *International Journal on Computer Science and Engineering (IJCSE),* vol. 3, no. 2, pp. 814-820, 2011.

[36] Z. Yijing, Z. Zheng, Z. Xiaoyi and L. Yang, "Q learning algorithm based UAV path learning and obstacle avoidence approach," in *2017 36th Chinese Control Conference (CCC),* 2017.

[37] K. Zhao and C. Huang, "Air combat situation assessment for UAV based on improved decision tree," in *2018 Chinese Control And Decision Conference (CCDC),* 2018.

[38] A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat and J. V. D. Herik, "Rewarding Air Combat Behavior in Training Simulations," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, Hong Kong, China, 2015.