PROJE RAPORU

MAVI PROJE

TAMAMLANDI

Trieme (2024)

Yazar(lar):

Famura

İçerik

1. Proje Tanımı

1.1. Giriş

Trieme V1, Teknofest 2024 İnsansız Deniz Aracı (İDA) yarışma parkurunu tamamlamak üzere tasarlanmıştır. Ne yazık ki projenin kaynak kodları gizlidir. Bu nedenle bu rapor herhangi bir kod içermemektedir. Ancak sistemi tek başıma (Famura) tasarladığım için mimariyi açıklama ve görselleştirme hakkına sahibim. Bu yüzden projeye ait mimari görseller raporda yer almaktadır. Ayrıca bu rapor yalnızca Trieme'nin **yazılım tarafına** odaklanmaktadır.

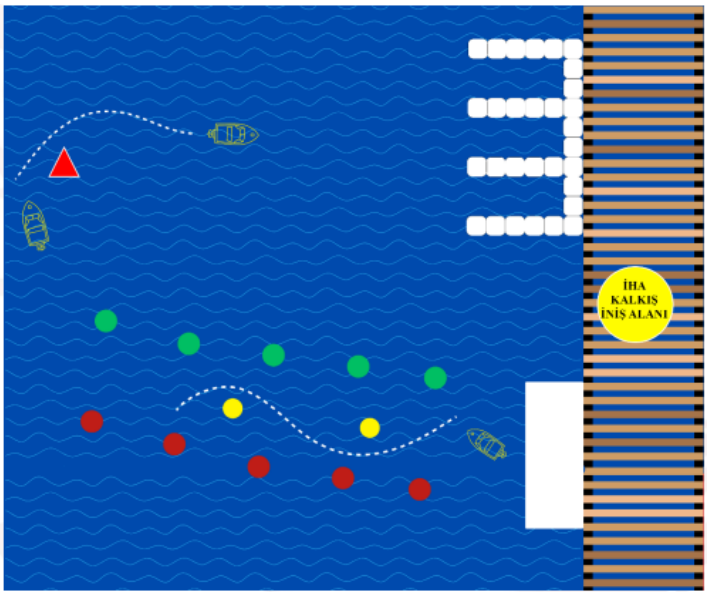
Ekip Şubat ayında kuruldu, yarışma Temmuz ayının başındaydı ve yazılım ekibi sadece iki kişiden (ben dahil) oluşuyordu. Bu sürede geminin kontrolünü sağlayan algoritmaları geliştirdik. Önümüzdeki yıl için yapay zekâ tabanlı bir model kullanmayı planlıyoruz.

1.2. Yarışma Detayları

1.2.1. Parkur Detayları

Yarışma parkuru üç ana bölümden oluşmaktadır:

- Parkur 1
- Parkur 2

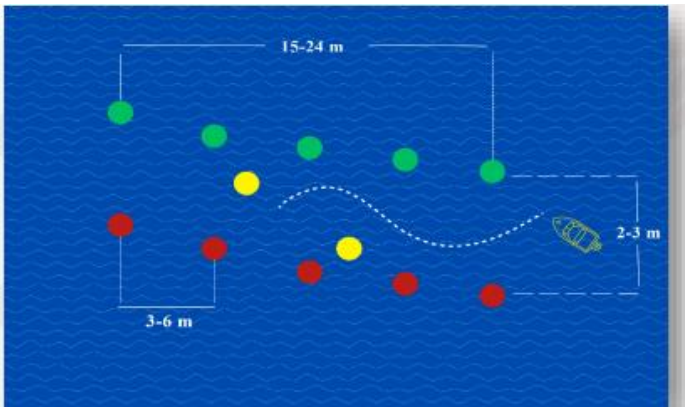


Gemi bu parkurların tamamını **tek bir kesintisiz denemede** tamamlamak ve ardından başlangıç noktasına geri dönmek zorundadır.

Tüm parkur için iki deneme hakkı verilmiştir.

Parkur sırasında geminin herhangi bir nesneye temas etmesi veya parkur dışına çıkması yasaktır. Bu durumlardan herhangi biri ceza puanı ile sonuçlanır. Ceza puanlarının tekrarı diskalifiye anlamına gelmektedir.

Aşağıda parkurların detaylı açıklamaları yer almaktadır.



1.2.1.1. Parkur 1

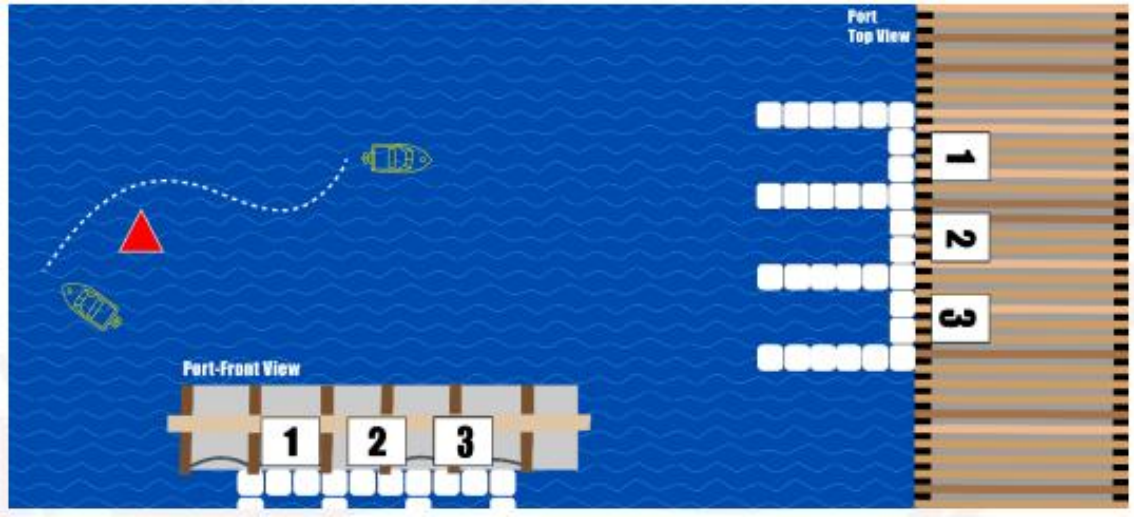
Parkur 1, geminin rota bulma, görüntü işleme ve engelden kaçınma algoritmalarını test etmek amacıyla tasarlanmıştır. Parkur boyunca sınır engelleri ve kaçınılması gereken engeller bulunmaktadır.

- Sol sınır engelleri: **kırmızı**
- Sağ sınır engelleri: **yeşil**
- Orta bölgede yer alan kaçınılması gereken engeller: **sarı**

Gemi, sınır engelleri arasında kalarak ve hiçbir nesneye çarpmadan parkuru tamamlamak zorundadır.

1.2.1.2. Parkur 2

Parkur 2, haberleşme ve rota bulma algoritmalarını test etmek için tasarlanmıştır. Otonom bir drone havalanarak mevcut üç park alanından hangisinin seçileceğine dair bilgiyi toplar. Ardından bu bilgiyi gemiye iletir ve gemi belirtilen park alanına park eder.



1.2.2. Ana Gereksinimler

Yarışmaya katılabilmek için yazılım ekibi tarafından karşılanması gereken bazı temel gereksinimler bulunmaktadır:

- Bir Yer Kontrol İstasyonu (GCS) bulunmalıdır
- Gemi GCS üzerinden kontrol edilebilmelidir
- Tüm telemetri verileri GCS üzerinden erişilebilir olmalıdır
- Gemi çevresini haritalandırmalı ve bu harita GCS üzerinde gerçek zamanlı görüntülenebilmelidir
- Gemi hem GCS hem de herhangi bir RC kontrolcü ile manuel olarak kontrol edilebilmelidir
- Gemi üç moda sahip olmalıdır: durma, manuel ve otonom
- Kullanıcılar bu modlar arasında istedikleri zaman geçiş yapabilmelidir
- Gemi moduna göre ışık yakmalıdır: Dururken kırmızı, manuel moddayken sarı, otonom moddayken yeşil ışıklar yanmalı

2. Proje Teknolojileri

2.1. Module(s)

- **Windows (Windows 10):**
  GCS Windows üzerinde çalışacak şekilde geliştirilmiştir. Casper Excalibur G770 üzerinde kullanılmıştır.
- **Ubuntu (Ubuntu 22.04):**
  Trieme, Ubuntu üzerinde çalışacak şekilde geliştirilmiştir. Jetson Orin Nano üzerinde kullanılmıştır.
- **Rasbery Pi OS:**
  Drone Raspberry Pi OS üzerinde çalışmaktadır. Raspberry Pi 4 kullanılmıştır.

2.2. Programlama Dilleri

- **Python - Pycharm**
  Projenin tamamı Python ile yazılmıştır ve Python 3 üzerinde çalışacak şekilde ayarlanmıştır. Tüm sanal ortamlar BIOS seviyesinde servis edilmekte ve ilgili elektronik modül güç aldığında otomatik olarak başlatılmaktadır.

2.3. Kütüphaneler ve Harici Eklentiler

- **Pyserial:**
  Elektronik modüller arası iletişimi sağlar. Telemetri modülleri, Wi-Fi ve mobil hat frekanslarıyla çakışmaması için elektronik ekip tarafından seçilmiştir.
- **YOLO:**
  Görüntü işleme için kullanılmıştır.
- **Threading:**
  Paralel iletişim süreçlerini yönetmek için kullanılmıştır.
- **PyGame:**
  GCS üzerinde telemetri verilerini ve canlı haritayı göstermek için kullanılmıştır.
- **PyMavlink:**
  Jetson Orin ile Pixhawk modülü arasındaki bağlantı için kullanılmıştır.

3. Proje Mimarisi

Mimari, Jetson Orin üzerindeki ana bilgisayar tarafından kontrol edilen ve Pixhawk, PyMavlink ve ROS ile koordine edilen sıralı fazlara ayrılmıştır. Sistem, ilerlemeyi kaydederek senaryolar arasında dinamik geçiş yapar ve sürekliliği garanti eder.

3.1. Sistem Akışı

3.1.1. Sistem Başlatma

- **Güç Verilmesi**
  Sisteme güç verildiğinde Jetson Orin aktif olur ve sistem başlatılır.

- **Kayıt Dosyasının Yüklenmesi**
  Sistem, mevcut görev senaryosunu belirlemek için kayıt dosyasını okur. Böylece herhangi bir kesinti sonrası araç kaldığı yerden devam edebilir.

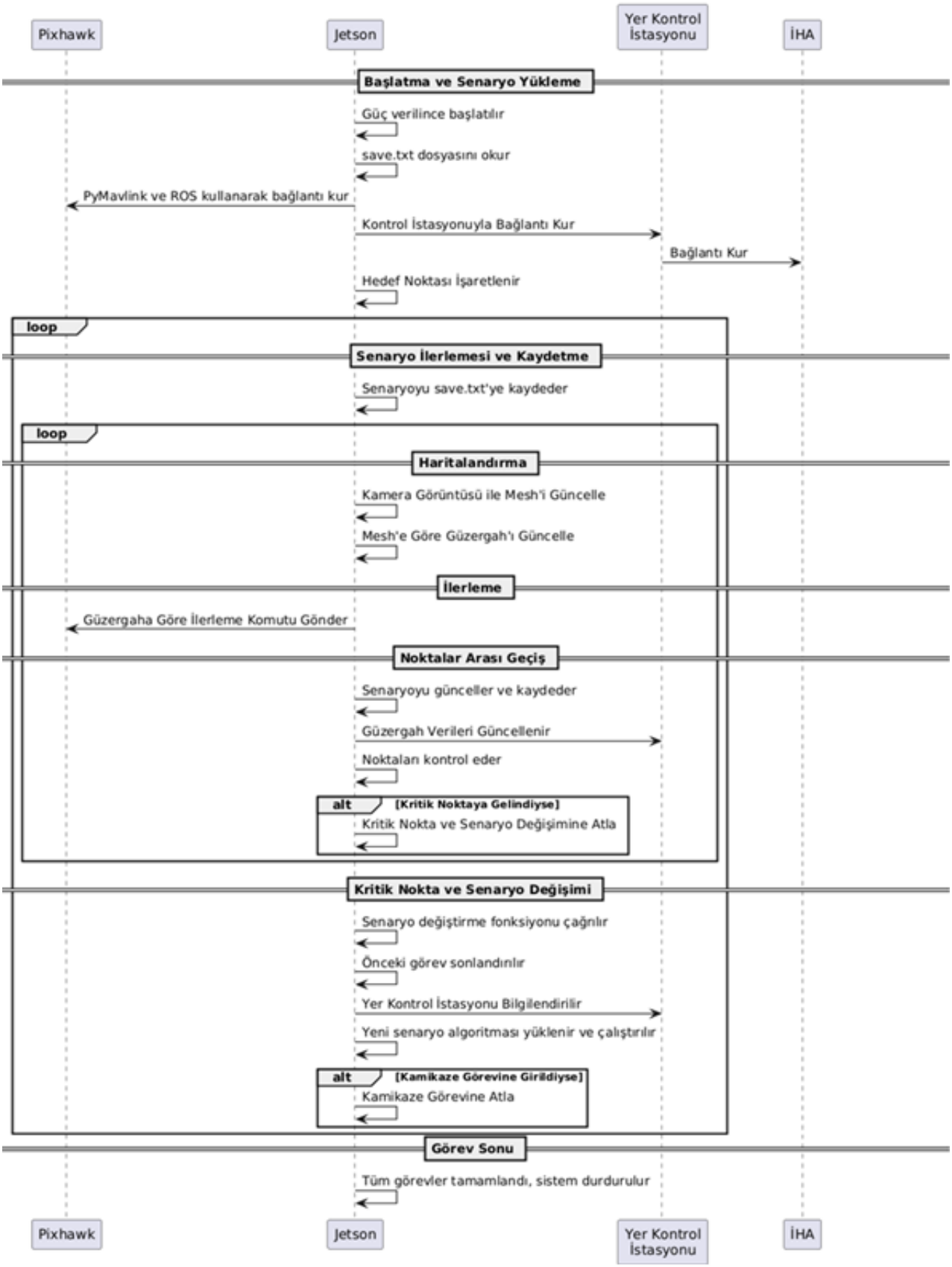
- **Senaryo Farkındalığı**
  Her senaryo sabit sayıda kontrol noktası ile tanımlanır. Araç bu noktaları kullanarak mevcut görevini tanır ve senaryolar arasında geçiş yapar.

3.1.2. Görev Yürütme

- **İlerleme Fonksiyonu**
  İlerleme fonksiyonu tetiklendiğinde Jetson Orin, Pixhawk uçuş kontrolcüsü ile PyMavlink ve ROS üzerinden iletişim kurar. Bu sayede İDA otonom olarak ilerler.

- **Senaryo Geçişi**
  Her senaryonun sonunda araç kritik bir noktaya ulaşır. Bu noktanın geçilmesi senaryo değiştirme fonksiyonunu tetikler; önceki görev sonlandırılır ve bir sonraki görev için gerekli algoritmalar yüklenir.

- **Tamamlama**
  Son kritik nokta geçildiğinde USV tüm görevleri tamamlar ve operasyonlarını sonlandırır.


Pixhawk
Jetson
Yer Kontrol İstasyonu
İHA
Başlatma ve Senaryo Yükleme
Güç verilince başlatılır
save.txt dosyasını okur
PyMavlink ve ROS kullanarak bağlantı kur
Kontrol İstasyonuyla Bağlantı Kur
Bağlantı Kur
Hedef Noktası İşaretlenir
loop
Senaryo İlerlemesi ve Kaydetme
Senaryoyu save.txt'ye kaydeder
loop
Haritalandırma
Kamera Görüntüsü ile Mesh'i Güncelle
Mesh'e Göre Güzergah'ı Güncelle
İlerleme
Güzergaha Göre İlerleme Komutu Gönder
Noktalar Arası Geçiş
Senaryoyu günceller ve kaydeder
Güzergah Verileri Güncellenir
Noktaları kontrol eder
alt
[Kritik Noktaya Gelindiyse]
Kritik Nokta ve Senaryo Değişimine Atla
Kritik Nokta ve Senaryo Değişimi
Senaryo değiştirme fonksiyonu çağrılır
Önceki görev sonlandırılır
Yer Kontrol İstasyonu Bilgilendirilir
Yeni senaryo algoritması yüklenir ve çalıştırılır
alt
[Kamikaze Görevine Girildiyse]
Kamikaze Görevine Atla
Görev Sonu
Tüm görevler tamamlandı, sistem durdurulur
Pixhawk
Jetson
Yer Kontrol İstasyonu
İHA

3.1.3. İHA ile Haberleşme

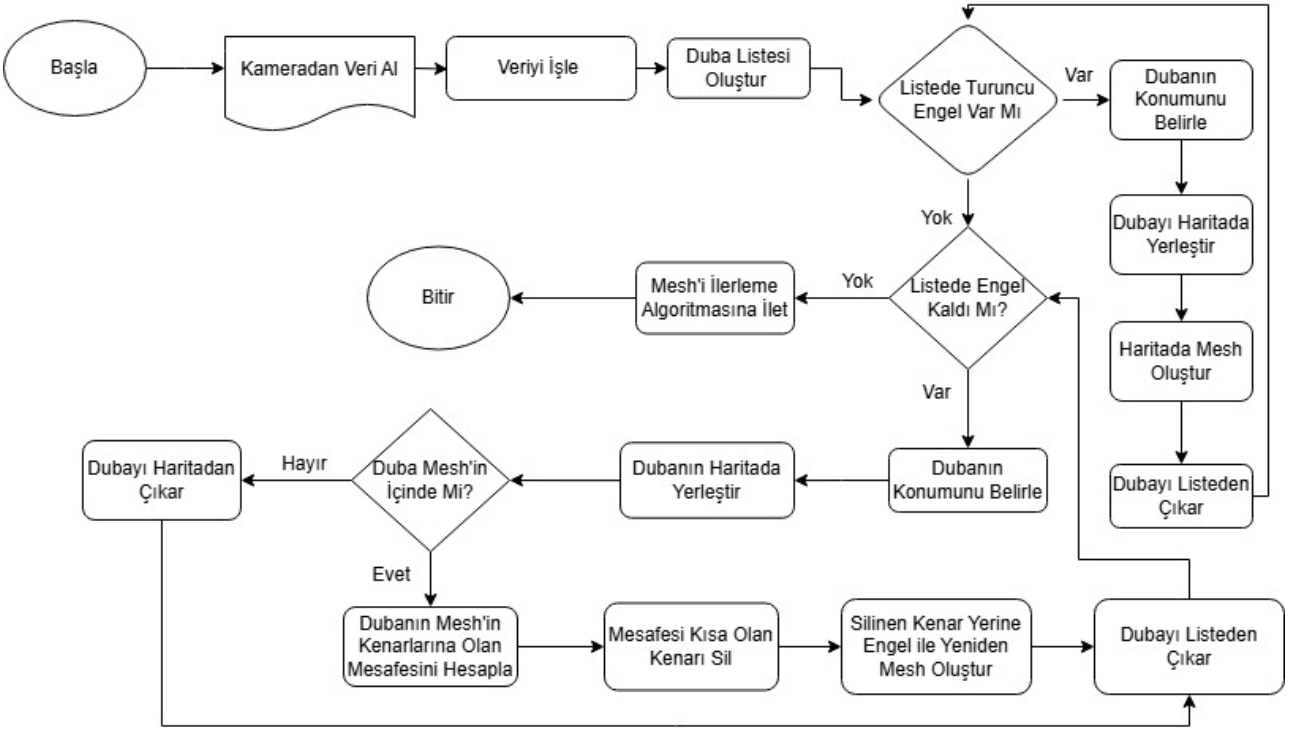
- **Veri Talebi**
  Park etme görevi için USV, hedef bilgilerini GCS'den talep eder.

- **Aktarma Mekanizması**
  GCS, verileri İHA'dan otonom olarak toplar ve USV'ye iletir. Bu yapı sayesinde UAV ve USV, GCS üzerinden birleşik bir haberleşme mimarisi altında çalışır.

3.2. Algoritmik Bileşenler

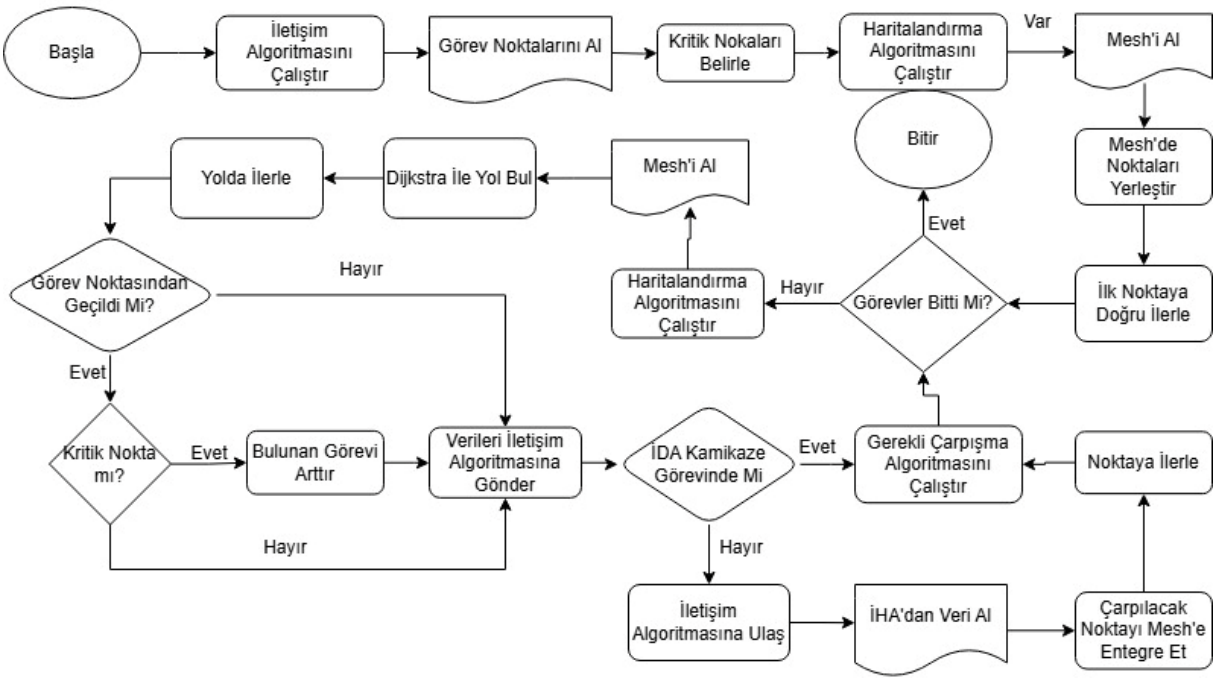
3.2.1. Haritalama Algoritması

- Kamera girdisi ve haberleşme verilerini entegre ederek mesh tabanlı bir harita oluşturur
- Turuncu şamandıraları bariyer olarak algılar ve aralarında mesh yapıları kurar
- Konumsal verileri ilerleme ve çarpışma kontrol algoritmalarına aktarır



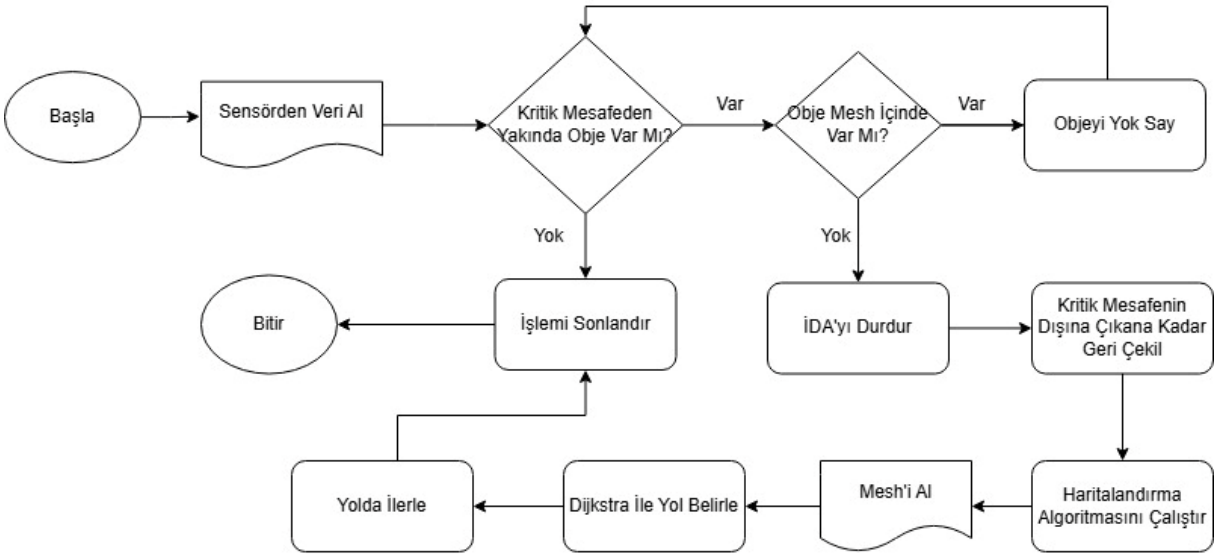
3.2.2. İlerleme Algoritması

- Sistemin çekirdeğini oluşturur
- Mesh yapısı üzerinde Dijkstra algoritmasını kullanarak en uygun rotayı belirler
- Diğer algoritmaların çalışmasını yönetir



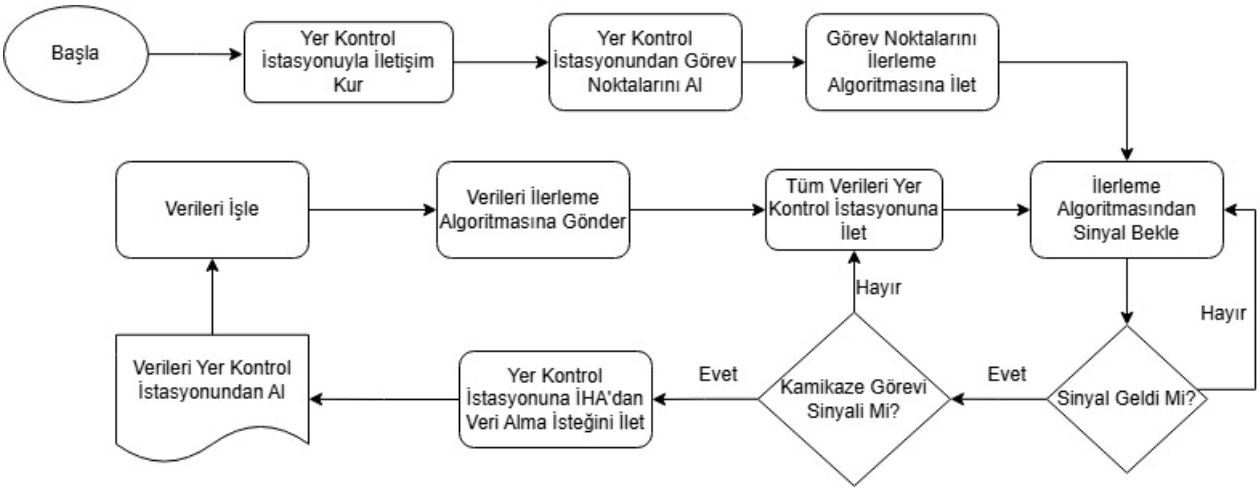
3.2.3. Çarpışma Kontrol Algoritması

- İlk iki görevde aktiftir
- Haritada bulunmayan beklenmedik nesneleri algılar ve bunlardan kaçınır
- Tetiklenmesi, önceki aşamalarda bir hata olduğunu gösterir ve sistemin resetlenmesini gerektirir



3.2.4. Haberleşme Algoritması

- GCS ile tüm veri alışverişini yönetir
- Güç kesilene kadar sürekli veri gönderip alır
- USV, UAV ve GCS arasındaki senkronizasyonu sağlar



4. Proje Analizi

4.1. Test

Proje hem fiziksel hem de simülasyon ortamlarında test edilmiştir. Algoritmalar Unity (C#) üzerinde ve gerçek saha koşullarında test edilmiştir. Fiziksel testlere ait bir video disk içerisinde yer almaktadır.

Testler algoritmaların düzgün çalıştığını göstermiştir; ancak görüntü işleme tarafında bazı sorunlar tespit edilmiştir. Sistem gece kusursuz çalışırken gündüz bazı problemler yaşamıştır. Normalde bunun tersi beklenirken, sorunun deniz yüzeyinden yansıyan güneş ışığı olduğu sonradan anlaşılmıştır. Bu durum yarışma sırasında herhangi bir probleme yol açmamıştır.

4.2. Geliştirilebilir Alanlar

Sahada diğer rakipleri gördükten sonra Trieme'yi yapay zekâ modeliyle geliştirmek istedik. Ancak bunu iki kişilik bir yazılım ekibiyle yapmanın zor olacağını biliyorduk. Bu nedenle takım kaptanımızdan ek insan gücü talep ettik.

Algoritmalarımız açısından mesh sistemi bir noktada **overengineering** oldu. Bu kadar karmaşık bir yapıya gerek yoktu. Alternatif olarak orta noktalar belirlemeyi düşündük. Kenar takip algoritmaları ile harita sınırları oluşturulabilir ve bu sınırlar üzerinden orta noktalar belirlenerek ArduPilot'a çok daha basit bir çözüm sunulabilirdi.

5. Proje Yorumları

5.1. Geliştirici Yorumları

2024 yarışması bana birçok şeyin önemini öğretti. Bunlara; proje raporu hazırlamak, yoğun bir takvimde zamanı yönetmek, stresli ortamlarda çalışmak, deadline'lara uymak, yapay zekâ kullanımı, gömülü sistemlerde yazılım geliştirme, ekip içi iletişim ve bazen katılmadığın kararlar altında çalışmak da dahil.

Bu fırsat için hâlâ çok minnettarım. Hem kendimi hem de öğrencilerimi (Anatoli gibi) geliştirmeme imkân sağladı.

Hooah

– Famura

6. Credits

- Famura – Yazılım Departmanı Lideri
- Musab Kılıç – Yazılım Ekip Üyesi (Ağırlıklı olarak görüntü işleme)

7. Referanslar

- MathWorks. (n.d.). *Dijkstra's algorithm*. In *MATLAB & Simulink documentation*. Retrieved October 18, 2025, from https://www.mathworks.com/help/matlab/ref/graph.shortestpath.html
- OpenCV. (n.d.). *YOLO object detection with OpenCV*. Retrieved October 18, 2025, from https://docs.opencv.org/
- PyGame Community. (n.d.). *PyGame documentation*. Retrieved October 18, 2025, from https://www.pygame.org/docs/
- PyMavlink Developers. (n.d.). *pymavlink: Python MAVLink library*. Retrieved October 18, 2025, from https://www.ardusub.com/developers/pymavlink.html
- Python Software Foundation. (n.d.). *Python 3 documentation*. Retrieved October 18, 2025, from https://docs.python.org/3/
- Raspberry Pi Foundation. (n.d.). *Raspberry Pi OS documentation*. Retrieved October 18, 2025, from https://www.raspberrypi.com/software/
- Robot Operating System (ROS). (n.d.). *ROS documentation*. Retrieved October 18, 2025, from https://www.ros.org/
- Serial Communication Library (PySerial). (n.d.). *PySerial documentation*. Retrieved October 18, 2025, from https://pyserial.readthedocs.io/
- Unity Technologies. (n.d.). *Unity documentation*. Retrieved October 18, 2025, from https://docs.unity.com/

8. Disk İçeriği

- Trieme.pdf
- Trieme.docx
- Ekip_Fotoğrafı.jpg
- Yarışma_Günü_Videosu.mp4
- Otonom_Testi.mp4
- Video_Mülakat_Videosu.mp4
- Yarışma_Önü_Test_Günü.mp4
- Let It Be by The Beatles