



ONDOKUZ MAYIS ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

GÖRÜNTÜ İŞLEME TABANLI NESNE TAKİP EDEBİLEN ARAÇ

PROJE RAPORU

OĞUZHAN BİÇER

MUKADDES KOÇ

MELİKE ÇINAR

Danışman

DR. ÖĞRETİM ÜYESİ GÖKHAN KAYHAN

HAZİRAN, 2022

TEŞEKKÜR

Her zaman yanımızda olan ailelerimize, dört yıl boyunca ve tüm dünyanın fazlasıyla etkilendiği pandemi döneminde her anlamda sorularımıza yanıt veren ve öğrenimden geri kalmamamız için çabalayan hocalarımıza, bu projede yol gösteren danışanımız Gökhan Kayhan’a teşekkür ederiz.

Ö Z E T

Dünya’da her geçen gün artan insan nüfusuyla birlikte otomobil üretimi de artmaktadır. Artan nüfus ve otomobiller trafik sorununa bu durum da kazaların artmasına, mal ve can kaybı yaşanmasına neden olmuştur. Kaza yaşanmasının sebeplerinden biri de sürücü dikkatsizliğidir. Bu sorunun önüne geçmek amacıyla görüntü işleme tabanlı otonom araçlar ortaya çıkmıştır. Bu projede, otonom aracın mekanik ve elektronik tasarımı yapılmış ve aracın nesne takibi için kullanılan görüntü işleme yöntemleri açıklanmıştır. Projenin temel amacı, otonom aracın yapay zeka algoritmaları ile sürüş sırasında çevresinden aldığı görüntüyü işleyerek nesne takibi yapmasıdır.

Görüntü işleme, birden fazla sistemde kullanılmaktadır. Görüntü işlemenin sağlanabilmesindeki en önemli şey çevreyi algılayabilmek ve ayırt edebilmektir. Bu algılama ve ayırt etme işlemleri, derin öğrenme, evrimsel sinir ağları ve algoritmalar ile sağlanmaktadır.

Proje olarak oluşturulan bu görüntü işleme tabanlı otonom araç için uygun donanım seçilerek ve gerekli yazılım sağlanarak araç tasarlanmış ve problemlerin çözümünün sağlanıp sağlanmadığı, ihtiyaçların giderilip giderilmediği tespit edilmiştir.

Anahtar kelimeler: görüntü işleme, nesne takibi, otonom araç, OpenCV ile görüntü işleme, derin öğrenme.

İÇİNDEKİLER

I GİRİŞ

1	AMAÇ	2
2	LİTERATÜR ÖZETİ	3

II MATERYAL

3	OTONOM ARAÇ VE GÖRÜNTÜ İŞLEME	7
3.0.1	Otonom Araçlar	7
3.0.2	Görüntü İşleme	8
3.0.3	Nesne Tespiti	12
3.0.4	Nesne Takibi	14
3.0.5	Derin Öğrenme	16
3.0.6	TensowFlow ile Görüntü sınıflandırma	17
4	ARAÇ TASARIMI	19
4.1	Donanım	19
4.1.1	4WD Çok Amaçlı Mobil Robot Platformu	20
4.1.2	Raspberry Pi 3 Model B	20
4.1.3	Raspberry Pi Kamera Modülü	22
4.1.4	L298N Çift Motor Sürücü Kartı	22
4.2	Yazılım	23
4.2.1	Raspberry Pi OS (Raspbian)	23

4.2.2	Pyhton	24
 III YÖNTEM		
5	PROJEYE GİRİŞ	26
5.0.1	Proje Ortamının Kurulumu	26
5.0.2	OpenCV-Python	27
5.0.3	Numpy kütüphanesi kurulumu	29
5.0.4	HSV	29
5.0.5	Kaynak kodun yorumlanması	31
5.0.6	Proje Bağlantı Şeması	33
 IV SONUÇ		
6	SONUÇ	35
	KAYNAKÇA	37
	EKLER	40

ŞEKİLLER LİSTESİ

1	Otonom araçların kullandığı teknolojiler [1]	8
2	Görüntü işlemenin temel basamakları [2]	9
3	TensowFlow ile nesne tespiti	17
4	TensowFlow ile nesne tespiti 2	18
5	Raspberry Pi 3 Model B [3]	21
6	Raspberry Pi 3 Model B Giriş / Çıkış [4]	21
7	Raspberry Pi 3 Kamera Modülü [5]	22
8	L298N Çift Motor Sürücü Kartı [6]	23
9	IP adresi	27
10	Sarı renkli cismin maskelenmesi	29
11	HSV Kaynak Kodu	30
12	Projenin kaynak kodu	32
13	Bağlantı Şeması	33
14	Eşikleme kaynak kodu	40
15	Bulanıklaştırma kaynak kodu	40
16	Erozyona uğratma kaynak kodu	41
17	Genişleme kaynak kodu	41
18	Kenar yakalama kodu	42
19	Kontur uygulama kaynak kodu	42

20	Renk yakalama kaynak kodu	43
----	-------------------------------------	----

ÇİZELGELER LİSTESİ

BÖLÜM: I

GİRİŞ

A M A Ç

Otonom sürüş, çevresindeki olayları kavrayabilen ve insan müdahalesi olmadan hareket edebilen otomobil sürüş teknolojisidir. Yapılan projenin amacı; otonom bir sürüş gerçekleştirebilmek için araç yapay zeka algoritmalarının sürüş sırasında algıladıklarını ayırt edebilmeyi öğrenmesini ve bu öğrenmeyle aracın hareketinin sağlanmasıdır.

Aracın sokaklarda yön işaretlerini veya engelleri, şeritleri tespit edebilip ona göre doğru bir aksiyon verebilmesi için ön işleme tekniği kullanılır. Bu otonom aracından çekilecek olan görüntüler, otonom aracı kontrol edebilmek için sistem tarafından kullanılacaktır. Aracın algıladıklarını ayırt edebilmesi için sürüş sırasında araçta bulunan (içinde veya üstünde) kameralar sayesinde çevrede bulunan nesneleri kayıt etmesi ve sonrasında bu kameralar ile elde edilen görüntülerle ilişki kurarak bir tepki vermesi hedeflenmektedir. Kısacası bu otonom aracın etrafında gördüğü nesnelere göre ayarlama yapması ve aksiyon almasının sağlanması yapılan projenin hedefidir.

LİTERATÜR ÖZETİ

Otonom araçlar günümüzde yakıt tasarrufu, trafik yoğunluğunun ve kazaların azaltılması gibi birçok amaç için kullanılmaktadır. Tam zamanlı çalışan sistemlerde artık görüntü işleme uygulamalarını yapmak ve kullanmak sık tercih edilir hale gelmiştir. Görüntülerden nesne tespiti yapmak için kullanılan görüntü işleme algoritmaları, yapay zekanın alt dallarından biridir. Görüntü işleme, otonom otomobiller ve insansız hava araçları, engelli ve yaşlı bireyler için yardımcı teknolojiler gibi farklı amaçlar için kullanılmaktadır. Bakıldığı zaman otonom araçların geçmişi 1920-1930’lu yıllarına kadar dayanmaktadır. Kronolojik zamanla incelenmek istenilirse;

- Otonom araçlar 1925 yılında Francis Houdina tarafından ilk kez düşünülmüştür.
- General Motors New York Dünya Fuar’ında 1939 yılında ilk kez halkla paylaşılmıştır.
- 1953’te bir düzen üstünde “RCA Labs” tarafından birkaç kablo ile yön verebilen ve kontrol sağlayabilen minimal bir araç üretilmiştir.
- İlk testler 1958 yılında yapılmıştır. Kendi kendini yönetebilen Chevrolet markalı bir araç üretilmiştir. Pratik bir kullanıma sahip değildir çünkü o dönemin verdiği teknolojik şartlar yüzündendir. Fakat sürücüsüz ilk araç olarak tasarlanmış ve tarihte yerini alabilmiştir.

- General Motors tarafından geliştirilen bir araba, 1962'de Seattle'daki bir ticaret fuarında gösterildi, ancak kayıtlara geçecek bir test yapılmadı. Araç elektronik direksiyon sistemi ile donatılmıştır.
- 1980'de Ernst Dickmanns isimli bir şahıs, Almanya Bundeswehr Üniversitesi'nde kendi ekibiyle bir MercedesBenz minibüs tasarladı. Daha sonra trafiğe kapalı bir alanda araç test edildi ve 63 km/s hıza ulaştı. Ardından EUREKA, 1987-1995 yılları arasında otonom araçlar için 749 milyon Euro değerindeki Prometheus Projesi'ni yürüttü [7].

Eureka Promethus Projesi (Avrupa En Yüksek Verimlilik ve Eşi görülmemiş Güvenlik Trafiği Programı, otonom araçlar alanında bugüne kadar yapılmış en büyük Ar-Ge projesidir. 1995 yılında, 20 yıl önce bilgisayara bilgi aktaran iki kameralı bir araba tasarlamaya başlayan Japon bilim adamları, 1977 yılında arabalarını teste hazırladılar. Yapılan testlerde, bu sürücüsüz otomobil 180 km/s hıza ulaşmayı başardı.

2000'li yıllarda ise otonom araçlar arasındaki yarış başladı. DARPA Grand Challenge, ABD ordusu tarafından ilk olarak 2000 yılında 1 milyon USD ödülle başlatıldı. Ödülü kazanmak için arabaların 142 millik mesafeyi 10 saatte tamamlaması gerekmektedir. Yarışa 15 araba katıldı fakat en iyi araba sadece 7 mill gidebildi [7].

2010 yılında Google, Ford, Mercedes Benz, General Motors Volkswagen, Audi, BMW gibi birçok büyük şirket otonom araç sistemlerini test etmeye başladı. 2013 yılına kadar birçok şirket, kendi ürettikleri araçların projelerini ve araçların gelecek planlarını geliştirmeye devam ettiklerini açıkladı. 2014 yılında Tesla Motors, ilk otopilot modeli olan "Model S." arabalarını tanıttı. Bu model, çeşitli görüntü işleme algoritmalarını kullanarak hız limitleri ve kontrol şeritleri belirleyebilmektedir. Ayrıca otomatik direksiyon ve frenleme özelliğine sahiptir. Model S otomobiller, aldıkları güncellemeler ile yazılımlarını daha da iyileştirmiştir. 2015 yılında Uber Şirketi de otonom araç endüstrisine girdiğini ve 20 adet Ford Fusion satın

aldığını duyurdu.

Mart 2015'te Tesla, San Francisco ve Seattle arasındaki otoyolda bir sürücü ile otopilot sistemini test etti ve araba bu yolculuğu neredeyse hiç yardım almadan yaptı [8]. Sürücü testinden güvenli bir şekilde geçse de ileriki zamanlarda kazalar olmuştur. İlk büyük kaza 2016'da Florida'da Tesla Model S otomatik pilottayken oldu. Sürücü kazada hayatını kaybetti. 2018 yılında da ABD'nin Arizona eyaletinde bir kaza meydana geldi; Uber'in Volvo SC 90 model arabası bir kişiye çarptı ve otonom araçlar ilk kez bir yayanın ölümüne neden oldu. 2016 yılında sürücüsüz otomobillerle ilk taksi hizmeti başladı. Singapur'da nuTonomy tarafından başlatılan hizmet, yolcuları istedikleri varış noktalarına götüren bağımsız bir hizmettir [7].

BÖLÜM: II

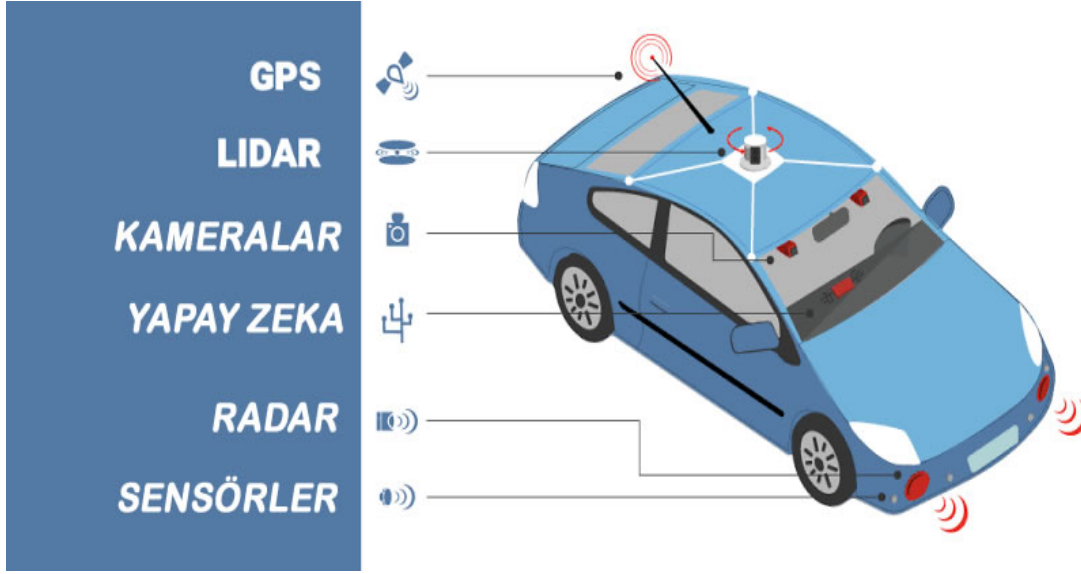
MATERYAL

OTONOM ARAÇ VE GÖRÜNTÜ İŞLEME

Yapılan proje hareketli nesneyi takip eden otonom araç olarak adlandırılabilir. Görüntü işleme tabanlı hareketli nesne takip edebilen aracın çalışma şeklini daha iyi anlayabilmek için otonom araçları, görüntü işleme, nesne tespiti ve takibi, derin öğrenme evrimsel sinir ağları konularını ayrı ayrı anlamak gerekmektedir.

3.0.1 Otonom Araçlar

Otonom, kelime anlamı olarak herhangi bir donanımın otomatik olarak kendini yönetmesi şeklinde ifade edilebilmektedir [9]. Otonom araçlar, otomatik kontrol sistemleri aracılığıyla insana ihtiyaç duymadan çevre koşullarını algılar ve kararlar alıp hareket edebilir. Şekil 1’de ifade edildiği gibi otonom aracın çevresinden bilgi almasını ilk olarak radar, lidar, GPS, odometri, bilgisayar görüşü gibi teknolojiler ve teknikler sağlar [10].

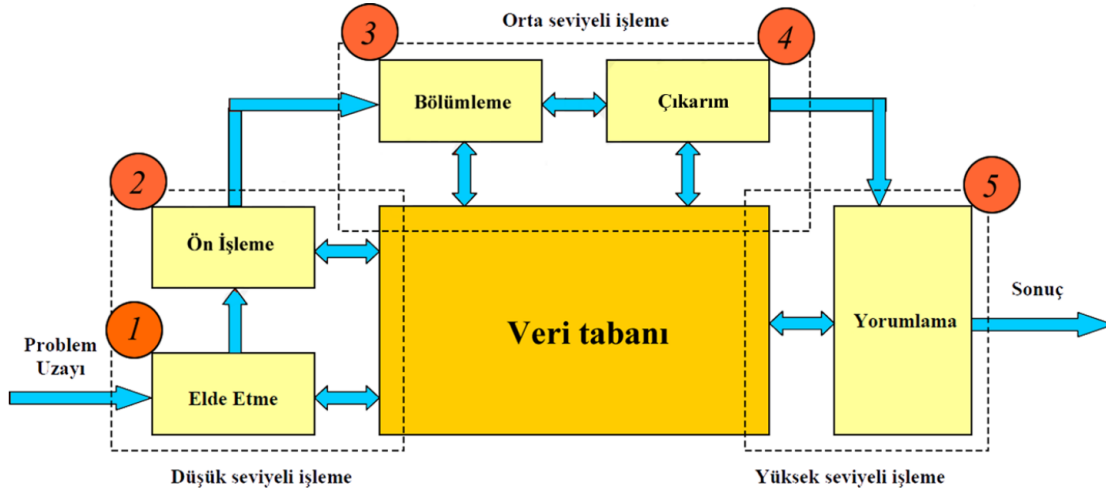


Şekil 1: Otonom araçların kullandığı teknolojiler [1]

3.0.2 Görüntü İşleme

Görüntü, genel anlamda üç boyutlu nesnelerden yansıyan ışığın çeşitli araçlar tarafından algılanarak iki boyutlu surete çevrilmesiyle oluşur. Her görüntü sürekli gibi görünse de piksel parçalarından oluşan iki boyutlu dizidir [11, 2].

Görüntü işleme, bilgisayar, tablet veya telefonlar ile alınan görüntü veya videoları inceleyerek içerisinde bulunan nesneleri tanımlama ve anlama işlemlerini sağlaması için geliştirilmiş, bu görüntü ve tanımlanan nesnelerden anlamlı bilgiler çıkarmak ve bu bilgilerden faydalanmak için kullanılan bir bilgisayar bilimi alanıdır. Görüntü işleme, daha çok öncesinde elde edilen görüntüleri netleştirmek, bulanıklaştırmak, ayırtırmak gibi bir çok işlemi gerçekleştirmek için tercih edilir [12, 13]. Sayısal görüntü işleme, alınan analog görüntünün sayısala dönüştürülmesi ve bilgisayarlarda çeşitli amaçlar için işlemlerden geçmesini sağlar [2]. Temel olarak aşağıda verilen Şekil 2 'deki adımları içerir.



Şekil 2: Görüntü işlemenin temel basamakları [2]

Görüntü işleme uygulamaları, insanlardaki görme sisteminin altında yatan mekanizmaların temel işleyişini taklit etmek amacıyla algılama cihazları, yapay zeka, makine öğrenimi ve derin öğrenmeden gelen girdi değerlerini kullanır. Bu uygulamalar, büyük miktarlarda görsel veri veya görüntülerle eğitilen algoritmalarla çalışır. Uygulamalar genel anlamda görsel verilerdeki desenleri algılayarak diğer görüntülerin içeriğini belirlemeye bu desenler sayesinde yardımcı olurlar. Bu algoritmalar, görüntülerdeki bozuklukların giderilmesini veya kaliteli görüntü haline gelmesini sağlayabileceği gibi nesnelerin tanımlanması, hareketli ve hareketsiz nesnelerin ayrıştırılması veya takibi gibi birçok amaç için de kullanılabilir [14]. Yapılan projenin yazılımında da kullanılan bazı görüntü işleme teknikleri aşağıda verilmiştir.

3.0.2.1 Görüntü işleme teknikleri

Görüntü işleme için öncelikle görüntü formatındaki bir verinin matrisleri ortaya çıkartılmaktadır. Daha sonra ortaya çıkan bu matrisler üzerinde gezilerek istenilen işlemler yapılır. Örneğin, 1920x1080 piksellik bir veri için, 1920x1080 matris oluşturulur. Bu veri renkli olduğunda ise RGB(red, green, blue)'den dolayı matris 3 ile çarpılır [15].

Eşikleme

Görüntü üzerinde yapacağımız işlemler matrislerde gezinerek bize istenilen sonuçları vermektedir. Herhangi bir resimdeki piksellerin genlik değerleri 0 (siyah) ile 255 (beyaz) arasında değişmektedir. Gri filtreli bir görüntü, 0 (beyaz) ile 255 (siyah) arasında değişken piksellere sahip 2 boyutlu bir dizi olarak yorumlanmaktadır. Bir görüntüdeki 125 değerine bir eşik değeri koyarsak, 50'nin üzerindeki renkleri gösterme, 50'nin altındakileri ise göster demiş oluruz. Bu durumda görüntüdeki nesnelerin ana hatları ortaya çıkmış olur.

Bulanıklaştırma

Bulanıklaştırma bir filtre uygulanmasıyla elde edilir. Bu filtreye low pass filter denir. Görüntüden yüksek frekanslı içeriği (parazit, kenarlar v.b.) kaldırmaya yarar. Bu frekans piksellerin değişim hızıdır. Keskin kenarlar yüksek frekanslı bölge olacağından düz alanlar düşük frekanslı içerik olur. Düşük geçişli filtrenin kullanımından dolayı detaylardan feragat etmiş oluruz. 3 tür bulanıklaştırma yöntemi vardır [16]. Bunlar :

1. Ortalama Bulanıklaştırma

Bir görüntüdeki pikselin yanındaki komşu piksel ile ortalamasının alınmasıyla yapılır. Yaygın diğer adları kutu filtreleme. Örneğin 3x3 piksellik bir kutucuk sırasıyla ilgili bölgeyi gezer ve üzerinden geçtiği piksellerin ortalamasını alır. Bir gürültü olduğu taktirde (mesela 120, 121, 122, 45, 127, 128 gibi) ortalamadan dolayı gürültü silinmiş olur. Uygulanması kolay bir yöntemdir.

2. Gauss Bulanıklaştırma

Bu filtreleme yönteminde gauss çekirdeği kullanılır. Mantık aynı fakat gauss çekirdeğinin içerisindeki değerler farklıdır. Gauss fonksiyonları, logaritmanın dördüncü dereceden

fonksiyonlarıdır. İçerisinde sigmalar vardır. Bu çekirdek aynı kutu filtreleme gibi görüntünün üzerinde dolaşır fakat ortalamadan daha çok çekirdeğin içerisinde bulunan önceden belirlenmiş değerlere göre işlem yapar.

3. Medyan Bulanıklaştırma

Bu bulanıklaştırma yöntemi ise ortalama bulanıklaştırma ile hemen hemen aynı işlevi görmektedir. Tek farkı piksellerin ortalamasını almaktansa medyanını alarak işlem yapar.

Morfolojik Operasyonlar

Morfolojik operasyonlar sırasıyla erozyon, genişleme, açılma, kapatma ve morfolojik gradyanlar olmak üzere 5 temel operasyondan oluşmaktadır.

1. Erozyon

Erozyon, parametrelere göre belirtilen bir alanda ön plandaki nesnenin sınırlarını aşındırır, gürültü olarak tanımlanan görüntüdeki bozuklukları temizler. Tüm bunlar görüntünün dizileri üzerinden “ones()” metodu ile belirlediğimiz bir matris boyutunda "1" lerden oluşan bir matris ile geri dönüp “erode()” metodu ile erozyon işlemi gerçekleştirilir.

2. Genişleme

Genişleme denilen kavram erozyondan farklı olarak beyaz olarak belirlenen yerleri inceltmek yerine genişletir. Bu işlem erozyonun bir nevi tam tersidir.

3. Açılma

Açılma, erozyon + genişlemedir. Gürültünün yok edilmesine yardım eder. “random.randint()” fonksiyonu ile görüntü boyutunda matris oluşturulduktan sonra elde ettiğimiz matrisi

"255" ile çarparak "0-255" arasındaki skalaya çıkarıyoruz. Beyaz gürültüyü ekledikten sonra oluşan bu gürültü görüntü ile üst üste geldiğinde gürültülü fotoğraf elde etmiş oluyoruz. "morphologyEx()" metodu ile açılma işlemi yaptıktan sonra görüntünün matrisi float değerlere çevirilmeli. Sonrasında "MORPH_OPEN" metodu ile açılma sağlanmış olur.

4. Kapatma

Açılmanın tam tersidir. Beyaz gürültünün aksine siyah gürültüye ihtiyacımız vardır. Nesnelerin üzerindeki küçük lekeleri yok etmekte kullanılır. Burada ise "255" ile çarpmak yerine "-255" ile çarpmak gerekir. "MORPH_CLOSE" kapatmayı sağlayan metoddur.

5. **Morfolejik Gradyan** Erezyon ve genişleme arasındaki farklılıktır. Bir görüntünün önce genişletip sonra erezyona uğrattıktan sonra bu ikisi arasındaki farkı alırsak morfolejik gradyanı elde etmiş oluruz. "MORPH_GRADIENT" ile oluşturulur.

3.0.2.2 Kullanım Alanları

Görüntü İşleme, teknolojinin gelişmesiyle artık hayatın her alanında yer almaktadır. Askeri, otomotiv, tıp, güvenlik, astronomi, fizik, sanat, robotik, biyomedikal, coğrafya, hayvancılık, gazetecilik, fotoğrafçılık, trafik gibi pek çok alanda insanlara kolaylık sağlamaktadır [17].

3.0.3 Nesne Tespiti

Genellikle nesne sınıflandırma ile karıştırılır. Nesne tespiti resim üzerindeki nesnelerin lokasyon olarak tespitidir. Amaç genişlik ve yükseklik değerlerinin bulunmasıdır. Resim üzerindeki nesnelerin ne olduğunun anlaşılması ise nesne sınıflandırmanın konusudur.

Kenar algılama (Edge Detection)

Bu yöntem ile yüksek frekansa sahip içeriklerin gösterilip, düşük frekansa sahip içeriklerin engellenmesiyle oluşturulur. Kenarlar yüksek frekanslı içeriklerdir ve bu işleme high pass filtering denir. Süreksizliklerin olduğu noktaları bulmayı amaçlayan çeşitli yöntemdir.

Kontur Algılama

Kontur tespiti, kesintisiz bir şekilde devam eden noktaların görüntüsünü ortaya çıkarır. Bu yöntem kullanıcıya nesnenin boyutu ve şekil analizine dair ipuçları verir.

Renk ile Nesne Tespiti

Öncelikle tespit edilmesi istenilen renk seçilir. Rengin aralığı HSV renk formatı için belirlenir. Örneğin mavi için blueLower= (80, 100, 0), blueUpper= (179, 255, 255) gibi. Daha sonra görüntüdeki bozuklukları ve istenmeyen görüntüleri yok etmek için bulanıklaştırma yöntemi uygulanmaktadır. Elde edilen bulanıklaştırılmış görüntüden sonra RGB renk formatından HSV renk formatına geçilir. Tüm bunlar sırasıyla yapıldıktan sonra seçilen renge sahip nesnenin tespit edilebilmesi için bir maske tanımlanır. Maskenin yanındaki nesnelerin temizlenmesi için erozyon ve genişleme yapılır.

Gürültüler giderildikten sonra konturun bulunması gereklidir. Eğer istenilen renk bulunduysa en büyük kontur alınır. Nesneyi çevrelemesi için kutucuk oluşturulur. Nesnenin merkezini bulabilmek için moment hazırlanır. Son olarak ise konturü çizdirip, merkezine bir nokta ile işaretleme yapılır. HSV formatında görüntüler pek anlamlı olmadığı için tekrar orjinal görüntü kullanılır.

3.0.4 Nesne Takibi

Gelişen teknoloji sayesinde bir çok alanda yardımcı olan derin öğrenme, görüntü işleme konusunda da nesne ile ilgili algılama, izleme gibi bir çok alanda yardımcı olmaktadır. Aşağıda derin öğrenmenin nesne ile ilgili işleri nasıl yaptığı anlatılmaktadır.

- Nesne algılamada, resimdeki bir nesneyi algılar, etrafına bir çerçeve koyar ve nesneyi sınıflandırır. Buna çekirdek tabanlı yöntem denir.
- Nesne izleme, bir dizi video karesi boyunca hareket eden nesneleri izlemeyi amaçlayan, bilgisayar görüşü içindeki bir disiplindir.
- Nesneler insanlar, hayvanlar, araçlar, basketbol maçındaki top gibi çeşitli öğeler olabilir.
- Nesne izleme, gözetim, tıbbi görüntüleme, trafik akışı analizi ve insan-bilgisayar analizi gibi birçok pratik uygulamaya sahiptir.

Teknik olarak nesne izleme, görüntüdeki nesneleri tanımlama ve bu tanımlanan nesnelere bir çerçeve atar. Daha sonrasında tanımlanan her nesneye bir kimlik atar ve bunlar hareket ettikçe yeni konumunu belirlemeye çalışır.

- Nesne Takibinin Zorlukları [18]:
 1. Tanımlanan nesnenin videonun bir sonraki karesinde tanımlanamama/işkilendirilememe.
 2. Nesnelerin tahmin edilen aksine hızlı bir şekilde hareket edip görüntüden kaybolması.
 3. Takip edilmek istenen nesnenin önüne veya çevresine izlenmesini engelleyen nesnelerin geçmesi.

4. Birden çok nesnenin takibi sırasında takip edilen nesnelerin kesişip, hangisinin hangisi olduğunun anlaşılmasındaki güçlük.
5. Nesnelerin kendisinden veya kameradan dolayı kaynaklanan hareket bulanıklılığı.
6. Nesnelerin farklı açılarda farklı şekillerde gözükmesinden dolayı oluşabilecek tutarsızlıklar.
7. Nesnelerin, izlenen cihaza yaklaştıkça veya uzaklaştıkça oluşacak olan boyutun büyümesi/küçülmesi durumunda ölçeğinin değişmesi.
8. Işık nedeniyle nesnelerin algılanamaması, takip edilecek olan nesnenin ayırt edilememe zorluğu.

Nesne takibi, temel olarak üç kategoriye ayrılmaktadır:

Nokta Takip Yöntemi

Bu metot görüntüdeki bir nesneye nokta tanımlar ve videonun bir sonraki karesinde o noktaların birbirine paralel olması beklenir. Genellikle hızlı ve kolay olmasından dolayı kalman filtresiyle beraber kullanılmaktadır. Gaussian dağılımına sahip olmayan sistemlerde bu metot başarısız olabilmektedir. Bundan dolayı ise farklı bir yöntem olarak parçacık filtresi geliştirilmiştir [19].

Çekirdek tabanlı yöntem

Bu yöntemde izlenmesi istenen nesneyi çerçeveler. Çerçeve içerisinde bulunan nesneden anlamlı bilgiler çıkarılır ve bu sayede nesne takibi sağlanır. Çerçeveye alınan nesnenin olasılık

yoğunluk bilgileri bir sonraki video çerçevelerinde takip edilebilmeye olanak sağlar. Tek boyutlu ve çok boyutlu olmak üzere 2'ye ayrılır [18].

Silüet Tabanlı Yöntem

Silüet tabanlı yöntem daha çok geometrik şekillerle açıklayamadığımız örneğin bitki, araba, insan gibi nesneleri ifade edemediğimizde kullanılır. Bu yöntemde ki amaç nesnelerin kenar bilgisi veya şekil bilgisinin görünümünü çıkartarak bir sonraki karelerde bu bilgileri aramaktadır. Çekirdek tabanlı yöntemle kıyaslandığında daha iyi işlem zamanı ve daha yüksek başarı oranına sahip olduğundan genelde çekirdek tabanlı yöntem kullanılmaktadır. Nokta tabanlı yöntem diğerlerinden daha iyi sürede işlemi bitirmesine rağmen daha düşük başarı sunmaktadır [19].

3.0.5 Derin Öğrenme

Bir veya birden fazla gizli katman içeren yapay sinir ağları ve benzeri makine öğrenme algoritmalarını kapsayan çalışma alanıdır. Birçok yeni ürün ve işin oluşmasına olanak tanıyan yapay zekanın alt dalıdır. Derin öğrenme ile akıllı telefonlar için sesli asistan, yüz tanıma sistemleri, kanserli hücre tespitleri, insansız deniz ve hava araçları, sürücüsüz otomobiller derin öğrenme modelleriyle gerçekleştirilmektedir.

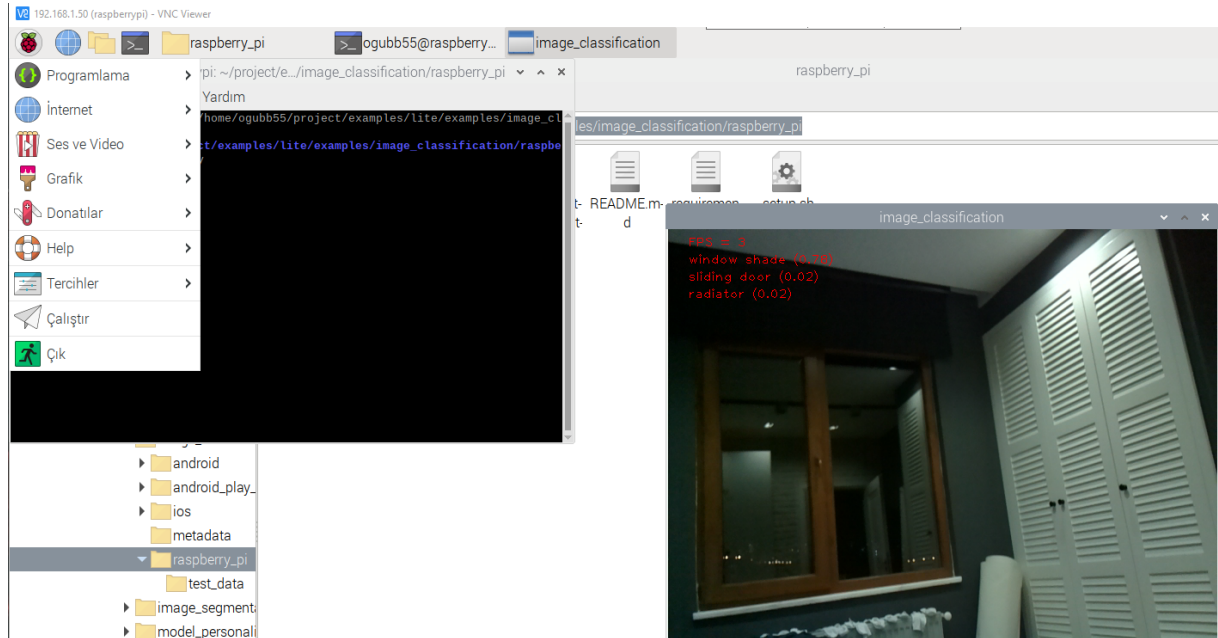
Derin öğrenmenin bir alt dalı olan Evrişimsel Sinir Ağları çoğunlukla görüntüleri analiz etmede kullanılmaktadır. Yapısı gereği evrişimsel sinir ağları giriş değeri olarak resim ya da videolar alır. Tabi resimleri alırken matris formatında alması gerekir.

3.0.6 TensorFlow ile Görüntü sınıflandırma

TensorFlow, Google tarafından geliştirilen açık kaynaklı bir makine öğrenmesi platformudur. TensorFlow ile derin öğrenme analizleri kolayca yapılabilir. Birden fazla farklı projelerde çalışıldığı zaman kütüphanelerin çakışmaması için sanal ortam kurulur. [20]

- mkdir tf_project
- cd tf_project
- pip install tensorflow
- import tensorflow as tf

Ortam kurulduktan sonra tensorflow kurulup import ile tf olarak kısaltılır. Py dosyası çağrıldığında artık gerçek zamanlı nesne tespiti yapılabilir.



Şekil 3: TensorFlow ile nesne tespiti



Şekil 4: TensowFlow ile nesne tespiti 2

ARAÇ TASARIMI

Bu bölümde görüntü işleme tabanlı nesne takip edebilen araç ile ilgili bu çalışma için gerekli görülen donanım ve yazılımlar tanıtılacaktır.

4.1 Donanım

Bu çalışmada kullanılan temel donanım parçaları:

- 4WD Çok Amaçlı Mobil Robot Platformu
- Raspberry Pi 3 Model B
- Raspberry Pi Kamera Modülü
- L298N Çift Motor Sürücü Kartı
- SanDisk 32 GB SD Kart

4.1.1 4WD Çok Amaçlı Mobil Robot Platformu

Dört motorlu çok amaçlı mobil robot platformu kit şeklinde satın alınarak monte edilmiştir.

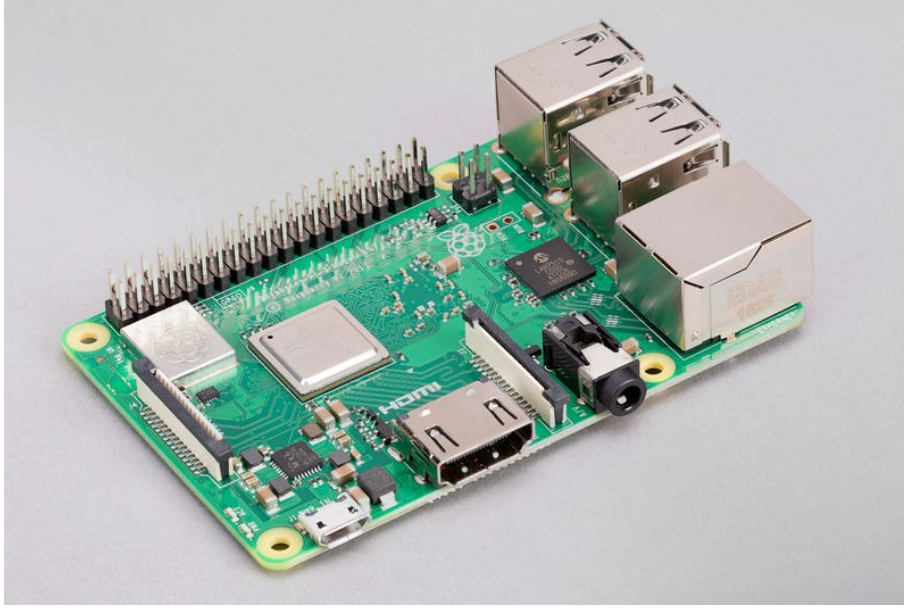
Kit içeriğinde:

1. Üst ve alt pleksi gövde
2. 4 Adet Motor
3. 4 Adet Tekerlek
4. 4 Adet Metal Aralayıcılar
5. 4'lü Pil Yuvası
6. 2 Adet Enkoder Diski
7. Çeşitli vida ve somunlar

bulunmaktadır. Platform üzerine çeşitli kontrol birimi, sensör, modül, kamera ve daha bir çok elektronik araç monte edilebilmektedir.

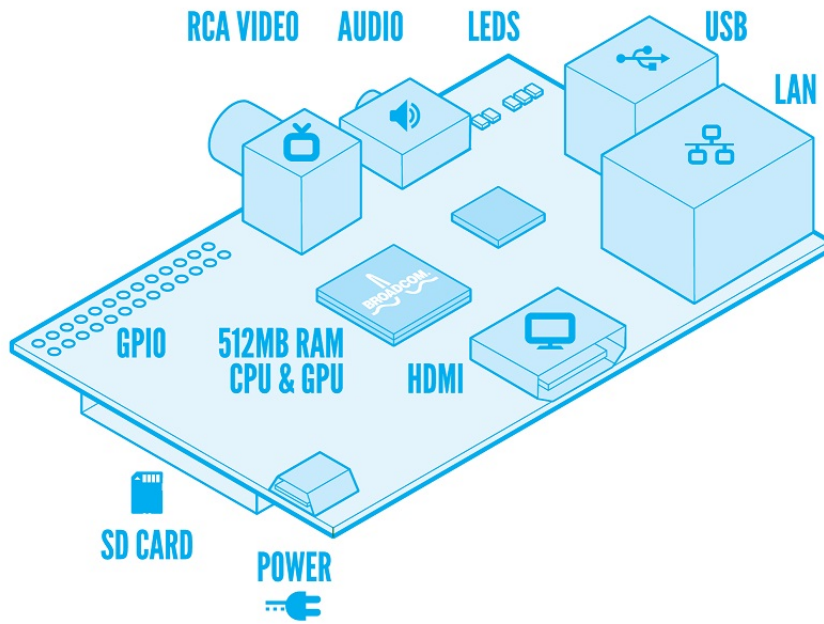
4.1.2 Raspberry Pi 3 Model B

Raspberry Pi Vakfı tarafından desteklenen Raspberry Pi, mini bilgisayar olarak adlandırılmaktadır. İşletim sistemi micro SD karta yüklenmektedir. SD kart, Raspberry Pi'nin üzerinde bulunan SD kart girişine takılır ve böylelikle Raspberry Pi çalıştırılmaktadır. Az enerjiye ihtiyaç duyduğu ve küçük olduğu için gömülü sistemlerde kullanılmaktadır [21].



Şekil 5: Raspberry Pi 3 Model B [3]

RASPBERRY PI MODEL B



Şekil 6: Raspberry Pi 3 Model B Giriş / Çıkış [4]

4.1.3 Raspberry Pi Kamera Modülü

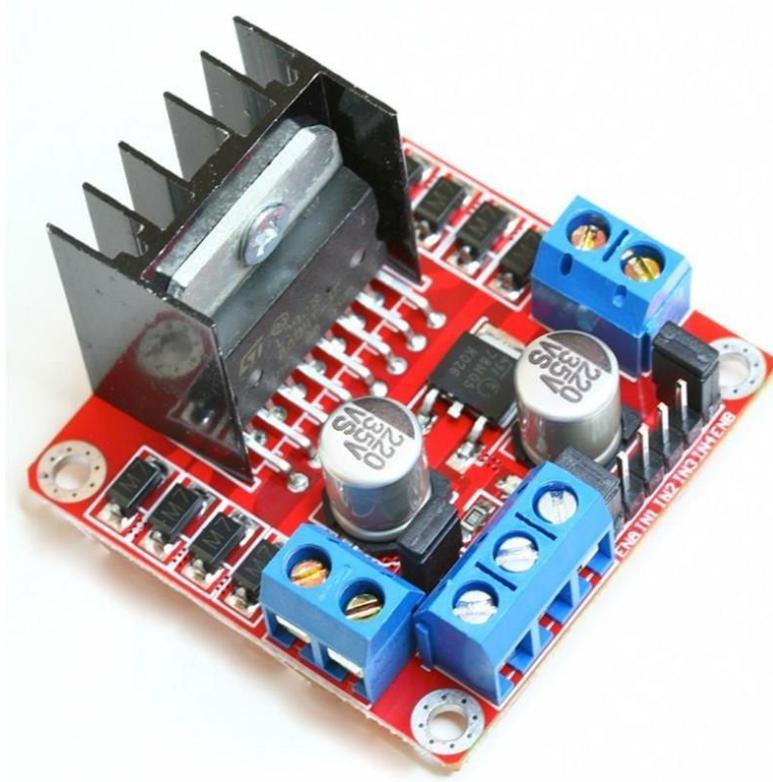
Raspberry Pi 3 bilgisayarını kullanarak oluşturulacak araç için çevresindeki nesneleri algılayabilmesi ve takip edebilmesi için gerekli bir diğer donanım aracı da aracı da Şekil 7’te verilen kamera modülüdür. Bu sayede uzaktan görüntü alınabilecek ve görüntü işlenebilecektir. Kamera modülü Raspberry Pi ile çalışan işletim sistemlerini desteklemektedir.



Şekil 7: Raspberry Pi 3 Kamera Modülü [5]

4.1.4 L298N Çift Motor Sürücü Kartı

İki kanallı bulunmaktadır ve kanal başına 2A akım veren, voltajı en fazla 24V olan motorları sürmek için hazırlanmış olan bir motor sürücü karttır. Kartta L298N motor sürücü entegresi kullanılmıştır. Çok çeşitli robotlarda ve motor kontrol uygulamalarında kullanılmaktadır.



Şekil 8: L298N Çift Motor Sürücü Kartı [6]

4.2 Yazılım

Projede, Raspberry Pi OS işletim sistemi üzerinde Python programlama dili kullanılmıştır. Python ile kameradan alınan girdi değerleri işlenerek aracın nesne takibi yapılabilmesi sağlanmıştır.

4.2.1 Raspberry Pi OS (Raspbian)

Raspberry Pi OS eski adıyla Raspbian, Raspberry Pi için tasarlanmış Debian tabanlı bir işletim sistemidir. Raspberry Pi mini bilgisayarını çalıştırmak için programlar kümesi içermektedir [22].

4.2.2 *Pyhton*

Pyhton 1990 civarında, Amsterdam’da Guido van Rossum tarafından icat edilen basit söz dizimleri sayesinde okunması, yazması ve öğrenilmesi kolay, güçlü bir programlama dilidir. Genellikle nesne yönelimli programlama dili de denilmektedir. Bir çok büyük yazılım projelerinde tercih edilmektedir. Bunun sebeplerinden biri de, projenin prototipinin oluşturmasının ve denenmesinin hızlı, sürdürülebilirlikten uzaklaşmadan gerçekleştirilebilmesidir [23].

BÖLÜM: III

YÖNTEM

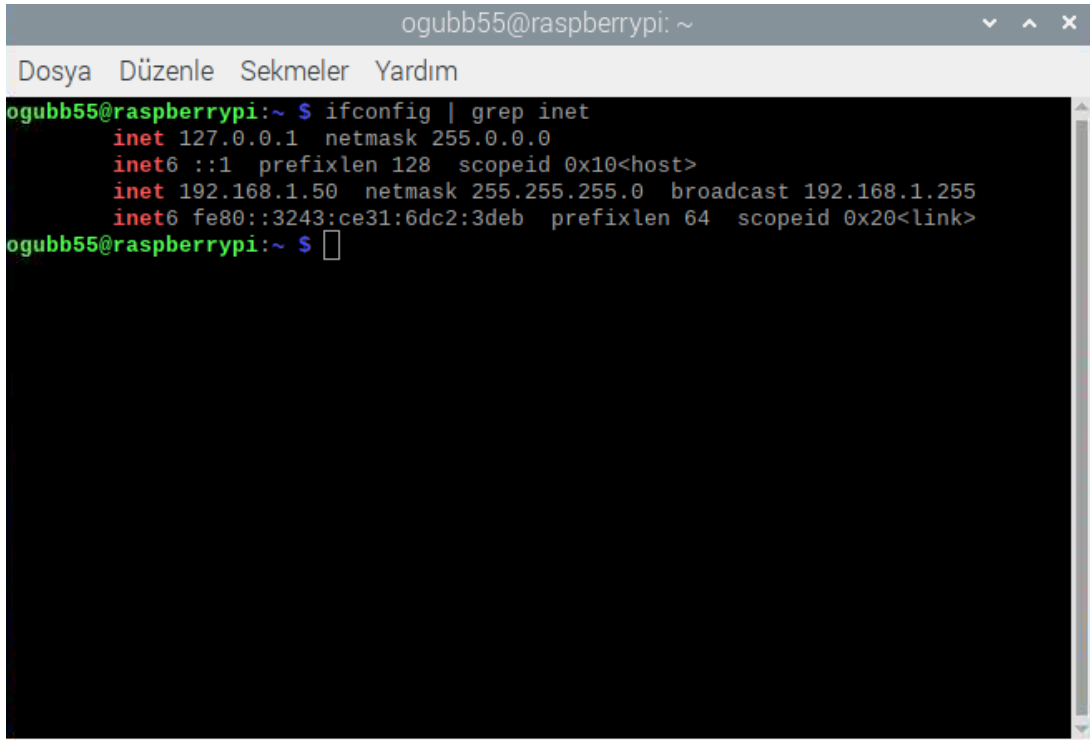
PROJEYE GİRİŞ

5.0.1 *Proje Ortamının Kurulumu*

Öncelikle raspberry pi kullanmak için

1. Raspbian işletim sistemini indirmek gerekir.
2. SD kartı bilgisayara takıp biçimlendirip yazmaya ve okumaya hazır hale getirilmelidir.
3. Ardından indirilen işletim sistemini sd karta yazdırılmalıdır.
4. SD kart raspberry pi kartına takılır ve sistem başlatılıp gerekli konfigürasyonlar yapılır.

Raspberry ile uzaktan bağlanmak için 3 farklı yol vardır. Bunlar; TTL seri kablo ile bağlantı, SSH ile bağlantı ve VNC ile bağlantı. Bu projede VNC Viewer ile uzaktan bağlantı yapılmıştır. VNC ile uzaktan bağlantı için önce raspberry pi üzerinde ayarlardan vnc seçeneğini aktif etmek gerekir ve bir ağ bağlantısına sahip olması gerekir. Ardından bağlantı yapacak bilgisayara Vnc Viewer yüklenilmesi gerekir. Raspberry terminaline girip ip adresimizi öğrenmemiz gerekir. Girdi ve çıktılar şekil 9 ile belirtilmiştir.



```
ogubb55@raspberrypi: ~  
Dosya Düzenle Sekmeler Yardım  
ogubb55@raspberrypi:~ $ ifconfig | grep inet  
inet 127.0.0.1 netmask 255.0.0.0  
inet6 ::1 prefixlen 128 scopeid 0x10<host>  
inet 192.168.1.50 netmask 255.255.255.0 broadcast 192.168.1.255  
inet6 fe80::3243:ce31:6dc2:3deb prefixlen 64 scopeid 0x20<link>  
ogubb55@raspberrypi:~ $
```

Şekil 9: IP adresi

Tüm bunların ardından ip adresini Vnc Viewer adlı programa yazıldığında ilk kez bağlanılacağı için kullanıcı adresi ve şifre istenicektir. Raspberry pi kullanıcı adı ve şifresi girildikten sonra bağlantı gerçekleşir.

5.0.2 OpenCV-Python

OpenCV, Python’da bilgisayar ile görme işlemleri için kullanılan popüler bir kütüphanedir. C++, Java gibi birçok dili ve Windows, Linux, Android gibi birçok işletim sistemini desteklemektedir. Çok boyutlu dizin ve matris veri yapılarını içeren ve bu diziler için üst düzey matematik hesaplamalarını içeren bir kütüphane olan NumPy kütüphanesini kullanmaktadır [24].

OpenCV kurulumu aşağıdaki adımlar izlenilerek yapılmaktadır.

1. Öncelikle sistemi güncellemek için

- `sudo apt-get update` `sudo apt-get upgrade` komutları girilir.

2. Python versiyonu kontrol edilir.

- `python3 -V`

3. Daha sonra pip ve virtualenv kurulumu yapılır.

- `sudo apt-get install python3-pip python3-virtualenv`

4. Ardından env adında ortam oluşturulup aktive edilir.

- `mkdir project`
- `cd project`
- `python3 -m pip install virtualenv`
- `python3 -m virtualenv env`
- `source env/bin/activate`

5. Ardından sistem paketlerinin yüklenmesi gerekir.

- `sudo apt install -y build-essential cmake pkg-config libjpeg-dev libtiff5-dev libpng-dev libavcodec-dev libavformat-dev libswscale-dev libv4l-dev libxvidcore-dev libx264-dev libfontconfig1-dev libcairo2-dev libgdk-pixbuf2.0-dev libpango1.0-dev libgtk2.0-dev libgtk-3-dev libatlas-base-dev gfortran libhdf5-dev libhdf5-serial-dev libhdf5-103 libqt5gui5 libqt5webkit5 libqt5test5 python3-pyqt5 python3-dev`

6. Projede Pi Camera kullanıldığı için pi cameranın sisteme yüklenmesi gerekir.

- `pip install "picamera[array]"`

7. Son olarak OpenCV kütüphanesi yüklenir. Kodun başına time yazılarak işlemin ne kadar süre aldığı öğrenilebilir. Ortalama 3 saat zaman alır.

- `time pip install opencv-contrib-python`

8. Test işlemi için ise

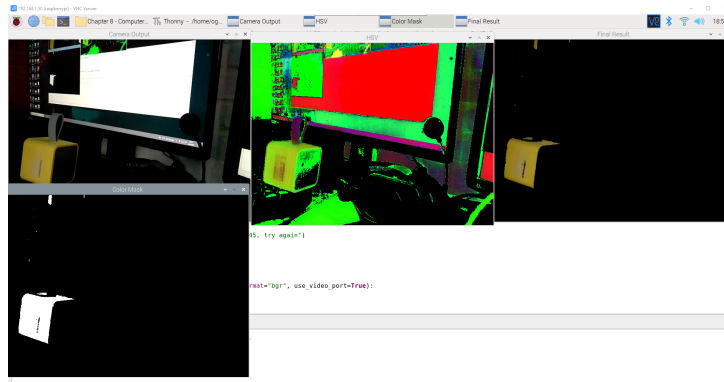
- `python`
- `import cv2`
- `cv2.__version__`

5.0.3 Numpy kütüphanesi kurulumu

Numpy kütüphanesi **sudo pip3 install --upgrade numpy** ile kurulmuştur.

5.0.4 HSV

HSV tester ile bir HSV değeri belirlenir ve ardından maskeleme yapar. Sarı bir cismi, hsv değeri 29 olarak belirlendiğinde başarıyla maskelemiştir. Çıktı Şekil:10 ile gösterilmiştir.



Şekil 10: Sarı renkli cismin maskelenmesi

Kaynak kodu ise Şekil:11 ile gösterilmiştir.

```

hsv_tester.py > ...
1  # gerekli paketlerin import edilmesi gerekir.
2  from picamera.array import PiRGBArray
3  from picamera import PiCamera
4  import time
5  import cv2
6  import numpy as np
7
8  # kamerayı başlatın ve ham kamera çekimine bir referans alın.
9  camera = PiCamera()
10 camera.resolution = (640, 480)
11 camera.framerate = 32
12 rawCapture = PiRGBArray(camera, size=(640, 480))
13
14 while True:
15     while True:
16         try:
17             hue_value = int(input("Hue değerini 10 ile 245 arasında giriniz: "))
18             if (hue_value < 10) or (hue_value > 245):
19                 raise ValueError
20         except ValueError:
21             print("Girdiğiniz değer 10 ile 245 arasında değil, tekrar deneyiniz")
22         else:
23             break
24
25     lower_red = np.array([hue_value-10,100,100])
26     upper_red = np.array([hue_value+10, 255, 255])
27
28     for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
29         image = frame.array
30
31         hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
32
33         color_mask = cv2.inRange(hsv, lower_red, upper_red)
34
35         result = cv2.bitwise_and(image, image, mask= color_mask)
36
37         cv2.imshow("Camera Output", image)
38         cv2.imshow("HSV", hsv)
39         cv2.imshow("Color Mask", color_mask)
40         cv2.imshow("Final Result", result)
41
42         rawCapture.truncate(0)
43
44         k = cv2.waitKey(5) && 0xFF
45         if "q" == chr(k & 255):
46             break
47

```

Şekil 11: HSV Kaynak Kodu

5.0.5 Kaynak kodun yorumlanması

Öncelikle kameranın çözünürlüğü belirlenip ardından bir HUE değeri belirlendi. Sonrasında Motor sürücüsündeki input 1 > GPIO22, input 2 > GPIO27, input 3 > GPIO17, input 4 > GPIO18 e bağlandı. H köprüsünden çıkan GND, raspberry üzerinde 6 numaralı pin olan GND ye bağlanmıştır. İleri gidiş ve dönüş hızları belirlendikten sonra Hue değerinin alt ve üst limiti belirlendi. Bir x ve y kordinatı belirlendikten sonra robotun yön davranışları işlenmeye başlandı. Eğer nesne x kordinatı üzerinde -x değerinde ise robot sola, +x değerinde ise sağa dönecektir. Nesne kameraya yaklaştıkça büyüyeceği için belli bir mesafeden sonra duracaktır, nesne yeterince büyük değilse veya hedef bulunamadıysa aramaya devam edecektir.

Kaynak kodu ise Şekil:12 ile gösterilmiştir.

```

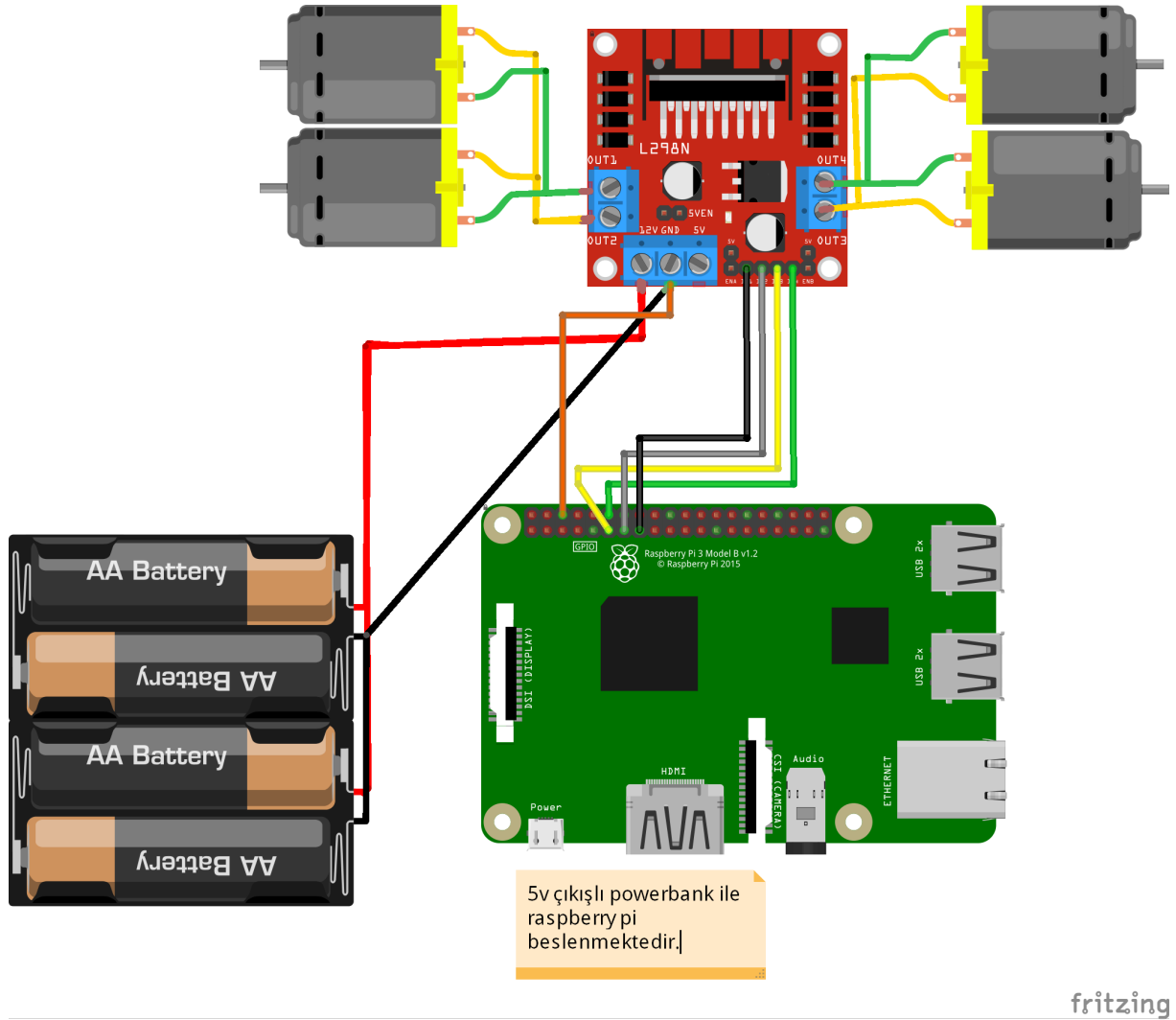
toptakip.py > ...
1  from picamera.array import PiRGBArray
2  from picamera import PiCamera
3  import cv2
4  import numpy as np
5  import gpiozero
6
7  camera = PiCamera()
8  image_width = 640
9  image_height = 480
10 camera.resolution = (image_width, image_height)
11 camera.framerate = 32
12 rawCapture = PiRGBArray(camera, size=(image_width, image_height))
13 center_image_x = image_width / 2
14 center_image_y = image_height / 2
15 minimum_area = 250
16 maximum_area = 100000
17
18 robot = gpiozero.Robot(left=(17,18), right=(27,22))
19 forward_speed = 0.3
20 turn_speed = 0.25
21 HUE_VAL = 28
22
23 lower_color = np.array([HUE_VAL-10,100,100])
24 upper_color = np.array([HUE_VAL+10, 255, 255])
25
26 for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
27     image = frame.array
28     hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
29     color_mask = cv2.inRange(hsv, lower_color, upper_color)
30     contours, hierarchy = cv2.findContours(color_mask, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
31
32     object_area = 0
33     object_x = 0
34     object_y = 0
35
36     for contour in contours:
37         x, y, width, height = cv2.boundingRect(contour)
38         found_area = width * height
39         center_x = x + (width / 2)
40         center_y = y + (height / 2)
41         if object_area < found_area:
42             object_area = found_area
43             object_x = center_x
44             object_y = center_y
45     if object_area > 0:
46         ball_location = [object_area, object_x, object_y]
47     else:
48         ball_location = None
49
50     if ball_location:
51         if (ball_location[0] > minimum_area) and (ball_location[0] < maximum_area):
52             if ball_location[1] > (center_image_x + (image_width/3)):
53                 robot.right(turn_speed)
54                 print("Sağa Dönülüyor")
55             elif ball_location[1] < (center_image_x - (image_width/3)):
56                 robot.left(turn_speed)
57                 print("Sola Dönülüyor")
58             else:
59                 robot.forward(forward_speed)
60                 print("İleri")
61         elif (ball_location[0] < minimum_area):
62             robot.left(turn_speed)
63             print("Hedef yeterince büyük değil, arama devam ediyor")
64         else:
65             robot.stop()
66             print("Hedef çok büyük veya yakın, dur")
67     else:
68         robot.left(turn_speed)
69         print("Hedef bulunamadı, arama devam ediyor")
70     rawCapture.truncate(0)

```

Şekil 12: Projenin kaynak kodu

5.0.6 Proje Bağlantı Şeması

DC motorlar + kutupları + kutuplarıyla, - kutupları ise - kutuplarıyla paralel bağlanıp H köprüsüne bağlantısı yapılmıştır. Projenin bağlantı şeması Şekil:13 ile gösterilmiştir.



Şekil 13: Bağlantı Şeması

BÖLÜM: IV

SONUÇ

SONUÇ

Derin öğrenme bir veya daha fazla yapay sinir ağlarını içeren ve makine öğrenme algoritmalarını kapsayan bir çalışma alanıdır. Derin öğrenmenin görüntü işleme alanında oldukça olumlu sonuçları vardır. Bu çalışmada genel olarak otonom araçların çevresini algılayabilme ve görüntü işleme konuları araştırılmış ve bunun sonucunda araç yapımı planlanmış ve gerçekleştirilmiştir.

Uygulama yazılımı konusunda OpenCV ile görüntü işleme yapılması kanaatine varılmış ve konu hakkında detaylı anlatımlar yapılmıştır. Resim ve videolardaki tespit etme, takip edilmek istenen nesne üzerinde bulanıklaştırmanın ve morfolojik operasyonların yapılış amacı, yapılış şekilleri araştırılmış ve anlatılmıştır. Basit bir görüntüde kenarların nasıl algılandığı ve kontur uygulamanın yolunun açıklanıp nesne tanımlama, takibi ve sınıflandırması konularına değinilmiştir. Kullanılan donanım ürünlerinin özellikleri belirtilmiş, aracın yazılımı için seçilen programlama dilleri ve yöntemleri hakkında bilgi verilmiştir.

Proje için gerekli ortamın elde edilış şekli, seçilen OpenCV'nin kurulumu anlatılmıştır. Ortam hazırlandıktan sonra aracın nesne takibi yapabilmesi için gerekli kaynak kodlar ve kat edilen adımlar hakkında bilgi verilmiş ve görsellerle desteklenmiştir. Proje yazılımından sonra projenin bağlanti şeması şekillendirilmiştir.

Araştırma sonucunda öğrenilen yöntemler ve bilgiler ışığında yazılan kodlar ve elde edilen donanım birleştirilerek belirlenen nesneyi algılayan ve nesne kameraya uzak nesneye doğru hareket eden, nesne kameraya yakınsa duran, görüntüyü kaybederse nesneyi aramaya çalışan otonom bir araç yapımı tamamlanmıştır.

KAYNAKÇA

- [1] Otonom araba görsel. <https://teknoloji-tasarim.com/wp-content/uploads/2019/11/otonom-ara%C3%A7lar.jpg>. Erişim: 2022-5-26.
- [2] Sevinç Talha. Görüntü İşleme nedir ve nerelerde kullanılır? <https://www.ceyrekmuhendis.com/goruntu-isleme-nedir-ve-nerelerde-kullanilir/>. Erişim: 2022-5-23.
- [3] Raspberry pi 3 model b+. <https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/>. Erişim: 2022-5-24.
- [4] Raspberry pi 3 giriş/Çıkış. <https://www.elektrikport.com/teknik-kutuphane/raspberry-pi-nedir-arduino-ile-farklari-nelerdir-/8305#ad-image-0>. Erişim: 2022-5-24.
- [5] Raspberry pi kamera. <https://www.direnc.net/raspberry-pi-kamera-modulu>. Erişim: 2022-5-24.
- [6] L298n voltaj regülatörlü Çift motor sürücü kartı. <https://www.robotistan.com/l298n-voltaj-regulatorlu-cift-motor-surucu-karti>. Erişim: 2022-6-07.
- [7] Göksün Yusuf. Otonom araçlar.

- [8] Feigenbaum Baruch. *AUTONOMOUS VEHICLES: A GUIDE FOR POLICYMAKERS*. Reason Foundation, Mar 2018.
- [9] YEREN Ayşe. Otonom nedir? <https://www.mediaclick.com.tr/tr/blog/otonom-nedir>. Erişim: 2022-5-28.
- [10] Otonom araba. https://tr.wikipedia.org/wiki/Otonom_araba. Erişim: 2022-5-28.
- [11] Cayıroğlu İbrahim. Görüntü İşleme. http://www.ibrahimcayiroglu.com/Dokumanlar/GoruntuIsleme/Goruntu_Isleme_Ders_Notlari-1.Hafta.pdf. Erişim: 2022-5-23.
- [12] Görüntü işleme nedir? <https://azure.microsoft.com/tr-tr/overview/what-is-computer-vision/>. Erişim: 2022-5-23.
- [13] PEKER Musa. Görüntü İşleme tekniği kullanılarak gerçek zamanlı hareketli görüntü tanıma, Jun 2009. Yüksek Lisans Tezi.
- [14] Yılmaz Atınç. Kamera kullanılarak görüntüleme yoluyla gerçek zamanlı güvenlik uygulaması, 2007. Yüksek Lisans Tezi.
- [15] Opencv kütüphanesi ile görüntü İşleme (uygulamalı). <https://medium.com/@adem.akdogan/opencv-k%C3%BCt%C3%BCphanesi-ile-g%C3%B6r%C3%BCnt%C3%BCs%C3%BCn%C3%BClme-uygulamal%C4%B1-af50033f7d8>. Erişim: 2022-5-13.
- [16] Opencv İle görüntü İşleme 2. <https://senolomer0.medium.com/opencv-i/%CC/%87le-g/%C3%B6r%C3%BCnt%C3%BC-i%CC%87%C5%9Fleme-uygulamal%C4%B1-af50033f7d8>. Erişim: 2022-5-29.

- [17] Rafael C Gonzalez and Richard E Woods. *Digital Image Processing, Global Edition*. Pearson Education, London, England, 3 edition, Sep 2017.
- [18] Nesne takibi (object tracking). <https://deryacakmak.medium.com/nesne-takibi-object-tracking-22a9cf5bdcfa>. Erişim: 2022-4-13.
- [19] Hanbay Kazım and Uzen. Hüseyin. Nesne tespit ve takip metotları: Kapsamlı bir derleme. *Tr. Doğa ve Fen Derg. Tr. J. Nature Sci.*, 6(2):4–6, 2017.
- [20] Tensorflow lite python image classification example with raspberry pi. https://github.com/tensorflow/examples/tree/master/lite/examples/image_classification/raspberry_pi. Erişim: 2022-6-07.
- [21] Özdemir Mehmet Fatih. Gömülü sistemlerde yüz tanıma algoritmalarının karşılaştırılması, Jun 2018. Yüksek Lisans Tezi.
- [22] Raspberry pi os. <https://www.raspbian.org/>. Erişim: 2022-5-15.
- [23] Lutz Mark. *Programming Python*. O'reilly, Sebastopol, CA, 2 edition, Mar 2001.
- [24] Opencv nedir?). <https://mesutpiskin.com/blog/opencv-nedir.html>. Erişim: 2022-3-13.

EKLER

```
import cv2
import matplotlib.pyplot as plt

#resmi içe aktar
img = cv2.imread("agac.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.figure()
plt.imshow(img, cmap = "gray")
plt.axis("off")
plt.show()

#eşikleme
_, thresh_img = cv2.threshold(img, thresh = 50, maxval = 255, type = cv2.THRESH_BINARY_INV) #50 nin üstündeki değerleri görmek istemiyoruz

plt.figure()
plt.imshow(thresh_img, cmap = "gray")
plt.axis("off")
plt.show()
```

Şekil 14: Eşikleme kaynak kodu

```
C:\Users\Bicer\Desktop\Derin Öğrenme\python\bulanik.py

kenar_algilama.py* x bulanik.py x

1 import cv2
2 import matplotlib.pyplot as plt
3 import numpy as np
4
5 import warnings
6 warnings.filterwarnings("ignore")
7
8 #blurring (detay azaltır gürültü engeller)
9 img = cv2.imread("yildiz.jpg")
10 img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
11 plt.figure(), plt.imshow(img), plt.axis("off"), plt.title("original"), plt.show()
12
13
14
15 #ortalama bulanıklaştırma yöntemi
16
17 dst2 = cv2.blur(img, ksize = (10,10))
18 plt.figure(), plt.imshow(dst2), plt.axis("off"), plt.title("ortalama blur")
```

Şekil 15: Bulanıklaştırma kaynak kodu


```
import cv2
import matplotlib.pyplot as plt
import numpy as np

#resmi içe aktar

img = cv2.imread("seminer_text.png")
plt.figure(), plt.imshow(img, cmap = "gray")
plt.axis("off"), plt.title("Orijinal Resim")

#erozyon

kernel = np.ones((5,5), dtype = np.uint8)
result = cv2.erode(img, kernel, iterations = 1)
plt.figure(), plt.imshow(result, cmap = "gray")
plt.axis("off"), plt.title("Erozyon ")
```

Şekil 16: Erozyona uğratma kaynak kodu

```
C:\Users\Bicer\Desktop\Derin Öğrenme\python\untitled1.py
kenar_algilama.py x untitled1.py* x
1 import cv2
2 import matplotlib.pyplot as plt
3 import numpy as np
4
5 #resmi içe aktar
6
7 img = cv2.imread("seminer_text.png")
8 plt.figure(), plt.imshow(img, cmap = "gray")
9 plt.axis("off"), plt.title("Orijinal Resim")
10
11
12 #genişleme
13 result2= cv2.dilate(img, kernel, iterations = 1)
14 plt.figure(), plt.imshow(result2, cmap = "gray")
15 plt.axis("off"), plt.title("Genişleme")
16
```

Şekil 17: Genişleme kaynak kodu

```

import cv2
import matplotlib.pyplot as plt
import numpy as np
#resmi içe aktarma
img = cv2.imread("times.jpg", 0)
plt.figure(), plt.imshow(img, cmap= "gray"), plt.axis("off")

edges = cv2.Canny(image = img, threshold1 = 0, threshold2 = 255)
plt.figure(), plt.imshow(edges, cmap= "gray"), plt.axis("off")

med_val = np.median(img)
print(med_val)

low = int(max(0, (1 - 0.33)*med_val))
high= int(min(255, (1 + 0.33)*med_val))

print(low)
print(high)

edges = cv2.Canny(image = img, threshold1 = low, threshold2 = high)
plt.figure(), plt.imshow(edges, cmap= "gray"), plt.axis("off")

#blur
blurred_img = cv2.blur(img, ksize = (3,3))
plt.figure(), plt.imshow(blurred_img, cmap= "gray"), plt.axis("off")

low = int(max(0, (1 - 0.33)*med_val))
high= int(min(255, (1 + 0.33)*med_val))

print(low)
print(high)

edges = cv2.Canny(image = img, threshold1 = 0, threshold2 = 255)
plt.figure(), plt.imshow(edges, cmap= "gray"), plt.axis("off")

```

Şekil 18: Kenar yakalama kodu

```

import cv2
import matplotlib.pyplot as plt
import numpy as np

# resmi içe aktar
img = cv2.imread("contour.jpg",0)
plt.figure(), plt.imshow(img, cmap = "gray"), plt.axis("off")

# farklı sürüm için
# image, contours, hierarch = cv2.findContours(img, cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE)

contours, hierarch = cv2.findContours(img, cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE)

external_contour = np.zeros(img.shape)
internal_contour = np.zeros(img.shape)

for i in range(len(contours)):

    # external
    if hierarch[0][i][3] == -1:
        cv2.drawContours(external_contour, contours, i, 255, -1)
    else: # internal
        cv2.drawContours(internal_contour, contours, i, 255, -1)

plt.figure(), plt.imshow(external_contour, cmap = "gray"), plt.axis("off")
plt.figure(), plt.imshow(internal_contour, cmap = "gray"), plt.axis("off")

```

Şekil 19: Kontur uygulama kaynak kodu

```

import cv2
import numpy as np
from collections import deque

# nesne merkezini depolayacak veri tipi
buffer_size = 16
pts = deque(maxlen = buffer_size)

# mavi renk aralığı HSV
blueLower = (84, 98, 0)
blueUpper = (179, 255, 255)

# capture
cap = cv2.VideoCapture(0)
cap.set(3,960)
cap.set(4,480)

while True:

    success, imgOriginal = cap.read()

    if success:

        # blur
        blurred = cv2.GaussianBlur(imgOriginal, (11,11), 0)

        # hsv
        hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
        cv2.imshow("HSV Image",hsv)

        # mavi için maske oluşturun
        mask = cv2.inRange(hsv, blueLower, blueUpper)
        cv2.imshow("mask Image",mask)
        # maskenin etrafında kalan gürültüleri sil
        mask = cv2.erode(mask, None, iterations = 2)
        mask = cv2.dilate(mask, None, iterations = 2)
        cv2.imshow("Mask + erozyon ve genişleme",mask)

        # kontur
        (contours,_) = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        center = None

        if len(contours) > 0:

            # en büyük konturu al
            c = max(contours, key = cv2.contourArea)

            # dikdörtgene çevir
            rect = cv2.minAreaRect(c)

            ((x,y), (width,height), rotation) = rect

            s = "x: {}, y: {}, width: {}, height: {}, rotation: {}".format(np.round(x),np.round(y),np.round(width),np.round(height),np.round(rotation))
            print(s)

            # kutucuk
            box = cv2.boxPoints(rect)
            box = np.int64(box)

            # moment
            M = cv2.moments(c)
            center = (int(M["m10"]/M["m00"]), int(M["m01"]/M["m00"]))

            # konturu çizdir: sarı
            cv2.drawContours(imgOriginal, [box], 0, (0,255,255),2)

            # merkeze bir tane nokta çizelim: pembe
            cv2.circle(imgOriginal, center, 5, (255,0,255),-1)

            # bilgileri ekrana yazdır
            cv2.putText(imgOriginal, s, (25,50), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (255,255,255), 2)

        # deque
        pts.appendleft(center)

        for i in range(1, len(pts)):

            if pts[i-1] is None or pts[i] is None: continue

            cv2.line(imgOriginal, pts[i-1], pts[i],(0,255,0),3) #

        cv2.imshow("Original Tespit",imgOriginal)

        if cv2.waitKey(1) & 0xFF == ord("q"): break

```

Şekil 20: Renk yakalama kaynak kodu