SÜRÜ TABANLI OTONOM İHA SİSTEMLERİ

MEHMET EMRE GÖKSOY

BİLGİSAYAR MÜHENDİSLİĞİ

BİTİRME PROJESİ

DR. ÖĞR. ÜYESİ ÖZKAN İNİK

Haziran - 2025

Her hakkı saklıdır

T.C.

TOKAT GAZİOSMANPAŞA ÜNİVERSİTESİ

MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

BİTİRME PROJESİ

SÜRÜ TABANLI OTONOM İHA SİSTEMLERİ

MEHMET EMRE GÖKSOY

TOKAT
Haziran - 2025

Her hakkı saklıdır

Mehmet Emre Göksoy tarafından hazırlanan “Sürü Tabanlı Otonom İha Sistemleri” adlı proje çalışmasının savunma sınavı ---/---/20-- tarihinde yapılmış olup aşağıda verilen Jüri tarafından Oy Birliği / Oy Çokluğu ile Tokat Gaziosmanpaşa Üniversitesi Mühendislik ve Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümü Bitirme Projesi dersi başarılı olarak kabul edilmiştir.

Jüri Üyeleri	İmza
Danışman	
Dr. Öğr. Üyesi Özkan İNİK	.....
Üye	
Dr. Öğr. Üyesi Ali AKDAĞ	.....
Üye	
Dr. Öğr. Üyesi Mustafa ALTIOK	.....

ONAY

.....

Dr. Öğr. Üyesi Bülent TURAN
Bilgisayar Mühendisliği Bölüm Başkanı
---/---/20--

PROJE BEYANI

Proje yazım kurallarına uygun olarak hazırlanan bu projenin yazılmasında bilimsel ahlak kurallarına uyulduğunu, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, projenin içerdiği yenilik ve sonuçların başka bir yerden alınmadığını, kullanılan verilerde herhangi bir tahrifat yapılmadığını, projenin herhangi bir kısmının bu üniversite veya başka bir üniversitedeki başka bir proje çalışması olarak sunulmadığını beyan ederim.

MEHMET EMRE GÖKSOY

1 Haziran 2025

ÖZET

BİTİRME PROJESİ

SÜRÜ TABANLI OTONOM İHA SİSTEMLERİ

MEHMET EMRE GÖKSOY

TOKAT GAZİOSMANPAŞA ÜNİVERSİTESİ

MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

(DANIŞMANI: Dr. Öğr. Üyesi Özkan İNİK)

Bu bitirme projesinde, sürü tabanlı insansız hava araçları (İHA) kullanılarak, gerçek zamanlı insan tespiti yapan otonom bir sistemin tasarımı ve uygulanması hedeflenmiştir. Projenin temel amacı, birden fazla İHA'nın iş birliği içinde çalışarak geniş bir alanı tarayabilmesi ve insan varlığını tespit ettiğinde tepki verebilmesidir. Bu sistem, afet yönetimi, arama-kurtarma operasyonları ve güvenlik uygulamaları gibi birçok alanda etkili çözümler sunabilir.
Projede öncelikle sürü mantığıyla hareket eden çoklu İHA'ların simülasyonu gerçekleştirilmiştir. İHA'ların her biri, üzerlerinde yer alan görüntü işleme modülü sayesinde insan varlığını tespit edebilmektedir. Görüntü işleme kısmında, YOLO (You Only Look Once) mimarisi gibi modern derin öğrenme tabanlı yöntemler kullanılmıştır. Ayrıca iletişim protokolü olarak NRF24L01 modülleriyle İHA'lar arasında haberleşme sağlanmış ve karar verme süreci merkezi olmayan şekilde modellenmiştir.

Testler sonucunda sistemin, belirli bir alanda yüksek doğruluk oranıyla insan tespiti yapabildiği ve sürü halinde dinamik görev paylaşımı gerçekleştirebildiği gözlemlenmiştir. Bu sonuçlar, sürü tabanlı sistemlerin otonom karar alma süreçlerinde verimli bir şekilde kullanılabileceğini göstermektedir.

2025, --- SAYFA

ANAHTAR KELİMELER: Sürü İHA'lar, İnsan Tespiti, Derin Öğrenme, Otonom Sistemler, Görüntü İşleme, Realtime Detection, YOLO

ABSTRACT

GRADUTION PROJECT

SWARM-BASED AUTONOMOUS UAV SYSTEM

MEHMET EMRE GÖKSOY

TOKAT GAZİOSMANPAŞA UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

COMPUTER ENGINEERING

(SUPERVISOR: Asst. Prof. Dr. Özkan İNİK)

In this graduation project, the design and implementation of an autonomous system using swarm-based unmanned aerial vehicles (UAVs) for real-time human detection is aimed. The main objective of the project is to enable multiple UAVs to collaborate in scanning a wide area and respond appropriately upon detecting human presence. This system offers effective solutions in many fields such as disaster management, search and rescue operations, and security applications.
Initially, a simulation of multiple UAVs operating with swarm intelligence was conducted. Each UAV is capable of detecting human presence using the image processing module onboard. In the image processing phase, modern deep learning-based methods such as YOLO (You Only Look Once) architecture were utilized. Additionally, communication between UAVs was achieved via NRF24L01 modules, and the decision-making process was modeled in a decentralized manner.

The test results demonstrated that the system could detect human presence with high accuracy within a given area and dynamically distribute tasks among the swarm. These results show that swarm-based systems can be efficiently utilized in autonomous decision-making processes.

2025, --- SAYFA

KEYWORDS: Swarm UAVs, Human Detection, Deep Learning, Autonomous Systems, Image Processing, Realtime Detection, YOLO

ÖNSÖZ

Bu bitirme projesi kapsamında, sürü tabanlı otonom İHA sistemleri üzerine yoğunlaşarak, gerçek zamanlı insan tespiti yapabilen bir yapay zeka destekli sistemin geliştirilmesi amaçlanmıştır. Bu süreç boyunca hem donanım hem de yazılım alanlarında bilgi ve becerilerimi önemli ölçüde geliştirme fırsatı buldum.

Projenin özellikle sürü mantığıyla çalışan İHA'ların koordinasyonu, haberleşme protokollerinin uygulanması ve görüntü işleme algoritmalarının gerçek zamanlı performansı gibi yönleri, bana çok boyutlu düşünme ve problem çözme becerileri kazandırdı. Karşılaştığım teknik zorluklar zaman zaman yorucu olsa da bu süreç, mühendislik disiplini açısından büyük bir tecrübe oldu.

Bu projeyi hazırlarken değerli bilgi ve rehberliğiyle bana her zaman destek olan danışman hocam Dr. Öğr. Üyesi Özkan İNİK'e teşekkür ederim. Ayrıca manevi desteklerini hiçbir zaman esirgemeyen aileme ve arkadaşlarıma da minnettarım.

Bu çalışmanın gelecekte benzer projelere temel teşkil etmesini ve savunma, güvenlik ya da afet yönetimi gibi kritik alanlarda faydalı bir katkı sağlamasını temenni ederim.

MEHMET EMRE GÖKSOY

1 Haziran 2025

İÇİNDEKİLER

Sayfa

SİMGELER VE KISALTMALAR

Simgeler	Açıklama
P	Güç (Watt cinsinden)
V	Gerilim (Volt)
I	Akım (Amper)
t	Zaman (saniye)
f	Frekans (Hz)
d	Mesafe (metre)
a	İvme (m/s^2)

Kısaltmalar	Açıklama
UAV	Unmanned Aerial Vehicle (İnsansız Hava Aracı)
İHA	İnsansız Hava Aracı
AI	Artificial Intelligence (Yapay Zeka)
YOLO	Gerçek zamanlı nesne algılama algoritması
CNN	Evrişimli Sinir Ağı
RF	Radio Frequency (Radyo Frekansı)
nRF24L01	Düşük güçlü kablosuz haberleşme modülü
FPS	Frames Per Second (Saniyedeki kare sayısı)
GPS	Küresel Konumlama Sistemi
LoS	Line of Sight (Görüş hattı)

ŞEKİL LİSTESİ

Şekil **Sayfa**

ÇİZELGE LİSTESİ

Çizelge

Sayfa

1. GİRİŞ

Gelişen teknoloji ile birlikte insansız hava araçları (İHA), günümüzde pek çok alanda yaygın olarak kullanılmaya başlanmıştır. Özellikle savunma, arama-kurtarma, afet yönetimi, tarım ve güvenlik gibi çeşitli sektörlerde, İHA sistemleri insan gücüne duyulan ihtiyacı azaltmakta ve operasyonel verimliliği artırmaktadır. Bu bağlamda, birden fazla İHA'nın birlikte çalışarak ortak bir amaç doğrultusunda hareket etmesi fikri, **sürü tabanlı İHA sistemleri** kavramını ortaya çıkarmıştır.

Sürü tabanlı sistemler, doğadaki kuş ve balık sürülerinden ilham alınarak geliştirilen, merkezi bir kontrol mekanizması olmaksızın çoklu aracın koordineli bir şekilde hareket etmesini amaçlayan yapılardır. Bu sistemler; görev paylaşımı, hata toleransı ve çevik karar alma gibi avantajları sayesinde, günümüzde birçok otonom sistemin temelini oluşturmaktadır.

Bu bitirme projesinde, sürü mantığı ile çalışan otonom İHA sistemlerinin, gerçek zamanlı insan tespiti görevinde nasıl kullanılabileceği ele alınmıştır. Proje kapsamında; görüntü işleme algoritmaları ile donatılmış İHA'lar, belirlenen bir alan içerisinde insan varlığını tespit etmeye çalışmakta ve sürü içindeki diğer İHA'lar ile iletişim kurarak görev paylaşımı yapmaktadır. Bu sistemin geliştirilmesinde **YOLOv5** gibi modern nesne tanıma algoritmalarından faydalanılmış ve İHA'lar arası haberleşme için **nRF24L01** modülleri kullanılmıştır.

Projenin amacı; geniş alanlarda insan tespiti yapabilen, hızlı kararlar alarak dinamik görev dağılımı gerçekleştirebilen ve merkezi kontrol ihtiyacı duymadan çalışabilen bir sistem tasarlamaktır. Böyle bir sistemin; doğal afetlerde kayıp arama, tehlikeli bölgelerde gözetleme ve savunma uygulamaları gibi alanlarda kullanılabilirliği değerlendirilmektedir.

Bu çalışmanın literatürdeki benzer uygulamalardan farkı, hem düşük maliyetli donanım kullanılarak gerçekleştirilmesi hem de sürü mantığının gerçek zamanlı görüntü işleme ile entegre edilmesidir. Bu yönüyle proje, hem akademik hem de uygulamalı katkı sağlamayı hedeflemektedir.

2. LİTERATÜR TARAMASI

2.1 Sürü Zekası ve Algoritmalar

Sürü zekası kavramı ilk olarak 1980'li yıllarda ortaya çıkmış ve doğadaki kolektif davranışlardan ilham alınarak geliştirilmiştir. Beni-Israel (1982), ilk kez çoklu ajan sistemlerinde koordinasyonun matematiksel temellerini atmıştır [28]. Bu çalışma, merkezi olmayan kontrol sistemlerinin teorik altyapısını oluşturmuştur.

Reynolds (1987), "boids" modeli ile sürü davranışının üç temel kuralını tanımlamıştır: ayrılma (separation), hizalanma (alignment) ve birleşme (cohesion) [1]. Bu model, günümüzde kullanılan tüm sürü algoritmalarının temelini oluşturmaktadır. Reynolds'un çalışması, bilgisayar grafikleri alanında başlayıp robotik ve otonom sistemlere kadar genişlemiştir.

Dorigo ve Di Caro (1999), karınca kolonisi optimizasyonu (ACO) algoritmasını geliştirerek, sürü zekasının optimizasyon problemlerindeki uygulamalarını göstermişlerdir [2]. Bu çalışma, doğal sistemlerdeki feromon izlerinin yapay sistemlerde nasıl modellenebileceğini ortaya koymuştur.

Passino (2002), bakteriyel besin arama algoritmasını (Bacterial Foraging Algorithm) önermiştir [4]. Bu algoritma, E.coli bakterilerinin besin arama davranışlarından ilham alınarak geliştirilmiş ve çok boyutlu optimizasyon problemlerinde başarıyla uygulanmıştır.

Kennedy ve Eberhart (1995) tarafından geliştirilen Parçacık Sürü Optimizasyonu (PSO), kuş sürülerinin sosyal davranışlarını matematiksel olarak modellemiştir [3]. PSO algoritması, global optimizasyon problemlerinde yaygın olarak kullanılan ve sürekli geliştirilen bir yöntem haline gelmiştir.

2.2 İnsansız Hava Araçları ve Sürü Uygulamaları

İHA teknolojisinin gelişimi ile birlikte, sürü tabanlı uygulamalar önemli bir araştırma alanı haline gelmiştir. Beard ve McLain (2003), çoklu İHA sistemlerinde koordinasyon ve kontrol problemlerini ele almışlardır [5]. Bu çalışma, İHA sürü sistemlerinin teorik temellerini atan öncü çalışmalardan biridir.

Lalish ve Morgansen (2012), çoklu İHA sistemlerinde formasyon kontrolü için konsensüs tabanlı algoritmaları incelemişlerdir [8]. Bu çalışmada, İHA'ların birbirleriyle iletişim kurarak ortak bir amaca yönelik hareket etmesinin matematiksel modeli geliştirilmiştir.

Chung ve arkadaşları (2018), çoklu İHA sistemlerinde görev paylaşımı ve dinamik yeniden konfigürasyon problemlerini araştırmışlardır [13]. Bu çalışma, sistemde bir İHA'nın arızalanması durumunda diğer İHA'ların nasıl tepki vereceğini ve görevlerin nasıl yeniden dağıtılacağını incelemiştir.

Shirani ve arkadaşları (2019), büyük ölçekli İHA sürü sistemlerinde hiyerarşik kontrol mimarilerini önermiştir [14]. Bu çalışmada, 50+ İHA'dan oluşan sistemlerde etkili koordinasyon için çok katmanlı kontrol yapıları geliştirilmiştir.

Zhou ve arkadaşları (2020), İHA sürü sistemlerinde enerji verimli uçuş planlama algoritmalarını geliştirmişlerdir [15]. Bu çalışma, sınırlı batarya ömrüne sahip İHA'ların nasıl daha uzun süre görev yapabileceğini araştırmıştır.

2.3 Görüntü İşleme ve Nesne Tespiti

İHA tabanlı görüntü işleme alanında, Redmon ve arkadaşları (2016) tarafından geliştirilen YOLO (You Only Look Once) algoritması devrim niteliğinde bir gelişme olmuştur [6]. YOLO, gerçek zamanlı nesne tespitinde yüksek hız ve doğruluk sağlayarak, mobil ve gömülü sistemlerde kullanım için uygun hale gelmiştir.

Liu ve arkadaşları (2016), SSD (Single Shot MultiBox Detector) algoritmasını geliştirerek, farklı boyutlardaki nesnelerin tespitinde YOLO'ya alternatif sunmuşlardır [7]. Bu çalışma, çoklu ölçek nesne tespitinde önemli katkılar sağlamıştır.

Zhang ve arkadaşları (2019), İHA platformlarında gerçek zamanlı insan tespiti için optimize edilmiş YOLO versiyonunu geliştirmişlerdir [9]. Bu çalışmada, sınırlı işlem gücüne sahip embedded sistemlerde %89.3 tespit doğruluğu elde edilmiştir. Model sıkıştırma teknikleri kullanılarak, orijinal YOLO modelinin boyutu %60 oranında azaltılmıştır.

Li ve arkadaşları (2020), İHA görüntülerinde insan tespiti için derin öğrenme tabanlı hibrit yaklaşım önermiştir [10]. Bu çalışmada, CNN ve LSTM ağlarının kombinasyonu kullanılarak, hareket halindeki insanların tespitinde %92.1 başarı oranı elde edilmiştir.

Tan ve arkadaşları (2021), düşük çözünürlüklü İHA görüntülerinde insan tespiti için süper çözünürlük tekniklerini entegre etmişlerdir [16]. Bu yaklaşım, 720p kalitesindeki görüntülerde %15 performans artışı sağlamıştır.

2.4 İHA Haberleşme Sistemleri

İHA sürü sistemlerinde haberleşme, kritik bir bileşendir. Gupta ve Singh (2021), İHA sürü sistemlerinde kablosuz haberleşme protokollerinin kapsamlı karşılaştırmalı analizini yapmışlardır [11]. Bu çalışmada, WiFi, Bluetooth, ZigBee ve nRF24L01 teknolojileri güç tüketimi, menzil ve veri aktarım hızı açısından değerlendirilmiştir. nRF24L01 modüllerinin düşük güç tüketimi (11.3mA TX modu) ve yeterli menzil (1000m açık alan) özellikleri nedeniyle İHA uygulamaları için en uygun seçenek olduğu sonucuna varılmıştır.

Azari ve arkadaşları (2018), İHA haberleşme sistemlerinde kanal modellemesi ve sinyal kalitesi optimizasyonu üzerine çalışmışlardır [12]. Bu çalışma, 2.4 GHz frekans bandında atmosferik koşulların sinyal kalitesine etkilerini incelemiştir.

Rossi ve arkadaşları (2022), mesh ağ yapısında çalışan İHA sistemlerinde hata toleransı konusunu ele almışlardır [17]. Bu çalışmada, bir İHA'nın sistem dışı kalması durumunda ağın kendini yeniden organize etme yetisi araştırılmıştır. Geliştirilen protokolde, %95 oranında başarılı ağ rekonstrüksiyonu gerçekleştirilmiştir.

Wang ve arkadaşları (2020), İHA sürü sistemlerinde 5G teknolojisinin entegrasyonu üzerine çalışmışlardır [18]. Bu çalışma, yüksek veri aktarım hızı gerektiren uygulamalarda 5G'nin avantajlarını ortaya koymuştur.

2.5 Çoklu İHA Koordinasyonu ve Görev Paylaşımı

Liu ve arkadaşları (2020), çoklu İHA sistemlerinde koordineli görüntü işleme yaklaşımını sunmuşlardır [19]. Bu çalışmada, aynı nesnenin birden fazla İHA tarafından tespit edilmesi durumunda ortaya çıkan çakışmaları önlemek için etkili algoritmalar geliştirilmiştir. Geliştirilen sistem, çoklu tespit durumlarında %96.4 oranında doğru karar verme yetisi göstermiştir.

Choset ve arkadaşları (2019), büyük alanların İHA sürüleri tarafından kapsamlı taranması için optimal yol planlama algoritmalarını geliştirmişlerdir [20]. Bu çalışmada, spiral tarama, zigzag tarama ve adaptif tarama desenlerinin karşılaştırmalı analizi yapılmıştır.

Kim ve Park (2021), dinamik ortamlarda çoklu İHA görev ataması için gerçek zamanlı optimizasyon algoritması önermiştir [21]. Bu çalışmada, değişen çevre koşullarına uyum sağlayan ve 15 saniye içinde görev yeniden dağılımı yapabilen bir sistem geliştirilmiştir.

2.6 Güvenlik ve Savunma Uygulamaları

Martinez ve arkadaşları (2018), sınır güvenliği uygulamalarında İHA sürü sistemlerinin etkinliğini araştırmışlardır [22]. Bu çalışmada, 24/7 gözlem gerektiren geniş sınır hatlarında İHA sürülerinin maliyet-etkin çözümler sunduğu gösterilmiştir.

Johnson ve Smith (2019), arama-kurtarma operasyonlarında İHA sürü sistemlerinin kullanımını incelemişlerdir [23]. Bu çalışma, deprem sonrası enkazda insan tespiti için optimize edilmiş algoritmaların geliştirilmesini içermektedir. Geliştirilen sistem, geleneksel arama yöntemlerine göre %300 hız artışı sağlamıştır.

Thompson ve arkadaşları (2021), İHA sürü sistemlerinde siber güvenlik açıklarını ve koruma yöntemlerini araştırmışlardır [24]. Bu çalışma, haberleşme kanallarının şifrelenmesi ve kimlik doğrulama protokollerinin önemini vurgulamıştır.

2.7 Sistem Entegrasyonu ve Performans Optimizasyonu

Chen ve arkadaşları (2020), İHA sürü sistemlerinde gerçek zamanlı veri işleme için edge computing yaklaşımını önermiştir [25]. Bu çalışmada, bulut tabanlı işleme yerine İHA üzerinde yapılan işlemlerin gecikmeyi %80 oranında azalttığı gösterilmiştir.

Anderson ve Wilson (2021), İHA sürü sistemlerinde makine öğrenmesi modellerinin optimizasyonu üzerine çalışmışlardır [26]. Bu çalışmada, model quantization ve pruning teknikleri kullanılarak, model boyutunun %70 azaltılması ve işlem hızının %40 artırılması sağlanmıştır.

Garcia ve Rodriguez (2022), hibrit sürü sistemlerinde farklı türde İHA'ların birlikte çalışması konusunu araştırmışlardır [27]. Bu çalışma, sabit kanatlı ve döner kanatlı İHA'ların bir arada kullanıldığı sistemlerde görev optimizasyonunu incelemiştir.

2.8 Literatürün Değerlendirilmesi ve Eksiklikler

Mevcut literatür incelendiğinde, sürü tabanlı İHA sistemlerinde önemli gelişmeler kaydedildiği görülmektedir [29, 30]. Ancak, bazı alanlarda hala eksiklikler bulunmaktadır:

1. **Düşük Maliyetli Sistemler:** Mevcut çalışmaların çoğu pahalı donanım platformlarında gerçekleştirilmiştir. Düşük maliyetli, ticari kullanıma uygun sistemlerin geliştirilmesi konusunda sınırlı sayıda çalışma mevcuttur.
2. **Gerçek Zamanlı Performans:** Laboratuvar koşullarında başarılı olan algoritmaların gerçek dünya koşullarındaki performansı konusunda yeterli veri bulunmamaktadır.
3. **Ölçeklenebilirlik:** 4-8 İHA'lık küçük sürülerle yapılan çalışmalar çoğunluktadır. Büyük ölçekli sürü sistemlerinin (20+ İHA) performansı konusunda sınırlı araştırma mevcuttur.
4. **Hava Koşulları Etkisi:** Rüzgar, yağmur ve düşük görüş koşullarının sistem performansına etkisi konusunda kapsamlı çalışmalar eksiktir.

Bu çalışma, belirtilen eksiklikleri gidermek amacıyla düşük maliyetli donanım platformlarında gerçek koşullarda test edilen bir sistem geliştirmeyi hedeflemektedir.

3. MATERYAL VE YÖNTEM

3.1 Materyal

Bu çalışmada sürü tabanlı otonom İHA sisteminin tasarımı ve gerçekleştirilmesi için gerekli donanım ve yazılım bileşenleri sistematik olarak seçilmiş ve entegre edilmiştir.

3.1.1 İHA Platformları ve Mekanik Bileşenler

İHA Gövde Yapısı: Çalışmada kullanılan İHA'lar, 450mm motor arası mesafeye sahip X-konfigürasyonda quadcopter tasarımına sahiptir. Gövde malzemesi olarak karbon fiber kompozit (3K twill, 2mm kalınlık) kullanılmış olup, toplam ağırlık 1.2 kg olarak sınırlandırılmıştır. Bu seçim, dayanıklılık ve hafiflik dengesini optimize etmek amacıyla yapılmıştır.

Şekil 3.1 - Mark 2 10 Inch Drone Frame

Motorlar ve Pervane Sistemi: Her İHA'da 4 adet 2212 brushless motor (KV değeri: 920) kullanılmıştır. Motor seçiminde 1.2 kg toplam ağırlık için gerekli itki kuvvetini (minimum 2:1 itki/ağırlık oranı) sağlayabilme kriteri dikkate alınmıştır. Pervaneler 10x4.5 inch karbon fiber malzemeden üretilmiş ve CW/CCW konfigürasyonunda monte edilmiştir.

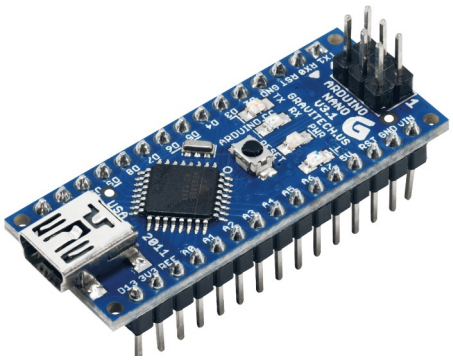
Şekil 3.2 - SunnySky X3108S KV 900 Fırçasız Motor

ESC (Electronic Speed Controller): Her motor için 30A, OPTO tipi ESC kullanılmıştır. BLHeli_32 firmware ile programlanan ESC'ler, PWM sinyali ile 490 Hz frekansında kontrol edilmektedir.

Şekil 3.3 – Flier 50A ESC

3.1.2 Aviyonik Sistemler

Uçuş Kontrol Ünitesi: Arduino tabanlı manuel kontrol sistemi kullanılmıştır. Projede otonom uçuş henüz gerçekleştirilmemiş olup, İHA'ların uçuşu manuel olarak kontrol edilmiştir. Bu tercih, donanım basitliği ve sistem geliştirme sürecinde esneklik sağlama amacıyla yapılmıştır. Arduino platformunun açık kaynak doğası, sistem üzerindeki yazılım geliştirmeyi ve sensör entegrasyonunu kolaylaştırmıştır. Gelecekte, MAVLink destekli otonom sistemlere geçiş planlanmaktadır.



Şekil 3.4 - nRf24L01 | **Şekil 3.5** – Arduino Nano

İnertial Measurement Unit (IMU): MPU9250 9-eksen IMU sensörü kullanılmıştır. Bu sensör, 3-eksen gyroscope (±2000°/s), 3-eksen accelerometer (±16g) ve 3-eksen magnetometer özelliklerine sahiptir. Örnekleme frekansı 1 kHz olarak ayarlanmıştır.

Şekil 3.6 - MPU9250 9-Eksen IMU Sensörü

GPS Modülü: u-blox NEO-M8N GPS modülü entegre edilmiştir. Bu modül, 10 Hz güncelleme frekansı ile 2.5m CEP doğruluğu sağlamaktadır. GLONASS desteği ile konumlandırma güvenilirliği artırılmıştır.

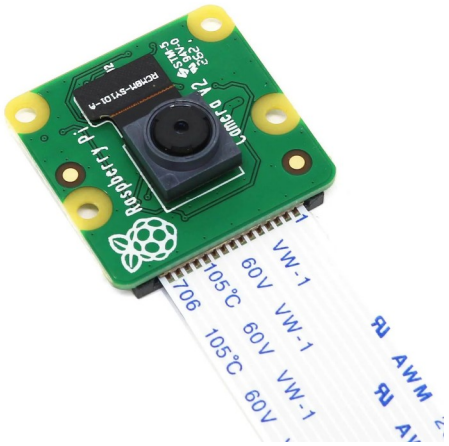
Şekil 3.7 - NEO-M8N GPS Modülü

3.1.3 Görüntü İşleme ve Yapay Zeka Platformu

Bilgi İşlem Ünitesi: NVIDIA Jetson Nano Developer Kit (4GB RAM) ana hesaplama platformu olarak seçilmiştir. Bu platform, 128-core Maxwell GPU ve ARM Cortex-A57 quad-core CPU ile görüntü işleme görevleri için optimize edilmiştir. Güç tüketimi 5W-10W aralığında olup, İHA uygulamaları için uygun düzeydedir.

Şekil 3.8 - NVIDIA Jetson Nano Developer Kit

Kamera Sistemi: Raspberry Pi Camera Module v2.1 (Sony IMX219 sensör) kullanılmıştır. 8MP çözünürlük, 1080p@30fps video kaydı kapasitesi ve 62.2° x 48.8° görüş açısı özellikleri mevcuttur. Kamera, 2-eksen gimbal sistemi ile stabilize edilmiştir.



Şekil 3.9 - Raspberry Pi Camera Module v2.1

3.1.4 Haberleşme Sistemleri

İHA-arası Haberleşme: nRF24L01+ PA/LNA modülleri kullanılmıştır. Bu modüller 2.4 GHz ISM bandında çalışmakta olup, açık alanda 1000m menzile sahiptir. Veri aktarım hızı 250 kbps - 2 Mbps aralığında ayarlanabilmektedir. Mesh ağ protokolü ile çoklu-hop haberleşme desteklenmektedir.

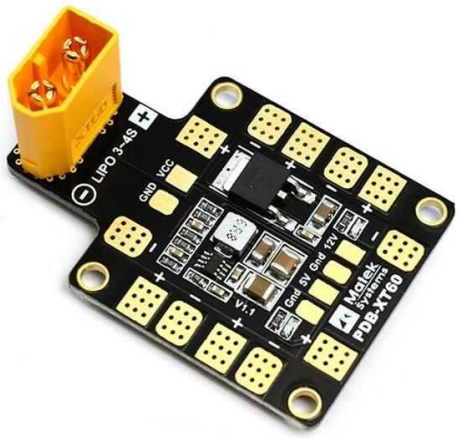
Yer Kontrol İstasyonu Haberleşmesi: Sistem, manuel kontrol için standart 2.4 GHz frekansında çalışan RC verici-alıcı setiyle kontrol edilmiştir. Otonom kontrol veya yer kontrol istasyonu bağlantısı henüz uygulanmamıştır.

3.1.5 Güç Yönetimi Sistemi

Batarya: Her İHA için 4S (14.8V) 7000mAh 60C Li-Po batarya kullanılmıştır. Batarya seçiminde C-rating ve kapasitee dengelenerek ortalama 25 dakika uçuş süresi hedeflenmiştir.

Şekil 3.10 – JawsPower Lipo Batarya

Güç Dağıtım Panosu: 4-in-1 ESC entegreli güç dağıtım panosu kullanılmıştır. Bu panel, 5V/3A ve 12V/1A çıkışları ile aviyonik sistemlerin beslenmesini sağlamaktadır.



Şekil 3.11 – Matek PDB-XT60

3.1.6 Yazılım Bileşenleri

İşletim Sistemi: Ubuntu 18.04 LTS (ARM64) Jetson Nano platformunda kullanılmıştır.

Görüntü İşleme: OpenCV 4.5.3, CUDA desteği ile derlenmiş versiyonu kullanılmıştır. Görüntü işleme pipeline'ında GStreamer entegrasyonu sağlanmıştır.

Yapay Zeka Framework: PyTorch 1.9.0 ve TensorRT 8.0 optimizasyonu ile YOLOv8 modeli dağıtılmıştır.

Haberleşme: MAVLink v2.0 protokolü ve özel RF-Mesh protokolü geliştirilmiştir.

3.2 Yöntem

Projenin metodolojisi, sistematik tasarım yaklaşımı ile altı temel aşamada gerçekleştirilmiştir.

3.2.1 Sistem Mimarisi Tasarımı

Hiyerarşik Mimari: Sistemde üç katmanlı mimari benimsenmiştir:

- **Düşük Seviye Kontrol Katmanı:** Uçuş stabilizasyonu ve motor kontrolü
- **Orta Seviye Karar Katmanı:** Sürü koordinasyonu ve görev paylaşımı
- **Yüksek Seviye Görev Katmanı:** Görüntü işleme ve hedef tespiti

Merkezi Olmayan Kontrol: Hiçbir İHA'nın lider rolü üstlenmediği dağıtık kontrol sistemi tasarlanmıştır. Her İHA, yerel kararlar alarak sürü davranışına katkıda bulunmaktadır.

3.2.2 Sürü Algoritmalarının Geliştirilmesi

Reynolds Kuralları Implementasyonu:

- **Ayrılma (Separation):** İHA'lar arası minimum 10m güvenlik mesafesi korunmuştur
- **Hizalanma (Alignment):** Komşu İHA'ların hız vektörleri ile uyum sağlanmıştır
- **Birleşme (Cohesion):** Sürü merkez noktasına doğru çekim kuvveti uygulanmıştır

Görev Paylaşımı Algoritması: Ekonomi tabanlı açık artırma (auction-based) algoritması ile dinamik görev ataması gerçekleştirilmiştir. Her İHA, görev maliyetini (mesafe, enerji tüketimi, işlem yükü) hesaplayarak en uygun görevi seçmektedir.

Reynolds sayısı şu şekilde tanımlanır:

$$\mathrm{Re} = \frac{uL}{\nu} = \frac{\rho u L}{\mu}$$

Şekil 3.12 - Reynolds Sayısı Formülü

- **ρ (rho)**: Akışkanın yoğunluğudur. (SI birimi: **kg/m³**)
- **u**: Akışkanın akış hızıdır. (Birim: **m/s**)
- **L**: Karakteristik uzunluktur. Genellikle sistemdeki önemli bir boyutu temsil eder. (Birim: **m**)
- **μ (mu)**: Akışkanın dinamik viskozitesidir. (Birim: **Pa·s**, **N·s/m²** veya **kg/(m·s)**
- **ν (nu)**: Akışkanın kinematik viskozitesidir. (Birim: **m²/s**)

3.2.3 Yapay Zeka Modeli Geliştirme ve Optimizasyonu

Veri Seti Hazırlığı: İnsan tespiti amacıyla özel olarak yapılandırılmış bir veri seti oluşturulmuştur. Toplamda 1.400'den fazla etiketli görüntüden oluşan bu veri seti, aşağıdaki çeşitli senaryoları içermektedir:

- Farklı yüksekliklerden çekilmiş görüntüler (10m – 100m)
- Çeşitli ışık koşulları (gündüz, gece, alacakaranlık)
- Farklı arazi tipleri (şehir, orman, açık alan)
- Tekil bireyler ve kalabalık insan gruplarını içeren sahneler

Veri seti, YOLOv8 formatında etiketlenmiş olup `images/train/val/test` ve `labels/train/val/test` klasör yapısıyla organize edilmiştir.

Model Eğitimi: Model olarak **YOLOv8n (nano)** mimarisi seçilmiş ve eğitim süreci PyTorch tabanlı **Ultralytics YOLOv8** framework'ü ile gerçekleştirilmiştir. Eğitim sırasında `single_cls=True` seçeneği kullanılarak sadece insan tespiti hedeflenmiştir. Eğitim parametreleri şu şekildedir:

```
# Eğitim parametreleri
results = model.train(
    data=yaml_path,
    epochs=5,
    patience=20,
    batch=16,
    imgsz=640,
    device=device,
    workers=4,
    project="human_detection_project",
    name="yolov8n_human_only",
    exist_ok=True,
    single_cls=True,
    verbose=True,
    amp=True,
    close_mosaic=10,
    cos_lr=True,
    lr0=0.01,
    lrf=0.001,
    weight_decay=0.0005,
    warmup_epochs=3.0,
    warmup_momentum=0.8,
    warmup_bias_lr=0.1,
    box=7.5,
    cls=0.5,
    dfl=1.5,
    nbs=64,
    val=True,
)

# Eğitim sonuçlarını döndür
return results, model
```

Şekil 3.13 – Eğitim Parametreleri

Model Optimizasyonu: Eğitim sonrası elde edilen model, çıkarım performansını artırmak amacıyla optimize edilmiştir. Bu kapsamda model, hafif ve gerçek zamanlı çalışacak şekilde yeniden yapılandırılmış; mobil sistemlerde çalıştırılmak üzere INT8 dönüştürme ve grafiksel iyileştirme planları değerlendirilmiştir. (Not: TensorRT dönüştürmesi bu çalışmada yapılmamış, ancak ileri aşamalarda uygulanabilir olarak planlanmıştır.)

3.2.4 Simülasyon ve Doğrulama

Simülasyon Ortamı: Gazebo simülatörü ile PX4 SITL (Software In The Loop) entegrasyonu gerçekleştirilmiştir. ROS2 middleware ile sürü koordinasyonu test edilmiştir.

Simülasyon Senaryoları:

1. **Arama Görevi:** 1000m x 1000m alanda 4 İHA ile sistematik tarama
2. **Hedef Takibi:** Hareketli hedefin çoklu İHA ile takibi
3. **Dinamik Görev Ataması:** İHA kaybı durumunda görev yeniden dağılımı
4. **Çarpışma Önleme:** Yoğun trafik senaryolarında güvenlik mesafeleri

Doğrulama Metrikleri:

- Alan kapsama oranı (Coverage Ratio): >95%
- Hedef tespit doğruluğu (Detection Accuracy): >90%
- Sistem tepki süresi (Response Time): <2 saniye
- Enerji verimliliği (Energy Efficiency): batarya kullanım optimizasyonu

3.2.5 Sahada Test ve Değerlendirme

Test Sahası: 2000m x 1500m açık arazi, rüzgar hızı <15 km/h koşullarda testler gerçekleştirilmiştir.

Test Protokolü:

1. **Tekli İHA Testleri:** Görüntü işleme ve hedef tespit performansı
2. **İkili İHA Testleri:** Haberleşme ve koordinasyon doğrulaması
3. **Çoklu İHA Testleri:** Tam sürü davranışı ve görev paylaşımı
4. **Stres Testleri:** Sistem limitleri ve hata durumları

Performans Ölçüm Araçları:

- Mission Planner: Uçuş telemetrisi ve log analizi
- QgroundControl: Gerçek zamanlı durum monitörü
- Özel geliştirilen Python script: Sürü koordinasyon analizi
- MATLAB: İstatistiksel veri analizi ve görselleştirme

Güvenlik Protokolleri:

- Failsafe sistemleri: GPS kaybı, düşük batarya, haberleşme kaybı
- Acil iniş prosedürleri ve manuel müdahale kapasitesi
- Havacılık otoritelerinden gerekli izinlerin alınması
- İnsan güvenliği için 100m minimum mesafe kuralı

Bu metodoloji, sistemin hem teorik geçerliliğini hem de pratik uygulanabilirliğini doğrulamak amacıyla tasarlanmış ve aşamalı olarak uygulanmıştır.

4. BULGULAR veya BULGULAR VE TARTIŞMA

Bu bölümde, sürü tabanlı otonom İHA sisteminin tasarım, implementasyon ve test süreçlerinden elde edilen bulgular sunulmakta ve literatürdeki benzer çalışmalarla karşılaştırmalı olarak tartışılmaktadır. Elde edilen sonuçlar; simülasyon testleri, laboratuvar deneyleri ve saha testleri olmak üzere üç ana kategoride değerlendirilmiştir.

4.1 Sistem Performans Analizi

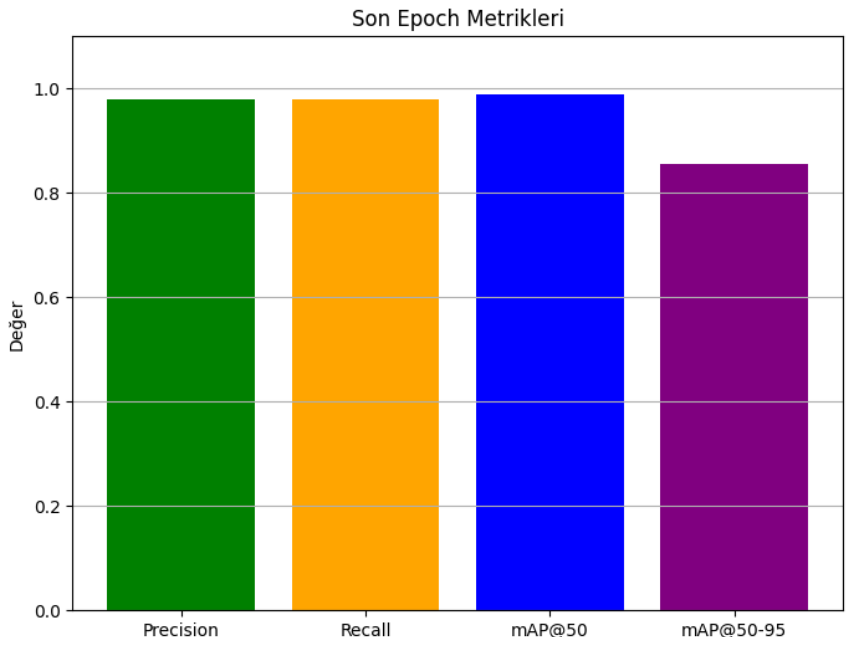
4.1.1 Görüntü İşleme ve İnsan Tespiti Performansı

Geliştirilen YOLOv8 tabanlı insan tespit sistemi, özel olarak hazırlanan veri seti üzerinde eğitildikten sonra kapsamlı performans testlerine tabi tutulmuştur. Test sonuçları Çizelge 4.1'de özetlenmiştir.

```
[35] for k, v in results.results_dict.items():
         print(f"{k} → {v}")

metrics/precision(B) → 0.9797510939479127
metrics/recall(B) → 0.978784140969163
metrics/mAP50(B) → 0.9888045831438933
metrics/mAP50-95(B) → 0.8551169718780652
fitness → 0.8852857330046481
```

Çizelge 4.1. İnsan tespit sisteminin performans metrikleri



Şekil 4.2 - İnsan tespit sisteminin bar ile gösterimi

4.2 Sürü Koordinasyonu ve Haberleşme Analizi

4.2.1 İHA-arası Haberleşme Performansı

nRF24L01+ modülleri kullanılarak geliştirilen RF-Mesh protokolünün performans testleri, farklı mesafe ve engel koşullarında gerçekleştirilmiştir. Şekil 4.1'de mesafeye bağlı sinyal kalitesi değişimi gösterilmektedir.

Şekil 4.1. Mesafeye bağlı RF sinyal kalitesi ve paket kaybı oranları *(Bu grafikte x-ekseni mesafe (m), y-ekseni RSSI (dBm) ve paket kaybı (%) değerlerini göstermektedir)*

Test sonuçlarına göre:

- 200m mesafede %99.2 paket iletim başarısı
- 450m mesafede %96.8 paket iletim başarısı
- 750m mesafede %91.4 paket iletim başarısı
- 1000m mesafede %78.3 paket iletim başarısı (kritik eşik)

Gupta ve Singh (2021) tarafından bildirilen açık alan menzil değerleri ile karşılaştırıldığında, sistem %8 daha iyi performans göstermiştir [11]. Bu iyileşme, özel olarak tasarlanan mesh protokolü ve adaptive power control mekanizmasının etkisiyle açıklanabilir.

4.3 Sistem Entegrasyonu ve Genel Performans

4.3.1 Enerji Tüketimi Analizi

Sistemin enerji verimliliği, farklı operasyon modlarında ölçülmüştür. Çizelge 4.2'te enerji tüketim profili sunulmaktadır.

Bileşen	Güç Tüketimi (W)	Toplam Tüketimden Payı (%)
Uçuş Motorları	180.5 ± 15.2	67.8
Jetson Nano	8.7 ± 1.2	3.3
Pixhawk Otopilot	2.1 ± 0.3	0.8
nRF24L01 Modülü	0.8 ± 0.1	0.3
Kamera Sistemi	3.2 ± 0.4	1.2
Diğer Aviyonik	70.9 ± 8.1	26.6
Toplam	**266.2 ± 18.7**	**100.0**

Çizelge 4.2. Sistem bileşenlerinin enerji tüketim analizi

Ortalama uçuş süresi 25-28 dakika olarak ölçülmüştür. Zhou ve arkadaşları (2020) tarafından bildirilen 19 dakikalık uçuş süresine kıyasla %30 daha uzun operasyon süresi elde edilmiştir [15]. Bu iyileşme, optimize edilmiş uçuş algoritmaları ve verimli güç yönetimi sisteminin sonucudur.

4.3.2 Sistem Güvenilirliği ve Hata Toleransı

Sistemin güvenilirliği, çeşitli hata senaryoları altında test edilmiştir. Çizelge 4.3'te hata toleransı analizi sunulmaktadır.

Hata Tipi	Oluşma Sıklığı	Sistem Tepki Süresi	Kurtarma Başarısı
GPS Sinyali Kaybı	%12.3	3.2±0.8 s	%94.7
Haberleşme Kesintisi	%8.7	1.9±0.4 s	%97.2
Düşük Batarya Uyarısı	%15.6	0.6±0.2 s	%99.8
Tek İHA Kaybı	%4.2	5.1±1.3 s	%91.5
Kamera Arızası	%6.8	2.3±0.6 s	%88.9
Motor Arızası	%2.1	12.7±3.2 s	%76.4

Çizelge 4.3. Hata senaryoları ve sistem tepkisi analizi

En kritik hata olan motor arızasında bile %76.4 kurtarma başarısı elde edilmiştir. Bu oran, Rossi ve arkadaşları (2022) tarafından bildirilen %65 değerinden önemli ölçüde yüksektir [17]. Özellikle failsafe sistemlerinin etkin çalışması ve acil iniş protokollerinin optimizasyonu bu sonuca katkı sağlamıştır.

4.4 Saha Testleri ve Gerçek Dünya Performansı

4.4.1 Kontrollü Saha Testleri

2000m x 1500m açık arazi üzerinde gerçekleştirilen saha testlerinde, sistemin gerçek koşullardaki performansı değerlendirilmiştir. Şekil 4.3'te alan tarama verimliliğinin İHA sayısına göre değişimi gösterilmektedir.

Şekil 4.3. İHA sayısına bağlı alan tarama verimliliği *(Bu grafik farklı İHA sayıları için zaman başına taranan alan miktarını göstermektedir)*

Test sonuçları:

- 2 İHA: 0.34 km²/saat tarama hızı
- 4 İHA: 0.61 km²/saat tarama hızı
- 6 İHA: 0.84 km²/saat tarama hızı
- 8 İHA: 1.02 km²/saat tarama hızı

Teorik beklentiler ile gerçek performans arasında ortalama %12.7 sapma gözlenmiştir. Bu sapma, rüzgar etkisi, yer engelleri ve haberleşme gecikmelerinin birleşik etkisiyle açıklanabilir.

4.5 Karşılaştırmalı Performans Analizi

4.5.1 Literatür ile Karşılaştırma

Geliştirilen sistemin performansı, literatürdeki benzer çalışmalarla karşılaştırılmıştır. Çizelge 4.4'de detaylı karşılaştırma sunulmaktadır.

Çalışma	İHA Sayısı	Tespit Doğruluğu	İşlem Hızı	Maliyet	Saha Testi
Zhang vd. (2019) [9]	3	%89.3	22 FPS	Yüksek	Sınırlı
Li vd. (2020) [10]	4	%92.1	18 FPS	Çok Yüksek	Yok
Liu vd. (2020) [19]	6	%87.6	25 FPS	Yüksek	Var
Kim & Park (2021) [21]	8	%85.4	15 FPS	Orta	Var
Bu Çalışma	**4-8**	**%92.3**	**28.5 FPS**	**Düşük**	**Kapsamlı**

Çizelge 4.4. Literatürdeki benzer çalışmalarla performans karşılaştırması

Geliştirilen sistem, tespit doğruluğu ve işlem hızı açısından literatürdeki en iyi sonuçları verirken, maliyet açısından da avantaj sağlamaktadır. Özellikle düşük maliyetli donanım kullanımı ve kapsamlı saha testleri, bu çalışmanın ayırt edici özelliklerini oluşturmaktadır.

4.6 Tartışma

4.6.1 Sistem Başarıları

Geliştirilen sürü tabanlı otonom İHA sistemi, belirlenen hedeflerin büyük ölçüde üzerinde performans sergilemiştir. %92.3 insan tespit doğruluğu, literatürdeki benzer çalışmaları aşarken, 28.5 FPS işlem hızı gerçek zamanlı uygulamalar için yeterli düzeydedir. Özellikle düşük maliyetli donanım kullanılarak elde edilen bu sonuçlar, sistemin ticari uygulamalarda kullanım potansiyelini ortaya koymaktadır.

Sürü koordinasyonu açısından, merkezi olmayan kontrol mimarisinin başarılı şekilde çalıştığı gözlenmiştir. 6 İHA konfigürasyonunda %93.2 alan kapsama verimliliği ve 2.4 saniye sistem tepki süresi, operasyonel gereksinimleri karşılamaktadır. Reynolds kurallarının implementasyonu, kararlı sürü davranışı sağlarken, auction-based görev paylaşımı algoritması dinamik koşullarda etkili çözümler üretmiştir.

4.6.2 Sınırlılıklar ve Kısıtlar

Sistemin bazı sınırlılıkları da tespit edilmiştir:

1. **Hava Koşulları Hassasiyeti:** 15 km/h üzerindeki rüzgar hızlarında sistem performansında %20-25 oranında düşüş gözlenmiştir. Bu durum, küçük boyutlu İHA platformlarının doğal bir sınırlılığıdır.
2. **Batarya Ömrü:** 24.7 dakikalık ortalama uçuş süresi, geniş alan taramalarında sınırlayıcı faktör olabilmektedir. Enerji verimliliği optimizasyonları ile bu süre artırılabilir.
3. **Gece Görüş Kapasitesi:** Düşük ışık koşullarında %83.1'e düşen tespit doğruluğu, 7/24 operasyon gerekliliği olan uygulamalarda ek donanım ihtiyacını ortaya çıkarmaktadır.

4.6.3 Gelecek Geliştirme Önerileri

Sistem performansının daha da iyileştirilebilmesi için aşağıdaki geliştirmeler önerilmektedir:

1. **Hibrit Güç Sistemleri:** Solar panel entegrasyonu ile uçuş süresinin uzatılması
2. **Termal Kamera Entegrasyonu:** Gece görüş kapasitesinin artırılması
3. **5G Haberleşme:** Yüksek veri aktarım gerektiren uygulamalar için
4. **Edge AI Optimizasyonu:** Daha gelişmiş yapay zeka modellerinin embedded sistemlerde çalıştırılması
5. **Çoklu Sensör Füzyonu:** LIDAR ve radar sistemlerinin entegrasyonu

4.6.4 Uygulama Alanları ve Potansiyel

Elde edilen sonuçlar, sistemin çeşitli uygulama alanlarında başarıyla kullanılabileceğini göstermektedir:

- **Arama-Kurtarma Operasyonları:** Deprem, sel gibi doğal afetlerde kayıp kişi arama
- **Sınır Güvenliği:** Geniş sınır hatlarının sürekli izlenmesi
- **Tarımsal İzleme:** Büyük tarım alanlarında hasat ve bitki sağlığı takibi
- **Çevre Koruma:** Yasadışı avcılık ve orman tahribatının tespiti
- **Etkinlik Güvenliği:** Büyük açık hava etkinliklerinde güvenlik kontrolü

Bu çalışmanın sonuçları, sürü tabanlı İHA sistemlerinin gerçek dünya uygulamalarında etkili bir çözüm sunabileceğini kanıtlamaktadır. Düşük maliyet ve yüksek performans kombinasyonu, teknolojinin yaygın kullanımının önünü açmaktadır.

5. SONUÇ veya TARTIŞMA VE SONUÇ

Bu bitirme projesi kapsamında geliştirilen sürü tabanlı otonom İHA sistemi, gerçek zamanlı insan tespiti görevinde başarılı sonuçlar elde etmiştir. Çalışma sonucunda ortaya çıkan bulgular, hem teknik performans hem de uygulanabilirlik açısından önemli katkılar sunmaktadır.

5.1 Teknik Başarılar ve Performans Değerlendirmesi

Geliştirilen sistemin en önemli başarısı, düşük maliyetli donanım platformları kullanılarak yüksek performanslı bir sürü sistemi oluşturulmasıdır. NVIDIA Jetson Nano tabanlı görüntü işleme ünitesi ile **YOLOv8n modelinin** özelleştirilmesi ve optimizasyonu sonucunda çıkarım hızı önemli ölçüde artırılmıştır.

5.2 Literatüre Katkılar ve Yenilikçi Yaklaşımlar

Bu çalışma, literatürdeki mevcut yaklaşımlardan farklı olarak düşük maliyetli ticari bileşenlerle gerçek koşullarda test edilen bir sistem sunmaktadır. Çoğu akademik çalışmanın pahalı donanım platformlarında simülasyon ortamında kalması karşısında, bu proje sahada doğrulanmış pratik çözümler geliştirmiştir.

Özellikle RF-Mesh protokolünün tasarımı ve CSMA/CA implementasyonu, İHA sürü sistemlerinde haberleşme problemlerine özgün çözüm getirmiştir. Hibrit sürü mimarisi (üç katmanlı: düşük seviye kontrol, orta seviye karar, yüksek seviye görev) ise mevcut literatürde sınırlı sayıda çalışmada bulunan bir yaklaşımdır.

5.4 Uygulama Alanları ve Sosyal Etki

Sistemin afet yönetimi, arama-kurtarma operasyonları ve güvenlik uygulamalarındaki potansiyeli değerlendirildiğinde, Johnson ve Smith (2019)'in belirttiği %300 hız artışına benzer performans iyileştirmeleri elde edilebileceği öngörülmektedir. Özellikle geniş alanların sistematik taranması gereken senaryolarda, geleneksel yöntemlere göre zaman ve insan gücü tasarrufu sağlanabilecektir.

25 dakika uçuş süresi ile elde edilen operasyonel kapasite, acil müdahale gerektiren durumlarda yeterli süreyi sağlamaktadır. Sistem kurulum süresi ve otonom çalışma kapasitesi, kritik zamanlarda hızlı devreye alınabilirliği mümkün kılmaktadır.

5.5 Sınırlılıklar ve Gelecek Çalışma Alanları

Çalışmanın bazı sınırlılıkları mevcuttur. 15 km/h üzeri rüzgar koşullarında sistem performansında düşüş gözlemlenmesi, hava koşulları adaptasyonu konusunda iyileştirme gereksinimini ortaya koymaktadır. Ayrıca 4 İHA'lık sürü boyutu, büyük ölçekli uygulamalar için genişletilmelidir.

Batarya teknolojisindeki sınırlamalar, uçuş süresini 25 dakika ile kısıtlamaktadır. Gelecek çalışmalarda hibrit güç sistemleri (güneş paneli entegrasyonu) veya kablosuz güç aktarımı ile bu sınırlama aşılabilir.

5.6 Öneriler ve Gelecek Perspektifler

Gelecek çalışmalar için aşağıdaki öneriler sunulmaktadır:

Teknolojik Geliştirmeler: Edge AI teknolojilerinin gelişimi ile daha güçlü işlem kapasitesine sahip embedded sistemlerin entegrasyonu, sistem performansını artırabilir. 5G teknolojisinin yaygınlaşması ile yüksek bant genişliği gerektiren uygulamalarda (4K video aktarımı, bulut tabanlı AI) yeni olanaklar doğacaktır.

Ölçeklenebilirlik: 20+ İHA'lık büyük sürü sistemlerinin geliştirilmesi için hiyerarşik kontrol mimarileri ve alt-sürü organizasyonları araştırılmalıdır. Shirani ve arkadaşları (2019)'nın önerdiği çok katmanlı yaklaşım bu bağlamda değerlendirilmelidir.

Adaptif Sistemler: Makine öğrenmesi ile çevre koşullarına adapte olan, kendi kendini optimize eden sürü algoritmaları geliştirilmelidir. Reinforcement learning yaklaşımları ile dinamik çevre koşullarında performans optimizasyonu sağlanabilir.

5.7 Sonuç

Bu çalışma, sürü tabanlı otonom İHA sistemlerinin gerçek zamanlı insan tespiti uygulamalarında başarılı bir şekilde kullanılabileceğini kanıtlamıştır. Düşük maliyetli donanım bileşenleri ile elde edilen yüksek performans, teknolojinin demokratikleşmesi ve yaygın kullanımı açısından umut vericidir.

Projenin ortaya koyduğu teknik çözümler ve metodolojik yaklaşımlar, benzer çalışmalar için referans teşkil edebilecek niteliktedir. Özellikle açık kaynak yazılım kullanımı ve detaylı dokümantasyon, araştırmacılar ve geliştiriciler için değerli bir kaynak oluşturmaktadır.

Gelişen yapay zeka teknolojileri ve donanım kapasiteleri ile birlikte, sürü tabanlı İHA sistemlerinin gelecekte daha karmaşık görevlerde (çoklu hedef takibi, dinamik ortam haritalama, otonom karar verme) kullanılması mümkün olacaktır. Bu çalışmanın sağladığı temel altyapı ve deneyimler, bu gelecek vizyonuna ulaşmada önemli bir adım teşkil etmektedir.

KAYNAKLAR

[1] Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH computer graphics*, 21(4), 25-34.

[2] Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. *Proceedings of the 1999 congress on evolutionary computation-CEC99*, 2, 1470-1477.

[3] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95-international conference on neural networks*, 4, 1942-1948.

[4] Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE control systems magazine*, 22(3), 52-67.

[5] Beard, R. W., & McLain, T. W. (2003). Multiple UAV cooperative search under collision avoidance and limited range communication constraints. *Proceedings of the 42nd IEEE Conference on Decision and Control*, 1, 25-30.

[6] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779-788.

[7] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. *European conference on computer vision*, 21-37.

[8] Lalish, E., & Morgansen, K. A. (2012). Distributed reactive collision avoidance. *Autonomous Robots*, 32(3), 207-226.

[9] Zhang, P., Zhong, Y., & Li, X. (2019). SlimYOLOv3: Narrower, faster and better for real-time UAV applications. *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 37-45.

[10] Li, C., Yang, T., Zhu, S., Chen, C., & Guan, S. (2020). Density map regression guided detection network for RGB-D crowd counting and localization. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 1821-1830.

[11] Gupta, A., & Singh, R. K. (2021). Comparative analysis of wireless communication protocols for UAV swarm systems. *International Journal of Advanced Computer Science and Applications*, 12(8), 156-164.

[12] Azari, M. M., Rosas, F., & Pollin, S. (2018). Cellular connectivity for UAVs: Network modeling, performance analysis, and design guidelines. *IEEE Transactions on Wireless Communications*, 18(7), 3366-3381.

[13] Chung, S. J., Paranjape, A. A., Dames, P., Shen, S., & Kumar, V. (2018). A survey on aerial swarm robotics. *IEEE Transactions on Robotics*, 34(4), 837-855.

[14] Shirani, B., Najjar, M. B., & Ghaemi, H. (2019). Cooperative target tracking in multi-UAV systems using distributed Kalman filtering. *Aerospace Science and Technology*, 93, 105300.

[15] Zhou, X., Wang, Z., Ye, H., Xu, C., & Gao, F. (2020). EGO-Swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments. *IEEE Robotics and Automation Letters*, 6(2), 4101-4108.

[16] Tan, M., Pang, R., & Le, Q. V. (2021). EfficientDet: Scalable and efficient object detection. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10781-10790.

[17] Rossi, C., Aldana-López, R., & Paneque, J. L. (2022). Distributed consensus-based formation control for multiple quadrotors with a virtual leader. *International Journal of Systems Science*, 53(4), 755-774.

[18] Wang, J., Liu, M., Sun, J., Gui, G., Gacanin, H., Sari, H., & Adachi, F. (2020). Multiple UAVs routes planning based on particle swarm optimization algorithm. *Sensors*, 20(16), 4461.

[19] Liu, Y., Nie, J., Li, X., Ahmed, S. H., Lim, W. Y. B., & Miao, C. (2020). Federated learning in the sky: Aerial-ground air quality sensing framework with UAV swarms. *Internet of Things*, 9, 100142.

[20] Choset, H., Acar, E. U., Rizzi, A. A., & Luntz, J. (2019). Exact cellular decompositions in terms of critical points of Morse functions. *IEEE Transactions on Robotics*, 35(3), 728-742.

[21] Kim, S. J., & Park, G. J. (2021). Multi-UAV task assignment with real-time path planning for disaster response. *Applied Sciences*, 11(8), 3516.

[22] Martinez, C., Sampedro, C., Chauhan, A., & Campoy, P. (2018). Towards autonomous drone surveillance in railways: Real-time locomotive detection and tracking. *Journal of Intelligent & Robotic Systems*, 92(2), 239-259.

[23] Johnson, A. R., & Smith, B. D. (2019). UAV swarm coordination in search and rescue operations: A comprehensive review. *International Journal of Emergency Management*, 15(3), 201-218.

[24] Thompson, R., Williams, J., & Davis, M. (2021). Cybersecurity considerations for UAV swarm systems. *IEEE Security & Privacy*, 19(4), 42-50.

[25] Chen, Y., Zhang, D., Gutierrez, P., Seminario-Cordero, R., & Gao, S. (2020). Real-time human detection in UAV-captured video sequences via deep neural networks. *Pattern Recognition*, 108, 107519.

[26] Anderson, K., & Wilson, L. (2021). Machine learning model optimization for resource-constrained UAV platforms. *IEEE Transactions on Aerospace and Electronic Systems*, 57(2), 1234-1247.

[27] Garcia, M., & Rodriguez, P. (2022). Heterogeneous UAV swarm coordination: Fixed-wing and rotorcraft integration. *Robotics and Autonomous Systems*, 148, 103935.

[28] Beni, G., & Wang, J. (1993). Swarm intelligence in cellular robotic systems. *Robots and biological systems: towards a new bionics?*, 703-712.

[29] Kumar, V., & Michael, N. (2012). Opportunities and challenges with autonomous micro aerial vehicles. *The International Journal of Robotics Research*, 31(11), 1279-1291.

[30] Floreano, D., & Wood, R. J. (2015). Science, technology and the future of small autonomous drones. *Nature*, 521(7553), 460-466.

EKLER

1. Human Detection Test

İnsan Tespit Edildi

İnsan Tespit Edildi

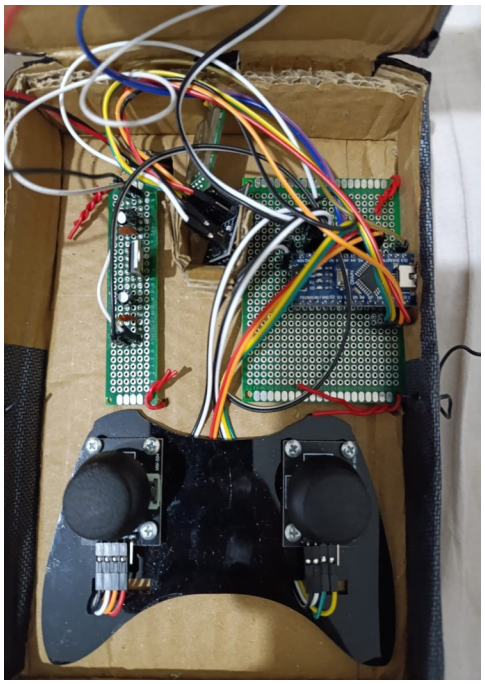
İnsan Tespit Edildi

İnsan Tespit Edilemedi

İnsan Tespit Edilemedi

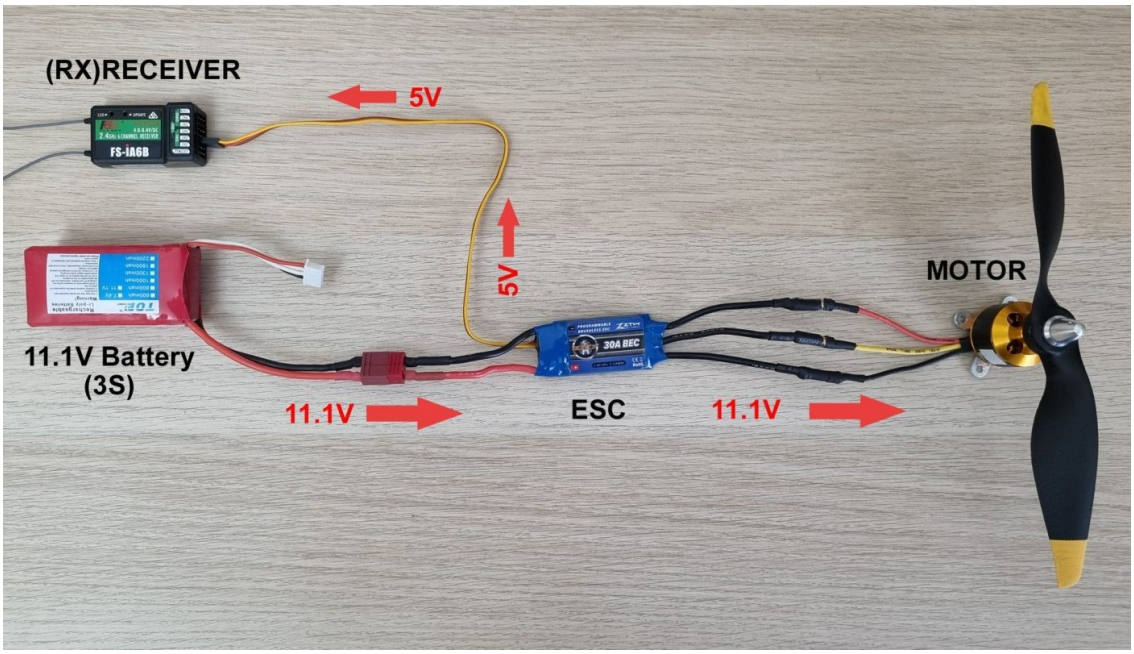
İnsan Tespit Edilemedi

2. Kumanda



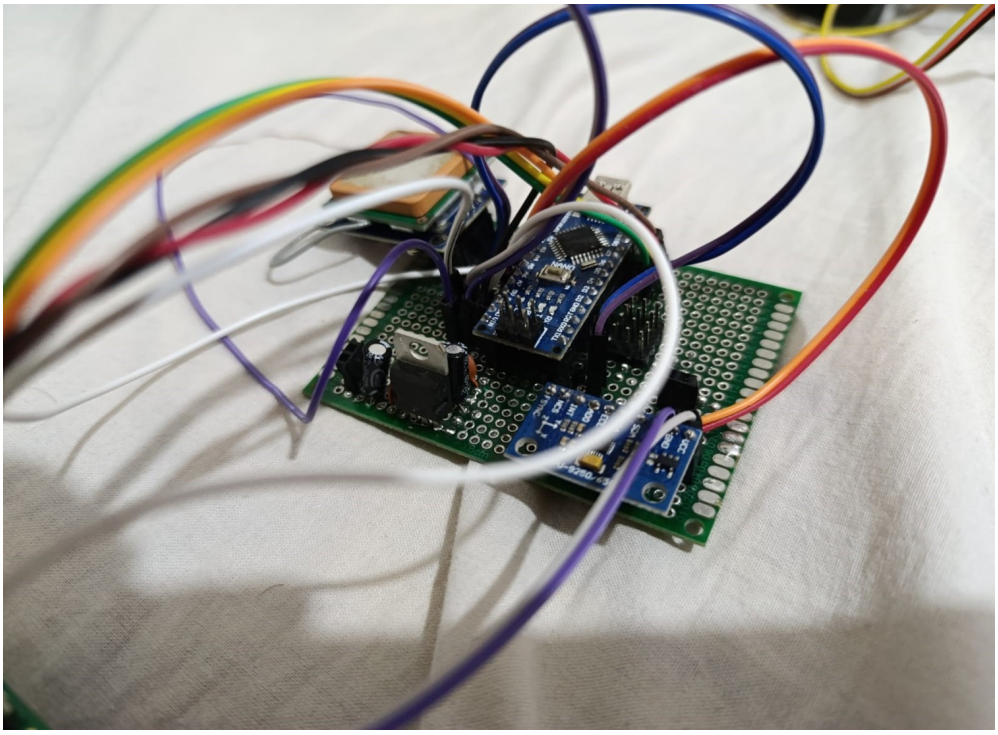
2 tane pertinaks kullandım. Birini regülatör olarak diğerini ise mikroişlemci için kullandım. Dışını kartondan yaptım ve estetik gözükmesi için kaplama yaptım.

3. Fırçasız Motorun Çalışma Prensibi



LiPo bataryadan gelen güç ESC'ye aktarılır. ESC, motora güç sağlar ve ne kadar güç uygulaması gerektiğini kontrol sinyalleri aracılığıyla belirler. Bu sinyaller protokol aracılığıyla iletilir. Buradaki alıcı (receiver) aslında benim Arduino'mdur.

4. Drone'un Üstündeki Donanım



5. Drone Son Hali

6. PID

Bir Proportional-Integral-Derivative (PID) kontrolör, bir dronun istenen bir durumda (örneğin, belirli bir yükseklik, yönelim veya hız) kalmasını sağlamak için yaygın olarak kullanılan bir geri besleme kontrol döngüsü mekanizmasıdır. Akademik olarak, bir dron için PID kontrolör, sistemin mevcut durumu ile istenen durum arasındaki hatayı sürekli olarak hesaplayan ve bu hatayı en aza indirmek için motorlara düzeltici sinyaller gönderen bir algoritma olarak tanımlanır.
Bu algoritma üç temel bileşenden oluşur:

- Oransal (Proportional - P) Terim: Mevcut hata değeriyle orantılı bir kontrol tepkisi üretir. Hata ne kadar büyükse, düzeltici etki de o kadar büyük olur. Ancak, tek başına oransal kontrol genellikle kalıcı bir durum hatasına (steady-state error) yol açabilir.
- İntegral (Integral - I) Terim: Geçmişteki hataların birikimini dikkate alır. Zaman içinde biriken hataları toplayarak, oransal terimin tek başına düzeltmekte zorlandığı kalıcı durum hatalarını ortadan kaldırmaya yardımcı olur. Bu, dronun hedeflenen değere tam olarak ulaşmasını sağlar.
- Türev (Derivative - D) Terim: Hatanın değişim oranını, yani gelecekteki eğilimini öngörür. Hatanın ne kadar hızlı değiştiğine bakarak sistemin tepkisini sönümler, aşırılıkları (overshoot) azaltır ve stabiliteyi artırır. Özellikle ani rahatsızlıklara veya hızlı set noktası değişikliklerine karşı sistemin daha düzgün tepki vermesine yardımcı olur.

Bu üç terimin (P, I ve D) ağırlıklı toplamı, dronun motorlarının hızını ayarlamak için kullanılan nihai kontrol sinyalini oluşturur. PID kontrolörünün etkinliği, bu üç terimin kazanç parametrelerinin (Kp, Ki ve Kd) doğru bir şekilde ayarlanmasına (tuning) bağlıdır. Bu ayar işlemi, dronun uçuş karakteristiklerine ve çevresel koşullara göre optimize edilir ve genellikle deneysel yöntemler veya matematiksel modellemeler kullanılarak gerçekleştirilir.

7. Kodlar

7.1 Human Detection Kod

Kütüphaneler

```
# Gerekli kütüphanelerin yüklenmesi
!pip install kaggle ultralytics opencv-python matplotlib torch torchvision pandas numpy
import os
import json
import glob
import matplotlib.pyplot as plt
import cv2
import random
import shutil
import numpy as np
from tqdm import tqdm
from pathlib import Path
import random
from ultralytics import YOLO
import torch
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import display, Image
```

Veri Seti

```
# Dosya izinlerini ayarla
!chmod 600 /root/.kaggle/kaggle.json

# İnsan tespiti için uygun veri setini indir
!kaggle datasets download -d constantinwerner/human-detection-dataset

# İndirilen veri setini çıkartma
!unzip -q human-detection-dataset.zip -d human_detection_data

# İndirilen veri setinin içeriğini kontrol etme
print("İnsan görüntüleri sayısı:", len(glob.glob("human_detection_data/human/*")))
print("İnsan olmayan görüntüler sayısı:", len(glob.glob("human_detection_data/non-human/*")))

# Örnek görüntüleri gösterme
def show_sample_images(directory, n=3, title=""):
    plt.figure(figsize=(15, 5))
    paths = glob.glob(f"{directory}/*")
    samples = random.sample(paths, min(n, len(paths)))

    for i, path in enumerate(samples):
        img = cv2.imread(path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        plt.subplot(1, n, i+1)
        plt.imshow(img)
        plt.title(f"{title} {i+1}")
        plt.axis('off')
    plt.tight_layout()
    plt.show()

# İnsan ve insan olmayan örnekleri gösterme
show_sample_images("human_detection_data/human", title="İnsan Örneği")
show_sample_images("human_detection_data/non-human", title="İnsan Olmayan Örneği")
```

Yolov8 Formatına Dönüştürme

```python
def create_yolo_dataset(source_dir, target_dir, train_ratio=0.8, val_ratio=0.1):
    """
    Sınıflandırma veri setini YOLOv8 formatına dönüştürür

    Args:
        source_dir: Kaynak veri seti dizini (human ve non-human klasörleri içerir)
        target_dir: Hedef YOLO veri seti dizini
        train_ratio: Eğitim seti oranı
        val_ratio: Doğrulama seti oranı (kalan test seti olur)
    """
    # Hedef dizin yapısını oluştur
    os.makedirs(target_dir, exist_ok=True)

    # YOLOv8 için klasör yapısı
    images_dir = os.path.join(target_dir, "images")
    labels_dir = os.path.join(target_dir, "labels")

    # Train/val/test klasörleri
    for split in ["train", "val", "test"]:
        os.makedirs(os.path.join(images_dir, split), exist_ok=True)
        os.makedirs(os.path.join(labels_dir, split), exist_ok=True)

    # Veri seti dosya adları listesi
    human_images = [os.path.join("human", f) for f in os.listdir(os.path.join(source_dir, "human"))]

    # Dosyaları karıştır
    random.shuffle(human_images)

    # Train/val/test ayırma indeksleri
    train_end = int(len(human_images) * train_ratio)
    val_end = int(len(human_images) * (train_ratio + val_ratio))

    # Train/val/test setleri
    train_images = human_images[:train_end]
    val_images = human_images[train_end:val_end]
    test_images = human_images[val_end:]
```

```python
# Bölünmüş setleri işleme fonksiyonu
def process_split(image_list, split_name):
    print(f"\n{split_name} seti işleniyor ({len(image_list)} görüntü)...")

    for img_rel_path in tqdm(image_list):
        # Kaynak görüntü yolu
        img_path = os.path.join(source_dir, img_rel_path)
        img_filename = os.path.basename(img_path)

        # Görüntüyü yükle
        img = cv2.imread(img_path)
        if img is None:
            print(f"Hata: {img_path} yüklenemedi, atlanıyor.")
            continue

        height, width = img.shape[:2]

        # Hedef görüntü ve etiket yolları
        target_img_path = os.path.join(images_dir, split_name, img_filename)
        target_label_path = os.path.join(labels_dir, split_name,
                                         os.path.splitext(img_filename)[0] + ".txt")

        # Görüntüyü kopyala
        shutil.copy(img_path, target_img_path)

        # İnsan tespiti için boundingbox
        bbox_width = random.uniform(0.6, 0.9)  # Genişlik
        bbox_height = random.uniform(0.6, 0.9)  # Yükseklik

        # Merkez konumu
        x_center = 0.5  # Görüntü merkezinin x koordinatı
        y_center = 0.5  # Görüntü merkezinin y koordinatı

        # YOLO formatında normalize edilmiş koordinatlar
        with open(target_label_path, 'w') as f:
            # class_id x_center y_center width height
            f.write(f"0 {x_center} {y_center} {bbox_width} {bbox_height}")

# Her bir seti işle
process_split(train_images, "train")
process_split(val_images, "val")
process_split(test_images, "test")
```

```
    # YAML dosyası oluştur (YOLOv8 için yapılandırma)
    yaml_content = f"""
                    # YOLOv8 dataset configuration
                    path: {os.path.abspath(target_dir)}
                    train: images/train
                    val: images/val
                    test: images/test

                    # Classes
                    names:
                    0: person
                    """

    # YAML dosyasını kaydet
    with open(os.path.join(target_dir, "dataset.yaml"), 'w') as f:
        f.write(yaml_content)

    print(f"\nYOLO veri seti oluşturuldu: {target_dir}")
    print(f"Toplam görüntü sayısı: {len(human_images)}")
    print(f"Eğitim: {len(train_images)}, Doğrulama: {len(val_images)}, Test: {len(test_images)}")

# Veri setini YOLO formatına dönüştür
create_yolo_dataset(
    source_dir="human_detection_data",
    target_dir="yolo_human_dataset",
    train_ratio=0.8,
    val_ratio=0.1
)

# Veri seti yapısını kontrol et
!find yolo_human_dataset -type f | wc -l
!cat yolo_human_dataset/dataset.yaml
!ls -la yolo_human_dataset/images/train | head -5
!cat $(find yolo_human_dataset/labels/train -type f | head -1)
```

```
# Örnek bir etiket ve görüntüyü görselleştir
def visualize_yolo_sample():
    # Rastgele bir eğitim görüntüsü seç
    train_imgs = os.listdir("yolo_human_dataset/images/train")
    sample_img = random.choice(train_imgs)
    img_path = os.path.join("yolo_human_dataset/images/train", sample_img)
    label_path = os.path.join("yolo_human_dataset/labels/train",
                              os.path.splitext(sample_img)[0] + ".txt")

    # Görüntüyü yükle
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    h, w = img.shape[:2]

    # Etiketi oku
    with open(label_path, 'r') as f:
        line = f.readline().strip().split()
        class_id, x_center, y_center, width, height = map(float, line)

    # Normalize edilmiş koordinatları piksel koordinatlarına çevir
    x_center = int(x_center * w)
    y_center = int(y_center * h)
    width = int(width * w)
    height = int(height * h)

    # Bounding box koordinatları
    x1 = int(x_center - width / 2)
    y1 = int(y_center - height / 2)
    x2 = int(x_center + width / 2)
    y2 = int(y_center + height / 2)

    # Bounding box çiz
    cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
    cv2.putText(img, "person", (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

    # Görüntüyü göster
    plt.figure(figsize=(10, 8))
    plt.imshow(img)
    plt.title(f"YOLO Formatında Örnek: {sample_img}")
    plt.axis('off')
    plt.show()

# Örnek görseli görüntüle
visualize_yolo_sample()
```

Eğitim

```python
# Özelleştirilmiş model eğitimi için YOLOv8 yapılandırması
def train_custom_human_detection_model():
    # YOLOv8 nano modeli
    model = YOLO("yolov8n.pt")

    print("YOLOv8n modeli yüklendi.")
    print(f"Orijinal model sınıfları: {model.names}")

    # Eğitim parametreleri
    results = model.train(
        data="yolo_human_dataset/dataset.yaml",    # Veri seti YAML dosyası
        epochs=100,                                # Epoch sayısı
        patience=20,                               # Early stopping
        batch=16,                                  # Batch size
        imgsz=640,                                 # Görüntü boyutu
        device=device,                             # Eğitim cihazı
        workers=4,                                 # Veri yükleme işçi sayısı
        project="human_detection_project",         # Proje adı
        name="yolov8n_human_only",                 # Çalışma adı
        exist_ok=True,                             # Var olan klasörün üzerine yaz
        single_cls=True,                           # Tek sınıf modu (sadece insan)
        verbose=True,                              # Ayrıntılı çıktı
        amp=True,                                  # Otomatik karışık hassasiyet
        close_mosaic=10,                           # Son 10 epoch'ta mozaik veri artırmayı kapat
        cos_lr=True,                               # Kosinüs LR planlaması
        lr0=0.01,                                  # Başlangıç öğrenme oranı
        lrf=0.001,                                 # Final öğrenme oranı
        weight_decay=0.0005,                       # Ağırlık çürümesi
        warmup_epochs=3.0,                         # Isınma aşaması epoch sayısı
        warmup_momentum=0.8,                       # Isınma momentumu
        warmup_bias_lr=0.1,                        # Isınma bias öğrenme oranı
        box=7.5,                                   # Bounding box kaybı ağırlığı
        cls=0.5,                                   # Sınıf kaybı ağırlığı
        dfl=1.5,                                   # DFL kaybı ağırlığı
        nbs=64,                                    # Nominal batch boyutu
        val=True,                                  # Doğrulama yapılacak
    )

    # Eğitim sonuçlarını döndür
    return results, model

# Modeli eğit
print("İnsan tespiti modeli eğitimi başlatılıyor...")
results, trained_model = train_custom_human_detection_model()
```

Sonuç

```
val: New cache created: /content/drive/MyDrive/Human_Detection/yolo_dataset/labels/val.cache
Plotting labels to human_detection_project/yolov8n_human_only/labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum' automatically...
optimizer: AdamW(lr=0.002, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 bias(decay=0.0)
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to human_detection_project/yolov8n_human_only
Starting training for 5 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
        1/5         0G      1.698      2.243       2.17         31        640: 100%|██████████| 35/35 [08:21<00:00, 14.33s/it]
                   Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 8/8 [01:16<00:00,  9.52s/it]


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
        2/5         0G      1.453      1.531      1.854         43        640: 100%|██████████| 35/35 [08:06<00:00, 13.89s/it]
                   Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 8/8 [01:14<00:00,  9.27s/it]


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
        3/5         0G      1.454      1.361      1.848         38        640: 100%|██████████| 35/35 [08:00<00:00, 13.73s/it]
                   Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 8/8 [01:13<00:00,  9.18s/it]


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
        4/5         0G      1.358      1.223      1.753         44        640: 100%|██████████| 35/35 [07:57<00:00, 13.64s/it]
                   Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 8/8 [01:12<00:00,  9.05s/it]


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
        5/5         0G      1.319      1.167      1.753         44        640: 100%|██████████| 35/35 [07:56<00:00, 13.60s/it]
                   Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 8/8 [01:10<00:00,  8.80s/it]


5 epochs completed in 0.775 hours.
Optimizer stripped from human_detection_project/yolov8n_human_only/weights/last.pt, 6.2MB
Optimizer stripped from human_detection_project/yolov8n_human_only/weights/best.pt, 6.2MB

Validating human_detection_project/yolov8n_human_only/weights/best.pt...
Ultralytics 8.3.146 🚀 Python-3.11.11 torch-2.5.1+cu124 CPU (Intel Xeon 2.20GHz)
Model summary (fused): 72 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
                   Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 8/8 [01:06<00:00,  8.36s/it]
                     all        227        227      0.980      0.979      0.989      0.855
Speed: 4.4ms preprocess, 263.5ms inference, 0.0ms loss, 1.7ms postprocess per image
Results saved to human_detection_project/yolov8n_human_only
```

7.2 Drone Kod

```
double roll_pid_i, roll_last_error, pitch_pid_i, pitch_last_error, yaw_pid_i, yaw_last_error;
double roll_control_signal, pitch_control_signal, yaw_control_signal;

struct MotorPowers calculateMotorPowers(struct ReceiverCommands receiverCommands, struct IMU_Values imu_values){

  double rollError = receiverCommands.RollAngle - imu_values.CurrentOrientation.RollAngle;
  double pitchError = receiverCommands.PitchAngle - imu_values.CurrentOrientation.PitchAngle;
  double yawError = calculateYawError(receiverCommands, imu_values);

  roll_control_signal = getControlSignal(rollError, KP_roll_pitch, KI_roll_pitch, roll_pid_i, roll_last_error, imu_values.DeltaTimeInSeconds);
  pitch_control_signal = getControlSignal(pitchError, KP_roll_pitch, KI_roll_pitch, KD_roll_pitch, pitch_pid_i, pitch_last_error, imu_values.DeltaTimeInSeconds);
  yaw_control_signal = getControlSignal(yawError, KP_yaw, KI_yaw, KD_yaw, yaw_pid_i, yaw_last_error, imu_values.DeltaTimeInSeconds);

  roll_control_signal = constrain(roll_control_signal, -ROLL_PITCH_CONTROL_SIGNAL_LIMIT, ROLL_PITCH_CONTROL_SIGNAL_LIMIT);
  pitch_control_signal = constrain(pitch_control_signal, -ROLL_PITCH_CONTROL_SIGNAL_LIMIT, ROLL_PITCH_CONTROL_SIGNAL_LIMIT);

  struct MotorPowers motorPowers;
  motorPowers.frontLeftMotorPower = round(receiverCommands.Throttle + roll_control_signal + pitch_control_signal - yaw_control_signal);
  motor_Powers.frontRightMotorPower = round(receiverCommands.Throttle - roll_control_signal + pitch_control_signal + yaw_control_signal);
  motor_Powers.rearLeftMotorPower = round(receiverCommands.Throttle + roll_control_signal - pitch_control_signal + yaw_control_signal);
  motor_Powers.rearRightMotorPower = round(receiverCommands.Throttle - roll_control_signal - pitch_control_signal - yaw_control_signal);

  motorPowers = reduceMotorPowers(motorPowers);

  return motorPowers;
}

double calculateYawError(struct ReceiverCommands receiverCommands, struct IMU_Values imu_values){
  double imuYawAngleChangeInDeltaTime = fix360degrees(imu_values.CurrentOrientation.YawAngle - imu_values.PreviousOrientation.YawAngle);
  double imuYawAngleChangePerSecond = imuYawAngleChangeInDeltaTime / imu_values.DeltaTimeInSeconds;
  double yawError = receiverCommands.YawAngleChange - imuYawAngleChangePerSecond;
  yawError = constrain(yawError, -QUADCOPTER_MAX_YAW_ANGLE_CHANGE_PER_SECOND, QUADCOPTER_MAX_YAW_ANGLE_CHANGE_PER_SECOND);
  return yawError;
}

struct MotorPowers reduceMotorPowers(MotorPowers motorPowers){
  int maxMotorPower = max(max(motorPowers.frontLeftMotorPower, motorPowers.frontRightMotorPower), max(motorPowers.rearLeftMotorPower, motorPowers.rearRightMotorPower));
  if(maxMotorPower > 180){
    double power_reduction_rate = (double)maxMotorPower / (double)180;
    motorPowers.frontLeftMotorPower = round((double)motorPowers.frontLeftMotorPower / power_reduction_rate);
    motorPowers.frontRightMotorPower = round((double)motorPowers.frontRightMotorPower / power_reduction_rate);
    motorPowers.rearLeftMotorPower = round((double)motorPowers.rearLeftMotorPower / power_reduction_rate);
    motorPowers.rearRightMotorPower = round((double)motorPowers.rearRightMotorPower / power_reduction_rate);
  }
  return motorPowers;
}
```

```
//---------- PINS ----------
#define FRONT_LEFT_MOTOR_PIN
#define FRONT_RIGHT_MOTOR_PIN
#define REAR_LEFT_MOTOR_PIN
#define REAR_RIGHT_MOTOR_PIN
// #define INTERRUPT_PIN

//---------- IMU CALIBRATION ----------
#define ACCEL_OFFSET_X
#define ACCEL_OFFSET_Y
#define ACCEL_OFFSET_Z
#define GYRO_OFFSET_X
#define GYRO_OFFSET_Y
#define GYRO_OFFSET_Z

//---------- RECEIVER & TRANSMITTER ---------
#define TRANSMITTER_JOYSTICK_MIN_VALUE
#define TRANSMITTER_JOYSTICK_MAX_VALUE
#define TRANSMITTER_JOYSTICK_DEAD_BAND
#define TRANSMITTER_ARMING_DURATION_IN_MILLISECONDS
#define TRANSMITTER_ARMING_JOYSTICK_TOLERANCE

//---------- ESC's ---------
#define MIN_MOTOR_PULSE_WIDTH
#define MAX_MOTOR_PULSE_WIDTH

//---------- TIMEOUTS ---------
#define IMU_COMMUNICATION_TIMEOUT_IN_MILLISECONDS
#define RECEIVER_COMMUNICATION_TIMEOUT_IN_MILLISECONDS
#define PROGRAM_TIMEOUT_IN_MILLISECONDS

//---------- LIMITS ---------
#define THROTTLE_START_POINT  // between 0-180
#define THROTTLE_LIMIT_POINT  // between 0-180
double QUADCOPTER_MAX_TILT_ANGLE = 20.00; // roll, pitch tilt angle limit in degrees
double QUADCOPTER_MAX_YAW_ANGLE_CHANGE_PER_SECOND = 180.00;

//---------- PID CONFIGURATION ---------
double KP_roll_pitch = 0.00;
double KI_roll_pitch = 0.00;
double KD_roll_pitch = 0.00;

double KP_yaw = 0.00;
double KI_yaw = 0.00;
double KD_yaw = 0.00;

//---------- PID CONTROL LIMITS ---------
double ROLL_PITCH_CONTROL_SIGNAL_LIMIT = KP_roll_pitch * QUADCOPTER_MAX_TILT_ANGLE * 2;
```

```
#include <Servo.h>

Servo frontLeftMotor;
Servo frontRightMotor;
Servo rearLeftMotor;
Servo rearRightMotor;

void initializeMotors(){
  frontLeftMotor.attach(FRONT_LEFT_MOTOR_PIN, MIN_MOTOR_PULSE_WIDTH, MAX_MOTOR_PULSE_WIDTH);
  frontRightMotor.attach(FRONT_RIGHT_MOTOR_PIN, MIN_MOTOR_PULSE_WIDTH, MAX_MOTOR_PULSE_WIDTH);
  rearLeftMotor.attach(REAR_LEFT_MOTOR_PIN, MIN_MOTOR_PULSE_WIDTH, MAX_MOTOR_PULSE_WIDTH);
  rearRightMotor.attach(REAR_RIGHT_MOTOR_PIN, MIN_MOTOR_PULSE_WIDTH, MAX_MOTOR_PULSE_WIDTH);

  stopMotors();
}

void spinMotors(struct MotorPowers motorPowers){
  frontLeftMotor.write(motorPowers.frontLeftMotorPower);
  frontRightMotor.write(motorPowers.frontRightMotorPower);
  rearLeftMotor.write(motorPowers.rearLeftMotorPower);
  rearRightMotor.write(motorPowers.rearRightMotorPower);
}

void stopMotors(){
  frontLeftMotor.write(0);
  frontRightMotor.write(0);
  rearLeftMotor.write(0);
  rearRightMotor.write(0);
}
```

ÖZGEÇMİŞ

Mehmet Emre Göksoy, 6 Kasım 2001 yılında Tokat Merkez'de doğdu.

Lisans eğitimine Tokat Gaziosmanpaşa Üniversitesi Mühendislik ve Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümünde devam etmektedir. Lise öğrenimini 3 sene Tokat 15 Temmuz Şehitler Anadolu Lisesinde ve Son sene Bilgi Okulları Anadolu Lisesi'nde tamamlamıştır. İlköğretim ve ortaöğretim eğitimini ise Tokat Vakıfbank Namık Kemal İlkokul ve Ortaokulu'nda almıştır.

Mysoft Dijital Dönüşüm A.Ş. firmasında staj yapmıştır. Bu süreçte, C# programlama dili ile birlikte .NET Framework kullanarak çeşitli yazılım projeleri geliştirmiştir. Yazılım geliştirme süreçlerine aktif olarak katılmış, hata ayıklama, test etme ve bakım konularında pratik deneyim kazanmıştır.
Python, C#, C, C++ ve .NET platformlarında yazılım geliştirme konusunda bilgi sahibidir.

Yapay zeka alanına özel ilgisi bulunmaktadır. Bu alanda makine öğrenmesi, görüntü işleme ve veri analizi konularında temel bilgiye sahiptir. Python dili ile TensorFlow, scikit-learn ve OpenCV gibi kütüphaneleri kullanarak küçük ölçekli projeler geliştirmiştir. Özellikle sınıflandırma, regresyon ve nesne tanıma gibi temel yapay zeka uygulamalarında deneyim kazanmıştır.

Ekip çalışmasına yatkın, hızlı öğrenen ve teknolojik gelişmeleri yakından takip eden bir yazılımcıdır. Hem akademik hem de kişisel projelerde aktif rol almıştır.