Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü
2024/2025 Eğitim-Öğretim Yılı Bitirme Projesi Sunumu

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

Hazırlayanlar: *Ammar ABDURRAUF, Ahmet KURT, Serdar YILDIRIM*

Danışman: *Dr. Öğr. Üyesi Eyüp Emre ÜLKÜ*

Özet

• Modern hava savunma sistemleri, geleneksel hava saldırılarını etkisiz hale getiriyor.

• Bu sistemler, insanlı hava araçları ve mühimmatlar için ciddi tehdit oluşturuyor.

• İnsansız ve otonom sistemler, bu tehdide karşı yeni bir çözüm olarak öne çıkıyor.

• Projede, çoklu İHA ve akıllı füzelerle koordineli saldırı stratejileri geliştirildi.

• Klasik kontrol algoritmaları, yapay zekâ tabanlı yöntemlerle entegre edildi.

• Amaç, füze ve İHA'ların kontrol, rota planlama ve hedefe yönelim kabiliyetini artırmak.

• Simülasyonlar, SR2 ve MATLAB/Python ortamlarında başarıyla test edildi.

• GPS olmayan senaryolara özel, nesne tanıma (YOLOv8) sistemi entegre edildi.

• Sonuç olarak, hava savunmasını aşabilen otonom bir saldırı sistemi geliştirildi.

Şekil 1. Hava savunma sistemi

Giriş

- Yer tabanlı hava savunma sistemleri, saldırı kabiliyetini ciddi ölçüde sınırlıyor.
- Geleneksel mühimmatlar ve insanlı hava araçları yüksek risk altında.
- Bu nedenle, otonom sistemlere dayalı stratejilere ihtiyaç artıyor.
- Projenin temel hedefi:

  İHA sürüleriyle koordineli hareket ve görev paylaşımı sağlamak,

  Füze sistemlerini zeki ve stabil hale getirmek.
- Pitch kontrolü için Proportional Navigation algoritması kullanıldı.
- Roll kararsızlığı, PID ve yapay zekâ kontrollü anti-roll sistemiyle giderildi.
- Yaw açısı için Bézier eğrisi ile rota planlaması yapıldı.
- İHA'larda görev paylaşımı, formasyon kontrolü ve iletişim geliştirildi.
- GPS olmayan ortamlarda çalışmak için YOLOv8 tabanlı hedef tanıma eklendi.
- Tüm bileşenler, SR2, MATLAB ve Python tabanlı simülasyonlarda test edildi.

Şekil 2. Sürü İHA

Materyal

- Simülasyonlar **Simplerockets 2 (SR2)** ortamında gerçekleştirildi.
- Füze ve sistem modelleri SR2 içinde oluşturuldu ve test edildi.
- **Vizzy** dili ile SR2 içinde kontrol algoritmaları kodlandı.
- **Python** programlama dili çeşitli görevlerde kullanıldı:
- Yapay zeka için: **TensorFlow**,
- Regresyon analizleri için: **Scikit-learn**,
- Görselleştirme için: **Matplotlib**,
- Veri işleme için: **Pandas** ve **NumPy**,
- Otomasyon için: **PyAutoGUI**, **PyKey**.
- SR2 ile Python arasında bağlantı için:
- **SR2Logger Modu** (WNPJ tarafından geliştirildi) kullanıldı.
- İHA'ların 2D hareket simülasyonları için:
- **MATLAB** ve Python ortamı tercih edildi.

Şekil 3. Kullanılan uygulama ve programlar

Yöntem - Genel Bakış

- Kontrol yüzeyleri için algoritmalar geliştirildi:
  - **Elevator**, **Rudder**, **Aileron**.
  - **Multithreading** ile gerçek zamanlı kontrol sağlandı.
- **Terminal Pitch** kontrolü:
  - **Proportional Navigation** yöntemi kullanıldı.
- **Terminal Yaw** kontrolü:
  - **Waypoint** tabanlı rota çizimi,
  - **Bézier eğrileri** ile yumuşak yönlendirme.
- **Roll kararsızlığı** için anti-roll sistem geliştirildi:
  - **PID**,
  - **Doğrusal Regresyon**,
  - **LSTM**,
  - **DDPG** algoritmalarını içeren **ansambl model**.
- **Model eğitim süreci** Python ortamında gerçekleştirildi.
- **Hedef tespiti** için:
  - **YOLOv8** nesne tanıma algoritması entegre edildi.

Çoklu İHA Koordinasyonu, Formasyon Kontrolü, Dinamik Görev Ataması Algoritmalarının Geliştirilmesi ve 2D Simülasyonu

---

Aşamalı Yaklaşım:

- **Temsili Rota İlerleyişinin Görselleştirilmesi:** Matlab ve Python ile çoklu İHA hareket planı, interpolasyon.

- **Tek İHA Uçuş Dinamikleri ve Temel Seyrüsefer:** İHA modeli, temel otopilot (konum, hız, yönelim, irtifa kontrolü).

- **Çoklu İHA Konsensüs Algoritmaları:** Lider-takipçi mimarisi, temel formasyon kontrolü.

- **Kapsamlı İHA Sürü Yönetimi ve Saldırı:** Dinamik görev planlama, tehdit tepkisi (NFZ, GBAD), lider arızası ve yeniden seçim.

Çoklu İHA Koordinasyonu, Formasyon Kontrolü, Dinamik Görev Ataması Algoritmalarının Geliştirilmesi ve 2D Simülasyonu

---

Materyal:

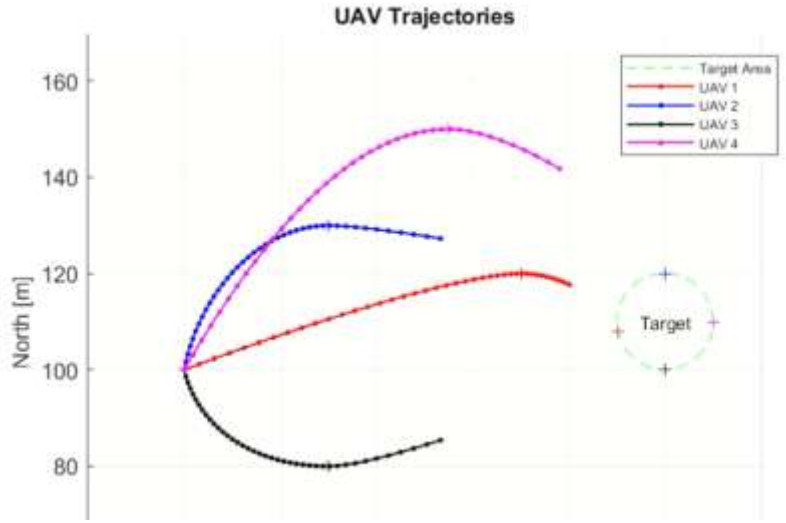
- **Matlab:** İHA hareketlerinin 2D simülasyonları için başlangıç aşaması.

- **Python 3.10:** Simülasyonların geliştirilmesi, test edilmesi.

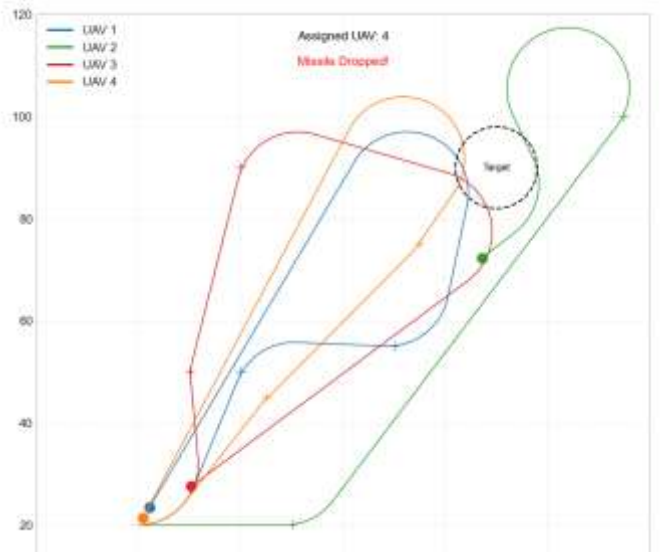
    - **Kullanılan Kütüphaneler**: NumPy, Matplotlib PyPlot, Matplotlib Animation, Matplotlib Patches

    - **Geliştirme Ortamı**: PyCharm IDE

Matlab ve Python Matplotlib ile Çoklu İHA Rota Takip Simülasyonu

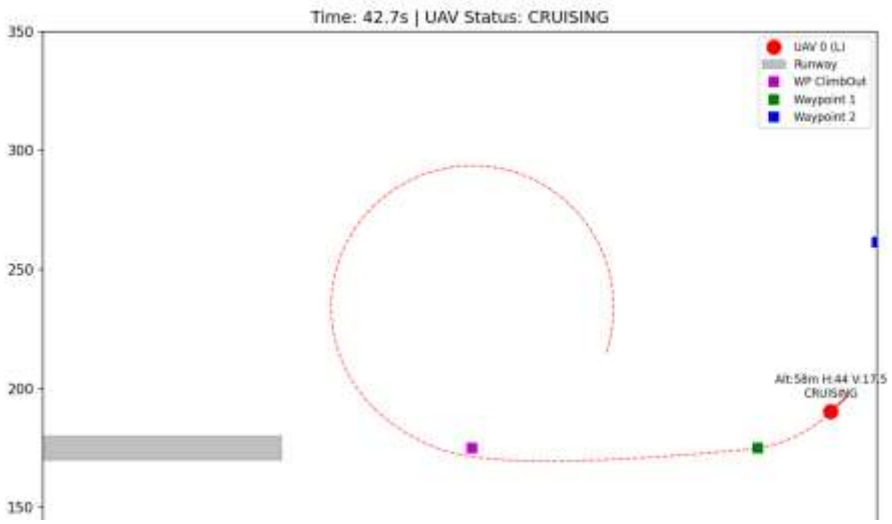


Şekil 4. Matlab ile temsili rota takip simülasyonu

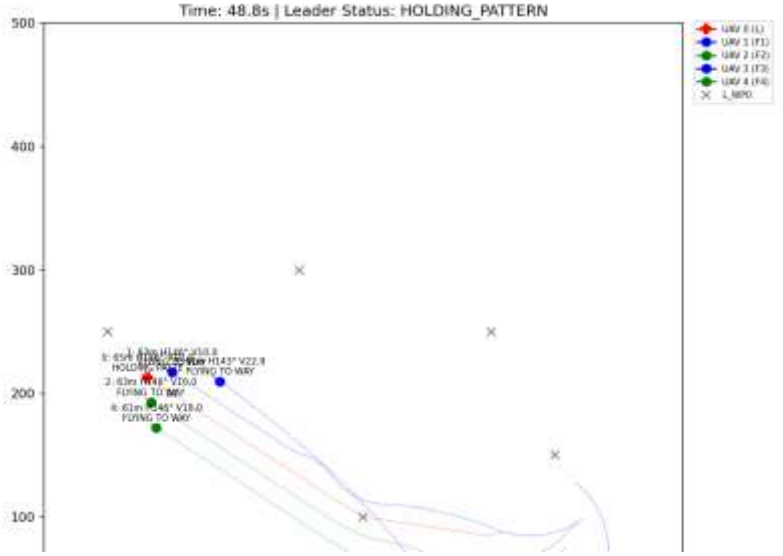


Şekil 5. Python Matplotlib ile temsili rota takip ve dönüş simülasyonu

Tek İHA ile Uçuş Testi ve Konsensüs Uygulaması Testleri



Şekil 6. Tek İHA ile kalkış ve nokta takip simülasyonu



Şekil 7. Çoklu İHA ile konsensüs algoritması testi

Temel İHA Dinamikleri ve Kontrol Algoritmaları

Yönelim Kontrolü: $\dot{\psi}_{cmd} = K_{P,\psi} \cdot (\psi_{des} - \psi)$

Hedef yön ile mevcut yön arasındaki farkı orantısal bir kazanç ile çarparak komut edilen dönüş oranını hesaplar.

Hız Kontrolü (Yatay): $a_{cmd} = K_{P,v} \cdot \left(v_{des} - v_{xy}\right)$

Hedef hız ile mevcut hız arasındaki farka dayalı olarak İHA'ya hızlanma veya yavaşlama komutu verir.

İrtifa Kontrolü (Dikey Hız): $\dot{h}_{cmd} = K_{P,h} \cdot (h_{des} - h)$

Hedef irtifa ile mevcut irtifa arasındaki farkı orantısal bir kazanç ile çarparak komut edilen dikey hızı (tırmanma/alçalma hızını) hesaplar.

Lider-Takipçi Formasyon Kontrolü

Lider hareket ettikçe, takipçinin dünya üzerindeki hedef konumunu dinamik olarak güncellemek hedeflenir. Tüm takipçi İHA'lar, liderin kendi koordinat sistemine göre belirlenmiş **benzersiz bir ofset** noktasına sahiptir.

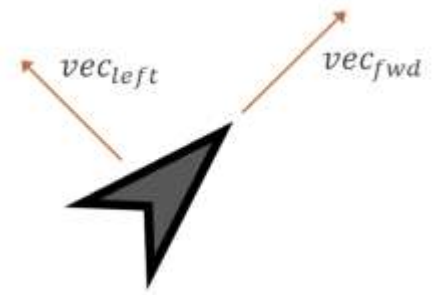
$$P_{follower}^{target} = P_{leader} + O_{along} \cdot vec_{fwd} + O_{across} \cdot vec_{left}$$

O_{along} : Takipçinin liderin ileri yönünde olması gereken mesafe (örn, liderin 15m arkası). (formation_offset_body[0])

O_{across} : Takipçinin liderin yan yönünde olması gereken mesafe (örn, liderin 15m solu). (formation_offset_body[1])

vec_{fwd} : Liderin o anki yönelimine göre hesaplanan ileri yön vektörü.

vec_{left} : Liderin o anki yönelimine göre hesaplanan sol yön vektörü.



Şekil 8. Lider İHA'ya ait yön vektörleri

Uçuşa Yasak Bölgeler (NFZ) ve GBAD Kaçınma

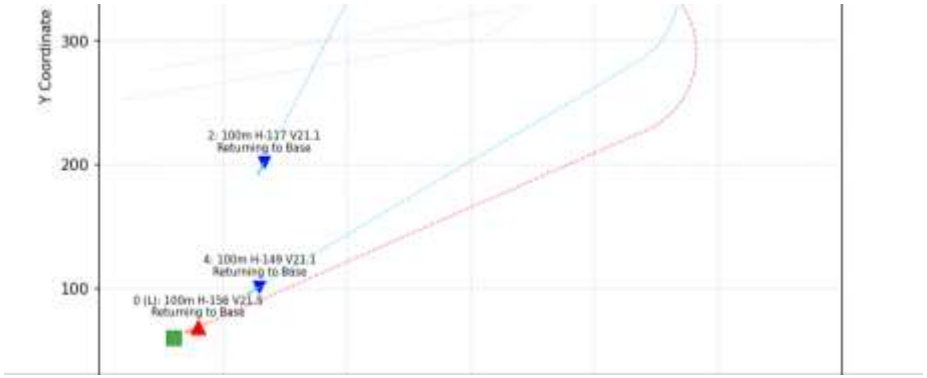
NFZ Algılama: İHA'lar, mevcut konumları ve bir sonraki kısa vadeli tahmini rotalarının NFZ'ler ile kesişip kesişmediğini sürekli kontrol ederek proaktif kaçınma sağlar.

Çarpışma Tespiti:
- Dairesel NFZ için: İHA'nın rotasının, NFZ merkezine olan en kısa mesafesinin NFZ yarıçapı + güvenlik marjından küçük olması: $d_{seg} \leq R_{\{NFZ\}} + R_{\{margin\}}$

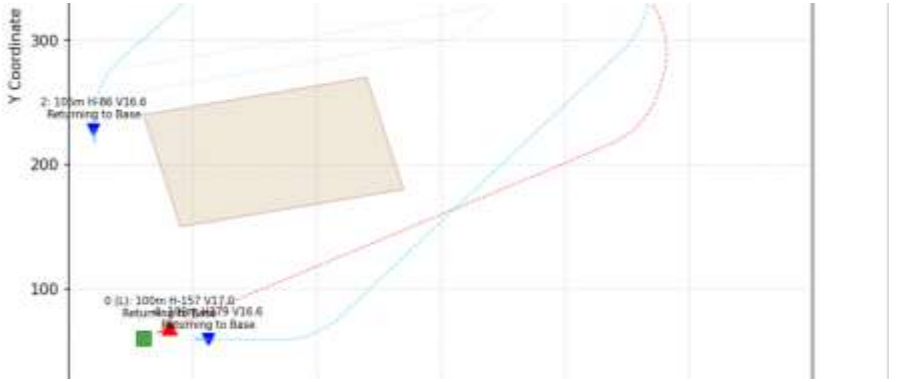
- Düzensiz NFZ için: İHA'nın veya rotasının NFZ içindeki alanla kesişimi.

Dinamik Sapma (Detour) Ara Noktası Oluşturma: Kesişim tespit edildiğinde, İHA NFZ'yi teğet geçecek en kısa ve güvenli yeni bir ara nokta hesaplar.

Uçuşa Yasak Bölgeler (NFZ) ve GBAD Kaçınma



Şekil 9. Simülasyona NFZ dahil edilmediğinde İHA'ların davranışı



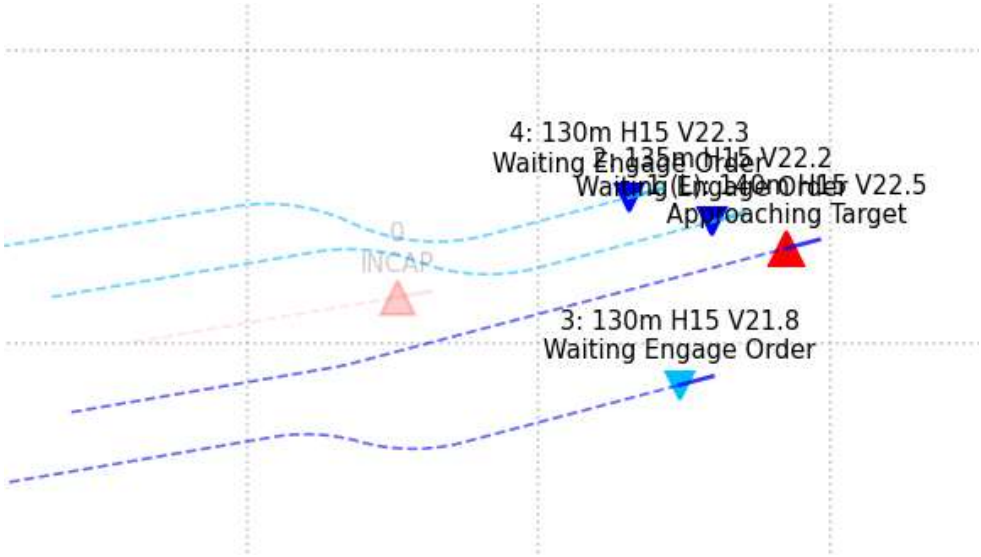
Şekil 10. Simülasyona NFZ dahil edildiğinde İHA'ların kaçınma davranışı

Lider Arızası ve Yeniden Lider Seçimi

Merkezi Yetki Devri ve Yeni Lider Seçim Kriterleri:

- **En Düşük ID'ye Sahip İHA:** Kalan aktif İHA'lar arasından en küçük ID numarasına sahip olan seçilir. Basit, hızlı ve kararlı bir seçim.

- **Üsse En Yakın İHA:** Özellikle görev dönüşü veya hasar sonrası hızlı toparlanma ihtiyacında tercih edilir. Aktif İHA'lar arasından üs konumuna en kısa mesafede (Öklid hesabı ile) olan İHA seçilir. Görev verimliliğini ve operasyonel geri dönüş süresini optimize eder.



Şekil 11. Lider arızasında yeni lider seçimi ve formasyon

Saldırı Stratejileri

1. Lider Direkt Saldırı:

Lider İHA: Hedefin radar alanına girer, hedefle angaje olur, ardından tehditten kaçınmak için kaçınma (evasion) manevrası yapar ve üsse döner.

Takipçi İHA'lar: Lideri formasyonda takip eder; GBAD karşı saldırı başlattığında, çevre hattından (perimeter) hedefle angaje olurlar.

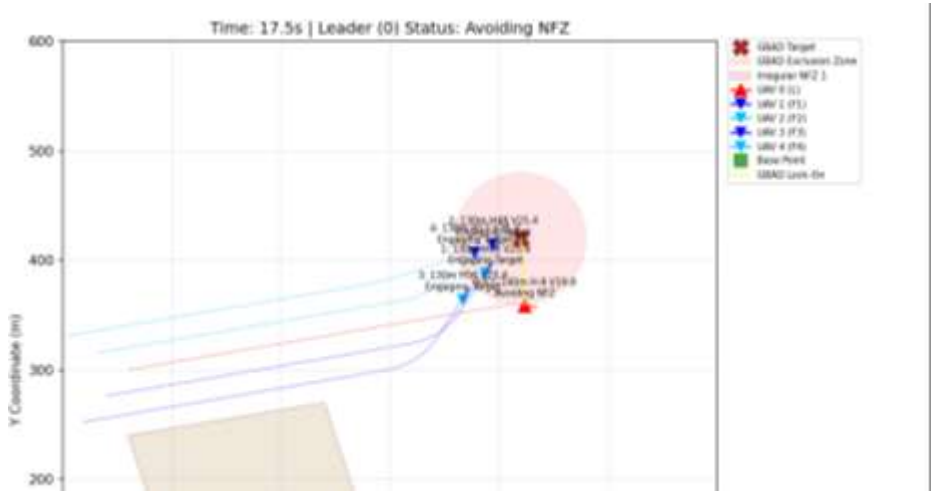
2. Kanattan Kuşatma ve Yakınsama:

Tüm İHA'lar, hedef GBAD'ı iki farklı kanattan (flank) yaklaşır (ID'ye göre ayrılırlar).

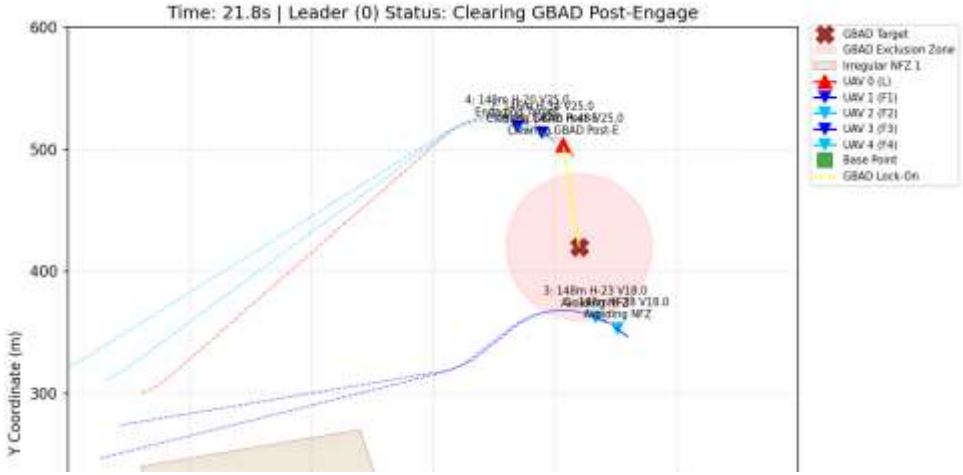
Kanat noktalarına ulaştıklarında, ortak bir yakınsama (converge) noktasına yönelerek angajmanı başlatır ve sonra dağılırlar.

Bu strateji, hedef savunmasını farklı açılardan aşmaya yönelik koordineli bir saldırı modelidir.

Saldırı Stratejileri



Şekil 12. “Lider direkt saldırı” senaryosunda etkileşim anı



Şekil 13. “Kanattan Kuşatma ve Yakınsama” senaryosunda etkileşim anı

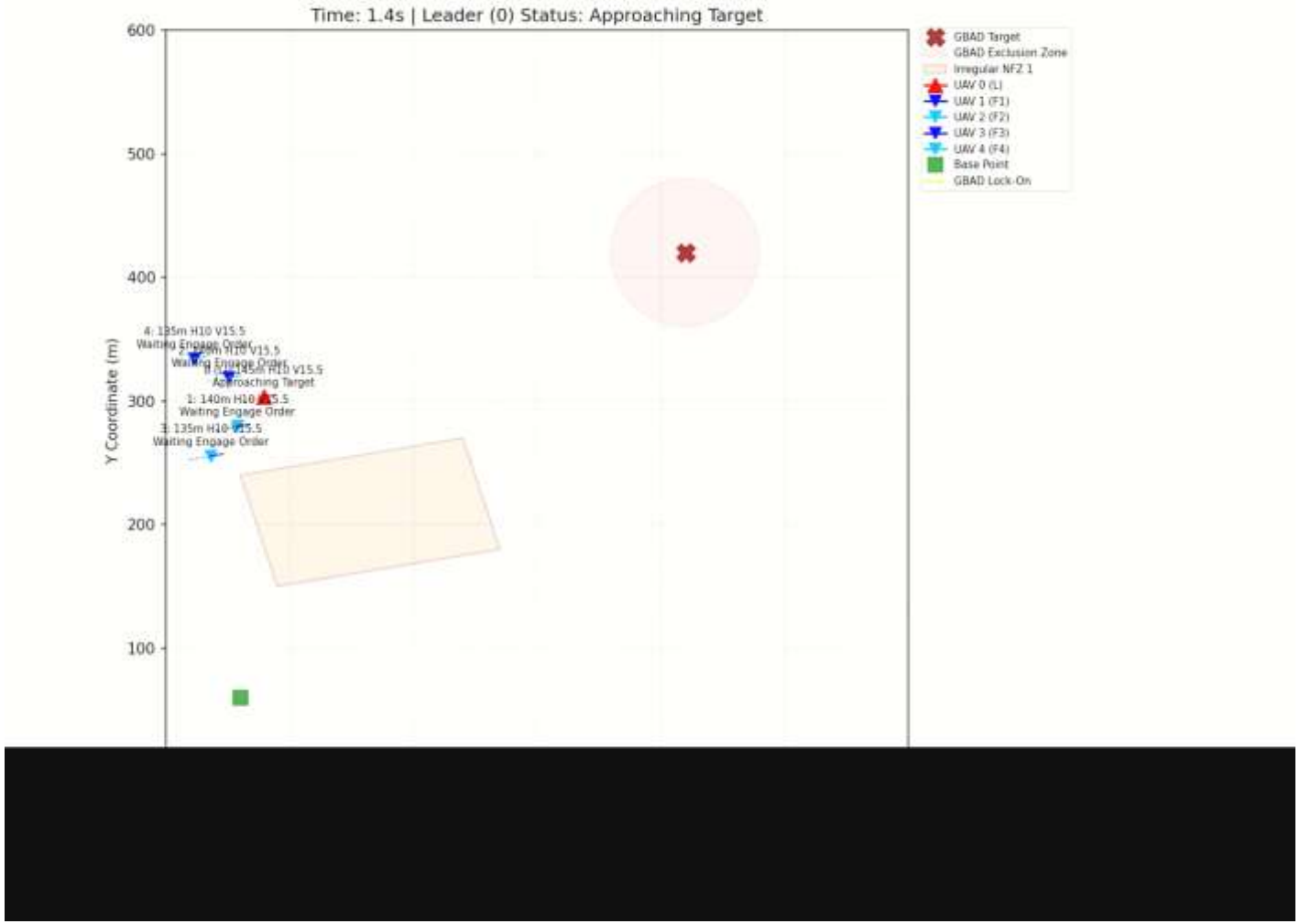
Kapsamlı ve Birleştirilmiş Simülasyon Örneği

```
--- Attack Strategy Selection ---
1. Leader Direct Attack (Leader attacks, followers support)
2. Flanking Maneuver and Converge (UAVs approach GBAD from two flanks and converge)
Select attack strategy (1 or 2):

--- NFZ (No-Fly Zone) Inclusion ---
Include NFZs in simulation? (yes/no):
NFZs will be included.

--- UAV Failure Scenario ---
Simulate UAV failure? (yes/no):
Enter failure time in seconds (e.g., 5.0):
```

Şekil 11. Programın terminalden başlatılması



Video 1. Lider direkt saldırı simülasyonu ve lider kaybı durumu

Python ile Gerçek Zamanlı 3D İHA Kontrolü

Amaç: SR2 simülatöründe TB2 benzeri bir İHA'yı Python ile otonom kontrol etmek.

Temel Yapı:

Python, UDP protokolüyle SR2'den telemetri verisi alır.

pyKey modülü ile klavye tuşları simüle edilerek kontrol yüzeyleri yönetilir.

Elevator (pitch) ve aileron (roll) kontrolü aktif; rudder (yaw) devre dışı.

Paralel Kontrol:

ThreadPoolExecutor ile her eksen için çoklu thread yapısı kuruldu.

Eşzamanlı kontrol ile kararlı ve gecikmesiz tepki sağlandı.

Navigasyon:

WaypointNavigator sınıfı ile hedef konumlara yönlendirme sağlandı.

PID kontrolü sayesinde yumuşatılmış manevralar gerçekleştirildi.

Python ile Gerçek Zamanlı 3D İHA Kontrolü

Materyal:

- **Juno:** İHA hareketlerinin 3D ortamda simülasyonların test edilmesi.

- **Python 3.10:** Simülasyonların geliştirilmesi.

    - **Kullanılan Kütüphaneler**: NumPy, Matplotlib

    - **Geliştirme Ortamı**: VS Code

Waypoint Tabanlı Otonom Navigasyon Sistemi

Mantık:

Her waypoint: enlem, boylam, yükseklik ve tolerans değeriyle tanımlanır.

calculate_distance() ile İHA ve hedef arasındaki mesafe hesaplanır.

Ana Fonksiyonlar:

add_waypoint(): Yeni hedef ekleme

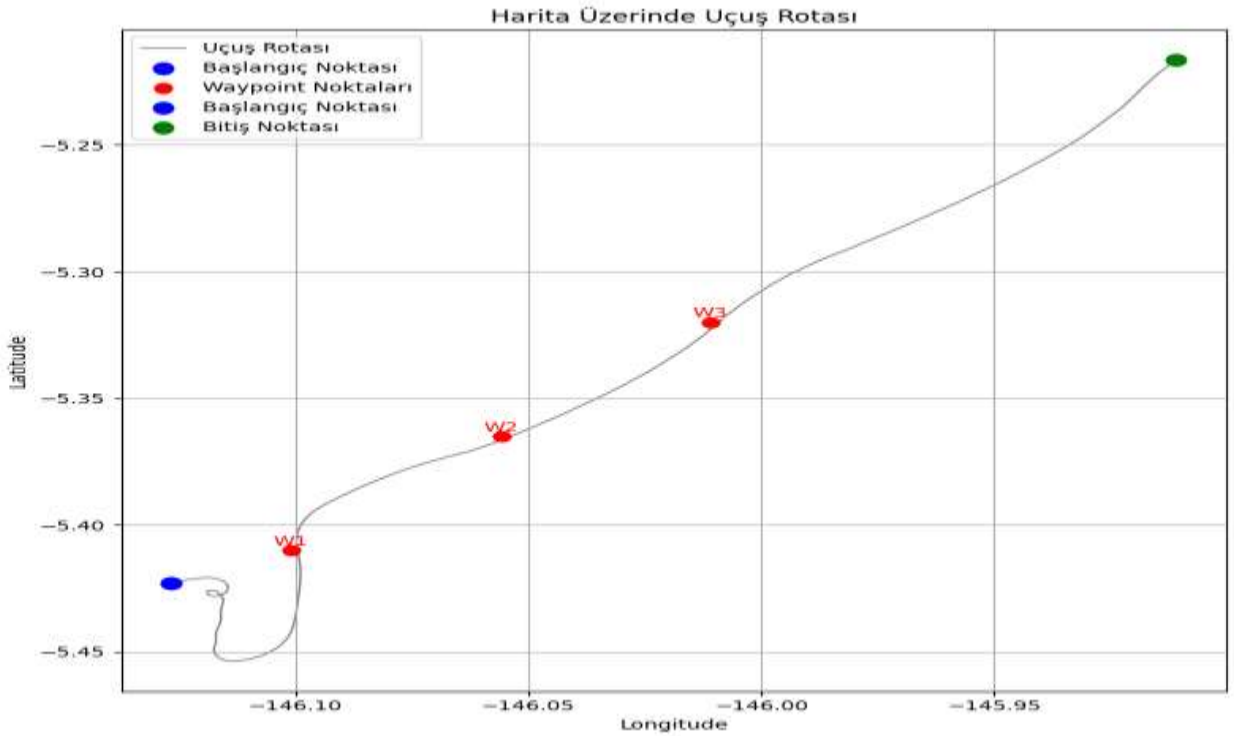
clear_waypoints(): Tüm hedefleri silme

check_waypoint_reached(): Aktif hedefe ulaşıldı mı?

Geçiş Koşulu:

Tolerans altı mesafe: bir sonraki hedefe geçiş tetiklenir.

Tek İHA ile Uçuş Testi



Şekil 14. Tek İHA ile izlenen rota

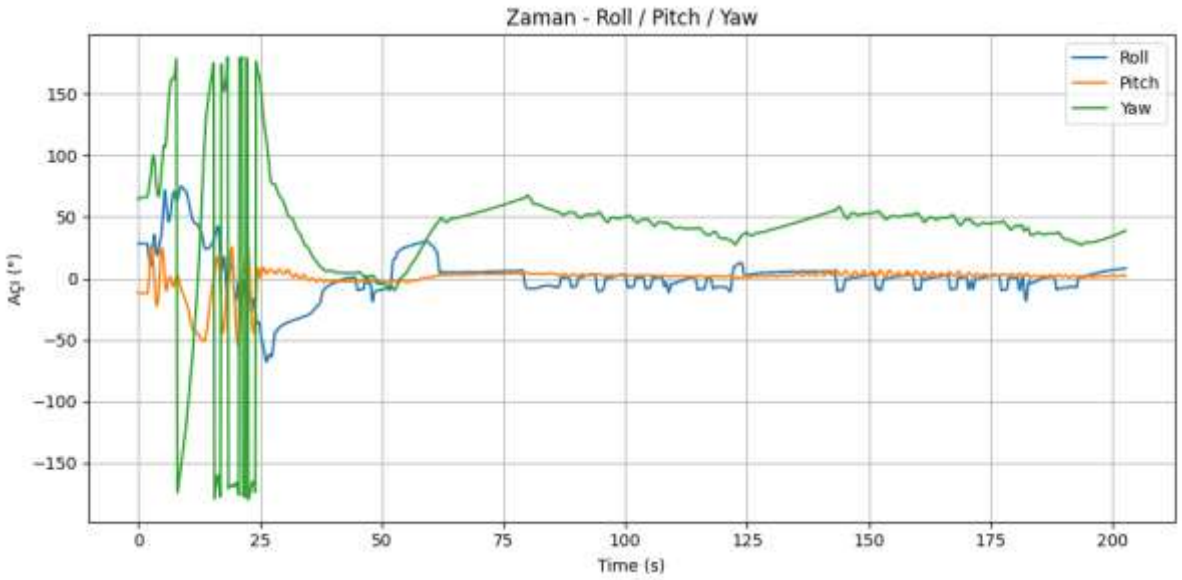
NavigationController ile Yönelim Hesaplaması

Yaw Tabanlı Yön Kontrolü:

- calculate_bearing() ile azimut açısı hesaplanır.
- Yaw hatasına göre uygun roll ve pitch komutları oluşturulur.

Yaklaşma Modu:

- 1000 m altı mesafede hassas manevralar aktive edilir.



Şekil 15. Tek İHA ile row/pitch/yaw – zaman değişimi

PID Tabanlı Stabilizasyon Mekanizması

Roll (Aileron) Kontrolü:

PID_Aileron: Roll hatasına göre PWM sinyali hesaplar.

P: Anlık hata şiddeti

I: Kümülatif hata birikimi

D: Hata değişim hızını düzenleme

Pitch (Elevator) Kontrolü:

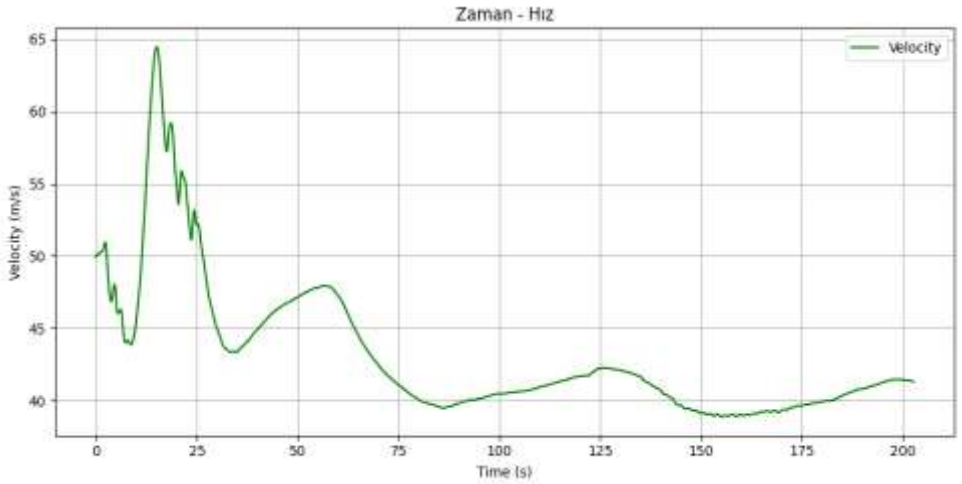
ver_vel (dikey hız) değeri ile pitch PWM belirlenir.

'w' ve 's' tuşları ile burun yönelimi sağlanır.

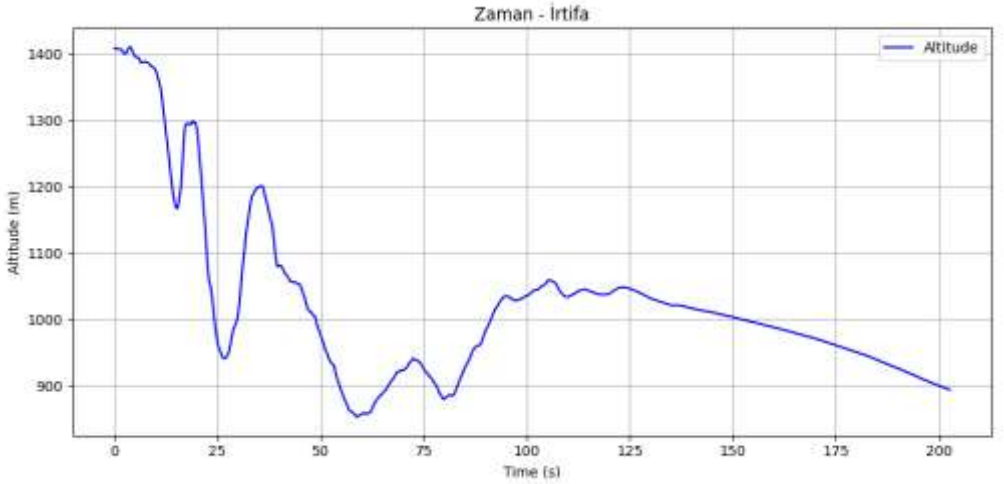
Duruma Göre Ayar:

Yaklaşma fazında PID tepkileri yumuşatılır.

Tek İHA ile Uçuş Testi



Şekil 16. Tek İHA ile hız – zaman değişimi



Şekil 17. Tek İHA ile irtifa – zaman değişimi

Çoklu Thread Yapısı ve Telemetri Mimarisi

Thread Mimarisi:
- Her kontrol ekseni için ayrı thread havuzları (roll & pitch).
- flight_data_event tetiklendiğinde thread aktif olur.

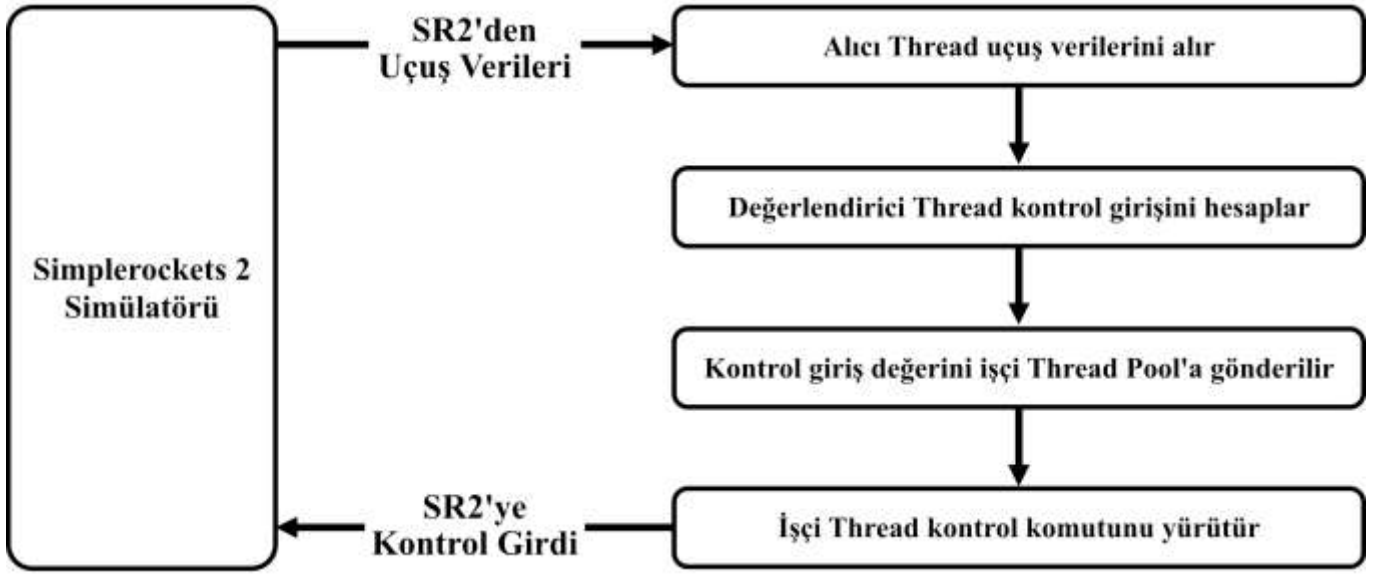
Avantajlar:
- Düşük gecikmeli kontrol
- Çakışmasız tuş simülasyonu

Veri Alımı:
- get_flight_data() ile UDP verisi okunur.
- roll, pitch, yaw, enlem, boylam, yükseklik, hız vs. anlık alınır.
- flight_data_stack'te saklanır ve thread'lere aktarılır.

Akıllı Mühimmat Güdüm Sistemlerinin Geliştirilmesi

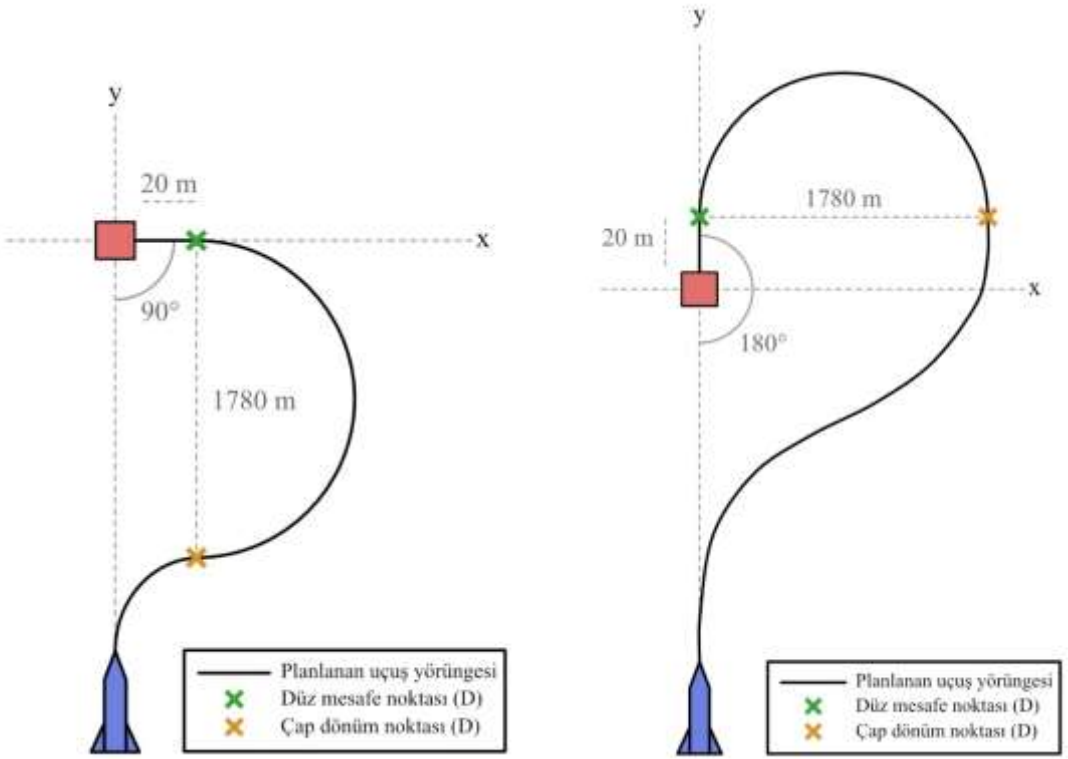
Python Üzerinden Füze Kontrolü

Video 2. Simplerockets 2'de füze kontrol simulsayonu



Şekil 18. Simplerockets 2 ile Python arasında iletişim akış diyagramı

Terminal Yaw Açısı Kontrolü



Şekil 19. Terminal yaw açıları için uçuş rota planlaması

- Hedefin koordinatının bilindiği varsayılırı
- Düzgün yörünge planı için Bezier eğrisi

Terminal Yaw Açısı Kontrolü

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

5 Düz mesafe D derece cinsinden şu formül ile hesaplanır:

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

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

7 Eğer heading yönü Kuzey veya Güney ise:

8 i. Aşağıdaki 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:

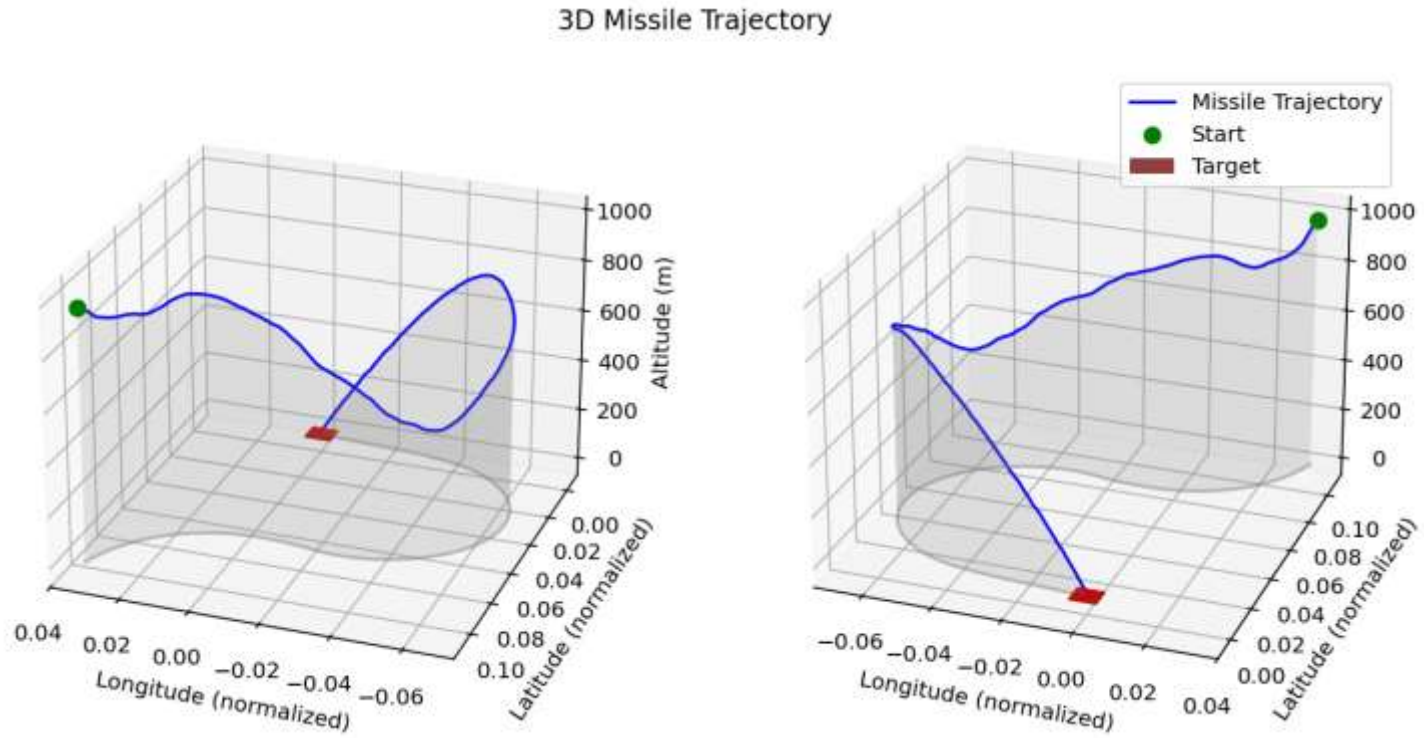
$waypoint_2^{\varphi} = waypoint_1^{\varphi} + \Delta\varphi_r$

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

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 + $(waypoint_1^{\varphi}, waypoint_1^{\lambda}) + (\varphi_{target}, \lambda_{target})$

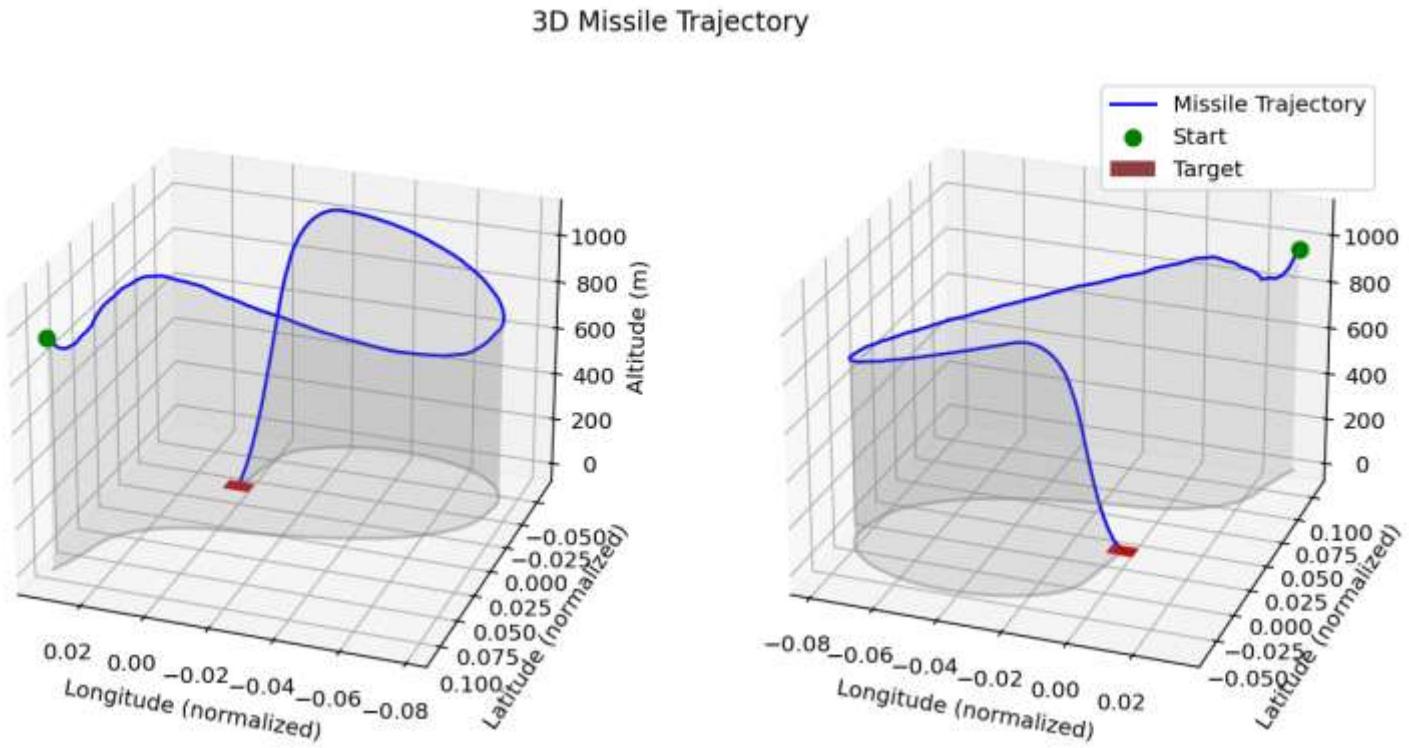
Terminal Yaw Açısı Kontrolü - 90°



https://youtu.be/E6KyTVSojUM

Şekil 20. 90 derece terminal yaw simülasyonun 3B uçuş yörüngesi

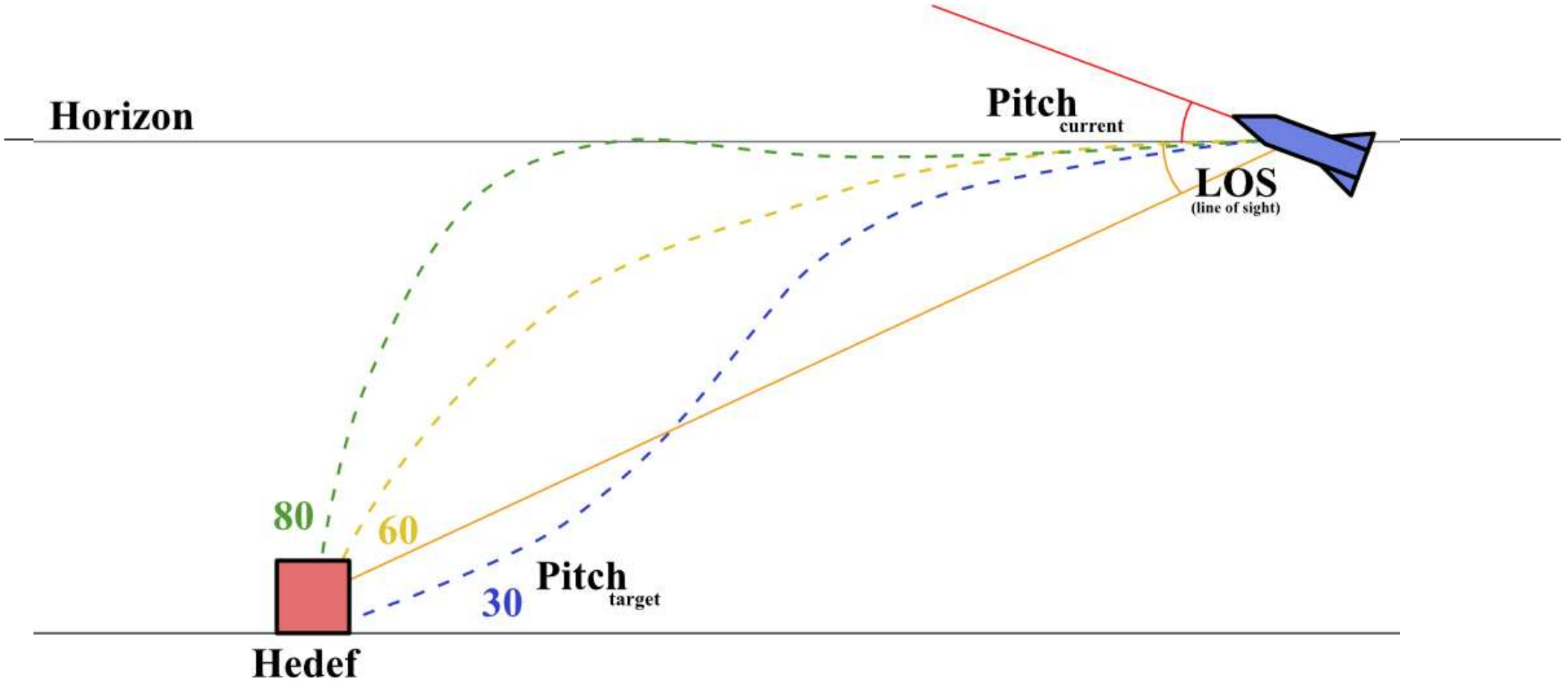
Terminal Yaw Açısı Kontrolü - 180°



https://youtu.be/8NQi_WI8_z0

Şekil 21. 180 derece terminal yaw simülasyonun 3B uçuş yörüngesi

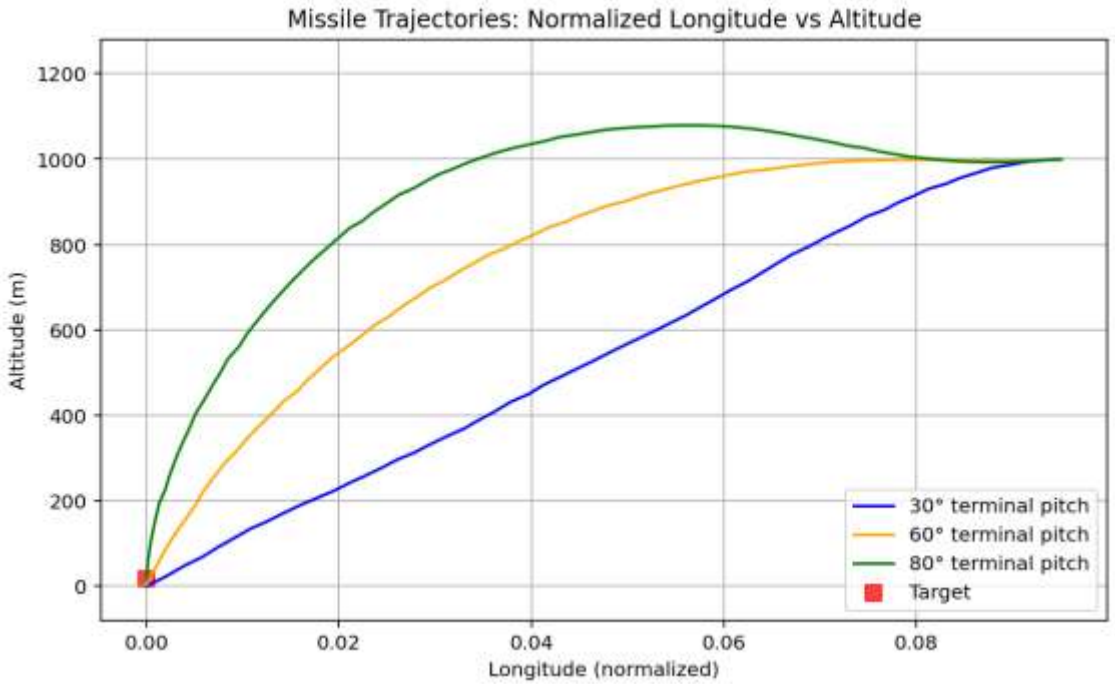
Terminal Pitch Açısı Kontrolü



Şekil 22. Terminal pitch açıları için uçuş rota planlaması

$$Elevator\ Input = \left(2 \cdot LOS - Pitch_{target} - Pitch_{current}\right) \times Kp\ - pitch\ rate\ \times Kd$$

Terminal Pitch Açısı Kontrolü

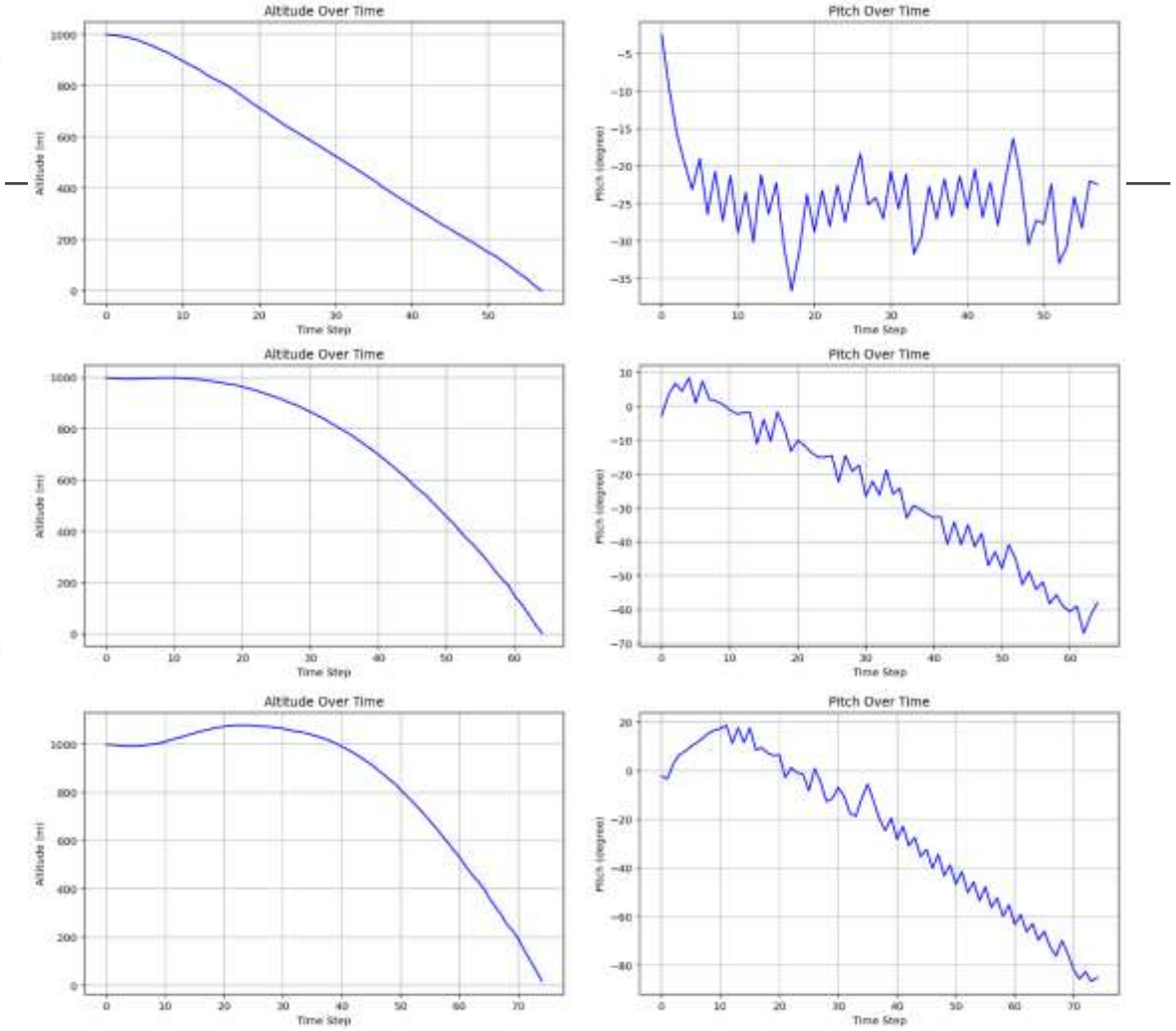


Şekil 23. Terminal pitch açıları simülasyonların sonucu (İrtifa vs Boylam)

https://youtu.be/Jalqhvjqka8

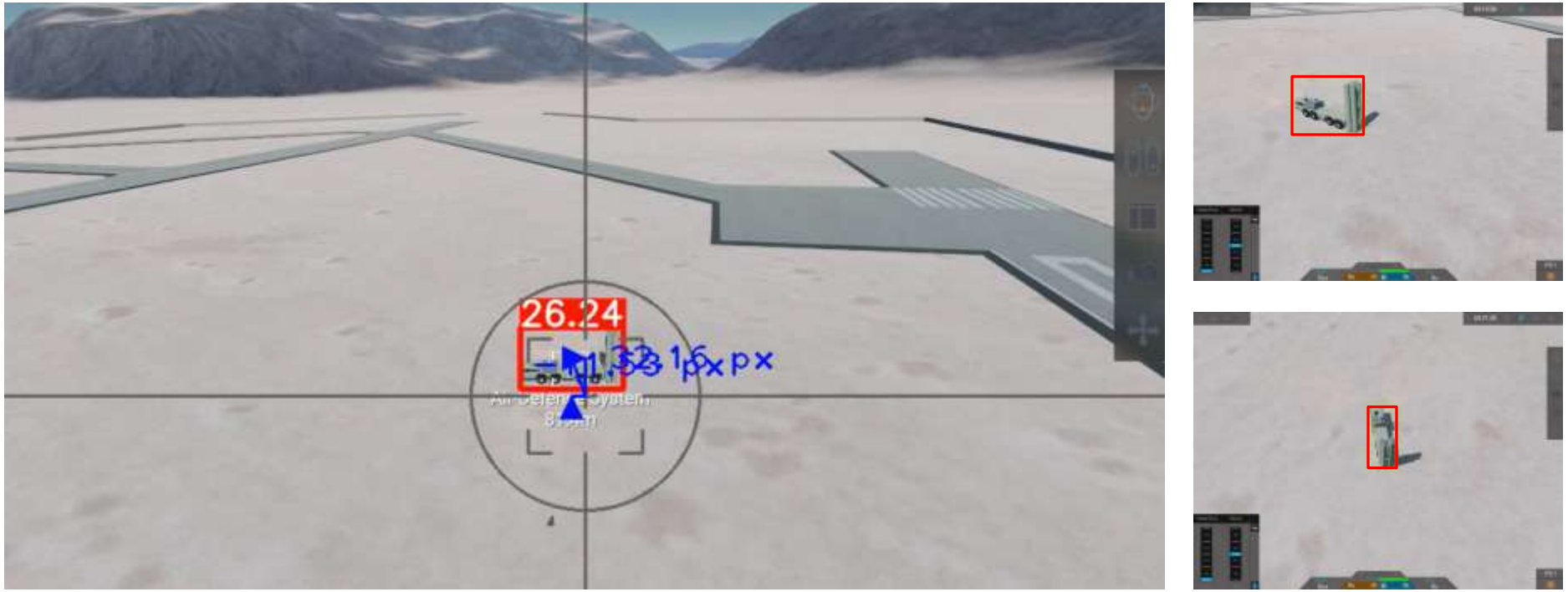
https://youtu.be/q4M9z1Wk3F8

https://youtu.be/HQMIdT-MxMg



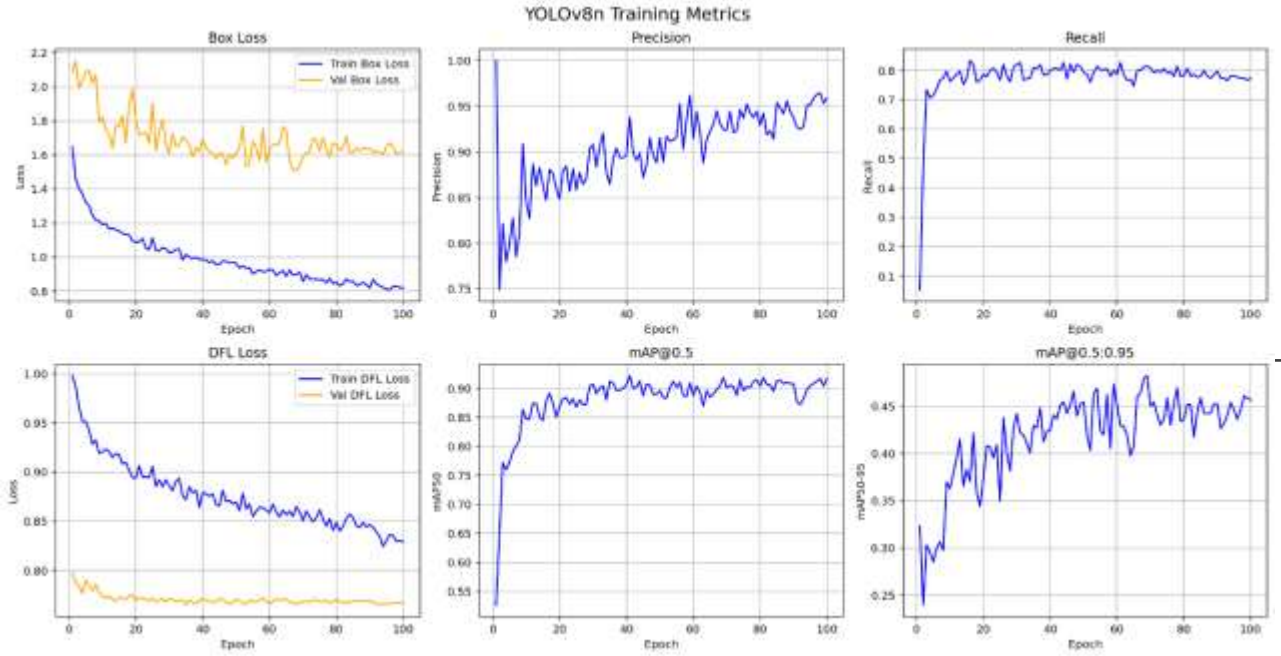
Şekil 24. Terminal pitch açıları simülasyonların sonucu (İrtifa ve Pitch değerleri)

Hedef Tespiti İçin YOLOv8

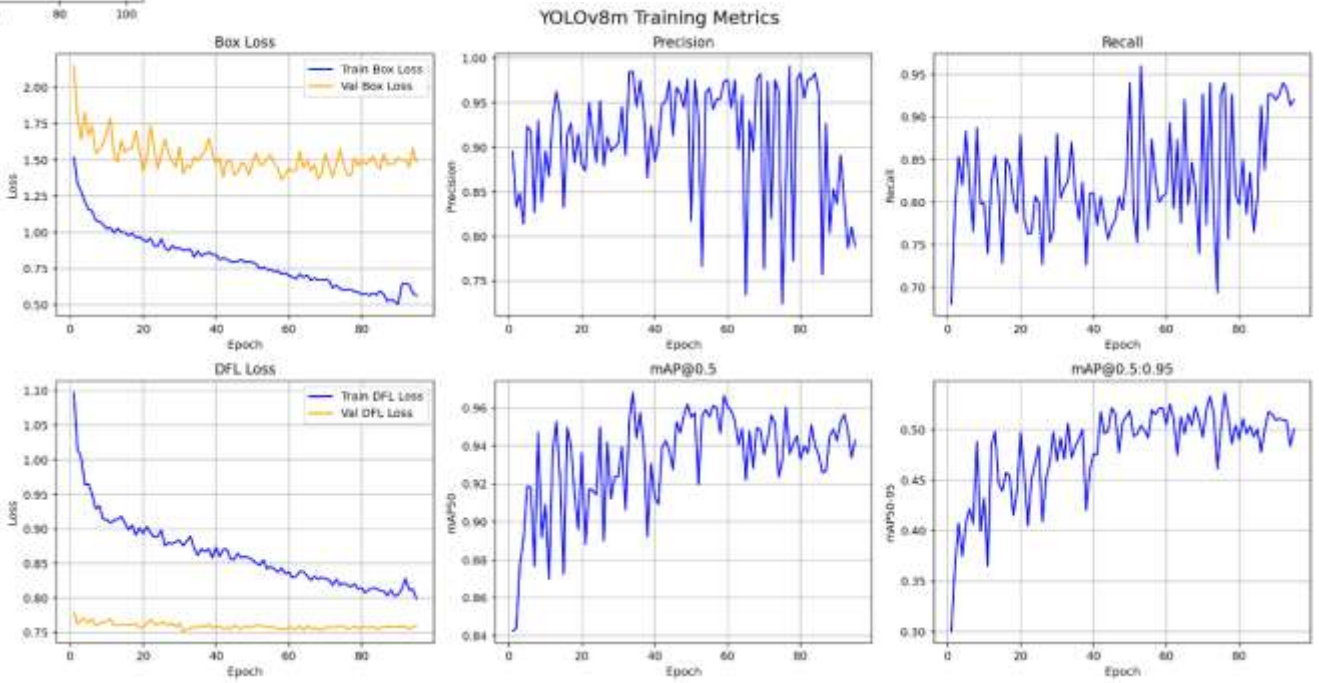


Şekil 25. YOLOv8 nesne tanıma modeli ile hedef algılaması

Hedef Tespiti İçin YOLOv8

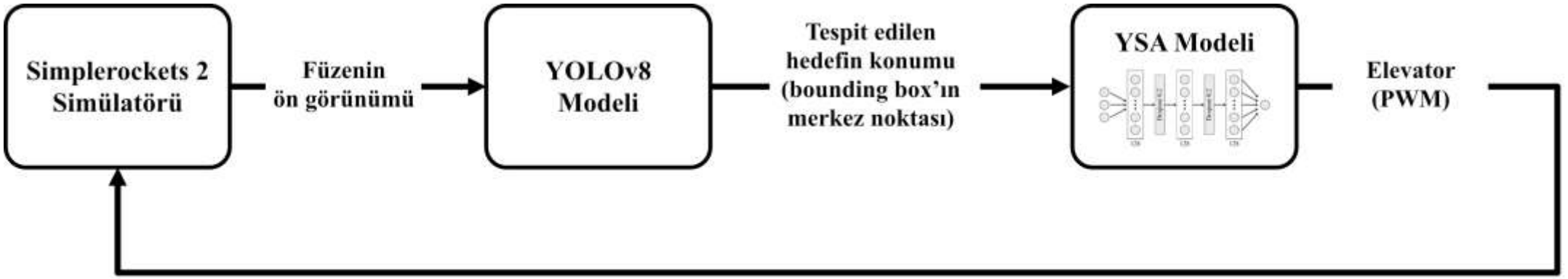


Şekil 26. YOLOv8 N modeli eğitim metrikleri



Şekil 27. YOLOv8 M modeli eğitim metrikleri

YOLOv8 Füze navigasyon sistemine entegrasyon

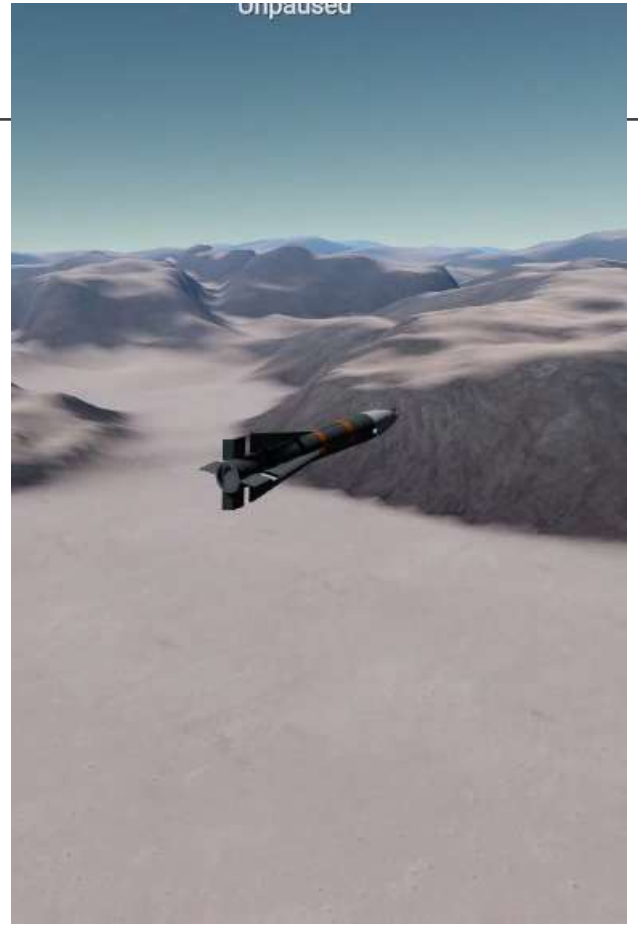


Şekil 28. YOLO ile füze pitch kontrolu sistemine entegrasyon akış diyagramı

https://youtu.be/thOxi6gd1_Y

Füze Roll Stabilizasyon Sistemi

Roll Stabilizasyon: off



Video3. Roll stabilizason sistemini kullanmadan rudder manevrası

Roll Stabilizasyon: on

Video4. Roll stabilizason sistemini kullanarak rudder manevrası

Füze Roll Stabilizasyon Sistemi

- Oransal-Türevsel-İntegral (PID) kontrolör tasarımı
- Doğrusal regresyon tabanlı kontrolör
- Long Short-Term Memory (LSTM)
- Deep Deterministic Policy Gradient (DDPG) pekiştirmeli öğrenme

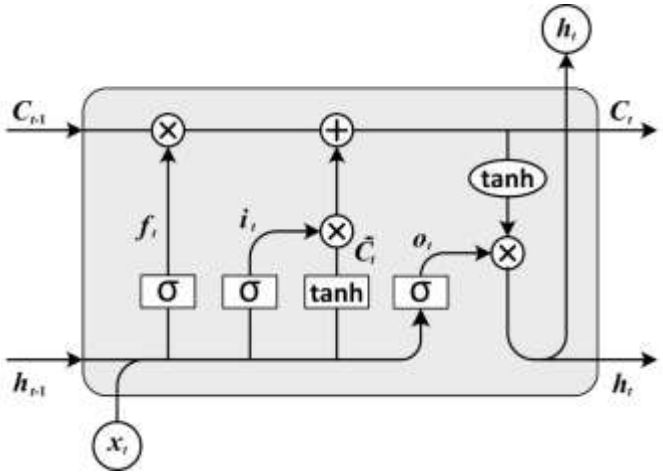
Füze Roll Stabilizasyon Sistemi

Doğrusal regresyon tabanlı kontrolör

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

Şekil 29. Doğrusal regresyon formülü

LSTM tabanlı kontrolör

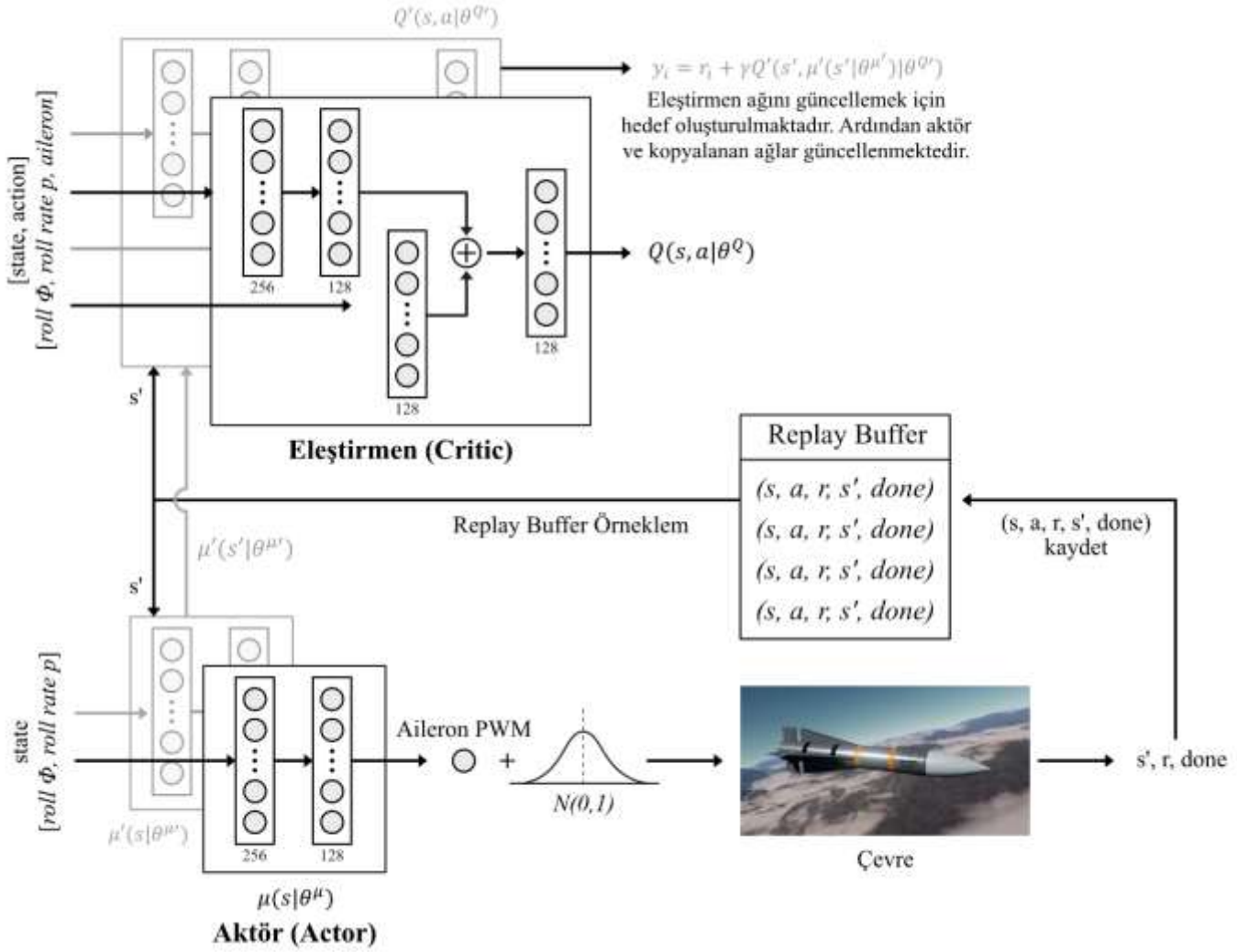


Şekil 30. LSTM hücresi

Füze Roll Stabilizasyon Sistemi RL - DDPG

DDPG:

- aktör-eleştirmen (actor-critic) mimari
- model içermeyen (model-free)
- sürekli değer (continuous)



Şekil 31. DDPG algoritması

DDPG

Reward function:

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

Burada ,

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

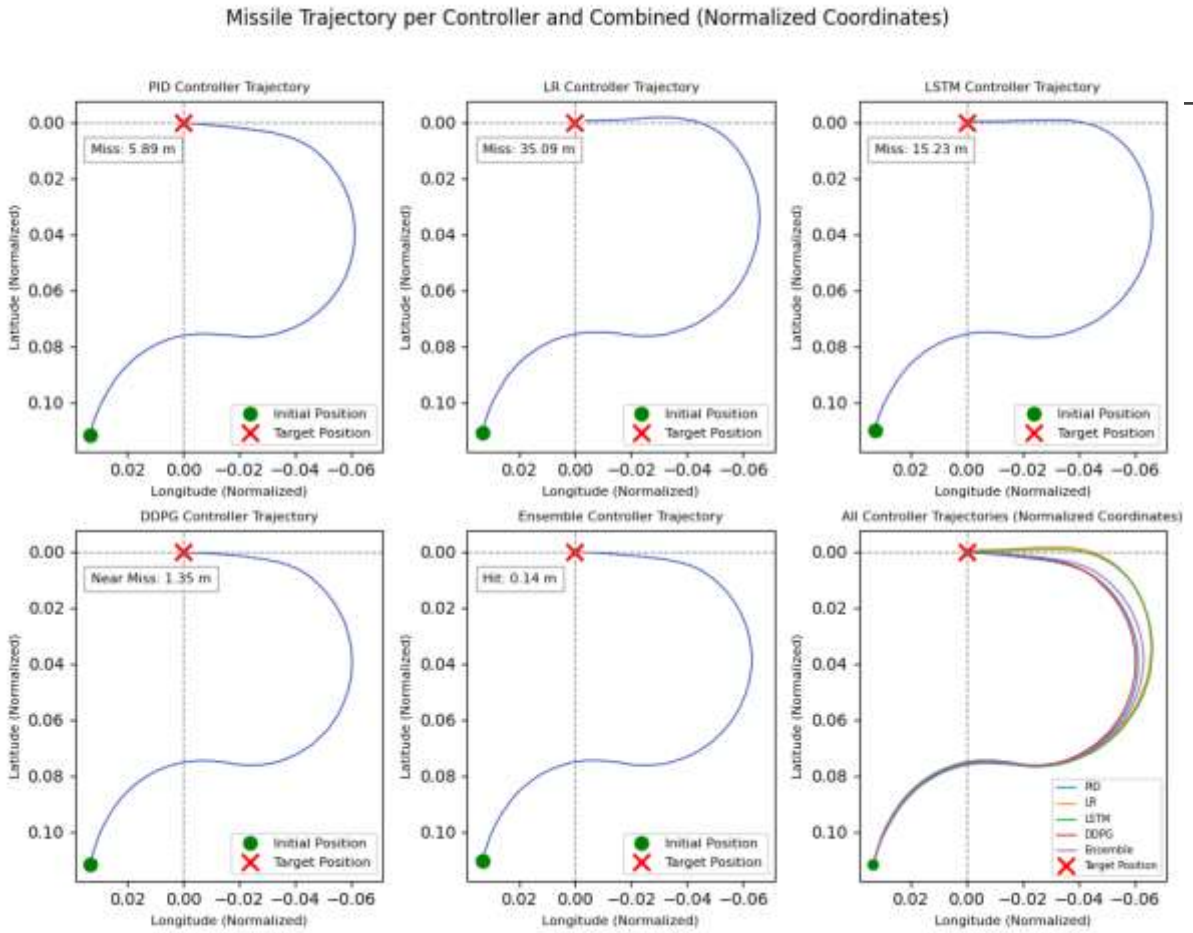
Tablo 1. DDGP modelinin etğitim hiperparametreleri

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

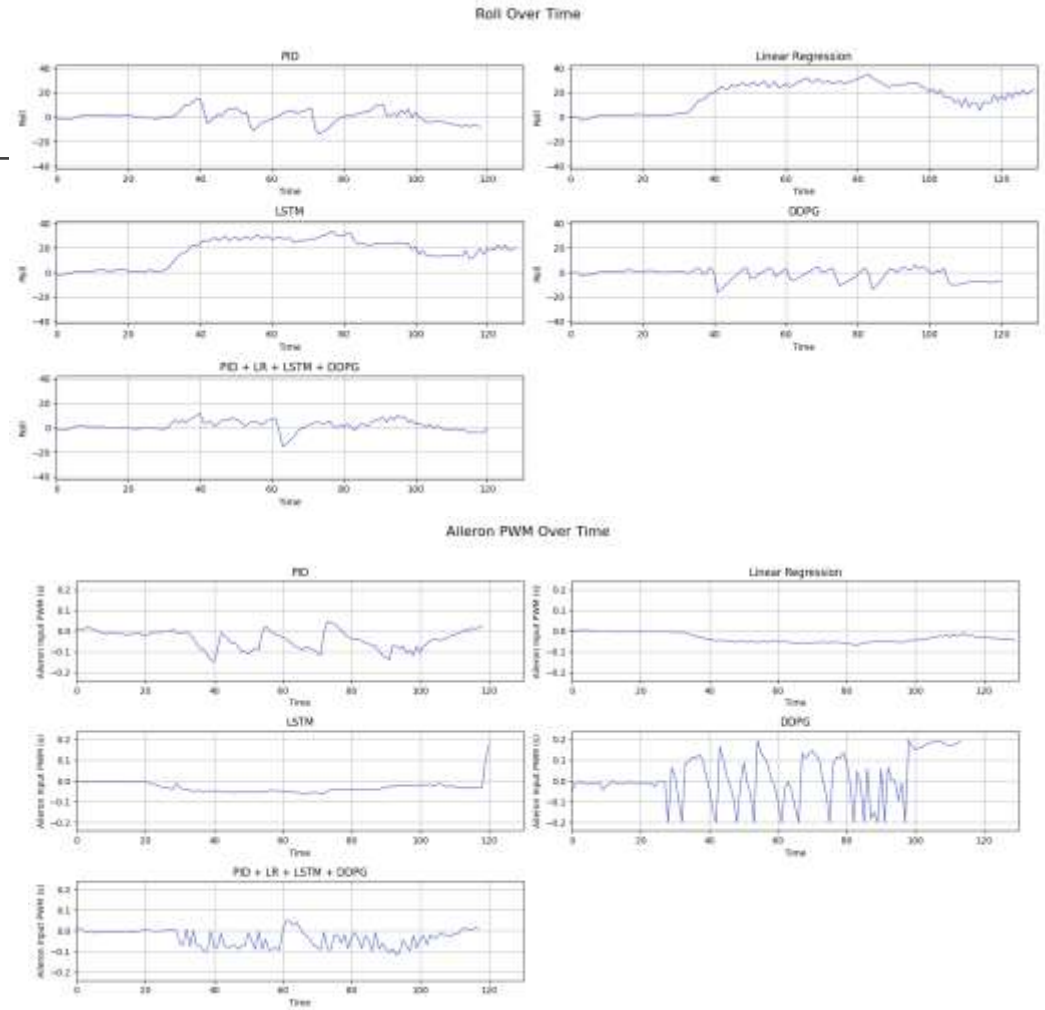
DDPG

Şekil 32. DDPG model eğitimin ortalama ödül

Füze Roll Stabilizasyon Sistemi

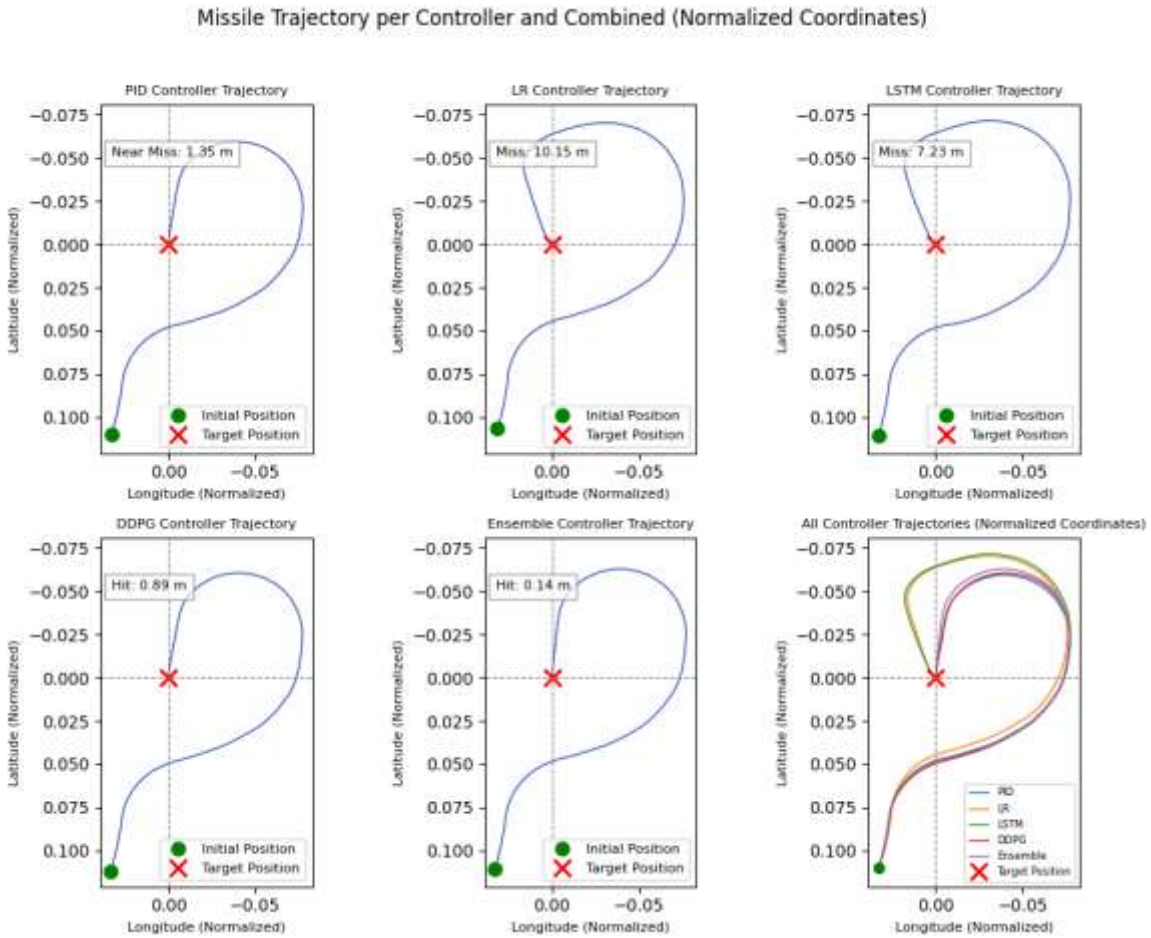


Şekil 33. 90 dereceli terminal yaw uçuş yörüngeleri

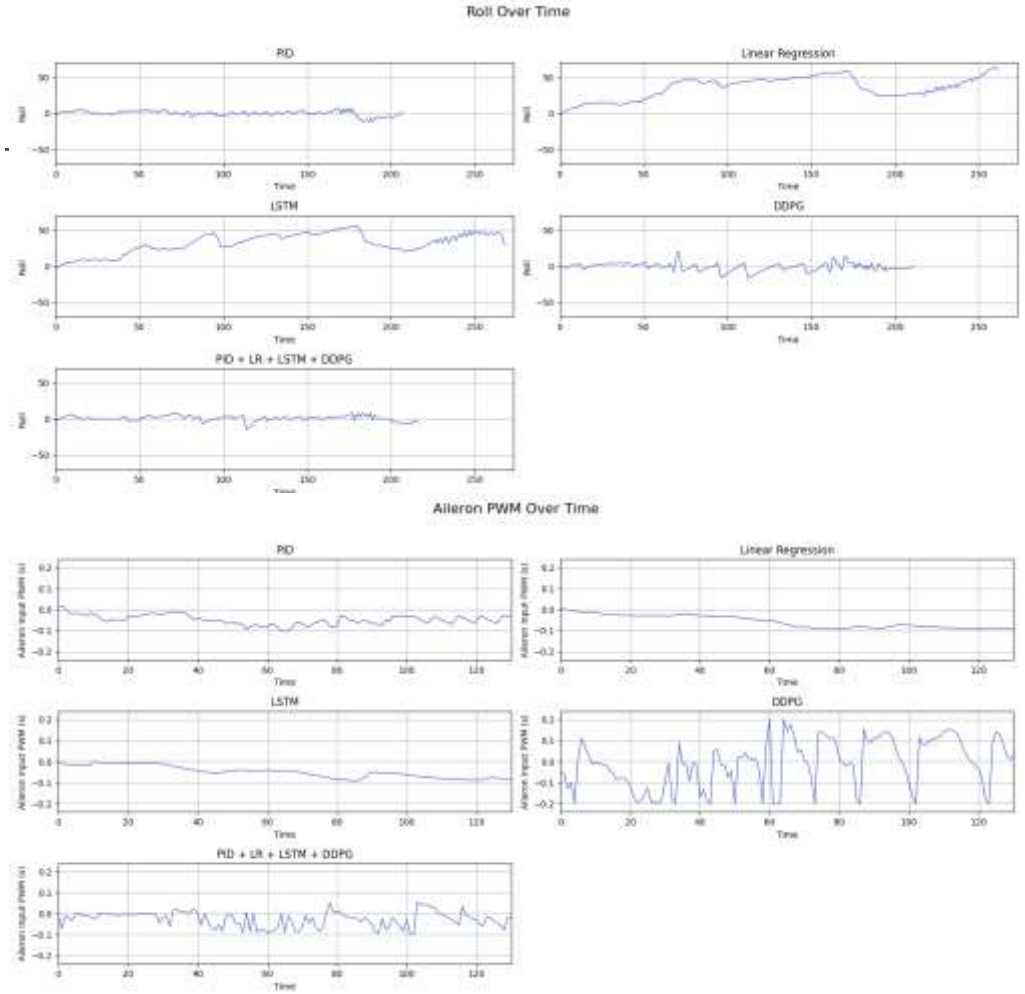


Şekil 34. 90 dereceli terminal yaw uçuşbilgileri (Roll ve Aileron girdi)

Füze Roll Stabilizasyon Sistemi



Şekil 35. 180 dereceli terminal yaw uçuş yörüngeleri



Şekil 36. 180 dereceli terminal yaw uçuşbilgileri (Roll ve Aileron girdi)

Füze Roll Stabilizasyon Sistemi

Tablo 3. 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

Tablo 4. 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 5. 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

Gelecek Çalışmalar

İleriye dönük çalışmalarda, bu çalışmada geliştirilen yöntemlerin doğrulamasını daha sağlıklı yapabilmek için daha gelişmiş ve esnek bir simülasyon ortamının kullanılması ya da özel olarak geliştirilmesi önerilmektedir.

Buna ek olarak, saha uygulamasına geçilebilmesi için gerçek zamanlı donanım testleri, uçuş denemeleri ve ilgili alt sistemlerin fiziksel entegrasyonu gibi adımların planlanması gerekmektedir.

Teşekkürler.