Teknoloji Fakültesi

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ 1. 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 LİSTESİ……4
ŞEKİL LİSTESİ…….5
TABLO LİSTESİ……...6
ÖZET……..7
BÖLÜM 1. GİRİŞ…..……..9
1.1. Bitirme Projesinin Amacı…….....9
1.1.1. Çoklu İHA Stratejileri……...9
1.1.2. Füze Kontrol ve Navigasyon Sistemleri Stratejileri…….10
1.2. Literatür Özeti……...…….11
BÖLÜM 2. MATERYAL VE YÖNTEM ……...13
2.1. Materyal…….13
2.1.1. Simülasyon Ortamı – Simplerockets 2 (Juno: New Origins)..….....13
2.1.2. Vizzy Programlama (IDE)……...……14
2.1.3. Python…….……15
2.1.4. SR2Logger Mod…..……15
2.1.5. MATLAB……..……...…….....16
2.1.6. Unity……..……...…….....16
2.2. Yöntem……..……17
2.2.1. Veri İletişimi ……..17
2.2.2. Çoklu İHA Koordinasyonu ve Otonom Görev Ataması …….20
2.2.3. Python Üzerinden Füze Kontrolü …….……..22
2.2.4. Füze Navigasyonu ve Kontrolü ….…….23
2.2.4.1. Terminal Yaw Açısı Kontrolü ……..……..24
2.2.4.2. Terminal Pitch Açısı Kontrolü ……...…….…….26
2.2.4.3. Hedef Tespiti için YOLOv8 Modelinin Eğitimi ……..28
2.2.4.4. Anti-Roll Stabilizasyon Sistemi …….……29
BÖLÜM 3. BULGULAR VE TARTIŞMA……..30
3.1. Ekler…....…….…….....30
KAYNAKLAR…….31

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

YZ: Yapay Zeka

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1. AIM-54C by Hughes Aircraft 14
Şekil 2.2. AIM-54C Phoenix by NoLuja.....14
Şekil 2.3. SR2Logger kullanarak UDP üzerinden veri göndermek için Vizzy programı ...18
Şekil 2.4. Python'dan SR2'ye füze kontrol akış şeması.....19
Şekil 2.5. Python'dan Unity'ye veri gönderim kodu.....20
Şekil 2.6. Unity'den SR2'ye veri gönderim kodu.....23
Şekil 2.7. İstenilen terminal yaw açıları ve uçuş yörüngesi 25
Şekil 2.8. Python'da Bezier eğrisi kodu ve görselleştirilmesi 26
Şekil 2.9. Hedef görsellerin ve sınırlayıcı kutularının örnekleri 28

TABLO LİSTESİ

Sayfa

Tablo 2.1. Kullanılan Python kütüphaneleri.....15

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 ortamında 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

Gittikçe daha sofistike hale gelen ve birçok ülkenin envanterine girmeye başlayan Kara Tabanlı Hava Savunma Sistemleri (GBAD), hava yoluyla yapılan saldırı görevlerini önemli ölçüde zorlaştırmaktadır. Yüksek teknolojiyle donatılmış ve sürekli geliştirilen Yüzeyden Havaya Füze (SAM) sistemleri, saldırı misyonları için gönderilen savaş uçağı pilotları açısından ciddi bir tehdit oluşturmaktadır. Ayrıca, bu tür operasyonlarda fırlatılan füzeler, GBAD sistemleri tarafından korunan bölgelere nüfuz etmekte zorlanmaktadır.

Bu sorunları aşmak amacıyla, GBAD tarafından korunan düşman bölgelerini hedef alan yüksek riskli görevlerde insansız hava araçlarının (İHA) ve özellikle sürü drone (swarm drone) yapılarının kullanımı hayati önem taşımaktadır. Bununla birlikte, füze kontrol ve navigasyon sistemlerinin geliştirilmesi, düşman savunmalarını aşma şansını artırmak ve hedef vurma hassasiyetini yükseltmek için gereklidir.

Bu bölümde, çalışmanın iki temel odak alanına ayrılan amaçları açıklanmaktadır: sürü drone stratejilerinin geliştirilmesi ve füze kontrol ile navigasyon sistemlerinin iyileştirilmesi. Ayrıca, bu çalışmada ele alınan sorunlara ilişkin güncel eğilimleri değerlendirmek amacıyla kapsamlı bir literatür taraması gerçekleştirilmiştir.

1.1. Bitirme Projesinin Amacı

Bu bölümde çalışmanın amaçları, çoklu İHA stratejileri ve füze kontrol ve navigasyon sistemleri stratejileri olmak üzere iki odak noktasına bölünerek sunulmaktadır.

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, düşmanın hava savunma sistemine yönelik bir saldırı için uçtan uca görev planlaması sağlamayı amaçlamaktadır. Bu, otonom çoklu İHA rota planlaması ve görev tahsisi ile çoklu füze iş birliği kontrolü ve navigasyonu gibi birkaç modülü içermektedir. Önceki çalışmaların çoğunlukla daha geniş zorluklara odaklanmasının aksine, bu görev, belirli bir gerçek dünya problemini ele almaktadır. Proje, yeni akıllı sistemler ve algoritmaların kullanılmasına vurgu yaparak bunların gerçek bir savaş senaryosundaki etkinliğini göstermeyi amaçlamaktadır. Ek olarak, dinamik ortamlarda çoklu İHA sistemlerini koordine etmek için otonom ve uyarlanabilir algoritmaların kullanılması sağlanacaktır ki bu konu, önceki çalışmalarda sınırlı ilgi görmüştür. Bununla birlikte, bu çalışma, İHA'lar için görev planlaması ve hedef belirleme üzerine odaklanırken, hava koşulları veya iletişim kesintileri gibi çevresel değişkenleri dikkate almamakta olup bu alanlarda gelecekteki araştırmalar için bir fırsat bırakmaktadır.

2. MATERYAL VE YÖNTEM

Bu çalışma, Materyaller ve Yöntemler olmak üzere iki ana başlık altında sunulmuştur.

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.

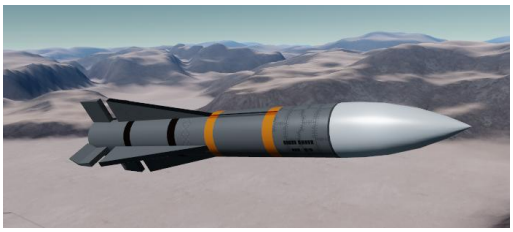
Bununla birlikte, proje süresince kullanılan Python programlama dili ve ilgili kütüphaneler ayrıntılı olarak sunulmuştur. Projede; yapay zeka ve DDPG algoritması için TensorFlow, doğrusal regresyon uygulamaları için Scikit-learn, görselleştirme işlemleri için Matplotlib, veri işleme için Pandas ve NumPy, 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.

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

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

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

Şekil 2.1. AIM-54C by Hughes Aircraft [19]



Şekil 2.2. AIM-54C Phoenix by NoLuja [20]

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.

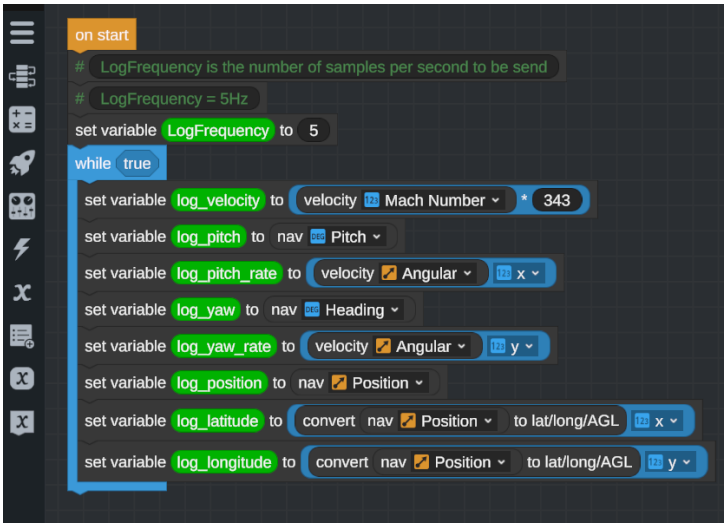
Son olarak, YOLO nesne tespit algoritmasının kullanımı ele alınmış ve bu algoritmanın füzenin hedefe yönlendirilmesinde nasıl bir rol oynadığı ortaya konulmuştur.

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.

```
1  import socket
2  import time
3
4  def send_data():
5      udp_ip = "127.0.0.1"
6      udp_port = 5005
7      sayac = 0
8
9      sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
10
11     try:
12         while True:
13             message = f"Veri !{sayac}"
14
15             sock.sendto(message.encode(), (udp_ip, udp_port))
16             print(f"'{message}' {udp_ip}:{udp_port} adresine gönderildi. {sayac}")
17             sayac+=1
18             time.sleep(1)
19     except Exception as e:
20         print(f"Hata: {e}")
21     finally:
22         sock.close()
23
24 if __name__ == "__main__":
25     send_data()
26
```

Şekil 2.4. Port aracılığıyla veri göndermek için örnek Python kodu

```
10  public class UdpListener : MonoBehaviour
        2 references
12      public int listenPort = 5005;
        4 references
13      private Thread listenerThread;
        4 references
14      private UdpClient udpClient;
15
        0 references
16  >   void Start() ···
24
        1 reference
25      private void StartListening()
26      {
27          udpClient = new UdpClient(listenPort);
28          IPEndPoint remoteEndPoint = new IPEndPoint(IPAddress.Any, listenPort);
29          try
30          {
31              while (true)
32              {
33                  byte[] receivedBytes = udpClient.Receive(ref remoteEndPoint);
34                  string receivedData = Encoding.UTF8.GetString(receivedBytes);
35
36                  UnityMainThreadDispatcher.Instance().Enqueue(() =>
37                  {
38                      Debug.Log($"[{remoteEndPoint}] Gelen veri: {receivedData}");
39                  });
40              }
41          }
42          catch (Exception e)
43          {
44              Debug.LogError($"Hata: {e.Message}");
45          }
46          finally
47          {
48              udpClient.Close();
49          }
50      }
        0 references
51  >   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. Python ile geliştirilen simülasyon, nesne yönelimli programlama (OOP) yaklaşımı ile tasarlanmıştır. Bu yaklaşım sayesinde, her bir İHA bağımsız bir nesne olarak modellenmiş ve bu nesnelerin birbirleriyle olan etkileşimleri kolaylaştırılmıştır. Python simülasyonunda, İHA'ların hedefe ulaşmak için kullanacağı rotalar otomatik olarak hesaplanmakta, bu hesaplamalarda İHA'ların hızları, hedefe olan mesafeleri ve manevra yetenekleri gibi faktörler göz önünde bulundurulmaktadır. Ayrıca, İHA'ların hedefe ulaşma sürelerini değerlendiren bir fonksiyon geliştirilmiştir. Bu fonksiyon aracılığıyla, hangi İHA'nın hedefe en kısa sürede ulaşabileceği belirlenmekte ve görev atama süreçleri örnek teşkil etmesi amacıyla gerçekleştirilmektedir.

Formasyon kontrolü ve koordinasyon konularında çeşitli yaklaşımlar incelenmiştir. Bu kapsamda, temel olarak kullanılan yöntemler arasında Lider-Takipçi Yöntemleri, Potansiyel Alan Tabanlı Yöntemler, Konsensus Algoritmaları, Parçacık Sürü Optimizasyonu (PSO) ve Model Öngörücü Kontrol (MPC) gibi teknikler yer almaktadır. Özellikle Lider-Takipçi Yöntemleri, bir lider İHA'nın diğer İHA'lara göre konumunu belirleyerek formasyon kontrolünü sağlaması açısından yaygın olarak kullanılmaktadır. Potansiyel Alan Yöntemleri İHA'ların birbirleriyle çarpışmasını önlerken hedefe yönelmesini sağlayan kuvvet temelli bir yaklaşım sunmaktadır. Konsensus Algoritmaları, dağıtık kontrol yapıları için etkili bir yöntem olup, İHA'ların ortak bir hedefe ulaşmak üzere uyumlu bir şekilde hareket etmesini sağlamaktadır. PSO ise, birden fazla İHA'nın kolektif bir şekilde en uygun rotayı bulmasını sağlayan bir yöntemdir. Bu yöntemde her bir İHA bir parçacık olarak değerlendirilmekte ve bu parçacıklar, toplu hareket ederek en iyi çözümü bulmaya çalışmaktadır. PSO'nun avantajı, basit uygulanabilirliği ve düşük hesaplama maliyeti ile geniş bir arama alanında etkili sonuçlar üretebilmesidir. Diğer taraftan MPC

yönteminde, gelecekteki hareketlerin tahmin edilerek en uygun rotaların belirlenmesini mümkün kılmaktadır.

İlerleyen aşamalarda, bu yöntemlerin uygunlukları değerlendirilecek ve simülasyon ortamına entegre edilerek İHA'ların koordinasyonunu sağlamak amacıyla kullanılacaktır. Ayrıca, farklı algoritmaların karşılaştırılması ve performans analizlerinin yapılması da projenin amaçları arasındadır.

2.2.3. Python üzerinden füze kontrolü

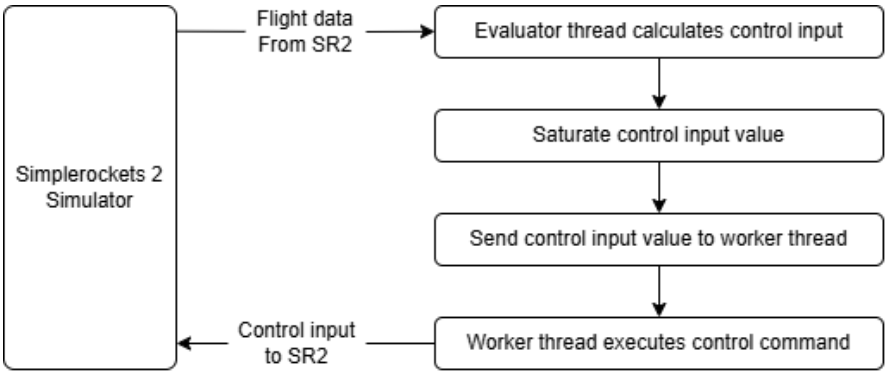
Bu projede, füze kontrolünü sağlamak amacıyla Python programı içerisinde bir kontrol mekanizması geliştirilmiştir. Bu mekanizma, füzenin elevator, rudder ve aileron yüzeylerinin dışarıdan kontrol edilmesini kapsamaktadır. Füzenin kontrol yüzeylerini dışarıdan yönetebilmek için PyKey kütüphanesi kullanılmaktadır [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.



Şekil 2.6. Python'dan SR2'ye füze kontrol akış şeması

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.

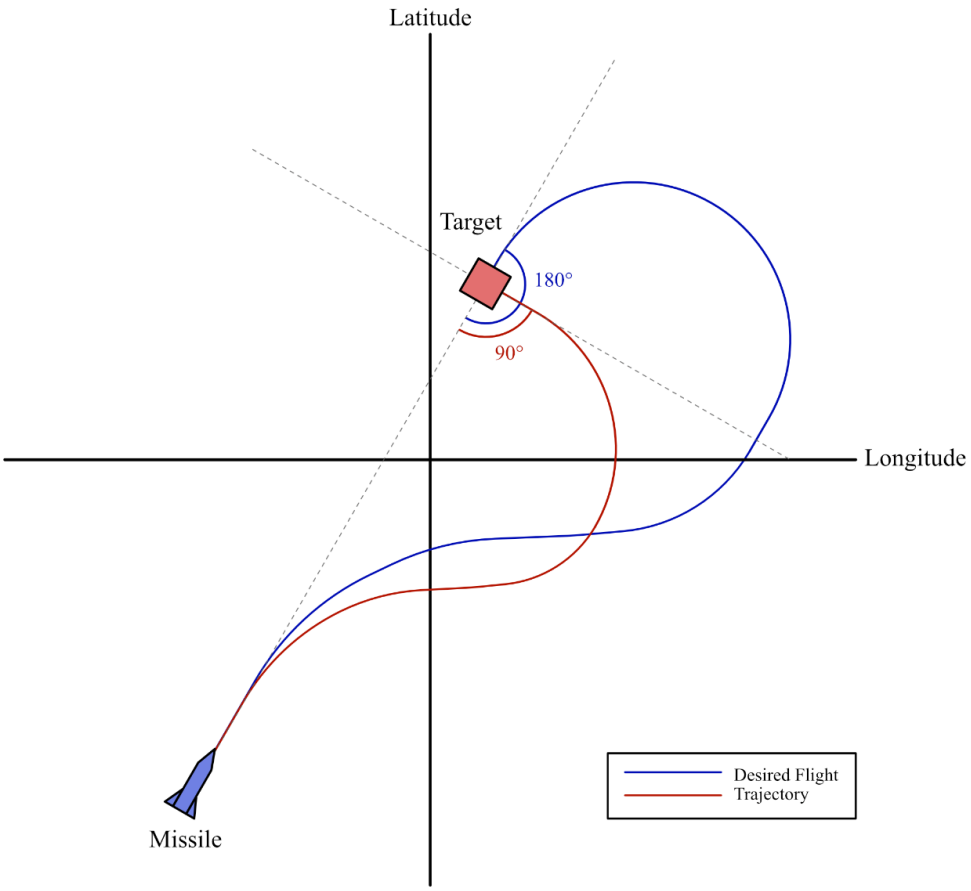
2.2.4.1. Terminal yaw açısı kontrolü

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 2.5'te terminal yaw açıları görselleştirilmektedir.



Şekil 2.7. İstenilen terminal yaw açıları ve uçuş yörüngesi

Şekil 2.5'te 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 aşağıda verilmiştir:

FORMÜL DAHA SONRA EKLENECEKTİR.

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

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

$$B(t) = (1-t)3P0 + 3(1-t)2tP1 + 3(1-t)t2P2 + t3P3, \quad t \in [0,1] \tag{2.1}$$

Burada:

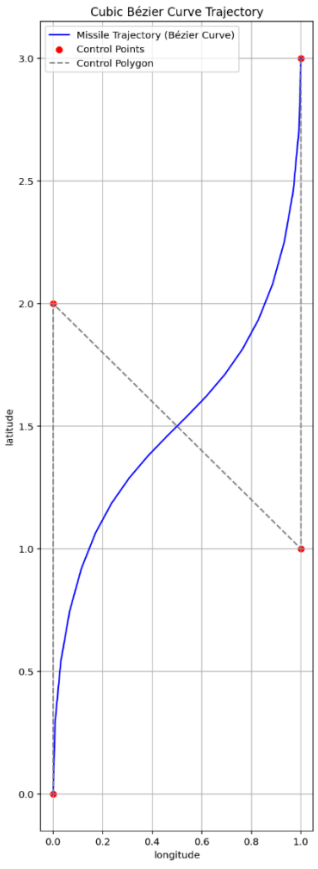
- P0, P1, P2, P3 kontrol noktalarını (her biri x ve y koordinatlarıyla) ifade etmektedir.
- t, eğri parametresidir (0 ile 1 arasında değişmektedir).
- $B(t) = \big(x(t), y(t)\big)$, t anındaki eğri üzerindeki bir noktanın koordinatlarını vermektedir.

```
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 2.8. 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.

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

$$a_n = N \cdot V_c \cdot \dot{\lambda} \qquad (2.2)$$

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.

Şekil 2.9. Hedef görsellerin ve sınırlayıcı kutularının örnekleri

b.) 2. 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: 0.001
- Batch size: 16
- Görüntü boyutu: 800x800
- Optimizasyon algoritması: AdamW [26]
- Epochs sayısı: 5 (önceden eğitilmiş model ile eğitildiği için az sayıda tekrar eğitilmiştir)

c.) 3. Füze navigasyon sistemine entegrasyon

Eğitilen YOLOv8 modeli, füze navigasyon sistemine entegre edilmiştir. Füze hedefe yaklaşırken ve hedefe yönelirken, kamera YOLOv8 algoritmasını aktive ederek gerçek zamanlı hedef tespiti yapmaktadır. Tespit edilen hedefin konumu (bounding box merkez noktası), füzenin pitch ve yaw açılarını dinamik olarak ayarlamak için bir geri besleme döngüsü olarak kullanılır ve böylece füze hedefe daha hassas bir şekilde yönlendirilmektedir.

YOLOv8 kullanımı sayesinde füze yalnızca IMU ve GPS sensörlerine bağlı kalmaz, aynı zamanda tespit edilen görsel bilgiler doğrultusunda dinamik yol düzeltmeleri de gerçekleştirebilmektedir.

2.2.4.4. Anti-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.

3. 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 amacıyla MATLAB ortamında gerçekleştirilmiştir. Bu simülasyonda, İHA'lar başlangıç noktasından hedefe

doğru hareket ederken, koordineli etkileşim için geçiş noktaları (waypoints) kullanılmıştır. İHA hareketleri, interpolasyon teknikleriyle daha gerçekçi hale getirilmiş ve doğal görünümler sağlanmıştır. Bu çalışma, İHA'ların hedefe ulaşma şekli, mesafeleri ve hızlarını dikkate alarak ileri simülasyon sürecine geçiş için bir temel oluşturmuştur.

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.

3.1. Ekler

Daha sonra eklenecektir.

4. Kaynaklar

[1] A. Ryan, M. Zennaro, A. Howell, R. Sengupta ve J. K. Hedrick, "An Overview of Emerging Results in Cooperative UAV Control," 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," 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," 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," 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," 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," 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," 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," içinde *International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, Paralakhemundi, 2016.

[16] "Juno: New Origins," [Çevrimiçi]. Available: https://www.simplerockets.com/.

[17] WeaponSystems.net, "WeaponSystems.net AIM-54 Phoenix," [Çevrimiçi]. Available: https://weaponsystems.net/system/219-AIM-54+Phoenix.

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

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

[20] 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].

[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," içinde *7th International Conference on Learning Representations*, New Orleans, 2019.

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

[28] 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/.

[29] 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.

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

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