



Teknoloji Fakültesi



MARMARA ÜNİVERSİTESİ TEKNOLOJİ FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ

Koordineli Saldırı Görevi için Otonom Çoklu İHA
ve Akıllı Mühimmat Gündüm Sistemlerinin Geliştirilmesi

PROJE YAZARI

Ammar ABDURRAUF, Ahmet KURT, Serdar YILDIRIM

170421930, 170421022, 170421043

DANIŞMAN

Dr. Öğr. Üyesi Eyüp Emre ÜLKÜ

İstanbul, 2025

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Öğrencileri Ammar ABDURRAUF, Ahmet KURT, Serdar YILDIRIM tarafından “Koordineli Saldırı Görevi için Otonom Çoklu İHA ve Akıllı Mühimmat Gündüm Sistemlerinin Geliştirilmesi” başlıklı bitirme projesi çalışması,/..../..... tarihinde sunulmuş ve jüri üyeleri tarafından başarılı bulunmuştur.

Jüri Üyeleri

Dr. Öğr. Üyesi Eyüp Emre ÜLKÜ (Danışman)

Marmara Üniversitesi (İMZA)

Prof.Dr. Şahin UYAVER

Marmara Üniversitesi (İMZA)

Arş.Gör. Duygu Kayaoglu

Marmara Üniversitesi (İMZA)

İÇİNDEKİLER

KISALTMALAR LİSTESİ.....	4
ŞEKİL LİSTESİ.....	5
TABLO LİSTESİ.....	7
ÖZET	8
1. GİRİŞ	10
1.1 Bitirme Projesinin Amacı.....	10
1.1.1 Çoklu İHA Stratejileri	11
1.1.2 Füze Kontrol ve Navigasyon Sistemleri Stratejileri	11
1.2 Literatür Özeti	12
2. MATERİYAL VE YÖNTEM.....	15
2.1 Materyal	15
2.1.1 Simülasyon ortamı – Simplerockets 2 (Juno: New Origins)	16
2.1.2 Vizzy programlama (IDE)	16
2.1.3 Python.....	16
2.1.4 SR2Logger mod.....	17
2.1.5 MATLAB	17
2.2 Yöntem	18
2.2.1 Veri İletişimi.....	18
2.2.2 İki Boyutlu Düzlemden Çoklu İHA Koordinasyonu ve Otonom Görev Ataması....	20
2.2.3 Python aracılığıyla 3D İHA simülasyonunun gerçek zamanlı kontrolü.....	29
2.2.4 Waypoint Tabanlı Otonom Navigasyon Sistemi	31
2.2.5 NavigationController ile Yaw Tabanlı Roll ve Pitch Açılarının Hesaplanması.....	32
2.2.6 PID Tabanlı Stabilizasyon ve Kontrol Mekanizması	33
2.2.7 Çok İş Parçacıklı Kontrol ve Telemetri Yönetimi.....	34
2.2.8 Python Üzerinden Füze Kontrolü	36
2.2.9 Terminal Yaw Açısı Kontrolü	38
2.2.10 Terminal Pitch Açısı Kontrolü	40
2.2.11 Hedef Tespiti İçin YOLOv8 Modelinin Eğitimi	43
2.2.12 Füze Roll Stabilizasyon Sistemi	47
3. MODEL EĞİTİMİ VE SİMÜLASYON SONUÇLARI	58
3.1 Füze Sisteminin Test Simülasyonları.....	58
3.1.1 Terminal Yaw ve Pitch Açısı Test Simülasyonu.....	58
3.1.2 YOLOv8 Modeli ile Pitch Açısı Kontrolü	61
3.1.3 Füze Roll Stabilizasyon Sistemi Testi	62

4. BULGULAR VE TARTIŞMA	67
5. EKLER.....	70
6. KAYNAKLAR.....	76

KISALTMALAR/ABBREVIATIONS

DDPG: Deep Deterministic Policy Gradient

GPS: Global Positioning System

GBAD: Ground-Based Air Defense

NFZ: No-Fly Zone

HSS: Hava Savunma Sistemi

IMU: Inertial Measurement Unit

IHA: İnsansız Hava Aracı

LOS: Line of Sight

LSTM: Long Short-Term Memory

OOP: Object Oriented Programming

PSO: Particle Swarm Optimization

MPC: Model Predictive Control

PID: Proportional Integral Derivative

PNG: Proportional Navigation Guidance

PWM: Pulse-width Modulation

RL: Reinforcement Learning

DL: Deep Learning

SAM: Surface-to-Air Missile

SR2: Simplerockets 2 (uçuş simulatörü)

UDP: User Datagram Protocol

YOLO: You Only Look Once

YSA: Yapay Sinir Ağları

YZ: Yapay Zeka

ŞEKİL LİSTESİ

	Sayfa
Şekil 2.1. AIM-54C by Hughes Aircraft	15
Şekil 2.2. AIM-54C Phoenix by NoLuja	15
Şekil 2.3. SR2Logger kullanarak UDP üzerinden veri göndermek için Vizzy programı	19
Şekil 2.4. UDP paketlerinden veri çıkarmak için readPacket fonksiyonu	20
Şekil 2.5. Tek İHA ile kalkış ve nokta takip simülasyonu	22
Şekil 2.6. Çoklu İHA ile konsensus algoritması denemesi	23
Şekil 2.7. "Lider direkt saldırı" senaryosunda angajman anı.....	25
Şekil 2.8. "Kanattan Kuşatma ve Yakınsama" senaryosunda angajman anı.....	26
Şekil 2.9. Simülasyona NFZ dahil edildiğinde İHA'ların kaçınma davranışları.....	27
Şekil 2.10. Simülasyona NFZ dahil edilmediğinde İHA'ların davranışları	27
Şekil 2.11. Simülasyona NFZ dahil edilmediğinde İHA'ların davranışları	28
Şekil 2.12. Simplerockets 2 simülasyon ortamında TB2 İHA modeli	29
Şekil 2.13. İHA simülasyon sonucu izlenen rota	31
Şekil 2.14. İHA simülasyon sonucu roll-pitch-yaw değer değişimleri	32
Şekil 2.15. İHA hız grafiği.....	34
Şekil 2.16. İHA simülasyon sonucu irtifa değişimi	34
Şekil 2.17. İHA simülasyon sonucu sırasıyla 4 hedef noktasının uzaklık değişimi	35
Şekil 2.18. Python'dan SR2'ye füze kontrol akış şeması	36
Şekil 2.19. İstenilen terminal yaw açıları ve uçuş yörüngesi.....	39
Şekil 2.20. Python'da Bezier eğrisi kodu ve görselleştirilmesi.....	40
Şekil 2.21. Hedef görsellerin ve sınırlayıcı kutularının örnekleri.....	43
Şekil 2.22. YOLOv8n modelin eğitim metrikleri	46
Şekil 2.23. YOLOv8m modelin eğitim metrikleri	46
Şekil 2.24. Elevator kontrol girdisini üreten YSA modeli	46
Şekil 2.25. LSTM hücre yapısı	51
Şekil 2.26. DDPG model yapısı ve algoritması	53
Şekil 2.27. DDPG model eğitimin ortalama ödülü	57
Şekil 3.1. 90° terminal yaw için planlanan uçuş yörüngesi ve noktaları	59
Şekil 3.2. 180° terminal yaw için planlanan uçuş yörüngesi ve noktaları	59
Şekil 3.3. 30°, 60° ve 80° terminal pitch için boylam ve irtifa grafiği	60
Şekil 3.4. Pitch açılarının karşılaştırılması.....	62
Şekil 3.5. Elevator girdi karşılaştırılması.....	62
Şekil 3.6. Roll açılarının ve aileron girdi karşılaştırılması	65
Şekil 5.1. 90° terminal yaw için simülasyon sonuçları.....	70
Şekil 5.2. 180° terminal yaw için simülasyon sonuçları.....	70
Şekil 5.3. 30° terminal pitch için simülasyon sonuçları	71
Şekil 5.4. 60° terminal pitch için simülasyon sonuçları	71
Şekil 5.5. 80° terminal pitch için simülasyon sonuçları	72
Şekil 5.6. Yuvarlanma stabilizasyonu senaryo 2 uçuş yörüngelerin karşılaştırılması	73
Şekil 5.7. Yuvarlanma stabilizasyonu senaryo 2 roll açılarının karşılaştırılması.....	73

Şekil 5.8. Yuvarlanma stabilizasyonu senaryo 2 aileron girdi karşılaştırılması.....	74
Şekil 5.9. Yuvarlanma stabilizasyonu senaryo 3 uçuş yörüngelerin karşılaştırılması	74
Şekil 5.10. Yuvarlanma stabilizasyonu senaryo 3 roll açılarının karşılaştırılması.....	75
Şekil 5.11. Yuvarlanma stabilizasyonu senaryo 3 aileron girdi karşılaştırılması.....	75

TABLO LİSTESİ

	Sayfa
Tablo 2.1. Kullanılan Python kütüphaneleri	17
Tablo 2.2. İHA simülasyon sonucu genel veriler	36
Tablo 3.1. Hedefe ulaşmadan önceki füze pitch açı değerlerinin son 10 örneği	60
Tablo 3.2. Sim. 1: Roll Açısı Hata Değerleri Karşılaştırması.....	63
Tablo 3.3. Sim. 2: Roll Kontrolörleri için Hata Metrikleri ve Hedef Sapma Karşılaştırması	66
Tablo 3.4. Sim. 3: Roll Kontrolörleri için Hata Metrikleri ve Hedef Sapma Karşılaştırması .	66

ÖZET

Modern savaş ortamlarında yer tabanlı hava savunma sistemlerinin gelişimi, hava saldırısının etkinliğini önemli ölçüde azaltmakta ve muharip uçaklar ve pilotları için ciddi bir tehdit oluşturmaktadır. Bu durum, hava sahasına sızmayı ve düşman hedeflerine yüksek hassasiyetle ulaşmayı zorlaştırmaktadır. Bu nedenle, insansız hava araçları (İHA) sürüleri ve akıllı füze sistemleri gibi insansız ve otonom sistemlere dayalı yeni saldırı stratejilerinin geliştirilmesi gerekliliği ortaya çıkmıştır. Bu proje, düşman hava savunma sistemlerine karşı çoklu İHA ve akıllı füzeler kullanarak koordineli saldırı stratejileri geliştirmeyi amaçlamaktadır. Çalışma kapsamında, geleneksel kontrol ve hareket planlama algoritmalarını yapay zekâ tabanlı yöntemlerle birleştirerek, füze ve İHA'lara yönelik kontrol ve navigasyon sistemlerinin etkinliğini artırmayı amaçlamaktadır.

İlk aşamada, İHA'ların formasyon kontrolü, görev paylaşımı ve iletişim gibi alt görevleri MATLAB ve Python ortamlarında geliştirilen 2 boyutlu simülasyonlarda modellenmiş ve analiz edilmiştir. Bu kapsamında, İHA'nın hava savunma sistemine tespit edilmeden yaklaşabilmesi için rota planlama, hedef koordinatlarına yönelme ve hedefle etkileşim gibi görevler dikkate alınmıştır. İHA, belirli bir mesafeye kadar hedefe yaklaşmakta ve bu noktada hedefle etkileşimden sonra güvenli bir rotaya yönelikerek bölgeden uzaklaşmaktadır. Simülasyon sonuçları, geliştirilen kontrol algoritmalarının ve görev planlama stratejilerinin, İHA seviyesinde yüksek etkililik ve görev başarısı sunduğunu ortaya koymaktadır.

Proportional Navigation algoritması kullanılarak füzenin terminal pitch açısı kontrol edilmiş ve Simplerockets 2 (SR2) simülasyon ortamında başarıyla test edilmiştir. Ardından, füzenin yuvarlanması (roll) kararsızlığını önlemek için PID kontrolörü ve yapay zeka tabanlı kontrolörleri kullanılarak anti-roll stabilizasyon sistemi geliştirilmiştir. Ayrıca, füzenin terminal yaw açısını kontrol etmek için Bézier eğrisi tabanlı bir rota planlama algoritması tasarlanarak optimal uçuş rotaları elde edilmiştir. Son olarak, GPS erişiminin kısıtlı olduğu senaryoları desteklemek amacıyla, YOLOv8 algoritması tabanlı bir nesne tanıma sistemi füze sistemine entegre edilmiştir. Bu çalışma ile, otonom sistemlerin hava savunma engellerini aşma kapasitesine yönelik kapsamlı bir çözüm önerisi sunulmaktadır.

ABSTRACT

In modern warfare, the advancement of ground-based air defense systems significantly reduces the effectiveness of air strikes and presents a critical threat to fighter aircraft and their pilots. This condition makes it difficult to infiltrate the airspace and reach enemy targets with high precision. Therefore, the need to develop new attacking strategies based on unmanned and autonomous systems such as unmanned aerial vehicle (UAV) swarms and smart missile systems has emerged. This project aims to develop coordinated attack strategies against enemy air defense systems using multiple UAVs and smart missiles. Within the scope of the study, it aims to increase the effectiveness of control and navigation systems for missiles and UAVs by combining traditional control and motion planning algorithms with artificial intelligence-based methods.

In the initial phase, sub-tasks such as formation control, task allocation, and communication among UAVs were modeled and analyzed through two-dimensional simulations developed in MATLAB and Python environments. Within this scope, tasks such as route planning, navigation toward target coordinates, and interaction with the target were considered to enable UAVs to approach air defense systems without detection. The UAV approaches the target up to a certain distance, performs the interaction, and then retreats from the area by following a secure route. Simulation results demonstrate that the developed control algorithms and mission planning strategies offer high efficiency and mission success at the UAV level.

The terminal pitch angle of the missile was controlled using the Proportional Navigation algorithm and successfully tested in the Simplerockets 2 (SR2) simulation environment. Subsequently, an anti-roll stabilization system was developed to prevent roll instability in the missile by employing both a conventional PID controller and artificial intelligence-based controllers. Furthermore, a Bézier curve-based trajectory planning algorithm was designed to control the missile's terminal yaw angle, allowing the generation of optimal flight paths. Finally, to support scenarios with limited GPS access, an object detection system based on the YOLOv8 algorithm was integrated into the missile system. This study presents a comprehensive solution for enhancing the capability of autonomous systems in overcoming air defense obstacles.

1. GİRİŞ

Modern savaş ortamlarında hava üstünlüğü sağlamak, giderek daha karmaşık hale gelen savunma teknolojileri nedeniyle her geçen gün zorlaşmaktadır. Özellikle kara tabanlı hava savunma sistemleri (Ground-Based Air Defense - GBAD), gelişmiş radar ve Yüzeyden Havaya Füze (Surface-to-Air Missile - SAM) sistemleri ile donatılarak, geleneksel hava saldırılara karşı yüksek düzeyde direnç göstermektedir. Bu sistemlerin etkinliği, hem insanlı savaş uçakları hem de füzeler için ciddi tehditler oluşturmaktadır, hedef bölgelere sızmayı ve operasyonel başarıyı önemli ölçüde zorlaştırmaktadır.

Bu durum, yeni nesil saldırı konseptlerinin geliştirilmesini zorunlu kılmıştır. Özellikle insan kaybını minimize etmek ve GBAD sistemlerinin etkinliğini kırmak amacıyla, insansız hava araçları (İHA) ve sürü İHA (swarm drone) teknolojileri ön plana çıkmaktadır. Bu sistemler, çoklu vektörden saldırı yapabilmeleri, hedefe eşzamanlı yaklaşabilmeleri ve yapay zekâ destekli karar mekanizmalarıyla donatılabilmeleri sayesinde, geleneksel yöntemlerin sınırlarını aşan yeni imkânlar sunmaktadır. Bununla birlikte, akıllı mühimmatlara entegre edilen gelişmiş güdümlü ve navigasyon sistemleri de düşman savunmalarını bypass edebilme yeteneği açısından kritik öneme sahiptir.

Bu çalışma, iki ana odak noktası etrafında şekillenmektedir: (1) sürü İHA yapılarının otonom görev paylaşımı, formasyon kontrolü ve rota planlama algoritmalarıyla geliştirilmesi ve (2) füze sistemlerinin terminal yaw/pitch kontrolü, roll stabilizasyonu ve GPS olmayan ortamlarda hedef tespiti gibi görevleri yerine getirebilmesini sağlayacak modern güdümlü sistemlerinin tasarımı. Yapay zekâ destekli yöntemlerin klasik kontrol algoritmalarıyla bütünlendirildiği bu yaklaşım sayesinde, yüksek riskli saldırı görevlerinde otonom sistemlerin etkinliğini artırmak hedeflenmektedir.

Bu kapsamda, çalışma yalnızca teknik çözüm önerileri sunmakla kalmayıp aynı zamanda GBAD sistemlerinin aşıldığı senaryolara odaklanan güncel literatürü de kapsamlı biçimde analiz ederek önerilen yöntemlerin mevcut çalışmalarından farklılığı noktaları ortaya koymaktadır.

1.1 Bitirme Projesinin Amacı

Bu çalışma, modern savunma senaryolarında öne çıkan iki temel alana odaklanmaktadır: çoklu İHA (İnsansız Hava Aracı) stratejilerinin geliştirilmesi ve akıllı füze sistemlerine yönelik kontrol ile navigasyon algoritmalarının iyileştirilmesi. Proje kapsamında, bu iki alan özelinde

geliştirilecek yöntemler aracılığıyla düşman hava savunma sistemlerine karşı daha etkili, koordineli ve otonom saldırısı stratejilerinin oluşturulması hedeflenmektedir.

1.1.1 Çoklu İHA Stratejileri

Çoklu İHA sistemleri, modern savunma ve saldırısı operasyonlarında giderek daha kritik bir rol oynamaktadır. Bu sistemler, birden fazla İHA'nın senkronize bir şekilde çalışarak hedeflere ulaşmasını, görevleri paylaşmasını ve dinamik ortamlara hızla adapte olmasını gerektirir. Bu çalışmanın temel amacı, çoklu İHA'ların otonom karar verme yeteneklerini geliştirerek, düşman hava savunma sistemlerine (HSS) karşı etkili ve koordineli saldırular gerçekleştirmelerini sağlamaktır.

Bu kapsamında, çoklu İHA sistemleri için geliştirilecek stratejiler, otonom rota planlama ve görev tahisi, formasyon kontrolü ve koordinasyon, dinamik ortamlara adaptasyon ve senkronize saldırısı stratejileri gibi hedeflere odaklanmaktadır. İHA'ların, düşman savunma sistemlerini atlatacak şekilde optimal rotalar belirlemesi ve görevleri otonom olarak paylaşması, radar sistemlerinden kaçınmasını ve hedeflere yüksek doğrulukla ulaşmasını sağlayacaktır. Ayrıca, İHA'ların uçuş sırasında optimal formasyonu koruyarak birbirleriyle etkili bir şekilde iletişim kurması, özellikle dinamik tehdit ortamlarında uyum içinde çalışmalarını mümkün kılacaktır.

Değişen çevresel koşullara hızla uyum sağlayabilmek için akıllı algoritmaların geliştirilmesi, İHA'ların görevlerini başarıyla tamamlamasını ve kayıpları en aza indirmesini sağlayacaktır. Bunun yanı sıra, birden fazla İHA'nın aynı anda farklı açılardan hedefe saldırarak düşman savunma sistemlerini etkisiz hale getirmesi, hedefin savunma kapasitesini aşmayı ve saldırının başarı şansını artırmayı hedeflemektedir.

Bu hedefler doğrultusunda, çalışmada İHA'ların otonom karar verme ve görev yönetimi yetenekleri geliştirilecektir. Ayrıca, simülasyon ortamlarında test edilen bu stratejilerin, gerçek dünya senaryolarında da etkili olması amaçlanmaktadır. Bu sayede, çoklu İHA sistemlerinin savunma operasyonlarında daha etkin ve güvenilir bir şekilde kullanılması hedeflenmektedir.

1.1.2 Füze Kontrol ve Navigasyon Sistemleri Stratejileri

Füze güdüm sistemleri, ilk ortaya çıktıları dönemden bu yana önemli ölçüde gelişerek basit navigasyondan, sensörler, aktuatörler ve kontrol algoritmalarını entegre eden karmaşık yapılar haline gelmiştir. Bu sistemler, füzenin dinamik uçuş ortamlarında hedefe doğru hassas bir şekilde yönlendirilmesini sağlamaktadır. Bu işlevin merkezinde, roll, pitch ve yaw eksenleri boyunca yönelimi geri besleme kontrollü olarak düzenleyen otopilot sistemi yer almaktadır. Modern füzeler yüksek hızlarda ve değişken koşullar altında çalıştığından, aerodinamik

bozulmalara karşı koyabilecek ve uçuş kararlılığını koruyabilecek sağlam bir güdüm sistemine ihtiyaç duymaktadır. Ancak, GPS ve IMU verilerine dayanan geleneksel navigasyon yöntemleri, özellikle GPS sinyalinin olmadığı ortamlarda güvenilirliğini kaybedebilmektedir.

Yapay zeka (YZ), havacılık ve uzay alanında karmaşık kontrol problemlerine uyarlanabilir çözümler sunarak önemli bir dönüşüm yaratmaktadır. Makine öğrenimi yöntemleri, örneğin yapay sinir ağları, sistemlerin doğrusal olmayan dinamiklerini modelleyebilirken, pekiştirmeli öğrenme (RL) ise kontrol stratejilerini gerçek zamanlı olarak optimize edebilmektedir. Yapay zeka, insansız hava araçlarının yörunge planlamasından, arızalara dayanıklı uydu kontrolüne kadar geniş bir uygulama alanına sahiptir. Geleneksel yöntemlerin aksine, YZ aerodinamik değişkenlikler veya sensör gürültüsü gibi belirsizliklere uyum sağlayarak sistemin dayanıklılığını artırır. Bu çalışma, füze kontrol ve navigasyondaki zorlukları çözmek amacıyla, klasik kontrol yöntemlerini tamamlayacak şekilde denetimli öğrenme ve pekiştirmeli öğrenme yaklaşımının entegrasyonunu araştırmaktadır.

Bu çalışmanın amacı, füze kontrol ve navigasyon sisteme yönelik çeşitli kontrol stratejilerini geliştirmek ve değerlendirmektir. Bu kapsamda, klasik Proportional-Integral-Derivative (PID) kontrolörlerin yanı sıra Doğrusal Regresyon, Long Short-Term Memory (LSTM), Deep Deterministic Policy Gradient (DDPG) pekiştirmeli öğrenme yöntemi ve birleştirilmiş (ensemble) modeller gibi YZ tabanlı yöntemler incelenmiştir. Çalışmanın üç temel hedefi bulunmaktadır: (1) füzenin anti-roll kontrolüne odaklanarak güdüm sisteminin geliştirilmesi, (2) terminal pitch ve yaw açılarını kontrol eden hareket planlama algoritmalarının uygulanması ve optimize edilmesi ve (3) GPS sinyalinin kullanılamadığı durumlarda hedef tespiti için bilgisayarla görme teknolojilerinin kullanılması. Çalışma yalnızca simülasyon tabanlı analizlerle sınırlı tutulmuş; donanım doğrulaması ve çok eksenli kontrol kapsam dışı bırakılarak algoritma geliştirme ve karşılaştırmalı performans analizlere odaklanılmıştır.

1.2 Literatür Özeti

Hava savunma sistemlerinin (HSS), bir ülkenin kara sahاسını hava saldırılara karşı koruma amacıyla geliştirilmesi ve işletilmesi, çeşitli ülkelerin askeri üslerinde hızla ilerlemiştir. Bu tür anti-füze teknolojilerinin askeri envanterde kullanımı, düşman İHA'larının ve savaş uçaklarının ülkenin sınırlarına girmesini engelleyerek ve roketler ile füzelerin ülke içine nüfuz etmelerini zorlaştırarak kara savunmasını önemli ölçüde güçlendirmektedir. Tek bir füze ile yapılan saldırılar, hava savunma sistemi tarafından kolaylıkla karşı saldırıya maruz kalabilmekte ve böylece düşmanın askeri üssüne ya da kara sahاسına etkili bir saldırı gerçekleştirilmesini zorlaştırmaktadır. Ayrıca, düşmanın hava savunma sistemini yok etmeye yönelik görevler, jet

uçakları ile yapıldığında pilotlar için son derece tehlikelidir. Bu nedenle, pilot kayıplarını en az indirmek ve düşman savunmalarını hedefleme operasyonlarında etkinliği artırmak için; insanlı jet uçaklarının yerine sürü İHA'larının kullanılması gibi çeşitli stratejiler geliştirilmiştir [1]. Buna ek olarak, düşmanın hava savunma sistemini aldatmak ve saldırmak amacıyla birden fazla akıllı bomba ve füze de kullanılmaktadır [2].

Çoklu İHA görev planlaması ve görev tahsisü üzerine yapılan çalışmalarda, çoğunlukla genel problem formülasyonlarına odaklanılmaktadır. Bu formülasyonlar genellikle, İHA yeteneklerine bağlı sınırlamalar ve uçuşa yasak bölge gibi ek kısıtlamalar ile belirli bir zaman diliminde 2D koordinat sisteminde belirli noktalara ulaşmakla sınırlıdır [3] [4]. Çözümler çoğunlukla, çoklu gezgin satıcı problemi varyasyonlarını ele almakla sınırlıdır, bu da savaş görevi senaryolarını ve çoklu İHA koordinasyonunu ele almak için fazla basit kalmaktadır [5]. Bu formülasyonlar genel senaryolar için bilgiler ve çözümler sağlasa da dinamik tehdit ortamı ve düşmanın hava savunma sistemi ile etkileşim gibi gerçek dünya problemlerine çözüm getirememektedir. Ek olarak, birçok çalışma teorik modeller ve sayısal simülasyonlarla sınırlı olup belirli gerçek dünya değişkenlerini göz ardı etmektedir.

Statik ortamda çoklu füze iş birliği kontrol problemi, bilinen kısıtlamalar dâhilinde önceden belirlenmiş hedef saldırısına yönelik optimal bir rota planlamaya odaklanırken; dinamik olarak değişen uzaysal ortamda, yani koşullar ve engellerin hızla değiŞebileceği hareketli hedef durumunda, füze iş birliği değişen uzaysal koşullara uyum sağlamak için daha fazla zekâ gerektirir [6]. Bu nedenle, dinamik ortamda bu problem daha karmaşık olarak kabul edilir ve sistemin iyileştirilmesi için daha akıllı bir kontrol gerektirir. Bu karmaşıklıkların üstesinden gelmek için umut vaat eden iki yaklaşım ise pek çok çalışmada araştırılmış olan pekiştirmeli öğrenme ve çoklu ajan karar verme yöntemleridir [7] [8] [9] [10] [11], ancak çoğunlukla MATLAB gibi yazılımlar kullanılarak gerçekleştirilen sayısal simülasyonlarla sınırlı kalmıştır.

GPS sinyallerinin zayıf olduğu veya bulunmadığı GPS olmayan ortamlarda çalışan akıllı füzeler ve mühimmat için, doğru navigasyon ve hedefleme sağlamak amacıyla alternatif yöntemler kullanılmaktadır. Bunlar arasında ayrıntılı vektör haritalar üretmek için yer fotogrametrisi, uydu, uçak ve İHA ile havadan uzaktan algılama ve GPS özellikli LiDAR yer almaktadır [12]. Ayrıca, araştırmacılar, konum, hız ve zaman bilgisi toplamak için GPS kullanmadan mühimmatın konumunu haritalamak ve belirlemek amacıyla atalet ölçüm birimleri (IMU), stereo kameralar, Wi-Fi, radyo frekansı tanımlama (RFID) ve sonar gibi diğer sensörleri incelemiştir [13] [14] [15]. Bu alternatifler üzerine yapılan birçok çalışma bulunmasına rağmen bu alandaki araştırmalar hâlen sınırlıdır.

Bu proje, düşman hava savunma sistemlerini hedef alan görevler için uçtan uca bir çözüm mimarisi sunmayı amaçlamaktadır. Bu kapsam, otonom çoklu İHA'lar için rota planlama ve görev tahsisi ile akıllı mühimmatlar için koordineli füze güdüm ve navigasyon sistemlerini bir arada ele alan bütüncül bir yaklaşım dayanmaktadır. Literatürdeki pek çok çalışma teorik modellemeler veya sınırlı görev tanımları üzerinde yoğunlaşıırken, bu proje gerçekçi bir savaş senaryosu çerçevesinde somut bir problemi çözmeye odaklanmaktadır. Ayrıca, dinamik tehdit ortamlarında İHA koordinasyonu için otonom ve uyarlanabilir algoritmaların entegrasyonu, önceki çalışmalarda yeterince işlenmemiş kritik bir boşluğu doldurmaktadır. Bununla birlikte, çalışma kapsamı çevresel faktörler (hava koşulları, iletişim kesintileri) gibi bazı değişkenleri dışarda bırakmaktadır; bu durum, gelecekte yapılacak çalışmalara yönelik önemli bir araştırma alanı sunmaktadır.

2. MATERİYAL VE YÖNTEM

Bu bölümde, çalışmanın yürütülmesinde kullanılan donanım, yazılım ve simülasyon ortamları “Materyal” başlığı altında; geliştirilen algoritmalar, kontrol mekanizmaları ve deneysel yaklaşımlar ise “Yöntem” başlığı altında ayrı ayrı sunulmaktadır. Bu yapı sayesinde, hem kullanılan araçların teknik özellikleri hem de problem çözümüne yönelik stratejik yaklaşımlar sistematik bir biçimde ele alınmaktadır.

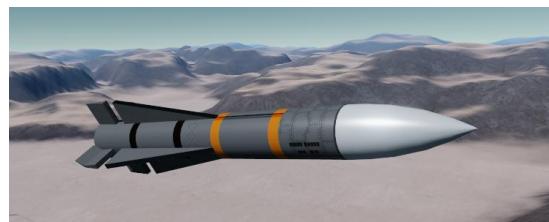
2.1 Materyal

Bu bölümde, proje kapsamında kullanılan araçlar, teknolojiler ve yazılımlar ayrıntılı olarak açıklanmaktadır. İlk olarak, Simplerockets 2 (SR2) simülasyon uygulaması içerisinde yürütülen simülasyon süreci ve bu süreçte kullanılan modellemeler detaylandırılmıştır. Simülasyonda kullanılan araç ve sistemlerin modellenmesi ile başlayan süreç, SR2 uygulamasına entegre edilmiş olan Vizzy programlama dilinin açıklanmasıyla devam etmektedir.

Bununla birlikte, proje süresince kullanılan Python programlama dili ve ilgili kütüphaneler ayrıntılı olarak sunulmuştur. Projede; yapay zeka ve DDPG algoritması için TensorFlow, doğrusal regresyon uygulamaları için Scikit-learn, görselleştirme işlemleri için Matplotlib, veri işleme için Pandas ve NumPy, kontrol ve otomasyon süreçleri için ise PyAutoGUI ve PyKey gibi çeşitli kütüphane ve araçlar kullanılmıştır. Ayrıca, SR2 simülatörü ile Python arasındaki iletişimini sağlayan ve WNPJ kullanıcı tarafından geliştirilmiş olan SR2Logger Modunun kullanımına da yer verilmiştir. Son olarak ise İHA'ların hareketlerini 2D grafikler üzerinde simüle eden MATLAB kullanımına degenilmiştir.



Şekil 2.1. AIM-54C by Hughes Aircraft



Şekil 2.2. AIM-54C Phoenix by NoLuja
(NoLuja, Juno - AIM-54C Phoenix (Fully working air to air missile), 2022)

2.1.1 Simülasyon ortamı – Simplerockets 2 (Juno: New Origins)

SR2, roketler ve havacılık araçları için özel olarak geliştirilmiş bir uçuş simülatöründür [16]. Simülatörde uygulanan fizik yasaları, gerçek dünya fizik kurallarına oldukça yakın olup, kullanıcıya gerçekçi bir deneyim sunmaktadır. Unity oyun motoru kullanılarak geliştirilen SR2, uygulama programlarının işlevselliğini genişletmek veya değiştirmek için özel modifikasyonların (modların) kolayca entegre edilmesine olanak tanımaktadır. Topluluk tarafından geliştirilmiş ve herkesin ücretsiz olarak kullanabileceği birçok mod mevcuttur.

SR2 ayrıca kullanıcıların kendi araç modellerini tasarlayıp inşa etmelerine de olanak sağlamaktadır. Bu proje kapsamında, akıllı mühimmat geliştirme deneyleri için kullanılan hava aracı modeli, Hughes Aircraft Company tarafından geliştirilen AIM-54C Phoenix Füzesi esas alınarak seçilmiştir [17]. Simülasyonda kullanılan füze modeli ise NoLuja tarafından geliştirilen AIM-54C Phoenix modelidir [18]. Şekil 2.1 ve Şekil 2.2, gerçek hayatı AIM-54C Phoenix füzesi ile SR2 simülatörüne entegre edilmiş füze modelinin görselleri sunmaktadır.

2.1.2 Vizzy programlama (IDE)

SR2 simülatörü içerisinde, Vizzy adı verilen ve görsel bloklar tabanlı bir programlama dili (sürükle-bırak yapısında) yer almaktadır. Scratch programlama diline benzer şekilde tasarılanan Vizzy, kullanıcıların roketler, hava araçları veya diğer taşıtlar için uçuş otomasyon betikleri geliştirmelerine olanak tanımaktadır.

Bu proje kapsamında, Vizzy dili; model hava aracının sensörlerinden veri toplamak ve bu verileri UDP protokolü üzerinden gerçek zamanlı olarak SR2 uygulaması dışına aktarmak amacıyla kullanılmıştır. Söz konusu veri aktarım süreci, WNP78 tarafından geliştirilen SR2Logger adlı bir mod aracılığıyla desteklenmiştir. SR2Logger moduna ilişkin ayrıntılı bilgilere bir sonraki bölümde yer verilecektir. Simülatörden dışarıya aktarılan veriler, daha sonra Python programı tarafından alınarak işlenmiştir.

2.1.3 Python

Bu projede ana programlama dili olarak Python tercih edilmiştir. Python'un sağlam yapısı ve görece basit sözdizimi, geliştirme sürecini oldukça kolaylaştırmıştır. Ayrıca Python'un nesne yönelimli programlama (OOP) yaklaşımını desteklemesi, modüler ve ölçeklenebilir bir yapı kurmayı daha elverişli hale getirmiştir. Threading (çoklu iş parçacığı) ve socket (soket programlama) gibi birçok yerleşik kütüphanenin sağladığı imkanlar, programın geliştirme

sürecine önemli ölçüde katkı sağlamıştır. Bu projede socket programlama ve çoklu iş parçacığı kullanımına bir sonraki bölümde yer verilecektir.

İHA formasyon ve görev ataması üzerine geliştirilen tüm 2 boyutlu simülasyon süreci de yine Python programlama dili kullanılarak gerçekleştirilmiştir.

Bunun yanı sıra, proje kapsamında geliştirme sürecini desteklemek amacıyla bazı üçüncü parti kütüphaneler de kullanılmıştır. Kullanılan başlıca kütüphaneler Tablo 2.11'de listelenmiştir.

Tablo 2.1. Kullanılan Python kütüphaneleri

No	Kütüphane	Açıklama
1	PyAutoGui	Simülasyon ortamı ayarlarını kontrol sağlamak
2	PyKey	Klavye üzerinde girdi kontrolü simule etmek
3	TensorFlow	Yapay zeka modelleri tasarlamak, eğitmek, ve kullanmak
4	Pandas	Verisetleri daha kolay işlemek
5	Numpy	Veriseti hazırlamak
6	Matplotlib	Verileri görselleştirmek
7	Scikit	Basit makine öğrenmesi modeli eğitmek

Projenin geliştirilmesi sırasında Anaconda, Spyder IDE ve PyCharm IDE kullanılmıştır. Anaconda, program geliştirmeyi kolaylaştırip sürümleri, paketleri ve Python ortamlarını yönetmeye yardımcı olmuştur. Spyder IDE, veri analizini kolaylaştırınan ücretsiz bir Python IDE'dir. PyCharm IDE ise, özellikle karmaşık kod tabanlarının yazımı, hata ayıklanması ve genel proje yönetiminde sağladığı gelişmiş özellikler nedeniyle simülasyon algoritmalarının geliştirilmesi sürecinde aktif olarak kullanılmıştır.

2.1.4 SR2Logger mod

SR2 içerisinde üretilen verilerin, Python tarafından işlenip görselleştirilebilmesi için uygulama dışına aktarılması gerekmektedir. Bu veri aktarımını kolaylaştırmak amacıyla, Vizzy aracılığıyla toplanan verilerin UDP protokolü üzerinden dış uygulamalara iletilmesini sağlayan bir mod olan SR2Logger kullanılmıştır. Söz konusu mod, WNP78 tarafından geliştirilmiş olup, aşağıdaki bağlantı üzerinden GitHub platformunda açık erişime sunulmuştur [19].

2.1.5 MATLAB

Projenin başlangıç aşamasında, İHA'ların hareketlerini 2D grafikler üzerinde simüle etmek amacıyla MATLAB ortamı kullanılmıştır. MATLAB, İHA'ların belirli bir başlangıç noktasından hedefe doğru hareket etmesini ve hedefi vurmasını simüle etmektedir. Bu süreçte, geçiş noktalarını (waypoints) kullanarak İHA'ların hedefe ulaşmaları sağlanmıştır. Ayrıca,

interpolasyon teknikleri kullanılarak İHA'ların hareketleri daha pürüzsüz bir şekilde görselleştirilmiştir.

MATLAB'de elde edilen bu temel simülasyonlar, İHA'ların hareketlerini anlamak ve algoritmaların doğru şekilde çalıştığını test etmek amacıyla gerçekleştirılmıştır. Daha sonra, daha kapsamlı ve esnek bir simülasyon ortamı oluşturmak amacıyla Python ortamına geçilmiştir.

2.2 Yöntem

Bu bölümde, çalışmada ele alınan problemin çözümüne yönelik geliştirilen yöntem ve yaklaşımlar sunulmaktadır. İlk olarak, elevator, rudder ve aileron kontrol algoritmaları açıklanmış; bu kontrollerin çoklu iş parçacığı (multithreading) kullanılarak nasıl uygulandığı detaylandırılmıştır.

Bunun yanı sıra, kontrol ve seyrüsefer (navigation) problemlerinin çözümüne yönelik olarak çeşitli deneyler gerçekleştirilmiş ve bu deneylerde kullanılan yöntemler açıklanmıştır. Terminal yaw açısının kontrolü için waypoint ve Bézier eğrileri temelli bir yöntem geliştirilmiş; terminal pitch açısının kontrolü için ise proportional navigation yöntemi kullanılmıştır.

Ayrıca, füze modelinde gözlemlenen roll kararsızlığı problemi, PID, Doğrusal Regresyon, LSTM ve DDPG modellerinden oluşan bir ansambl (ensemble) model tabanlı anti-roll sistemi ile çözülmüştür. Söz konusu anti-roll modellerinin eğitim süreci de bu kapsamda ayrıntılı olarak açıklanmıştır.

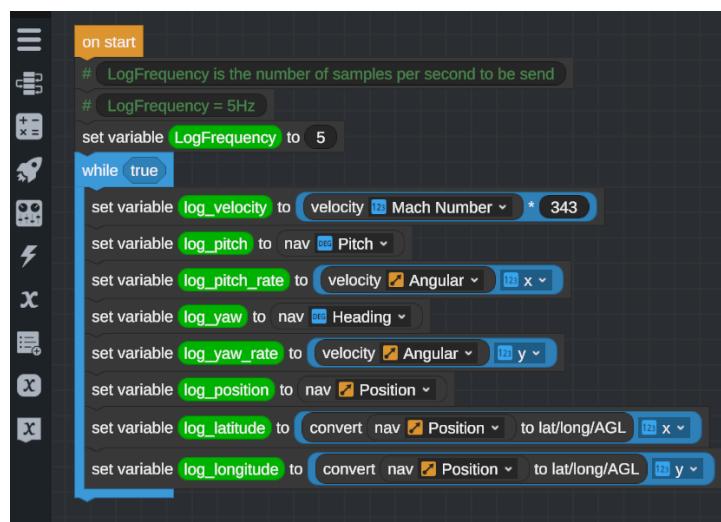
Son olarak, YOLO nesne tespit algoritmasının kullanımı ele alınmış ve bu algoritmanın füzenin hedefe yönlendirilmesinde nasıl bir rol oynadığı ortaya konulmuştur.

2.2.1 Veri İletişimi

SR2 içerisindeki veriler, SR2Logger modunu kullanarak UDP protokolü aracılığıyla dış uygulamalara iletilmektedir. İlk olarak, SR2Logger modunun SR2 uygulamasına doğru şekilde entegre edilmesi gerekmektedir. Vizzy üzerinden dış bir uygulamaya veri aktarımı sağlanması için, gönderilecek verilerin tanımlanacağı değişken adlarının ‘log_’ ön eki ile oluşturulması gerekmektedir. Örneğin, ‘log_pitch_angle’ veya ‘log_vertical_velocity’ gibi değişkenler tanımlanarak SR2Logger modunun bu verileri otomatik olarak iletmesi sağlanmaktadır. İletim sıklığı (frekansı), Vizzy içerisinde tanımlanan LogFrequency isimli bir değişken aracılığıyla kullanıcı tarafından belirlenebilir. Ayrıca modun kullanımı sırasında

hostname ve port numarası gibi bağlantı parametreleri de tanımlanmalıdır. Şekil 2.3, Vizzy ortamında SR2Logger tarafından UDP üzerinden dışarıya gönderilmeye hazır bir değişken tanımını örnek olarak göstermektedir.

Bunun yanı sıra, UDP protokolü yalnızca SR2'den veri almak değil, dış uygulamalardan SR2'ye veri göndermek için de kullanılmaktadır. Python programı, UDP protokolü üzerinden belirlenen bir IP adresi ve port numarasına veri göndermektedir. Bu süreçte, Python'un socket kütüphanesi kullanılarak bir UDP istemcisi oluşturulmuş ve belirli aralıklarla veri iletimi sağlanmıştır. Gönderilen veri, SR2Logger tarafından sağlanan bilgilerden ya da Python'da oluşturulan simülasyon verilerinden oluşabilir.



Şekil 2.3. SR2Logger kullanarak UDP üzerinden veri göndermek için Vizzy programı

Veri, SR2Logger tarafından UDP yoluyla gönderildikten sonra, Python programı tarafından socket kütüphanesi kullanılarak alınmaktadır. Alınan veri, tercih edilen bir kodlama formatına göre parse edilir ve daha sonra kolay erişim sağlamak amacıyla bir sözlük yapısında saklanır. Şekil 2.4'de, SR2Logger'dan gelen verilerin nasıl alındığını gösteren bir Python kod örneği yer almaktadır.

```

import socket, struct

TypeFormats = [
    "",      # null
    "d",     # double (float64)
    "?",     # bool
    "ddd"]   # Vector3d

class Packet:
    def __init__(self, data):
        self.data = data

```

```

        self.pos = 0
    def get(self, l): # get l bytes
        self.pos += l
        return self.data[self.pos - l:self.pos]
    def read(self, fmt): # read all formatted values
        v = self.readmult(fmt)
        if len(v) == 0: return None
        if len(v) == 1: return v[0]
        return v
    def readmult(self, fmt): # read multiple formatted values
        return struct.unpack(fmt, self.get(struct.calcsize(fmt)))
    @property
    def more(self): # is there more data?
        return self.pos < len(self.data)

def readPacket(dat):
    p = Packet(dat)
    values = {}
    while p.more:
        nameLen = p.read("H")
        name = p.get(nameLen).decode()

        tp = p.read("B")
        typeCode = TypeFormats[tp]
        val = p.read(typeCode)
        values[name] = val

    return values

```

Şekil 2.4. UDP paketlerinden veri çıkarmak için readPacket fonksiyonu

2.2.2 İki Boyutlu Düzleme Çoklu İHA Koordinasyonu ve Otonom Görev Ataması

Çoklu İHA sistemlerinin koordinasyonu ve formasyon kontrolü, bu projenin temel araştırma konularından birini oluşturmaktadır. İHA'ların belirli bir hedefe ulaşmak için optimal rotalar belirlemesi, senkronize hareket etmesi ve dinamik ortamlara uyum sağlama amacıyla çeşitli algoritmaların araştırılması ve uygulanabilirliklerinin değerlendirilmesi planlanmıştır. Bu kapsamında, İHA'ların hareketlerinin simülasyonu, formasyon kontrolü ve görev tahsisi gibi konular üzerinde çalışmalar sürdürülmüştür.

Projenin ilk aşamasında, İHA'ların hedeflenen hareketlerini 2D grafikler üzerinde simüle etmek amacıyla MATLAB ortamında temel bir kod geliştirilmiştir. Bu simülasyon, İHA'ların başlangıç noktasından hedefe doğru hareketini ve bu süreçte geçiş noktalarını (waypoints) kullanarak hedefe ulaşmalarını modellemektedir. Simülasyon ortamı, İHA'ların rotalarını pürüzsüz bir şekilde gösterebilmek için interpolasyon tekniklerinden faydalananmakta olup, hedefe ulaşma sürecinin görselleştirilmesini sağlamaktadır.

İHA koordinasyonu ve görev ataması üzerine yürütülen çalışmalar, daha kapsamlı, daha güçlü bir 2D simülasyona ihtiyaç duyulması ve geliştirilen algoritmaların eş zamanlı olarak uygulamada görülebilmesi adına ilerletilmiştir. Sistemin temel uçuş dinamiklerinin kavranmasından başlayarak, çoklu İHA sistemleri arasındaki karmaşık etkileşimleri ve görev senaryolarını kapsayacak şekilde aşamalı bir yaklaşımla devam edilmiştir.

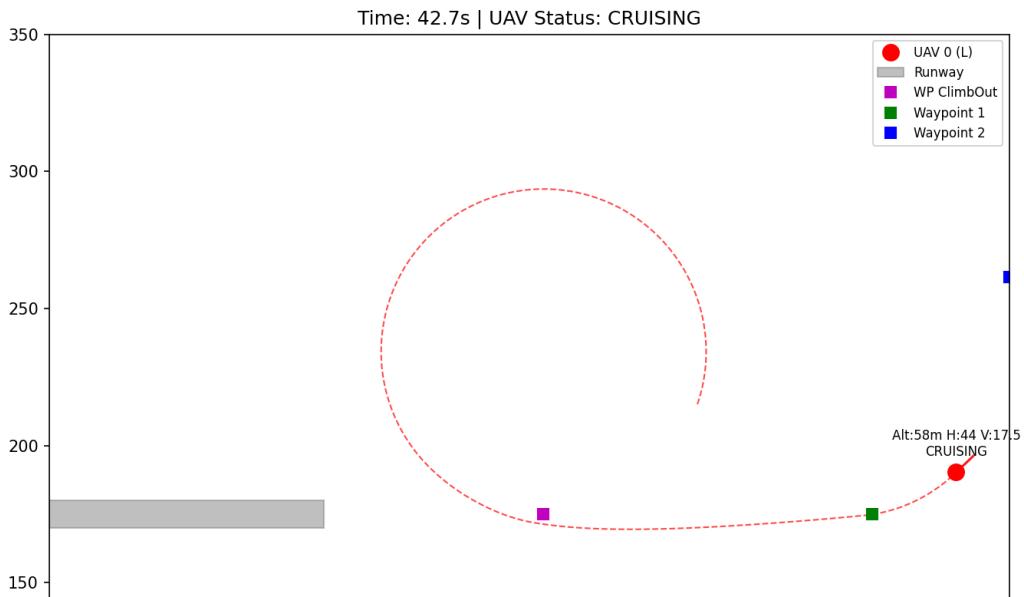
a.) Tek İHA uçuş dinamikleri ve temel seyrüsefer testleri

İlk aşamada, tek bir İHA'nın temel uçuş karakteristiğini ve otopilot davranışlarını anlamak amacıyla “Tek İHA Uçuş Testi” adlı bir simülasyon programı geliştirilmiştir. Bu program, bir İHA'nın pistten kalkış yapması, belirli bir irtifaya tırmanması ve önceden tanımlanmış ara noktaları takip etmesi gibi temel seyrüsefer görevlerini başarılı bir şekilde yerine getirme kabiliyetini doğrulamak üzere tasarlanmıştır.

Bu simülasyonda, bir İHA'nın konumu $p = [x, y, z]^T$, hızı $v = [\dot{x}, \dot{y}, \dot{z}]^T$ ve yönelimi ψ gibi temel durum değişkenleri soyut bir İHA sınıfı içinde yönetilmektedir. İHA'nın davranışsal modları, bir sonlu durum makinesi olarak kurgulanmış “UAVStatus” adlı bir “Enum” yapısı ile kontrol edilmektedir. Bu durumlar, İHA'nın yerdeki bekleme, pistte hızlanma, kalkış sonrası tırmanma ve seyir gibi operasyonel aşamalarını temsil etmektedir.

İHA'nın hareket denklemleri, ilgili metod içinde uygulanan temel kontrol yasalarıyla sağlanmaktadır. Yönelim kontrolü için, istenen yönelim açısı ψ_d ile mevcut yönelim açısı ψ arasındaki hata $e_\psi = \psi_d - \psi$ kullanılarak bir orantısal (P) kontrolör uygulanmakta; dönüş oranı $\dot{\psi}$ için kontrol yasası da $\dot{\psi} = K_p^\psi e_\psi$ şeklinde uygulanmıştır. Burada K_p^ψ bir orantısal kazançtır. Bu kontrol prensibi, havacılık kontrol sistemleri literatüründe temel bir yer tutmaktadır [20]. Hız kontrolü için ise, istenen yatay hız V_d^{xy} ile mevcut yatay hız V^{xy} arasındaki farka dayalı, basit bir ivme/yavaşlama modeli kullanılmaktadır. İrtifa kontrolünde de benzer şekilde, istenen irtifa ile mevcut irtifa arasındaki hata kullanılarak dikey hız belirlenmektedir.

Kalkış fazında, İHA'nın hızı belirli bir eşik değere ulaştığında otomatik olarak tırmanma durumuna geçisi, kalkış performans karakteristiklerini yansıtan basitleştirilmiş bir modeldir. Devamında gerçekleşen nokta takip durumu Şekil 2.5'te görülebilmektedir. Bu program, tek bir İHA'nın seyrüsefer ve otopilot fonksiyonlarının izole bir ortamda test edilmesine olanak tanıyarak, çoklu İHA sistemlerinin geliştirilmesi için güvenilir bir zemin hazırlamıştır.



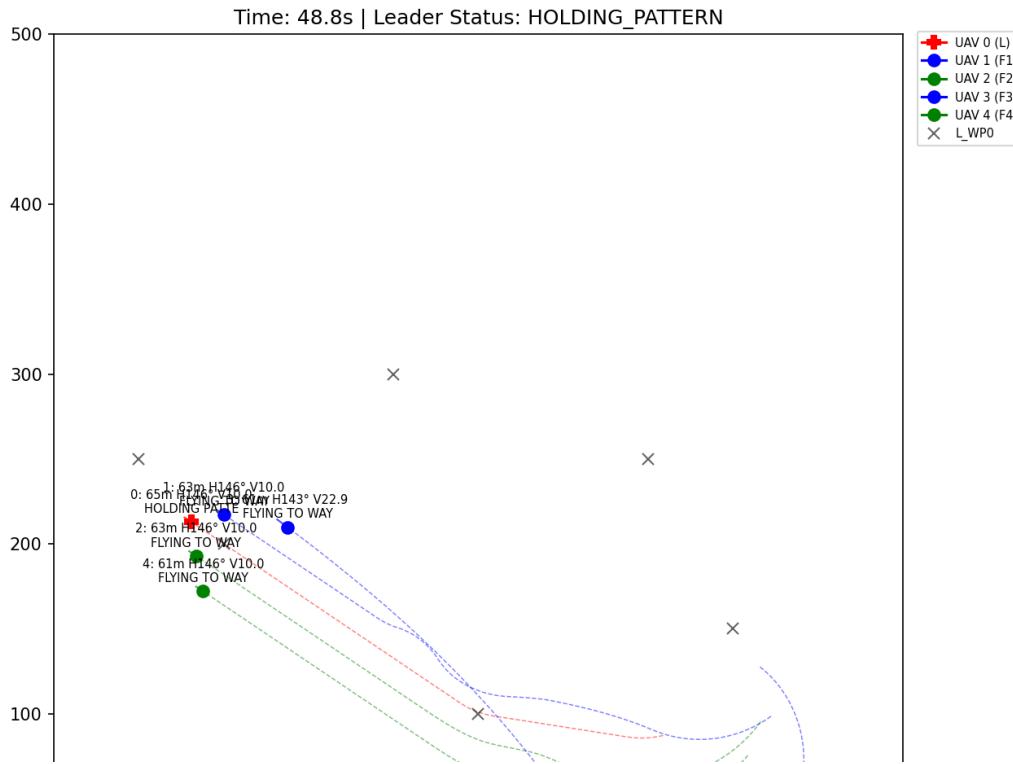
Şekil 2.5. Tek İHA ile kalkış ve nokta takip simülasyonu

b.) Çoklu İHA sistemlerinde konsensüs algoritmaları üzerine ilk çalışmalar

Tek İHA'nın temel uçuş yetenekleri doğrulandıktan sonra, birden fazla İHA'nın bir grup olarak uyumlu bir şekilde hareket etmesini sağlayacak koordinasyon mekanizmaları üzerine yoğunlaşılmıştır. Bu doğrultuda, bir ön prototip program geliştirilmiştir. Bu program, konsensüs algoritmalarının temelini oluşturan lider-takipçi (leader-follower) mimarisini ve formasyon koruma prensiplerini anlamak üzere tasarlanmış olup, projenin nihai çıktısı olmamakla birlikte, kavramsal doğrulama ve temel algoritma prensiplerinin kavranması açısından önemli bir aşama olarak değerlendirilmektedir.

Bu prototip programda uygulanan formasyon kontrol stratejisi, Lider Referanslı Sanal Yapı (Leader-Referenced Virtual Structure) yaklaşımının [21] temel bir formunu teşkil etmektedir. Bu yaklaşım kapsamında, her takipçi İHA için liderin gövde koordinat sistemine göre sabit bir göreceli konum $o = [o_x, o_y, o_z]^T$ tanımlanmaktadır; bu ofsetler “FORMATION_OFFSETS” dizisi ile belirlenmektedir.

Hız kontrolünde, liderin hedef hızı $V_{L,d}^{xy}$ ile takipçinin liderle olan göreceli mesafesindeki hata (*along_error*) kullanılarak bir orantısal hız düzeltmesi uygulanmakta, böylece Şekil 2.6'da görüldüğü gibi takipçinin liderle olan uzunlamasına mesafesi optimize edilmektedir. Bu program, çoklu İHA formasyon kontrolünde lider-takipçi mimarisinin fizibilitesini kanıtlamış ve daha karmaşık görev senaryoları için bir başlangıç noktası sunmuştur.



Şekil 2.6. Çoklu İHA ile konsensüs algoritması denemesi

c.) Kapsamlı İHA sürü yönetimi: Dinamik görev planlama, koordinasyon ve adaptif tehdit tepkisi

Projenin temel çıktısı ve en kapsamlı aşamasını, önceki çalışmaların üzerine inşa edilen ve çoklu İHA sistemlerinin karmaşık bir operasyonel senaryo altında koordinasyonunu, dinamik görev atamasını ve çevresel tehditlere karşı adaptasyonunu hedefleyen ana simülasyon programı oluşturmaktadır. Bu program, kullanıcıya çeşitli saldırı stratejileri, uçuşa yasak bölgelerin (NFZ) dahil edilmesi ve İHA arıza senaryoları gibi özelleştirilebilir parametreler sunarak, gerçek dünya operasyonlarındaki dinamizm ve belirsizliği daha detaylı modellemektedir.

Bu çoklu İHA simülasyonu, bir lider-takipçi (leader-follower) formasyon kontrol stratejisini temel almaktadır. İHA'ların davranışları, görev aşamalarına, formasyon gereksinimlerine ve dinamik çevre koşullarına (örneğin, uçuşa yasak bölgeler - NFZ'ler ve Kara Tabanlı Hava Savunma (GBAD) sistemi) göre adaptif olarak şekillenmektedir.

- **Formasyon kontrolü: Lider referanslı sanal yapı yaklaşımı:**

Simülasyonda İHA'lar arasındaki formasyon uyumu, özellikle takipçi İHA'ların lider İHA'yı izlemesi, Lider Referanslı Sanal Yapı (Leader-Referenced Virtual Structure) yaklaşımına dayanmaktadır. Bu yaklaşım, literatürde geniş kabul görmüş olup [21] çeşitli uygulamalarda tercih edilmektedir. Formasyon, lider İHA'nın anlık konumu $p = [x, y, z]^T$ ve yönelimi ψ_L ekseninde tanımlanan sabit bir ofset vektörü $o = [o_x, o_y, o_z]^T$ ile bir sanal yapı olarak kabul edilmektedir. Her takipçi İHA, bu sanal yapı içinde kendisine atanmış olan göreceli konumu koruma hedefine sahiptir. Kodda yer alan “FORMATION_OFFSETS” parametresi bu yapının geometrisini belirlemektedir. Takipçi İHA'nın istenen yatay konumu şu şekilde hesaplanmaktadır: $p_{F,xy}^d = p_{L,xy} + o_x vec_L + o_y vec_L^\perp$

Burada, $vec_L = [\cos(\psi_L), \sin(\psi_L)]^T$ liderin ileri yön vektörü ve $vec_L^\perp = [-\sin(\psi_L), \cos(\psi_L)]^T$ liderin sol yön vektöridür. Bu sayede, takipçi liderin gövde koordinat sistemine (body frame) göre sabit bir noktayı izleyebilmektedir.

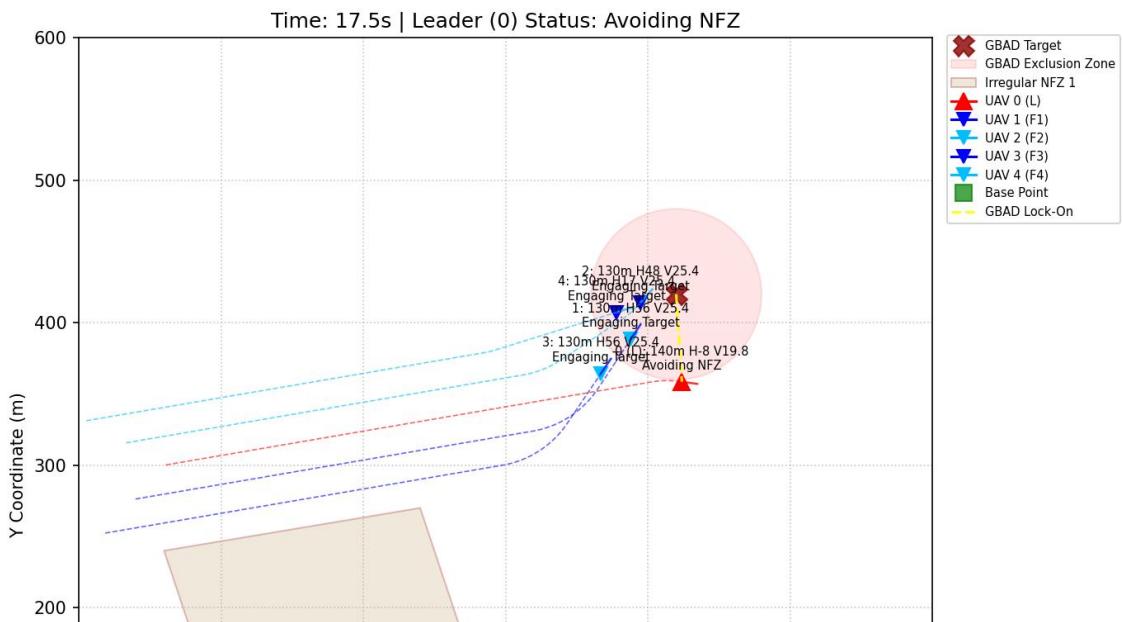
İstenen irtifa $z_F^d = z_L + o_z$ şeklinde belirlenmekte ve formasyonun üç boyutlu iç yapısının korunmasına hizmet etmektedir. Takipçinin istenen hızı ise, liderin hedef hızı ile eşitlenmekte, ancak takipçinin liderle olan uzunlamasına mesafesindeki hata, $e_x = o_x - (p_{F,xy}^d - p_{L,xy}) \cdot vec_L$ 'ye orantılı bir hız düzeltmesi $K_p^V e_x$ uygulanmaktadır. Bu mekanizma, takipçinin liderle olan mesafesini optimize etmeye yarayan basit bir orantısal hız geri besleme (proportional speed feedback) mekanizmasıdır. Her İHA'nın dahili kontrol döngüleri, belirlenen bu istenen durumlara ulaşmak için orantısal kontrol yasaları uygulamaktadır. Bu temel kontrolörler, bazı havacılık kontrol sistemleri çalışmalarında detaylıca ele alınan otopilot tasarımlarının [20] [22] kısmen basitleştirilmiş formlarıdır.

- **Görev yönelimi ve dinamik durum yönetimi:**

Simülasyon, İHA'ların farklı görev aşamaları arasında geçiş yapmasını sağlayan bir durum makinesi (state machine) modeli kullanmaktadır. Her İHA'nın durumu (UAVStatus Enum'u), o anki davranışını ve görevdeki rolünü belirlemektedir. Bu durum bazlı yaklaşım, otonom sistemlerin görev planlamasında ve yürütülmesinde yaygın olarak kullanılan [23] bir model olup, robotik ve yapay zekaya dair eserlerde açıklanmaktadır. Görev akışı, önceden tanımlanmış olaylara (örn., bir hedefe ulaşma veya çevresel tehditler) dayalı olarak

durum geçişlerini tetiklemektedir. Kodda, lider ve takipçi İHA'lar için ayrı ayrı durum geçiş mantıkları bulunmaktadır.

"Lider direkt saldırı" stratejisi senaryosunda, lider İHA belirlenen bir füze bırakma noktasına yönelmektedir. Hedefe yeterince yaklaşıldığında, lider hedefle angajman durumuna geçmekte ve angajman süresince p_{UAV} konumunda kalmaya çalışırken hedef GBAD sistemine karşı Şekil 2.7'de görüldüğü gibi simüle edilmiş bir saldırısı gerçekleştirmektedir. Angajman süresi tamamlandığında, lider İHA'nın konumu $p_{GBAD,xy}$ olan tehditten uzaklaşmak için bir kaçınma manevrası gerçekleştirilmektedir. Bu kaçınma yönelimi $\psi_{evade} = \text{atan2}(p_{UAV,y} - p_{GBAD,y}, p_{UAV,x} - p_{GBAD,x})$ olarak hesaplanmaktadır, yani tehditten direkt olarak uzaklaşma yönüne doğru bir yönelim atanmaktadır. Bu manevra belirli bir süre devam ettiğten sonra İHA üssü dönüş fazına geçmektedir.

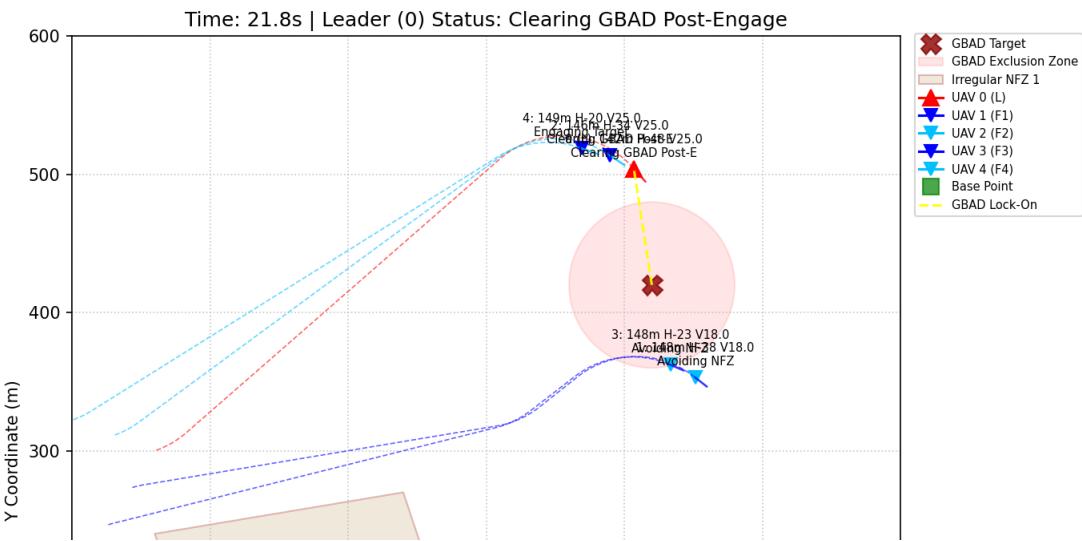


Şekil 2.7. "Lider direkt saldırı" senaryosunda angajman anı

"Kanattan Kuşatma ve Yakınsama" stratejisi senaryosunda, İHA'lar başlangıçta hedef GBAD sistemine doğrudan yönelmek yerine, GBAD sisteminin yan bölgelerinde önceden tanımlanmış iki farklı kanat noktasına yönelirler. İHA'lar, ID'lerine göre bu kanat noktalarına dağıtilır (örn., tek ID'liler bir kanata, çift ID'liler diğerine). Kanat noktalarına ulaşıldığında, İHA'lar hedefe doğru bir yakınsama noktası $p_{converge}$ 'ye yönelik olarak hareket eder. Bu strateji, İHA'ların hedef savunmasını farklı açılardan aşmaya yönelik koordineli bir saldırısı temsil eder. Angajman süresi

tamamlandığında, İHA'lar hedef sonrası temizleme manevrasıyla bölgeden uzaklaşarak üsse dönerler.

Takipçiler genellikle liderin durumunu referans alarak kendi durumlarını güncellemektedir. Ayrıca, GBAD sisteminin karşı saldırısı gibi dış olaylar, küresel değişkenler aracılığıyla İHA'ların angajman davranışlarını tetiklemektedir. Bu durum bazlı akışlar, gibi kaynaklarda açıklanan hibrit kontrol sistemleri (hybrid control systems) prensiplerine [24] uygun olarak, ayırt durum geçişleri ile sürekli kontrolü birleştirmektedir.

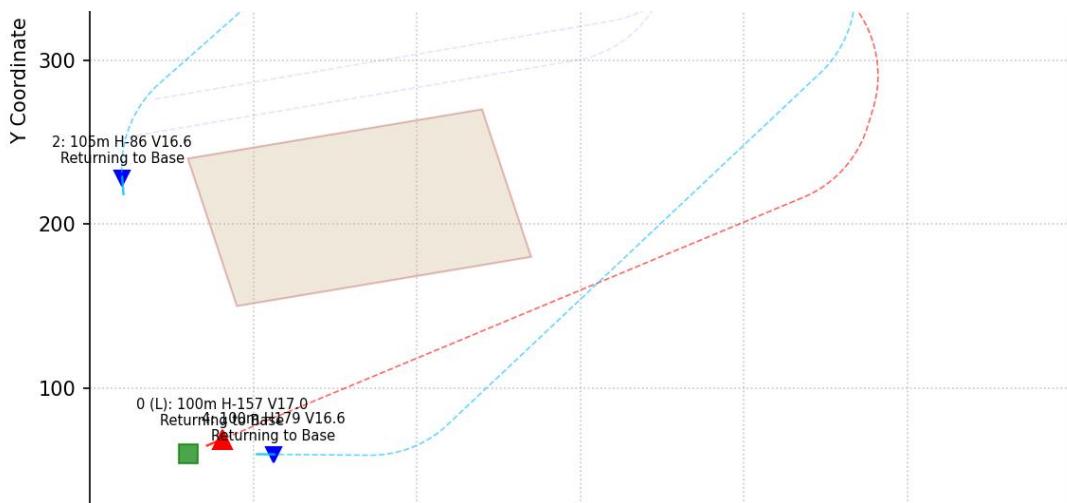


Şekil 2.8. “Kanattan Kuşatma ve Yakınsama” senaryosunda angajman anı

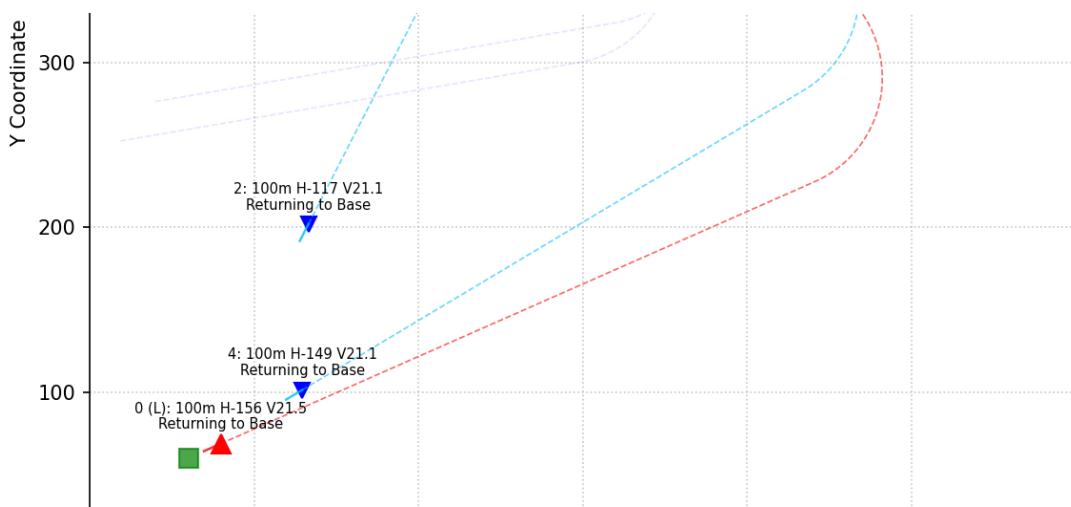
- **Uçuşa yasak bölge (NFZ) ve GBAD kaçınma: Kural tabanlı reaktif ve yol planlama entegrasyonu:**

İHA'ların uçuşa yasak bölgelerden kaçınması, temel olarak kural tabanlı reaktif davranış (rule-based reactive behavior) prensibine dayanmaktadır. Ancak bu simülasyon, üsse dönüş sırasında yol kesişimi tahmini ve dinamik ara nokta oluşturma ile bu kaçınma stratejisini zenginleştirmiştir. Bir tehdit (burada NFZ) algılandığında, İHA öncelikli olarak bu tehditten uzaklaşmayı veya onu atlamayı hedeflemekte ve mevcut görevini geçici olarak askıya almaktadır. Bir İHA'nın mevcut konumu p_{UAV} ile bir sonraki tahmini konumu $p_{UAV,next}$ arasındaki çizgi segmentinin, bir NFZ'nin merkezine c_{NFZ} ve yarıçapına r_{NFZ} sahip dairesel bir NFZ ile kesişimi, $d(c_{NFZ}, \overline{p_{UAV} p_{UAV,next}}) \leq r_{NFZ}$ koşuluyla değerlendirilmektedir. Burada d bir noktadan bir doğru parçasına olan en kısa mesafeyi ifade eder. Benzer şekilde, çokgen NFZ'ler için de çarpışma tespiti (collision detection)

algoritmaları [25] esas alınarak kesişim testleri yapılmaktadır. Eğer bir rota kesişimi tespit edilirse, İHA dinamik olarak NFZ'yi teğet geçecek en kısa "detour" ara noktasını hesaplamaktadır. Şekil 2.9'da bu yeni hesaplanan rota, Şekil 2.10'da ise herhangi bir engel olmadığından İHA'ların davranışları görülebilmektedir. İHA'ların davranışları Bu durum, literatürde ele alınan ara nokta tabanlı yol planlama ve yeniden rotalama (waypoint-based path planning and rerouting) algoritmalarına [26] benzerlik göstermekte ve daha karmaşık bir kaçınma stratejisi sunmaktadır.



Şekil 2.9. Simülasyona NFZ dahil edildiğinde İHA'ların kaçınma davranışları

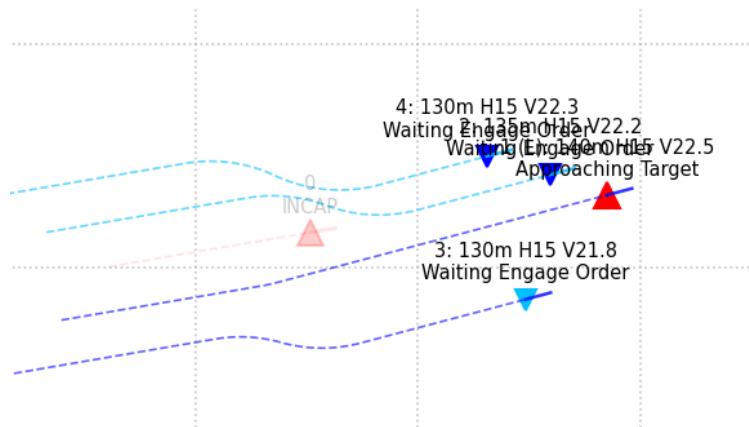


Şekil 2.10. Simülasyona NFZ dahil edilmediğinde İHA'ların davranışları

- **Lider arızası ve yeniden lider seçimi: Merkezi yetki devri:**

Lider İHA'nın işlevsiz hale gelmesi durumunda yeni bir liderin seçilmesi mekanizması, merkezi yetki devri (centralized authority handover) ile gerçekleştirilmektedir. Otonom sistemlerde liderin kaybedilmesi kritik bir durum olup, görevin sürekliliği için yeni bir liderin hızlı ve kararlı bir şekilde atanması gerekmektedir; bu durum, "lider arızasından kurtarma" (leader failure recovery) [21] olarak adlandırılmaktadır. İlgili yeni lider seçim fonksiyonu, Şekil 2.11'de görüldüğü gibi, arızalanan lideri devreden çıkararak, kalan aktif ve görev yapabilir İHA'lar arasından yeni bir lider seçimi yapmaktadır. Bu seçim, önceden belirlenmiş kriterlere dayanmaktadır:

- En Düşük ID'ye Sahip İHA: En basit ve doğrudan kriter olup, mevcut İHA'lar arasında ID numarası en küçük olan aktif ve lider olmayan İHA yeni lider olarak atanmaktadır. Bu yöntem, hiyerarşideki önceliğin belirlenmesi için sıkça kullanılmaktadır.
- Üsse En Yakın İHA: Görevin üsse dönüş aşamasında veya hasar sonrası hızlı toparlanma ihtiyacı olduğunda tercih edilen bir kriterdir. Bu durumda, aktif İHA'lar arasından üs konumu p_{base} 'ye en kısa öklid mesafesine ($\|p_{UAV} - p_{base}\|_2$) sahip olan İHA, yeni lider olarak atanır. Bu kriter, görev verimliliğini ve operasyonel geri dönüş süresini optimize etmeyi hedefler. Bu seçim kriterleri, dağıtık kontrol sistemlerindeki takım üyelerinin koordinasyon ve rol atama yöntemlerine [27] benzerlik göstermektedir. Seçim sonrası, yeni liderin görev durumu (örn. GBAD sistemi bölgesinde ise temizleme manevrası başlatma, üsse dönme) uygun şekilde güncellenmekte ve diğer takipçi İHA'lar da yeni lidere referanslarını ayarlayarak formasyonu yeniden kurmaktadır. Ayrıca, belirli bir NFZ ihlali durumunda liderin otomatik olarak görevden alınması ve üsse dönmeye zorlanması mekanizması, bu alandaki çalışmalara [21] paralel olarak, sistemin belirli kritik ihlallere karşı otonom bir tepki vermesini ve görev sürekliliğini saglamasını amaçlamaktadır.



Şekil 2.11. Simülasyona NFZ dahil edilmediginde İHA'ların davranışları

2.2.3 Python aracılığıyla 3D İHA simülasyonunun gerçek zamanlı kontrolü



Şekil 2.12. Simplerockets 2 simülasyon ortamında TB2 İHA modeli modeli

Bu bölümde, TB2 tipi bir insansız hava aracının (İHA) SR2 (SimpleRockets 2) simülatörü üzerinde otonom olarak yönlendirilmesini sağlamak amacıyla Python programlama dili kullanılarak bir kontrol sistemi geliştirilmiştir. Geliştirilen sistem, simülatörden UDP protokolü ile alınan gerçek zamanlı telemetri verilerinin işlenmesi ve bu verilere dayalı olarak İHA'nın kontrol yüzeylerinin otomatik olarak yönetilmesini içermektedir. Şekil 2.12'de Simplerockets 2 simülasyon ortamında TB2 İHA modeli gösterişmektedir.

Simülasyon ortamında, elevator (pitch), rudder (yaw) ve aileron (roll) kontrol yüzeylerine klavye tuşları aracılığıyla müdahale edilmektedir. Python programı içerisinde pyKey_windows modülü kullanılarak 'w', 's', 'q', 'e' gibi tuşlar simüle edilmekte ve bu sayede kontrol yüzeylerine dışarıdan komut verilebilmektedir. Özellikle bu çalışmada, yaw (rudder) kontrolü devre dışı bırakılmış; yalnızca roll ve pitch eksenleri üzerinden yönelim kontrolü sağlanmıştır.

Python kodu içerisinde çok iş parçacıklı (multi-threaded) bir yapı oluşturulmuştur. Her bir kontrol ekseni için ThreadPoolExecutor sınıfı kullanılarak birden fazla worker thread tanımlanmıştır. Bu thread'ler yalnızca kendilerine evaluator (değerlendirici) thread tarafından görev verildiğinde aktif hale gelmektedir. Böylece, aynı anda birden fazla işlem gerçekleştirilerek daha kararlı ve gecikmesiz bir kontrol sağlanmaktadır.

Gerçekleştirilen sistemin en temel bileşenlerinden biri waypoint tabanlı navigasyon yapısıdır. WaypointNavigator sınıfı kullanılarak, İHA'ya belirli coğrafi koordinatlarda hedefler atanmakta ve bu hedefler arasında seyir gerçekleştirmesi sağlanmaktadır. Hedefe olan mesafe ve yön farkı hesaplanarak, uygun roll ve pitch değerleri PID (Proportional-Integral-Derivative) tabanlı kontrolörler aracılığıyla hesaplanmakta ve buna göre klavye tuşları tetiklenmektedir.

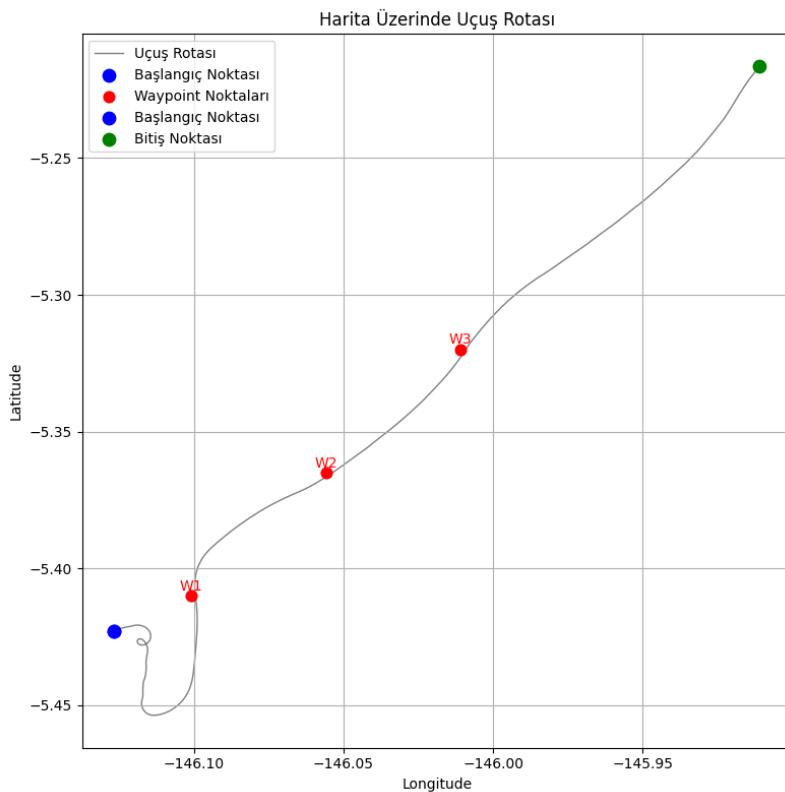
Özellikle NavigationController sınıfı içerisinde yer alan algoritmalar sayesinde, hedefe yaklaşım sırasında daha hassas yönelim komutları oluşturulmakta, mesafe artışı gibi durumlar fark edilerek kontrol davranışları otomatik olarak ayarlanmaktadır. Yaklaşma mesafesi 1000 metre olarak tanımlanmış ve bu mesafenin altına düşüldüğünde sistem "yaklaşma modu"na geçmektedir.

Roll kontrolü için geliştirilen PID algoritması, roll sapmalarını azaltmaya yönelik optimize edilmiştir. Yönelim hatasının yanı sıra, roll değişim hızı da denkleme dahil edilerek daha kararlı bir uçuş sağlanmıştır. Pitch kontrolünde ise düşey hız (ver_vel) değeri de hesaba katılarak burun açısının kontrolü yapılmaktadır.

Simülatörden alınan telemetri verileri arasında; yükseklik, enlem, boylam, hız, dikey hız, pitch, roll, yaw ve bu eksenlerin değişim hızları bulunmaktadır. Bu veriler get_flight_data fonksiyonu ile sürekli olarak alınmakta ve flight_data_stack içerisinde kaydedilmektedir. Simülasyon süresince bu veriler üzerinde işlem yapılmakta ve uygun kontrol çıktıları üretilmektedir.

Yapılan deneyler sonucunda, bu sistemin SR2 ortamında başarılı bir şekilde İHA'yı hedeflenen noktalara yönlendirebildiği, roll kararsızlıklarının ise PID temelli anti-roll yapısı sayesinde azaltıldığı gözlemlenmiştir. Geliştirilen bu yapı, hem geleneksel kontrol algoritmalarına hem de yapay zekâ destekli modellere kolaylıkla entegre edilebilecek şekilde esnek bir mimariye sahiptir.

2.2.4 Waypoint Tabanlı Otonom Navigasyon Sistemi



Şekil 2.13. İHA simülasyon sonucu izlenen rota

İHA'nın otonom bir şekilde belirlenen hedeflere yönlendirilmesi için WaypointNavigator sınıfı geliştirilmiştir. Bu sınıf, her bir waypoint'ın (konum noktasının) enlem, boylam, yükseklik ve tolerans bilgilerini tutmakta ve aktif waypoint'e ulaşılıp ulaşılmadığını kontrol etmektedir. Navigasyonun mantığı, İHA'nın hedefe olan uzaklığı calculate_distance() fonksiyonu ile hesaplanarak tolerans eşiğinin altına düştüğünde bir sonraki waypoint'e geçilmesi esasına dayanır.

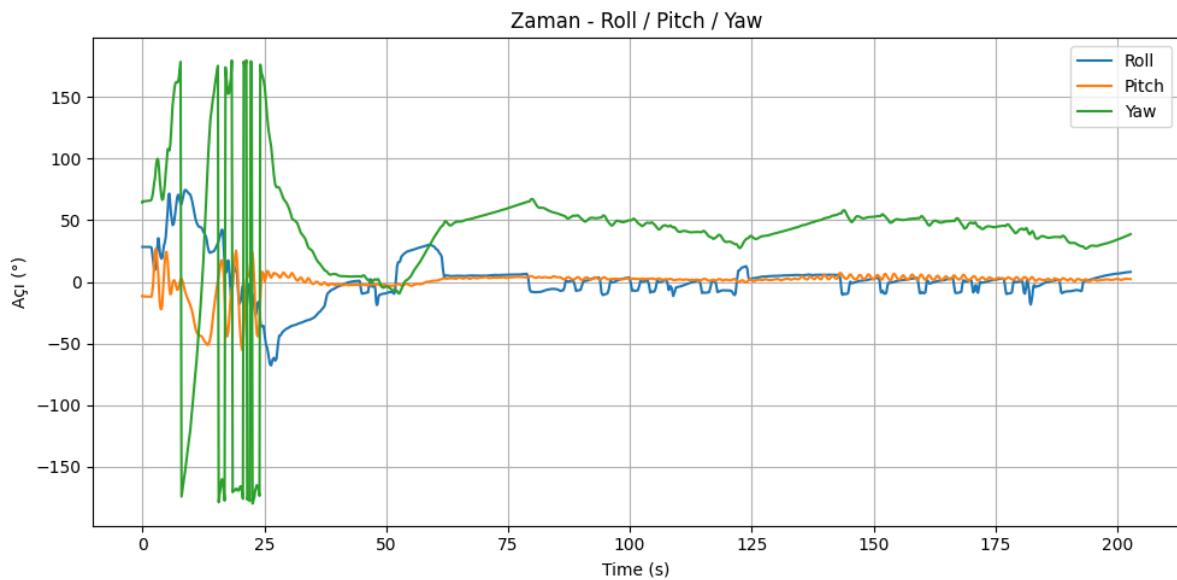
Bu yapı sayesinde rota üzerindeki tüm hedefler sırasıyla geçilerek İHA'nın rota takibi yapılır. WaypointNavigator sınıfı aşağıdaki işlevleri kapsar:

- Yeni waypoint ekleme (add_waypoint)
- Waypoint'lerin tamamını temizleme (clear_waypoints)

- Aktif waypoint'e ulaşım kontrolü (check_waypoint_reached)

Navigasyonun aktif olduğu durumlarda TB2 sınıfı içerisinde yer alan get_flight_data() fonksiyonu ile her döngüde güncellenen pozisyon verileri üzerinden waypoint'e olan mesafe hesaplanır ve yönlendirme yapılır. Şekil 2.13'de simülasyon sonucu izlenen rota gösterilmektedir.

2.2.5 NavigationController ile Yaw Tabanlı Roll ve Pitch Açılarının Hesaplanması



Şekil 2.14. İHA simülasyon sonucu roll-pitch-yaw değer değişimleri

NavigationController sınıfı, İHA'nın yönelimi için gerekli roll (yalpa) ve pitch (yükseleme/aşağı inis) komutlarını üretir. Bu komutlar, İHA'nın bulunduğu konum ile aktif waypoint'in konumu arasındaki yön farkına göre hesaplanır. Özellikle calculate_bearing() fonksiyonu, iki nokta arasındaki azimut açısını verir ve bu açı, current_yaw ile karşılaştırılarak yaw hatası hesaplanır.

Yaw hatasına göre:

- Roll değeri, yön değiştirme için kullanılır.
- Pitch değeri, yaklaşma aşamasına göre belirlenir (örneğin, hedefe yaklaşıldığından pitch sabitlenir).

Yaklaşma aşaması, mesafe 1000 m'nin altına düştüğünde devreye girer. Bu sayede son yaklaşımada daha hassas manevralar yapılabilir.

2.2.6 PID Tabanlı Stabilizasyon ve Kontrol Mekanizması

Geliştirilen sistemde, İHA'nın yönelim kararlılığını sağlamak için hem **roll** hem de **pitch** eksenlerinde PID (Proportional-Integral-Derivative) tabanlı kontrol mekanizmaları uygulanmıştır. Bu yapı sayesinde, İHA'nın uçuşu boyunca dış etkilerden ya da rotadan sapmalardan kaynaklanan kararsızlıklar giderilerek hedef doğrultusunda dengeli bir ilerleme sağlanmaktadır.

a) Roll (Aileron) Kontrolü

PID_Aileron sınıfı, hedef roll açısı ile mevcut roll arasındaki farkı dikkate alarak aileron PWM (kontrol sinyali) üretir. Bu sinyalin hesaplanması:

- **P (Proportional)** bileşeni anlık hata değerine duyarlıdır.
- **I (Integral)** bileşeni birikmiş hata miktarına göre sistemin kalıcı hataları düzeltmesini sağlar.
- **D (Derivative)** bileşeni ise hata değişim hızına tepki verir, ani değişimleri sönmeyici etki oluşturur.

Hesaplanan aileron_pwm değeri, _aileron_control() fonksiyonu aracılığıyla 'q' (sola roll) ve 'e' (sağa roll) tuşlarına basılarak yönlendirme şeklinde uygulanır. Yaklaşma fazına girildiğinde (hedefe 1000 m'den az mesafe kaldığında), bu kontrol daha yumuşak ve hassas hâle getirilir.

b) Pitch (Elevator) Kontrolü

Pitch eksenindeki kontrol ise pitch_stabilizer_function() fonksiyonu içerisinde gerçekleştirilir. Hedef pitch açısı ile mevcut pitch arasındaki fark ve füzenin dikey hızı (ver_vel) kullanılarak elevator_pwm değeri hesaplanır. Bu değer:

- elevator_pwm > 0 ise 's' tuşu (burun aşağı),
- elevator_pwm < 0 ise 'w' tuşu (burun yukarı)

şeklinde klavye girişleriyle uygulanır. Bu yapı, pitch kontrolünü gerçek zamanlı ve dinamik hâle getirir.

Her iki kontrol ekseninde de PID değerleri, simülasyonun kararlılığı ve aşırı tepkileri önlemek amacıyla sınırlanmış ve modlara göre (örneğin yaklaşma modu) esnek biçimde ayarlanabilir şekilde yapılandırılmıştır. Şekil 2.14, İHA simülasyon sonucu roll-pitch-yaw değerlerin grafiği göstermektedir

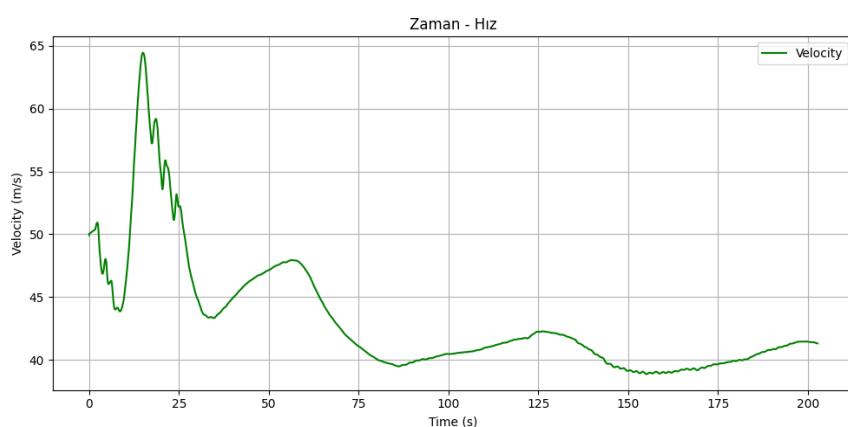
2.2.7 Çok İş Parçacıklı Kontrol ve Telemetri Yönetimi

Sistem, Python'un çok iş parçacıklı yapısından (multithreading) faydalananak roll ve pitch eksenleri için eşzamanlı kontrol sağlar. Her eksen için bir ThreadPoolExecutor oluşturulmuş ve bu havuzlara üçer adet **worker thread** atanmıştır. Bu thread'ler yalnızca yeni uçuş verisi geldiğinde (flight_data_event tetiklendiğinde) çalışır ve kontrol komutlarını uygular.

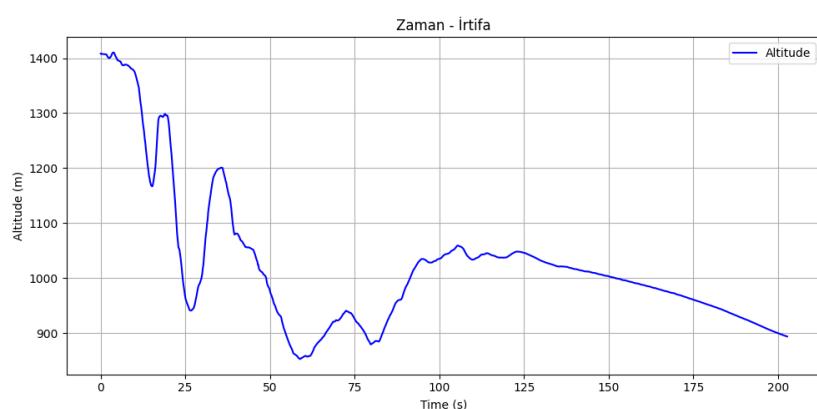
Bu mimari sayesinde:

- Roll ve pitch eksenleri bağımsız ama eşzamanlı olarak stabilize edilir.
- Her PWM komutu, ilgili eksene ait thread havuzunda bir iş parçacığı tarafından uygulanır.
- Klavye simülasyonları çakışmadan yapılır ve kaynaklar daha verimli kullanılır.
- Kontrol gecikmeleri önemli ölçüde azaltılır.

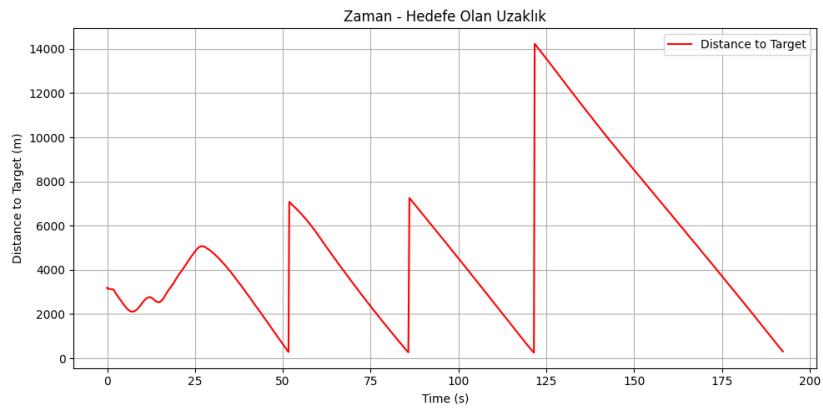
Şekil 2.15-2.17'de telemetri verisi grafikleri gösterilmektedir.



Şekil 2.15. İHA hız grafiği



Şekil 2.16. İHA simülasyon sonucu irtifa değişimi



Şekil 2.17. İHA simülasyon sonucu sırasıyla 4 hedef noktasının uzaklık değişimi

Simülatör ile Python arasındaki bağlantı, UDP protokolü kullanılarak sağlanır. `get_flight_data()` fonksiyonu, SR2 simülatöründen gelen veri paketlerini alır ve `readPacket()` fonksiyonu aracılığıyla ayırtırır.

Alınan telemetri verileri şunları içerir:

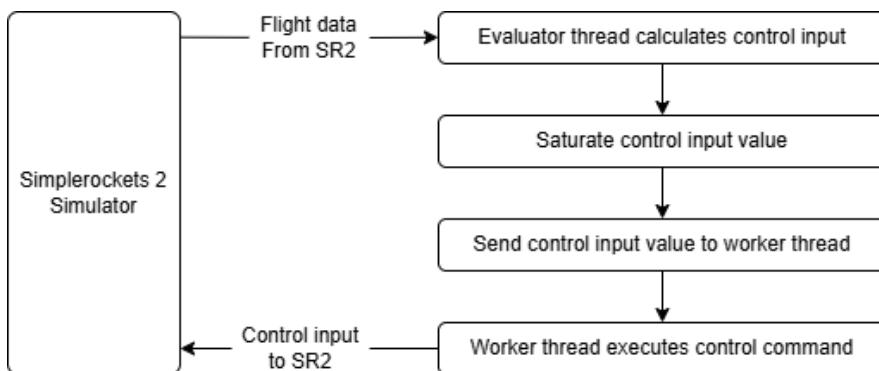
- Konum bilgileri: latitude, longitude, altitude
- Yönelim bilgileri: roll, pitch, yaw
- Hız bileşenleri: velocity, ver_vel, roll_rate, pitch_rate, yaw_rate
- Durum bilgileri: grounded (yere inmiş mi), destroyed (imha olmuş mu)

Bu veriler `flight_data_stack` isimli liste yapısında en güncel hâliyle tutulur ve tüm kontrol mekanizmaları bu veriye erişerek karar verir. Ayrıca, her yeni veri seti geldiğinde kontrol thread'leri bu verilerle birlikte tetiklenir. Bu yapı, hem verinin hem de tepkinin senkronize çalışmasını sağlar. İHA simülasyon sonucu genel veriler Tablo 2.2'de gösterilmektedir.

Tablo 2.2. İHA simülasyon sonucu genel veriler

Metrik	Değer
Toplam Uçuş Süresi (s)	202.73
Maksimum İritfa (m)	1410.30
Ortalama İrtifa (m)	1028.27
Minimum İrtifa (m)	852.67
Maksimum Hız (m/s)	64.46
Ortalama Hız (m/s)	43.23
Maksimum Roll Açısı (°)	74.57
Maksimum Pitch Açısı (°)	29.94
Başlangıç-Hedef Mesafe (m)	3198.01
Son-Hedef Mesafe (m)	6147.0
Waypoint Sayısı	3

2.2.8 Python Üzerinden Füze Kontrolü



Şekil 2.18. Python'dan SR2'ye füze kontrol akış şeması

Bu projede, füze kontrolünü sağlamak amacıyla Python programı içerisinde bir kontrol mekanizması geliştirilmiştir. Bu mekanizma, füzenin elevator, rudder ve aileron yüzeylerinin dışarıdan kontrol edilmesini kapsamaktadır. Füzenin kontrol yüzeylerini dışarıdan yönetebilmek için PyKey kütüphanesi kullanılmaktadır [28]. Bu kütüphane, klavye tuşlarının sanal olarak basılmasını simüle etmektedir.

SR2 simülatöründen gelen uçuş verileri Python tarafından alındıktan ve işlendikten sonra, Python programı, füzenin kontrolünü sağlamak için klavye girişlerini otomatik olarak simüle etmektedir. SR2 simülatöründe, ‘w’ ve ‘s’ tuşları elevator kontrolü, ‘a’ ve ‘d’ tuşları rudder kontrolü, ‘q’ ve ‘e’ tuşları ise aileron kontrolü için kullanılmaktadır. Python programı, bu tuş girişlerini dışarıdan tetikleyerek füzenin yönlendirilmesini sağlamaktadır.

Her bir kontrol yüzeyi için iki tür iş parçacığı kullanılmaktadır: evaluator thread ve worker thread. Evaluator thread, tuş basım süresini saniye cinsinden hesaplar ve ilgili worker thread'e bu tuş basımını gerçekleştirmesi için komut verir. Evaluator thread tarafından hesaplanan kontrol girdileri, geleneksel kontrol yöntemleri ile yapay zeka tabanlı yöntemler kullanılarak elde edilmektedir. Bu yöntemlerin detayları bir sonraki bölümde ayrıntılı şekilde açıklanacaktır.

Evaluator thread tarafından belirlenen kontrol komutu çıktı süresi, SR2 tarafından gönderilen verilerin frekansına uyumlu olması amacıyla trim edilmektedir. Böylece worker thread, evaluator thread tarafından gönderilen kontrol komutunu anlık olarak uygulayabilmektedir. Ayrıca, her bir kontrol ekseni için birden fazla worker thread eşzamanlı olarak çalıştırılmaktadır. Bu süreçte ThreadPool yaklaşımı benimsenerek

Python ortamında çoklu iş parçacığı yönetimi kolaylaştırılmıştır. Kontrol algoritmasının akış şeması Şekil 2.18'da gösterilmektedir.

Worker thread'ler, thread pool içerisinde beklemeye kalır ve yalnızca evaluator thread tarafından bir kontrol komutu gönderildiğinde çalışır. Thread pool içerisinde yer alan worker thread sayısı ihtiyaca göre değiştirilebilir; ancak varsayılan olarak, her bir kontrol yüzeyi için thread pool'a üç adet worker thread atanmıştır.

Bu çalışmada ele alınan temel problem, çoklu sensörler kullanarak füzenin hedefe otonom şekilde yönlendirilmesini sağlayan bir algoritmanın geliştirilmesidir. Bu kapsamda, IMU ve GPS sensörleri temel sensörler olarak kullanılmaktadır. Ayrıca, füzenin ön kısmına yerleştirilen bir kamera, hedef tespitini desteklemek ve isabet hassasiyetini artırmak amacıyla görsel veri toplamak için kullanılmaktadır. Swarm (sürü) füze sisteminin başlıca amaçlarından biri, hedefin imha edilme olasılığını artırmaktır. Bu nedenle, sürüdeki her bir füzenin hedefe farklı açılardan ve eşzamanlı olarak isabet etmesi beklenmektedir.

Sonraki bölümlerde, navigasyon ve kontrol probleminin çözümüne yönelik çeşitli yöntemler ele alınmaktadır. İlk olarak, her bir füzenin terminal yaw açısına ulaşmasını sağlamak için waypoint kullanımı ile başlanır. Bireysel füzeler için terminal yaw açısı algoritmalarının formülasyonu açıklanmaktadır. Bu çalışmada hedeflenen terminal yaw açıları 0° , 90° ve 180° olarak belirlenmiştir. 270° terminal yaw açısı ise, 90° terminal açısına benzer hesaplamalara dayandığından ayrıca ele alınmamıştır. Özellikle 90° ve 180° terminal açıları için optimal waypoint trajektorilerinin oluşturulmasında Bezier eğrisi formülü kullanılmaktadır.

Terminal pitch açısı da isabetli vuruş olasılığını artıran önemli unsurlardan biridir. Bu çalışmada, terminal pitch açıları 30° ila 80° arasında formüle edilmiştir. Problemin çözümünde oransal navigasyon (proportional navigation) yöntemi kullanılmaktadır. Füze hedefe yöneldiğinde, ön kısmındaki kamera tarafından görsel veriler toplanır. Nesne tespit algoritması sayesinde füze, pitch açısını otomatik olarak ayarlayarak dikey eksendeki yörüngesini daha hassas bir şekilde hedefe yönlendirebilmektedir.

Ayrıca, bu çalışmada karşılaşılan önemli sorunlardan biri de füzenin roll açısının kararsızlığı olmuştur. Füze, rudder veya elevator kontrol girdisi aldığında roll açısında istenmeyen dalgalanmalar meydana gelmektedir. Bu durum, rudder ve elevator kontrol performansını olumsuz etkilemektedir. Bu nedenle, rudder ve elevator kontrolü sırasında roll sapmalarını dengelemek amacıyla bir anti-roll sistemi gerekmektedir. Anti-roll sisteminin geliştirilmesinde, hem geleneksel kontrol yöntemleri (PID kontrolörü) hem de yapay zeka tabanlı yöntemler (doğrusal regresyon, LSTM ve pekiştirmeli öğrenme) uygulanmıştır.

2.2.9 Terminal Yaw Açısı Kontrolü

Swarm füze saldırısında maksimum etkinliğin sağlanabilmesi için, her bir füzenin hedefe farklı açılardan saldırısı yapabilecek şekilde saldırısı açısını ayarlayabilmesi gerekmektedir. Bu kapsamında kontrol edilen en önemli parametrelerden biri terminal yaw açısındandır; bu, füzenin hedefe çarpmadan önceki son yatay yön açısını ifade etmektedir.

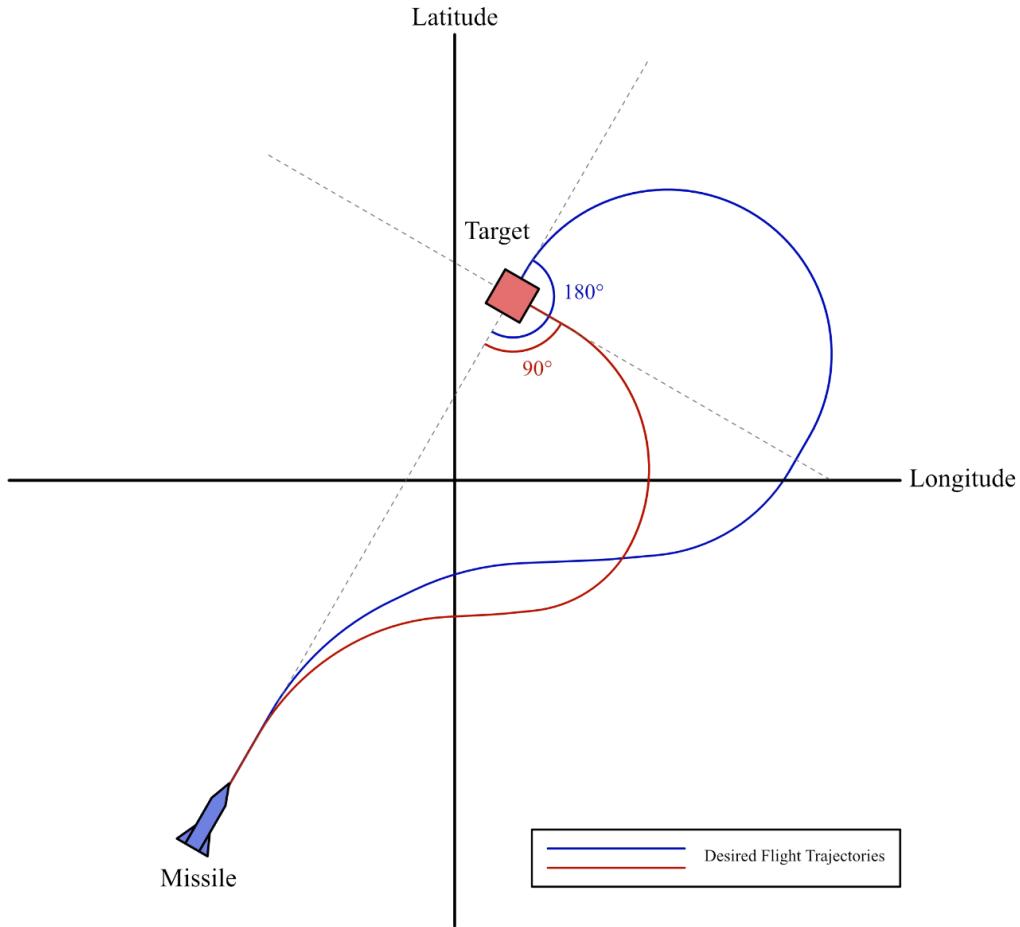
Bu çalışmada odaklanılan üç terminal yaw açısı şunlardır:

- 0° (hedefin ön tarafından yapılan saldırısı)
- 90° (hedefin sağ tarafından yapılan saldırısı)
- 180° (hedefin arka tarafından yapılan saldırısı)

270° terminal yaw açısı ise ayrıca ele alınmamıştır, çünkü bu açı 90° terminal yaw açısının ters yöndeki hesaplamasına karşılık gelmektedir. Şekil 2.19'de terminal yaw açıları görselleştirilmektedir.

Şekil 2.19'de gösterildiği üzere, sadece waypoints (ara noktalar) belirlemek değil, aynı zamanda bu waypoint koordinatlarını hesaplamak da karmaşıktır. Bunun nedeni, füzenin hedefe göre olan lokal koordinatlarının dünya üzerinde kullanılan küresel koordinatlara (enlem ve boylam) dönüştürülmesinin gereklisidir.

90° ve 180° terminal yaw açıları için waypoint hesaplamalarında kullanılan algoritma Algoritma 1 ve 2 de anlatılmıştır.



Şekil 2.19. İstenilen terminal yaw açıları ve uçuş yörüngesi

90° ve 180° terminal yaw açılarını kontrol etmek için, waypoint oluşturma ve Bezier eğrisine dayalı bir yörünge planlama algoritması kullanılmaktadır. Bezier eğrisi, füze için düzgün (smooth) ve esnek bir rota üretebildiği için tercih edilmiştir; bu da füzenin verimli manevralar yaparak waypoint'lere yaklaşırken keskin açı değişimlerinden kaçınmasını sağlar.

Bezier eğrisi formülü aşağıda verilmiştir ve Şekil 2.20, bu formülün Python'daki Matplotlib kütüphanesi ile görselleştirilmiş halini göstermektedir:

$$B(t) = (1 - t)3P_0 + 3(1 - t)2tP_1 + 3(1 - t)t^2P_2 + t^3P_3, t \in [0,1] \quad (2.1)$$

Burada:

- P_0, P_1, P_2, P_3 kontrol noktalarını (her biri x ve y koordinatlarıyla) ifade etmektedir.
- t , eğri parametresidir (0 ile 1 arasında değişmektedir).

- $B(t) = (x(t), y(t))$, t anındaki eğri üzerindeki bir noktanın koordinatlarını vermektedir.

```

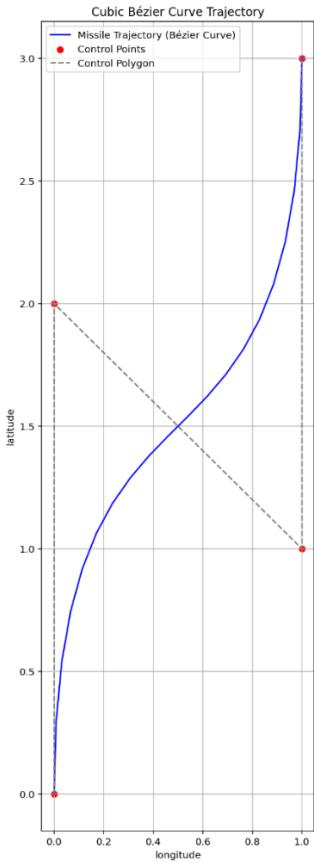
def bezier_curve(t, P0, P1, P2, P3):
    x = (1 - t)**3 * P0[0] +
        3 * (1 - t)**2 * t * P1[0] +
        3 * (1 - t) * t**2 * P2[0] +
        t**3 * P3[0]
    y = (1 - t)**3 * P0[1] +
        3 * (1 - t)**2 * t * P1[1] +
        3 * (1 - t) * t**2 * P2[1] +
        t**3 * P3[1]
    return x, y

# Define control points
P0 = (0, 0)
P1 = (0.12, 0.57)
P2 = (0.63, 0.21)
P3 = (1, 1)

# Generate curve points
t_values = np.linspace(0, 1, 10)
curve_points = np.array([bezier_curve(
    t, P0, P1, P2, P3) for t in t_values])

# Plot the Bézier curve
plt.plot(curve_points[:, 0], curve_points[:, 1],
         label="Bézier Curve", color="blue")

```



Şekil 2.20. Python'da Bezier eğrisi kodu ve görselleştirilmesi

2.2.10 Terminal Pitch Açısı Kontrolü

Terminal yaw açısına ek olarak, terminal pitch açısı da hedefin hangi kısmının vurulacağını belirlediği için kritik bir parametredir. Pitch açısını kontrol etmek için Proportional Navigation Guidance (PNG) yöntemi kullanılmaktadır. Bu yöntem, hedefe yönelik uçuş yolunu düzeltmek amacıyla interceptor ve füze kontrolünde kullanılan en basit ve etkili yaklaşımardan biri olduğu için tercih edilmiştir.

Bu çalışmada sunulan simülasyonda PNG, füzenin hedefe göre olan bağıl hızı ve hedefe yönelik Line of Sight (LOS) (görüş hattı) açısı bilgilerini kullanarak çalışmaktadır. Genel olarak, PNG algoritması füzenin lateral (yanal) ivmesini, LOS açısının değişim hızına (rate of LOS) orantılı olarak ayarlamaktadır.

Algoritma 1. 90° Terminal yaw açısı için yörünge hesaplama algoritması

- 1 Sabit değişkenler tanımlanır:
 - Düz mesafe: D
 - Gezegenin yarıçapı: r
 - Çap dönüşüm noktası: R
 - 2 Öncelikle mevcut GPS verilerinin (enlem φ , boylam λ , ve yaw ψ) alınması beklenir.
 - 3 İlk döngüde başlangıç konumu ($\varphi_{initial}, \lambda_{initial}$) olarak kaydedilir.
 - 4 Hedef pozisyon bilindiğinde ($\varphi_{target}, \lambda_{target}$):
 - 5 Düz mesafe D derece cinsinden şu formül ile hesaplanır:
$$D = \text{degrees}(\widehat{D} / r)$$
 - 6 Yaw ψ değeri, uçuş yönünü belirlemek için bir heading açısına dönüştürülür.
 - 7 Eğer heading yönü Kuzey veya Güney ise:
 - i. Aşağıdaki hesaplamlar yapılır:
 - 9 $\Delta\varphi = \cos(90 - \psi_{current}) \times D$
 - $\Delta\lambda = \cos(\psi_{current}) \times D$
 - $\Delta\varphi_r = \cos(\psi_{current}) \times R$
 - $\Delta\varphi_r = \cos(90 - \psi_{current}) \times R$
 - ii. Yönün kuzey ya da güney olmasına bağlı olarak, eğer güney ise:
$$\text{waypoint}_1^\varphi = \varphi_{target} + \Delta\varphi$$
$$\text{waypoint}_1^\lambda = \lambda_{target} - \Delta\lambda$$
 - 11 Sonra düz uçuş segmentinin ilerisinde bir dönüş noktası tanımlanır:
$$\text{waypoint}_2^\varphi = \text{waypoint}_1^\varphi + \Delta\varphi_r$$
$$\text{waypoint}_2^\lambda = \text{waypoint}_1^\lambda - \Delta\varphi_r$$
 - 12 Başlangıç noktası ($\varphi_{initial}, \lambda_{initial}$) ile dönüş noktası ($\text{waypoint}_2^\varphi, \text{waypoint}_2^\lambda$) arasında düzgün bir geçiş rotası oluşturmak için Bezier eğrisi kullanılır.
 - 13 Nihai izlenecek rota (Final Waypoints): Bezier eğrisinden elde edilen yol noktaları + $(\text{waypoint}_1^\varphi, \text{waypoint}_1^\lambda) + (\varphi_{target}, \lambda_{target})$
-

Algoritma 2. 180° Terminal yaw açısı için yörünge hesaplama algoritması

1 Sabit değişkenler tanımlanır:

Düz mesafe: D

Gezegenin yarıçapı: r

Çap dönüşüm noktası: R

2 Öncelikle mevcut GPS verilerinin (enlem φ , boylam λ , ve yaw ψ) alınması beklenir.

3 İlk döngüde başlangıç konumu ($\varphi_{initial}, \lambda_{initial}$) olarak kaydedilir.

4 Hedef pozisyon bilindiğinde ($\varphi_{target}, \lambda_{target}$):

5 Düz mesafe D derece cinsinden şu formül ile hesaplanır:

$$D = \text{degrees}(\widehat{D} / r)$$

6 Yaw ψ değeri, uçuş yönünü belirlemek için bir heading açısına dönüştürülür.

7 Eğer heading yönü Kuzey veya Güney ise:

8 i. Aşağıdaki hesaplamlar yapılır:

$$\Delta\varphi = \sin(90 - \psi_{current}) \times D$$

$$\Delta\lambda = \sin(\psi_{current}) \times D$$

$$\Delta\varphi_r = \sin(\psi_{current}) \times R$$

$$\Delta\varphi_r = \sin(90 - \psi_{current}) \times R$$

10 ii. Yönüñ kuzey ya da güney olmasına bağlı olarak, eğer güney ise:

$$\text{waypoint}_1^\varphi = \varphi_{target} - \Delta\varphi$$

$$\text{waypoint}_1^\lambda = \lambda_{target} + \Delta\lambda$$

11 Sonra düz uçuş segmentinin ilerisinde bir dönüş noktası tanımlanır:

$$\text{waypoint}_2^\varphi = \text{waypoint}_1^\varphi - \Delta\varphi_r$$

$$\text{waypoint}_2^\lambda = \text{waypoint}_1^\lambda - \Delta\varphi_r$$

12 Başlangıç noktası ($\varphi_{initial}, \lambda_{initial}$) ile dönüş noktası ($\text{waypoint}_2^\varphi, \text{waypoint}_2^\lambda$) arasında düzgün bir geçiş rotası oluşturmak için Bezier eğrisi kullanılır.

13 Nihai izlenecek rota (Final Waypoints): Bezier eğrisinden elde edilen yol noktaları + $(\text{waypoint}_1^\varphi, \text{waypoint}_1^\lambda) + (\varphi_{target}, \lambda_{target})$



Şekil 2.21. Hedef görsellerin ve sınırlayıcı kutularının örnekleri

Bu durumda pitch açısını kontrol etmek için kullanılan PNG'nin matematiksel formülasyonu şu şekilde ifade edilebilir:

$$a_n = N \cdot V_c \cdot \lambda \quad (2.2)$$

Burada:

- a_n , pitch kontrolüne çevrilecek olan lateral ivmeyi ifade etmektedir.
- N , navigation constant (genellikle 3 ile 5 arasında bir değere sahiptir).
- V_c , füze ile hedef arasındaki kapanma hızıdır (closing velocity).
- λ , pitch eksenindeki (düsey eksen) LOS açısının değişim hızıdır.

PNG yaklaşımına ek olarak, bu çalışmada object detection (nesne tespiti) yöntemine dayalı deneysel bir yöntemle de pitch açısının kontrolü gerçekleştirılmıştır. Füze hedefe yaklaştığında, ön kısımda yer alan kamera aktif hale getirilerek YOLO (You Only Look Once) gibi algoritmalarla nesne tespiti yapılır. Elde edilen görsel veriler, füzenin pitch açısının hassas bir şekilde hedefe yönlendirilmesini sağlamak amacıyla işlenir. Hedefin görsel olarak başarılı bir şekilde tespit edilmesinden sonra, pitch kontrol sistemi PNG'den görsel tabanlı düzeltme sistemine geçer ve bu sistem tespit edilen hedefin bounding box bilgisine dayanır. Hedef kamera çerçevesinin alt kısmında görünüyorrsa, füze pitch açısını artırır. Tersine, hedef üst kısmında görünüyorsa, füze pitch açısını azaltır. Bu işlem, hedef kamera çerçevesinin ortasında kilitlenene kadar devam eder.

2.2.11 Hedef Tespiti İçin YOLOv8 Modelinin Eğitimi

Füzenin üzerindeki kamera aracılığıyla hedefi doğru bir şekilde tanımlayabilmesi için YOLOv8 algoritması [29] kullanılarak bir nesne tespiti modeli eğitilmiştir.

YOLO (You Only Look Once), tam görüntü üzerinden tek geçişte doğrudan bounding box (sınırlayıcı kutu) ve sınıf olasılıklarını tahmin eden gerçek zamanlı bir nesne tespit sistemidir.

YOLOv8, daha gelişmiş bir backbone ve neck mimarisi ile anchor-free (çapa içermeyen) tasarım ve decoupled head (ayrık baş) yapısı sayesinde doğruluğu artırır ve hesaplama maliyetini azaltır [30]. YOLOv8, yüksek çıkarm (inference) hızı ve yeterli doğruluğu nedeniyle tercih edilmiştir ve bu özelliklerini sayesinde füze navigasyon sistemleri gibi gerçek zamanlı uygulamalar için son derece uygundur.

a.) Veri seti hazırlama

YOLOv8 modelinin eğitim süreci, ilgili hedeflerin yer aldığı bir veri seti toplanarak başlatılmıştır. Veri seti, sahadaki değişken koşullara karşı modelin dayanıklılığını artırmak amacıyla farklı açılardan ve çeşitli mesafelerden görülen hedefleri içeren çeşitli senaryolardan oluşmaktadır.

Veri seti, SR2 simülatörü kullanılarak otomatik olarak alınan ekran görüntüleriyile toplanmış ve hedef olarak belirlenen Surface-to-Air Missile (SAM) launcher vehicle (karadan havaya füze fırlatma aracı) Roboflow aracı [31] ile etiketlenmiştir. Toplamda 1.000 ekran görüntüsü toplanmış ve her görüntüde hedefin konumunu gösteren bir bounding box çizilmiştir. Şekil 2.21 hedefe ait bazı örnek görüntüler ve bunların bounding box'larını göstermektedir.

Dataset, -15° ile 15° arasında döndürme işlemleri içeren data augmentation (veri artırma) yöntemiyle genişletilmiştir ve sonuç olarak 1.480 eğitim verisi, 150 doğrulama verisi ve 110 test verisi elde edilmiştir.

b.) Eğitim süreci

Model, Python ortamında Ultralytics YOLO framework kullanılarak ve daha önce COCO veri seti üzerinde eğitilmiş olan önceden eğitilmiş model ile eğitilmiştir. Eğitim sırasında kullanılan temel parametreler aşağıdaki gibidir:

- Learning rate: $10^{-4} – 10^{-8}$
- Batch size: 16
- Görüntü boyutu: 800x800
- Optimizasyon algoritması: AdamW [32]
- Epochs sayısı: 100 (önceden eğitilmiş model ile tekrar eğitilmiştir)

Şekil 2.22 ve Şekil 2.23, YOLOv8 algoritmasının N ve M modelleri için eğitimi süresince elde edilen metrik geçmişini göstermektedir. N modeli yaklaşık 3,2 milyon parametreye sahipken, M modeli 25,9 milyon parametre içermektedir. Bu parametre sayılarındaki belirgin fark,

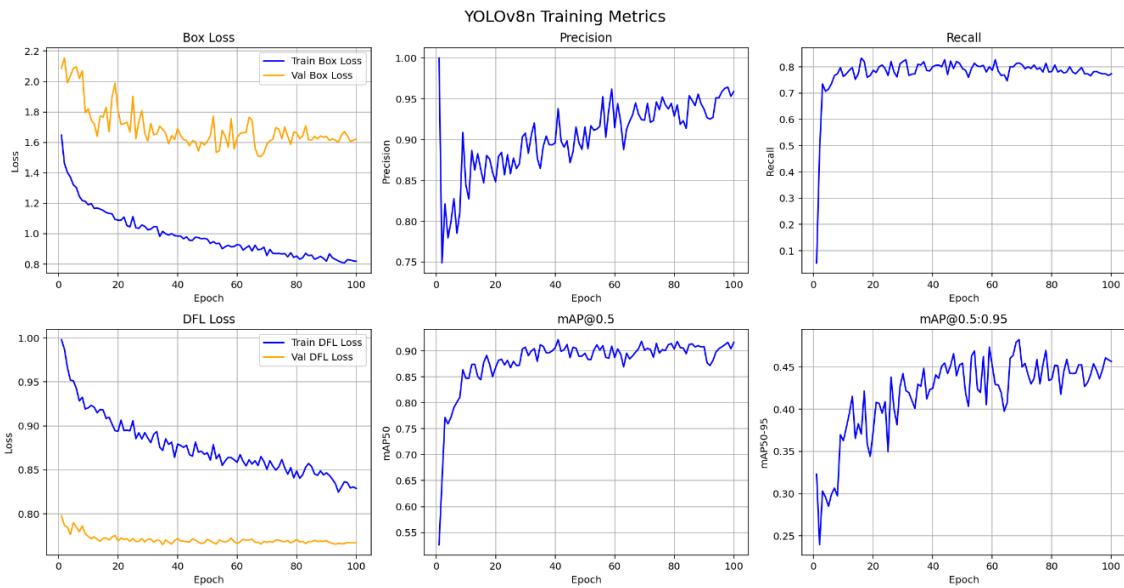
modellerin performansını doğrudan etkilemektedir. N modeli daha hafif ve hızlı çıkarım (inference) süresi sunmasına rağmen, doğruluk (accuracy) açısından M modelinin gerisinde kalmaktadır.

c.) Füze navigasyon sistemine entegrasyon

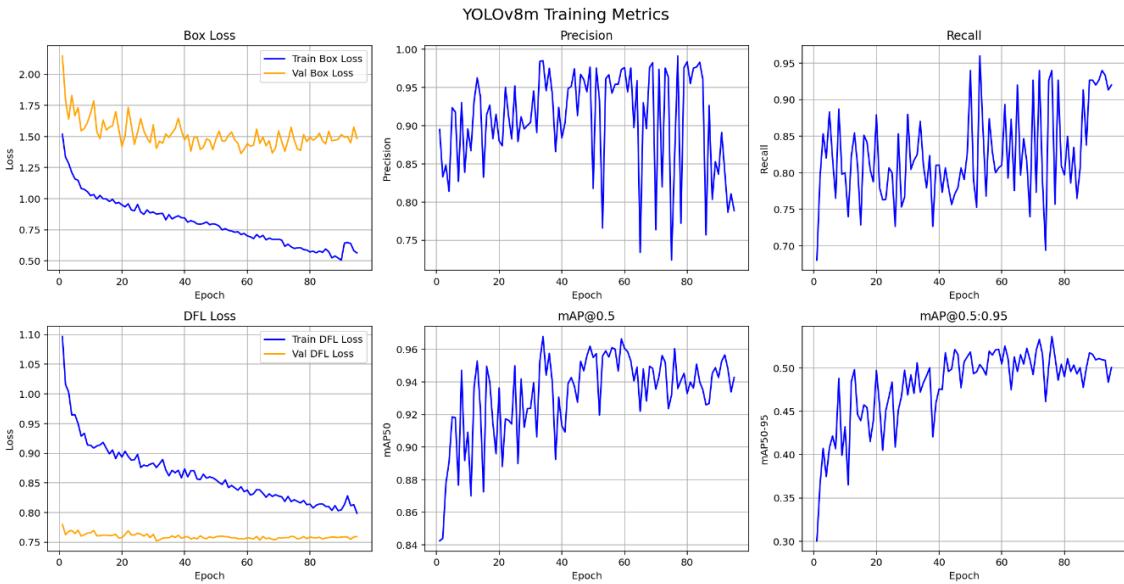
Eğitilmiş YOLOv8 modeli, füze navigasyon sistemine entegre edilmiştir. Füze hedefe yaklaşırken ve yönelirken, entegre kamera sistemi aracılığıyla YOLOv8 algoritması devreye girerek gerçek zamanlı hedef tespiti gerçekleştirilmektedir. Tespit edilen hedefin konumu (bounding box'ın merkez noktası), füzenin pitch ve yaw açılarını dinamik olarak ayarlamak üzere bir geri besleme döngüsü içinde kullanılmakta ve bu sayede füzenin hedefe daha hassas bir şekilde yönelmesi sağlanmaktadır.

Füzenin pitch açısını kontrol eden elevator yüzeyinin kontrol girdisini elde etmek amacıyla, bir yapay sinir ağı (YSA) modeli eğitilmiş ve sistemin kontrol birimine entegre edilmiştir. Bu YSA modeli; kamera sensöründen elde edilen görüntüdeki hedefin merkez noktasının Y ekseninden sapma miktarı (distance Y), bounding box'ın genişliği (width) ve yüksekliği (height) olmak üzere üç girdi almaktadır. Model, bu girdilere karşılık olarak elevator kontrolü için gerekli PWM değerini üretmektedir. Tasarlanan YSA modelinin mimarisini Şekil 2.24'te gösterilmektedir.

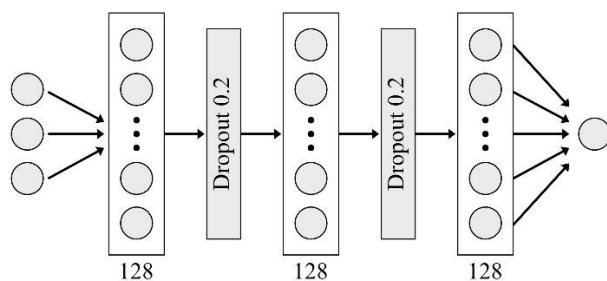
Kontrol sistemi, füzenin hedef olarak yer tabanlı bir araç tespit etmesi durumunda, GPS verileri yerine kamera verileri ve YSA modelinden elde edilen kontrol tahminlerini kullanacak şekilde tasarlanmıştır. Böylece füze, yalnızca IMU ve GPS sensörlerine bağımlı kalmaksızın, görsel veriye dayalı olarak yol düzeltmeleri gerçekleştirebilmekte ve hedefe yönelikmede daha esnek bir kontrol stratejisi izleyebilmektedir.



Şekil 2.22. YOLOv8n modelin eğitim metrikleri



Şekil 2.23. YOLOv8m modelin eğitim metrikleri



Şekil 2.24. Elevator kontrol girdisini üreten YSA modeli

2.2.12 Füze Roll Stabilizasyon Sistemi

Füzelerdeki roll stabilitesi, kontrol ve navigasyon sistemlerinde karşılaşılan temel zorluklardan biridir. Rudder ve elevator manevraları sırasında meydana gelen istenmeyen roll açısı değişimleri, füzenin rota stabilitesini bozmakta ve pitch ile yaw kontrolünün etkinliğini azaltmaktadır. Bu sorunu çözmek için geliştirilen anti-roll stabilizasyon sistemi, füze hedefe doğru manevra yaparken roll açısının sıfır dereceye yakın tutulmasını amaçlamaktadır.

Bu sistem, rudder ve elevator kontrol sistemleriyle paralel çalışan bağımsız bir kontrolcü olarak tasarlanmıştır. Sistem, SR2'den iletilen IMU sensöründen aldığı roll açısı ve roll hızı bilgilerini giriş olarak alır ve ortaya çıkan roll sapmalarını dengelemek için aileron kontrol sinyallerini PWM (Pulse-width modulation) formatında çıkış olarak üretmektedir.

Bu çalışmada 4 farklı kontrolör tasarlanmıştır.

a.) Oransal-Türevsel-İntegral (PID) kontrolör tasarıımı

Oransal-İntegral-Türevsel (PID) kontrol yöntemi, geliştirilen diğer kontrolülerin performanslarının karşılaştırılabilmesi amacıyla temel (referans) kontrolcü olarak kullanılmıştır. Roll stabilizasyon sisteminde PID kontrolörü, yaw veya pitch manevraları sırasında sistemde meydana gelen dengesizlik kaynaklı yuvarlanma hareketlerini dengeleyerek stabilizasyon sağlamakla görevlidir. PID kontrolörü, füzenin sensörlerinden aldığı veriler doğrultusunda yuvarlanma açısı hmasını, yuvarlanma hızı değerini ve birikimli yuvarlanma açısı hmasını hesaplayarak uygun kontrol sinyalini üretmektedir.

PID kontrolörünün tasarımındaki başlıca zorluklardan biri, K_p (oransal kazanç), K_i (integral kazanç) ve K_d (türevsel kazanç) sabitlerinin hassas bir şekilde ayarlanmasıdır. Bu parametrelerin en uygun değerlerinin belirlenmesinde deneysel (deneme-yanılma) yöntemi kullanılmıştır.

Ayrıca, PID kontrolörü, füze uçuş simülasyonları sırasında veri toplama amacıyla da temel kontrolcü olarak kullanılmıştır. Toplanan veriler, daha sonra yapay zekâ tabanlı kontrol yöntemlerinin eğitimi için kullanılmak üzere bir eğitim veriseti haline getirilmiştir. Bu veriler; yuvarlanma açısı (derece cinsinden), yuvarlanma hızı (derece/saniye cinsinden) ve PWM tabanlı kontrol girişini (saniye cinsinden) içermektedir. Uçuş süreci boyunca yaklaşık 5800 adet yuvarlanma açısı-yuvarlanma hızı-aileron PWM kontrol girdisi üçlüsü toplanarak, yapay zekâ modellerinin eğitimi için kullanılabilcek bir verisetine dönüştürülmüştür.

b.) Doğrusal regresyon tabanlı kontrolör

Doğrusal regresyon, özellikle karmaşıklığı yüksek olmayan problemler için etkili ve yaygın olarak kullanılan basit bir öğrenme algoritmasıdır. Bu çalışmada doğrusal regresyon modelinin uygulanmasındaki temel amaç, belirli giriş verilerine karşılık gelecek çıkış değerini tahmin edebilecek bir fonksiyonun öğrenilmesidir. Bu bağlamda, giriş özellikleri yuvarlanma açısı ve yuvarlanma hızı iken, çıkış değeri ise aileron kontrolü için kullanılan PWM (Pulse Width Modulation) değeridir.

Makine öğrenimi sürecinde, giriş X ile çıkış Y arasında bir eşleme yapan bir fonksiyon tanımlanmakta ve bu doğrultuda aşağıdaki varsayımdan (hipotez) fonksiyonu elde edilmektedir [33]:

$$h(x) = \sum_{i=0}^n \theta_i x_i = \theta^T x, x_0 = 1 \quad (2.3)$$

Burada,

- $h(x)$: öğrenmeye çalıştığımız hipotez fonksiyonudur.
- θ_i giriş X'ten çıkış Y'ye doğrusal fonksiyon eşlemesini tanımlayan parametrelerdir (ağırlıklar).
- x_i öğrenme probleminin özelliklerini ifade eder (bu durumda girdiler yuvarlanma açısı ve yuvarlanma hızıdır).
- n giriş değişkenlerinin sayısıdır.

Vektörleştirilmiş formda, hem θ hem de x vektörler olarak temsil edilir. x_0 'ın, kesme terimini (intercept term) temsil etmek için 1 olarak ayarlandığını belirtmek önemlidir.

animlanan hipotez fonksiyonundan yola çıkılarak, giriş verisi x 'in çıkış nasıl yansıtılacağına dair bir strateji geliştirilmesi gerekmektedir. Bu amaçla, hipotez fonksiyonu $h(x)$ 'in, veri kümesindeki gerçek çıkış değerleri y 'ye olabildiğince yakınsaması sağlanmalıdır. Her bir parametre θ için $h(x^{(i)})$ 'nin $y^{(i)}$ 'ye ne kadar yakın olduğunu ölçen aşağıdaki denklem, maliyet fonksiyonu (cost function) $J(\theta)$ olarak adlandırılmaktadır:

$$J(\theta) = \frac{1}{2} \sum_{i=0}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \quad (2.4)$$

Regresyon modelini eğitmek amacıyla, maliyet fonksiyonu $J(\theta)$ 'yı en aza indiren optimal ağırlık parametreleri θ 'yı bulmak için En Küçük Kareler Yöntemi (Least Mean Squares, LMS) olarak da bilinen gradyan inişi (gradient descent) algoritması kullanılmaktadır. Her bir eğitim örneği i için güncelleme kuralı şu şekilde tanımlanır:

$$\theta_j \leftarrow \theta_j + \alpha \left(y^{(i)} - h_{\theta}(x^{(i)}) \right) x_j^{(i)} \quad (2.5)$$

Burada,

- θ_j , j-ninci girdi için ağırlık parametresidir.
- α , maliyeti en aza indirmeye yönelik adım boyutunu belirleyen öğrenme oranıdır.
- $y^{(i)}$, örnek i için gerçek çıktıdır (kanatçık girişi).
- $h_{\theta}(x^{(i)})$, örnek i için tahmin edilen çıktıdır.
- $x_j^{(i)}$, örnek i için tahmin edilen çıktıdır.

Doğrusal regresyon modelinin eğitilmesinden sonra, aşağıdaki gibi doğrusal bir denklem elde edilir:

$$y = -0.00193 \cdot \Phi + 0.00550 \cdot p + 0.00005 \quad (2.6)$$

- Φ yuvarlanma açısı (roll) girişi (derece °)
- p yuvarlanma hızıdır (roll rate, saniye başına derece)

Tahmin edilen çıktı değeri, füzenin roll stabilizasyonunu sağlamak amacıyla Python çok iş parçacıklı (multithreading) mekanizması kullanılarak simülasyon ortamına iletilmektedir.

c.) Long Short-Term Memory (LSTM)

Sensörlerden elde edilen uçuş verileri ardışık bir yapıya sahiptir; bu da geçmiş, mevcut ve gelecekteki veri noktaları arasında zamansal bir ilişki olduğunu göstermektedir. Bu durum, her bir veri noktasının bağımsız olmadığı; aksine, sistemin önceki durumlarından etkilendiği anlamına gelmektedir. Örneğin, füzenin mevcut yuvarlanma (roll) açısı ve yuvarlanma hızı, önceki zaman adımlarındaki değerlerden etkilenmekte ve aynı şekilde sonraki zaman adımlarındaki değerleri de etkilemektedir. Bu zamansal bağımlılık, zaman serisi verilerinin temel bir özelliğidir ve gerek kestirim gerekse kontrol amacıyla model tasarımları yapılırken

mutlaka dikkate alınmalıdır. Geleneksel ileri beslemeli (feedforward) yapay sinir ağları, ardışık veri içerisindeki gizli patternleri yakalayamadıkları için bu tür görevlerde yetersiz kalmaktadır.

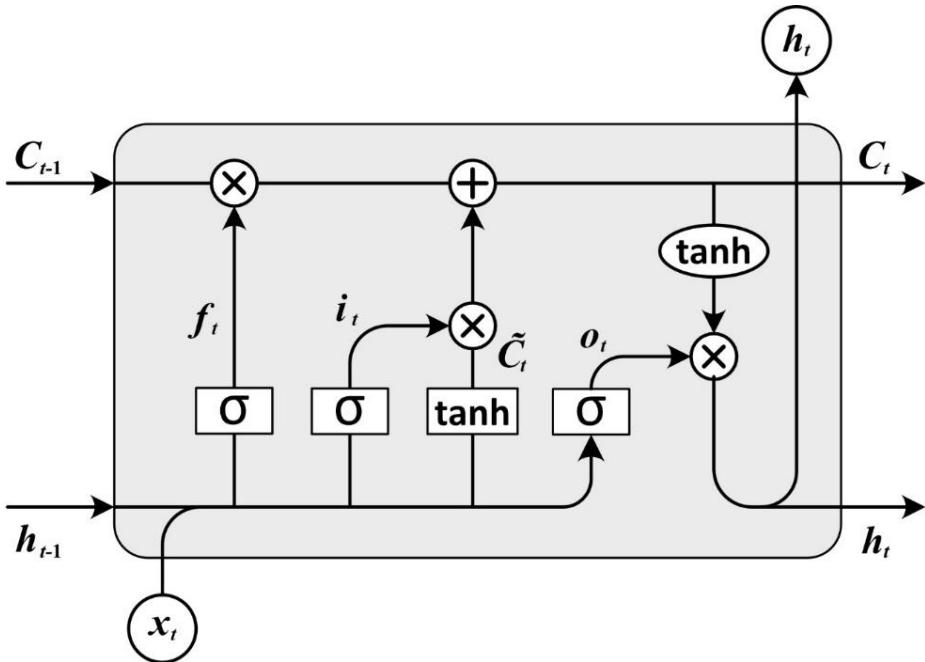
Uzun Kısa Süreli Bellek (Long Short-Term Memory, LSTM) ağları, bu tür ardışık veri problemlerini çözmek amacıyla geliştirilmiş Tekrarlayan Sinir Ağı (Recurrent Neural Network, RNN) türüdür. LSTM, klasik RNN'lerin karşılaştığı gradyan sömümlenmesi (vanishing gradient) ve gradyan patlaması (exploding gradient) gibi problemlerin üstesinden gelmek için daha karmaşık bir bellek hücresi mimarisi içermektedir. LSTM'nin detaylı mimarisi Şekil 2.25'te gösterilmektedir.

Bu bağlamda, LSTM tabanlı bir kontrolör tasarıminın temel amacı, uygun aileron kontrol girişlerini ögrenerek füzenin yuvarlanma açısından sapmaları azaltmaktadır. Doğrusal regresyon modellerinin aksine, LSTM modelleri yalnızca bir anlık veriye göre çıkış üretmeye kalmaz; bunun yerine, önceki 10 zaman adımına (yaklaşık 2 saniyelik geçmiş; 5 Hz örneklemeye frekansına göre) ait yuvarlanma açısı ve yuvarlanma hızı bilgilerini işlemektedir. Bu zamansal bağlamın modele sağlanması, füzenin gerçek zamanlı olarak stabilize edilmesi için gerekli kontrol sinyallerinin daha isabetli şekilde tahmin edilmesini mümkün kılmaktadır.

Bu çalışmada kullanılan model, her biri 64 nörondan oluşan iki katmanlı bir yapay sinir ağıdır. Eğitim sürecini dengelemek ve modelin optimal çözüme daha hızlı yakınsamasını sağlamak amacıyla her katman arasında Batch Normalization katmanları yerleştirilmiştir [34]. Modelin eğitimi, başlangıç öğrenme oranı 0.005 olarak belirlenen Adam optimizasyon algoritması ile gerçekleştirılmıştır.

Eğitim süreci, her biri 32 örnekten oluşan mini-batch'lerle, toplam 200 epoch boyunca yürütülmüştür. Eğitim sırasında modelin genelleme performansını değerlendirmek amacıyla verinin %10'u doğrulama verisi olarak ayrılmıştır. Öğrenme oranının durağan bir değerde takılı kalmasını önlemek ve daha hızlı yakınsama sağlamak için, ReduceLROnPlateau mekanizması kullanılmıştır. Bu mekanizma; 16 epoch boyunca gelişme gözlemlenmediğinde öğrenme oranını 0.5 katsayı ile azaltmakta ve minimum öğrenme oranı olarak 10^{-8} sınırı belirlenmiştir.

Modelin optimizasyonunda, sürekli değer regresyon problemlerine uygun olan Ortalama Kare Hata (Mean Squared Error - MSE) loss fonksiyonu kullanılmıştır.



Şekil 2.25. LSTM hücre yapısı

d.) Deep Deterministic Policy Gradient (DDPG) pekiştirmeli öğrenme

Lineer regresyon ve LSTM gibi makine öğrenimi tabanlı modeller, geleneksel PID denetleyiciler kullanılarak gerçekleştirilen uçuş demonstrasyonlarından toplanan veriler ile eğitilmektedir. Bu nedenle, bu modellerin ürettiği kontrol davranışları, çoğunlukla PID denetleyicinin davranışından önemli ölçüde farklılık göstermemektedir. Öte yandan, pekiştirmeli öğrenme (Reinforcement Learning - RL) yöntemleri farklı bir yaklaşım benimsemektedir. RL ajanı, çevresiyle doğrudan etkileşim kurarak; aldığı ödül ve cezalar aracılığıyla uygun kontrol stratejilerini öğrenmektedir.

Pekiştirmeli öğrenme algoritmaları; model tabanlı veya modelsiz oluşlarına, değer-fonksiyonu (value function) odaklı ya da politika (policy) odaklı olmalarına göre çeşitli kategorilere ayrılmaktadır. RL yönteminin seçiminde bir diğer önemli etken, çevrenin sunduğu ve ihtiyaç duyduğu veri türüdür. Anti-roll (yuvarlanma önleyici) sistem özelinde, gözlem uzayı sürekli değerlerden oluşmaktadır; örneğin roll açısı ve roll hızı sürekli verilerdir. Aynı şekilde, kontrol çıktıları da sürekli bir değer olan aileron PWM kontrol sinyalidir. Bu bağlamda, aktör-eleştirmen (actor-critic) temelli yöntemlerden biri olan Deep Deterministic Policy Gradient (DDPG) algoritması, bu problem için oldukça uygun bir çözüm olarak öne çıkmaktadır.

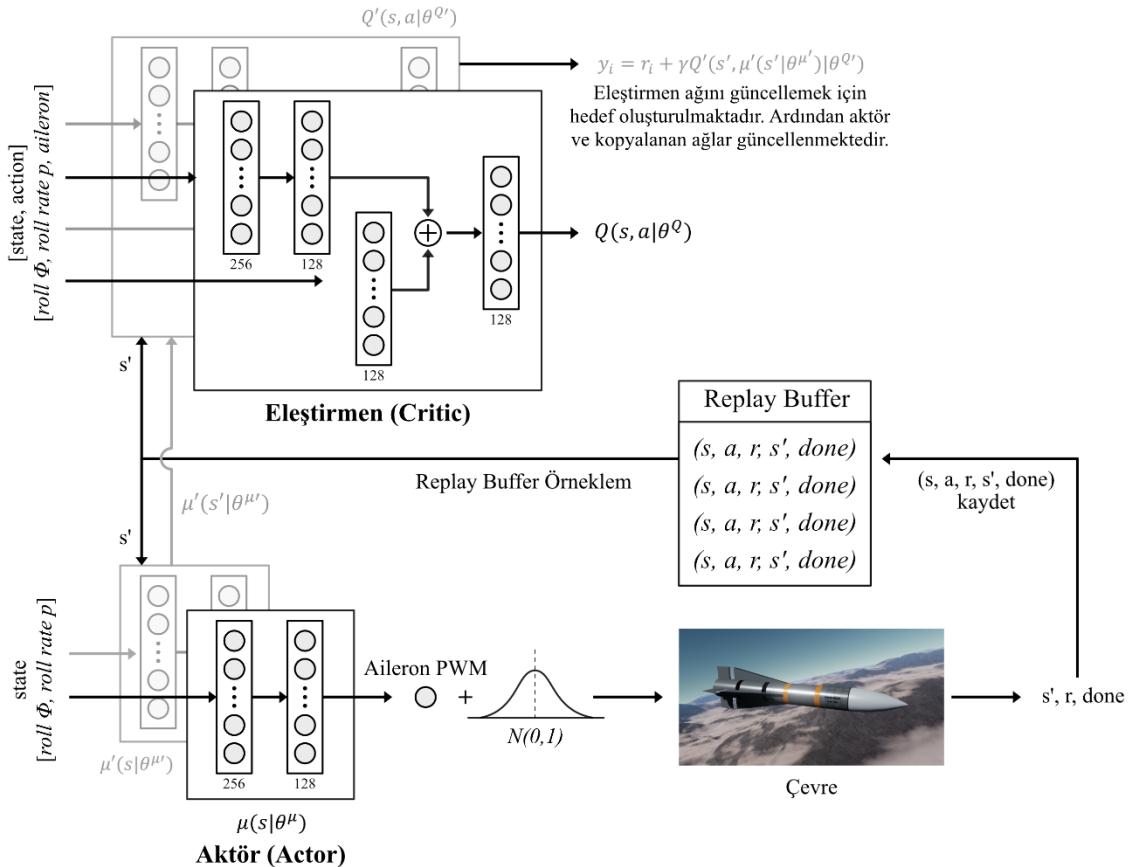
DDPG Mimarisi

Deep Deterministic Policy Gradient (DDPG), aktör-eleştirmen (actor-critic) mimarisi temelinde geliştirilmiş, model içermeyen (model-free) bir pekiştirmeli öğrenme yöntemidir. DDPG, eş zamanlı olarak hem değer fonksiyonunu hem de politika fonksiyonunu öğrenir. Bu yöntemde aktör, verilen durumu (state) en uygun eyleme (action) eşlemeye çalışırken, eleştirmen (critic) ise seçilen eylemlerin belirli durumlar için ne kadar iyi olduğunu öğrenir. DDPG, politika ve değer fonksiyonunu, deneyimlerin depolandığı replay buffer (deneyim belleği) kullanılarak çevrimdışı (offline) biçimde eğitir.

DDPG yöntemi ilk olarak [35]'nde tanıtılmış olup, günümüzde çeşitli uygulamalarda kullanılarak en son teknoloji performansları elde edilmiştir. Bu DDPG mimarisinde yer alan temel bileşenler şunlardır:

- İki katmandan oluşan (256, 128 düğüm) bir eleştirmen ağı $Q(s, a|\theta)$,
- Dört katmandan oluşan (256, 128, 128, 128 düğüm) bir aktör ağı $\mu(s|\theta)$. Eleştirmen ve aktör ağlarının detaylı mimarileri Şekil 2.26'de gösterilmiştir.
- Durum-geçiş çiftlerini (s, a, r, s') depolayan replay buffer,
- Replay buffer'dan mini-batch'ler halinde rastgele örneklemeye yapılarak çevrimdışı politika ve değer fonksiyonu öğrenimi. Örneklemenin ardışık olmaması, bellekten ilişkili olmayan veri örneklerinin çekilmesini sağlar,
- Eğitim sırasında yakınsama ve kararlılığı artırmak amacıyla aktör ve eleştirmen ağlarının hedef (target) tahmini için ağ kopyalarının kullanılması. Eleştirmen ağı $Q(s, a|\theta)$ güncellenirken aynı anda hedef hesaplamada kullanılır ve bu durum öğrenmede sapmaya yol açabilir. DDPG yönteminin geliştiricisi tarafından önerilen çözüm, ağların kopyalarının kullanılmasıdır. Kopyalanan ağların ağırlıkları, gerçek ağların ağırlıklarını izlemek için üstel ortalama yöntemiyle güncellenir ($[\theta_{target} \leftarrow \tau \cdot \theta + (1 - \tau) \cdot \theta_{target}]$, τ genellikle 0.005 gibi küçük bir değer kullanılmaktadır),
- Özelliklerin normalize edilmesi için batch normalization katmanlarının kullanılması; bu strateji aynı zamanda düzenlileştirici (regularizer) olarak işlev görür,
- Politika tarafından seçilen eylemlere gürültü eklenerek gerçekleştirilen politika dışı (off-policy) keşif stratejisi. $\mu(s') = \mu(s|\theta) + N$, Burada gürültüyü belirlemek için Ornstein-Uhlenbeck işlemi kullanılmıştır:

$$\mu(s') = \mu(s|\theta) + \omega (\mu_{given\ mean} - \mu(s|\theta)) + N(0, \sigma) \quad (2.7)$$



Şekil 2.26. DDPG model yapısı ve algoritması

Algoritma

DDPG aganının eğitim süreci, çevrimiçi (online) olarak gözlem alanı (state space, s), eylem (action, a), sonraki durum (next state, s') ve ödül (reward, r) bilgisini içeren deneyimlerin toplanması ile başlar ve bu deneyimlerin tuple hâlinde replay buffer (deneyim belleği) içine kaydedilmesiyle devam eder. DDPG, bir sonraki durum için açgözlü (greedy) eylemi tahmin etmeyi amaçlayan hedef deterministik politika fonksiyonunu kullanır. Bu yaklaşım, diğer pekiştirmeli öğrenme yöntemlerinde olduğu gibi değer fonksiyonu öğrenmek yerine doğrudan politika fonksiyonunun bulunması yoluyla gerçekleştirilir. Bu çalışmada algoritmanın detaylı uygulanışı ise Algoritma 3'te ifade edilmektedir.

Eleştirmen ağının güncellenmesinde kullanılan loss fonksiyonu, hedef eleştirmen ağının (target critic network) çıktısını temel alır. Hedef eleştirmen ağının, hedef değerin elde edilmesi amacıyla eylem ve yeni durumu (next state) girdi olarak alır. Bu durum, eleştirmen ağının girişlerinin [durum, eylem], aktör ağının girişlerinin ise yalnızca [durum] olduğunu göstermektedir. Ayrıca aktör ağının, kendi gradyan bilgisine ek olarak eleştirmen ağının gradyanından da

yararlanmaktadır. Bu çalışmada kullanılan DDPG yönteminin algoritması ve uygulanışı, [35] içerisinde belgelenen orijinal uygulamadan kısmen farklılık göstermektedir.

Hem doğrusal fonksiyon yaklaşımı hem de DDPG yöntemi için kullanılan ödül fonksiyonu aşağıda gösterilmiştir:

$$r_t = (5 - \Phi - p) - r_{t-1} \quad (2.13)$$

Burada ,

- r , ödüldür,
- Φ yuvarlanma açısı (roll),
- p yuvarlanma hızıdır (roll rate)

Önerilen ödül fonksiyonu, açık bir şekilde roll ve roll rate değerleri arttıkça cezaların da artacağını ifade etmektedir. Fonksiyonda yer alan sabit 5 değeri, her adımda (step) ajan için verilen pozitif ödülü temsil etmektedir ve bu sayede DDPG ajanının bir eğitim episodu boyunca öğrenme sürecini sürdürmesi teşvik edilmektedir. Bu çalışmada kullanılan ödül fonksiyonu, $r_t = f(s_t) - f(s_{t-1})$, formülüyle tanımlanmıştır. Bu yapı, yalnızca sonuca odaklanmak yerine ajanın hedefe yönelik ilerlemesini teşvik etmektedir. Ayrıca, bu yöntem aracılığıyla zamana bağlı fark (temporal difference) ΔR değeri elde edilerek ajanın davranışındaki iyileşme veya gerileme gözlemlenebilmektedir.

Anti-roll sistemine ait durum uzayı (state space), iki değeri içermektedir: füzenin yalpa (roll) açısı ve yalpa açısal hızı. Politika (policy) ise aileron kontrolü için tek bir vektör değeri üretmektedir; bu değer, PWM sinyalini temsil etmektedir. Bu çalışmada, eylem değeri $[-0.5, 0.5]$ aralığında sınırlanmıştır ve ajan tarafından aileron kontrol tuşuna kaç saniye basılması gerektiğini ifade etmektedir.

Algoritma 3. Deep deterministic policy gradient algoritması

1 Aktör ve eleştirmen (critic) ağları, Xavier normal başlatıcı kullanılarak başlatılmaktadır:

$$\sigma = \sqrt{\left(\frac{2}{n_{in} + n_{out}}\right)} \quad (2.8)$$

Burada σ , standart sapmayı; n_{in} ve n_{out} ise sırasıyla giriş ve çıkış düğüm sayılarını ifade eder. Aktör ağı $\mu(s|\theta^\mu)$ ile, eleştirmen ağı ise $Q(s, a|\theta^Q)$ ile gösterilir. θ^μ ve θ^Q , sırasıyla ilgili ağların ağırlıklarını temsil etmektedir.

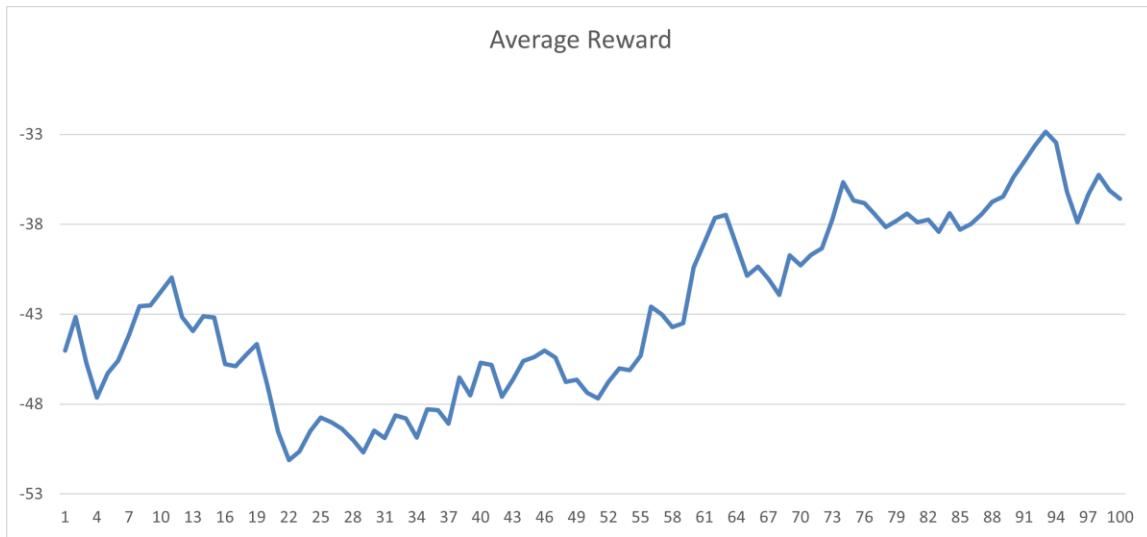
- 2 Hedef ağlar μ' ve Q' , ilgili aktör ve eleştirmen ağlarıyla aynı ağırlıklarla başlatılmaktadır.
 - 3 Replay Buffer R , ortamı 20 bölüm süresince simüle ederek deneyim çiftleri (s, a, r, s') toplamak amacıyla ön hazırlık aşamasıyla başlatılmaktadır.
 - 4 Her bölüm (episode) n için:
 - 5 state = simülasyonu sıfırla
 - 6 s = normalize(state)
 - 7 Her adım için (maksimum duruma ulaşılana kadar veya füze yok edilene kadar):
 - 8 Rastgele gürültü oluştur;
$$N = \begin{cases} \text{Random noise, with probability } \varepsilon \\ 0, \quad \text{with probability } 1 - \varepsilon \end{cases}$$
 - 9 Eylem seç; $a = \mu(s|\theta^\mu) + N$
 - 10 Eylemi gerçekleştir, ödül r ve yeni durum s' elde et
 - 11 Deneyim çiftini $(s, a, r, s')R$ içine kaydet
 - 12 R 'den rastgele bir minibatch deneyim çifti B örnekle (yeni girdilerin %80 olasılıkla örnekleşmesiyle)
 - 13 Eleştirmen ağını, kaybı en aza indirgerek güncelle:
$$y_i = r_i + \gamma Q'(s', \mu'(s'|\theta^{\mu'})|\theta^{Q'}) \quad (2.9)$$
$$L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2 \quad (2.10)$$
 - 14 Aktör ağını güncelle:
$$\nabla_{\theta^\mu} \mu \approx \frac{1}{N} \sum_i [\nabla_a Q(s_i, \mu(s_i)|\theta^Q) \nabla_{\theta^\mu} \mu(s_i|\theta^\mu)] \quad (2.11)$$
 - 15 Hedef ağları güncelle:
$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'} \quad (2.12)$$
$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$
 - 16 Eğer işlem tamamlandıysa:
İç döngüyü sonlandır; bir sonraki bölüme geç.
-

Farklı DDPG yapılandırmaları denenerek, çeşitli kontrol stratejilerinin performansı değerlendirilmiştir. Bu yapılandırmalar kapsamında; ajan girdisi olarak kullanılan durum (state) sayıları, hangi durumların dahil edildiği (örneğin yalnızca roll ve roll rate ya da bunlara ek olarak pitch, pitch rate, yaw, yaw rate ve velocity gibi diğer uçuş dinamikleri), sinir ağı üzerindeki düğüm (node) sayısı ve kullanılan aktivasyon fonksiyonları (leaky ReLU, tanh vb.) gibi parametrelerde farklı kombinasyonlar test edilmiştir. Ayrıca, çeşitli ödüllü fonksiyonları da formüle edilerek DDPG ajanının eğitimi sürecinde uygulanmıştır.

En iyi sonuç veren yapılandırmaya ilişkin detaylar önceki bölümlerde sunulmuştur. Eğitim süreci toplam 120 bölümden oluşmuş olup, bunun ilk 20 bölümü “ısınma” (warm-up) evresi olarak gerçekleştirılmıştır. Kullanılan hiperparametreler Tablo 2.’te verilmiştir. Şekil 2.27, füze için roll stabilizasyonu görevinde eğitilen DDPG ajanının ortalama ödül değerlerinin bölüm bazında değişimini göstermektedir. Bu grafik, ajanın zaman içinde öğrenme performansını ve kararlılık düzeyini değerlendirmek açısından önemli bir göstergedir.

Tablo 2.3. DDPG eğitiminde kullanılan hiperparametreler

Hyperparameter	Değer	Açıklama
Toplam episode	100	
Max step	1000	Bir episode için maksimum adım sayısı
Critic LR	0.001	Learning rate eleştirmen
Actor LR	0.0025	Learning rate aktör
Batch size	128	
Replay Buffer boyutu	10^6	
Gamma γ	0.99	Temporal Difference için kullanılmaktadır
Tau τ	0.001	Hedef ağları güncellemek için kullanılmaktadır
Sampling Constant P	0.8	Replay Bufferdaki daha yeni girişler önceliklidir



Şekil 2.27. DDPG model eğitimin ortalama ödül

3. MODEL EĞİTİMİ VE SİMÜLASYON SONUÇLARI

Bu bölümde geliştirilen güdüm ve kontrol sistemlerine ait eğitim süreçleri ile gerçekleştirilen simülasyonların sonuçları sunulmuştur. Farklı senaryolar altında sistemin başarımı test edilmiştir.

3.1 Füze Sisteminin Test Simülasyonları

Bu alt bölümde, geliştirilen füze kontrol sisteminin çeşitli test senaryolarındaki performansı değerlendirilmektedir. Özellikle terminal aşamada hedefe yönelik ve sistem stabilitesi üzerinde durulmuştur.

3.1.1 Terminal Yaw ve Pitch Açısı Test Simülasyonu

Bu simülasyonda, pitch (elevator) ve yaw (rudder) kontrol sistemlerinin etkinliği, füzenin hedefe yönlendirilmesiyle test edilmiştir. Önceki bölümde açıklanan yol noktası (waypoint) hesaplama ve belirleme algoritması kullanılarak simülasyon aşağıdaki adımlarla gerçekleştirilmiştir:

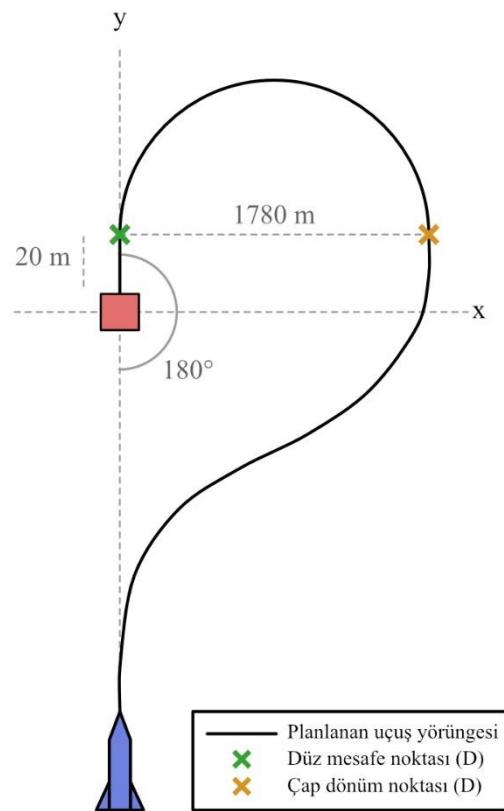
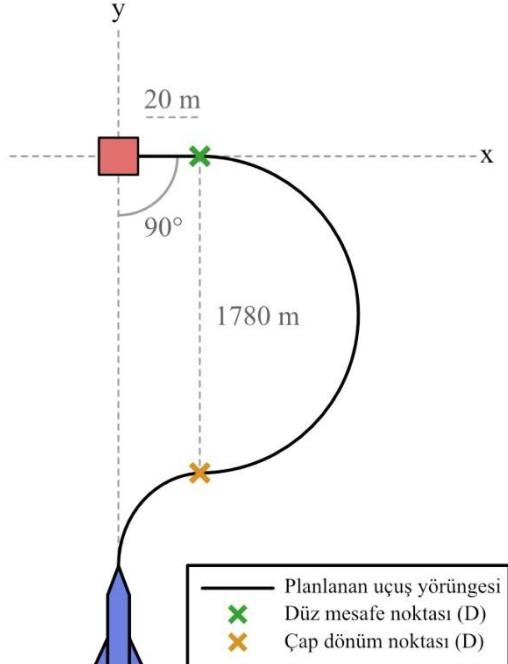
90° ve 180° terminal yaw açıları için:

1. Veri alıcı iş parçacığı başlatılır.
2. Anti-roll kontrol iş parçacığı başlatılır.
3. Elevator kontrolörü iş parçacığı başlatılır.
4. Otopilot görev kontrol iş parçacığı başlatılır.
5. Füze yere çarpıp imha olana kadar simülasyona devam edilir.

90° ve 180° terminal yaw açıları için hedeflenen iz (trajektori), füze perspektifinden belirli yol noktalarının hesaplanmasıyla oluşturulmuştur. Trajektori planlaması yapılabilmesi için bilinmesi gereken temel noktalar düz mesafe noktası (D) ve çap dönüş noktası (R). Bu iki farklı terminal yaw açısı için belirlenen yol noktalarının görselleştirmesi Şekil 3.1 ve Şekil 3.2'de sunulmuştur. Şekillerde de görüldüğü üzere, söz konusu yol noktaları simülasyonun başlangıcında, önceki bölümde ayrıntılı biçimde açıklanan Algoritma 1 ve 2 kullanılarak hesaplanmaktadır. Düz mesafe noktası, füzenin hedefe olan mesafesinin 20 metreye düşüğü ve bu noktadan itibaren pitch açısını hedefe yönelik üzere ayarladığı konumdur. Her ne kadar belirlenen bu mesafe teorik olarak 20 metre olsa da, gerçekleştirilen deneysel çalışmalar, füzenin genellikle bu mesafeye ulaşmadan önce istenen yaw açısına göre hedefe yönelikmeye başladığını göstermektedir.

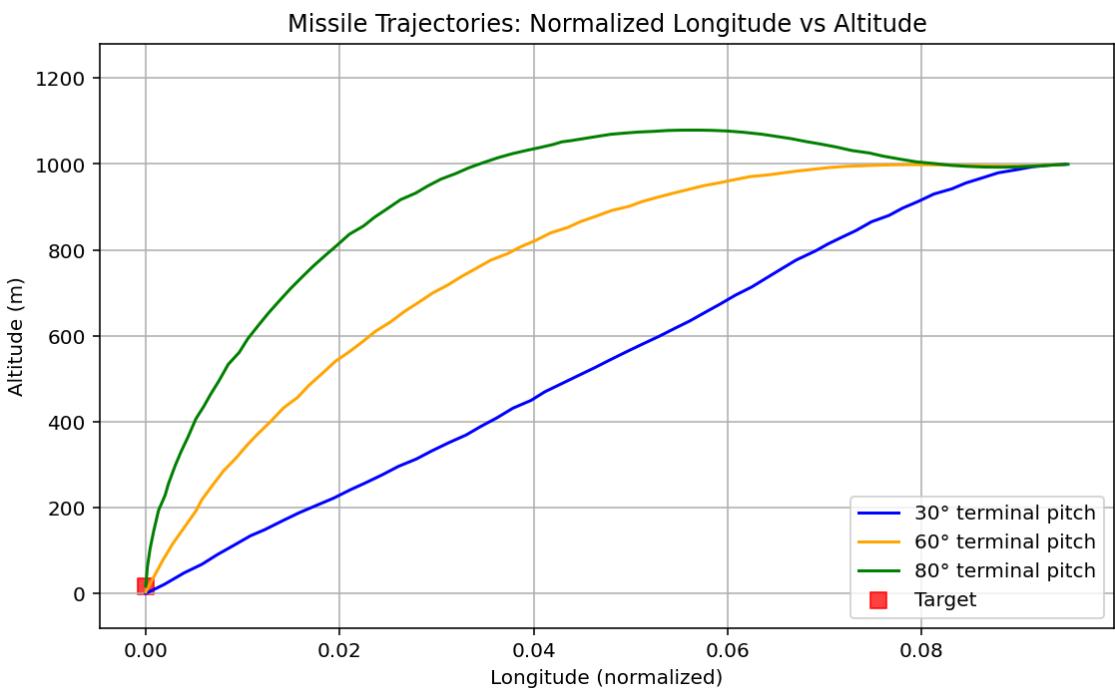
Diger yandan, çap dönüş noktası, füzenin yönünü ters istikamete çevirmek amacıyla tam dümen (full rudder) komutunun uygulanmaya başlanması gereken noktadır. Yapılan deneyler

neticesinde, füzenin 180° 'lik bir dönüş manevrasını başarıyla gerçekleştirebilmesi için ihtiyaç duyduğu ortalama dönüş çapı, yaklaşık 0.08° boylam farkına karşılık gelmekte olup, bu da yaklaşık 1780 metreye tekabül etmektedir. Bu veriler, füze manevra kabiliyetinin ve yönlendirme algoritmasının fiziksel uzaydaki karşılıklarını daha doğru modellemek açısından önemli parametreler sunmaktadır.



Simülasyon sonuçlarına ilişkin veriler ise Ekler bölümünde yer alan Şekil 5.1 ve Şekil 5.2'de sırasıyla 90° ve 180° terminal yaw açıları için daha ayrıntılı bir şekilde gösterilmiştir. Gerçekleştirilen simülasyon deneylerinde, kullanılan ortamın ideal olmaması ve uygulamalar arası iletişimdeki gecikmeler nedeniyle, uçuş verilerinde (örneğin enlem ve boylam koordinatları) ve kontrol girişlerinin uygulanmasında sapmalar meydana gelmiştir. Bu durum, füzenin izlediği yolun belirlenen ideal trajektoriden kısmen sapmasına neden olmuştur. Ancak, simülasyon sonuçları, hedefe yönelik istenen terminal yaw açısına göre belirlenen yol noktası algoritmasının son derece etkili olduğunu ve hedefe başarılı bir şekilde nüfuz edebildiğini göstermiştir.

Terminal pitch kontrol algoritmasının değerlendirilmesi ise farklı istenen terminal pitch açıları (30° , 60° ve 80°) kullanılarak saldırı testleri simüle edilerek gerçekleştirilmiştir. Şekil 3.3, bu deneylerin genel seyrini ve farklı terminal pitch açıları için elde edilen sonuçları görsel olarak sunmaktadır. Simülasyonlara ilişkin daha ayrıntılı grafiksel veriler ise Ekler bölümünde yer alan Şekil 5.3–5.5'te gösterilmiştir. Ayrıca, Tablo 3.1'de, füzenin hedefe çarpmadan önceki son iki saniyesine ait pitch açıları yer almaktır, bu veriler 5 Hz örnekleme frekansına göre toplamda 10 veri noktasını içermektedir.



Şekil 3.3. 30° , 60° ve 80° terminal pitch için boylam ve irtifa grafiği

Tablo 3.1. Hedefe ulaşmadan önceki füze pitch açı değerlerinin son 10 örneği

Adım	İstenilen terminal pitch açısı (mutlak açı)		
	30°	60°	80°
1	30.44	54.04	66.06
2	27.23	51.86	72.44
3	27.67	58.28	76.19
4	22.40	55.62	69.923
5	32.93	59.09	75.12
6	30.77	60.61	81.98
7	24.11	58.98	85.59
8	28.24	67.01	82.74
9	22.00	61.83	86.67
10	22.38	58.00	85.04

Tablo 3.1'de sunulan veriler, füzenin hedefe yaklaşırken ulaştığı pitch açıları ile hedeflenen terminal pitch açıları arasında oldukça düşük hata payı olduğunu göstermektedir. Bu durum, kullanılan PN (Proportional Navigation) algoritmasının etkili bir şekilde çalıştığını ve füzenin yönlendirilmesinde başarılı bir performans sergilediğini ortaya koymaktadır.

3.1.2 YOLOv8 Modeli ile Pitch Açısı Kontrolü

Model YOLOv8 modeli, füzenin hedefe yaklaşma aşamasında elevator kontrolünü desteklemek amacıyla kullanılmaktadır. Simülasyon, hedefin kamera görüş alanına (line of sight) girmesi durumunda hedefin tespit edilmesi senaryosu üzerinden gerçekleştirılmıştır. Bu çalışmada, görsel verilerin daha hızlı işlenebilmesi amacıyla parametre sayısı daha az olan YOLOv8n modeli tercih edilmiştir. Yapılan deneysel çalışmalar sonucunda, füze ile hedef arasındaki mesafe 650 metreye indiğinde, modelin hedefi algılayabildiği gözlemlenmiştir.

YOLO modeli, hedefi giriş görüntüsünde tespit ettikten sonra ilgili bounding box (sınır kutusu) oluşturulmakta ve bu kutu, elevator için PWM (Pulse Width Modulation) değeri tahmininde kullanılmaktadır. PWM sinyalini tahmin etmek amacıyla general function approximator olarak yapay sinir ağı (YSA) modeli kullanılmıştır.

YOLOv8n tabanlı görsel pitch kontrol sistemi performansının dayanıklılığını (robustness) gösterebilmek amacıyla, simülasyon ortamında bir hata durumu (fault) senaryosu oluşturulmuştur. Bu senaryoda, gyroscope sensöründen elde edilen pitch değeri ile GPS verisi üzerinden hesaplanan hedef konumu bilgisine rassal gürültüler eklenmiştir. Gürültü ekleme işlemi, sistemdeki sensör verilerinin bozulması sonucu saldırı etkinliğinin azaldığı durumları modellemeyi amaçlamaktadır.

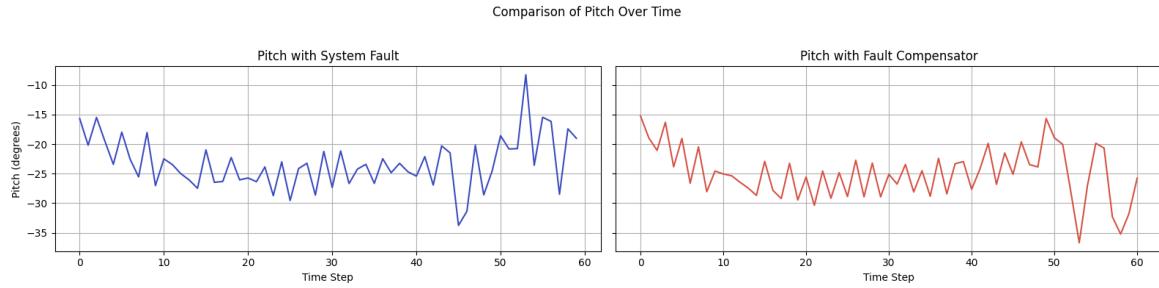
Hata simülasyonu aşağıdaki algoritmaya göre gerçekleştirilmiştir:

Eğer hedefe olan mesafe 650 metre veya daha az ise:

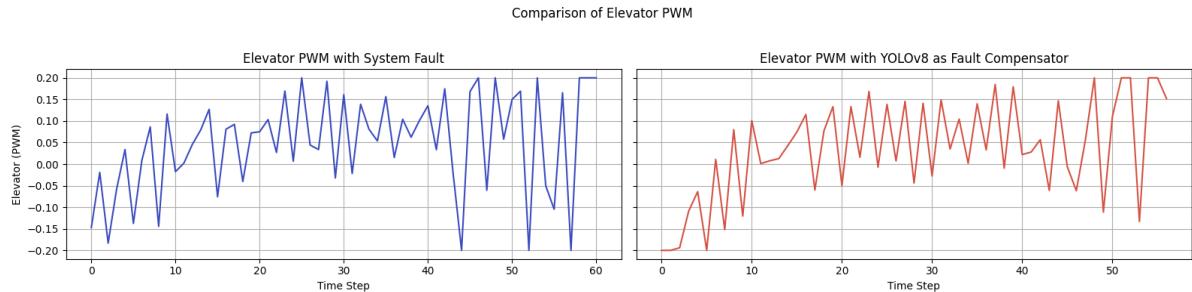
- %50 olasılıkla pitch değerine $[-5, 5]$ aralığında rastgele bir gürültü eklenir.
- %50 olasılıkla pitch rate değerine $[-2, 2]$ aralığında rastgele bir gürültü eklenir.
- %50 olasılıkla LOS (line-of-sight) açısına $[-5, 5]$ aralığında rastgele bir gürültü eklenir.

Deneysel sonuçlar, pitch kontrol sisteminin sensör arızası (fault) durumlarında saldırı etkinliğini önemli ölçüde kaybettiğini, hatta füzenin hedeften sapmasına neden olduğunu ortaya koymustur. Buna karşın, YOLOv8 ile entegre edilmiş sistemin, diğer sensörlerden kaynaklanan arızalarda kontrol girişini düzeltici etkide bulunduğu ve kontrol performansını artırdığı gözlemlenmiştir. Simülasyon sonuçlarına göre, YOLOv8 kullanılmayan senaryoda füze hedeften 6.20 metre saparken, YOLOv8'in kullandığı durumda hedefe yalnızca 1.07 metre sapma ile, neredeyse isabet derecesinde yaklaşmıştır.

Şekil 3.4 ve Şekil 3.5, sensör arızalarının olduğu bir ortamda YOLOv8 modelinin fault kompansatörü (bozukluğu telafi edici) olarak kullanıldığı ve kullanılmadığı durumlara ait simülasyon sonuçlarını göstermektedir.



Şekil 3.4. Pitch açılarının karşılaştırılması



Şekil 3.5. Elevator girdi karşılaştırılması

Grafikler incelendiğinde, yaklaşık olarak 50. adımında pitch verisinin sistemsel arıza nedeniyle bozulmaya başladığı gözlemlenmektedir. Bu durum, GPS ve gyroscope verilerinde hata olması durumunda, YOLOv8 tabanlı yedek sistemlerin füze isabet oranını artırabileceğini göstermektedir. Bu tür sistemsel yedekliliklerin, belirsizlik ve bozucu etkenlerin yoğun olduğu saldırı senaryolarında kritik öneme sahip olduğu sonucuna varılmıştır.

3.1.3 Füze Roll Stabilizasyon Sistemi Testi

Bu bölümde, roll stabilizasyonu amacıyla tasarlanmış tüm kontrolörler test edilip tartışılacaktır. Toplamda dört farklı kontrolör geliştirilmiştir: PID kontrolörü, Lineer Regresyon tabanlı model, LSTM tabanlı model ve DDPG yöntemi ile geliştirilen Takviyeli Öğrenme (Reinforcement Learning) modeli. Buna ek olarak, bu dört kontrolörün birleştirilmesiyle oluşturulan Ensemble yöntemi de değerlendirilmeye dahil edilmiştir.

Geliştirilen kontrolörlerin etkinliğini ölçmek amacıyla çeşitli analizler gerçekleştirilmiştir. İlk olarak, her bir kontrolörün roll stabilizasyon performansını değerlendirmek için füzenin irtifasını sabit tutarak düz uçuş simülasyonları gerçekleştirilmiştir. Bu simülasyonlarda

Tablo 3.2. Simülasyon 1: Roll Açısı Hata Değerleri Karşılaştırması

Controller	MAE (°)	RMSE (°)	MaxAE (°)
PID	8.48	10.97	37.19
Linear Regression	8.18	10.16	35.97
LSTM tabanlı	9.95	11.38	38.20
DDPG	9.42	13.28	39.36
Ensemble yöntemi	6.76	8.73	28.54

başlangıç koşulları; füze hızı 200 m/s (yaklaşık 0,57 Mach) ve başlangıç roll açısı sapması 30 derece olacak şekilde belirlenmiştir.

Değerlendirme metrikleri olarak ise şu kriterler kullanılmıştır:

- Mean Absolute Error, $MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$
- Root Mean Square Error, $RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$
- Max Absolute Error = $\max_{i=1 \rightarrow n} |y_i - \hat{y}_i|$

İlgili değerlendirme metriklerinin sonuçları Tablo 3.2'de sunulmuştur. Şekil 3.6 ise, her bir kontrolörün değerlendirme sırasında elde edilen roll ve aileron değişimlerini grafiksel olarak göstermektedir.

Birinci senaryo kapsamında gerçekleştirilen simülasyon testlerinin sonuçlarına göre, Ensemble yöntemi, MAE, RMSE ve MaxAE değerleri açısından en düşük hata oranlarını göstererek diğer yöntemlere kıyasla en başarılı performansı sergilemiştir. PID ve Lineer Regresyon tabanlı kontrolörlerin oldukça benzer performanslar gösterdiği gözlemlenmiştir; bu durum, Lineer Regresyon yönteminin eğitildiği veri kümesindeki davranışını başarılı bir şekilde taklit ettiğini ortaya koymaktadır. LSTM yöntemi en yüksek MAE değerini almış, bu da söz konusu yöntemin PID ve LR'ye kıyasla henüz yeterince kararlı olmadığını göstermektedir. DDPG yöntemi ise en yüksek RMSE ve MaxAE değerlerini elde ederek bu senaryoda en zayıf performansı göstermiştir. Ancak, ilerleyen bölümlerde detaylandırılacağı üzere, DDPG yöntemi saldırısı senaryosunda daha başarılı bir performans sergilemiştir.

Simülasyon Senaryosu 2 ve 3 kapsamında, roll stabilizasyon sistemi, füzenin belirli bir noktadan hedefe yöneldiği ve istenen terminal yaw açısının 90° ve 180° olduğu saldırısı görevleriyle değerlendirilmiştir. Bu değerlendirme, kontrolörlerin gerçek görev koşullarındaki etkinliğini test etmeyi amaçlamaktadır.

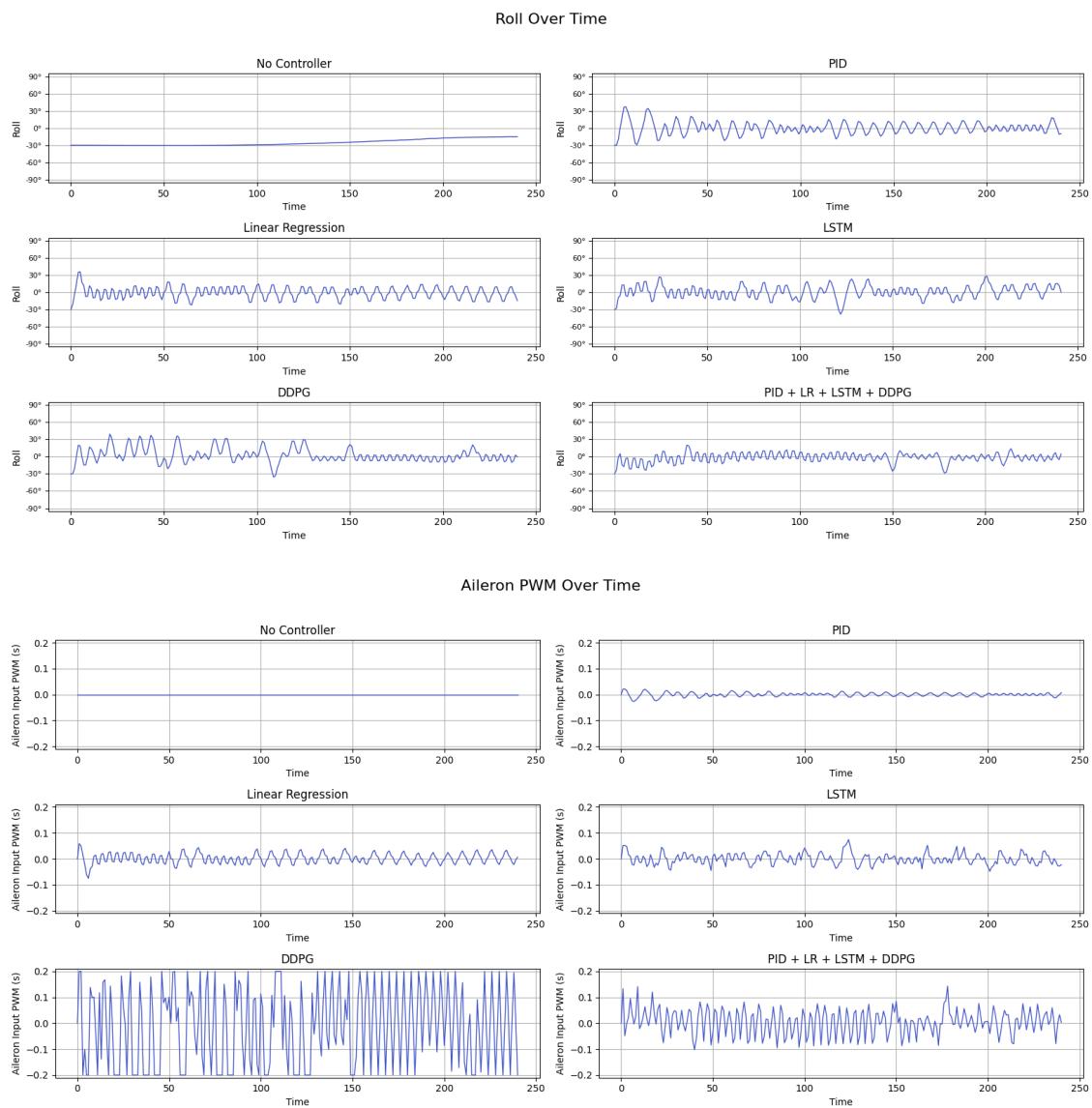
Simülasyonda kullanılan başlangıç koşulları şu şekildedir:

- Hız: 215 m/s (yaklaşık 0,63 Mach)
- Pitch açısı: 10°
- Hedefe doğru yönlenmiş ve hedefe olan mesafe: 2,7 km

90° ve 180° terminal yaw açılarına sahip saldırısı görevlerini temsil eden Simülasyon 2 ve Simülasyon 3'ün sonuçları sırasıyla Tablo 3.3 ve Tablo 3.4'te sunulmaktadır. Simülasyon 2'de (90° terminal yaw), DDPG ve Ensemble yaklaşımları, diğer yöntemlere kıyasla üstün bir performans sergilemiştir. Ensemble yöntemi, en düşük MAE (1.50°), RMSE (4.63°) ve hedef sapma değeri (0.14 m) ile en iyi sonuçları elde etmiş, bu da yüksek düzeyde kararlı roll kontrolü ve hassas hedef vurma yeteneğine işaret etmektedir. DDPG metodu ise benzer şekilde düşük MAE (1.57°) ve RMSE (5.02°) değerleri ile ve yalnızca 1.35 m'lik hedef sapmasıyla bu başarıyı yakından takip etmiş, böylece orta düzeyde manevra gerektiren senaryolarda etkinliğini doğrulamıştır.

Simülasyon 3'te terminal yaw açısının 180° 'ye çıkarılmasıyla birlikte manevra karmaşıklığı da artmıştır. Buna rağmen Ensemble yöntemi, en düşük hedef sapma değeri (0.14 m) ve düşük hata metriklerini koruyarak diğer tüm yöntemleri geride bırakmış ve daha agresif koşullarda güçlü genelleme yetenekleri sergilemiştir. DDPG denetleyicisi de 0.89 m'lik hedef sapması ve makul düzeyde hata değerleri ile yüksek etkinliğini korumuş, karmaşık senaryolarda dayanıklılığını göstermiştir. Daha basit bir yapıya sahip olan PID kontrolörü, 1.35 m'lik hedef sapmasıyla hedefe oldukça yakın bir performans sergileyerek yapılandırılmış ortamlarda güvenilirliğini sürdürdüğünü ortaya koymuştur.

Buna karşın, Lineer Regresyon ve LSTM tabanlı kontrolörler her iki senaryoda da düşük bir performans göstermiştir. Özellikle Simülasyon 3'te, Lineer Regresyon yöntemi son derece yüksek hata metrikleri (MAE: 35.68° , RMSE: 38.74° , MaxAE: 63.99°) ile birlikte 10.15 m gibi ciddi bir hedef sapma değeri göstermiştir. LSTM tabanlı denetleyici de 7.23 m'lik hedef sapması ile etkili bir roll kararlılığı sağlayamamıştır. Bu sonuçlar, dinamik ve doğrusal olmayan füze kontrol ortamlarında tekli model temelli öğrenme yaklaşımlarının sınırlamalarını ortaya koymakta; buna karşın derin pekiştirmeli öğrenme ve Ensemble stratejilerinin hassas yönlendirme ve roll stabilizasyonu sağlama konusundaki avantajlarını vurgulamaktadır. Simülasyon Senaryo 2 (90° terminal yaw) ve Senaryo 3 (180° terminal yaw) için elde edilen



Şekil 3.6. Roll açılarının ve aileron girdi karşılaştırılması

trajektori sonuçları ile roll açısı ve aileron girişlerine ilişkin grafikler Ekler bölümünde Şekil 5.6-5.11’te sunulmuştur.

Tablo 3.3. Simülasyon 2: Roll Kontrolörleri için Hata Metrikleri ve Hedef Sapma Karşılaştırması

Controller	MAE (°)	RMSE (°)	MaxAE (°)	Target Miss (m)
PID	1.75	5.39	14.84	5.89
Linear Regression	8.15	20.68	35.04	35.09
LSTM tabanlı	8.06	20.41	33.94	15.23
DDPG	1.57	5.02	16.51	1.35
Ensemble yöntemi	1.50	4.63	15.70	0.14

Tablo 3.4. Simülasyon 3: Roll Kontrolörleri için Hata Metrikleri ve Hedef Sapma Karşılaştırması

Controller	MAE (°)	RMSE (°)	MaxAE (°)	Target Miss (m)
PID	2.68	3.55	12.03	1.35
Linear Regression	35.68	38.74	63.99	10.15
LSTM tabanlı	31.90	34.78	55.54	7.23
DDPG	3.99	5.41	20.45	0.89
Ensemble yöntemi	3.00	3.90	14.43	0.14

4. BULGULAR VE TARTIŞMA

İHA simülasyonları kapsamında gerçekleştirilen çalışmalar, sistemin 2 boyutlu grafik modelleme yeteneğiyle, otonom çoklu İHA sistemlerinin karmaşık görev profillerini başarıyla icra etme ve dinamik çevresel kısıtlamalara adaptif olarak yanıt verme kabiliyetini güçlü bir biçimde ortaya koymaktadır. Bu modelleme, İHA'ların "Lider Referanslı Sanal Yapı" prensibine dayalı formasyonlarda nasıl organize olduğunu, kullanıcı tarafından seçilebilen "Lider Direkt Saldırı" veya "Kanat Hareketi ve Yakınsama Stratejisi" gibi çeşitli görev hedeflerine nasıl ulaştığını sergilemektedir. Özellikle, uçuşa yasak bölgelerden (NFZ) kaçınma stratejileri, basit reaktif davranışın ötesine geçerek, rota kesişim tahmini ve dinamik ara nokta oluşturma gibi proaktif yöntemlerle zenginleştirilmiştir. Ayrıca, Kara Tabanlı Hava Savunma (GBAD) sisteminin dinamik bir tehdit olarak simülasyona dahil edilmesi ve bu tehdide karşı otonom tepkiler geliştirilmesi (örn. İHA'ların hedef alınması ve görevden düşmesi), sistemin adaptasyon yeteneğinin önemli bir göstergesidir. Elde edilen sonuçlar, formasyon koruma için kullanılan temel orantısal kontrol algoritmalarının etkinliğini, görev yönetimi için uygulanan dinamik durum makinesinin esnekliğini ve lider arızası durumunda belirlenmiş kriterlere göre gerçekleştirilen otonom lider yeniden seçimi mekanizmasının görev sürekliliğini sağlama potansiyelini ayrıca göstermektedir.

Füze kontrol ve navigasyon sistemi geliştirme sürecinde çeşitli stratejiler ve algoritmalar tasarlanmış ve uygulanmıştır. Çalışmalar, terminal pitch açısını kontrol etmek amacıyla Python ortamında Proportional Navigation algoritmasının formüle edilip geliştirilmesi ile başlamıştır. Geliştirilen algoritma, SR2 simülatöründe test edilmiş ve beklenen sonuçları vermiştir.

Buna ek olarak, hava savunma sistemini hedef olarak tanıabilen bir görsel algılama altyapısı oluşturmak amacıyla, YOLOv8 modeli kullanılmıştır. Kamera sensörlerinden alınan görsel verilerle entegre çalışan bu yapı, füzenin hedefe doğru pitch açısını dinamik olarak ayarlamasına olanak tanımıştır. GPS ve jiroskop gibi sensör verilerinin güvenilirliğini kaybettiği durumlarda, bu görsel tabanlı sistem bir hata telafi mekanizması (fault compensator) olarak devreye girmektedir. Gerçekleştirilen simülasyonlar, GPS sinyali bozulduğunda füzenin hedefi vurmadığını göstermiştir. Ancak YOLOv8 tabanlı hata telafi sistemi devreye alındığında, füze hedefi küçük bir sapmayla başarılı bir şekilde vurabilmiştir.

Ayrıca, uçuş rotası planlama ve terminal yaw açısını belirlemeye yönelik stratejiler de geliştirilmiştir. Farklı yaw açıları (90° ve 180°) doğrultusunda hedefi vuracak uçuş rotasını belirleyebilen bir algoritma tasarlanmış ve test edilmiştir. Füzenin istenilen ara noktalara yönlendirilmesi için Bézier eğrisi formülü kullanılarak, daha düzgün ve optimal bir uçuş rotası

elde edilmesi sağlanmıştır. Simülasyon sonuçlarını gösteren video bağlantısı Ekler bölümünde sunulmuştur.

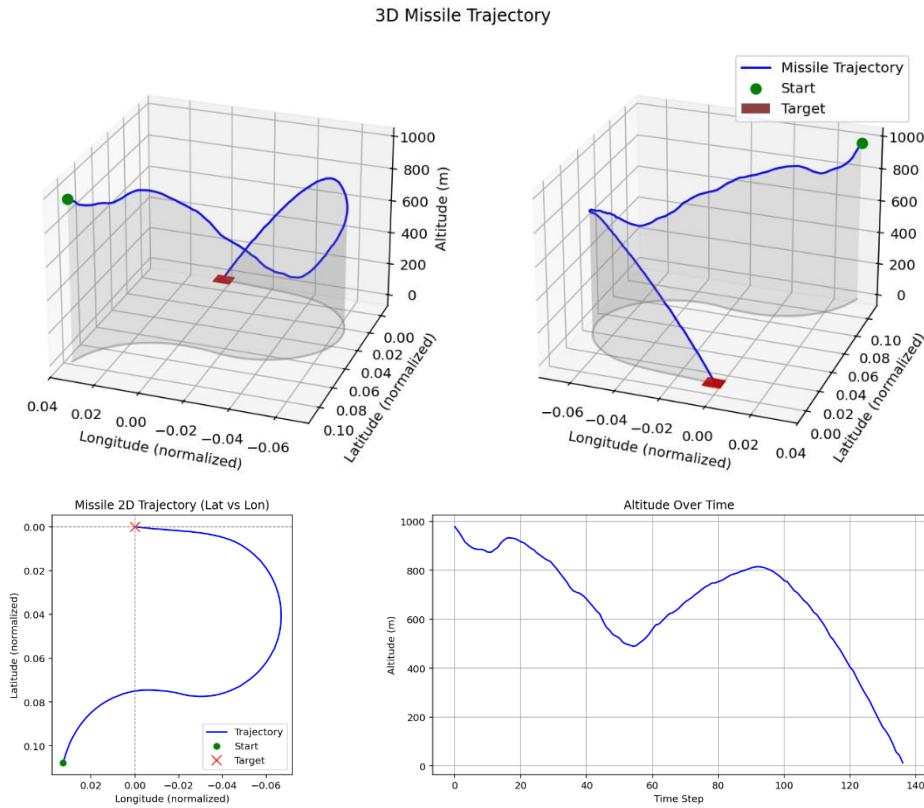
Bunun ardından, Anti-Roll Stabilizasyon Sisteminin geliştirilmesi kapsamında çeşitli kontrolörler tasarlanmıştır. İlk olarak, geleneksel PID kontrolörü formüle edilmiş ve her bir değişkenin sabit katsayıları belirlenmiştir. Daha sonra, PID kontrolörü kullanılarak simülasyonlardan elde edilen verilerle yapay zeka (YZ) tabanlı kontrolörler geliştirilmiştir. Bu kapsamda, simülasyon verileri ile eğitilen basit bir doğrusal regresyon modeli oluşturulmuştur. Aynı veri seti ile eğitilen LSTM tabanlı bir model de geliştirilmiş ve oldukça dayanıklı bir LSTM modeli elde edilmiştir. Son olarak, anti-roll sistemini geliştirmek için pekiştirmeli öğrenme (RL) algoritmalarından biri olan Deep Deterministic Policy Gradient (DDPG) yöntemi uygulanmıştır. Bu yöntemde, RL ajansı SR2 simülatöründe eğitilerek çevreyi keşfetmiş ve kendisine verilen kontrol probleminin en uygun politikasını öğrenmiştir. Elde edilen tüm anti-roll sistemine ait kontrol modelleri analiz edilmiş ve karşılaştırılmıştır. Tüm kontrolörlerin test edildiği simülasyon sonuçlarına göre, DDPG ve Ensemble (tüm kontrolörlerin birleşiminden oluşan yapı) tabanlı kontrolörler, füzenin roll stabilitesini başarılı bir şekilde korumuş ve hedefi isabetli biçimde vurmuştur. Buna karşın, bazı diğer kontrolörler roll açısından daha yüksek hata değerleri üretmiş ve bu durum füzenin hedefi ıskalamasına neden olmuştur. Bununla birlikte, DDPG kontrolörünün hedefi başarıyla vurduğu gözlemлense de, ürettiği aileron kontrol sinyallerinin büyülüгü, diğer yöntemlere kıyasla daha yüksektir. Bu durum, DDPG'nin daha agresif manevralar gerçekleştirdiğini ve dolayısıyla daha yüksek kontrol eforu gerektirdiğini göstermektedir.

Bu çalışma, modüler bir yaklaşımla geliştirilmiş olup, hem İHA (İnsansız Hava Aracı) hem de füze bileşenleri için çeşitli alt modüller tasarlanmıştır. Ancak geliştirilen modüllerin sayıca fazla ve yapısal olarak bağımsız olması, simülasyonların tümleşik bir sistemde birleştirilmesini güçlendirmiştir. Zaman kısıtları nedeniyle, hava savunma sisteme yönelik saldırı senaryosunu içeren tüm modüllerin tek bir simülasyon ortamında entegrasyonu gerçekleştirilememiştir. Ayrıca, kullanılan Simplerockets 2 simülasyon uygulaması, temelde roket uçuşları için tasarlandığından, çoklu İHA senaryolarının modellenmesi açısından yeterince uygun ve verimli değildir.

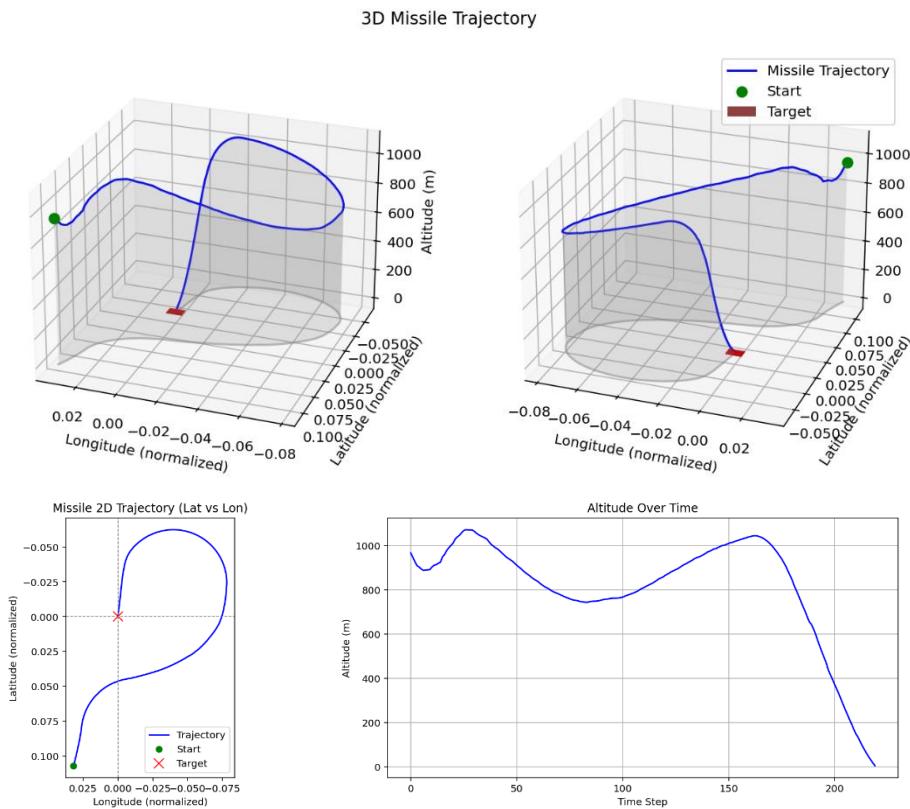
İleriye dönük çalışmalarında, bu çalışmada geliştirilen yöntemlerin doğrulamasını daha sahaklı yapabilmek için daha gelişmiş ve esnek bir simülasyon ortamının kullanılması ya da özel olarak geliştirilmesi önerilmektedir. Buna ek olarak, saha uygulamasına geçilebilmesi için gerçek

zamanlı donanım testleri, uçuş denemeleri ve ilgili alt sistemlerin fiziksel entegrasyonu gibi adımların planlanması gerekmektedir.

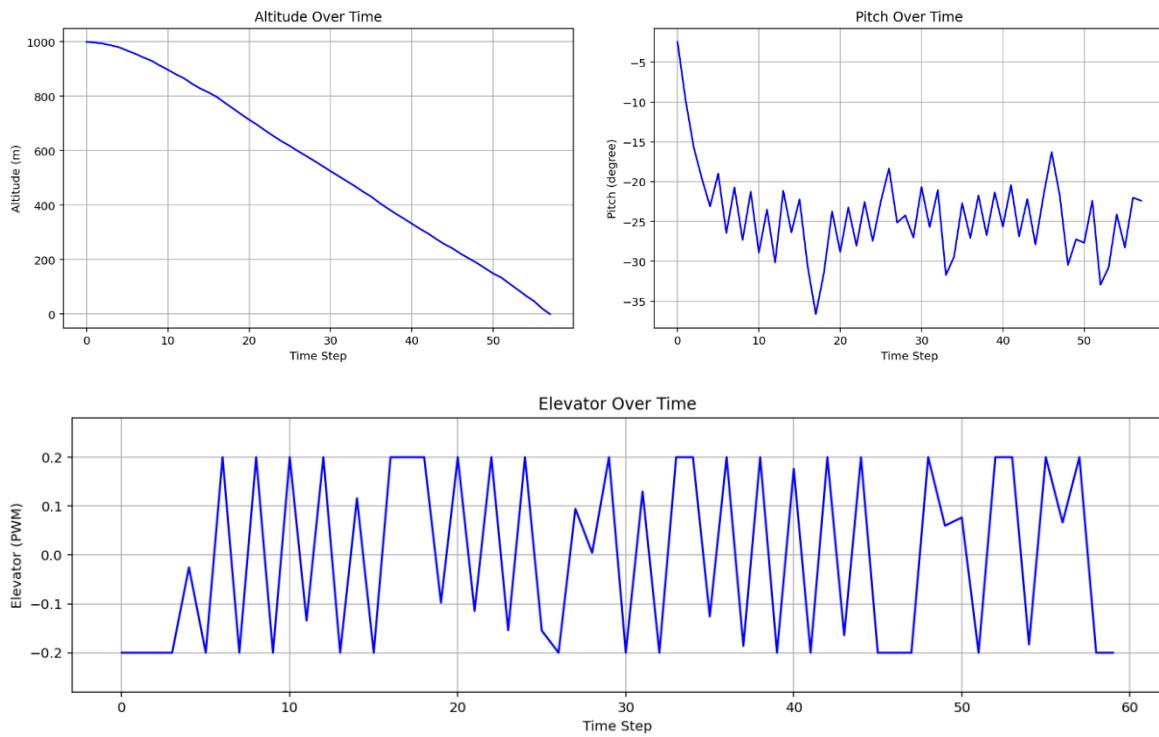
5. EKLER



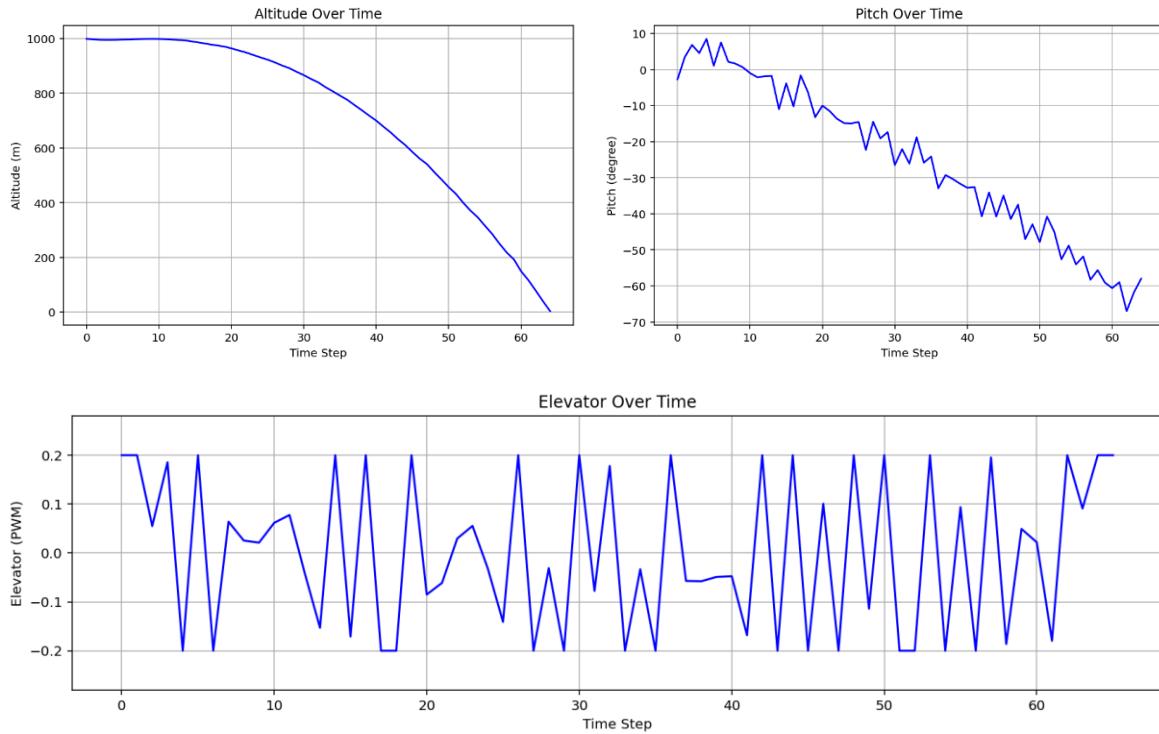
Şekil 5.1. 90° terminal yaw için simülasyon sonuçları



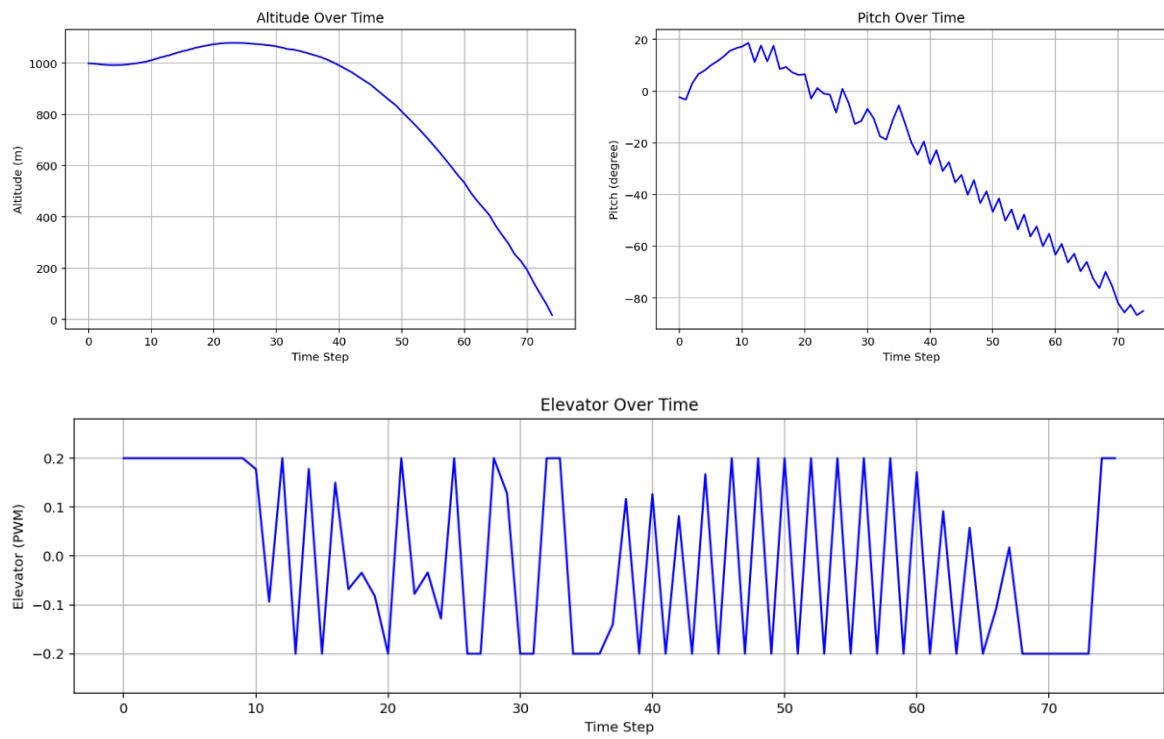
Şekil 5.2. 180° terminal yaw için simülasyon sonuçları



Şekil 5.3. 30° terminal pitch için simülasyon sonuçları

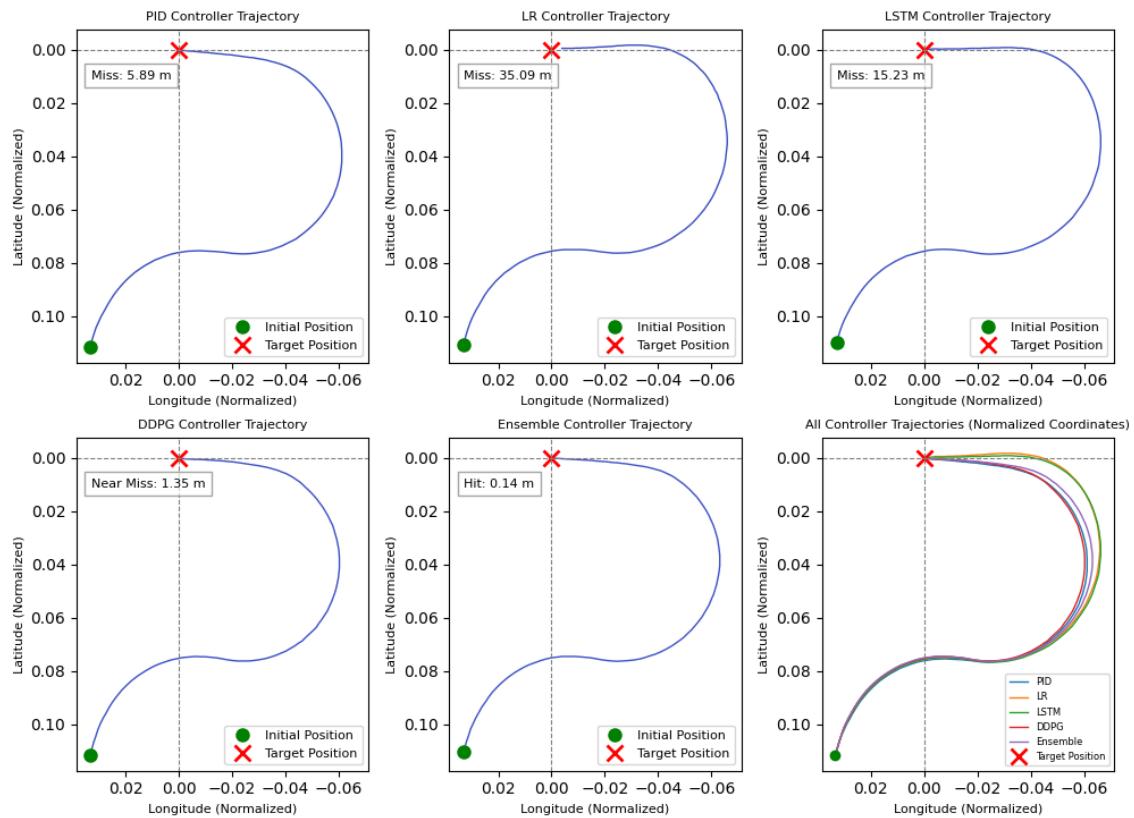


Şekil 5.4. 60° terminal pitch için simülasyon sonuçları

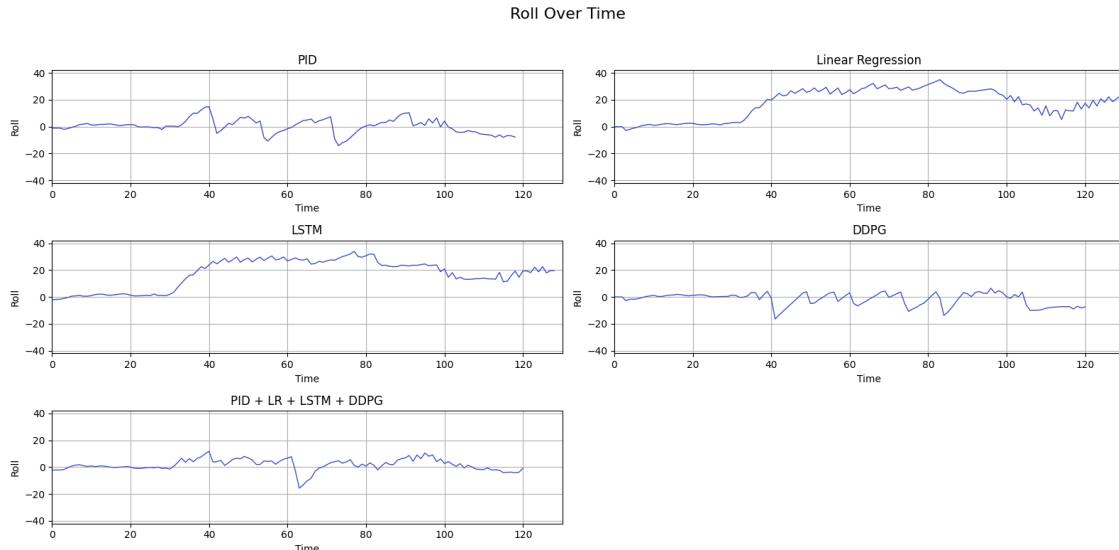


Şekil 5.5. 80° terminal pitch için simülasyon sonuçları

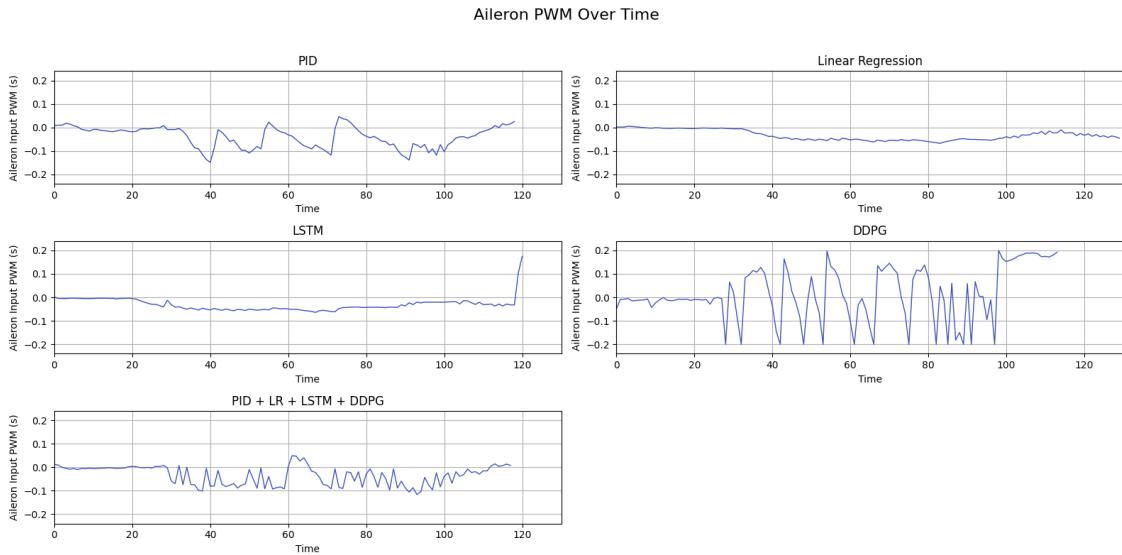
Missile Trajectory per Controller and Combined (Normalized Coordinates)



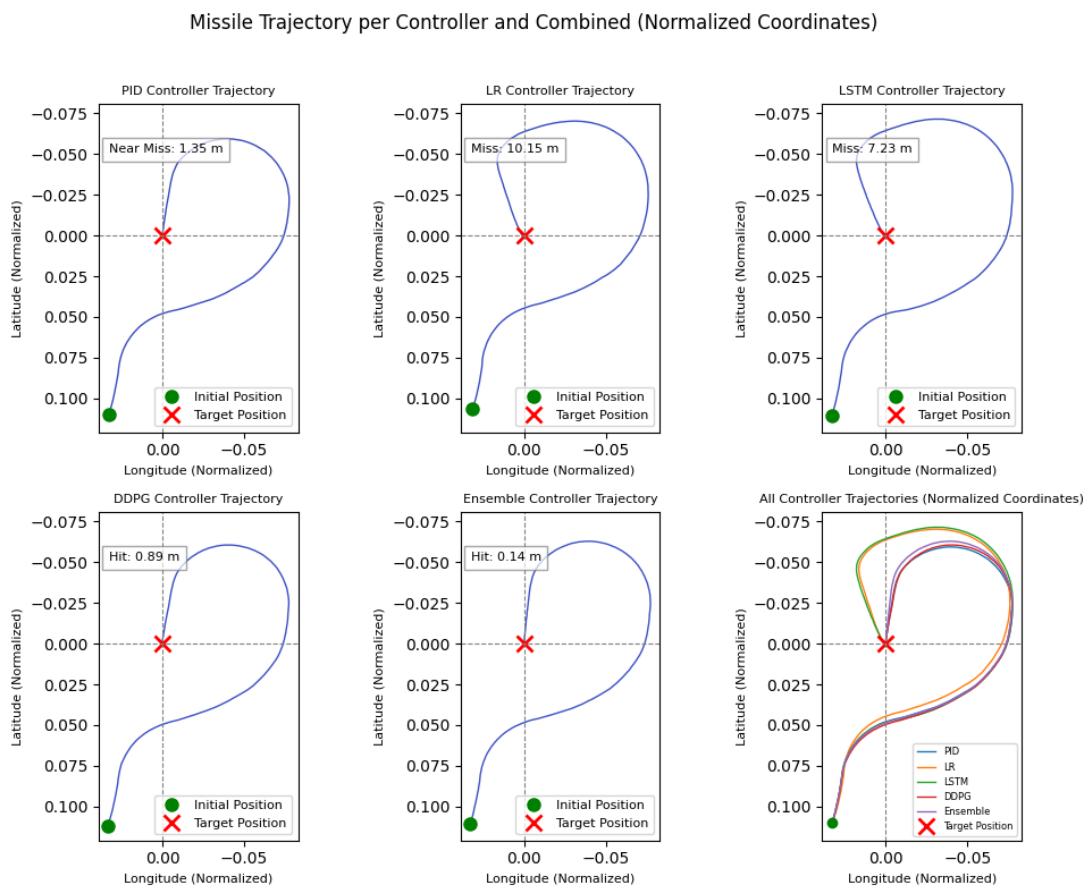
Şekil 5.6. Yuvarlanma stabilizasyonu senaryo 2 (terminal yaw 90 derece) uçuş yörüngelerinin karşılaştırılması



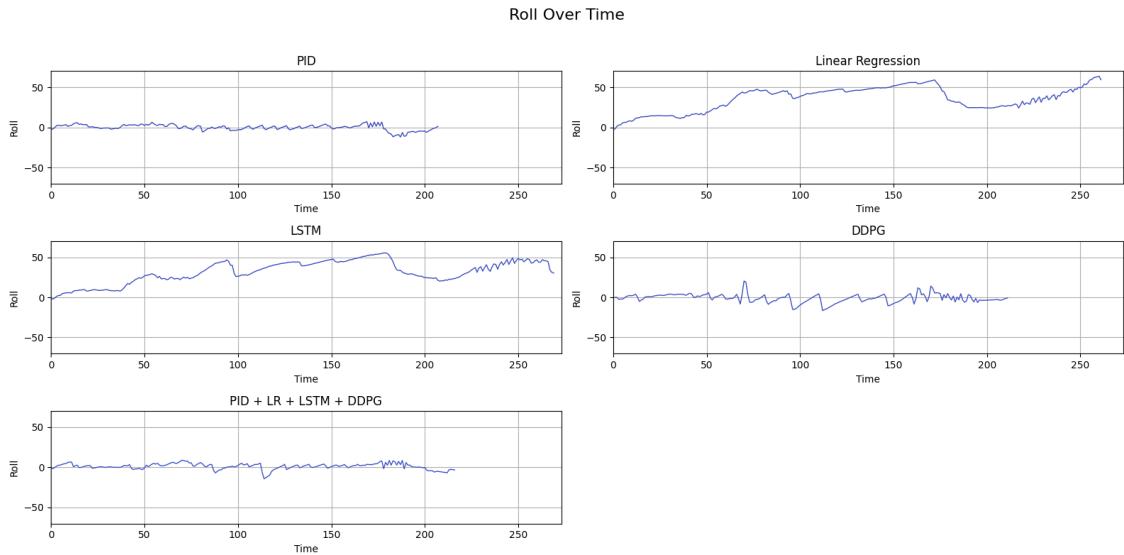
Şekil 5.7. Yuvarlanma stabilizasyonu senaryo 2 (terminal yaw 90 derece) roll açılarının karşılaştırılması



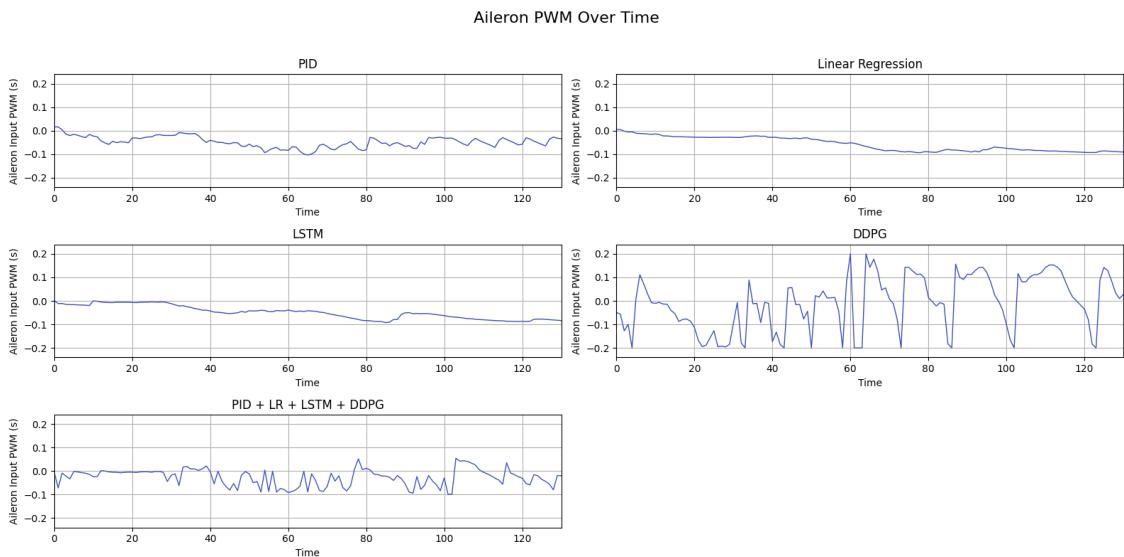
Şekil 5.8. Yuvarlanma stabilizasyonu senaryo 2 (terminal yaw 90 derece) aileron girdi karşılaştırılması



Şekil 5.9. Yuvarlanma stabilizasyonu senaryo 3 (terminal yaw 180 derece) uçuş yörüngelerin karşılaştırılması



Şekil 5.10. Yuvarlanma stabilizasyonu senaryo 3 (terminal yaw 180 derece) roll açılarının karşılaştırılması



Şekil 5.11. Yuvarlanma stabilizasyonu senaryo 3 (terminal yaw 180 derece) aileron girdi karşılaştırılması

6. KAYNAKLAR

- [1] A. Ryan, M. Zennaro, A. Howell, R. Sengupta and J. K. Hedrick, "An Overview of Emerging Results in Cooperative UAV Control," in *43rd IEEE Conference on Decision and Control (CDC)*, Nassau, 2004.
- [2] J. Danczuk, "Bayraktars and grenade-dropping quadcopters: How Ukraine and Nagorno-Karabakh highlight present air and missile defense shortcomings and the necessity of Unmanned Aircraft Systems," Army University Press, August 2023. [Online]. Available: <https://www.armyupress.army.mil/Journals/Military-Review/English-Edition-Archives/March-2024/Bayraktars-and-Grenade-Dropping-Quadcopters/>.
- [3] J. Bellingham, M. Tillerson, A. Richards and J. P. How, "Multi-Task Allocation and Path Planning for Cooperating UAVs," in *Cooperative Control: Models, Applications and Algorithms*, Kluwer Academic Publishers, 2003, p. 23–41.
- [4] S. Biswas, S. G. Anavatti and M. A. Garratt, "Path planning and task assignment for multiple UAVs in dynamic environments," in *Unmanned Aerial Systems: Theoretical Foundation and Applications*, 2021, pp. 81-102.
- [5] T. Huang, Y. Wang, X. Cao and D. Xu, "Multi-UAV Mission Planning Method," in *3rd International Conference on Unmanned Systems (ICUS)*, Harbin, China, 2020.
- [6] X.-p. Xu, X.-t. Y. W.-y. Yang, K. An, W. Huang and Y. Wang, "Algorithms and applications of intelligent swarm cooperative control: A comprehensive survey," *Progress in Aerospace Sciences*, 2022.
- [7] I. Bello, H. Pham, Q. V. Le, M. Norouzi and S. Bengio, "Neural Combinatorial Optimization with Reinforcement Learning," November 2016. [Online]. Available: <https://arxiv.org/abs/1611.09940>.
- [8] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina and L. Song, "Learning Combinatorial Optimization Algorithms over Graphs," April 2017. [Online]. Available: <https://arxiv.org/abs/1704.01665>.
- [9] M. Nazari, A. Oroojlooy, L. V. Snyder and M. Takáč, "Reinforcement Learning for Solving the Vehicle Routing Problem," in *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, Montréal, 2018.
- [10] H. Lu, X. Zhang and S. Yang, "A Learning-based Iterative Method for Solving Vehicle Routing Problems," 2019. [Online]. Available: <https://openreview.net/forum?id=BJe1334YDH>.
- [11] Y. Kang, Z. Pu, Z. Liu, G. Li, R. Niu and J. Yi, "Air-to-Air Combat Tactical Decision Method Based on SIRMs Fuzzy Logic and Improved Genetic Algorithm," in *Lecture Notes in Electrical Engineering*, vol. 644, Springer, 2021.
- [12] J. Carter, K. Schmid, K. Waters, L. Betzhold, B. Hadley, R. Mataosky and J. Halloran, *Lidar 101: An Introduction to Lidar Technology, Data, and Applications*, NOAA Coastal Services Center, 2012.

- [13] J. Pestana, J. L. Sanchez-Lopez, P. Campoy and S. Saripalli, "Vision Based GPS-denied Object Tracking and Following for Unmanned Aerial Vehicles," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Sweden, 2013.
- [14] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos and A. Martinelli, "Vision-Controlled Micro Flying Robots: From System Design to Autonomous Navigation and Mapping in GPS-Denied Environments," *IEEE Robotics & Automation Magazine*, pp. 26 - 40, 20 August 2014.
- [15] G. Balamurugan, J. Valarmathi and V. P. S. Naidu, "Survey on UAV navigation in GPS denied environments," in *International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, Paralakhemundi, 2016.
- [16] "Juno: New Origins," [Online]. Available: <https://www.simplerockets.com/>.
- [17] WeaponSystems.net, "WeaponSystems.net AIM-54 Phoenix," [Online]. Available: <https://weaponsystems.net/system/219-AIM-54+Phoenix>.
- [18] NoLuja, "JUNO," [Online]. Available: <https://www.simplerockets.com/c/9d732I/AIM-54C-Phoenix-Fully-working-air-to-air-missile>.
- [19] WNP78, "SR2Logger," 2020. [Online]. Available: <https://github.com/WNP78/SR2Logger>.
- [20] H. Zhu and S. Wu, "Leader–Follower Formation Reconfiguration Control for Fixed-Wing UAVs Using Multiplayer Stackelberg–Nash Game," *Drones*, vol. 9, no. 6, p. 439, 2025.
- [21] R. Olfati-Saber and R. M. Murray, "Consensus Problems in Networks of Agents With Switching Topology and Time-Delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520 - 1533, 2004.
- [22] Y. Jiang, T. Bai and Y. Wang, "Formation Control Algorithm of Multi-UAVs Based on Alliance," *Drones*, vol. 6, no. 12, p. 431, 2022.
- [23] B. Siciliano and O. Khatib, *Handbook of Robotics*, Springer, 2016.
- [24] P. Tabuada, *Verification and Control of Hybrid Systems: A Symbolic Approach*, Springer, 2009.
- [25] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.
- [26] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*, Princeton University Press, 2012.
- [27] S. Alqefari and M. E. B. Menai, "Multi-UAV Task Assignment in Dynamic Environments: Current Trends and Future Directions," *Drones*, vol. 9, no. 1, p. 75, 2025.
- [28] G. Venkataraman, "GitHub - pyKey," 2019. [Online]. Available: <https://github.com/gauthsvenkat/pyKey>.
- [29] G. Jocher, A. Chaurasia and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>.

- [30] J. Terven, D.-M. Córdova-Esparza and J.-A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Mach. Learn. Knowl. Extr.*, vol. 5, pp. 1680-1716, 2023.
- [31] B. Dwyer, J. Nelson and T. Hansen, "Roboflow (Version 1.0)," 2024. [Online]. Available: <https://roboflow.com>.
- [32] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *7th International Conference on Learning Representations*, New Orleans, 2019.
- [33] A. Ng and T. Ma, CS229 Lecture Notes, 2023.
- [34] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," February 2015. [Online]. Available: <https://arxiv.org/abs/1502.03167>. [Accessed April 2025].
- [35] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous control with deep reinforcement learning," September 2015. [Online]. Available: <https://arxiv.org/abs/1509.02971>. [Accessed November 2024].