

# Görüntü İşleme Tabanlı Hedef İmha Aracı

*Hasan Gökhan MERCAN<sup>1</sup>*

Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği Bölümü

190542003@firat.edu.tr

## Özet

Bu çalışma, görüntü işleme tekniklerine dayalı, kırmızı bir topu takip eden otonom bir araç sisteminin geliştirilmesini ele almaktadır. Sistem, Evrişimsel Sinir Ağları (Convolutional Neural Networks, CNN) ile gerçek zamanlı görüntü analizi ve sınıflandırma yetkinliklerini hedeflemiş, ancak donanım kısıtlamaları nedeniyle Hue-Saturation-Value (HSV) renk uzayına dayalı algoritmalar kullanılmıştır. Görüntü toplama için ESP32-CAM modülü, görüntü işleme için Orange Pi 3 LTS ve motor kontrolü için Arduino Mega ile L298N sürücüsü kullanılmıştır. Araç, dört adet 12V DC motorla tahrik edilmiş; 6.000 mAh ve 12.000 mAh bataryalarla desteklenmiştir. Test sonuçları, sistemin kontrollü ortamlarda kırmızı topu başarıyla takip ettiğini, ancak dinamik aydınlatma ve hızlı hareket senaryolarında performansın düştüğünü göstermiştir. Gelecekte, daha güçlü donanımlar ve optimize edilmiş CNN modelleriyle performansın artırılması hedeflenmektedir. Bu çalışma, düşük maliyetli gömülü sistemlerde görüntü işleme uygulamalarına katkı sağlamaktadır.

## 1. Giriş

Otonom araç teknolojileri, yapay zeka, görüntü işleme ve robotik sistemlerin entegrasyonu ile modern mühendislik ve bilgisayar bilimlerinin yenilikçi alanlarından biridir. Görüntü işleme, otonom araçların nesne takibi ve navigasyon gibi kritik görevlerinde merkezi bir rol oynar. Bu bağlamda, Evrişimsel Sinir Ağları (CNN), görüntü verilerinden anlamlı özellikler çıkararak gerçek zamanlı sınıflandırma görevlerinde yüksek doğruluk sağlar. CNN'ler, evrişim, havuzlama ve tam bağlantılı katmanlarıyla, görüntülerdeki uzaysal ilişkileri koruyarak karmaşık desenleri algılar [1]. Evrişim katmanları yerel özellikleri (örneğin, renk ve şekil) tespit ederken, havuzlama katmanları hesaplama verimliliğini artırır; tam bağlantılı katmanlar ise özellikleri sınıflandırmada kullanır [2].

CNN'lerin başarısı, büyük veri setleri ve yüksek hesaplama gücüne dayanır. TensorFlow ve PyTorch gibi çerçeveler, GPU'ların gelişimiyle karmaşık CNN modellerinin tasarlanmasını sağlamıştır [3]. Ancak, bu algoritmaların yüksek hesaplama gereksinimleri, düşük maliyetli gömülü sistemlerde (örneğin, Orange Pi veya Arduino) uygulanabilirliklerini sınırlar. Literatürde, otonom araçlar genellikle yüksek performanslı işlem birimleri (GPU, TPU) ile geliştirilir; ancak bu çözümler maliyet ve enerji tüketimi açısından pratik değildir [4]. Buna karşılık, düşük maliyetli gömülü sistemler ekonomik bir alternatif sunar, ancak CNN tabanlı algoritmaları gerçek zamanlı desteklemede yetersiz kalır.

Bu çalışmada, kırmızı bir topu takip eden, görüntü işleme tabanlı otonom bir araç sistemi geliştirilmiştir. Sistem, ekranı üçe bölerek (sağ, sol, orta) topun konumuna göre hareket eder ve top yakın olduğunda durur. CNN tabanlı bir model TensorFlow ile eğitime çalışılmış, ancak Orange Pi 3 LTS'nin sınırlı işlem kapasitesi nedeniyle gerçek zamanlı çalıştırılamamış; bunun yerine HSV tabanlı renk sınıflandırma algoritmaları kullanılmıştır. Çalışma, donanım kısıtlamalarının CNN tabanlı sistemler üzerindeki etkilerini analiz eder ve düşük maliyetli platformlarda otonom araç uygulamaları için pratik bir örnek sunar.

## 2. Materyal ve Metot

### 2.1. Sistem Tasarımı

Bu çalışmada, kırmızı bir topu takip eden otonom bir araç sistemi geliştirilmiştir. Sistem, görüntü toplama, işleme, motor kontrolü ve güç yönetimi için donanım ve yazılım bileşenlerini entegre eder. Görüntü ekranı yatay olarak üçe bölünerek (sağ, sol, orta) topun konumu tespit edilmiş; topun yakınlığı, kontur alanına göre belirlenerek araç hareketleri (sağa dön, sola dön, ileri git, dur) kontrol edilmiştir. Arduino Mega, Orange Pi 3 LTS ile seri iletişim yoluyla motor kontrol komutlarını alır. Şekil 2.1, sistemin donanım mimarisini gösterir.

#### 2.1.1. ESP32-CAM Modülü

*Tanım:* ESP32-CAM, Espressif Systems tarafından geliştirilen, OV2640 kamera sensörlü, Wi-Fi ve Bluetooth destekli düşük maliyetli bir mikrodenetleyici modüldür.

*Amaç:* Gerçek zamanlı görüntü toplama için kullanılır. 640x480 piksel RGB görüntüler yakalayarak Orange Pi 3 LTS'ye HTTP protokolü üzerinden akış sağlar. Kompakt yapısı, gömülü sistemlerde görüntü işleme için uygundur.



Şekil 2.2: ESP32-CAM modülü.

### 2.1.2. Orange Pi 3 LTS

*Tanım:* Orange Pi 3 LTS, Allwinner H6 dört çekirdekli Cortex-A53 işlemciye sahip, 2 GB RAM'li, Linux tabanlı bir tek kart bilgisayardır.

*Amaç:* Ana işlem birimi olarak, ESP32-CAM'den gelen görüntü akışını işler, HSV tabanlı algoritmayı çalıştırır ve kırmızı topun konumunu (sağ, sol, orta) tespit eder. Seri iletişim yoluyla Arduino Mega'ya motor kontrol komutları gönderir. Sınırlı işlem kapasitesi nedeniyle CNN modeli gerçek zamanlı çalıştırılmamış, HSV algoritması tercih edilmiştir.



Şekil 2.3: Orange Pi 3 LTS'nin sistemdeki bağlantı düzeni ve işlem birimi yerleşimi.

### 2.1.3. Arduino Mega 2560

*Tanım:* Arduino Mega 2560, ATmega2560 tabanlı, çok sayıda giriş/çıkış pini sunan bir mikrodenetleyici kartıdır.

*Amaç:* Motor kontrol birimi olarak, Orange Pi 3 LTS'den seri iletişimle gelen komutları (örneğin, '0' için dur, '1' için ileri, '3' için sağa dön, '4' için sola dön) alır ve L298N sürücüsüne sinyaller göndererek DC motorları kontrol eder.



Şekil 2.4: Arduino Mega 2560.

#### 2.1.4. L298N Çift Kanallı DC Motor Sürücüsü

*Tanım:* L298N, iki DC motoru veya bir step motoru kontrol eden, H-köprüsü tabanlı bir motor sürücü modülüdür.

*Amaç:* Dört adet 12V DC motorun hızını ve yönünü kontrol eder. Arduino Mega'dan gelen PWM sinyalleriyle aracın hareketini (sağa, sola, ileri) sağlar.



Şekil 2.5: L298N motor sürücüsü.

#### 2.1.5. 12V DC Motorlar (4 Adet)

*Tanım:* 12V DC motorlar, elektrik enerjisini mekanik harekete dönüştüren yüksek torklu cihazlardır.

*Amaç:* Aracın tahrik sistemini oluşturur. L298N sürücüsüyle kontrol edilerek diferansiyel tahrikle yön kontrolü sağlar.

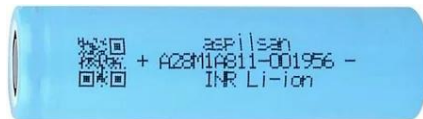


Şekil 2.6: 12V DC motor.

#### 2.1.6. 6.000 mAh Batarya (İşlem Birimi İçin)

*Tanım:* 6.000 mAh kapasiteli lityum-iyon batarya, taşınabilir cihazlar için güç kaynağıdır.

*Amaç:* Orange Pi 3 LTS ve ESP32-CAM'in güç ihtiyacını karşılar. Gerilim regülatörü ile 5.2 V çıkış sağlar.



Şekil 2.7: 6.000 mAh bataryanın bağlantı şeması ve işlem birimi güç entegrasyonu.

### 2.1.7. 12.000 mAh Batarya (Motorlar İçin)

*Tanım:* 12.000 mAh kapasiteli lityum-iyon batarya, yüksek akım gerektiren uygulamalar için tasarlanmıştır.

*Amaç:* DC motorlar ve L298N sürücüsünün güç ihtiyacını karşılar. Gerilim regülatörü ile güç dalgalanmaları önlenir.



Şekil 2.8: 12.000 mAh batarya.

### 2.1.8. Gerilim Regülatörü

*Tanım:* Gerilim regülatörü (LM2596 DC-DC buck converter), giriş gerilimini sabit bir çıkışa dönüştürür.

*Amaç:* Bataryalardan gelen 7.4-8 V gerilimi 5.2 V'a düşürerek Orange Pi 3 LTS, ESP32-CAM ve motorlar için stabil güç sağlar.



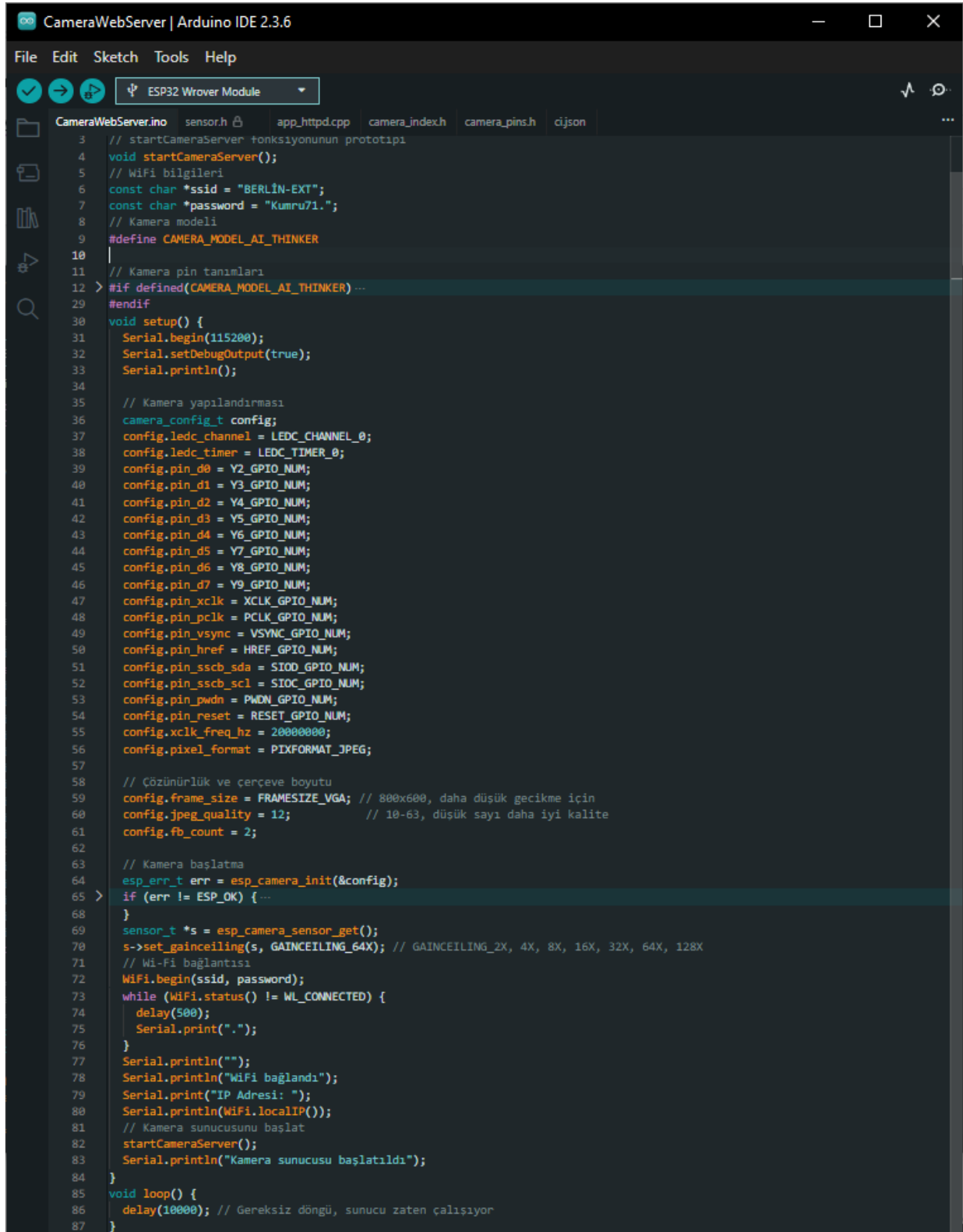
Şekil 2.9: Gerilim regülatörü.

## 2.2. Yazılım Mimarisi ve Görüntü İşleme Süreci

Sistemin yazılım mimarisi, ESP32-CAM, Orange Pi 3 LTS ve Arduino Mega arasında koordineli bir iş akışı sağlar. Aşağıda, her bir bileşenin yazılım işlevleri açıklanmıştır:

### 2.2.1. ESP32-CAM Görüntü Yayını

ESP32-CAM, OV2640 kamera sensörüyle 640x480 piksel çözünürlüğünde JPEG formatında görüntü yakalar ve Wi-Fi üzerinden HTTP protokolüyle Orange Pi 3 LTS'ye akış sağlar. `setup()` fonksiyonu, kamera yapılandırmasını (çözünürlük, JPEG kalitesi, kare hızı) başlatır ve Wi-Fi bağlantısını kurar. `startCameraServer()` fonksiyonu, bir web sunucusu oluşturarak görüntü akışını IP adresi üzerinden yayınlar. Bu, Orange Pi 3 LTS'nin gerçek zamanlı görüntü verilerine erişmesini sağlar. Görüntü kalitesini optimize etmek için `set_gainceiling()` gibi yöntemler kullanılarak sensör ayarları yapılmıştır.



```
CameraWebServer.ino  sensor.h  app_httpd.cpp  camera_index.h  camera_pins.h  cJSON

3 // startCameraServer fonksiyonunun prototipi
4 void startCameraServer();
5 // Wifi bilgileri
6 const char *ssid = "BERLIN-EXT";
7 const char *password = "Kumru71.";
8 // Kamera modeli
9 #define CAMERA_MODEL_AI_THINKER
10
11 // Kamera pin tanımları
12 > #if defined(CAMERA_MODEL_AI_THINKER) ...
13 #endif
29
30 void setup() {
31     Serial.begin(115200);
32     Serial.setDebugOutput(true);
33     Serial.println();
34
35     // Kamera yapılandırması
36     camera_config_t config;
37     config.ledc_channel = LEDC_CHANNEL_0;
38     config.ledc_timer = LEDC_TIMER_0;
39     config.pin_d0 = Y2_GPIO_NUM;
40     config.pin_d1 = Y3_GPIO_NUM;
41     config.pin_d2 = Y4_GPIO_NUM;
42     config.pin_d3 = Y5_GPIO_NUM;
43     config.pin_d4 = Y6_GPIO_NUM;
44     config.pin_d5 = Y7_GPIO_NUM;
45     config.pin_d6 = Y8_GPIO_NUM;
46     config.pin_d7 = Y9_GPIO_NUM;
47     config.pin_xclk = XCLK_GPIO_NUM;
48     config.pin_pclk = PCLK_GPIO_NUM;
49     config.pin_vsync = VSYNC_GPIO_NUM;
50     config.pin_href = HREF_GPIO_NUM;
51     config.pin_sscb_sda = SIOD_GPIO_NUM;
52     config.pin_sscb_scl = SIOC_GPIO_NUM;
53     config.pin_pwdn = PWDN_GPIO_NUM;
54     config.pin_reset = RESET_GPIO_NUM;
55     config.xclk_freq_hz = 20000000;
56     config.pixel_format = PIXFORMAT_JPEG;
57
58     // Çözünürlük ve çerçeve boyutu
59     config.frame_size = FRAMESIZE_VGA; // 800x600, daha düşük gecikme için
60     config.jpeg_quality = 12; // 10-63, düşük sayı daha iyi kalite
61     config.fb_count = 2;
62
63     // Kamera başlatma
64     esp_err_t err = esp_camera_init(&config);
65 > if (err != ESP_OK) { ...
66 }
67
68 sensor_t *s = esp_camera_sensor_get();
69 s->set_gainceiling(s, GAINCEILING_64X); // GAINCEILING_2X, 4X, 8X, 16X, 32X, 64X, 128X
70 // Wi-Fi bağlantısı
71 WiFi.begin(ssid, password);
72 while (WiFi.status() != WL_CONNECTED) {
73     delay(500);
74     Serial.print(".");
75 }
76 Serial.println("");
77 Serial.println("Wifi bağlandı");
78 Serial.print("IP Adresi: ");
79 Serial.println(WiFi.localIP());
80 // Kamera sunucusunu başlat
81 startCameraServer();
82 Serial.println("Kamera sunucusu başlatıldı");
83
84 }
85 void loop() {
86     delay(10000); // Gereksiz döngü, sunucu zaten çalışıyor
87 }
```

Şekil 2.10: ESP32-CAM'in yayın kodu.

### 2.2.2. Orange Pi 3 LTS Görüntü İşleme

Orange Pi 3 LTS, Python tabanlı bir yazılım ile ESP32-CAM'den gelen görüntü akışını işler. OpenCV kütüphanesi kullanılarak geliştirilen kamera\_calistir() fonksiyonu, görüntüleri RGB'den HSV renk uzayına dönüştürür ve kırmızı topu tespit eder.

```
1  import cv2
2  import numpy as np
3  import serial
4  import threading
5  import time
6  import sys
7  import os
8
9  # Seri portu başlat
10 try:
11     port = serial.Serial('/dev/ttyUSB0', 9600, timeout=10) # Orange Pi için seri port
12     time.sleep(5) # Arduino'nun bağlanması için bekle
13 except serial.SerialException as e:
14     print(f"Seri port hatası: {e}")
15     sys.exit(1)
16
17 # IP kamerasını başlat
18 ip_adresi = 'http://192.168.1.19:81/stream' # HTTP akış URL'si
19 cap = cv2.VideoCapture(ip_adresi) # IP kamerası için URL
20 if not cap.isOpened():
21     print(f"IP kamera açılmadı: {ip_adresi}. Lütfen IP adresini, portu veya protokolü kontrol edin.")
22     sys.exit(1)
23
24 # Kırmızı renk için HSV aralığı
25 kirmizi_alt = np.array([0, 150, 100]) # Kırmızı alt sınır
26 kirmizi_ust = np.array([10, 255, 255]) # Kırmızı üst sınır
27 kirmizi_alt2 = np.array([170, 150, 100]) # Kırmızı için ikinci aralık
28 kirmizi_ust2 = np.array([180, 255, 255])
29
30 # Alan eşikleri
31 MIN_ALAN = 500 # Nesnenin algılanması için minimum alan
32 BUYUK_ALAN = 5000 # Nesnenin "yeteri kadar büyük" sayılması için eşik
33
34 # İş parçacıklarının çalışmasını kontrol etmek için bir flag
35 dur = threading.Event()
36
37 # Kamera görüntüsünü işleyen ve kırmızı nesne algılayan fonksiyon
```

Şekil 2.11.1: Orange Pi 3 LTS'nin görüntü işleme kodu.

Görüntü yatay olarak üçe bölünerek (sağ, sol, orta) topun konumu belirlenir. cv2.inRange() yöntemiyle kırmızı renk için iki HSV eşik aralığı tanımlanır, cv2.erode() ve cv2.dilate() ile gürültü azaltılır, cv2.findContours() ile en büyük kontur seçilerek topun merkezi hesaplanır.

```
37 # Kamera görüntüsünü işleyen ve kırmızı nesne algılayan fonksiyon
38 def kamera_calistir():
39     son_b_komut_zamani = time.time() # Son 'b' komutunun gönderilme zamanını tut
40     b_komut_araligi = 3 # 3 saniyelik aralık
41
42     while not dur.is_set():
43         ret, frame = cap.read()
44         if not ret:
45             print("IP kameradan görüntü alınamadı! Bağlantıyı veya URL'yi kontrol edin.")
46             break
47
48         # Görüntü boyutlarını al
49         height, width = frame.shape[:2]
50         # Görüntüyü dikey olarak üçe böl
51         bolum_genislik = width // 3
52         sol_bolum = (0, bolum_genislik)
53         sag_bolum = (2 * bolum_genislik, width)
54
55         # Görüntüyü HSV renk uzayına çevir
56         hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
57
58         # Kırmızı renk için maske oluştur
59         mask1 = cv2.inRange(hsv, kirmizi_alt, kirmizi_ust)
60         mask2 = cv2.inRange(hsv, kirmizi_alt2, kirmizi_ust2)
61         mask = mask1 + mask2
62
63         # Gürültüyü azaltmak için morfolojik işlemler
64         kernel = np.ones((5, 5), np.uint8)
65         mask = cv2.erode(mask, kernel, iterations=2)
66         mask = cv2.dilate(mask, kernel, iterations=2)
67
68         # Durum metni için başlangıç değeri
69         durum_metni = "Nesne tespit edilmedi"
70
71         # Konturları bul
72         contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
73
74         # Kırmızı nesneyi çerçevele ve konumuna göre komut gönder
75         if contours:
76             max_contour = max(contours, key=cv2.contourArea)
77             alan = cv2.contourArea(max_contour)
78             if alan > MIN_ALAN: # Minimum alan sınırı
79                 x, y, w, h = cv2.boundingRect(max_contour)
80                 merkez_x = x + w // 2
81
82                 # Nesnenin hangi bölgede olduğunu kontrol et
83                 if sol_bolum[0] <= merkez_x < sol_bolum[1]:
84                     durum_metni = "Sol - Komut: d (4)"
85                     print("Kırmızı nesne solda algılandı, 'd' komutu gönderildi")
86                     port.write(b'4') # Solda: d komutu
87                 elif sag_bolum[0] <= merkez_x < sag_bolum[1]:
88                     durum_metni = "Sağ - Komut: c (3)"
89                     print("Kırmızı nesne sağda algılandı, 'c' komutu gönderildi")
90                     port.write(b'3') # Sağda: c komutu
91             else:
92                 durum_metni = "Nesne tespit edilmedi"
93             else:
94                 durum_metni = "Nesne tespit edilmedi"
95
96         # Durum metnini konsola yazdır
97         print(f"Durum: {durum_metni}")
98
99         # Döngüde çok hızlı çalışmayı önlemek için küçük bir gecikme
100         time.sleep(0.01) # 10 ms gecikme
```

Şekil 2.11.2: Orange Pi 3 LTS'nin görüntü işleme kodu.



Topun konumu ve kontur alanı (minimum 500 piksel, büyük eşik 5000 piksel) temel alınarak hareket komutları (0 için dur, 1 için ileri, 3 için sağa dön, 4 için sola dön) belirlenir. Bu komutlar, /dev/ttyUSB0 portu üzerinden 9600 baud hızıyla seri iletişimle Arduino Mega'ya gönderilir.

```
91         else:
92             # Orta bölgede: Nesne boyutuna göre komut seç
93             if alan > BUYUK_ALAN:
94                 durum_metni = "Orta (Büyük) - Komut: b (0)"
95                 print("Kırmızı nesne ortada ve yeteri kadar büyük, 'b' komutu gönderildi")
96                 port.write(b'0') # Orta ve büyük: b komutu
97                 son_b_komut_zamani = time.time() # Zamanlayıcıyı sıfırla
98             else:
99                 durum_metni = "Orta (Küçük) - Komut: a (1)"
100                 print("Kırmızı nesne ortada ama yeteri kadar büyük değil, 'a' komutu gönderildi")
101                 port.write(b'1') # Orta ve küçük: a komutu
102         else:
103             # Küçük konturlar için 'b' komutu gönder
104             if time.time() - son_b_komut_zamani >= b_komut_araligi:
105                 durum_metni = "Küçük kontur - Komut: b (0)"
106                 print("Küçük kontur algılandı, 'b' komutu gönderildi")
107                 port.write(b'0')
108                 son_b_komut_zamani = time.time()
109         else:
110             # Kontur yoksa her 3 saniyede bir 'b' komutu gönder
111             if time.time() - son_b_komut_zamani >= b_komut_araligi:
112                 durum_metni = "Nesne yok - Komut: b (0)"
113                 print("Kırmızı nesne tespit edilmedi, 'b' komutu gönderildi")
114                 port.write(b'0')
115                 son_b_komut_zamani = time.time()
116
117         # Durum metnini konsola yazdır
118         print(f"Durum: {durum_metni}")
119
120         # Döngüde çok hızlı çalışmayı önlemek için küçük bir gecikme
121         time.sleep(0.01) # 10 ms gecikme
122
123         # Kamera ve kaynakları temizle
124         cap.release()
125
126     # Ana program
127     try:
128         # Kamera iş parçasığını başlat
129         kamera_thread = threading.Thread(target=kamera_calistir)
130         kamera_thread.start()
131
132         # İş parçasığının tamamlanmasını bekle
133         kamera_thread.join()
134
135     except KeyboardInterrupt:
136         print("Klavye kesmesi alındı, program sonlandırılıyor...")
137         dur.set()
138
139     # Temizlik
140     try:
141         port.write(b'0') # Son bir komut gönder
142         port.close()
143     except:
144         pass
145
146     # Programı tamamen sonlandır
147     print("Program sonlanıyor...")
148     os._exit(0) # Ctrl+C gibi anında sonlandır
```

Şekil 2.11.3: Orange Pi 3 LTS'nin görüntü işleme kodu.

### 2.2.3. Arduino Mega Motorların Kontrolü

Arduino Mega, Orange Pi 3 LTS'den seri iletişimle gelen komutları alır ve L298N motor sürücüsüne PWM sinyalleri göndererek dört adet 12V DC motoru kontrol eder. setup() fonksiyonu, seri iletişimi başlatır ve motor pinlerini yapılandırır. Gelen komutlara göre (0 için dur, 1 için ileri, 3 için sağa dön, 4 için sola dön), motorların yönü ve hızı ayarlanır. Diferansiyel tahrik sistemiyle, araç kırmızı topa yaklaşmak için uygun hareketleri gerçekleştirir.

```
arduinoMotorCalistir11.05.ino
7
8 void setup() {
9   // Pinleri çıkış olarak ayarla
10  pinMode(ENA, OUTPUT);
11  pinMode(IN1, OUTPUT);
12  pinMode(IN2, OUTPUT);
13  pinMode(IN3, OUTPUT);
14  pinMode(IN4, OUTPUT);
15
16  // Seri haberleşmeyi başlat (9600 baud)
17  Serial.begin(9600);
18  Serial.println("1: İleri, 2: Geri, 3: Sağ, 4: Sol");
19
20  // Başlangıçta motorları durdur
21  motorStop();
22 }
23
24 void loop() {
25   // Seri porttan veri gelirse oku
26   if (Serial.available() > 0) {
27     char command = Serial.read(); // Gelen komutu oku
28     // Yeni satır veya taşıma karakterlerini temizle
29     if (command == '\n' || command == '\r') {
30       return;
31     }
32
33     // Komutlara göre motor kontrolü
34     if (command == '1') {
35       motorForward(100); // 200 PWM hızında ileri
36       Serial.println("Motorlar ileri hareket ediyor");
37     }
38     else if (command == '2') {
39       motorBackward(100); // 200 PWM hızında geri
40       Serial.println("Motorlar geri hareket ediyor");
41     }
42     else if (command == '3') {
43       motorRight(100); // 200 PWM hızında sağa dönüş
44       Serial.println("Motorlar sağa dönüyor");
45     }
46     else if (command == '4') {
47       motorLeft(100); // 200 PWM hızında sola dönüş
48       Serial.println("Motorlar sola dönüyor");
49     }
50   } else { ...
51   }
52 }
53
54 // Motorları ileri hareket ettiren fonksiyon
55 void motorForward(int speed) { ...
56 }
57
58 // Motorları geri hareket ettiren fonksiyon
59 void motorBackward(int speed) { ...
60 }
61
62 // Motorları sağa döndüren fonksiyon
63 void motorRight(int speed) { ...
64 }
65
66 // Motorları sola döndüren fonksiyon
67 void motorLeft(int speed) { ...
68 }
69
70 // Motorları durduran fonksiyon
71 void motorStop() { ...
72 }
```

Şekil 2.12: Arduino Mega'nın motor kontrol kodu.

## 2.3. Görüntü İşleme Yöntemleri

### 2.3.2. HSV Tabanlı Görüntü İşleme Algoritması

CNN modelinin gerçek zamanlı çalıştırılmaması nedeniyle, kırmızı topun tespiti için OpenCV kütüphanesiyle HSV tabanlı bir algoritma geliştirilmiştir. Algoritmanın temel bileşenleri şunlardır:

- *Renk Algılama:* Görüntü, RGB'den HSV renk uzayına dönüştürülerek kırmızı renk için iki eşik aralığı ([0, 150, 100] - [10, 255, 255] ve [170, 150, 100] - [180, 255, 255]) tanımlanmıştır.
- *Gürültü Azaltma:* 5x5 çekirdek ile erozyon ve genişletme işlemleri uygulanarak gürültü azaltılmıştır.
- *Kontur Algılama:* En büyük kontur (minimum 500 piksel) seçilerek topun konumu ve sınırlayıcı kutusu ([x, y, w, h]) hesaplanmıştır.
- *Ekran Bölünmesi ve Hareket Kontrolü:* Görüntü üçe bölünerek (sağ, sol, orta) topun merkez noktasına göre hareket belirlenmiş; kontur alanına göre yakınlık tespit edilerek araç durdurulmuştur. Komutlar, 9600 baud hızında seri iletişimle Arduino Mega'ya gönderilmiştir.

## 2.4. Performans Metrikleri

Sistemin performansı, doğruluk, kesinlik, duyarlılık ve F1-skoru metrikleriyle değerlendirilmiştir. Doğruluk, topun konum ve mesafesinin doğru tespitini ölçer; kesinlik ve duyarlılık, sınıflandırma başarısını değerlendirir; F1-skoru ise bunların harmonik ortalamasını sunar. Metrikler, Denklem 1–Denklem 3'te tanımlanmıştır.

## 2.5. Deneysel Çıktılar

Sistemin çalışması sırasında Orange Pi 3 LTS terminalinde elde edilen loglar, kırmızı nesnenin tespit edilip edilmediği ve buna bağlı olarak gönderilen motor kontrol komutlarını yansıtır. Aşağıdaki görseller, deneysel süreçlerin gerçek zamanlı çıktılarını göstermektedir:

- *Şekil 2.13:* Terminalde "Nesne tespit edilmedi" mesajı ve "b" komutu gönderildiği durumun logu. Kırmızı nesne algılanmadığında sistemin varsayılan olarak dur komutunu ilettiği görülmektedir. "Nesne yok - Komut: b (0)" mesajı ve tekrarlanan "Nesne tespit edilmedi" logları. Sistem, nesne bulunmadığında her 3 saniyede bir dur komutunu tekrarlar. "Orta (Küçük) - Komut: a (1)" mesajı ve "a" komutunun gönderildiği durum. Kırmızı nesne orta bölgede tespit edilmiş, ancak kontur alanı büyük eşik değerini aşmadığı için ileri hareket komutu iletilmiştir. "Sağ - Komut: c (3)" mesajı ve "c" komutunun gönderildiği durum. Kırmızı nesne sağ bölgede algılanmış ve sağa dönme komutu iletilmiştir.

```
Seç Komut İstemi
Durum: Nesne tespit edilmedi
Durum: Nesne tespit edilmedi
Kırmızı nesne ortada ama yeteri kadar büyük değil, 'a' komutu gönderildi
Durum: Orta (Küçük) - Komut: a (1)
Kırmızı nesne sağda algılandı, 'c' komutu gönderildi
Durum: Sağ - Komut: c (3)
Kırmızı nesne tespit edilmedi, 'b' komutu gönderildi
Durum: Nesne yok - Komut: b (0)
Durum: Nesne tespit edilmedi
```

Şekil 2.13: Arduino Mega'nın motor kontrol kodu.

- Şekil 2.14: Terminalde "Nesne yok - Komut: b (0)" mesajı ve bağlantı kesintisi (Connection reset) uyarısı. Sistem, bağlantı sorunlarıyla karşılaştığında dur komutunu korumuştur ayrıca sistem hareket kontrolünü dengelemek için her 3 saniyede bir b yani dur komutu göndermektedir.

```
Seç Komut İstemi
Durum: Nesne tespit edilmedi
Durum: Nesne tespit edilmedi
Kırmızı nesne tespit edilmedi, 'b' komutu gönderildi
Durum: Nesne yok - Komut: b (0)
Durum: Nesne tespit edilmedi
Durum: Nesne tespit edilmedi
```

Şekil 2.14: Arduino Mega'nın motor kontrol kodu.

- Şekil 2.15: Terminalde Orange Pi 3 LTS'nin başlangıç ekranı ve "IP kamera açılmadı" hatası ile birlikte "Nesne tespit edilmedi" logları. IP adresi veya protokol hatası nedeniyle görüntü akışının kesildiği bir durum gösterilmektedir.

Bu loglar, sistemin kırmızı nesneyi algılama, konum belirleme ve motor kontrol komutlarını iletme süreçlerini gerçek zamanlı olarak belgelemektedir.

```
Seç Komut İstemi
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\Hasan Mercan>ssh orangepi@192.168.1.28
orangepi@192.168.1.28's password:

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/pro

      _ _ _ _ _
     / _/ _/ _/
    / _/ _/ _/
   / _/ _/ _/
  / _/ _/ _/

Welcome to Orange Pi 3.0.8 Jammy with Linux 5.16.17-sun50iw6

System load: 143%      Up time: 0 min
Memory usage: 21% of 1.94G  IP: 192.168.1.28
192.168.1.18
CPU temp: 73°C      Usage of /: 22% of 30G

[ General system configuration (beta): orangepi-config ]

Last login: Tue May 20 20:29:14 2025 from 192.168.1.21
orangepi@orangepi3-lts:~$ ls
deep Desktop Documents Downloads Music Pictures Public Templates Videos
orangepi@orangepi3-lts:~$ cd deep/
orangepi@orangepi3-lts:~/deep$ cd deep/
orangepi@orangepi3-lts:~/deep/deep$ cat komutlar.txt
source ~/deep/bin/activate
orangepi@orangepi3-lts:~/deep/deep$ source ~/deep/bin/activate
(deep) orangepi@orangepi3-lts:~/deep/deep$ ls
komutlar.txt motorcalistirb3.log motorcalistirb3.py motorcalistir.py motorcalistirv.py

(deep) orangepi@orangepi3-lts:~/deep/deep$ python motorcalistirb3.py
[ WARN:0@35.159] global cap_ffmpeg_impl.hpp:453 _opencv_ffmpeg_interrupt_callback Stream t
imeout triggered after 30080.893127 ms
IP kamera açılmadı: http://192.168.1.14:81/stream. Lütfen IP adresini, portu veya protoko
lü kontrol edin.
(deep) orangepi@orangepi3-lts:~/deep/deep$ python motorcalistirb3.py
[ WARN:0@35.145] global cap_ffmpeg_impl.hpp:453 _opencv_ffmpeg_interrupt_callback Stream t
imeout triggered after 30067.338476 ms
IP kamera açılmadı: http://192.168.1.14:81/stream. Lütfen IP adresini, portu veya protoko
lü kontrol edin.
(deep) orangepi@orangepi3-lts:~/deep/deep$ python motorcalistirb3.py
Kontrol: 'a' (1), 'b' (0), 'c' (3), 'd' (4), 'q' (çık): Durum: Nesne tespit edilmedi
Durum: Nesne tespit edilmedi
Durum: Nesne tespit edilmedi
Durum: Nesne tespit edilmedi
```

Şekil 2.1: Arduino Mega'nın motor kontrol kodu.

### 3. Deneysel Kurulumlar

Deneyler, kontrollü bir iç mekan test alanında gerçekleştirilmiştir. ESP32-CAM ile görüntü toplama, Orange Pi 3 LTS ile görüntü işleme ve Arduino Mega ile motor kontrolü yapılmıştır. Test senaryoları, kırmızı topun farklı konumlarda (sağ, sol, orta) ve mesafelerde (uzak, yakın) takibini içerir. Araç, topun konumuna göre hareket etmiş ve yakınlıkta durmuştur. Deneyler, Python ve OpenCV kullanılarak gerçekleştirilmiş; CNN modeli denenmiş, ancak donanım kısıtlamaları nedeniyle HSV algoritması tercih edilmiştir. Tablo 1, sistemin hiperparametrelerini gösterir.

### 4. Tartışma ve Sonuçlar

Bu çalışma, düşük maliyetli gömülü sistemlerde kırmızı bir topu takip eden otonom bir araç sistemi geliştirmiştir. CNN modeli, Orange Pi 3 LTS'nin sınırlı işlem kapasitesi nedeniyle gerçek zamanlı çalıştırılamamış; bunun yerine HSV tabanlı algoritma kullanılmıştır. Testler, sistemin kontrollü ortamlarda topun konumunu ve mesafesini doğru tespit ettiğini, ancak dinamik aydınlatma ve hızlı hareket senaryolarında performansın düştüğünü göstermiştir. Gelecekte, NVIDIA Jetson gibi güçlü donanımlar, veri artırma ve MobileNet gibi hafif CNN modelleriyle performans artırılabilir. Bu çalışma, otonom araçlarda görüntü işleme tekniklerine teorik ve pratik katkı sunar.

### Kaynaklar

- [1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet classification with deep convolutional neural networks*. Advances in Neural Information Processing Systems, 25, 1097-1105.
- [2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- [3] Abadi, M., et al. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. arXiv preprint arXiv:1603.04467.
- [4] Liu, W., et al. (2016). *SSD: Single shot multibox detector*. European Conference on Computer Vision, 21-37.