HAVACILIKTA YAPAY ZEKA YARIŞMASI ÖN TASARIM RAPORU

Takım Adı: AeroMA

Takım ID: 886794

Başvuru ID: 4811913

İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

KISALTMALAR

API: Uygulama Programlama Arayüzü (Application Programming Interface).

CUDA: Paralel Hesaplama Platformu ve Programlama Modeli (Compute Unified Device Architecture).

CuDNN: CUDA Derin Sinir Ağı Kütüphanesi (CUDA Deep Neural Network library).

FPS: Saniye Başına Kare Sayısı (Frames Per Second).

GPU: Grafik İşlem Birimi (Graphics Processing Unit).

İHA: İnsansız Hava Aracı.

IoU: Kesişimin Birleşime Oranı (Intersection over Union).

JSON: JavaScript Nesne Notasyonu (JavaScript Object Notation).

mAP: Ortalama Kesinlik Değerlerinin Ortalaması (mean Average Precision).

ONNX: Açık Sinir Ağı Değişimi (Open Neural Network Exchange).

ÖTR: Ön Tasarım Raporu.

SLAM: Eş Zamanlı Konumlandırma ve Haritalama (Simultaneous Localization and Mapping).

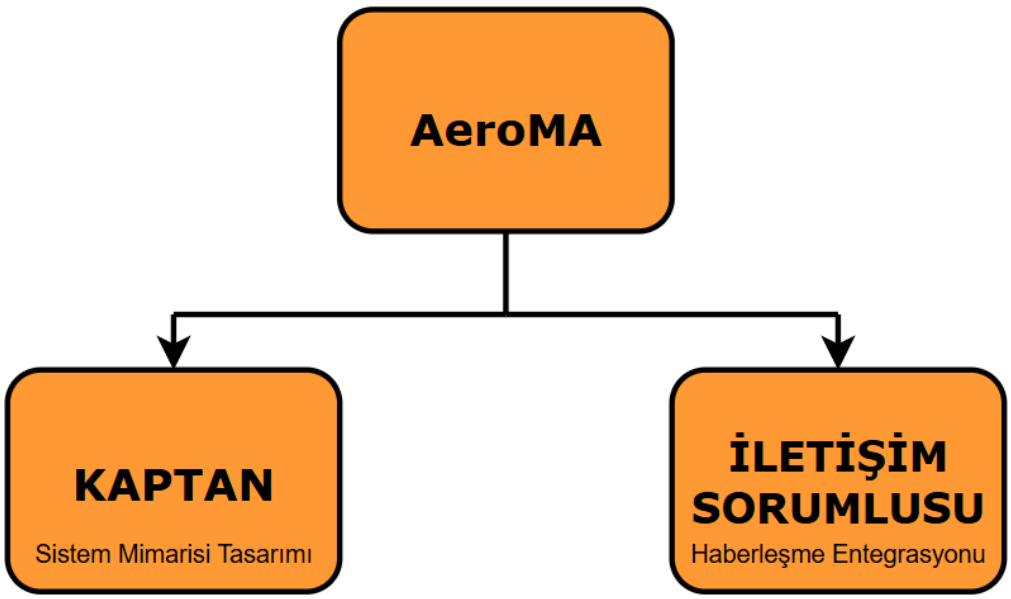
UAİ: Uçan Ambulans İniş Alanı.

UAP: Uçan Araba Park Alanı.

1. TAKIM ŞEMASI

Takımımızda işlerin hızlı yürümesi ve bilgisayarımızın gücünü en iyi şekilde kullanabilmek için iki ana gruba ayrıldık. Görev dağılımımızı, her birimizin uzmanlaşmak istediği alanlara göre şu şekilde planladık:

- **Takım Kaptanı:** Yapay zekâ modelimizin hangi mimariyi kullanacağına karar veriyor ve YOLO tabanlı modellerin eğitilmesinden sorumlu. Ayrıca, modelin yarışma bilgisayarındaki grafik işlemcide (GPU) çok daha hızlı çalışabilmesi için TensorRT ve ONNX gibi optimizasyon araçlarını kullanarak sistemin performansını en üst seviyeye çıkarıyor.
- **İletişim Sorumlusu / Üye 1:** Görüntü işleme kodlarını sisteme entegre ediyor ve yarışma sunucusuyla bizim bilgisayarımız arasındaki haberleşmeyi sağlayan JSON tabanlı API yapısını kuruyor. Aynı zamanda aracın konumunu tahmin eden görsel odometri (SLAM) mantığını geliştirip nesne takip algoritmalarını ana sisteme dahil ediyor.



Şekil 1: Takım Şeması

2. PROJE MEVCUT DURUM DEĞERLENDİRMESİ

Havacılıkta Yapay Zekâ Yarışması için başladığımız çalışmalarda, araştırma ve tasarım kısımlarını geride bıraktık ve artık sistemimizi hayata geçirdiğimiz prototip aşamasına geçtik. Şimdiye kadar yaptıklarımızı şöyle özetleyebiliriz:

- **Araştırma ve Algoritma Seçimi**: Yarışma görevlerini en iyi nasıl çözeriz diye literatürdeki güncel yöntemleri inceledik. Hem hızlı çalışması hem de doğruluğu yüksek olduğu için nesne tespitinde YOLOv11 mimarisini kullanmaya karar verdik. Hareketli araçları ayırt edebilmek için ise ByteTrack algoritmasının en uygun seçenek olduğunu gördük. [2]

- **Çalışma Ortamının Kurulması**: Modellerimizi eğitmek ve test etmek için gerekli olan CUDA ve PyTorch gibi teknik kütüphanelerin kurulumlarını tamamladık. Kendi bilgisayarlarımızda (RTX ekran kartlı sistemler ve MacBook M3) sistemin genel akışını ve haberleşme yapısını hatasız çalışacak şekilde hazırladık.

- **İlk Testler ve Veri Analizi**: TEKNOFEST'in paylaştığı eski veri setlerini kullanarak ilk denemelerimizi yaptık. Özellikle kameranın çok dik olduğu açılarda nesneleri tanımanın zorlaştığını fark ettik. Bu sorunu çözmek için "veri artırma" (augmentation) dediğimiz yöntemle modelimizi bu zor açılara karşı daha dayanıklı hale getirecek bir strateji belirledik.

- **Mühendislik Tasarımı**: Sistemimiz, havacılık yazılımlarının gerektirdiği yüksek hata toleransı (fault tolerance) ve düşük gecikme (low latency) prensiplerine göre "Mikroservis benzeri" modüler bir yapıda tasarlanmıştır. Görüntü okuma, YOLO tabanlı nesne tespiti ve SLAM tabanlı konum kestirimi görevleri aynı işlem döngüsünü tıkamaması adına *Multiprocessing/Multithreading* mimarisiyle birbirinden izole edilmiştir. Bu asenkron yapı sayesinde, herhangi bir modülde yaşanabilecek anlık bir gecikme veya çökme, diğer modüllerin (örneğin JSON API haberleşmesinin) çalışmasını engellemez. Ayrıca donanım kaynaklarının (GPU/CPU) en verimli şekilde kullanılması için yapay zeka modellerimiz ONNX formatına dönüştürülüp TensorRT ile hızlandırılarak uç cihaz (Edge AI) optimizasyonu sağlanmış ve sistemdeki darboğazlar (bottleneck) mühendislik düzeyinde çözülmüştür.

3. ALGORİTMALAR VE SİSTEM MİMARİSİ

3.1. Veri Setleri

Sistemimizin yağmurda, siste veya farklı uçuş irtifalarında yüksek kararlılıkla çalışabilmesi için geniş kapsamlı ve dengeli bir veri stratejisi izlenmektedir:

Temel Veri Kaynakları: Yarışma kurulunun paylaştığı uçuş videoları temel eğitim setimizi oluştururken, modelin genelleme yeteneğini artırmak için *VisDrone* ve *UAVDT* gibi açık kaynaklı İHA veri setleriyle çapraz besleme yapılacaktır.

- **Veri Artırma (Data Augmentation):** Havacılık dinamiklerine uygun olarak iki aşamalı veri artırma uygulanacaktır. * *Piksel Tabanlı:* Kamera sarsıntısı ve atmosferik bozulmaları simüle etmek için Gaussian Noise, Motion Blur ve Brightness Variations.

- **Uzamsal ve Sentetik:** Küçük nesnelerin (araçlar) tespit doğruluğunu artırmak için YOLO mimarisine entegre *Mosaic* ve *MixUp* teknikleri ile irtifa değişimlerini tolere edecek *Random Scale/Crop* işlemleri. Bu stratejilerle veri havuzumuzun 3 katına çıkarılması ve overfitting'in önlenmesi hedeflenmektedir.

- **Sınıf Dengesizliği ve Etiketleme:** Eğitim verilerindeki araç ve iniş alanı (UAP/UAİ) sayıları arasındaki dengesizliği gidermek amacıyla *Oversampling* ve azınlık sınıflara yüksek ceza katsayısı uygulayan kayıp fonksiyonları kullanılacaktır. Etiketler standart YOLO formatında normalize edilecek ve %80 Train, %10 Validation, %10 Test olarak izole edilecektir.

3.2. Algoritmalar

Yarışmadaki üç zorlu görevi de saniyede 7.5 kare (FPS) hızında, hiç takılmadan ve gerçek zamanlı yapabilmek için hem derin öğrenme modellerini hem de geleneksel görüntü işleme yöntemlerini bir araya getirdiğimiz hibrit bir yapı kurduk:

- **Birinci Görev (Nesne Tespiti ve Durum Analizi):** Hedeflerin yüksek doğruluk ve düşük gecikme ile tespiti için derin öğrenme tabanlı YOLOv11 mimarisi entegre edilmiştir. Sınıflandırma ve yer tespiti hatalarını minimize etmek için kayıp fonksiyonu olarak CIoU (Complete Intersection over Union) kullanılmıştır. Tespit edilen nesnelerin otonom takibi ve hareket analizi için algoritmik boru hattımıza (pipeline) ByteTrack entegre edilmiştir. Sistem, her bir nesnenin kinematik durumunu Kalman Filtresi ile modellemekte ve ardışık karelerdeki (frames) tespitleri Macar Algoritması (Hungarian Algorithm) yardımıyla eşleştirmektedir. "Hareketli/Hareketsiz" ayrımı; kameranın kendi ego-hareketinden kaynaklanan optik akış (optical flow) gürültüsü filtrelendikten sonra, nesnenin ağırlık merkezinin (centroid) zaman içindeki vektörel yer değiştirmesi analiz edilerek yapılmaktadır. [1][2]

- **İniş Durumu Analizi (UAP/UAİ):** Havacılıkta hata payına yer olmadığı için iniş alanlarının uygunluğunu sadece yapay zekanın tahminine bırakmadık. Alanın koordinatları ile çevredeki nesnelerin koordinatları arasında bir **IoU (Kesişimin Birleşime Oranı)** hesabı yapıyoruz. Eğer o alanın içinde en ufak bir nesne bile varsa, sistemimiz kesin bir şekilde "İnişe Uygun Değil" uyarısı veriyor.

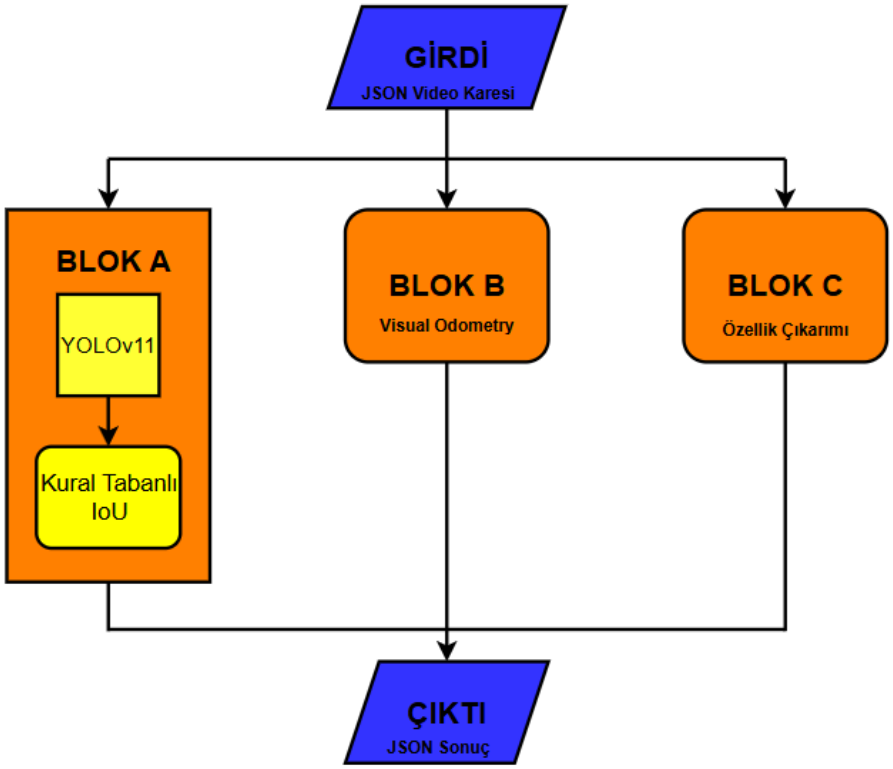
• **İkinci Görev (Pozisyon Kestirimi):** Yarışmada GPS sinyali kesilirse ne yapacağımızı da düşündük. Sadece kameradan gelen görüntüleri kullanarak aracın ne kadar yol aldığını hesaplayan **Görsel Odometri (Deep Visual Odometry)** yöntemini kullanıyoruz. Kareler arasındaki piksel kaymalarını matematiksel olarak hesaplayıp aracın X, Y ve Z eksenindeki konumunu tahmin ediyoruz.[4]

• **Üçüncü Görev (Referans Obje Eşleme):** Yarışma anında bize ilk defa gösterilecek olan nesneleri tanımak için modelimizi yeniden eğitmekle vakit kaybetmeyeceğiz. Bunun yerine **SuperGlue** veya **SIFT** gibi yöntemlerle nesnenin dokusunu analiz edip eşleştirme yapacağız. Bu sayede termal kameradan baksak bile nesneyi şak diye tanıyabileceğiz. [3]

3.3. Akış Şeması

Bu kısımda yarışma görevlerini gerçek zamanlı (7.5 FPS) ve kesintisiz gerçekleştirebilmek için kurguladığımız sistemin asenkron çalışma akışı **Şekil 1**'de ifade edilmektedir.

Sistemimiz sunucudan bir görüntü karesi aldığı anda onu üç kola ayırıyor: **Blok A** nesneleri bulup iniş uygunluğunu kontrol ediyor, **Blok B** aracın o an nerede olduğunu tahmin ediyor, **Blok C** ise yeni nesneleri eşleştiriyor. En sonunda bu üç koldan gelen bilgiler birleşip tek bir JSON paketi olarak sunucuya geri gidiyor.



Şekil 2: Sistemin Asenkron İşlem Mimarisi ve Genel Akış Şeması

4. ÖZGÜNLÜK

Biz bu projeyi tasarlarken sadece hazır yapay zekâ modellerini alıp çalıştırmayı değil, havacılıkta karşımıza çıkacak gerçek sorunlara mühendislik çözümleri üretmeyi hedefledik. AeroMA ekibi olarak projemizde fark yarattığını düşündüğümüz ve bizi diğerlerinden ayıracak özgün yaklaşımlarımız şunlardır:

- **Uçağın Hareketinden Kaynaklanan "Yalancı Hareketlerin" Ayıklanması:** Projeye başladığımızda fark ettiğimiz en büyük sorun şuydu: Uçağın altındaki kamera sürekli hareket ettiği için yerdeki sabit duran arabalar bile sanki hareket ediyormuş gibi görünüyor. Eğer sadece nesne tespiti yaparsak sistem burada yanılabilir. Biz buna çözüm olarak sadece o anki görüntüye bakmak yerine, **ByteTrack** algoritmasını kullanarak nesnelerin zaman içindeki gidişatını (vektörlerini) takip ediyoruz. Uçağın kendi gidiş hızını hesaba katan bir filtreleme katmanı ekledik. Böylece sistemimiz; "Bu araba gerçekten mi gidiyor yoksa uçak ilerlediği için mi öyle görünüyor?" sorusuna doğru cevabı veriyor. Bu yöntemle gereksiz hataların önüne geçiyoruz.

- **IoU Tabanlı Kesin (Deterministik) Çarpışma ve İniş Kontrolü:** Uçan Araba Park (UAP) ve Uçan Ambulans İniş (UAİ) alanlarının inişe uygunluk analizi, derin öğrenme modelinin hata payı barındıran sınıflandırmasına bırakılmamıştır. Özgün bir mimariyle, tespit edilen UAP/UAİ sınırlayıcı kutuları (bounding box) ile ortamdaki diğer nesnelerin kutuları arasında uzamsal **IoU (Kesişimin Birleşime Oranı)** matrisi hesaplanmaktadır. Alan ile herhangi bir nesne arasında en ufak bir örtüşme (overlap) tespit edildiğinde sistem deterministik olarak "İnişe Uygun Değil" bayrağı kaldırmakta ve havacılıktaki "Sıfır Hata" (Zero Tolerance) prensibini yazılımsal olarak garanti altına almaktadır.

- **"Zero-Shot" Dinamik Nesne Adaptasyonu:** Yarışma anında bize daha önce hiç eğitmediğimiz, ne olduğunu bilmediğimiz yeni nesneler gösterilebilir. Normalde bir nesneyi tanımak için binlerce fotoğrafla eğitim yapmak gerekir ama bizim o an böyle bir vaktimiz olmayacak. Bu yüzden projemizde **SuperGlue ve SIFT** gibi doku analizine dayanan yöntemleri kullandık. Bu yöntemler sayesinde sistemimiz nesnenin ne olduğunu bilmese bile, bize verilen referans fotoğraftaki desenleri ve köşeleri anlık olarak uçuş görüntüsüyle eşleştiriyor. Bu sayede sistemi baştan eğitmemize gerek kalmadan, farklı açılardan veya termal kameradan baksak bile yeni nesneleri "şak" diye tanıyabiliyoruz.

5. PROJE TAKVİMİ

Proje kapsamında hedeflenen kilometre taşları, yarışma takvimine uygun olarak geliştirme, entegrasyon ve optimizasyon aşamalarını içerecek şekilde aşağıdaki gibi planlanmıştır:

- **Şubat-Mart 2026 (Tamamlandı):** Yarışma görevlerinin analizi, literatür taraması ve sistem mimarisinin teorik altyapısının oluşturulması.
- **Nisan 2026 (Tamamlanıyor):** Geliştirme ortamlarının (CUDA destekli donanım hızlandırıcıları ve ARM tabanlı test ortamları) kurulması, Ön Tasarım Raporu'nun (ÖTR) hazırlanarak teslim edilmesi.
- **Mayıs 2026:** Açık kaynaklı İHA verileri (VisDrone) ile yarışma verilerinin birleştirilerek hibrit veri setinin oluşturulması ve ilk YOLOv11 model eğitimlerinin başlatılması.
- **Haziran 2026:** ByteTrack nesne takip algoritması ile IoU tabanlı iniş kontrol mantığının koda dökülmesi ve görsel odometri (SLAM) iskeletinin tasarlanması.
- **Temmuz 2026:** Çevrimiçi Yarışma Simülasyonu için sistemin kapalı ağda test edilmesi ve sunucu haberleşme (JSON/API) protokollerinin doğrulanması.
- **Ağustos 2026: Sistem Entegrasyon Kampı (Sprint):** Ayrı olarak geliştirilen donanım ve yazılım modüllerinin asenkron mimaride birleştirilmesi. Farklı hava koşulları (sis, yağmur simülasyonu) altında stres testlerinin yapılması ve Final Tasarım Raporu'nun teslimi.
- **Eylül-Ekim 2026:** TEKNOFEST Finalleri, donanım kurulumlarının yarışma alanında yapılması ve teknik sunumun gerçekleştirilmesi.

6. SONUÇLAR VE İNCELEME

Projenin ön tasarım aşamasında gerçekleştirilen testler ve bu testler sonucunda elde edilen incelemeler, sistemin final performansını maksimize etmek üzere aşağıda özetlenmiştir:

6.1. Ön Test Senaryoları ve Gözlemler: Geçmiş TEKNOFEST verileri üzerinde yapılan ön testlerde, kamera açısının dikleştiği durumlarda nesnelerin perspektif bozulmasına uğradığı tespit edilmiştir. Bu sorunun üstesinden gelmek için veri seti yatay, dikey ve çapraz uçuş açılarını içerecek şekilde çeşitlendirilmiştir. Uygulanan yöntemlerin sonuçları, açılı çekimlerde model doğruluğunun (mAP) artırılması için veri artırma (augmentation) işlemlerinin zorunlu olduğunu göstermiştir.

- **Baseline Analizi:** Özgün model eğitimine referans olması amacıyla literatürdeki standart YOLOv11 mimarisi incelenmiştir. Mevcut çalışmalar, 70-90 derecelik dik açılı hava görüntülerinde standart modellerin mAP performansının %40 seviyelerine düşebileceğini göstermektedir. Bu analiz, bizim perspektif odaklı veri artırma stratejimizin başarımız için kritik olduğunu kanıtlamaktadır.

6.2. Uygulanacak Stres Testi Senaryoları: Sistemimizin sadece güneşli ve açık havalarda değil, en kötü şartlarda bile görevini yapabilmesini istiyoruz. Bunun için sistemimizi şu zorluklarla sınayacağız:

- **Kötü Hava Koşulları:** Eğitim verilerimize bilerek karıncalanma (noise) ve hareket bulanıklığı (motion blur) ekliyoruz. Böylece uçak ani bir dönüş yapsa veya görüntü bir anlığına bozulsa bile modelimiz "ben bu görüntüyü daha önce görmüştüm" diyebilecek.

- **Gözden Kaybolma Durumlar:** Nesnelerin önüne bazen bir ağaç, bazen de bir binanın gölgesi girebilir. ByteTrack algoritmamızı burada test ediyoruz; nesne bir anlığına görünmez olsa bile sistemimiz ona verdiği numarayı (ID) unutmayacak ve nesne tekrar göründüğünde kaldığı yerden takibe devam edecek.

6.3. Yaşanan Sorunlar, Hatalar ve Çözüm Yaklaşımları

- **Yalancı Hareket Sorunu:** Uçak giderken yerdeki duran arabalar sanki hızla gidiyormuş gibi görünüyordu. Bu bizim için en büyük teknik engeldi. Bunu çözmek için uçağın gidiş hızını, kameradan gördüğümüz nesne hızıyla matematiksel olarak dengeleyen bir filtreleme yöntemi kullandık. Bu sayede sadece uçağın değil, nesnenin kendi hızını ölçmeyi başardık.

- **Performans Darboğazı:** Derin öğrenme modeli ile özellik eşleme (SLAM) algoritmalarının aynı anda çalışmasının FPS düşüşüne neden olabileceği gözlemlenmiştir. Çözüm olarak asenkron mimari tasarlanmış ve donanım hızlandırıcısı (TensorRT) kullanılarak 7.5 FPS kısıtlamasının rahatlıkla aşılması planlanmıştır. Elde edilen tecrübeler, donanım kaynaklarının paralel iş parçacıkları (multithreading) ile yönetilmesinin kritik olduğunu göstermiştir. Yapılan teorik hesaplamalara göre, asenkron mimari ve TensorRT kullanımıyla ana döngünün kare başına 80-100 ms işlem süresine ineceği ve saniyede minimum 10 FPS hızına ulaşılarak sistemin darboğaza girmeyeceği öngörülmektedir.

7. KAYNAKÇA

Bu bölümde raporda kullanılan kaynaklara yer verilir.

Tablo 1: Kaynakça

[1] Sapkota, R., Flores-Calero, M., Qureshi, R., Badgujar, C., Nepal, U., Poulose, A., ... & Karkee, M. (2025). YOLO advances to its genesis: A decadal and comprehensive review of the You Only Look Once (YOLO) series. *Artificial Intelligence Review*, *58*(9), 274.

[2] Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., ... & Wang, X. (2022, October). Bytetrack: Multi-object tracking by associating every detection box. In *European conference on computer vision* (pp. 1-21). Cham: Springer Nature Switzerland.

[3] Sarlin, P. E., DeTone, D., Malisiewicz, T., & Rabinovich, A. (2020). Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4938-4947).

[4] Scaramuzza, D., & Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE robotics & automation magazine*, *18*(4), 80-92.

[5] TEKNOFEST. (2026). Havacılıkta yapay zeka yarışması şartnamesi ve teknik şartnamesi.

[6] Ultralytics. (Tarih Yok). *YOLOv11 documentation and application architecture*. Erişim adresi: https://docs.ultralytics.com/

[7] Yousif, K., Bab-Hadiashar, A., & Hoseinnezhad, R. (2015). An overview to visual odometry and visual SLAM: Applications to mobile robotics. *Intelligent Industrial Systems*, *1*(4), 289-311.