Teknoloji Fakültesi

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ 2. ARA RAPORU

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

---

PROJE YAZARI

Ammar ABDURRAUF, Ahmet KURT, Serdar YILDIRIM

170421930, 170421022, 170421043

DANIŞMAN

Dr. Öğr. Üyesi Eyüp Emre ÜLKÜ

İL, TEZ YILI

İstanbul, 2025

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Öğrencisi …… nın "……" başlıklı bitirme projesi çalışması, …./…./….. tarihinde sunulmuş ve jüri üyeleri tarafından başarılı bulunmuştur.

Jüri Üyeleri

Prof. Dr. Adı SOYADI (Danışman)

Marmara Üniversitesi ..... (İMZA) .....

Doç. Dr. Adı SOYADI (Üye)

Marmara Üniversitesi ..... (İMZA) .....

Dr. Öğr. Üyesi Adı SOYADI (Üye)

Marmara Üniversitesi ..... (İMZA) .....

İÇİNDEKİLER

Sayfa

KISALTMALAR/ABBREVIATIONS

DDPG: Deep Deterministic Policy Gradient

GPS: Global Positioning System

HSS: Hava Savunma Sistemi

IMU: Inertial Measurement Unit

İHA: 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

TABLO LİSTESİ

Sayfa

ÖZET

Modern savaş ortamlarında yer tabanlı hava savunma sistemlerinin gelişimi, hava saldırılarının etkinliğini önemli ölçüde azaltmakta ve muharip uçaklara 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, Proportional Navigation algoritması kullanılarak füzenin terminal pitch açısı kontrol edilmiş ve Simplerockets 2 (SR2) simülasyon ortamında başarıyla test edilmiştir. Ardından, füzenin yuvarlanma (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. İHA'ların formasyon kontrolü, görev paylaşımı ve iletişim gibi alt görevleri ise MATLAB ve Python ortamlarında geliştirilen 2 boyutlu simülasyonlarda modellenmiş ve analiz 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 first stage, the terminal pitch angle of the missile was controlled using the Proportional Navigation algorithm and was successfully tested in the Simplerockets 2 (SR2) simulation environment. Secondly, an anti-roll stabilization system was developed using the PID controller and artificial intelligence-based controllers to prevent the roll instability of the missile. Additionally, a Bézier curve-based route planning algorithm was developed to control the missile's terminal yaw angle, resulting in the determination of optimal flight trajectories. Subtasks related to UAV operations, such as formation control, task allocation, and communication, were modeled and analyzed through 2D simulations developed in MATLAB. Furthermore, to address scenarios with limited GPS availability, an object recognition system based on the YOLOv8 algorithm was integrated into the missile system. Overall, this study proposes a comprehensive solution to enhance 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 saldırılarına karşı yüksek düzeyde direnç göstermektedir. Bu sistemlerin etkinliği, hem insanlı savaş uçakları hem de füzeler için ciddi tehditler oluşturmakta, 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 güdüm 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 güdüm sistemlerinin tasarımı. Yapay zekâ destekli yöntemlerin klasik kontrol algoritmalarıyla bütünleştirildiği 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 çalışmalardan farklılaştığı 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 saldırı stratejilerinin oluşturulması hedeflenmektedir.

1.1.1. Çoklu İHA Stratejileri

Çoklu İHA sistemleri, modern savunma ve saldırı 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 saldırılar gerçekleştirmelerini sağlamaktır.

Bu kapsamda, çoklu İHA sistemleri için geliştirilecek stratejiler, otonom rota planlama ve görev tahsisi, formasyon kontrolü ve koordinasyon, dinamik ortamlara adaptasyon ve senkronize saldırı 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 pekiştirmeli öğrenme (RL) ve derin öğrenme (DL) gibi yapay zeka yöntemleri kullanılarak 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 çıktıkları dönemden bu yana önemli ölçüde gelişerek basit navigasyondan, sensörler, aktüatörler 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 yörünge 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 artır. Bu çalışma, füze kontrol ve navigasyonundaki zorlukları çözmek amacıyla, klasik kontrol yöntemlerini tamamlayacak şekilde denetimli öğrenme ve pekiştirmeli öğrenme yaklaşımlarının entegrasyonunu araştırmaktadır.

Bu çalışmanın amacı, füze kontrol ve navigasyon sistemine 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 sahasını hava saldırılarına 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 sahasına etkili bir saldırı gerçekleştirilmesini zorlaştırmaktadır. Ayrıca, düşman hava savunma sistemini yok etmeye yönelik görevler, jet uçakları ile yapıldığında pilotlar için son derece tehlikelidir. Bu nedenle, pilot kayıplarını en aza 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 tahsisi 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 incelemişlerdir [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 yaklaşıma dayanmaktadır. Literatürdeki pek çok çalışma teorik modellemeler veya sınırlı görev tanımları üzerinde yoğunlaşırken, 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. MATERYAL VE YÖNTEM

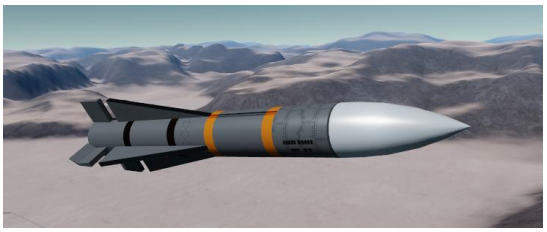
Bu bölümde, çalışmanın yürütülmesinde kullanılan donanım, yazılım ve simülasyon ortamları “Materyal” başlığı altında; geliştirilen algoritmalar, kontrol mekanizmaları ve deneysel yaklaşımlar ise “Yöntem” başlığı altında ayrı ayrı sunulmaktadır. Bu yapı sayesinde, hem kullanılan araçların teknik özellikleri 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 iletişimi 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 değinilmiştir.

Şekil 2.1. AIM-54C by Hughes Aircraft *[16]*



Şekil 2.2. AIM-54C Phoenix by NoLuja [17]

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

SR2, roketler ve havacılık araçları için özel olarak geliştirilmiş bir uçuş simülatörüdür [18]. 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 [19]. Simülasyonda kullanılan füze modeli ise NoLuja tarafından geliştirilen AIM-54C Phoenix modelidir [20]. Şekil 2.1 ve Şekil 2.2'de, gerçek hayattaki AIM-54C Phoenix füzesi ile SR2 simülatörüne entegre edilmiş füze modelinin görselleri sunulmaktadır.

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 tasarlanan 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.

Bunun yanı sıra, proje kapsamında geliştirme sürecini desteklemek amacıyla bazı üçüncü parti kütüphaneler de kullanılmıştır. Kullanılan başlıca kütüphaneler Tablo 2.1'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 ve Spyder IDE kullanıldı. Anaconda, program geliştirmeyi kolaylaştırıp sürümleri, paketleri ve Python ortamlarını yönetmeye yardımcı olmuştur. Spyder IDE, veri analizini kolaylaştıran ücretsiz bir Python IDE'dir.

Bu projede Python ile ayrıca öncelikli olarak MATLAB'de başlatılan İHA simülasyonları gerçekleştirilmiştir. Python ortamında geliştirilen simülasyon, nesne yönelimli programlama (OOP) yaklaşımıyla oluşturulmuştur.

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 [21].

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 gerçekleştirilmiştir. Daha sonra, daha kapsamlı ve esnek bir simülasyon ortamı oluşturmak amacıyla Python ortamına geçilmiştir.

2.1.6. Unity

Unity ortamı, Python tarafından gönderilen verilerin alınması ve görselleştirilmesi amacıyla kullanılmaktadır. Uygulama içerisinde, UDP protokolü ile iletilen verileri dinleyen bir bileşen bulunmaktadır. Bu bileşen, arka planda çalışan bir iş parçacığı (thread) kullanarak veri akışını kesintisiz bir şekilde takip eder. Gelen veriler, Unity'nin ana iş parçacığında işlenerek gerekli görselleştirmeler veya simülasyon güncellemeleri gerçekleştirilir. Bu yapı sayesinde, Python'dan gelen gerçek zamanlı veriler Unity ortamında anlık olarak takip edilebilir ve kullanıcıya görsel geri bildirim sağlanabilir.

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 (navigasyon) 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 sağlanabilmesi için, gönderilecek verilerin tanımlandığı 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 için 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. Aşağıda, 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
```

UDP paketlerinden veri çıkarmak için readPacket fonksiyonu kullanılabilmektedir.

```python
import socket
import time

def send_data():
    udp_ip = "127.0.0.1"
    udp_port = 5005
    sayac = 0

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    try:
        while True:
            message = f"Veri !{sayac}"

            sock.sendto(message.encode(), (udp_ip, udp_port))
            print(f"'{message}' {udp_ip}:{udp_port} adresine gönderildi. {sayac}")
            sayac+=1
            time.sleep(1)
    except Exception as e:
        print(f"Hata: {e}")
    finally:
        sock.close()

if __name__ == "__main__":
    send_data()
```

Şekil 2.4. Port aracılığıyla veri göndermek için örnek Python kodu

```csharp
public class UdpListener : MonoBehaviour
{
    public int listenPort = 5005;
    private Thread listenerThread;
    private UdpClient udpClient;

    void Start() ···

    private void StartListening()
    {
        udpClient = new UdpClient(listenPort);
        IPEndPoint remoteEndPoint = new IPEndPoint(IPAddress.Any, listenPort);
        try
        {
            while (true)
            {
                byte[] receivedBytes = udpClient.Receive(ref remoteEndPoint);
                string receivedData = Encoding.UTF8.GetString(receivedBytes);

                UnityMainThreadDispatcher.Instance().Enqueue(() =>
                {
                    Debug.Log($"[{remoteEndPoint}] Gelen veri: {receivedData}");
                });
            }
        }
        catch (Exception e)
        {
            Debug.LogError($"Hata: {e.Message}");
        }
        finally
        {
            udpClient.Close();
        }
    }
    void OnDestroy() ···
```

Şekil 2.5. C# kullanarak UDP üzerinden veri göndermek için Unity programı

2.2.2. 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 sağlaması amacıyla çeşitli algoritmaların araştırılması ve uygulanabilirliklerinin değerlendirilmesi planlanmaktadır. Bu kapsamda, İHA'ların hareketlerinin simülasyonu, formasyon kontrolü ve görev tahsisi gibi konular üzerinde çalışmalar sürdürülmektedir.

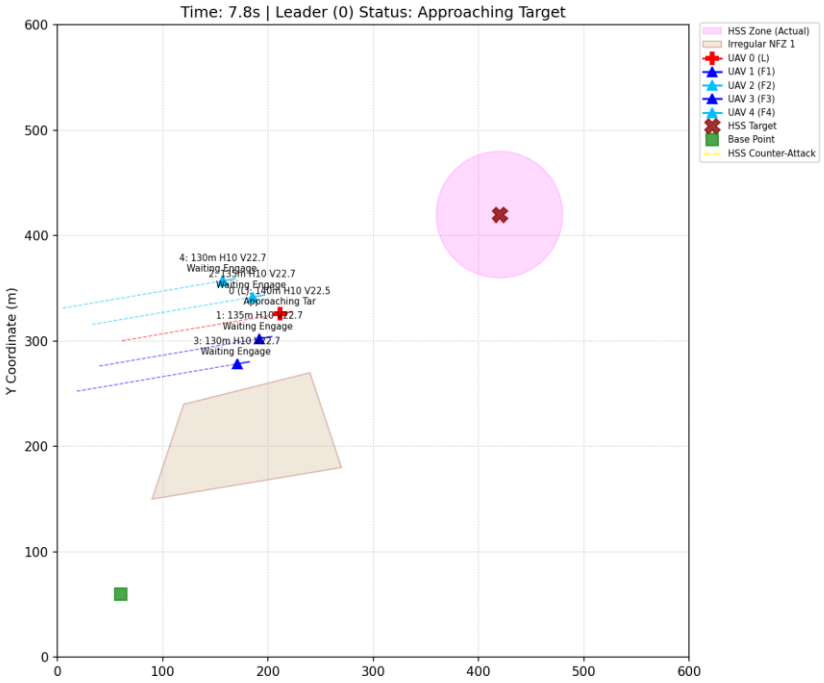
Projenin ilk aşamasında, İHA'ların hareketlerini 2D grafikler üzerinde simüle etmek amacıyla MATLAB ortamında temel bir kod geliştirilmiştir. Bu simülasyon, İHA'ların başlangıç noktasından hedefe doğru hareketini ve bu süreçte geçiş noktalarını (waypoints) kullanarak hedefe ulaşmalarını modellemektedir.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

MATLAB ortamında elde edilen temel sonuçların ardından, daha esnek ve otomatik bir simülasyon altyapısı oluşturmak amacıyla Python programlama diline geçilmiştir. İlk aşamalar, tek bir İHA'nın uçuş yeteneklerini doğrulamaya ayrılmıştır. Bu testler sırasında, bir İHA'nın kalkıştan itibaren belirli bir irtifaya güvenli bir şekilde tırmanması, hava seyrinde önceden belirlenmiş ara noktaları takip etmesi ve nihayetinde kontrollü bir iniş gerçekleştirmesi gibi kritik uçuş fazları incelenmiştir. Bu adımlar, hız, yönelme ve irtifa kontrolü için geliştirilen algoritmaların hassasiyetini ve güvenilirliğini bireysel platform düzeyinde anlamak için temel bir zemin sunmuştur. Elde edilen bulgular, her bir İHA'nın temel uçuş görevlerini öngörülebilir bir şekilde yerine getirebileceğini ve bağımsız olarak seyrüsefer yapabileceğini göstermiştir.

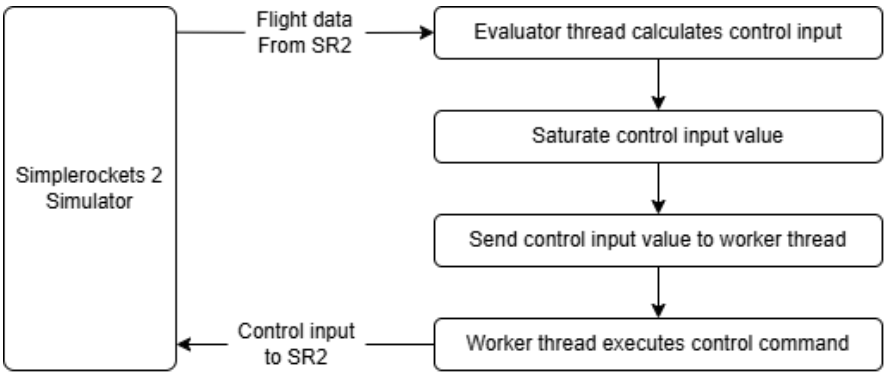
Bir sonraki aşamada, çoklu İHA sistemlerinde koordinasyonun sağlanması amacıyla konsensus prensipleri araştırılmış ve uygulanmıştır. Bu bağlamda, bir lider-takipçi formasyon kontrol modeli benimsenmiştir. Bu modelde, belirlenen bir İHA lider rolünü üstlenirken, diğer İHA'lar liderin hareketini belirli uzaysal ofsetlerle (konum, hız ve yönelimdeki sapmalar) takip ederek tanımlanmış bir formasyon düzenini korumaya çalışmıştır. Lider İHA'nın önceden belirlenmiş bir dizi ara noktayı takip etmesiyle, takipçi

İHA'ların bu formasyon yapısını bozmadan senkronize bir şekilde liderle hareket etmesi başarıyla sergilenmiştir. Bu deneyler, çoklu İHA'lar arasında uyumlu bir operasyonel düzen oluşturmanın ve sürdürmenin temellerini atmıştır.

Çalışmanın bu bölümünün mevcut odağı ve esas amacı, önceki aşamalarda geliştirilen kontrol ve koordinasyon yeteneklerini daha karmaşık ve gerçekçi bir operasyonel senaryoya entegre eden çoklu İHA saldırı görevi simülasyonu üzerinedir. Bu simülasyon, İHA'ların bir hedefe (HSS Target) intikal etmesini, angajman görevini tamamlamasını ve ardından üsse güvenli bir şekilde geri dönmesini içeren kapsamlı bir görev akışını modellemektedir. Özellikle, çevresel kısıtlamalara uyum, bu aşamanın kritik bir parçasını oluşturmaktadır. Hem dairesel hem de düzensiz poligonal şekilli Uçuşa Yasak Bölgeler (NFZ) tanımlanmış ve İHA'ların bu bölgelere yaklaşırken veya yanlışlıkla içine girdiklerinde otomatik ve proaktif kaçınma manevraları gerçekleştirmesi sağlanmıştır. Ayrıca, lider İHA'nın görev sırasında bir NFZ'yi ihlal etmesi gibi potansiyel arıza durumlarına karşı, dinamik lider değişimi (leader election) mekanizması entegre edilmiştir. Bu mekanizma, bir liderin devre dışı kalması durumunda, kalan aktif takipçi İHA'lar arasından belirlenen kriterlere göre (örneğin, üsse en yakın olan) yeni bir lider seçilerek görevin kesintisiz bir şekilde sürdürülmesini temin etmeyi amaçlamaktadır.



Şekil 2.6. Matplotlib ile çoklu İHA saldırı görevi simülasyonu erken aşaması



Şekil 2.7. Python'dan SR2'ye füze kontrol akış şeması

2.2.3. Python üzerinden füze kontrolü

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 [22]. 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 2.4'te gösterilmektedir.

Worker thread'ler, thread pool içerisinde beklemede 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.

2.2.4. Füze navigasyonu ve kontrolü

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.

Bu bölümde, 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 10° ila 90° 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 (örneğin PID kontrolörü) hem de yapay zeka tabanlı yöntemler (örneğin doğrusal regresyon, sinir ağları ve pekiştirmeli öğrenme) uygulanmıştır.

2.2.4.1. Terminal yaw açısı kontrolü

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

Bu çalışmada odaklanılan üç terminal yaw açısı şunlardır:

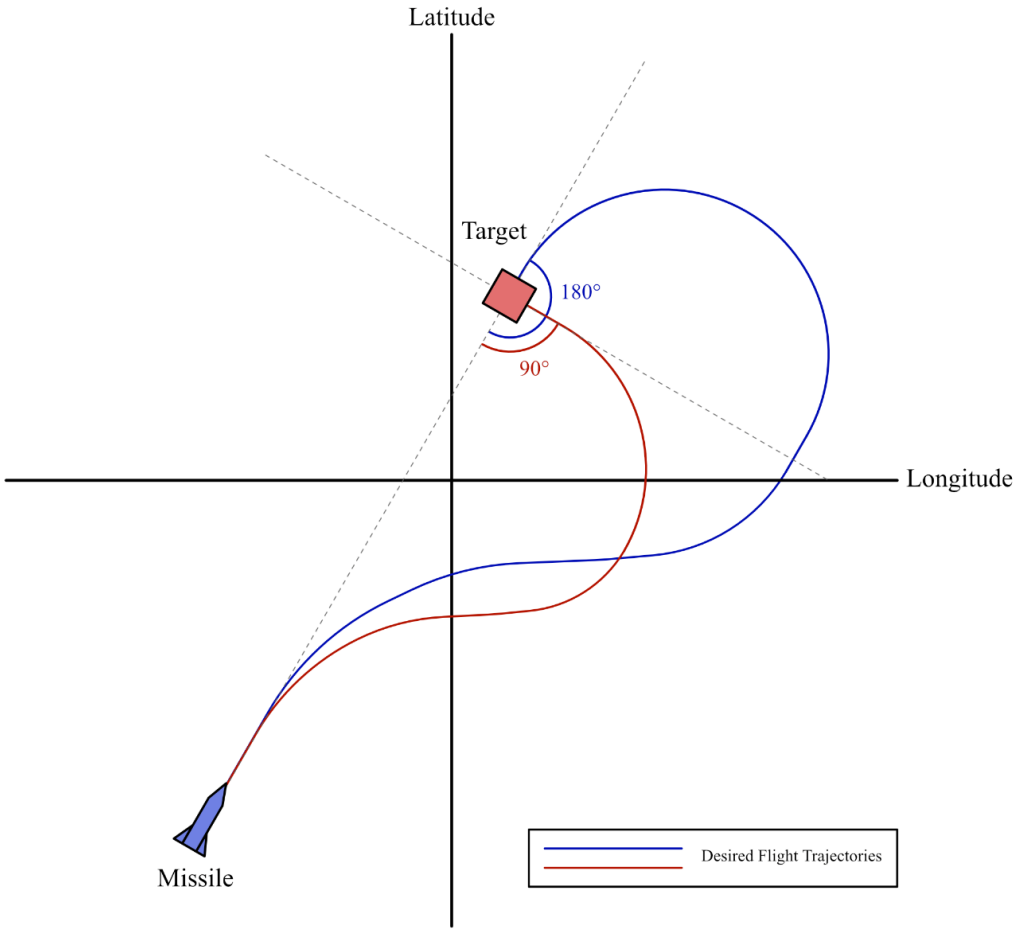
- 0° (hedefin ön tarafından yapılan saldırı)
- 90° (hedefin sağ tarafından yapılan saldırı)
- 180° (hedefin arka tarafından yapılan saldırı)

270° terminal yaw açısı ise ayrıca ele alınmamıştır, çünkü bu açı 90° terminal yaw açısının ters yöndeki hesaplamasına karşılık gelmektedir. Şekil 2.8'de terminal yaw açıları görselleştirilmektedir.

Şekil 28'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 gerekmesidir.

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

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.



Şekil 2.8. İstenilen terminal yaw açıları ve uçuş yörüngesi

Bezier eğrisi formülü aşağıda verilmiştir ve Şekil 2.8, bu formülün Python'daki Matplotlib kütüphanesi ile görselleştirilmiş halini göstermektedir:

$$B(t) = (1-t)3P0 + 3(1-t)2tP1 + 3(1-t)t2P2 + t3P3, \;\; t \in [0,1] \qquad (2.1)$$

Burada:

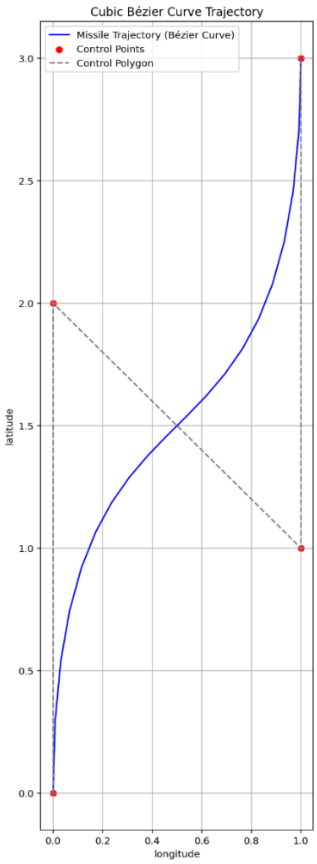
- P0, P1, P2, P3 kontrol noktalarını (her biri x ve y koordinatlarıyla) ifade etmektedir.
- t, eğri parametresidir (0 ile 1 arasında değişmektedir).
- $B(t) = \big(x(t), y(t)\big)$, t anındaki eğri üzerindeki bir noktanın koordinatlarını vermektedir.

```python
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 2.9. Python'da Bezier eğrisi kodu ve görselleştirilmesi

2.2.4.2. 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 yaklaşımlardan 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 çalışır. Genel olarak, PNG algoritması füzenin lateral (yanal) ivmesini, LOS açısının değişim hızına (rate of LOS) orantılı olarak ayarlar.

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

1 Sabit değişkenler tanımlanır:

Düz mesafe: D

Gezegenin yarıçapı: r

Çap dönüm 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 When target position is known (φ_{target}, λ_{target}):

5 Düz mesafe D derece cinsinden şu formül ile hesaplanır:

D = 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 hesaplamalar yapılır:

9

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

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

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

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

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

$$waypoint_1^{\varphi} = \varphi_{target} + \Delta\varphi$$

$$waypoint_1^{\lambda} = \lambda_{target} - \Delta\lambda$$

11 Sonra düz uçuş segmentinin ilerisinde bir dönüş noktası tanımlanır:

$$waypoint_2^{\varphi} = waypoint_1^{\varphi} + \Delta\varphi_r$$

$$waypoint_2^{\lambda} = waypoint_1^{\lambda} - \Delta\varphi_r$$

12 Başlangıç noktası ($\varphi_{initial}$, $\lambda_{initial}$) ile dönüş noktası ($waypoint_2^{\varphi}, waypoint_2^{\lambda}$) arasında düzgün bir geçiş rotası oluşturmak için Bezier eğrisi kullanılır.

13 Nihai izlenecek rota (Final Waypoints): Bezier eğrisinden elde edilen yol noktaları + ($waypoint_1^{\varphi}, waypoint_1^{\lambda}$) + ($\varphi_{target}$, λ_{target})

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

1 Sabit değişkenler tanımlanır:

Düz mesafe: D

Gezegenin yarıçapı: r

Çap dönüm 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 When target position is known (φ_{target}, λ_{target}):

5 Düz mesafe D derece cinsinden şu formül ile hesaplanır:

D = 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 hesaplamalar yapılır:

9

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

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

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

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

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

$$waypoint_1^{\varphi} = \varphi_{target} - \Delta\varphi$$

$$waypoint_1^{\lambda} = \lambda_{target} + \Delta\lambda$$

11 Sonra düz uçuş segmentinin ilerisinde bir dönüş noktası tanımlanır:

$$waypoint_2^{\varphi} = waypoint_1^{\varphi} - \Delta\varphi_r$$

$$waypoint_2^{\lambda} = waypoint_1^{\lambda} - \Delta\varphi_r$$

12 Başlangıç noktası ($\varphi_{initial}$, $\lambda_{initial}$) ile dönüş noktası ($waypoint_2^{\varphi}, waypoint_2^{\lambda}$) arasında düzgün bir geçiş rotası oluşturmak için Bezier eğrisi kullanılır.

13 Nihai izlenecek rota (Final Waypoints): Bezier eğrisinden elde edilen yol noktaları + ($waypoint_1^{\varphi}, waypoint_1^{\lambda}$) + ($\varphi_{target}$, λ_{target})

Şekil 2.10. Hedef görsellerin ve sınırlayıcı kutularının örnekleri

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

$$a_n = N \cdot V_c \cdot \dot{\lambda} \tag{2.2}$$

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).
- $\dot{\lambda}$, pitch eksenindeki (düşey eksen) LOS açısının değişim hızıdır.

PNG yaklaşımına ek olarak, bu çalışmada object detection (nesne tespiti) yöntemine dayalı deneysel bir yöntemle de pitch açısının kontrolü gerçekleştirilmiştir. 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 görünüyorsa, füze pitch açısını artırır. Tersine, hedef üst kısımda görünüyorsa, füze pitch açısını azaltır. Bu işlem, hedef kamera çerçevesinin ortasında kilitlenene kadar devam eder.

2.2.4.3. Hedef tespiti için YOLOv8 modelinin eğitimi

Füzenin üzerindeki kamera aracılığıyla hedefi doğru bir şekilde tanımlayabilmesi için YOLOv8 algoritması [23] 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 [24]. YOLOv8, yüksek çıkarım (inference) hızı ve yeterli doğruluğu nedeniyle tercih edilmiştir ve bu özellikleri 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 görüntüleriyle toplanmış ve hedef olarak belirlenen Surface-to-Air Missile (SAM) launcher vehicle (karadan havaya füze fırlatma aracı) Roboflow aracı [25] ile etiketlenmiştir. Toplamda 1.000 ekran görüntüsü toplanmış ve her görüntüde hedefin konumunu gösteren bir bounding box çizilmiştir. Şekil X, hedefe ait bazı örnek görüntüleri 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 [26]
- Epochs sayısı: 100 (önceden eğitilmiş model ile tekrar eğitilmiştir)

Şekil 2.10 ve 2.11, 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 sayısındaki belirgin fark, modellerin performansını doğrudan etkilemektedir. N modeli daha hafif ve hızlı çıkarım (inference) süresi sunmasına rağmen, doğruluk (accuracy) açısından M modelinin gerisinde kalmaktadır.

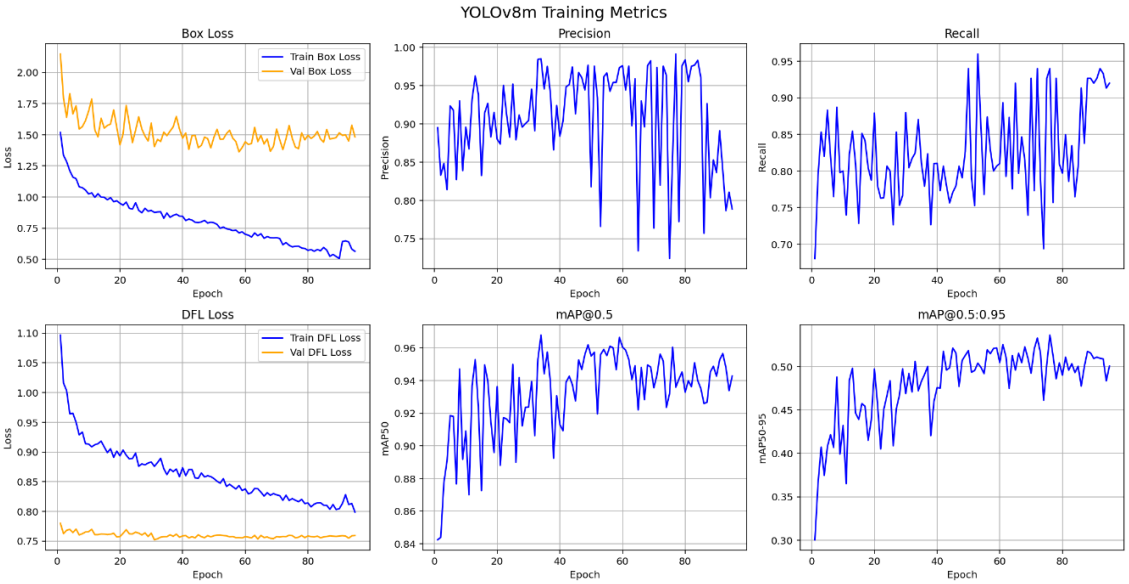
c.) Füze navigasyon sistemine entegrasyon

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 gerçekleştirmektedir. 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.

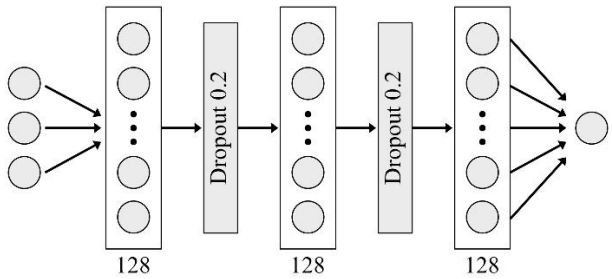
Füzenin pitch açısını kontrol eden elevator yüzeyinin kontrol girdisini elde etmek amacıyla, bir yapay sinir ağı (YSA) modeli eğitilmiş ve sistemin kontrol birimine entegre edilmiştir. Bu YSA modeli; kamera sensöründen elde edilen görüntüdeki hedefin merkez noktasının Y ekseninden sapma miktarı (distance 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 mimarisi Şekil 2.12'de 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 yönelmede daha esnek bir kontrol stratejisi izleyebilmektedir.

Şekil 2.11. YOLOv8n modelin eğitim metrikleri



Şekil 2.12. YOLOv8m modelin eğitim metrikleri



Şekil 2.13. Elevator kontrol girdisini üreten YSA modeli

2.2.4.4. 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 tasarımı

Oransal–İntegral–Türevsel (PID) kontrol yöntemi, geliştirilen diğer kontrolcülerin 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 hatasını, yuvarlanma hızı değerini ve birikimli yuvarlanma açısı hatasını hesaplayarak uygun kontrol sinyalini üretmektedir.

PID kontrolörünün tasarımındaki başlıca zorluklardan biri, Kp (oransal kazanç), Ki (integral kazanç) ve Kd (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 kullanılabilecek 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 varsayım (hipotez) fonksiyonu elde edilmektedir [27]:

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

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.

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

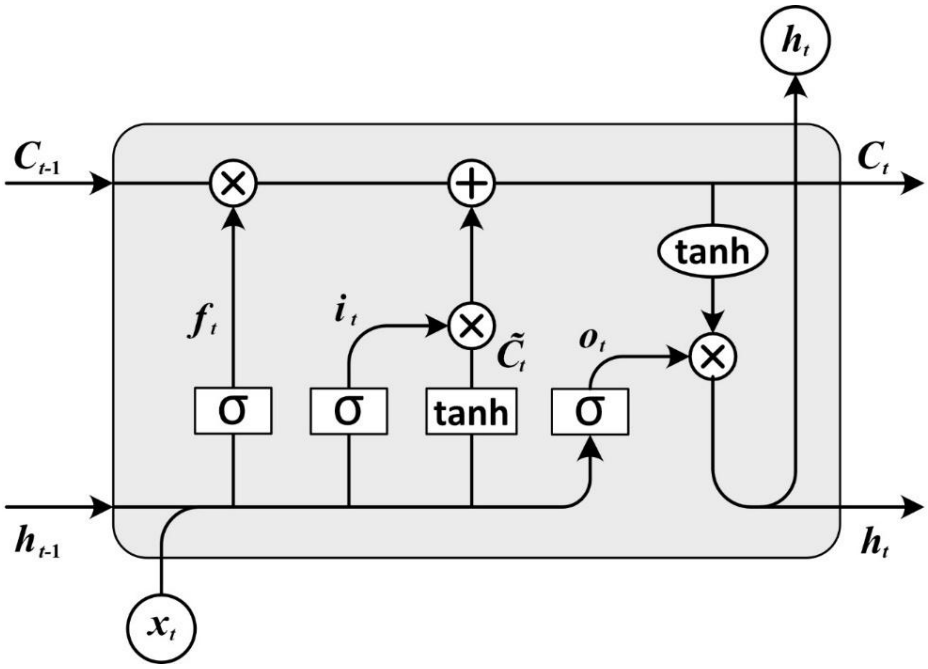
Uzun Kısa Süreli Bellek (Long Short-Term Memory, LSTM) ağları, bu tür ardışık veri problemlerini çözmek amacıyla geliştirilmiş Tekrarlayan Sinir Ağı (Recurrent Neural Network, RNN) türüdür. LSTM, klasik RNN'lerin karşılaştığı gradyan sönümlenmesi (vanishing gradient) ve gradyan patlaması (exploding gradient)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

Bu bağlamda, LSTM tabanlı bir kontrolör tasarımının temel amacı, uygun aileron kontrol girişlerini öğrenerek füzenin yuvarlanma açısındaki sapmaları azaltmaktır. Doğrusal regresyon modellerinin aksine, LSTM modelleri yalnızca bir anlık veriye göre çıkış üretmekle kalmaz; bunun yerine, önceki 10 zaman adımına (yaklaşık 2 saniyelik geçmişe; 5 Hz örnekleme 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.

LSTM’nin detaylı mimarisi Şekil 2.13’te gösterilmektedir.

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 [28]. Modelin eğitimi, başlangıç öğrenme oranı 0.005 olarak belirlenen Adam optimizasyon algoritması ile gerçekleştirilmiştir.

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 katsayısı ile azaltmakta ve minimum öğrenme oranı olarak 10^-8 sınırı belirlenmiştir.



Şekil 2.14. LSTM hücre yapısı

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

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 davranışı, 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 çıktısı 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 [29]'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 Z'te gösterilmiştir (daha sonra eklenecektir).
- Durum-geçiş çiftlerini (s, a, r, s') depolayan replay buffer,
- Replay buffer'dan mini-batch'ler halinde rastgele örnekleme 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 ([$\theta_{target} \leftarrow \tau \cdot \theta + (1 - \tau) \cdot \theta_{target}$], burada 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:

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

Algoritma

DDPG ajanının 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 ağı (target critic network) çıktısını temel alır. Hedef eleştirmen ağı, 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 ağı, kendi gradyan bilgisine ek olarak eleştirmen ağının gradyanından da yararlanmaktadır. Bu çalışmada kullanılan DDPG yönteminin algoritması ve uygulanışı, [29] 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)

Algoritma 3. Deep deterministic policy gradient algoritması

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

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

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;

$$N = \begin{cases} Random\ noise, & with\ probability\ \varepsilon \\ 0, & with\ probability\ 1-\varepsilon \end{cases}$$

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 örneklenmesiyle)

13 Eleştirmen ağını, kaybı en aza indirgeyerek 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:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \quad (2.12)$$

16 Eğer işlem tamamlandıysa:

İç döngüyü sonlandır; bir sonraki bölüme geç.

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 sınırlandırılmıştır ve ajan tarafından aileron kontrol tuşuna kaç saniye basılması gerektiğini ifade etmektedir.

3. MODEL EĞİTİMİ VE SİMÜLASYON SONUÇLARI

3.1. Füze Sisteminin Test Simülasyonları

3.1.1. Termıinal Yaw ve Pitch Açısı Test Simulasyonu

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.

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. Tasarlanan oransal seyir (proportional navigation) yöntemi, yüksek doğrulukla sonuçlar üretmiş ve etkinliğini kanıtlamıştır.

grafikler ve resimler sonradan eklenecek ve tartışılacaktır

3.1.2. YOLOv8 modeli ile Pitch Açısı Kontrolu

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 gerçekleştirilmiştir. 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:

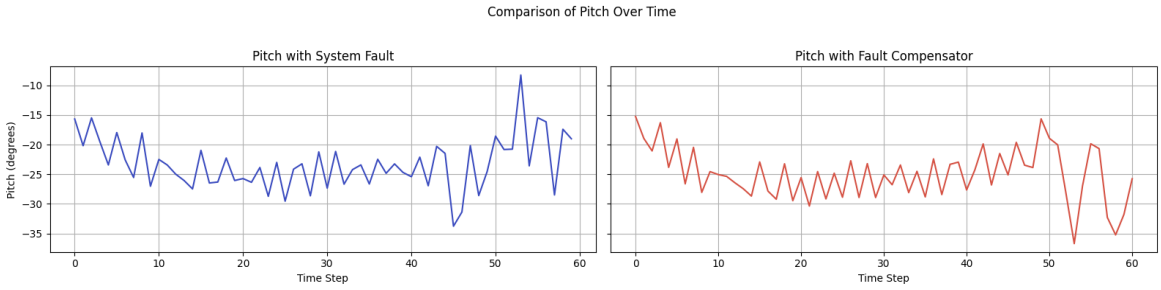
Eğer hedefe olan mesafe 650 metre veya daha az ise:

- %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.

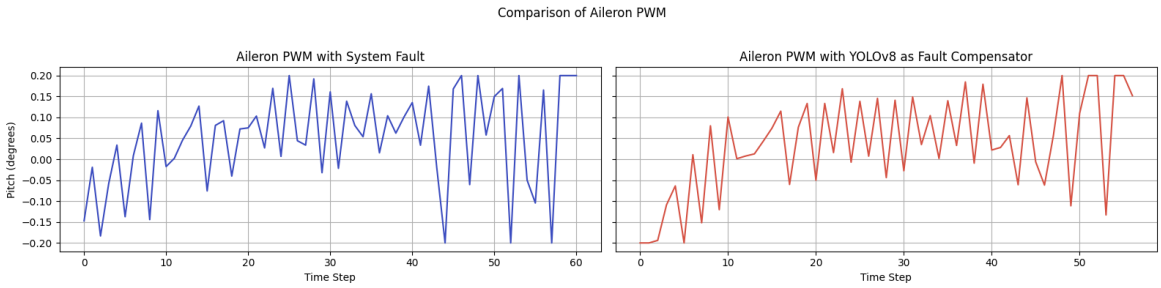
Deneysel sonuçlar, pitch kontrol sisteminin sensör arızası (fault) durumlarında saldırı etkinliğini önemli ölçüde kaybettiğini, hatta füzenin hedeften sapmasına neden olduğunu ortaya koymuştur. Buna karşın, YOLOv8 ile entegre edilmiş sistemin, diğer sensörlerden kaynaklanan arızalarda kontrol girişini düzeltici etkide bulunduğu ve kontrol performansını

artırdığı gözlemlenmiştir. Simülasyon sonuçlarına göre, YOLOv8 kullanılmayan senaryoda füze hedeften 6.20 metre saparken, YOLOv8'in kullanıldığı durumda hedefe yalnızca 1.07 metre sapma ile, neredeyse isabet derecesinde yaklaşmıştır.

Şekil 3.1 ve 3.2, sensör arızalarının olduğu bir ortamda YOLOv8 modelinin fault kompansatörü (bozukluğu telafi edici) olarak kullanıldığı ve kullanılmadığı durumlara ait simülasyon sonuçlarını göstermektedir.



Şekil 3.1. Pitch açıların karşılaştırılması



Şekil 3.2. Elevator girdi karşılaştırılması

Grafikler incelendiğinde, yaklaşık olarak 50. adımda 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 değerlendirmeye dahil edilmiştir.

Tablo 3.1. 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

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 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 3.1'de sunulmuştur. Şekil 3.3 ise, her bir kontrolörün değerlendirmesi sırasında elde edilen roll ve aileron değişimlerini grafiksel olarak göstermektedir.

Birinci senaryo kapsamında gerçekleştirilen simülasyon testlerinin sonuçlarına göre, Ensmble 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 davranışı 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 saldırı 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 saldırı

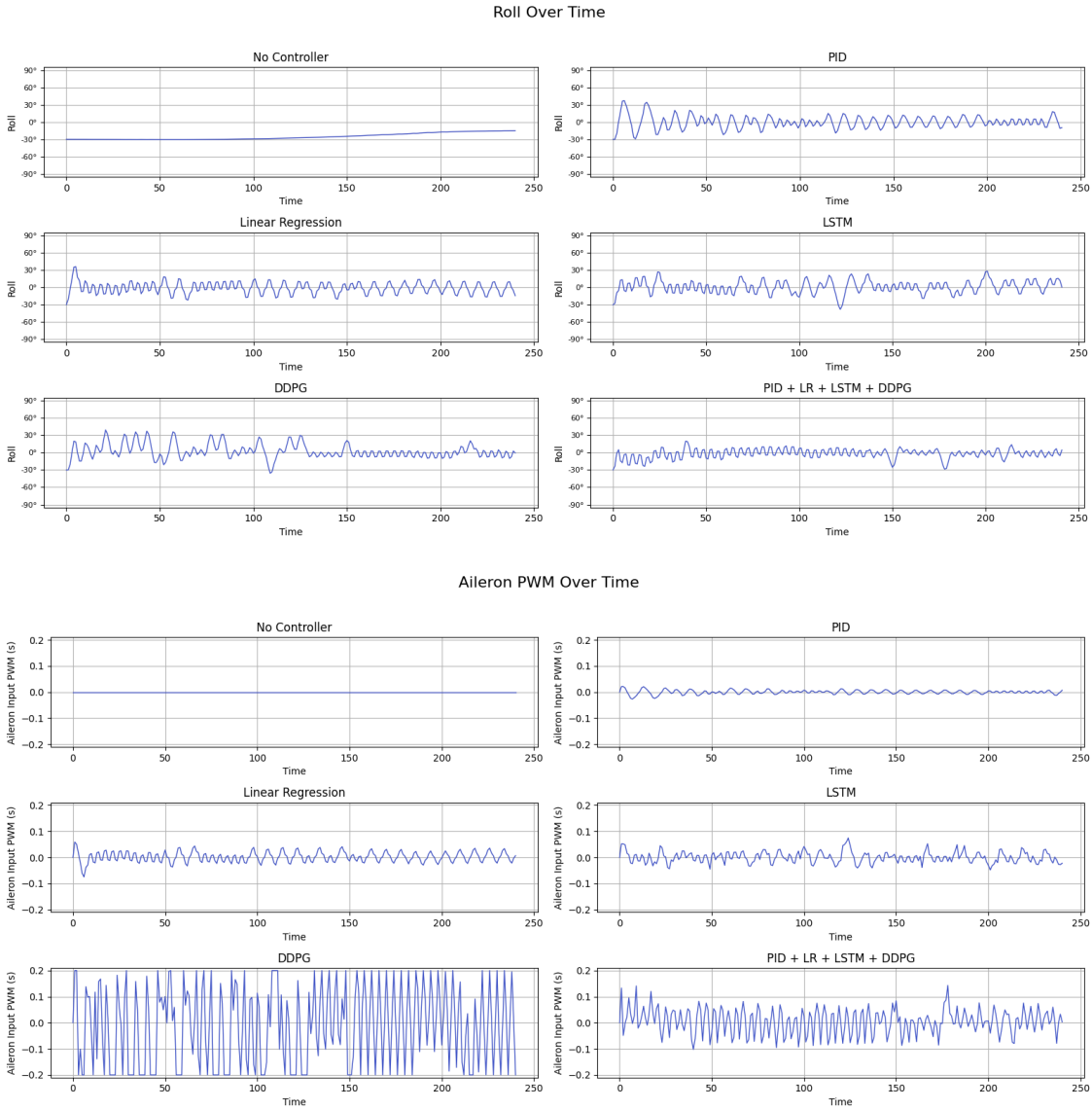
görevleriyle değerlendirilmiştir. Bu değerlendirme, kontrolörlerin gerçek görev koşullarındaki etkinliğini test etmeyi amaçlamaktadır.
Simülasyonda kullanılan başlangıç koşulları şu şekildedir:

- 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 saldırı görevlerini temsil eden Simülasyon 2 ve Simülasyon 3'ün sonuçları sırasıyla Tablo 3.2 ve Tablo 3.3'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



Şekil 3.3. Roll açıların ve aileron girdi karşılaştırılması

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 trajektori sonuçları ile roll açısı ve aileron girişlerine ilişkin grafikler Ekler bölümünde Şekil 4.1-4.6'te sunulmuştur.

Tablo 3.*2.* 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 3.3. 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

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. Simülasyon sonuçlarını gösteren video bağlantısı Ekler bölümünde sunulacaktır.

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 ajanı 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.

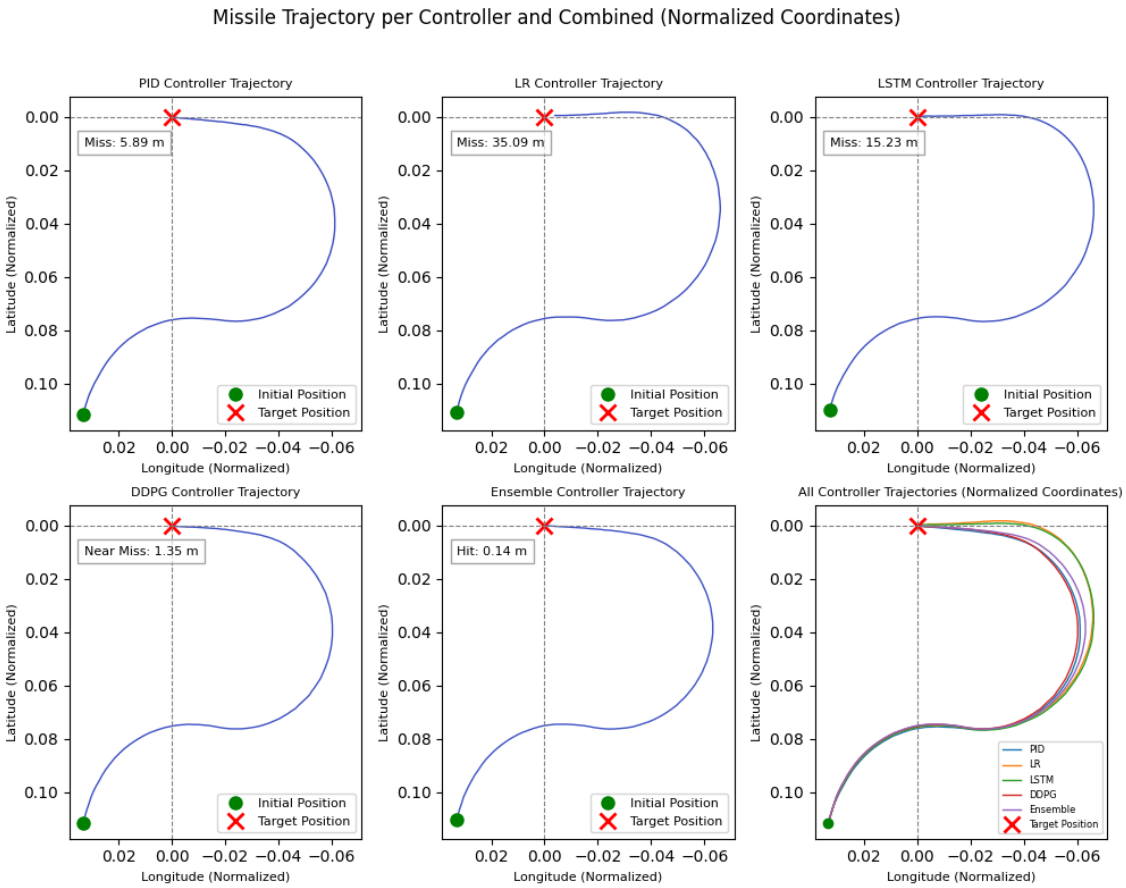
Son olarak, uçuş rotası planlama ve terminal yaw açısını belirlemeye yönelik stratejiler de geliştirilmiştir. Farklı yaw açıları (0°, 90° ve 180°) doğrultusunda hedefi vuracak uçuş rotasını belirleyebilen bir algoritma tasarlanmış ve test edilmiştir. Füzenin istenilen ara noktalara yönlendirilmesi için Bézier eğrisi formülü kullanılarak, daha düzgün ve optimal bir uçuş rotası elde edilmesi sağlanmıştır.

İHA simülasyonları tarafında, hareketlerin 2D grafiklerle modellenmesi, otonom çoklu İHA sistemlerinin karmaşık görev profillerini başarıyla yerine getirme ve çeşitli çevresel kısıtlamalara dinamik olarak tepki verme kabiliyetini güçlü bir şekilde ortaya koymaktadır. Bu modelleme, İHA'ların lider-takipçi formasyonlarında nasıl organize olduğunu, belirlenen hedeflere nasıl ulaştığını ve uçuşa yasak bölgelerden nasıl kaçındığını görsel olarak sunmaktadır. Elde edilen sonuçlar, temel kontrol algoritmalarının etkinliğini ve farklı operasyonel senaryolara adaptasyon yeteneğini göstermektedir. Mevcut çalışma, sistemin kabiliyetlerini daha da artırmak, yeni özellikler entegre etmek ve daha zorlayıcı senaryolarda

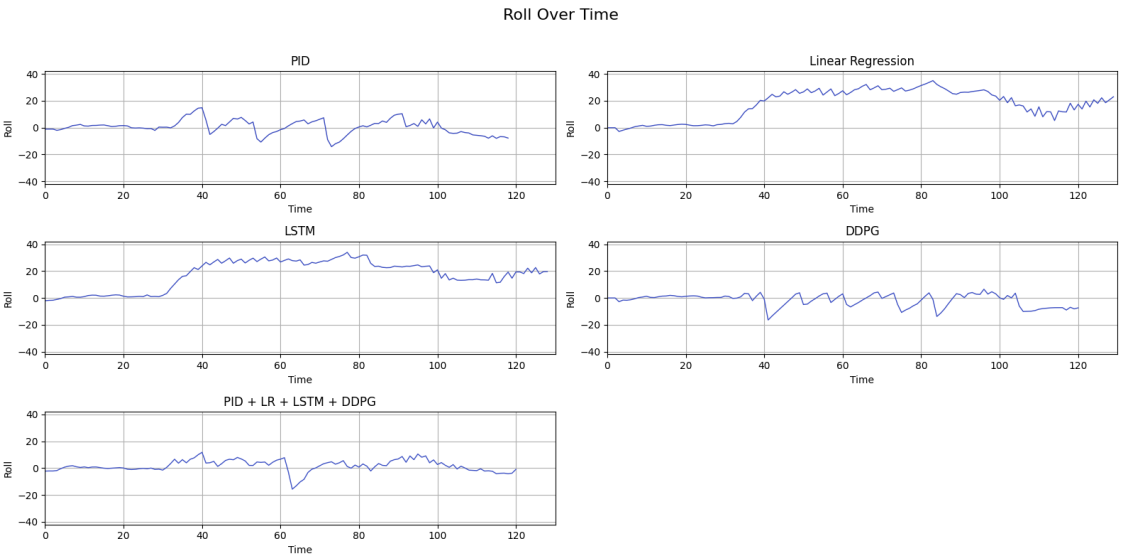
performansını optimize etmek amacıyla ilerleyen, kapsamlı bir araştırma ve geliştirme sürecinin belirgin bir aşamasını oluşturmaktadır.

Geliştirilen tüm sistemlerin nihai aşamada, önceden tanımlanmış senaryolar kapsamında kapsamlı testleri gerçekleştirilecektir. Ayrıca, geliştirilen çeşitli algoritmalardan elde edilen verilerin toplanması ve analizi yapılarak daha doğru ve güvenilir sonuçlara ulaşılması ve çalışmanın genel performansının değerlendirilmesi hedeflenmektedir. Bununla birlikte, rapor yazım süreci de devam ettirilerek daha da geliştirilecektir. Ayrıca, *Füze Roll Stabilizasyonu için Klasik ve Yapay Zeka Tabanlı Kontrol Stratejileri* konusuna odaklanan bir bilimsel makale hazırlanacak ve uluslararası hakemli bir dergiye gönderilmesi hedeflenmektedir.

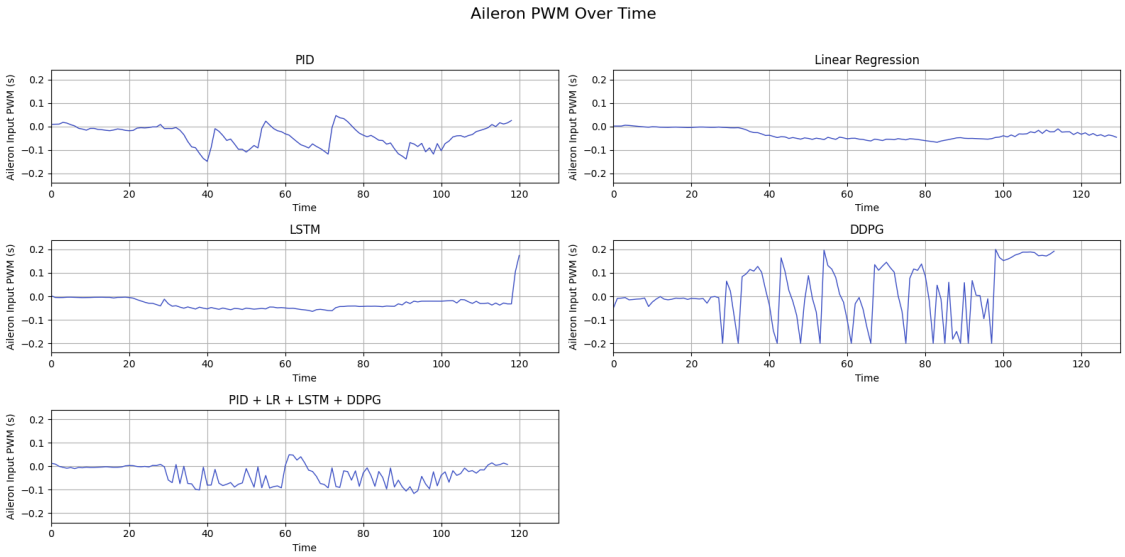
4.1. Ekler



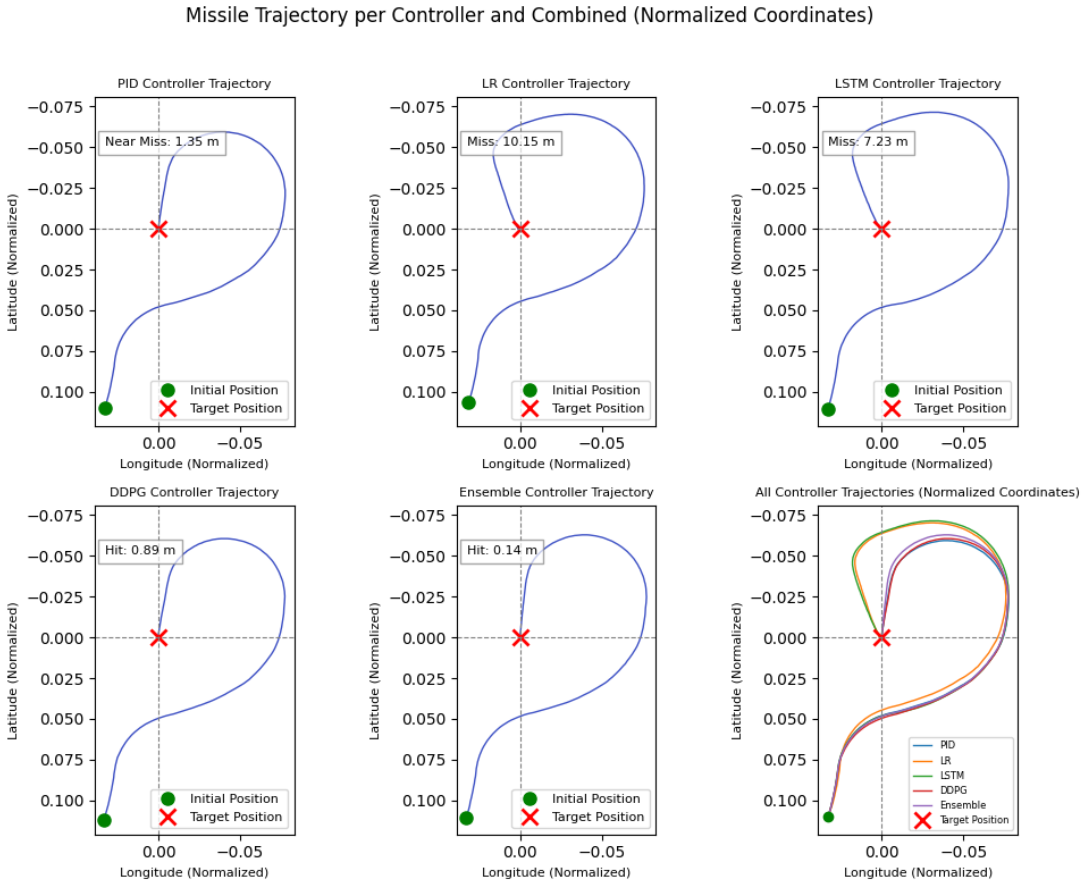
Şekil 4.1. Senaryo 2 (terminal yaw 90 derece) uçuş yörüngelerin karşılaştırılması



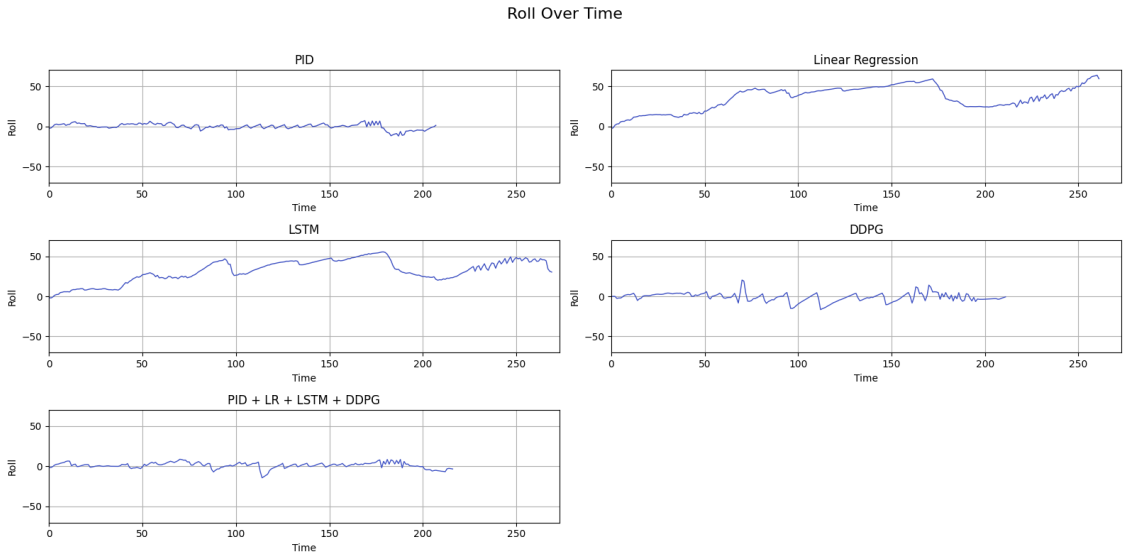
Şekil 4.2. Senaryo 2 (terminal yaw 90 derece) roll açıların karşılaştırılması



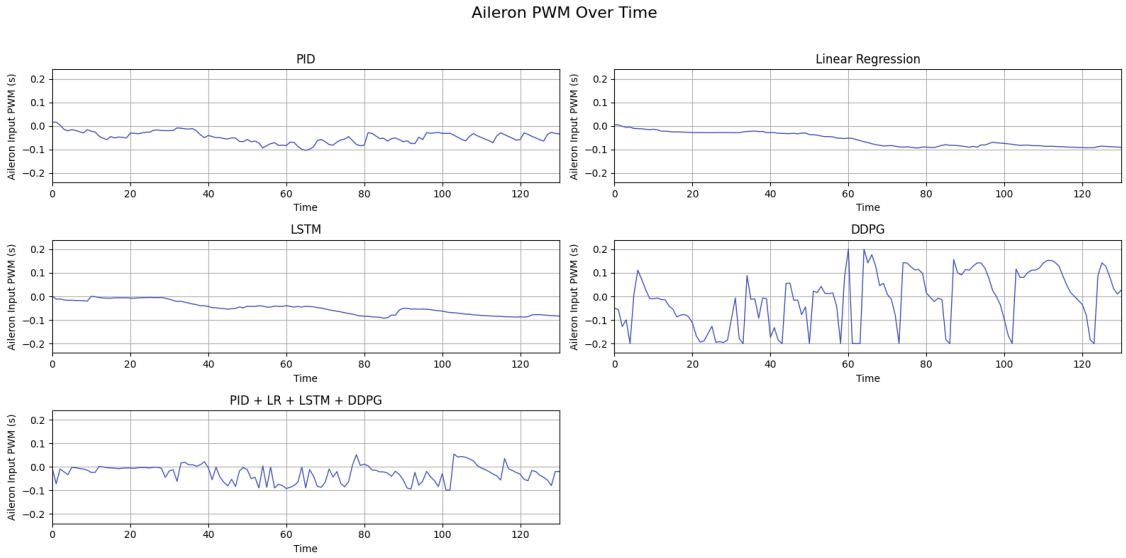
Şekil 4.3. Senaryo 2 (terminal yaw 90 derece) aileron girdi karşılaştırılması



Şekil 4.4. Senaryo 3 (terminal yaw 180 derece) uçuş yörüngelerin karşılaştırılması



Şekil 4.5. Senaryo 3 (terminal yaw 180 derece) roll açıların karşılaştırılması



Şekil 4.6. Senaryo 3 (terminal yaw 180 derece) aileron girdi karşılaştırılması

5. Kaynaklar

[1] A. Ryan, M. Zennaro, A. Howell, R. Sengupta ve J. K. Hedrick, «An Overview of Emerging Results in Cooperative UAV Control,» %1 içinde *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. [Çevrimiçi]. 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 ve J. P. How, «Multi-Task Allocation and Path Planning for Cooperating UAVs,» %1 içinde *Cooperative Control: Models, Applications and Algorithms*, Kluwer Academic Publishers, 2003, p. 23–41.

[4] S. Biswas, S. G. Anavatti ve M. A. Garratt, «Path planning and task assignment for multiple UAVs in dynamic environments,» %1 içinde *Unmanned Aerial Systems: Theoretical Foundation and Applications*, 2021, pp. 81-102.

[5] T. Huang, Y. Wang, X. Cao ve D. Xu, «Multi-UAV Mission Planning Method,» %1 içinde *3rd International Conference on Unmanned Systems (ICUS)*, Harbin, China, 2020.

[6] X.-p. Xu, X.-t. Y. W.-y. Yang, K. An, W. Huang ve 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 ve S. Bengio, «Neural Combinatorial Optimization with Reinforcement Learning,» November 2016. [Çevrimiçi]. Available: https://arxiv.org/abs/1611.09940.

[8] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina ve L. Song, «Learning Combinatorial Optimization Algorithms over Graphs,» April 2017. [Çevrimiçi]. Available: https://arxiv.org/abs/1704.01665.

[9] M. Nazari, A. Oroojlooy, L. V. Snyder ve M. Takáč, «Reinforcement Learning for Solving the Vehicle Routing Problem,» %1 içinde *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, Montréal, 2018.

[10] H. Lu, X. Zhang ve S. Yang, «A Learning-based Iterative Method for Solving Vehicle Routing Problems,» 2019. [Çevrimiçi]. Available: https://openreview.net/forum?id=BJe1334YDH.

[11] Y. Kang, Z. Pu, Z. Liu, G. Li, R. Niu ve J. Yi, «Air-to-Air Combat Tactical Decision Method Based on SIRMs Fuzzy Logic and Improved Genetic Algorithm,» %1 içinde *Lecture Notes in Electrical Engineering*, cilt 644, Springer, 2021.

[12] J. Carter, K. Schmid, K. Waters, L. Betzhold, B. Hadley, R. Mataosky ve J. Halleran, Lidar 101: An Introduction to Lidar Technology, Data, and Applications, NOAA Coastal Services Center, 2012.

[13] J. Pestana, J. L. Sanchez-Lopez, P. Campoy ve S. Saripalli, «Vision Based GPS-denied Object Tracking and Following for Unmanned Aerial Vehicles,» %1 içinde *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Sweden, 2013.

[14] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos ve 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 ve V. P. S. Naidu, «Survey on UAV navigation in GPS denied environments,» %1 içinde *International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, Paralakhemundi, 2016.

[16] Wikimedia, «Wikimedia Commons,» 19 August 2024. [Çevrimiçi]. Available: https://commons.wikimedia.org/wiki/File:AIM-54_Phoenix_cropped.jpg. [Erişildi: 15 March 2025].

[17] NoLuja, «Juno - AIM-54C Phoenix (Fully working air to air missile),» 2022. [Çevrimiçi]. Available: https://www.simplerockets.com/c/9d732I/AIM-54C-Phoenix-Fully-working-air-to-air-missile. [Erişildi: November 2024].

[18] «Juno: New Origins,» [Çevrimiçi]. Available: https://www.simplerockets.com/.

[19] WeaponSystems.net, «WeaponSystems.net AIM-54 Phoenix,» [Çevrimiçi]. Available: https://weaponsystems.net/system/219-AIM-54+Phoenix.

[20] NoLuja, «JUNO,» [Çevrimiçi]. Available: https://www.simplerockets.com/c/9d732I/AIM-54C-Phoenix-Fully-working-air-to-air-missile.

[21] WNP78, «SR2Logger,» 2020. [Çevrimiçi]. Available: https://github.com/WNP78/SR2Logger.

[22] G. Venkataraman, «GitHub - pyKey,» 2019. [Çevrimiçi]. Available: https://github.com/gauthsvenkat/pyKey.

[23] G. Jocher, A. Chaurasia ve J. Qiu, «Ultralytics YOLOv8,» 2023. [Çevrimiçi]. Available: https://github.com/ultralytics/ultralytics.

[24] J. Terven, D.-M. Córdova-Esparza ve J.-A. Romero-González, «A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS,» *Mach. Learn. Knowl. Extr.*, cilt 5, pp. 1680-1716, 2023.

[25] B. Dwyer, J. Nelson ve T. Hansen, «Roboflow (Version 1.0),» 2024. [Çevrimiçi]. Available: https://roboflow.com.

[26] I. Loshchilov ve F. Hutter, «Decoupled Weight Decay Regularization,» %1 içinde *7th International Conference on Learning Representations*, New Orleans, 2019.

[27] A. Ng ve T. Ma, CS229 Lecture Notes, 2023.

[28] S. Ioffe ve C. Szegedy, «Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,» February 2015. [Çevrimiçi]. Available: https://arxiv.org/abs/1502.03167. [Erişildi: April 2025].

[29] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver ve D. Wierstra, «Continuous control with deep reinforcement learning,» September 2015. [Çevrimiçi]. Available: https://arxiv.org/abs/1509.02971. [Erişildi: November 2024].

[30] A. Ryan, M. Zennaro, A. Howell, R. Sengupta ve J. K. Hedrick, «An Overview of Emerging Results in Cooperative UAV Control,» %1 içinde *43rd IEEE Conference on Decision and Control (CDC)*, Nassau, 2004.

[31] 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,» August 2023. [Çevrimiçi]. Available: https://www.armyupress.army.mil/Journals/Military-Review/English-Edition-Archives/March-2024/Bayraktars-and-Grenade-Dropping-Quadcopters/.

[32] 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.

[33] S. Biswas, S. G. Anavatti ve M. A. Garratt, «Path planning and task assignment for multiple UAVs in dynamic environments,» %1 içinde *Unmanned Aerial Systems: Theoretical Foundation and Applications*, 2021, pp. 81-102.

[34] T. Huang, Y. Wang, X. Cao ve D. Xu, «Multi-UAV Mission Planning Method,» %1 içinde *3rd International Conference on Unmanned Systems (ICUS)*, Harbin, China, 2020.