T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
OTOMOTİV MÜHENDİSLİĞİ ANABİLİM DALI

GÖRÜNTÜ İŞLEME TEKNİKLERİ İLE OTONOM SÜRÜŞ ALGORİTMASI TASARLANMASI

YÜKSEK LİSANS TEZİ

GÖRKEM ŞAHİN

DENİZLİ, HAZİRAN - 2024

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
OTOMOTİV MÜHENDİSLİĞİ ANABİLİM DALI

GÖRÜNTÜ İŞLEME TEKNİKLERİ İLE OTONOM SÜRÜŞ ALGORİTMASI TASARLANMASI

YÜKSEK LİSANS TEZİ

GÖRKEM ŞAHİN

DENİZLİ, HAZİRAN - 2024

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

Görkem ŞAHİN

ÖZET

GÖRÜNTÜ İŞLEME TEKNİKLERİ İLE OTONOM SÜRÜŞ ALGORİTMASI TASARLANMASI
YÜKSEK LİSANS TEZİ
GÖRKEM ŞAHİN
PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
OTOMOTİV MÜHENDİSLİĞİ ANABİLİM DALI

(TEZ DANIŞMANI: DR. ÖĞR. ÜYESİ YALÇIN BULUT)

DENİZLİ, HAZİRAN- 2024

Bu tez çalışması, otonom sürüş alanındaki tekniklerin bir sentezi olup, gelecek araştırmalara ışık tutacak bir temel oluşturmak amacı ile yazılmıştır. Her yıl on binlerce insan trafik kazalarında hayatlarını kaybetmektedir. Bu kazaların pek çok sebebi olsa da en büyük sebebi insan hatalarından kaynaklanmaktadır. Bu tez çalışmasında görüntü işleme teknikleri kullanarak oluşturulan otonom robot, insan hatalarının azaltılması için gerekli olan otonom sürüşe geçiş için daha önce atılan adımlara bir yenisini eklemeyi amaçlamaktadır. Otonom sürüş için gerekli olan yazılım ve donanım bilgisini kazanmak ve bu sektöre yenilikler getirilmesi için çalışılmıştır. Görüntü işleme ile şerit algılama, tabelaları algılama çalışmaları yapılmıştır. Otonom araçların kazalardan kaçınabilmesi için çok hızlı karar almaları gerekmektedir. İnsanların yoldaki tehlikeleri algılamak ve tepki vermek için yaklaşık 390 ila 600 milisaniyeye ihtiyaç duyduğunu gösteren yeni bir çalışma yayınlanmıştır. Genç sürücüler tehlikeleri yaşlı sürücülerden neredeyse iki kat daha hızlı tespit etmektedir. Radar veya Lidar sensörleri ve bir kamera sistemi ile donatılmış sürücüsüz araçların tepki süresi 150-200 milisaniyedir. Sürücüsüz bir araç insanlardan çok daha hızlı tepki vermektedir. Bu çalışmada, piyasada bulunan otonom araçların tepki sürelerine ulaşmak ve daha erken tepki veren bir otonom araç tasarlamak amaçlanmıştır. Çalışmada kullanılan otonom aracın tepki süresi 75-100 milisaniyedir. Piyasada bulunan otonom araçlara göre iki kat daha hızlı tepki veren bir araç tasarlanmıştır.

ANAHTAR KELİMELER: Trafik Kazalarında İnsan Faktörü**,** Otonom Sürüş, Görüntü İşleme, C++ Programlama Dili, Arduino Programlama Dili

ABSTRACT

DESIGNING AUTONOMOUS DRIVING ALGORITHM WITH IMAGE PROCESSING TECHNIQUES
MSC THESIS
GÖRKEM ŞAHİN
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
AUTOMOTIVE ENGINEERING

(SUPERVISOR:DR. ÖĞR. ÜYESİ YALÇIN BULUT)

DENİZLİ, JUNE 2024

This thesis is a synthesis of techniques in the field of autonomous driving and is written to provide a foundation for future research. Every year tens of thousands of people lose their lives in traffic accidents. Although there are many reasons for these accidents, the biggest reason is human error. In this thesis, the autonomous robot created using image processing techniques aims to add a new step to the steps previously taken for the transition to autonomous driving, which is necessary to reduce human errors. It has been studied to gain the software and hardware knowledge required for autonomous driving and to bring innovations to this sector. Lane detection, sign detection and traffic light detection studies were carried out with image processing. Autonomous vehicles need to make very fast decisions to avoid accidents. They published a new study showing that humans need about 390 to 600 milliseconds to perceive and react to hazards on the road. Younger drivers detect hazards almost twice as fast as older drivers. Driverless cars equipped with radar or lidar sensors and a camera system have a reaction time of 150-200 milliseconds. A driverless vehicle reacts much faster than humans. In this study, it is aimed to reach the response times of autonomous vehicles available in the market and to design an autonomous vehicle that reacts earlier. The response time of the autonomous vehicle used in the study is 75-100 milliseconds. A vehicle that reacts twice faster than the autonomous vehicles available in the market is designed.

KEYWORDS: Human Factor in Traffic Accidents**,** Autonomous Driving, Image Processing, C++ Programming Language, Arduino Programming Language

İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

Sayfa

SEMBOL LİSTESİ

Tezde sembol kullanılmamıştır.

ÖNSÖZ

Yüksek lisans eğitimimde bana her konuda destek olan saygıdeğer aile bireylerime ve Dr. Öğr. Üyesi Yalçın BULUT'a teşekkür ederim.

1. GİRİŞ

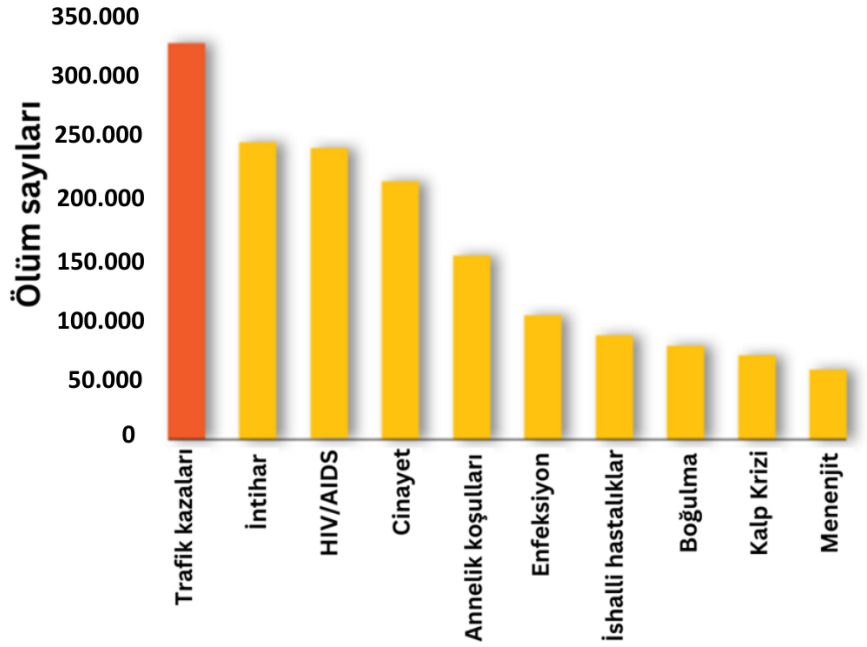
1.1 İnsan Ölümlerinde Trafik Kazalarının Etkisi

Arabalar, milyonlarca insanın günlük yaşamında hareket etmelerini sağlayıp onlara özgürlük duygusu kazandırarak ve şehirlerin gelişiminde kritik bir rol oynayarak, tarihsel evrimde önemli bir yer edinmiştir. Araç kullanmak, birçok insan için rutin bir alışkanlık olsa da en küçük bir dikkatsizlik kazalara sebep olmaktadır. Şekil 1.1'de görüldüğü gibi Türkiye'de her yıl yüz binden fazla insan, karayolu trafik kazalarında hayatını kaybetmektedir (Türkiye İstatistik Kurumu, 2022).

Yıl	Toplam kaza sayısı	Ölümle biten kaza sayısı	Maddi kayıpla biten kaza sayısı
2011	1 228 928	131 845	1 097 083
2012	1 296 634	153 552	1 143 082
2013	1 207 354	161 306	1 046 048
2014	1 199 010	168 512	1 030 498
2015	1 313 359	183 011	1 130 348
2016	1 182 491	185 128	997 363
2017	1 202 716	182 669	1 020 047
2018	1 229 364	186 532	1 042 832
2019	1 168 144	174 896	993 248
2020	983 808	150 275	833 533
2021	1 186 353	187 963	998 390
2022	1 232 957	197 261	1 035 696

Şekil 1.1: 2011-2022 yılları arasında Türkiye'de yaşanan kazalar

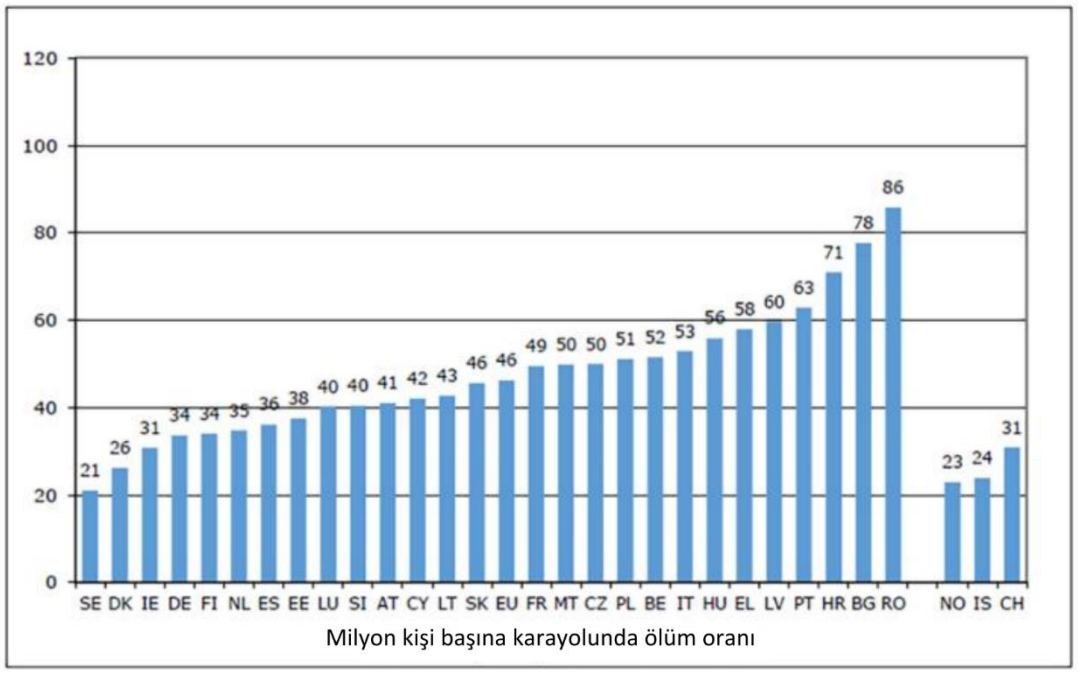
Bu ölümlerin çoğu, hızlı ekonomik büyümeyle birlikte motorlu araçların arttığı ve düşük-orta gelirli ülkelerde karayolu trafik yaralanmalarının yaygın olduğu yerlerde meydana gelmektedir. Karayolu trafik yaralanmaları sadece bir halk sağlığı sorunu değil, aynı zamanda bir kalkınma sorunudur. Şekil 1.2'de görüldüğü gibi düşük ve orta gelirli ülkeler, bu tür kazaların GSYİH (Gayri safi yurt içi hasıla) değerinin yaklaşık %3 kadarını kaybetmeleri nedeniyle ekonomik olarak da etkilenmektedir. (Dünya Sağlık Örgütü, 2015)



Şekil 1.2: 15-29 yaş grubundaki kişiler arasında ilk on ölüm nedeni

Şekil 1.2'de görüldüğü üzere 2012 yılında 15-29 yaşları arasındaki insanların ölüm nedenlerini incelediğimizde, trafik kazaları ölüm sebeplerinin başında gelmektedir. Ölüm oranları; trafik güvenliği önlemleri, yolların durumu, araç güvenlik özellikleri, sürücü davranışları, trafik düzenlemeleri ve sağlık hizmetleri gibi bir dizi faktörden etkilenmektedir. Trafik kazalarının ölüm oranları, birçok faktöre bağlı olarak ülkeden ülkeye, bölgeden bölgeye ve zaman içinde değişmektedir. Ayrıca, bir ülkede ekonomik durum, ulaşım altyapısı ve eğitim seviyesi gibi sosyoekonomik faktörler de trafik kazalarının ölüm oranlarını etkilemektedir. Ülkeler arasında büyük farklılıklar olmakla birlikte trafik güvenliği önlemlerini sıkı bir şekilde uygulayan ülkeler daha güvenli bir trafik ortamına sahiptir. Bu nedenle ölüm oranları düşüktür. Diğer ülkelerde ise trafik güvenliği konusundaki önlemlerin zayıf olması durumunda ölüm oranları artmaktadır.

Şekil 1.3’te görüldüğü üzere 2018 yılında, 28 AB Üye Devleti'nde yaklaşık 25,100 karayolu ölüm vakası rapor edilmiştir. Bu rakam, 2010 yılına kıyasla %21'lik bir düşüşü temsil etmektedir. Geçen yıl, 28 AB Üye Devleti'nde ortalama ölüm oranı 1 milyon kişi başına 49 karayolu ölümü olarak tespit edilmiştir, bu da bir önceki yıla göre %1'lik bir düşüş anlamına gelmektedir. Ancak, bu durum, 2025 yılına kadar karayollarındaki ölümlerin yarıya indirilmesi hedefine ulaşmanın mümkün olmadığını göstermektedir. (Avrupa Komisyonu, 2022).



Şekil 1.3: Ülkelere göre milyon kişi başına düşen ölüm vakaları

Güvenlik öncelikle bir 'insan faktörleri' meselesidir. Sürücü kusurları otoyollardaki kazaların ilk nedeni olarak tespit edilmiştir. Yapılan bir literatür araştırmasına göre, tüm trafik kazalarının yaklaşık %90'ının insan hatalarından kaynaklandığı belirtilmektedir (Smiley ve diğ. 1987).

1.2 Gelişmiş Sürücü Destek Sistemlerinin Tarihi ve İnsan Faktörünün Azaltılması Yönündeki Çalışmalar

Trafik kazalarını önlemek için bilim insanları önce sürücülere yardımcı olan sistemler geliştirmeyi amaçlamıştır. Sürücüye yardımcı olan bu sisteme ADAS

(Advanced Driver Assistance Systems- Gelişmiş Sürücü Destek Sistemleri) ismi verilmiştir.

Gelişmiş Sürücü Destek Sistemlerinin birincil işlevi, gerçek zamanlı tavsiye, talimat ve uyarılar sağlayarak sürücülerin görevini kolaylaştırmaktır. Bu tür sistemler genellikle "yardımcı sürücü sistemleri" veya "sürücü destek sistemleri" terimleriyle de tanımlanmaktadır. Sürücü destek sistemleri danışmanlık, yarı otomatik veya otomatik modda çalışır, bunların hepsi sürüş görevinde sürücüye yardımcı olmayı amaçlamaktadır. Bununla birlikte trafik güvenliğini sağlamak amacıyla geliştirilmiştir. (Rosengren, 1995). Bu konudaki bir örnek, zamanının ötesinde olan DRIVE 1'in en büyük projesi GIDS (Generic Intelligent Driver Support -Genel Akıllı Sürücü Desteği) projesidir. GIDS sürücü destek sisteminin amacı, çok sayıda sensör ve uygulamadan gelen bilgileri filtreleyerek, yorumlayarak ve sunarak araç operatörünü tehdit eden bilgi kirliliğine karşı koymaya yardımcı olmaktır (Michon, 1993). Geçmişte bu alanda profesyonel sürücülerin sürüş koşullarına ilişkin birçok çalışma yapılmıştır. Örneğin, o dönemde bilinen çalışmalar arasında yapılan bir anket, profesyonel sürücülerin %50'sinden fazlasının uykuya daldığını ve kazaya ramak kaldığını itiraf ettiğini bildirmiştir (Ouwerkerk, 1986). Elektronik araç içi teknolojilerinin en önemli özelliklerinden biri sürücülere bilgi sağlamaktır. Sadece yol ortamında bulunan farklı mesaj türleri (yazılı, sesli vb.) aracılığıyla değil, aynı zamanda farklı cihaz türleri (Radar, Lidar vb.) aracılığıyla araç içine entegre edilmelidir (Bekiaris ve diğ. 1997). Sürekli izleme durumunda, bir sürüş görevindeki olaylara tepki süresi bir saniye gibi bir süreyle sınırlandırılırken, birden fazla işlevin izlenmesi gerekiyorsa bu süreler artmaktadır. Aynı zamanda, olası arızanın ve kaynağının tespit edilmesi gerekir ki bu da saniyeler sürebilmektedir. Bu şekilde, iş yükünü azaltmaya yönelik bir girişimin aslında iş yükünün artmasına yol açma olasılığı çok yüksektir (Hancock ve diğ. 1992).

Otomasyonun klasik amacı, insan eliyle kontrol, planlama ve problem çözmenin yerini otomatik cihazların almasıdır. Ancak, bu sistemler hala denetim ve ayarlama (kullanılan sensörlerin kontrolü vb.) için insana ihtiyaç duymaktadır. Bir kontrol sistemi ne kadar gelişmişse, insan operatörün katkısının o kadar azalacağı öne sürülmüştür (Bainbridge, 1983).

ADAS uygulamasında temel bir sorun ADAS'ın kabul edilmesi, yani araç üzerindeki doğrudan kontrolün bir kısmından vazgeçilmesidir. Örneğin, öndeki araca

kısa mesafe kalması durumunda kontrolün devralınması, farklı tipteki Çarpışma Önleme Sistemlerinin devreye girmesidir (Nilsson ve diğ. 1991). Sürücüler bu tür çarpışma önleyici sistemler ve diğer ADAS biçimlerinden olumlu bir güvenlik etkisi beklese de aynı zamanda buna karşı çekinceleri de vardır. Kontrolün bir cihaza devredilmesi ve otomatik frenleme fonksiyonu sistemlerine olan güvensizlik, Gelişmiş Sürücü Destek Sistemlerinin olumsuz yönleri olarak değerlendirilmektedir (Hoedemaeker, 1996).

Günümüzde ADAS (Advanced Driving Assistance Systems- Gelişmiş Sürücü Destek Sistemleri olarak adlandırılan sistem, tam otomatik bir otoyol sistemine giden yoldaki sistemler ve alt sistemler topluluğu olarak düşünülmektedir. (Avrupa Komisyonu, 1998).

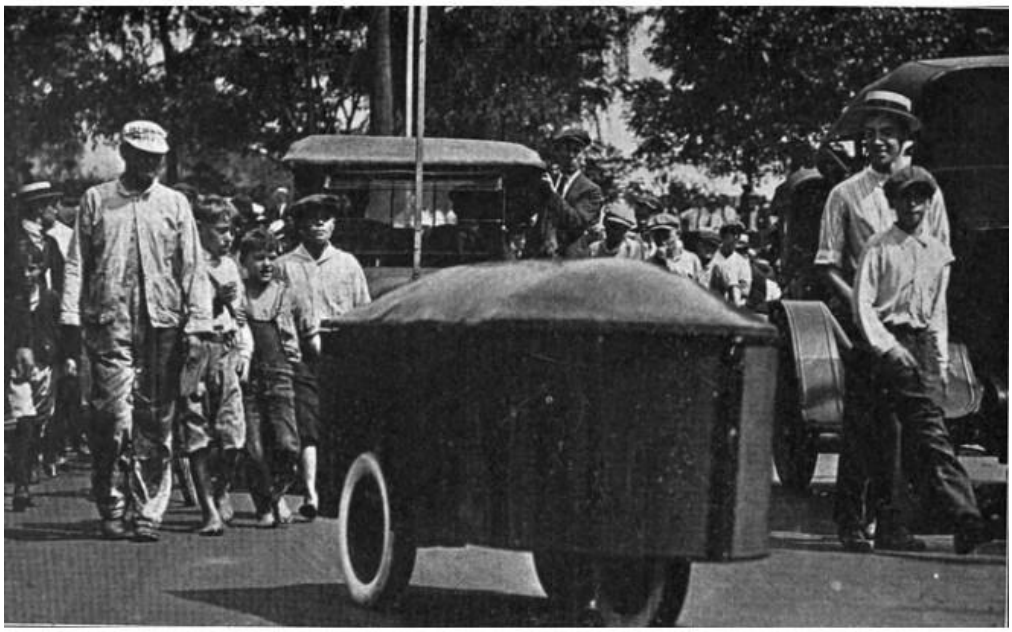
2. OTONOM ARAÇLAR

Otonom araçlar, insan hatasını en aza indirerek trafik güvenliğini arttırmayı amaçlamaktadır, ancak trafik kazalarını tamamen bitirebileceklerinin garantisi yoktur. Otonom araçlar, genellikle sensörler, kameralar, Lidar (Işık Algılama ve Uzaklık Ölçümü) ve Radar (Radyo Algılama ve Uzaklık Ölçümü) gibi teknolojileri kullanarak çevrelerini algılar ve bu verileri işleyerek araç kontrolünü sağlarlar. Bu, sürücü hatalarını, dikkatsizliği ve yorgunluğu azaltır. Ancak, otonom araçlarla ilgili bazı zorluklar ve riskler vardır. Sensörlerin yanlış okuma yapabilmesi, yazılım hataları, çevresel koşulların etkisi (örneğin, yoğun yağmur veya kar), altyapı eksiklikleri (trafik tabelaların bulunmaması vb.) ve diğer araçlarla etkileşim gibi faktörler otonom araçların tamamen güvenilir olmasını zorlaştırır. Ayrıca, otonom araçlarla geleneksel araçlar arasındaki karışık trafik durumları (insan ve yapay zekâ faktörünün aynı anda trafikte bulunması) ve insan sürücülerin beklenmedik davranışları (uyuması, bayılması vb.) gibi faktörler de dikkate alınmalıdır. Sonuç olarak, otonom araçlar trafik kazalarını azaltmayı amaçlar, ancak bu tür kazaları tamamen ortadan kaldırmak için bir dizi teknik, yasal ve altyapı sorunlarının çözülmesi gerekmektedir (Jurvetson, 2009). Şekil 2.1'de Jurvetson ve ekibi tarafından geliştirilen otonom bir aracın test sürüşü görüntülenmektedir.

Şekil 2.1: 2009'da kullanılan bir otonom araç

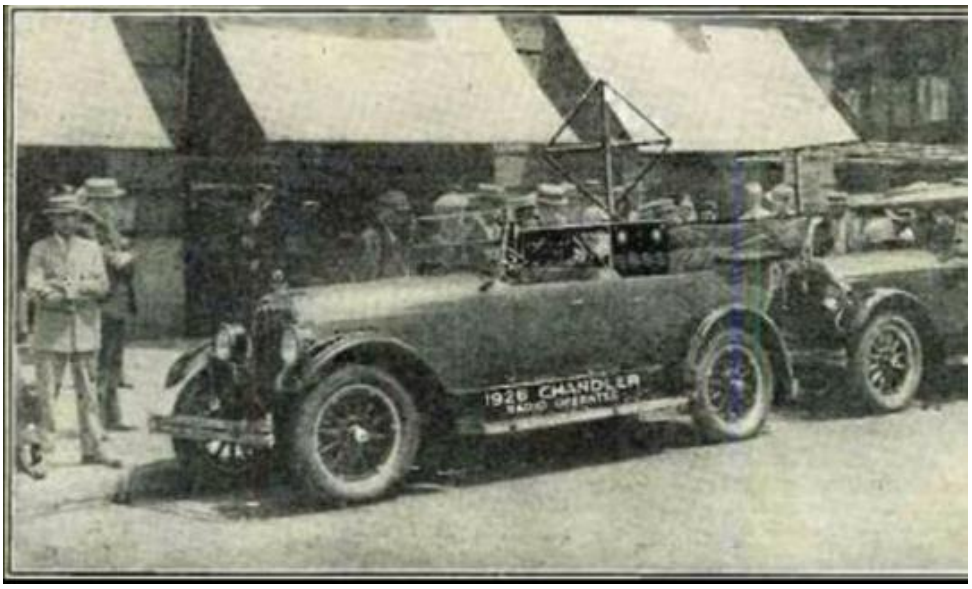
2.1 Otonom Araçların Tarihçesi

Sürücüsüz otomobilin hikayesi neredeyse otomobilin kendisi kadar eskidir. İlk gösterilerden biri 1920'lerde radyo kontrollü sürücüsüz bir otomobildir. Şekil 2.2'de görülen bu araç, öndeki 'sürücüsüz' araca yerleştirilen verici antene radyo sinyalleri göndermek için arkada ikinci bir araca ihtiyaç duymaktadır (The Milwaukee Sentinel, 1926).



Şekil 2.2: İlk sürücüsüz otomobil denemesi **(**The Daily Ardmoreite, 1921**)**

1925 yılında New York Dünya Fuarında General Motors Highways and Horizons (Motorlar Otoyollar ve Ufuklar), sergisinde American Wonder (Amerikan Harikası) aracını tanıttı. Otonom arabalar, aileleri ABD'nin dört bir yanına güvenli ve verimli bir şekilde götürecekti. En azından vizyon buydu. Amerikalı endüstriyel tasarımcı ve aerodinamiğin öncülerinden olan Geddes daha sonra Magic Motorways (Sihirli Otoyollar) adlı kitabında vizyonunu özetlemiş, otoyol tasarımı ve taşımacılığındaki gelişmeleri teşvik etmiş ve insanların araç kullanma sürecinden çıkarılması gerektiğini savunmuştur. Bel Geddes bu ilerlemelerin 1960 yılında gerçeğe dönüşeceğini öngörmüştür (Geddes, 1940). 1960 yılında gerçeğe dönüşmese de fikir değişmemiştir. Altyapıda yapılacak küçük ya da büyük ayarlamalarla otomobil, insan takibi ya da müdahalesi olmadan yol alabilecektir (Fenton ve diğ. 1991). American Wonder aracı Şekil 2.3'te görülmektedir.



Şekil 2.3: American Wonder Aracı

2000 yılı öncesi teknoloji, beklenmedik durumlarla başa çıkmaya hazır değildi, dolayısıyla kontrollü ortamlara ihtiyaç vardı (örneğin otonom araçlara ayrılmış özel otoyollar). Ancak kontrollü ortamlarda teknolojinin çalışması için büyük ölçekli altyapı yatırımları (örneğin kaldırıma teller yerleştirmek) gerekecekti, ancak teknolojinin pazara yeterince nüfuz etmesi sağlanamadığı sürece bu yatırımlar karşılıksız kalmıştır. Bu yüzden Otonom araçlarla ilgili çalışmalara 21.yüzyılın başlarına kadar ara verilmiştir. Bilgisayar ve ağ teknolojilerindeki (örneğin cep telefonları, internet, sensörler) gelişmelere bağlı olarak, sürücüsüz otomobil fikri yeni milenyumda yeniden canlanmıştır. Şekil 2.4'te görüldüğü gibi, Defense Advanced Research Projects Agency- Savunma İleri Araştırma Projeleri Ajansı (DARPA) 2004 yılından başlayarak sürücüsüz araçların tasarımı ve inşası için çeşitli yarışmalar düzenledi. İlk yarışmada hiçbir araç Mojave Çölü'ndeki 240 km'lik parkuru tamamlayamazken, bir yıl sonra 5 araç, en hızlısı ortalama 30,7 km/saat hızla olmak üzere off-road parkurunu tamamladı. Sadece iki yıl sonra, 4 araç trafikte ve trafik kurallarına uyarak 96 km'lik kentsel alan parkurunu bitirebildi. Bu gelişmeler sadece dünyanın dört bir yanındaki araştırma enstitülerinin değil, aynı zamanda sadece ABD'de yıllık 2 trilyon ABD doları hacminde olduğu tahmin edilen taşımacılık pazarı olan endüstrinin de dikkatini çekmiştir (Mui, 2013) (Lamon ve diğ. 2006) (Chong ve diğ. 2011) (Broggi ve diğ. 2013).

Şekil 2.4: Urban Challenge için geliştirilmekte olan bir araç

Son yirmi yılda sürücüsüz araç araştırma platformlarının önemli örnekleri Navlab'ın mobil platformu (Thorpe ve diğ. 1991), Pavia Üniversitesi ve Parma Üniversitesi'nin aracı ARGO (Broggi ve diğ. 1999) ve UBM'nin araçları VaMoRs ve VaMP olmuştur (Gregor ve diğ. 2002).

DARPA meydan okumalarından bu yana, birçok sürücüsüz otomobil yarışması ve denemeler gerçekleştirilmiştir (2004 DARPA Grand Challenge (DARPA, 2004)); 2005 DARPA Grand Challenge (DARPA, 2005); ve 2007 DARPA Urban Challenge (DARPA, 2007)).

İlgili örnekler arasında 2006'dan bu yıla kadar düzenlenen Avrupa Kara-Robot Denemesi (ELROB) (Schneider ve Wildermuth, 2011); 2009-2013 yılları arasında düzenlenen Akıllı Araç Gelecek Mücadelesi (Xin ve diğ. 2014); 2009-2017 yılları arasında düzenlenen Otonom Araç Yarışması (SparkFun, 2018); 2010 yılında düzenlenen Hyundai Otonom Mücadelesi (Cerri ve diğ. 2011); 2010 yılında VisLab Intercontinental Autonomous Challenge (Broggi ve diğ. 2012); 2011 ve 2016 yıllarında düzenlenen Grand Cooperative Driving Challenge (GCDC) (Englund ve

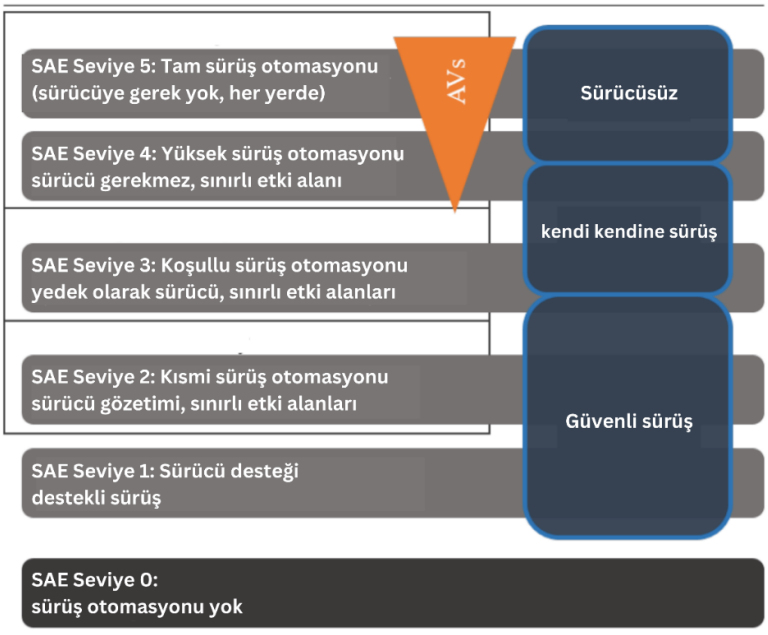
diğ. 2016); ve 2013 yılında düzenlenen Proud-Public Road Urban Sürücüsüz Araç Testi, (Broggi ve diğ. 2015) olmuştur.

2.2 Otonom Araç Seviyeleri

Şekil 2.5'te görüldüğü gibi geçtiğimiz yıllarda, araçların otonomisinin sınıflandırılması için çeşitli öneriler yapılmıştır. SAE (Society of Automotive Engineers -Amerikan Otomotiv Mühendisleri Birliği) 2016 yılında otonom araçları seviye 0 ile seviye 5 arasında 6 farklı kategoriye ayırmıştır. Küçük değişiklikler olsa da güncel olarak kullanılan otonom araç seviyeleri aşağıdaki gibidir.

Seviye 0: Her durumda hala bir insan sürücü tarafından sürülen, ancak çeşitli (birleşik) gelişmiş yardım sistemlerine sahip olan (ör. uyarlanabilir hız sabitleyici, şerit asistanı) sistemlerdir. Bu sistemler, desteksiz sürüşe kıyasla daha kolay, daha konforlu ve en önemlisi daha güvenli bir sürüş deneyimi sağlamaktadır. (SAE International 2016).

Kendi kendine sürüş: Belirli, önceden tanımlanmış durumlarda (örneğin otoyollar veya trafik sıkışıklığı), araç otonom olarak sürebilmekte ve tüm durumların üstesinden gelebilmektedir. Bu durumlarda, araç otonom olarak güvenli bir acil durum duruşuna gelebilir. Kontrol insan sürücüye geçerken (örneğin otoyoldan çıkmadan önce) yeterli



Şekil 2.5: Otonom Araç Seviyeleri **(SAE International, 2016)**

geçiş süresine izin veren açık bir protokol izlenir. Bu, 3. ve 4. seviyelerdeki "yüksek otonomi" kategorisine eşdeğerdir (SAE International 2016).

Sürücüsüz: Araç, her yolda her durumda otonom sürüş yapabilmektedir (SAE International (2016) seviye 5). Hiçbir insan müdahalesi veya izlemesi gerekmez. "Tam otonomi" olarak da adlandırılan bu kategori, bu bölümün başında tanımlanan vizyonu temsil etmektedir.

2.3 Otonom Araçlarda Kullanılan Sensörler

Otonom Araçlarda kullanılan sensörler aşağıda belirtilmiştir. Her sensörün avantajları olduğu gibi dezavantajları da bulunmaktadır Tablo 2.1'de bu sensörlerin boyut, çözünürlük ve menzil gibi özellikleri karşılaştırılmıştır.

2.3.1 Radar (Radyo Algılama ve Menzil)

Radar, (Radio Detection and Ranging- Radyo Algılama ve Menzil) teknolojisinin kısaltması olup, belirli bir aralıktaki nesneleri tespit etmek için radyo dalgalarını kullanan bir cihazdır. İletilen dalgalar yayılma yolu boyunca bir nesneyle kesiştiğinde, Radar anteninin görüş alanı (Field of View- Görüş Alanı) içinde geri saçılan sinyali (yankı) topladığı yüzey tarafından yansıtılır. Gidiş-dönüş gecikme süresi, radyo dalgalarının bilinen hızıyla birlikte, nesnenin Radar sisteminden uzaklığının ve hızının kesin olarak belirlenmesini sağlamaktadır (Buller ve diğ. 2018).

2.3.2 Lidar (Işık Algılama ve Menzil Belirleme)

Lidar, 1970'lerde uzay platformlarında ve hava platformlarında kullanılmak üzere geliştirilmiş bir teknoloji olan Işık Algılama ve Mesafe Ölçmenin kısaltmasıdır. Radar teknolojisine benzer bir şekilde, Lidar sistemleri, bir lazer diyottan yayılan kızılötesi veya yakın kızılötesi aralıklardaki bir ışık darbesinin sistemin alıcısı tarafından alınana kadar geçen sürenin ölçülmesine dayanır; bu aynı zamanda uçuş süresi (Time of Flight) prensibi olarak da bilinir (Wojtanowski ve diğ. 2014).

2.3.3 Ultrasonik Sensör

Ultrasonik sensörler, endüstriyel uygulamalardaki birçok algılama görevi için uygundur. Katı, sıvı, tanecikli veya toz halindeki nesneleri algılama kapasitesine sahiptirler. Ultrasonik sensörler, otomotiv uygulamaları için 40 kHz ila 70 kHz aralığında sonik dalgalar iletmek üzere sonik transdüserlere dayanır. Bu frekans aralığı insanlar için duyulabilir aralığın ötesindedir, bu da insan kulağı için güvenli olmasını sağlar. Bu, bir otomobilin park sisteminin net bir alım sağlamak için 100 dB'den fazla ses basıncı üretebileceği göz önüne alındığında önemli bir faktördür, bu da bir jet motorundan gelen duyulabilir ses basıncına eşdeğerdir (Nagaoka ve diğ. 2018).

2.3.4 Kamera

Kendi kendine giden araçlar, çevredeki ortamı algılamak için büyük ölçüde kameralara güvenir. Elektromanyetik spektruma göre, çoğu kamera görünür veya kızılötesi olarak sınıflandırılmaktadır. Görünür kameralar (örneğin monoküler görüş (Chang ve diğ. 2019), (Kalal ve diğ. 2012) ve stereo görüş (Vatavu ve diğ. 2015), (Erbs ve diğ. 2011)) insan gözüne benzer şekilde 400 ila 780 nm arasında değişen dalga boylarını yakalar.

2.4 Sensörlerin Avantajları ve Dezavantajları

Otonom sürüşte kullanılan sensörlerin çeşitli avantajları ve dezavantajları bulunmaktadır. Bu avantaj ve dezavantajlar otonom aracı tasarlarken, sensörün kullanıp kullanılmayacağı konusunda karar alınmasına yardımcı olacaktır.

2.4.1 Lidar (Işık Algılama ve Menzil Belirleme):

Lidar sensörünün en büyük avantajlarından biri de yüksek çözünürlük ve hassasiyet sağlamasıdır. Aracın çevresinde bulunan nesneleri yüksek çözünürlüklü verilere dönüştürür. Görüntü çözünürlüğü, genellikle piksel olarak ifade edilir. Piksel, bir görüntüyü oluşturan küçük noktalardır. Çözünürlük ne kadar yüksekse, milimetre

kare başına düşen bilgi miktarı o kadar artar, bu da görüntünün daha fazla ayrıntı içermesini sağlar. Lidar sensörünün bir diğer avantajı da hızlı veri toplama yeteneğine sahip olmasıdır. Hızlı veri toplama yeteneği, Lidar sensörünün belirli bir zaman diliminde büyük miktarda veriyi hızlı bir şekilde toplamasını sağlamaktadır. Bu yetenek, gerçek zamanlı uygulamalarda, hızlı karar alma süreçlerinde veya hızlı tepki gerektiren diğer senaryolarda büyük önem taşımaktadır. Lidar sensörünün bir diğer avantajı da farklı ortam koşullarında, özellikle gece ve düşük ışık koşullarında etkilidir. Işığa ihtiyaç duymadan çalışabilen bir sensör olduğu için gece yolculuklarında kullanılmaktadır. Lidar sensörünün dezavantajlarından en önemlisi de maliyetli olmasıdır. Otonom araç geliştirirken dikkat edilmesi gerek huşulardan biri de maliyettir. Bu yüzden Lidar sensörünün tercih edilmesi için maliyetin göz ardı edilmemesi gerekmektedir. Bir diğer dezavantajı ise bazı hava koşullarında (örneğin, yoğun sis veya yağmur) performansı etkilenmektedir. Son olarak diğer sensörlere kıyasla daha dar bir görüş açısına sahiptir.

2.4.2 Radar (Radyo Algılama ve Menzil):

Radar sensörünün en büyük avantajı farklı hava koşullarından etkilenmiyor olmasıdır. Çünkü Radar teknolojisi, elektromanyetik dalgaların kullanılmasıyla çalışır ve bu dalgalar, görsel veya ışığa dayalı sensörlerin aksine atmosferik koşullardan etkilenmez. Diğer bir yandan Radar sensörü, uzun menzile sahiptir ve nesnelerin hızını doğru bir şekilde ölçmektedir. Bu özelliği de otonom araçlarda kullanılan Radar sensörünü en çok tercih edilenlerden birisi yapmaktadır. Bu avantajlarının yanında bazı dezavantajları da vardır. Lidar sensörünün aksine düşük çözünürlüğe sahiptir. Yüksek frekansta çalıştığı için bazen nesneler arasındaki boşluğu belirlemede zorlanmaktadır.

2.4.3 Ultrasonik Sensör

Ultrasonik sensörün en büyük avantajlarından biri yüksek hassasiyete sahip olmasıdır. Nesnelerin konumunu ve mesafesini hassas bir şekilde algılar. Uzak mesafelerde bile doğru ölçümler yapar. Toz, nem, ışık değişimleri gibi çeşitli çevresel

koşullara dayanıklıdır. Diğer sensörlere göre uygun maliyetlidir. Yumuşak yüzeyler veya sıvılar ultrasonik dalgaları soğurur veya yansıtır, bu da doğru mesafe ölçümlerini engellemektedir. Açık alanlarda, yüksek hava hızları veya sıcaklık değişimleri gibi faktörler ultrasonik sensörlerin performansını etkilemektedir. Birden fazla ultrasonik sensör aynı ortamda kullanıldığında, çakışma ve karışma sorunları ortaya çıkmaktadır.

2.4.4 Kamera

Kamera sensörleri, yüksek çözünürlüklü görüntüler sağlar, bu da çevresel detayları daha iyi algılama ve tanımlamasını sağlamaktadır. Renkli görüntüler sağlayabilen kamera sensörleri, nesnelerin ve yol işaretlerinin renklerini ayırt etmesini sağlamaktadır, bu da çevresel bilgiyi daha iyi anlamayı sağlar. Kamera teknolojisinin yaygınlığı ve üretimindeki gelişmeler sayesinde, kamera sensörleri genellikle daha uygun maliyetlidir. Yapay zekâ ve derin öğrenme tekniklerinin kullanılmasıyla, kamera sensörleri nesneleri tanıma ve sınıflandırma yeteneklerini arttırmaktadır.

Kötü hava koşulları, özellikle yoğun yağmur, sis veya kar, kamera sensörlerinin görüşünü engellemekte veya bozmaktadır. Aşırı güneş ışığı veya yansımalı yüzeyler, kamera sensörlerinin performansını olumsuz etkiler. Hızlı hareket eden nesneler veya düşük kontrastlı ortamlar gibi belirli koşullarda, kamera sensörleri nesne algılama hassasiyetinde zorlanmaktadır.

2.4.5 Otonom Araçlarda Kullanılan Sensörlerin Performans Karşılaştırılması

Otonom araçta kullanılacak olan sensörleri seçerken Tablo 2.1’de gösterilen özelliklerden faydalanarak seçim yapılması gerekmektedir. Tasarlanacak olan otonom aracın hangi tür sensörlere ihtiyacı varsa, o sensör tercih edilmesi gerekmektedir.

Tablo 2.1: Otonom Araç Sensörlerinin Karşılaştırılması

	LIDAR	**RADAR**	**KAMERA**	**ULTRASONİK**
Birincil Teknoloji	Lazer ışını	Radyo dalgası	Işık	Ses dalgası
Menzil	~200 m	~250 m	~200 m	~5 m
Çözünürlük	İyi	Ortalama	Çok İyi	Zayıf
Hava koşullarından etkilenir	Evet	Evet	Evet	Evet
Aydınlatma koşullarından etkilenir	Hayır	Hayır	Evet	Hayır
Hızı algılar	İyi	Çok iyi	Zayıf	Zayıf
Mesafeyi algılar	İyi	Çok iyi	Zayıf	İyi
Parazit duyarlılığı	İyi	Zayıf	Çok iyi	İyi
Boyut	Hacimli	Küçük	Küçük	Küçük

3. LİTERATÜRDE GÖRÜNTÜ İŞLEME TEKNİKLERİ

Otonom sürüş, görüntü işleme tekniklerinin önemli bir uygulama alanıdır. Otonom araçlar, çevrelerini algılamak, yol durumunu değerlendirmek ve doğru kararlar almak için görüntü işleme tekniklerinden yararlanmaktadır.

Dijital görüntü işleme, insan yorumu için gelişmiş resimsel bilgi sağlaması ve görüntü verilerinin depolanması, iletilmesi ve makine algısı için temsili için işlenmesini ifade etmektedir. Görüntü İşleme, uydulara, uzay sondalarına ve uçaklara yerleştirilen kameralardan/sensörlerden alınan ham görüntüleri veya çeşitli uygulamalar için normal günlük yaşamda çekilen resimleri geliştirmeye yönelik bir tekniktir. Bu görüntü işleme alanı son zamanlarda önemli ölçüde gelişmiş ve bilim ve teknolojinin çeşitli alanlarına yayılmıştır. Görüntü işleme temel olarak görüntü elde etme, görüntü iyileştirme, görüntü segmentasyonu, özellik çıkarma, görüntü sınıflandırma vb. ile ilgilenir (Chitradevi, B., ve P. Srimathi, 2014). Son elli yıl boyunca görüntü işleme alanında çeşitli teknikler geliştirilmiştir. Tekniklerin çoğu insansız uzay araçlarından, uzay sondalarından ve askeri keşif uçuşlarından elde edilen görüntülerin iyileştirilmesi için geliştirilmiştir. Görüntü İşleme sistemleri, güçlü personel bilgisayarlarının, büyük boyutlu bellek cihazlarının, grafik yazılımlarının vb. kolayca bulunabilmesi nedeniyle popüler hale gelmektedir (Rao, 1989).

3.1 Dijital Görüntü İşleme

Dijital görüntü işleme terimi genellikle iki boyutlu bir resmin dijital bir bilgisayar tarafından işlenmesini ifade etmektedir (Kenneth R. Castleman ve Prentice-Hall, 1996). Dijital Görüntü İşleme teknikleri şunlardır:

- Görüntü ön işleme

- Görüntü iyileştirme

- Görüntü segmentasyonu

- Özellik çıkarma

- Görüntü sınıflandırma

- Yol ve şerit tespiti

- Nesne tespiti

- Trafik işareti tanıma

- Hız ve mesafe ölçümü

3.1.1 Görüntü Ön İşleme

Uydulardan, geleneksel ve dijital kameralardan elde edilen görüntüler, görüntüleme alt sistemlerinin sınırlamaları ve görüntü yakalama sırasındaki aydınlatma koşulları nedeniyle kontrast ve parlaklıktan yoksundur. Görüntüler farklı gürültü türlerine sahip olmaktadır. Görüntü iyileştirmede amaç, sonraki analizler veya görüntü gösterimi için görüntü özelliklerini (çözünürlük, piksel yoğunluğu, kontrast vb.) vurgulamaktır (Gaurav Kumar ve Pradeep Kumar Bhatia, 2014).

3.1.2 Görüntü İyileştirme

Örnekler arasında kontrast ve kenar iyileştirme, sözde renklendirme, gürültü filtreleme, keskinleştirme ve büyütme yer alır. Görüntü iyileştirme, özellik çıkarma, görüntü analizi ve görüntü görüntülemede faydalıdır. İyileştirme işleminin kendisi verilerdeki doğal bilgi içeriğini artırmaz. Sadece belirli görüntü özelliklerini vurgular. İyileştirme algoritmaları etkileşimlidir ve uygulamaya bağlıdır (Chitradevi, B., ve P. Srimathi, 2014).

1. Kontrast Germe

2. Gürültü Filtreleme

3. Histogram değişikliği

3.1.2.1 Kontrast Germe

Bazı görüntüler (örneğin su kütleleri, çöller, yoğun ormanlar, kar, bulutlar ve heterojen bölgeler üzerindeki puslu koşullar altında) homojendir, yani seviyelerinde çok fazla değişiklik yoktur. Histogram gösterimi açısından, çok dar tepelerin oluşumu olarak karakterize edilirler. Homojenlik, sahnenin yanlış aydınlatılmasından da kaynaklanmaktadır (Rao, 1989). Eşit şekilde dağılmış olan piksel değerlerinin siyah ve beyaz değerlere doğru dağıtılması işlemidir.

3.1.2.2 Görüntü Filtreleme

Gürültü Filtreleme, bir görüntüdeki gereksiz bilgileri filtrelemek için kullanılır. Ayrıca görüntülerden çeşitli gürültü türlerini kaldırmak için de kullanılır. Çoğunlukla bu özellik etkileşimlidir. Alçak geçiş, yüksek geçiş, ortalama, medyan vb. gibi çeşitli filtreler mevcuttur (Rao, 1989).

3.1.2.3 Histogram Değişikliği

Histogram, görüntü iyileştirmede çok büyük öneme sahiptir. Görüntünün özelliklerini yansıtır. Histogramı değiştirerek görüntü özellikleri değiştirilmektedir. Histogram eşitleme, bir aralıktaki her değere sahip yaklaşık aynı sayıda piksel olacak şekilde piksel değerlerini yeniden dağıtan doğrusal olmayan bir esnemedir. Sonuç düz bir histograma yaklaşır. Bu nedenle, kontrast tepe noktalarında artmakta ve kuyruklarda azalmaktadır (Rao, 1989).

3.1.3 Görüntü Segmentasyonu

Segmentasyon, görüntü işlemedeki temel problemlerden biridir. Görüntü bölütleme, bir görüntüyü kendisini oluşturan parçalara veya nesnelere ayıran işlemdir. Bu alt bölümlemenin hangi seviyede gerçekleştirileceği çözülmekte olan probleme bağlıdır, yani, bir uygulamada ilgilenilen nesneler izole edildiğinde bölümleme durmalıdır, örneğin, otonom havadan karaya hedef tespitinde, ilgimizin bir yoldaki

araçları tespit etmek olduğunu varsayalım, ilk adım yolu görüntüden bölümlemek ve daha sonra yolun içeriğini potansiyel araçlara kadar bölümlemektir. Görüntü bölütleme için görüntü eşikleme teknikleri kullanılmaktadır (Chitradevi, B., ve P. Srimathi, 2014).

3.1.4 Özellik Çıkarma

Ön işleme ve istenen segmentasyon seviyesine ulaşıldığında, özellikler elde etmek için segmentlere bazı özellik çıkarma teknikleri uygulanır ve bunu sınıflandırma ve son işleme tekniklerinin uygulanması izler. Tanıma sisteminin verimliliği üzerinde gözlemlenebilir bir etkiye sahip olduğu için özellik çıkarma aşamasına odaklanmak önemlidir. Bir özellik çıkarma yönteminin özellik seçimi, yüksek tanıma performansına ulaşmada en önemli faktördür. Özellik çıkarma, "sınıf içi örüntü değişkenliğini en aza indirirken ve sınıflar arası örüntü değişkenliğini artırırken, sınıflandırma amaçları için en uygun olan ham veri bilgisinin çıkarılması" olarak tanımlanmıştır. Bu nedenle, uygulanacak girdiye göre uygun bir özellik çıkarma tekniğinin seçimi son derece dikkatli yapılmalıdır. Tüm bu faktörler göz önünde bulundurulduğunda, belirli bir alanda özellik çıkarımı için mevcut olan ve çok çeşitli durumları kapsayan çeşitli tekniklere bakmak önemli hale gelmektedir (L.Hal, 1979).

3.1.5 Görüntü Sınıflandırma

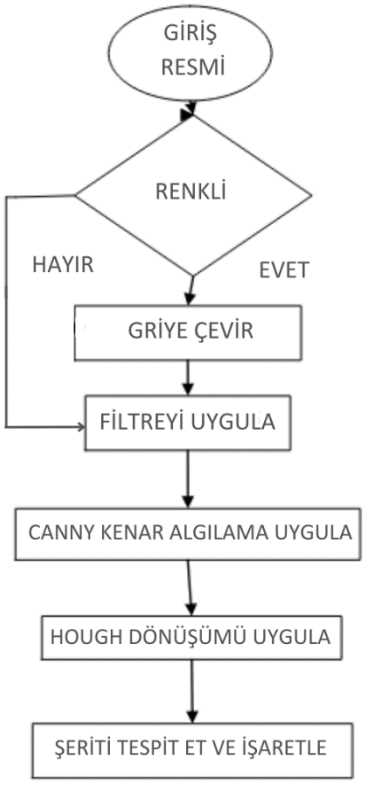
Görüntü sınıflandırma, bir pikselin veya piksel grubunun gri değerine göre etiketlenmesidir (Chellappa, 1992). Sınıflandırma, en sık kullanılan bilgi çıkarma yöntemlerinden biridir. Sınıflandırmada, genellikle bir dizi piksel için birden fazla özellik kullanılır, yani belirli bir nesnenin birçok görüntüsüne ihtiyaç vardır. Uzaktan algılama alanında, bu prosedür belirli bir coğrafi alanın görüntülerinin elektromanyetik spektrumun birden fazla bölgesinde toplandığını ve iyi bir uyum içinde olduğunu varsayar. Bilgi çıkarma tekniklerinin çoğu, bu tür görüntülerin spektral yansıma özelliklerinin analizine dayanır ve çeşitli 'spektral analiz' türlerini gerçekleştirmek için tasarlanmış özel algoritmalar kullanır. Multispektral sınıflandırma işlemi iki yöntemden biri kullanılarak gerçekleştirilmektedir: Denetimli

veya Denetimsiz (Rao, 1989). Gözetimli sınıflandırmada, kentsel, sulak alan, orman vb. gibi bazı arazi örtüsü türlerinin kimliği ve konumu, arazi çalışmaları ve topoğrafyaların bir kombinasyonu yoluyla önsel olarak bilinmektedir. Analist, uzaktan algılanan verilerde bu arazi örtüsü türlerinin homojen örneklerini temsil eden belirli alanları bulmaya çalışır. Bu alanlar genellikle Eğitim Siteleri olarak adlandırılır, çünkü bu bilinen alanların spektral özellikleri, görüntünün geri kalanının nihai arazi örtüsü haritalaması için sınıflandırma algoritmasını 'eğitmek' için kullanılır. Her bir eğitim sahası için çok değişkenli istatistiksel parametreler hesaplanır. Bu eğitim alanlarının içindeki ve dışındaki her piksel daha sonra değerlendirilir ve üyesi olma olasılığı en yüksek olan bir sınıfa atanır (M. Mansourpour ve diğ. 2006).

3.1.6 Yol ve Şerit Tespiti

Uyarlanabilir hız sabitleyici, çarpışma önleme sistemi, gece görüşü, kör nokta algılama ve trafik işareti algılama gibi birçok sistem otonom araçların bir parçasıdır (F. Mariut ve diğ. 2012). Şerit tespiti, yol üzerindeki şerit işaretlerini bulma ve daha sonra bu konumları akıllı bir sisteme sunma işlemidir (S. Srivastava ve diğ. 2014). Şeritleri tespit etmek için kullanılan arayüzlerden bazıları kameralar, lazer mesafe görüntüleri, Lidar ve GPS cihazlarıdır. Önerilen birçok sistemde şerit tespiti, boyalı yolların yüzeyindeki yol işaretleri gibi belirli unsurların konumlandırılmasından oluşur (K. Ghazali ve diğ. 2011). Park etmiş ve hareket eden araçlar, kötü kaliteli çizgiler, ağaçların, binaların ve diğer araçların gölgeleri, keskin virajlar, düzensiz şerit şekilleri, birleşen şeritler, yoldaki yazılar ve diğer işaretler, sıra dışı kaplama malzemeleri ve farklı eğimler gibi çeşitli zorluklar şerit tespitinde sorunlara neden olmaktadır. Şerit tespiti üzerine aktif araştırmalar yapılmış ve çeşitli gösterimler, tespit ve izleme teknikleri ve modalitelerden oluşan çok çeşitli algoritmalar önerilmiştir (Kim, 2008).

Şekil 3.1'de şerit tespitinin genel yöntemi, öncelikle araca sabitlenmiş bir kamera yardımıyla yolun görüntüsünü almaktır. Daha sonra işlem süresini en aza indirmek için görüntü gri tonlamalı bir görüntüye dönüştürülür. İkinci olarak, görüntüdeki gürültü varlığı doğru kenar tespitini engelleyecektir. Bu nedenle, bilateral filtre, gabor filtresi, trilateral filtre gibi gürültüleri gidermek için filtreler uygulanmalıdır. Daha sonra kenar dedektörü, kenarları elde etmek için otomatik eşikleme ile canny filtresi kullanılarak bir kenar görüntüsü üretmek için kullanılır. Daha sonra kenarlı görüntü, sağ ve sol şerit sınır segmenti üretecek olan kenarları tespit ettikten sonra çizgi dedektörüne gönderilir. Şerit sınırı taraması, taramayı gerçekleştirmek için Hough dönüşümü tarafından algılanan kenar görüntüsündeki bilgileri kullanır. Tarama, sağ ve sol tarafta bir dizi nokta döndürür. Son olarak, şerit sınırlarını temsil etmek için bu veri noktalarına bir çift hiperbol yerleştirilir. Görselleştirme amacıyla hiperboller orijinal renkli görüntü üzerinde gösterilir (O.O. Khalifa ve A.H.A Hashim, 2009).



Şekil 3.1: Şerit Algılama Algoritması

3.1.7 Nesne Tespiti

Nesne tespiti, dijital görüntülerde belirli bir sınıftaki görsel nesnelerin (insanlar, hayvanlar veya arabalar gibi) örneklerini tespit etmekle ilgilenen önemli bir bilgisayarla görme görevidir. Nesne tespitinin amacı, bilgisayarla görme uygulamalarının ihtiyaç duyduğu en temel bilgi parçalarından birini sağlayan hesaplamalı modeller ve teknikler geliştirmektir. Nesne tespiti için en önemli iki ölçüt doğruluk (sınıflandırma doğruluğu ve lokalizasyon doğruluğu dahil) ve hızdır (Z Zou ve diğ. 2023).

Viola Jones Dedektörleri: 2001 yılında Viola ve Jones, herhangi bir kısıtlama (örn. ten rengi segmentasyonu) olmaksızın ilk kez insan yüzlerinin gerçek zamanlı tespitini gerçekleştirmiştir (P. Viola ve M. Jones, 2001).

HOG Dedektörleri: 2005 yılında Dalal ve Triggs, Histogram Of Oriented Gradients- Yönelimli Gradyanların Histogramı (HOG) özellik tanımlayıcısını önermiştir. HOG, zamanının ölçekle değişmeyen özellik dönüşümü ve şekil bağlamlarının önemli bir gelişimi olarak kabul edilmektedir. Özellik değişmezliğini (öteleme, ölçek, aydınlatma vb. dahil) ve doğrusal olmamayı dengelemek için HOG tanımlayıcısı, eşit aralıklı hücrelerden oluşan yoğun bir ızgara üzerinde hesaplanacak ve örtüşen yerel kontrast normalizasyonu ("bloklar" üzerinde) kullanılacak şekilde tasarlanmıştır (N. Dalal ve B. Triggs, 2005).

3.1.8 Trafik İşareti Tanıma

Trafik işareti tanıma, bir aracın trafikteki çeşitli trafik işaretlerini algılayıp tanıması ve bu işaretlere uygun şekilde tepki vermesi işlemi olarak tanımlanmaktadır. Otonom sürüş sistemlerinin ve sürücü destek sistemlerinin önemli bir parçası sayılmaktadır çünkü araçların trafik kurallarını doğru bir şekilde takip etmelerine ve güvenli bir şekilde seyretmelerine yardımcı olmaktadır. Araç üzerindeki kameralar aracılığıyla çevredeki trafik işaretlerinin görüntülerinin toplanması gerekmektedir. Bu işaretler genellikle trafik ışıkları, dur işareti, hız limiti işaretleri, dönüş işaretleri, öncelik işaretleri ve diğer yol işaretlerini içermektedir. Toplanan görüntülerin ön işlenmesi, gürültüyü azaltma, kontrast artırma ve boyutunu standartlaştırma gibi

adımları içermektedir. Daha sonra ise makine öğrenmesi, görüntü işleme ve desen tanıma gibi teknikler kullanılmaktadır.

3.1.9 Hız ve Mesafe Ölçümü

Hız ve mesafe ölçümü, bir aracın hızını ve çevresindeki nesnelere olan mesafesini ölçme işlemidir. Otonom sürüş sistemleri ve sürücü destek sistemleri tarafından kullanılmakta ve aracın güvenli bir şekilde seyretmesine yardımcı olmaktadır. Radar, Lidar, Kamera, Ultrasonik sensörler ve hız ölçer gibi farklı teknolojiler kullanılarak hız ve mesafe tespit edilmektedir. Toplanan verilerin işlenmesi ve ön işlenmesi, gürültüyü azaltma, veri formatını standartlaştırma ve veri hızını artırma gibi adımları içermektedir.

3.2 Görüntü İşleme Algoritmaları

Görüntü işleme alanında kullanılan birçok algoritma bulunmaktadır. Bu algoritmalar, farklı görevleri gerçekleştirmek üzere tasarlanmıştır ve çeşitli uygulamalarda kullanılmaktadırlar.

3.2.1 Yönlendirilmiş Gradyanların Histogramı (HOG)

Bu algoritma, yoğun bir ızgarada görüntü gradyan yönelimlerinin iyi normalize edilmiş yerel histogramlarını değerlendirmeye dayanmaktadır. Benzer özellikler son on yılda giderek daha fazla kullanılmaktadır. Temel fikir, yerel nesne görünümü ve şeklinin, ilgili gradyan veya kenar konumları hakkında kesin bilgi olmadan bile, yerel yoğunluk gradyanlarının veya kenar yönlerinin dağılımı ile oldukça iyi karakterize edilebilmesidir (N Dalal ve B Triggs, 2005). Şekil 3.2’de özellik çıkarma ve nesne algılama zincirinin aşamaları gösterilmektedir.

Şekil 3.2: Özellik çıkarma ve nesne algılama zincirinin aşamaları

3.2.2 HOG Avantaj ve Dezavantajları

Yönlendirilmiş Gradyanların Histogramı algoritmasının avantajlarından bazıları; nesnelerin kenarlarını, konturlarını ve diğer önemli özelliklerini başarıyla çıkarmaktadır. Nesneleri doğru bir şekilde tanıma ve sınıflandırmada yüksek başarı sağlamaktadır. Yüksek çözünürlükteki görüntülerde bile nesnelerin önemli özelliklerini belirler, yüksek hassasiyet ve doğruluk sağlar. Görüntünün belirli bölgelerindeki yerel gradyan bilgilerini kullanarak özellik çıkarır. Nesnelerin dönme, ölçek değişimi ve aydınlatma değişimlerine karşı daha dirençli olmasını

sağlamaktadır. Özellik çıkarımı için basit hesaplama işlemleri kullanır ve genellikle hızlı bir şekilde çalışır. Gerçek zamanlı uygulamalarda kullanım için uygun hale getirir. Basit bir algoritma yapısına sahiptir ve farklı nesnelerin tanınması için farklı şekillerde kullanılır. Algoritmanın esnek ve uyarlanabilir olmasını sağlamaktadır.

Yönlendirilmiş Gradyanların Histogramı algoritmasının dezavantajlarından bazıları yalnızca nesnelerin kenarlarını ve konturlarını dikkate alır. Bazı durumlarda nesnelerin izole edilmesi veya birbirinden ayırt edilmesi zor olmaktadır. Görüntünün aydınlatma koşullarına, konumuna ve ölçeğine oldukça duyarlıdır. Değişken ışık koşullarında veya farklı açılardan görüntülerde performansı etkiler. Yüksek çözünürlüklü görüntülerde bu algoritma hesaplama açısından oldukça yoğun olur, daha fazla işlem gücü gerektirir. Tek başına, bazı nesne türlerini veya karmaşık sahneleri tanımlamak için yetersizdir. Diğer özellik çıkarım teknikleri veya derin öğrenme yöntemleriyle birleştirilmesi gerekmektedir.

3.2.3 Kontrast İyileştirme Algoritmaları

Kontrast iyileştirme hem insan hem de bilgisayar görüşü için görüntü işlemede önemli bir alandır. Tıbbi görüntü işleme için ve konuşma tanıma, doku sentezi ve diğer birçok görüntü / video işleme uygulamasında bir ön işleme adımı olarak yaygın olarak kullanılmaktadır (S. C. Pei, Y. C. Zeng, ve C. H. Chang, 2004). Bu yöntemlerden bazıları basit doğrusal/doğrusal olmayan gri seviye dönüşüm fonksiyonlarını kullanırken, bazıları da kenar, bağlı bileşen bilgisi gibi farklı görüntü özelliklerinin karmaşık analizini kullanır (R. C. Gonzalez ve R. E. Woods, 1992).

Histogram Eşitleme: Görüntülerin kontrastını iyileştirmek için kullanılan çok popüler bir teknik histogram eşitlemedir (Jain, 1989). Basitliği ve neredeyse tüm görüntü türlerinden daha iyi performans göstermesi nedeniyle en yaygın kullanılan yöntemdir. Histogram eşitleme, giriş gri seviyelerinin olasılık dağılımına dayalı olarak görüntünün gri seviyelerini yeniden eşleyerek işlemini gerçekleştirir (S.D. Chen, ve A. R. Ramli, 2003).

Histogram Özellikli Eşitleme: Kontrast sınırlarının korunmasını sağlamak için adaptif histogram eşitleme kullanır. Görüntüyü küçük bloklara böler ve her blok için histogram eşitleme uygular.

Gaussian Filtreleme: Gaussian filtreleme, görüntüdeki yüksek frekanslı bileşenleri azaltarak görüntünün kontrastını artırır. Bu, görüntüdeki gürültüyü azaltırken daha yumuşak bir görüntü elde edilmesini sağlar.

Histogram Özleştirme: Görüntüdeki piksel değerlerini belirli bir aralığa özleştirmek için kullanılır. Bu yöntem, görüntünün kontrastını artırırken belirli bir renk aralığında odaklanmayı sağlar.

Kontrast İyileştirme Yöntemleri (örneğin, Yerel Kontrast Geliştirme): Görüntüdeki yerel kontrastı artırmak için kullanılan tekniklerdir. Bu yöntemler, görüntüdeki her bir bölgenin kontrastını ayrı ayrı arttırır.

3.2.4 Kontrast Geliştirme Algoritmaları Avantajları ve Dezavantajları

Algoritmaların avantajları; görüntülerdeki piksel değerlerinin arasındaki farkı artırarak görüntü kalitesini iyileştirir. Görüntülerin daha belirgin hale gelmesini sağlar ve görüntülerin daha canlı ve etkileyici görünmesini sağlamaktadır. Görsel sunumları veya analizleri daha etkili hale getirir. Görüntüdeki nesnelerin veya özelliklerin daha iyi tanınmasını ve algılanmasını sağlamaktadır. Düşük kontrastlı görüntülerde nesnelerin daha net bir şekilde belirginleştirilmesine yardımcı olmaktadır. Düşük kaliteli veya düşük kontrastlı görüntülerin daha çekici ve estetik hale gelmesini sağlamaktadır.

Bu algoritmanın dezavantajları; bazı kontrast geliştirme algoritmaları, görüntülerde istenmeyen deformasyonlara neden olmaktadır. Görüntülerin gerçekliğini bozmakta ve doğruluğunu azaltmaktadır. Kontrast geliştirme algoritmaları, gürültü seviyelerini arttırır ve görüntülerde istenmeyen gürültüyü belirginleştirir.

3.2.5 Titreme ve Yarım Tonlama Algoritması

Titreme ve yarım tonlama algoritmaları, genellikle resimlerin baskı veya ekran üzerinde daha iyi görünmesini sağlamak için kullanılan görüntü işleme tekniklerdir.

Titreme (Dithering): Titreme, bir pikselin yalnızca tek bir renge sahip olamayacağı durumlarda, piksel değerlerini bir dizi desenle karıştırarak yanılsama yaratma işlemidir. Yani, titreme, sınırlı bir renk paletiyle çalışırken daha fazla renk tonunu göstermeye çalışırken ortaya çıkan bir görsel yanılsamadır. Tipik bir titreme algoritması, her bir pikselin değerini, pikselin değerini en yakın renge eşleştirmek yerine, bu renkler arasında ortalamalar alarak veya bir dizi desen kullanarak değiştirir. Düşük renk derinliği veya palet modlarında daha pürüzsüz görüntüler elde etmek için kullanılmaktadır.

Yarım Tonlama (Halftoning): Yarım tonlama, bir resmi siyah-beyaz veya renkli bir resmin siyah-beyaz haline getirilmesi ve ardından bu siyah-beyaz resmin baskıda veya ekran üzerinde daha doğru bir şekilde yeniden üretilmesi için kullanılan bir tekniktir. Yarım tonlama algoritmaları, siyah-beyaz veya renkli bir resmi küçük noktalara veya çizgilere dönüştürerek, bu noktaların veya çizgilerin bir araya gelmesiyle görsel yanılsama oluşturur. Bu noktaların veya çizgilerin büyüklüğü ve düzeni, baskı veya ekran üzerindeki görüntünün algılanan parlaklık ve tonlamasını belirler. Düşük çözünürlüklü veya düşük renk derinliğine sahip ortamlarda görüntülerin daha iyi görünmesini sağlamak için kullanılmaktadır. Ancak, titreme ve yarım tonlama algoritmalarının kullanımı, orijinal görüntünün bazı detaylarının kaybına veya görüntünün pürüzsüzlüğünün azalmasına neden olmaktadır.

3.2.6 Titreme ve Yarım Tonlama Algoritması Avantaj ve Dezavantajları

Titreme (Dithering) Avantajları: Sınırlı renk paletleri veya düşük renk derinliği durumlarında daha fazla renk tonunu simüle etmek için kullanılır. Daha zengin ve daha doğru renkler elde etmeye yardımcı olur. Düşük çözünürlüklü veya palet modlarında kullanıldığında, daha pürüzsüz ve daha doğal görüntüler sağlar.

Titreme Dezavantajları: Titreme, piksel değerlerini değiştirerek görüntülerde detay kaybına neden olur. Düşük kontrastlı alanlarda veya küçük detaylarda görüntülerin netliğini azaltır.

Yarım Tonlama (Halftoning) Avantajları: Yarım tonlama, baskıda daha iyi kalite ve daha doğru renk yeniden üretimi sağlar. Siyah-beyaz veya renkli baskılarda daha fazla tonlamaya olanak tanır.

Yarım Tonlama Dezavantajları: Küçük noktalar veya çizgiler kullanılarak yapılan yarım tonlama, bazı detayların kaybolmasına neden olur. İnce detaylar veya düşük kontrastlı alanlar görüntüde kaybolur.

3.2.7 Başka Fark Algoritması

Başka fark haritası (Binary Feature Difference) algoritması, özellikle stereo görüntüler arasında benzer nesnelerin tespiti ve eşleştirilmesi için kullanılan bir tekniktir. Derin öğrenme tabanlı nesne algılama ve eşleştirme sistemlerinde yaygın olarak kullanılmaktadır. Stereo iki kamera ile farklı noktalardan elde edilen görüntülerin birleştirilip mesafe hesabını mümkün hale getiren pasif ölçüm tekniğidir.

Öznitelik Çıkarımı: İki görüntü arasındaki benzer özellikleri tanımlamak için öznitelik çıkarımı yapılır. Öznitelikler, görüntülerdeki belirgin noktalar, kenarlar veya diğer dikkate değer yapılar olabilir.

Eşleştirme: İki görüntüdeki öznitelikler arasında bir eşleme yapılır. Benzer özniteliklere sahip noktaların belirlenmesini içerir.

Başka Fark Haritası Oluşturma: Eşleştirilmiş noktalar arasındaki farklar hesaplanır ve bir başka fark haritası oluşturulur. İki görüntü arasındaki benzerliklerin ve farkların bir göstergesidir.

Eşikleme ve Binarizasyon: Başka fark haritası, belirli bir eşik değeri üzerinde bölümlere ayrılır ve binarize edilir. ("Binarize etmek", bir görüntüyü veya bir veri kümesini ikili bir formata dönüştürmek anlamına gelir.) Bu, farklılık haritasının yalnızca belirli bir eşik değerinin üzerindeki farkları göstermesini sağlar.

Nesne Tespiti ve Eşleştirme: Binarize edilmiş başka fark haritası, nesne tespiti ve eşleştirme işlemlerinde kullanılır. Stereo görüntüler arasındaki benzer nesnelerin tespiti ve konumlandırılmasında kullanılır. Başka fark haritası algoritması, stereo görüntülerde benzer nesnelerin tespiti ve eşleştirilmesi için etkili bir tekniktir. Nesne algılama, hareket takibi ve benzeri uygulamalarda kullanılır.

3.2.8 Başka Fark Algoritması Avantaj ve Dezavantajları

Avantajlar: Başka fark haritası algoritması, stereo görüntüler arasında hassas ve doğru eşleştirmeler sağlamaktadır. Benzer nesnelerin doğru bir şekilde tespit edilmesini ve konumlandırılmasını sağlamaktadır. Öznitelik çıkarımı ve eşleştirme işlemlerini hızlı bir şekilde gerçekleştirir. Gerçek zamanlı uygulamalarda kullanım için uygun hale getirir. Binarize edilmiş başka fark haritası, özniteliklerin sayısını azaltır ve işlenecek veri miktarını önemli ölçüde azaltır. Algoritmanın daha etkili ve verimli çalışmasını sağlar. Stereo görüntülerdeki nesne algılama ve eşleştirme yanı sıra diğer benzer uygulamalarda da kullanılır. Derin öğrenme tabanlı sistemlerle birlikte kullanıldığında etkilidir.

Dezavantajlar: Başka fark haritası algoritması, stereo görüntüler arasındaki farkları hassas bir şekilde tespit etmek için bir eşik değeri kullanır. Ortam koşullarına ve görüntü kalitesine bağlı olarak değişir ve algoritmanın performansını etkiler. Başka fark haritası algoritması, öznitelik çıkarımı ve eşleştirme işlemlerinde hata duyarlıdır. Düşük kaliteli veya gürültülü görüntülerde, yanlış eşleştirmelere neden olur. Optimal bir eşik değeri seçimi önemlidir. Algoritmanın performansını doğrudan etkiler ve belirlenmesi zordur.

3.2.9 Özellik Algılama Algoritması

Özellik algılama algoritmaları, bir görüntüde veya bir veri kümesinde belirli özellikleri tanımlamak ve çıkarılmasını sağlayan tekniklerdir. Görüntülerdeki kenarlar, köşeler, noktalar, desenler veya daha karmaşık yapılar gibi çeşitli görüntü veya veri örüntülerini içerir. Özellik algılama, görüntü işleme, makine öğrenimi ve

bilgisayar görüşü gibi alanlarda yaygın olarak kullanılmaktadır ve birçok uygulama için temel bir bileşen oluşturur.

Harris Köşe Algılama: Görüntülerdeki köşeleri tanımlamak için kullanılan bir tekniktir. Harris köşe algılama algoritması, görüntünün farklı yönlerinde ve ölçeklerinde kenarlılık fonksiyonlarını hesaplayarak köşeleri belirler.

Shi-Tomasi Köşe Algılama: Harris köşe algılama algoritmasının iyileştirilmiş bir versiyonudur. Shi-Tomasi algoritması, Harris algoritmasından daha hassas ve daha kararlıdır.

FAST (Features from Accelerated Segment Test): Hızlı ve etkili bir şekilde köşe noktalarını algılamak için kullanılan bir yöntemdir. FAST algoritması, köşe adayları için bir segment testi yaparak görüntülerdeki köşeleri bulur.

SIFT (Scale-Invariant Feature Transform): Ölçek, dönüş ve aydınlatma değişikliklerine karşı dayanıklı özelliklerin tespiti için kullanılan bir algoritmadır. SIFT, görüntülerdeki benzersiz nokta özelliklerini tanımlar ve eşleştirir.

SURF (Speeded-Up Robust Features): SIFT'in hızlı bir versiyonudur. SURF, görüntülerdeki ölçek ve dönüş değişikliklerine karşı dayanıklı özellikleri bulmak için kullanılır.

ORB (Oriented FAST and Rotated BRIEF): Köşe noktaları ve özniteliklerin algılanması ve eşleştirilmesi için hızlı bir yöntemdir. ORB, özellikle gerçek zamanlı uygulamalarda kullanılmak üzere tasarlanmıştır.

3.2.10 Özellik Algılama Algoritması Avantaj ve Dezavantajları

Avantajlar: İyi bir özellik algılama algoritması, görüntülerdeki veya veri kümesindeki özellikleri yüksek hassasiyetle tanımlar. Nesnelerin veya yapıların doğru bir şekilde algılanmasını sağlar. Ölçek ve dönüş değişikliklerine karşı dayanıklıdır. Özelliklerin farklı boyutlarda ve farklı yönlere dönüştürülmüş görüntülerde bile algılanmasını sağlar. Hızlı ve etkili işleme sağlar. Gerçek zamanlı uygulamalarda kullanım için uygun hale getirir. Nesne tanıma, nesne tespiti, hareket takibi, görüntü

eşleştirme ve diğer birçok uygulama için kullanılır. Geniş bir uygulama yelpazesi sunar.

Dezavantajlar: Görüntülerdeki veya veri kümesindeki hata ve gürültüye karşı duyarlıdır Yanlış pozitif veya yanlış negatif algılamalara neden olur. Bazı özellik algılama algoritmaları, yoğun hesaplama gücü gerektirmektedir. Büyük veri kümesi veya yüksek çözünürlüklü görüntüler üzerinde çalışırken, işlemci gücü ve bellek kullanımı artar. Doğru çalışması için uygun parametrelerin seçilmesi önemlidir. Deneme yanılma veya deneyim gerektirir. Büyük boyutlu veri kümeleriyle çalışırken, özellik algılama algoritmalarının işlem gücü ve bellek kullanımı artar. Büyük ölçekli uygulamalarda sorunlara neden olur.

3.2.11 Kör Ters Evrişim Algoritması

Kör ters evrişim (Blind Deconvolution) algoritması, bir görüntüdeki bulanıklığı (blurring) kaldırmak için kullanılan bir görüntü işleme tekniğidir. Bulanık bir görüntü ve bir tahmin edici fonksiyon (PSF- Point Spread Function) kullanarak, orijinal net görüntüyü geri kazanmayı amaçlar. Bulanıklık, bir görüntünün netliğini azaltan ve detayları bulanıklaştıran bir etkidir. Optik sistemlerde, kamera sensörlerinde veya hareketli nesnelerin neden olduğu hareket bulanıklığı gibi çeşitli faktörlerden kaynaklanmaktadır. Kör ters evrişim algoritmaları, bu bulanıklığı tersine çevirerek orijinal net görüntüyü tahmin etmeye çalışır. Bulanıklığın nasıl oluştuğunu veya PSF'nin doğru formunu tam olarak bilmedikleri için "kör" olarak adlandırılırlar. Bu nedenle, bu algoritmalar, bulanık bir görüntü ve bir tahmin edilen PSF arasında bir model varsayımı kullanarak orijinal net görüntüyü tahmin etmeye çalışırlar. Çözüm alanında bir optimizasyon işlemi yoluyla gerçekleştirir.

3.2.12 Kör Ters Evrişim Algoritması Avantaj ve Dezavantajları

Avantajlar: Kör ters evrişim algoritmaları, bir görüntüdeki bulanıklığı kaldırmak için kullanılır. Görüntülerin netliğini artırır ve detayların daha iyi görünmesini sağlar. Kör ters evrişim algoritmaları, bulanıklığın nasıl oluştuğunu tam olarak bilmedikleri için model bağımsızdır. Çeşitli uygulamalarda geniş bir kullanım alanı sağlar. Gerçek

zamanlı veya hızlı bir şekilde çalışır. Uygulamaların hızlı tepki vermesini sağlar. Farklı türdeki bulanıklıkları algılar ve kaldırır. Görüntülerdeki çeşitli bulanıklık türlerini ele alabilme yeteneği sağlar.

Dezavantajlar: Kör ters evrişim algoritmalarının en büyük dezavantajı, doğru PSF'nin (Point Spread Function) tahmin edilmesinin zorluğudur. PSF'nin doğru formunun bilinmemesi veya yanlış tahmin edilmesi, algoritmanın doğruluğunu ve kararlılığını etkiler. Kör ters evrişim algoritmaları genellikle hesaplama gücü gerektirir ve zaman alıcıdır. Büyük boyutlu görüntüler veya karmaşık bulanıklık modelleriyle çalışırken, işlem gücü ve bellek kullanımı artar.

Bu tez kapsamında kullanılan görüntü işleme tekniği ilgi alanı (ROI- Region of Interest) için histogramları hesaplar. Algoritma, Rect fonksiyonunun parametrelerini (örneğin, ilgi alanlarının konumu ve boyutu) ayarlayarak farklı ilgi alanları için histogramları hesaplamak üzere kolayca uyarlanmaktadır. Algoritma, çerçevenin genişliği üzerinde yineleme yaparak ve sabit genişlik ve yükseklikteki ilgi alanlarını çıkararak histogramları verimli bir şekilde hesaplamaktadır. Bu yaklaşım, tüm görüntü için histogram hesaplamaya kıyasla hesaplama ek yükünü azaltır. Histogram değerlerini vektörlerde saklayarak, tüm görüntü için piksel değerlerini saklama ihtiyacını ortadan kaldırır. Bu, özellikle büyük video kareleri için daha az bellek kullanılmasını sağlamaktadır. Uygulama ve donanım özelliklerine bağlı olarak, algoritma video akışlarının gerçek zamanlı işlenmesi için uygun ortam sağlamaktadır. Belirli ilgi alanları için histogramların verimli bir şekilde hesaplanması, daha yüksek işleme hızlarına katkıda bulunur. Bu da gerçek zamanlı uygulamalarda büyük avantaj sağlamaktadır. Ayrıca Canny kenar tespiti ve Hough dönüşümü algoritmaları da kullanılmıştır.

4. OTONOM ARACIN GERÇEK HAYATA UYARLANMASI

4.1 Otonom Aracın Parçaları

Bu bölümde gerçek hayatta uygulaması yapılacak olan aracın parçaları tanıtılmıştır. Görüntü işleme teknikleri kullanılarak test edilecek olan otonom aracın kullanılan parçaları Tablo 4.1'de gösterilmiştir.

Tablo 4.1: Otonom Araç Parçaları

Parça	**Adet**
Toplu Tekerlek	1
Tekerlek	2
Motor Sürücü	1
DC Motor	2
Bilgisayar	1
Lİ-PO	1
TAŞINABILIR GÜÇ KAYNAĞI	1
Mikrokontrolcü	1

4.1.1 Şasi:

Şekil 4.1: Aracın Şasisi

Tasarım ve Modelleme:

Şasi için kullanılan tasarım hazır model kullanılarak elde edilmiştir. 3D baskı malzemesi seçerken dayanıklılık, hafiflik ve maliyet gibi faktörleri göz önünde bulundurulmuştur. PLA ve ABS gibi plastik malzemeler genellikle popüler seçeneklerdir. Şasi için PLA malzemeden yapılmış olan 2 parça gövde kullanılmıştır. Bir zemine tüm parçalar sığmadığı için 2 parça gövde tercih edilmiştir. Bu parçalar 3 boyutlu yazıcıdan çıkartılmıştır. Parçalar 3 boyutlu yazıcıdan çıkarıldıktan sonra montaj için gerekli vida delikleri açılmıştır. Şasi seçiminde, taşıma kapasitesi, sensör yerleşimi ve diğer bileşenlerin montajı gibi faktörler dikkate alınmıştır. Şasi tasarımında ağırlık dağılımı dengeli şekilde yapılmıştır. Bu, aracın dengeli ve istikrarlı bir şekilde hareket etmesine yardımcı olmaktadır. Şekil 4.1'de bu projede kullanılan şasi gösterilmiştir.

4.1.2 Toplu tekerlek:

Şekil 4.2: Araçta kullanılan toplu tekerlek

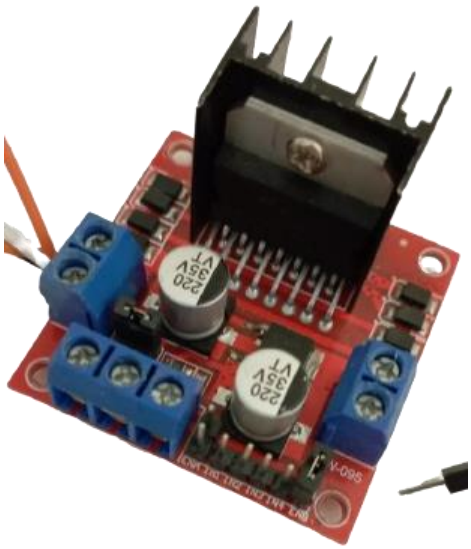
Toplu tekerlek, genellikle robotlar ve diğer hareketli cihazlarda kullanılan bir hareketli sistem elemanıdır. Bir toplu rulmanın üzerine monte edilmiş bir top şeklindeki tekerleği içerir. Top, bir destek yapıya bağlıdır ve genellikle 360 derece dönebilir. Toplu tekerleğin temel işlevi, bir cihazın hareketini desteklemek ve yönlendirmektir. Bu tür tekerlekler, bir yüzey üzerinde kolayca dönebilirler, bu da cihazın çeşitli yönlere hareket etmesini sağlar. Özellikle robotların altına monte edilen toplu tekerlekler, düzgün bir şekilde hareket etmelerini ve çeşitli yönlere yönlendirilebilmelerini sağlamaktadır. Toplu tekerlekler genellikle hafif ve düz yüzeylerde kullanılırlar. Ayrıca, düşük sürtünme ve 360 derece dönebilme yetenekleri, cihazların daha esnek ve manevra kabiliyetine sahip olmasını sağlar. Bu tür tekerlekler, robotik projeler, oyuncaklar ve diğer hareketli sistemlerde yaygın olarak kullanılmaktadır. Bu tez kapsamında yapılan araçta ise ön kısımda toplu tekerlek kullanımı tercih edilmiştir. Şekil 4.2'de bu projede kullanılan toplu tekerlek gösterilmiştir.

4.1.3 Taşıyıcı tekerler:

Şekil 4.3: Araçta kullanılan taşıyıcı tekerler

"Robot tekerleri," bir robotun hareketini sağlamak için kullanılan tekerlek türlerini ifade eder. Robotlar farklı tipte tekerleklerle donatılabilir ve tekerlek seçimi genellikle robotun tasarım, hareket kabiliyeti ve kullanım amacına bağlı olarak yapılır. Tekerlek ölçüleri 66 x 26 milimetredir. Tekerlek Ağırlığı 38 gramdır. Aracın zemine iyi tutunabilmesi için dişli tekerlekler tercih edilmiştir. Tekerler DC Motorlara montajlanabilecek şekilde tasarlanmışlardır. Şekil 4.3'te bu projede kullanılan tekerler gösterilmiştir.

4.1.4 Motor Sürücü: L298N



Şekil 4.4: Araçta kullanılan motor sürücü

H-köprüsü, bir yüke uygulanan gerilimin polaritesini değiştiren elektronik bir devredir. Bu devreler genellikle robotikte ve diğer uygulamalarda DC motorların ileri veya geri çalışmasını sağlamak için kullanılır. Adı, bir "H" harfinin dalları olarak yapılandırılmış dört anahtarlama elemanı ve çapraz çubuk olarak bağlanan yük ile ortak şematik diyagram temsilinden türetilmiştir (Williams, 2002).

L298N, bir motor sürücü entegresidir ve özellikle mikrodenetleyiciler veya diğer kontrol devreleri tarafından kontrol edilen DC motorları veya adım motorları sürmek için kullanılır. L298N, motor sürücüleri arasında oldukça popülerdir ve geniş bir kullanım alanına sahiptir. Bu projede DC Motorları L298N adlı motor sürücü kontrol etmektedir. Şekil 4.4'te bu projede kullanılan motor sürücü gösterilmiştir.

4.1.5 5V DC Motor:

DC motor (Direct Current Motor) (Doğru Akım Motoru), doğru akım (DC) enerjisini mekanik enerjiye çeviren elektromekanik bir cihazdır. DC motorlar, bir manyetik alan içinde dönen bir rotorun, bir sabit manyetik alan veya manyetik alan oluşturan bir statorun etkisi altında dönmesi prensibine dayanır. Bu dönme hareketi, motorun çıkış şaftına bağlı bir yükü döndürme veya bir tekerleği hareket ettirme gibi mekanik işleri gerçekleştirmek için kullanılmaktadır. Bu projede Şekil 4.5'te gösterilen sarı DC Motorlardan iki adet kullanılmıştır.

Şekil 4. 5: Araçta kullanılan motor sürücüler

4.1.6 Bilgisayar:

Şekil 4.6: Araçta kullanılan bilgisayar

Raspberry Pi 4 Tek kartlı bilgisayar (single-board computer), Raspberry Pi Vakfı tarafından geliştirilen ve üretilen bir minyatür bilgisayar kartıdır. Raspberry Pi, düşük maliyetli, düşük güç tüketimli ve geniş bir geliştirici topluluğu tarafından desteklenen bir bilgisayar platformu olarak popülerdir. Raspberry Pi 4 Tek kartlı bilgisayar (single-board computer), önceki modellerden daha güçlü bir donanım setine sahiptir ve geniş bir kullanım alanına hitap eder. Şekil 4.6'da bu projede kullanılan bilgisayar gösterilmiştir.

İşlemci: Broadcom BCM2711 dört çekirdekli ARM Cortex-A72 işlemci (64-bit), 1.5 GHz hızında.

Bellek (RAM): 2GB, 4GB veya 8GB LPDDR4 SDRAM seçenekleri bulunmaktadır.

Grafik İşlem Birimi (GPU): Broadcom VideoCore VI grafik işlem birimi, çift ekran desteği.

Depolama: microSD kart yuvası üzerinden genişletilebilir depolama.

Bağlantılar:

2 adet HDMI portu ile çift ekran desteği.

Gigabit Ethernet portu.

2 adet USB 3.0 portu ve 2 adet USB 2.0 portu.

Kablosuz bağlantı: Wi-Fi 802.11ac ve Bluetooth 5.0.

Giriş/Çıkış (I/O) Portları:

GPIO (Genel Amaçlı Giriş/Çıkış) pinleri.

Kamera ve ekran bağlantı noktaları.

3.5mm ses Jak.

Güç:

5V, 3A mikro USB güç girişi.

Soğutma:

Soğutma için fan ve ısı emici ile uyumlu tasarım.

İşletim Sistemi Desteği:

Raspberry Pi'ye özel olarak tasarlanmış olan Raspbian tabanlı Raspberry Pi OS ve diğer çeşitli Linux dağıtımları.

4.1.7 Mikrodenetleyici:

Şekil 4.7: Araçta kullanılan mikrodenetleyici

Arduino Uno Mikrodenetleyici, popüler bir açık kaynaklı donanım ve yazılım platformu olan Arduino'nun bir mikrodenetleyici kartıdır. Arduino platformu, özellikle hobi elektroniği ve gömülü sistem projeleri için tasarlanmış, kullanımı kolay, programlaması basit mikrodenetleyici kartları sağlar. Arduino Uno Mikrodenetleyici modeli, Arduino'nun en temel ve yaygın kullanılan kartlarından biridir. Şekil 4.7'de bu projede kullanılan mikrodenetleyici gösterilmiştir.

Mikrodenetleyici: Atmel ATmega328P mikrodenetleyici içerir.

Bellek: 32 KB Flash bellek (program belleği). 2 KB SRAM (geçici bellek). 1 KB EEPROM (kalıcı bellek).

Programlama: Arduino Uno Mikrodenetleyici, Arduino IDE (Entegre Geliştirme Ortamı) üzerinden programlanır. Arduino IDE, C ve C++ benzeri bir dil kullanır ve kullanıcıların kolayca kod yazmalarını sağlar. Bu yüzden tercih edilmiştir.

4.1.8 Güç Kaynağı: Taşınabılır Güç Kaynağı

Şekil 4.8: Araçta kullanılan batarya

Raspberry Pi 4 Tek kartlı bilgisayar (single-board computer), 5V ve en az 3A güç gerektirir. Raspberry Pi 4 Tek kartlı bilgisayar (single-board computer) için en iyi sonuçları almak için taşınabilir güç kaynağı, en az 5V ve 3A çıkışa sahip bir USB portu içermelidir. Ayrıca, taşınabilir güç kaynağı içinde bir voltaj düzenleyici (voltaj regülatör) olması da önemlidir. Çünkü Raspberry Pi'nin istikrarlı bir şekilde çalışabilmesi için belirli bir voltajda olması gerekmektedir. Voltaj düzenleyici olmayan taşınabilir güç kaynağında dalgalanmalar olabilir ve bu da Raspberry Pi'nin stabilitesini etkilemektedir.

Bu projede PHILIPS DLP6733 marka bir taşınabilir güç kaynağı kullanılmıştır. 10.000 mAh pil kapasitesi vardır. Pil türü lityum polimerdir. Raspberry Pi 4 Tek kartlı bilgisayara (single-board computer) istediği 5V 3A çıkışı verebilmektedir. Bunun için bir adet micro-usb ve bir adet type-c çıkışı vardır. Şekil 4.8’de bu projede kullanılan batarya gösterilmiştir.

4.1.9 Li-Po (lityum – polimer) Batarya:

Şekil 4.9: Araçta kullanılan güç kaynağı

Profuse markasına ait bu pil birçok uygulama için kullanılan bir şarj edilebilir lityum polimer pil (Li-Po) türüdür. Bu pilin özellikleri şu şekildedir:

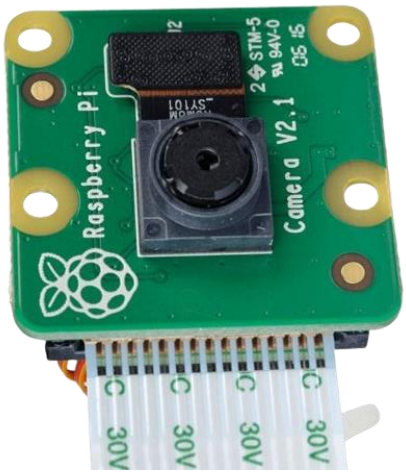
Voltaj: Bu pil, nominal olarak 7.4 volt gerilime sahiptir. Bu, çeşitli elektronik cihazlarda ve hobi uygulamalarında kullanılan yaygın bir voltajdır.

Deşarj Kapasitesi (35C): "35C" deşarj kapasitesi, pilin anlık deşarj yeteneğini ifade eder. Bu durumda, pilin anlık deşarj kapasitesi, pilin kapasitesinin 35 katıdır. Yüksek C değerleri, pilin yüksek akımlarla hızlı deşarj edilebileceği anlamına gelir.

Kapasite: 2200mAh kapasitesi, pilin bir seferde sağlayabileceği enerji miktarını temsil eder. Bu, pilin ne kadar süre boyunca bir cihazı çalıştırabileceğini ve güç sağlayabileceğini gösterir.

Şekil 4.9'da bu projede kullanılan güç kaynağı gösterilmiştir.

4.1.10 Kamera:



Şekil 4.10: Araçta kullanılan kamera

Raspberry Pi Kamera, Raspberry Pi mikro bilgisayar platformu ile entegre edilebilen bir kamera modülüdür. Bu kamera, geniş bir kullanım yelpazesi için tasarlanmış ve Raspberry Pi'nin özelliklerini genişletmek, görüntüleme ve video kaydı yapmak gibi çeşitli projelerde kullanılmak üzere geliştirilmiştir. Raspberry Pi Kamera modülü, kullanıcılara düşük maliyetli ve kompakt bir kamera çözümü sunar. Şekil 4.10'da bu projede kullanılan kamera gösterilmiştir.

Çözünürlük: Raspberry Pi Kamera modülü, farklı modellerde farklı çözünürlük seçeneklerine sahiptir. Örneğin, Raspberry Pi Kamera Modülü V2, 8 megapiksel çözünürlüğe sahiptir.

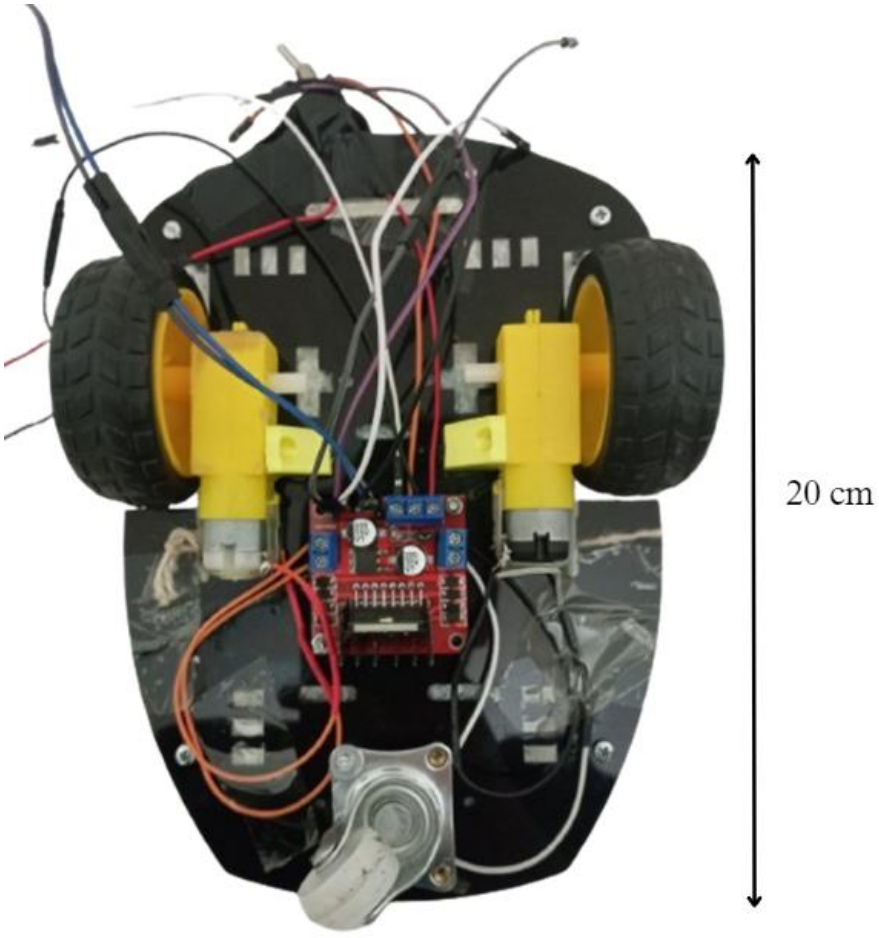
Video Çekimi: Video çekim yeteneği sayesinde, Raspberry Pi Kamera modülü, yüksek kaliteli video kayıtları yapabilir. Çeşitli çözünürlük ve kare hızı seçenekleri sunar.

Bağlantı: Raspberry Pi Kamera modülü, Raspberry Pi kartına doğrudan bir CSI (Kamera Serial Interface) bağlantısı üzerinden bağlanır. Bu sayede modül, Raspberry Pi ile bütünleşmiş bir şekilde çalışabilir.

4.2 Otonom Aracın Montajı

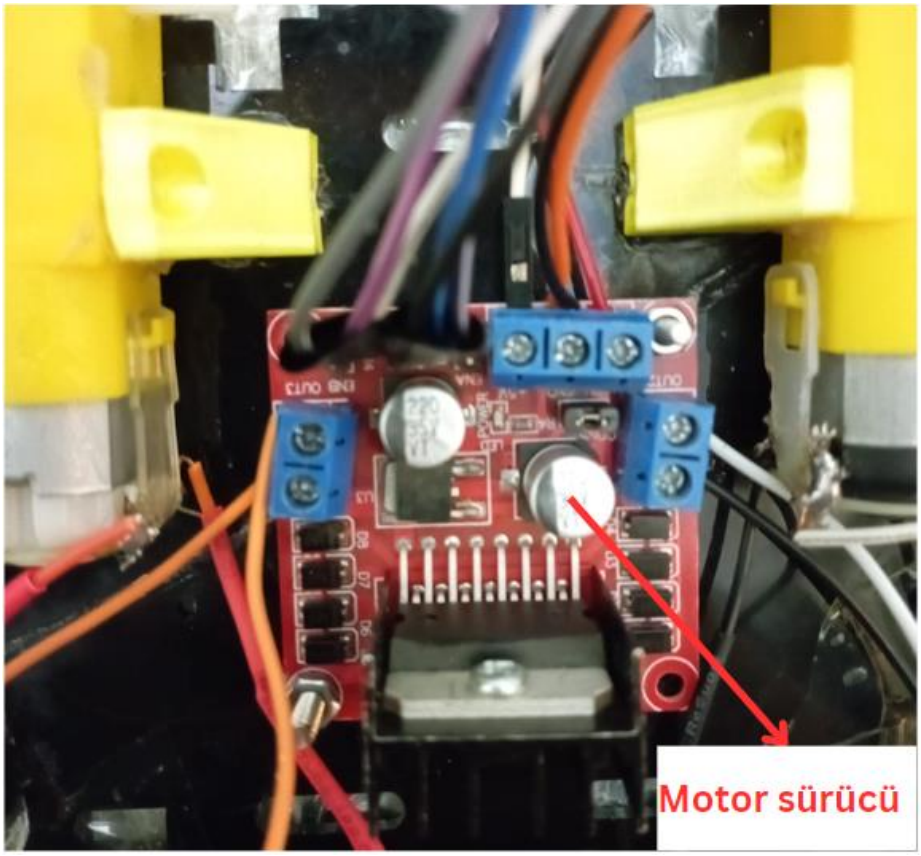
4.2.1 Alt Kısım

Şasinin alt kısmına DC Motorlar yerleştirilmiştir. DC motorlar genellikle polarize edilmiştir, bu nedenle motorun pozitif ve negatif uçlarını doğru şekilde bağlamak önemlidir. Yanlış polarite bağlantıları, motorun yanlış yönde dönmesine veya hasar görmesine neden olabilir. Bu nedenle DC Motorlar şasiye monte edilmeden önce pozitif ve negatif uçlarının kabloları aynı yönde bağlanıp lehimlenmiş ardından montajı yapılmıştır. Ön kısımda bulunan toplu tekerlek aracın dengesini sağlamasını ve dönüşleri daha kolay yapmasını sağlamaktadır. Toplu teker aracın ön kısmına monte edilmiştir. İlk kat için kablo düzenlemeleri yapılmıştır. Şekil 4.11'de aracın ilk katının montajı gösterilmiştir.



Şekil 4.11: Aracın ilk katının montajı

DC Motorların L298n Motor sürücüye bağlantıları yapılmıştır. Motor sürücüden Arduino'ya iki motorun da girdi ve çıktı pinleri ile PWM (Pulse Width Modulation) (Darbe Genişliği Modülasyonu) pinlerinin bağlantıları da yapılmıştır. Motor sürücüsünün 12V giriş kısmına Li-Po (Lityum-Polimer) pilin pozitif ucu, GND (Ground) (Toprak) kısmına Li-Po (Lityum-Polimer) pilin negatif ucu bağlanmıştır. GND (Ground) (Toprak) kısmına aynı zamanda Arduino Uno Mikrodenetleyicinin topraklaması yapılmıştır. Motor sürücüden alınan harici 5V çıkışı ile Arduino Uno Mikrodenetleyicinin 5V girişi beslenerek Arduino Uno Mikrodenetleyicinin çalışması sağlanmıştır. Her bir DC Motor için 3 adet bağlantı vardır. Bu bağlantılar pozitif, negatif ve PWM sinyallerini içerir. Bu sinyaller Arduino Uno Mikrodenetleyiciye bağlanarak DC Motorların ileri ve geri dönüş yapmasını sağlayan bağlantılardır. PWM ise DC Motorların 0 ile 255 değerleri arasında 256 farklı değerle çalışmasını sağlar. Bu da motorları istenilen hızlara ayarlamak için kullanılmaktadır. Şekil 4.12'de aracın montajından bir kare gösterilmiştir.



Şekil 4.12: Montajın detaylı görüntüsü

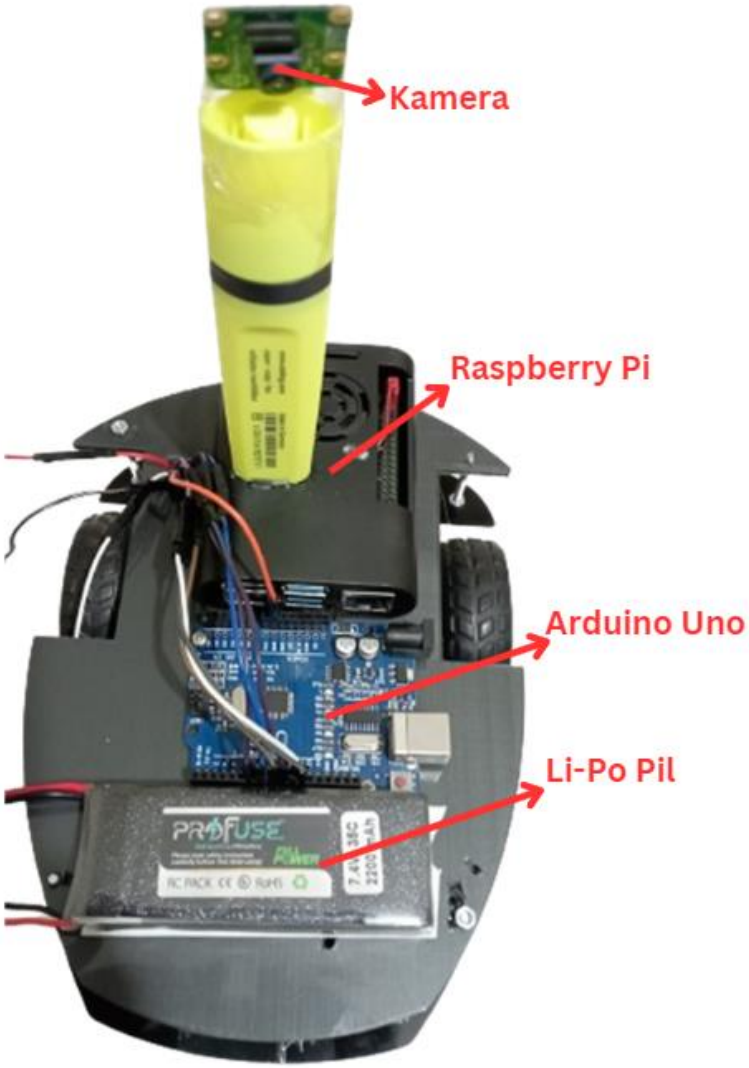
4.2.2 Birinci Kat

Birinci katta büyük yer kaplayan taşınabilir güç kaynağına yer verilmiştir. Taşınabilir güç kaynağı, Raspberry Pi 4 Tek kartlı bilgisayarı çalıştırmak için gerekli gücü sağlayacaktır. Raspberry Pi 4 Tek kartlı bilgisayar, 5V, 3A güç istemektedir. Bu gücü sağlamak için Li-Po (Lityum-Polimer) pil harici taşınabilir güç kaynağı kullanma ihtiyacı doğmuştur bu yüzden ilk kata taşınabilir güç kaynağı montajı yapılmıştır. Taşınabilir güç kaynağının şasi ile arasına balonlu naylon malzeme çekilmiştir. Bu olası çarpmaların etkisini azaltmak ve taşınabilir güç kaynağının zarar görmesini engellemek için tercih edilmiştir. Şekil 4.13'te ilk kat gösterilmiştir.

Şekil 4.13: Montajı yapılan otonom aracın ilk katı

4.2.3 İkinci Kat

İkinci kata geri kalan Raspberry Pi 4 Tek kartlı bilgisayar, Arduino Uno Mikrodenetleyici, Li-Po (Lityum-Polimer) pil ve Raspberry Pi Kamera yerleştirilmiştir. Kameranın önündeki yolu daha iyi görebilmesi için yerden 20cm yüksekliğe yerleştirilmiştir. Kamera şasinin ortasına gelecek şekilde ayarlanmıştır. Bu sayede otonom sürüş sırasında aracın yolu ortalaması sağlanacaktır. Denetleyici ve bilgisayara gelecek olan kablolar bir delikten geçirilerek tekerlere teması engellenmiştir. Şekil 4.14'te ikinci kat gösterilmiştir.



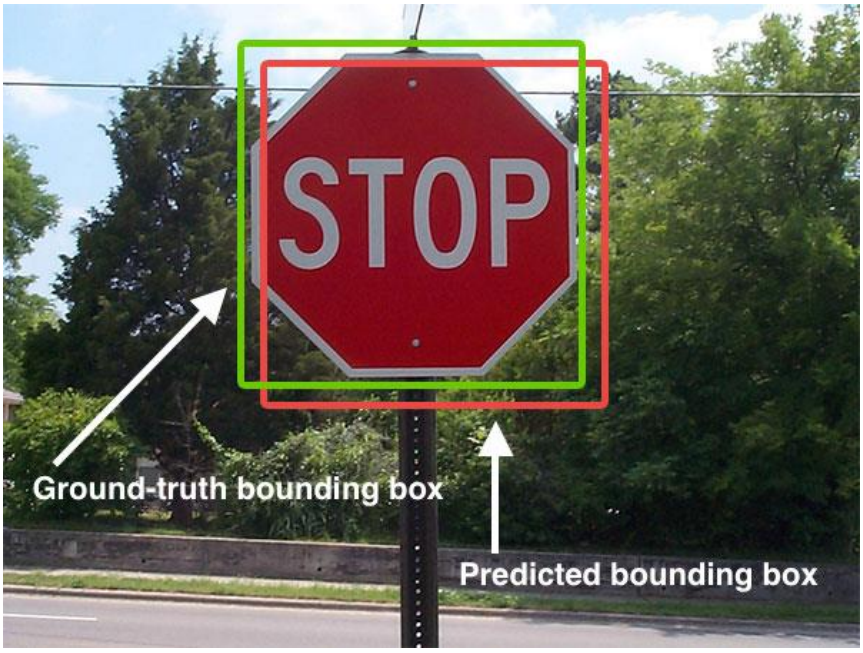
Şekil 4.14: Aracın bitmiş hali

5. YAZILIMLAR

5.1 Bilgisayar Görüsünün Kısa Tarihi

Bilgisayarla görme görevleri, dijital görüntülerin elde edilmesi, işlenmesi, analiz edilmesi ve anlaşılmasına yönelik yöntemleri ve örneğin kararlar şeklinde sayısal veya sembolik bilgiler üretmek için gerçek dünyadan yüksek boyutlu verilerin çıkarılmasını içerir. (Klette, 2014) (Linda G. Shapiro; George C. Stockman, 2001) (Morris, 2004) (Bernd Jähne; Horst Haußecker, 2000)

1960'ların sonlarında bilgisayarla görme, yapay zekâya öncülük eden üniversitelerde başladı. Robotlara akıllı davranışlar kazandırmak için bir basamak olarak insan görsel sistemini taklit etmeyi amaçlıyordu. (Szeliski, 2010). 1966 yılında, bunun bir lisans yaz projesiyle, (Sejnowski, Terrence J., 2018) bir bilgisayara bir kamera bağlayarak ve bilgisayara "gördüklerini tarif ettirerek" (Papert, 1966), (Boden, 2006) başarılabileceğine inanılıyordu. Şekil 5.1'de dur levhası gösterilmiştir.



Şekil 5.1: Dur levhası (Rosebrock, 2016)

5.2 OpenCV (Open Source Computer Vision) (Açık Kaynak Bilgisayar Görüntüsü Kütüphanesi)

OpenCV, özellikle gerçek zamanlı bilgisayarla görme için bir programlama fonksiyonları kütüphanesidir (Pulli ve diğ. 2012). Resmi olarak 1999 yılında başlatılan OpenCV projesi, başlangıçta CPU yoğun uygulamaları geliştirmek için bir Intel Research girişimiydi ve gerçek zamanlı ışın izleme ve 3D ekran duvarlarını içeren bir dizi projenin parçasıydı (Kaehler ve Kaehler, 2016).

Görüntü işleme konusunda piyasa çokça kütüphane bulunmaktadır. Ancak bu proje için kullanılacak olan C++ programlama dili ile optimize şekilde çalıştığı için OpenCV kütüphanesi tercih edilmiştir. Ayrıca Raspberry Pi Kamerası için de kütüphanesinde gerekli kodları bulundurmaktadır. OpenCV, bilgisayar görüntüsü ve makine görüşü uygulamaları geliştirmek için kullanılan ücretsiz ve açık kaynaklı bir kütüphanedir. OpenCV gerçek zamanlı görüntü işleme, nesne tespiti, yüz tanıma, video analizi, kamera kalibrasyonu, üç boyutlu modelleme ve bir dizi diğer bilgisayar görüntüsü uygulaması için geniş bir araç seti sağlar. Bu kütüphane projede kullanılacak olan görüntü işleme algoritmaları için yeterli araç setini sağlamaktadır.

5.3 C++ Programlama Dili

1979 yılında Danimarkalı bir bilgisayar bilimcisi olan Bjarne Stroustrup, C++ Programlama Dilinin öncülü olan "C with Classes" üzerinde çalışmaya başladı (Stroustrup, Bjarne Stroustrup's FAQ: When was C++ invented?, 2010). Yeni bir dil yaratma motivasyonu Stroustrup'un doktora tezi için programlama deneyiminden kaynaklanmıştır. Stroustrup, Simula'nın büyük yazılım geliştirme için çok yararlı özelliklere sahip olduğunu, ancak dilin pratik kullanım için çok yavaş olduğunu, BCPL'nin ise hızlı ancak büyük yazılım geliştirme için uygun olamayacak kadar düşük seviyeli olduğunu fark etti. Stroustrup AT&T Bell Labs'de çalışmaya başladığında UNIX çekirdeğini dağıtık hesaplama açısından analiz etme sorunuyla karşılaştı. Doktora deneyimini hatırlayan Stroustrup, C dilini Simula benzeri özelliklerle geliştirmeye koyuldu (Stroustrup, 1991).

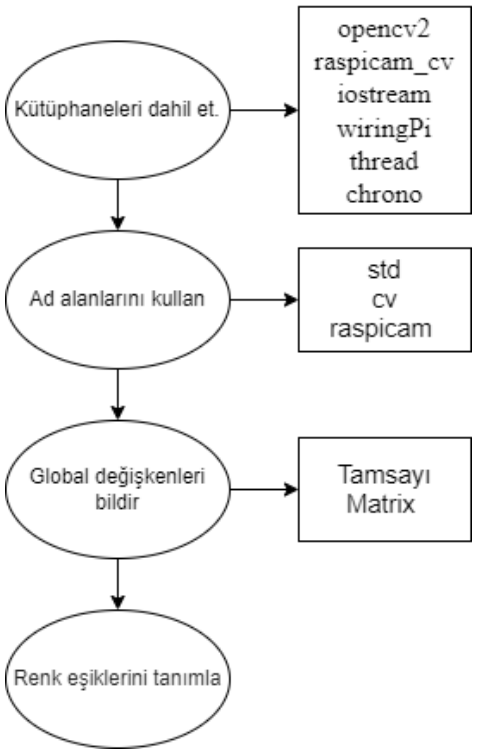
C++, düşük seviyeli işlemlere doğrudan erişim sağlar ve etkili bir şekilde çalışan derleyici optimizasyonlarına izin verir. Bu, yüksek performanslı uygulamalar geliştirmek için ideal bir dil yapısına sahip olduğu anlamına gelir. C++, nesne yönelimli programlama (OOP) ve prosedürel programlamayı destekler. Bu, geliştiricilere ihtiyaca göre kodlama yaklaşımını seçme esnekliği sağlar. C++, birçok endüstri ve uygulama alanında kullanılan bir standart haline gelmiştir. Özellikle sistem programlama, oyun geliştirme, gömülü sistemler ve performans kritik uygulamalar için tercih edilen bir dildir.

5.4 Arduino Uno Mikrodenetleyici

Arduino Uno Mikrodenetleyici gibi mikrodenetleyici kartları, DC motorları sürmek için yaygın olarak kullanılır, çünkü Arduino platformu, kullanıcıların basit ve etkili bir şekilde dijital ve analog cihazları kontrol etmelerini sağlayan bir ortam sunar. Arduino Uno Mikrodenetleyici veya benzeri bir kart, DC motorları doğrudan kontrol etme yeteneğine sahip değildir. Ancak, bir mikrodenetleyici kartı, motor sürücü devresini kontrol etmek için gerekli olan sinyalleri üretebilir. L298N, bir DC motor sürücü entegresidir ve DC motorların hızını ve yönlendirmesini kontrol etmek için gerekli sinyalleri sağlar. Ancak, bir mikrodenetleyici (örneğin Arduino Uno Mikrodenetleyici) kullanmak, kullanıcıya motorların kontrolünü programlamak ve koşulları belirlemek için daha fazla esneklik sağlar. Arduino, kullanıcıların sensörleri, düğmeleri, uzaktan kumandaları veya başka bir kontrol cihazını entegre etmelerine de olanak tanır. Bu nedenle, DC motorları sürmek için L298N Motor Sürücü kullanırken bir Arduino Uno Mikrodenetleyici kullanmak, motorları programlanabilir ve esnek bir şekilde kontrol etmeyi sağlar.

6. PROGRAMLAMA

Bu bölümde programla sözde kod ve akış şemaları ile açıklanmıştır. Sözde kod, bilgisayar bilimleri alanında algoritmalar ve programlar oluşturulurken ve aktarılırken kullanılan, günlük konuşma diline benzer ve belli bir programlama dilinin detaylarından uzak anlatımlar olarak tanımlanmıştır. Akış şeması ise algoritmaları ve işlemleri birbirine oklarla bağlı değişik tiplerdeki kutular içerisinde gösteren yaygın bir şema tipidir. Akış şemaları çeşitli alanlardaki işlem ve uygulamaların yönetilmesi, belgelendirilmesi, tasarlanması ve çözümlenmesinde kullanılmaktadır. Şekil 6.1'de başlangıç kodunun akış şeması gösterilmiştir. Detaylı kodlar Ekler bölümünde gösterilmiştir.



Şekil 6.1: Başlangıç kodları akış şeması

Kütüphaneleri dahil etme: Programlamada kütüphane, belirli bir programlama dilinde yazılmış ve yeniden kullanım için düzenlenmiş önceden derlenmiş rutinler, fonksiyonlar, sınıflar ve veri yapılarından oluşan bir koleksiyondur. Bu durumla ilgili sözde kodlar şu şekildedir:

- opencv2 kütüphanesini dahil et.
- raspicam_cv kütüphanesini dahil et.
- iostream kütüphanesini dahil et.
- wiringPi kütüphanesini dahil et.
- thread kütüphanesini dahil et.
- Chrono kütüphanesini dahil et.

Ad Alanları kullanma: Ad alanları, büyük projelerde ad çakışmalarını önlemek için bir yöntem sağlar. Bu durumla ilgili sözde kodlar şu şekildedir:

- Standart C++ fonksiyonlarını kullanmak için std ön adını kullan.
- opencv2 kütüphanesini kullanmak için cv ön adını kullan.
- raspicam_cv kütüphanesini için raspicam ön adını kullan.

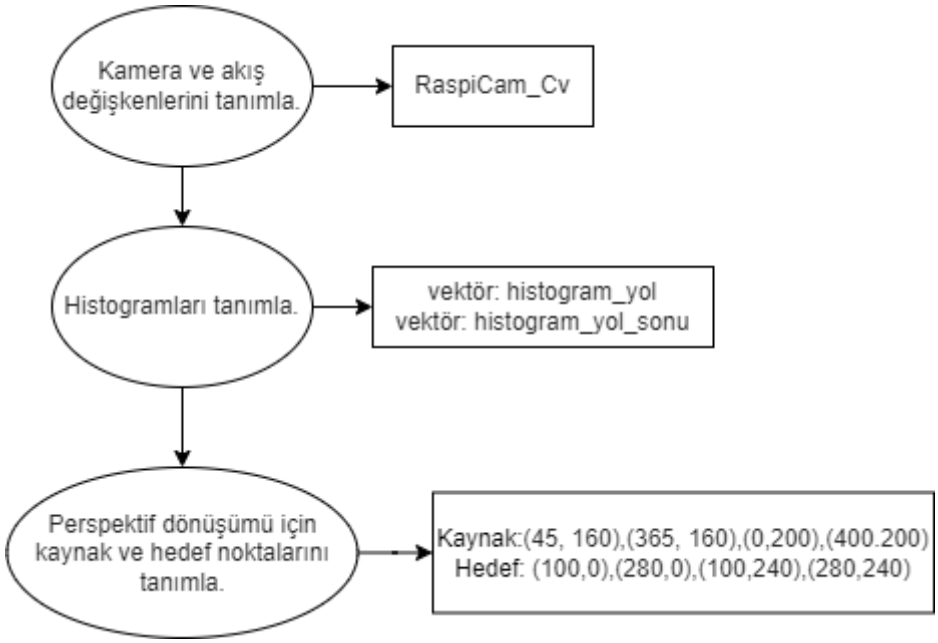
Global değişkenleri bildirme: C++'daki global değişkenler, herhangi bir fonksiyon, sınıf veya blok kapsamının dışında bildirilen değişkenlerdir. Bu, herhangi bir özel nitelendirme olmaksızın fonksiyonlar, sınıflar ve diğer dosyalar dahil olmak üzere programın herhangi bir bölümünden erişilebilir oldukları anlamına gelir. Bu durumla ilgili sözde kodlar şu şekildedir:

Tamsayı olan global değişkenler: sol_serit_pozisyonu, sag_serit_pozisyonu, merkez_goruntu, serit_merkezi, Sonuc, yol_sonu, yaya_mesafe, park_mesafe

Matrix olan global değişkenler: goruntu, goruntu_kopya, hsv_goruntu, hsv_goruntu2, kirmizi_maske, yesil_maske, turuncu_maske, Matrix, Pers_goruntu, esik_goruntu, kenar_goruntu, final_goruntu, final_goruntu_kopya, final_goruntu_kopya2, serit_ilgi_alani, ilgi_alani_serit_sonu, gri_goruntu

Renk eşiklerini tanımlama: Bu değerler yolda şeritlerin algılanması için gereken alt ve üst değerleri göstermektedir. Bu durumla ilgili sözde kodlar şu şekildedir:

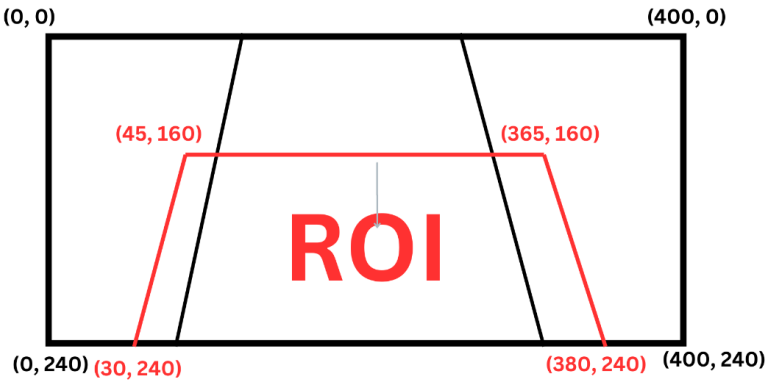
Tamsayı zemin_dusuk_deger = 100, zemin_yuksek_deger = 255 tamsayı minimum_renk = 110, maksimum_renk = 179, minimum_doygunluk = 138, maksimum_doygunluk = 255, minimum_değer = 78, maksimum_değer = 225 Şekil 6.2'de kamera ve histogram akış şeması gösterilmiştir.



Şekil 6.2: Kamera ve histogram akış şeması

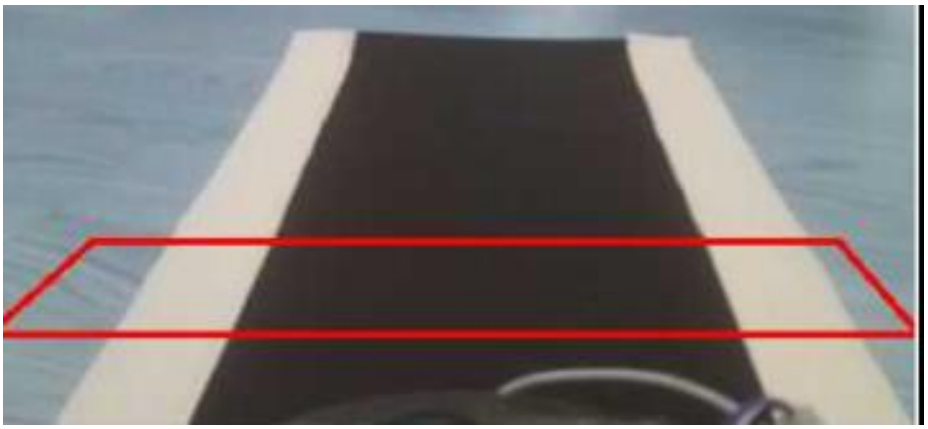
İlgi alanı (ROI – Region of Interest), bir yolun sınırları belirlemek için kullanılan fonksiyondur. Bu noktada kameradan gelen görüntünün tamamını işlemek, bilgisayarın gereksiz işlem yapmasına ve istenmeyen verilerin hesaplamaya karışmasına sebep olmaktadır. Bu sebeple kameradan gelen görüntüyü istenilen noktalardan sınırlandırmak, sadece ilgilenilen alanın verileri ile hesaplama yapılmasını sağlamaktadır. Şekil 6.3'te ilgi alanı çalışma mantığı gösterilmiştir.

Kaynak = (45, 160), (365, 160), (30,240), (380,240)



Şekil 6.3: İlgi alanı çalışma mantığı

Hedef = (100,0), (280,0), (100,240), (280,240)



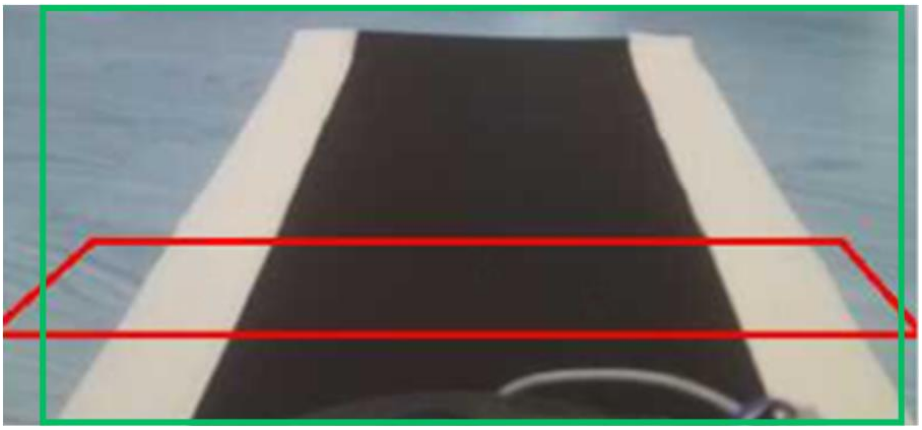
Şekil 6.4: İlgi alanının oluşturulması

Bu tez kapsamında geliştirilen algoritma, gideceğe yöne karar verirken kullandığı veriler kritik bir rol teşkil etmektedir. Bu verilerin optimize edilmesi, aracın vereceği kararın isabetli olup olmamasını doğrudan etkilemektedir. Bu nedenle ilgi alanı seçimi kritik bir öneme sahiptir. İlgi alanı seçiminde yolun çok ilerisinden veri alınması, aracın erkenden sola veya sağa dönmesine sebep olabilir. Birçok yapılan denemenin ve testin ardından en iyi değerin 16 cm olduğu tespit edilmiştir. Bu sebeple ilgi alanı, aracın önündeki 16 cm olan bölümü görebileceği şekilde seçilmiştir. Yukarıda şekil 6.4'te ilgi alanının oluşturulması gösterilmiştir.

Perspektif dönüşümü için fonksiyon: Perspektif Fonksiyonu

Bilgisayar görüsündeki kuş bakışı görünüm, sahnenin veya nesnelerin doğrudan yukarıdan görüldüğü bir perspektifi ifade etmektedir. Gökyüzünden sahneye bakıldığında sahip olunacak manzarayı göstermektedir. Bu perspektif kamera kalibrasyonu, geometrik dönüşümler ve görüntü işleme gibi teknikler kullanılarak elde edilmektedir. Görüntünün kuşbakışı görüntüye dönüştürülmesi için perspektif dönüşümü yapılmıştır. Perspektif dönüşümü, araç etrafındaki ortamı daha iyi anlamak için kameralar tarafından sağlanan bilgileri daha doğru bir şekilde işlemek için kullanılmaktadır. Perspektif dönüşümü, kamera veya sensörün bakış açısı nedeniyle oluşan bozulmayı düzeltmeye yardımcı olmaktadır. Perspektif dönüşümü olmadan, kameraya daha yakın nesneler daha büyük görünürken, daha uzaktaki nesneler daha küçük görünür ve bu da şerit işaretlerinin ve konumlarının algılanmasını bozmaktadır. Genel olarak perspektif dönüşümü, bozulmayı düzelterek ve görüntülerde şerit

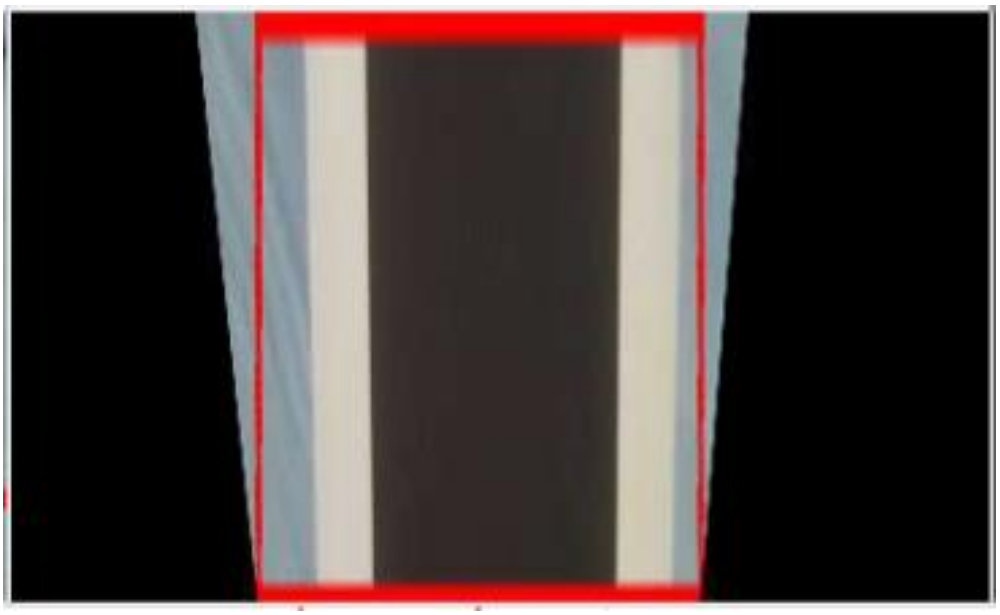
işaretlerinin tutarlı bir temsilini sağlayarak otonom sürüş sistemlerinde şerit takibinin doğruluğunu, sağlamlığını ve güvenliğini artırmada kritik bir rol oynamaktadır. Şekil 6.5'te perspektif alanın oluşturulması gösterilmiştir.



Şekil 6.5: Perspektif alanın oluşturulması

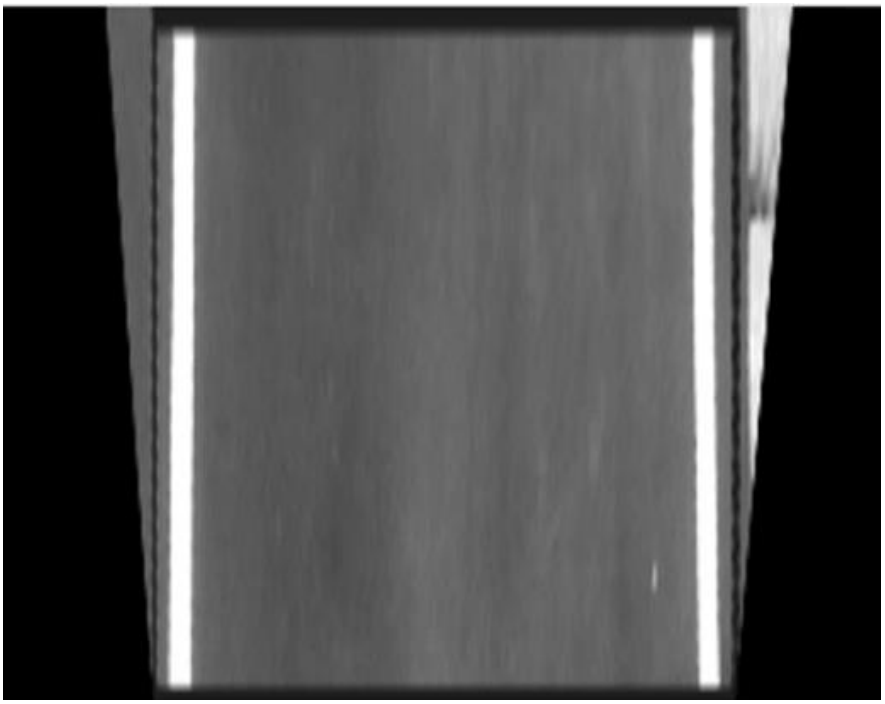
Şekil 6.6'da perspektif dönüşümünün uygulanması gösterilmiştir.

- Matrix = Perspektif Dönüşümü Yap (Kaynak, Hedef)
- Renk uzayını değiştir Perspektif_goruntu gri_goruntu (mavi yeşil kırmızı renk uzayından, kırmızı yeşil mavi renk uzayına dönüştür.)



Şekil 6.6: Perspektif dönüşümünün uygulanması

- 240 ile 255 arasındaki gri değerleri al (gri_goruntu, 240, 255, esik_goruntu)



Şekil 6.7: Görüntünün griye çevrilmesi

Şekil 6.8: Eşik değerlerin uygulanması

Şekil 6.7 ve 6.8'de görüntünün griye çevrilmesi ile ilgili görseller gösterilmiştir.

Burada şerit çizgilerini görüntüdeki diğer bileşenlerden ayırma işlemi yapılmaktadır. Şeritleri takip edebilmek için öncelikle şeritlerin tespit edilmesi gerekmektedir. Bunun için eşikleme fonksiyonu kullanılmıştır. Eşikleme fonksiyonu için daha önce perspektif dönüşümü yapılan görüntü gri formata çevrilmiştir.

Gri formata dönüştürülen görüntüde 256 farklı gri ton değeri vardır. 0 değeri en siyah değeri, 255 değeri ise en beyaz değeri göstermektedir. Şeritlerin tespit edilebilmesi için eşik değeri olarak 240 değeri alınmıştır. 240 değerinin altındaki renkler maskelenmiştir yani görüntüden çıkarılmıştır. 240 ile 255 arasındaki değerler ise görüntüye dahil edilmiştir. Bu sayede şekil 6.9'da görülen görüntü elde edilmiştir.

- Canny dönüşümü uygula(gri_goruntu,kenar_goruntu)

Şekil 6.9: Canny dönüşümü uygulanması

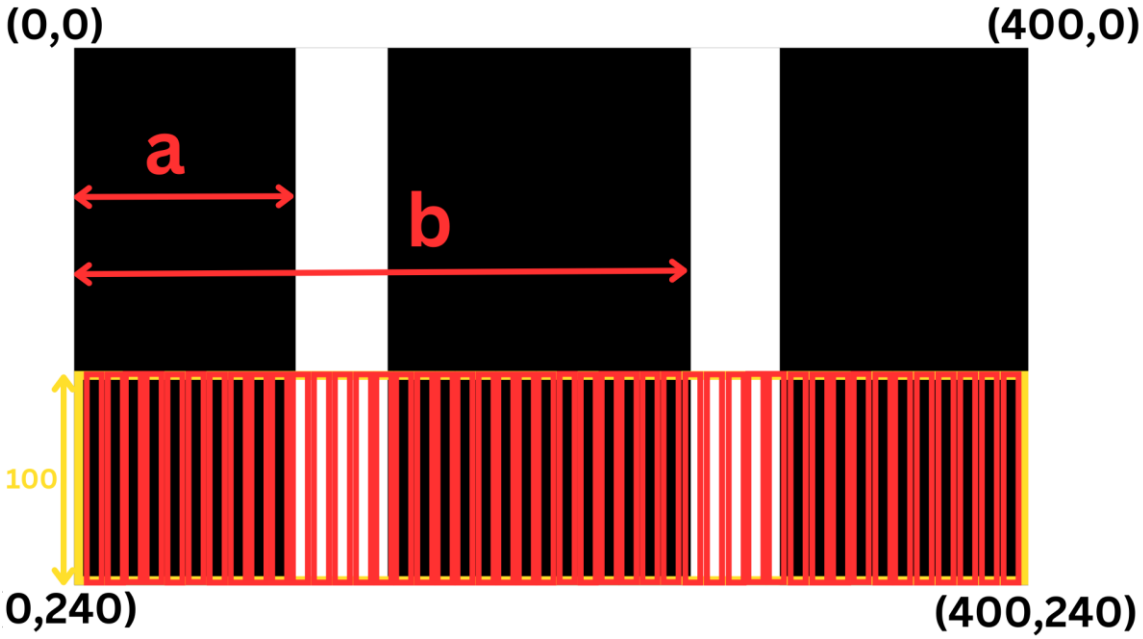
Canny kenar tespit edicisi, görüntülerdeki çok çeşitli kenarları algılamak için çok aşamalı bir algoritma kullanan bir kenar algılama algoritmasıdır. 1986 yılında John F. Canny tarafından geliştirilmiştir. Şerit kenarlarının tespit edilmesi, şeritlerin nerede başlayıp nerede bittiğinin anlaşılması için kullanılmaktadır. Bu işlemin uygulanmaması durumunda şeritlerin etrafında bulunan beyaz cisimler de şerit olarak algılanır. Bu da yanlış hesaplamalara sebep olmaktadır. Canny kenar tespit edicisi ile kenarlar daha keskin bir doğrulukla tespit edilmiştir. Bu da birçok hatanın önüne geçilmesini sağlamıştır.

Eşiklenmiş görüntülerin birleştirilmesi

Daha önce elde edilen eşik görüntü ile Canny kenar tespit edicisi ile elde edilen görüntü birleştirilmiştir.

Histogram hesaplama işlevi

Şerit algılama için histogram hesaplanması



Şekil 6.10: Şerit algılama algoritmasının açıklaması

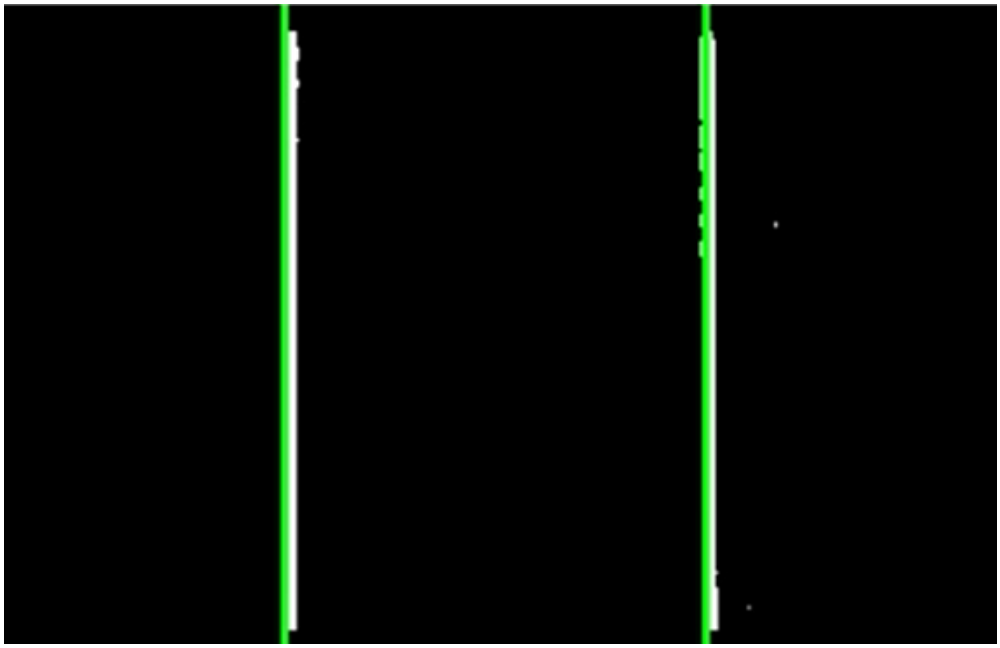
Yukarıda şekil 6.10'da gösterilen görüntüde şeritlerin nasıl tespit edildiği açıklanmıştır.

Rect = (x, y, x+a, y+b)

Görüntünün sol alt köşesinden (0, 240) koordinatından 100 birim yukarı giderek ilgi alanı belirlenmiştir. (0, 240) ile (0,140) koordinatları arasına her birinin x eksenindeki uzunluğu 1 piksel olan dikdörtgenler çizilmiştir. Her bir dikdörtgenin boyutu (1, 100) pikseldir. Her yenilemede bu 100 piksel taranarak 0 ile 255 değerleri tespit edilmiştir ve değerler dinamik bir dizine kaydedilmiştir. Bu dizinde toplam 400 değer vardır. Bu dizin bir for döngüsü ile taranmaktadır. Şekil 6.10'da gösterilen görüntüde ilk pikselde hiç beyaz piksel bulunmamaktadır. Böylece 100x0 = 0 değeri dizinin ilk değeri olarak kaydedilmiştir. Beyaz piksellerin olduğu kısımlara

gelindiğinde oluşabilecek maksimum değer 100x255 = 25500 değeridir. Beyaz değerlerin en yoğun olduğu dizindeki değerlerin olduğu bölümden bir çizgi çizilerek şeritlerin orta noktası tespit edilmiştir. Bunun için görüntü x eksenine göre ortadan ikiye bölünmüştür. İlk 200 dizin değerinde beyaz piksellerin en yoğun olduğu noktadan bir çizgi çizilmiştir. Aynı şekilde dizinde bulunan son 200 değerin içinden de beyaz piksellerin en yoğun olduğu değer tespit edilerek o değerin bulunduğu noktadan görüntü boyunca dik bir çizgi çizilmiştir. Oluşan görüntü aşağıda şekil 6.11'de gösterilmiştir.

Arr = [0, 0, 0, 0, 0, 0, ….23495, 25500, 2121, 0, 0, 0, 0, 0, 0, 0…. 1246, 24879, 25500, 0, 0, 0, 0, 0, 0, 0,]



Şekil 6.11: Şeritlerin tespit edilmesi

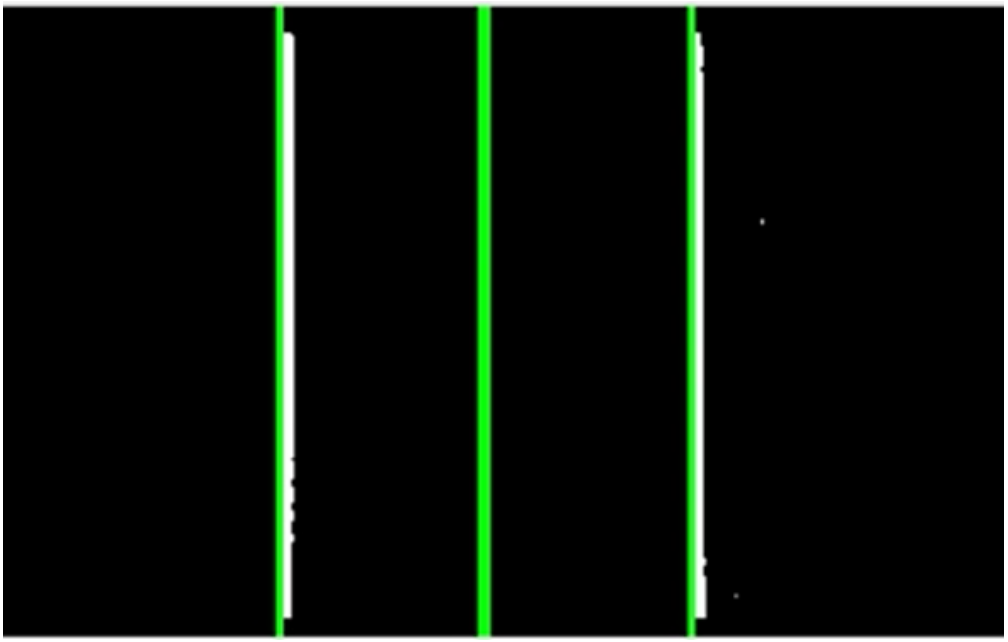
Şerit sonu için yoğunlukların toplamının hesaplanması

yol_sonu = toplam(histogram_yol_sonu)

Algılanan şeritlerin görüntü üzerinde çizilmesi

- Çizgi çiz final_goruntu (sol_serit_pozisyonu)'dan (sol_serit_pozisyonu) 'a kadar
- Çizgi çiz final_goruntu (sag_serit_pozisyonu)'dan (sag_serit_pozisyonu)'a kadar

Çizgi merkezi, (sağ şerit pozisyonu – sol şerit pozisyonu) / 2 + sol şerit pozisyonu formülü ile hesaplanmıştır.

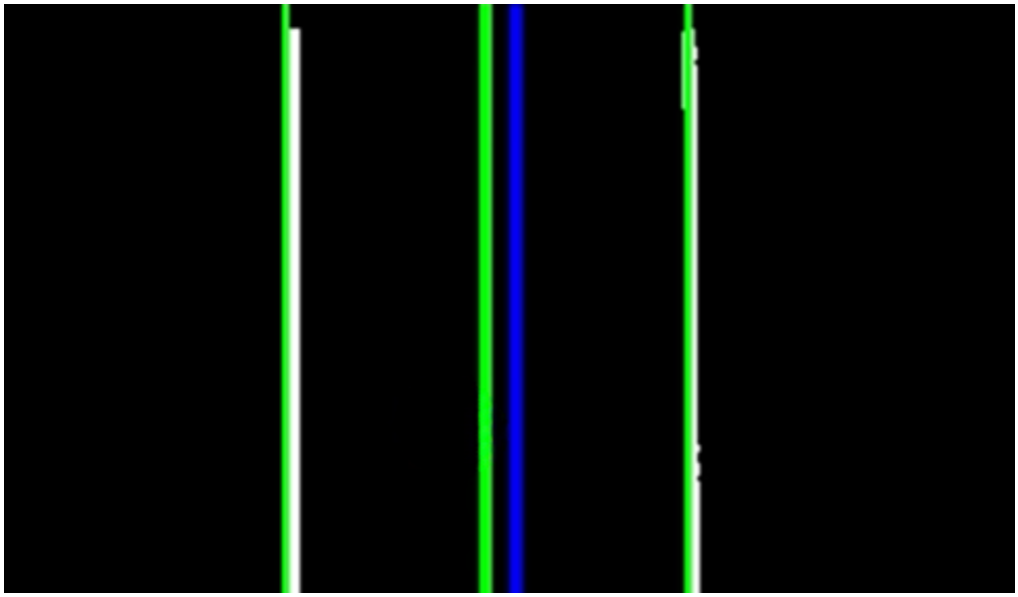


Şekil 6.12: Şeritlerin ortasına çizgi çekilmesi

Şerit merkezinin hesaplanması

serit_merkezi = (sag_serit_pozisyonu - sol_serit_pozisyonu) / 2 + sol_serit_pozisyonu

Şerit merkezinin hesaplanması ve çizginin çekilmesi şekil 6.12'de gösterilmiştir.



Şekil 6.13: Görüntünün orta noktasının belirlenmesi

Görüntü üzerinde şerit merkezini ve referans merkezini çizilmesi

Yukarıda şekil 6.13'te görüntünün orta noktasının belirlenmesi gösterilmiştir. Çizgilerin merkezini hesapladıktan sonra görüntünün orta noktasını da mavi çizgi ile gösterilmiştir. Araç yolun ortasına düzgün bir şekilde yerleştirildikten sonra mavi çizgi ile yeşil çizginin üst üste gelmesi sağlanarak aracın kalibrasyonu tamamlanmıştır.

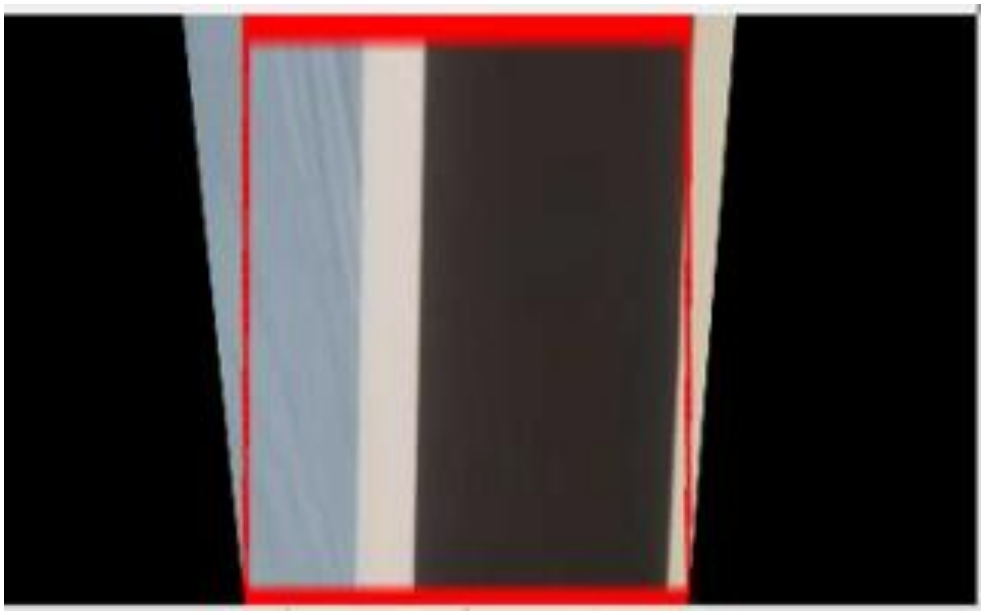
- Çizgi çiz final_goruntu from (serit_merkezi)'nden (serit_merkezi)'a kadar
- Çizgi çiz final_goruntu from (merkez_goruntu)'dan (merkez_goruntu)'ye kadar

Merkezden sapmanın hesaplanması

Sonuc = serit_merkezi - merkez_goruntu

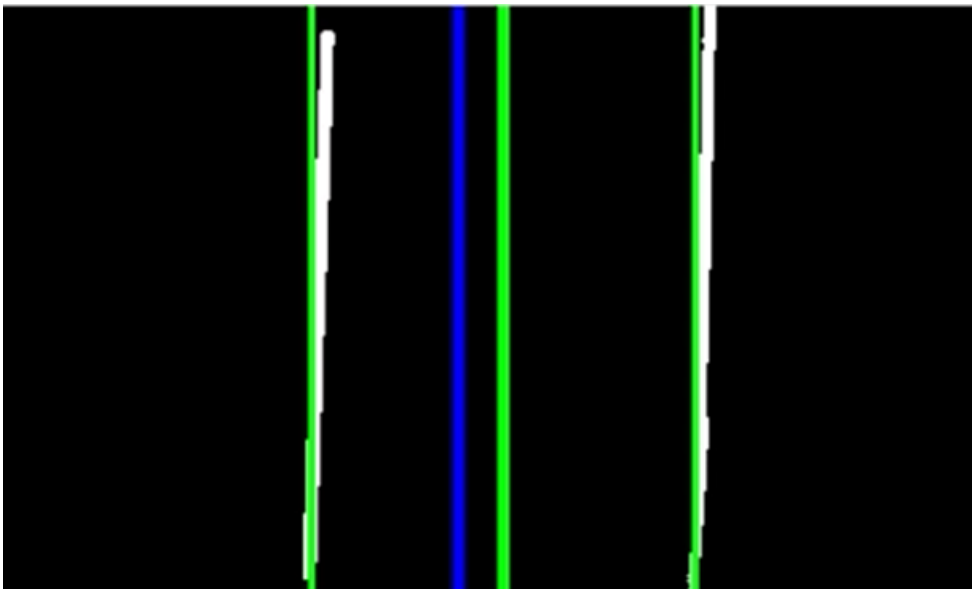
Bu noktada aracın sağa gitmesi ve sola gitmesi için gereken formül yukarıda gösterilmiştir. Şerit merkezi yani şeritlerin ortanca değerinden çekilen yeşil çizgi ile, merkez görüntü yani görüntünün tam ortasından çekilen mavi çizginin arasında oluşan sapma bize aracın tam olarak nasıl bir hareket yapacağını anlaşılmasını sağlamaktadır.

Şekil 6.14: Sağa dönüş testi standart görünüm



Şekil 6.15: Sağa dönüş testi perspektif görünümü

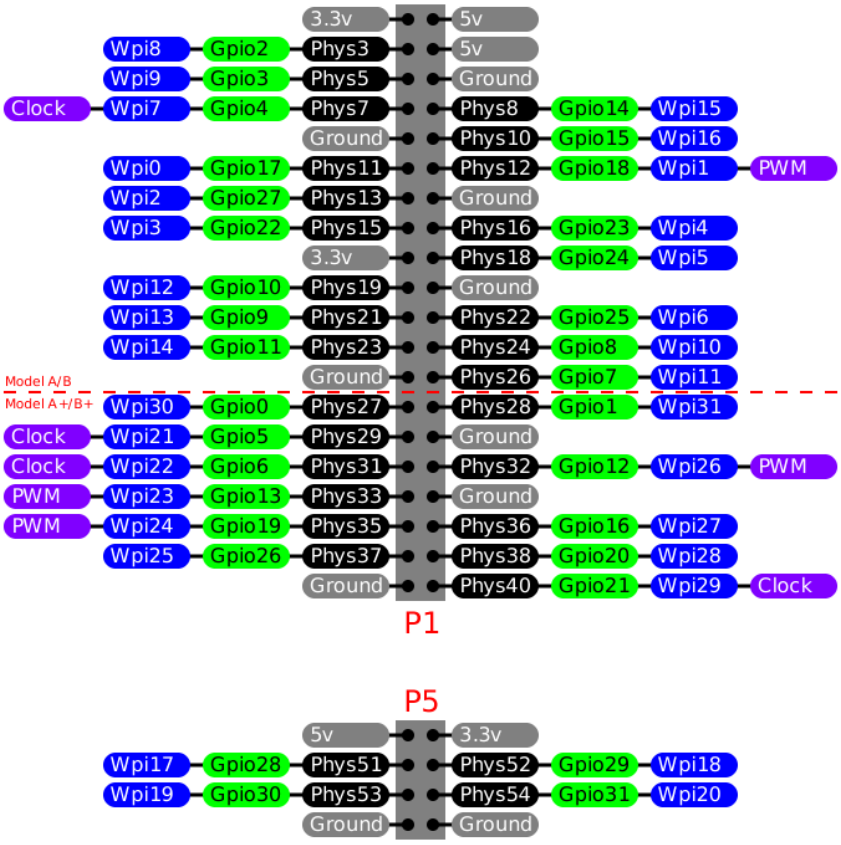
Yukarıda şekil 6.14 ve 6.15'te sağa dönüş testi ile ilgili görseller gösterilmiştir.



Şekil 6.16: Şeritlerin tespiti ve dönüş yönünün hesaplanması

Şekil 6.16'da görüldüğü üzere aracı bir miktar sola çevirdiğimizde şerit merkez çizgisi ile görüntü merkez çizgisi arasında bir fark oluşmuştur. Bu farkı bir değer ile ifade ettiğimizde Sonuç = şerit merkezi – merkez görüntü formülünden 17 gelmektedir. Bu sayede şerit merkez çizgisi ile görüntü merkez çizgisi arasında oluşan farklar bir sayısal değer ile ifade edilmesi sağlanmıştır. Kodun ilerleyen safhalarında bu değerler aracın sağa veya sola gitmesi için alınan kararlarda kullanılmıştır. Örneğin eğer sonuç değeri 0'dan büyük ise sağa git veya eğer sonuç değeri 0'dan küçük ise sola git. Bu kısımdan sonra artık Raspberry Pi 4 bilgisayarı ile hesaplanan sonuçlar Arduino Uno'ya yollanarak aracın hareket etmesi sağlanmıştır.

Aşağıda şekil 6.17'de Raspberry Pi 4 bilgisayarının Genel Amaçlı Giriş/Çıkış (General Purpose Output Input – GPIO) pinleri gösterilmiştir.



Şekil 6.17: Raspberry Pi 4 çıkış pinleri

GPIO pinlerini ayarlama

wiringPi kütüphanesi kurulumu;

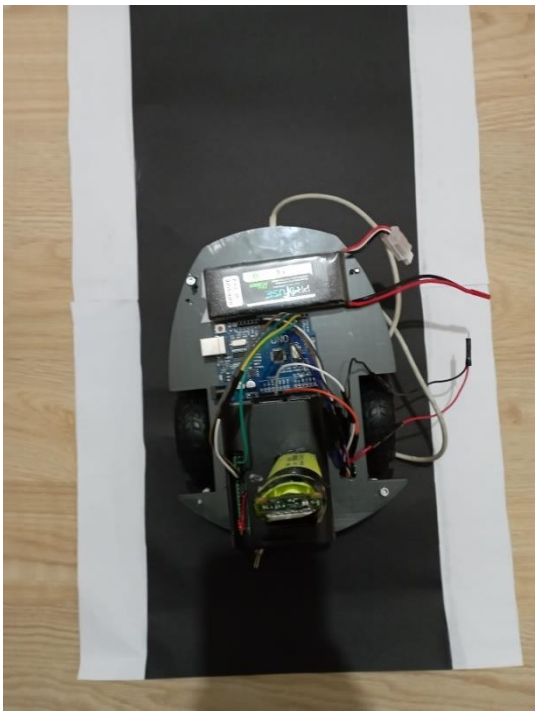
- Pin 21'i çıkış pini olarak ayarla
- Pin 22'yi çıkış pini olarak ayarla
- Pin 23'ü çıkış pini olarak ayarla
- Pin 24'ü çıkış pini olarak ayarla

7. TEST ORTAMININ HAZIRLANMASI

7.1 Yolun Hazırlanması

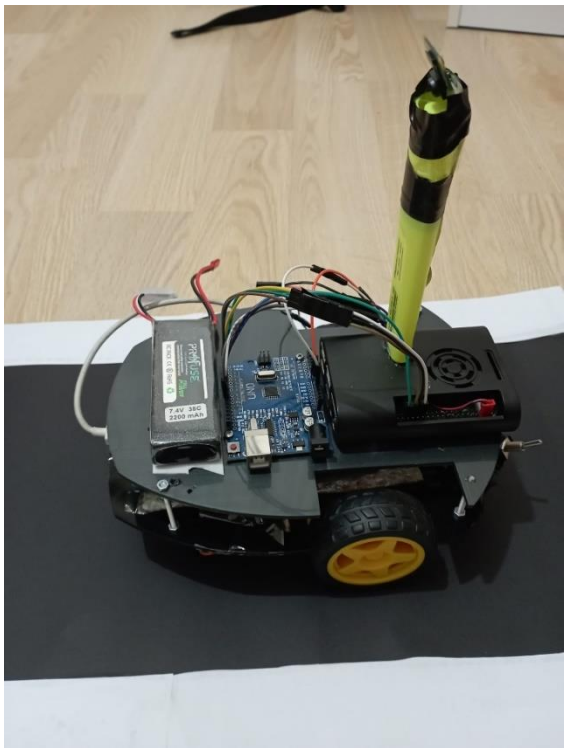
Şekil 7.1: Test yolunun hazırlanması

Şekil 7.1'de test yolu gösterilmiştir. Aracı test etmek için oluşturulan yolda zıt renkler tercih edilmiştir. Siyah ile beyaz arasındaki farkı daha kolay yakalaması görüntüde oluşacak olan gürültüleri azaltacaktır. Bu da oluşturulan algoritmanın performansını ölçmek için daha doğru sonuçlar verecektir. Bu sebeple siyah yol üzerine beyaz şeritler tercih edilmiştir. Siyah yolun elde edilmesi için 10 adet 19mm kalınlığında 50x70 cm genişliğinde 100 gram ağırlığında fon kartonlar tercih edilmiştir. Aracın yol üzerinde kaymaması için yol zemine sabitlenmiştir. Düz yol ve virajlı yollar testler için hazırlanmıştır. Işık yansımasının önüne geçmek için mat fon karton tercih edilmiştir. Işık yansımaları, görüntüde gürültülere sebep olmakta ve sonuçları olumsuz etkilemektedir.



Şekil 7.2: Aracın yola yerleştirilmesi

Şekil 7.2 ve 7.3'te aracın yola yerleştirilmesi gösterilmiştir. Araç yolu ortalayacak şekilde piste yerleştirildikten sonra kamera kalibrasyonu yapılması için yola yerleştirilmiştir. Görüntünün ortalanması algoritmanın düzgün çalışabilmesi için çok önemlidir. Araç kenarlarında 5 cm boşluk kalacak şekilde yola yerleştirilmiştir. Yola yerleştirilen aracın kamerası yolu ortalayacak şekilde dikey ve yatayda açıları ayarlandıktan sonra sabitlenmiştir. Aracın hareketi sırasında oluşan titreşimden ötürü sallanmalar olmakta ancak sonucu olumsuz etkilememektedir. Testler için siyah ve beyaz dengesi ayarlanmalıdır. Trackbar (iz çucuğu) üzerinden beyaz şeritleri algılaması ve diğer renkleri maskelemesi sağlanarak kameradan alınan görüntü çalışmaya uygun hale getirilmiştir.



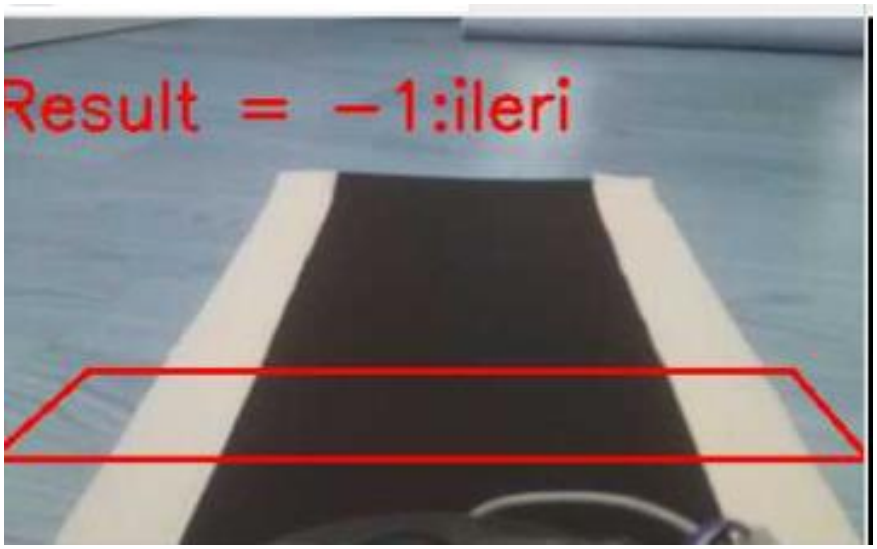
Şekil 7.3: Aracın yola yerleştirilmesi (farklı açı)

8. TESTLER

8.1 Giriş

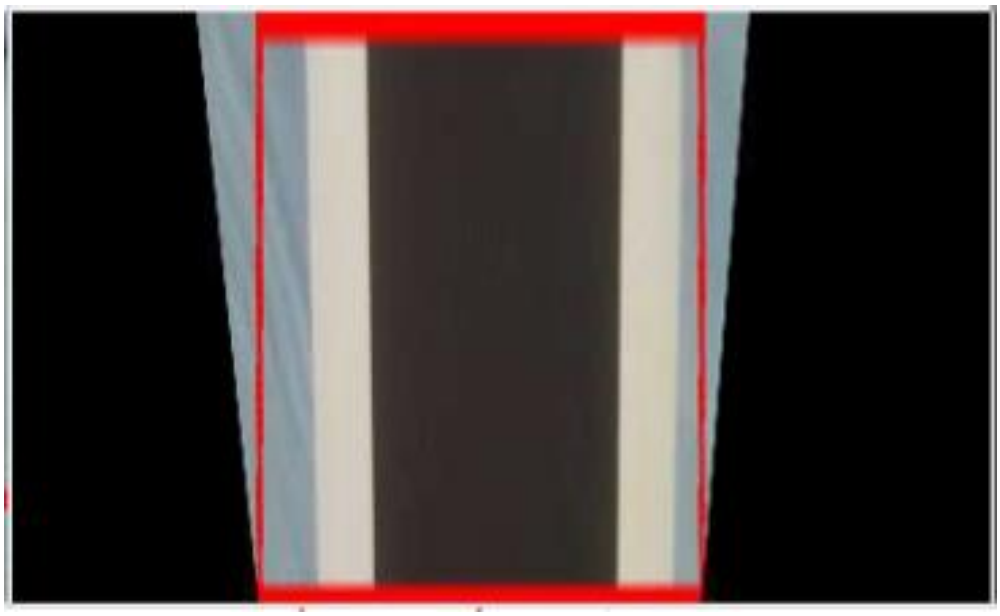
Aracı yolun ortasına düzgün bir şekilde yerleştirildiğinde, aracın kamerası önündeki yolun bir kısmını görmektedir. Bu kısım kırmızı dikdörtgende gösterilen iki yatay çizgi ile sınırlandırılmıştır. Bu yatay çizgilerin ilki, aracın şasisinin bittiği noktadan 4 cm uzaklıktadır. Diğer yatay çizgi, bu yatay çizgiye 16 cm uzaklıktadır. Dolayısıyla araç, önündeki yolun 16 cm'lik bir bölümünü görmekte ve bu bölümde yapılan hesaplamalara göre karar vermektedir. Aracın karar mekanizması, kameradan aldığı verilerden başlar. Raspberry Pi 4 Tek kartlı bilgisayar (single-board computer) ile işlenen C++ kodları, sonuçları Arduino Uno Mikrodenetleyici 'ye aktararak anlık durumu motor sürücüye iletmesini sağlar. Arduino Uno Mikrodenetleyici, Motor Sürücüye son komutu yollar. Motor sürücü de son komutu DC Motorlara yollar. Araç bu şekilde yol üzerinde düzgün bir şekilde hareket etmeyi başarmaktadır. Temel komutlar, ileri git, sola dön, sağa dön, dur ve geri git komutlarıdır. Ayrıca araç Dur tabelası, hemzemin geçit tabelası, yaya geçidi tabelası gördüğünde durmaktadır. Kırmızı ışıkta durmakta, ışık yeşile döndüğünde harekete geçmektedir. Bu projede temel seviyede hareket edebilen, görüntü işleme tekniği ile gördüğü objeleri tespit edebilen ve tepki verebilen bir otonom robot hayata geçirilmiştir.

8.1.1 İleriye Gidiş Testi



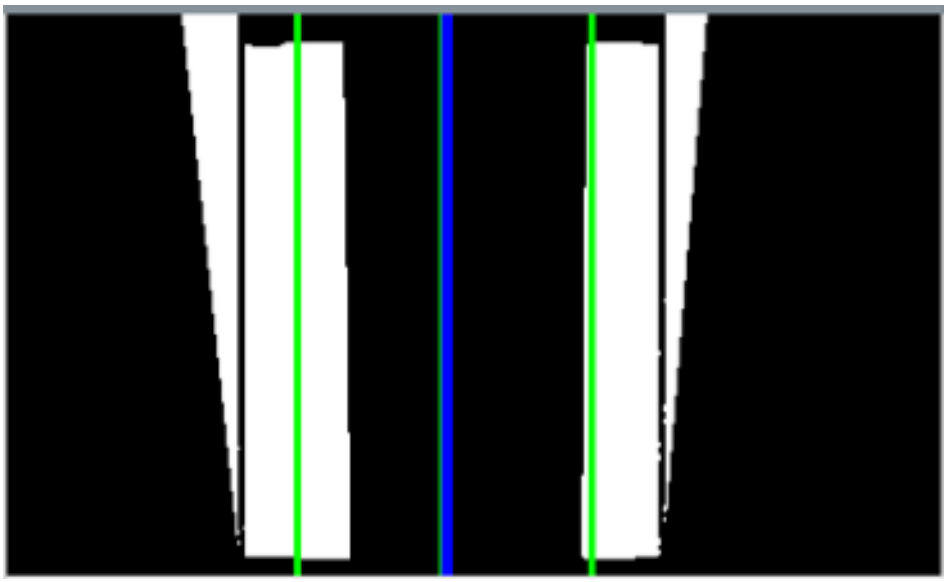
Şekil 8.1: İleri gidiş testi

Şekil 8.1'de ileri gidiş testi gösterilmiştir. İlk olarak araç yukarıda şekilde gösterilen görüntüyü bilgisayara bağlı kameradan almaktadır. Bu görüntü 400*240 piksel boyutlarındadır. Yatayda 400 ve dikeyde 240 piksele sahip olan bu görüntüyü yatayda ikiye böldüğümüzde ortadan geçen 199'uncu piksel referans noktasıdır. Piksel sayımı sol üstten başlamaktadır. OpenCV kütüphanesinde koordinat olarak görüntünün sol üst (0, 0), sağ üst (400, 0), sol alt (0, 240) ve sağ alt (400,240) şeklinde temsil edilmektedir. Yatayda 199'uncu pikseli temsil eden (199, 0) noktası ile (199, 240) noktaları arasında bir dikey çizgi çekilir. Bu çizgi görüntünün referans çizgisidir. Referans çizgisinin sağında kalan pikseller (200,400) ve solunda kalan pikseller (0, 198) pikselleridir.



Şekil 8.2: İleri gidiş testi perspektif görünümü

Şekil 8.2'de ileri gidiş testi perspektif görünümü gösterilmiştir. Bu piksellerin hesaplamalarını yapmadan önce görüntüyü perspektif bakıştan kuşbakışına çevirmek gerekmektedir. Bu işlem bir önceki şekilde gösterilen kırmızı dikdörtgenin köşelerini yukarıdan bakacak şekilde aynı enlem ve boylama eşit olacak şekilde piksellere yerleştirerek yeni bir görüntü elde edilir. Bu görüntü ilgi alanımızı belirtmektedir. Bu alan dışında kalan alanların hesaplamaları dahil edilirse araç yanlış yönlendirileceği için, bu seçtiğimiz ilgi alanını hesaplamak doğru sonuçları elde etmemizi sağlayacaktır.



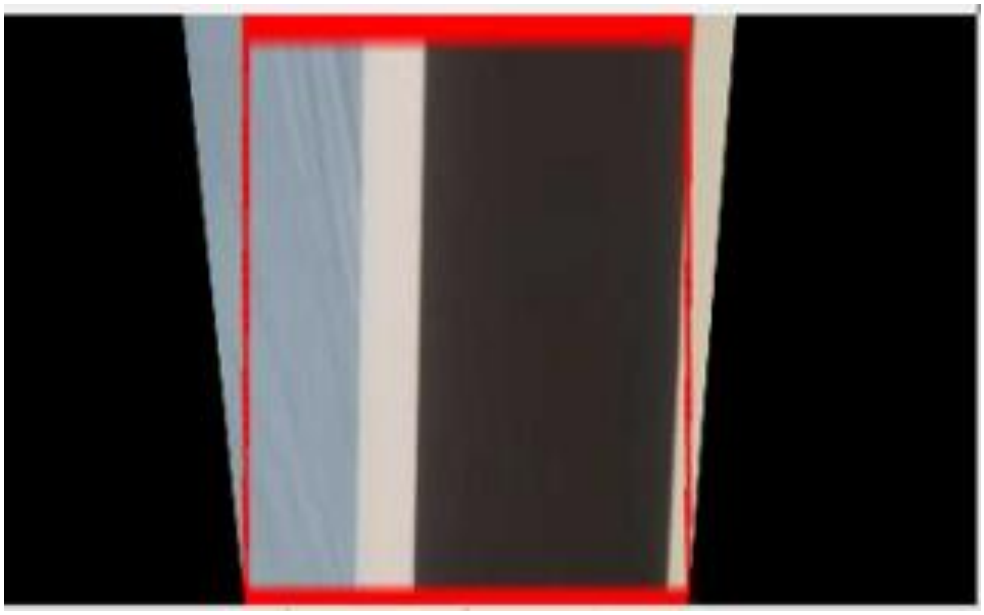
Şekil 8.3: Son görünüm

Son olarak görüntü iki renk uzayına indirgendiğinde ortadan geçen mavi referans çizgisi görüntüyü iki eşit parçaya bölmektedir. Bu iki eşit parçanın beyaz ağırlıkları ayrı ayrı hesaplanmaktadır. İki tarafın da beyaz dengesi eşit olduğundan sonuç -1 gelmektedir. -8 ile 8 arasındaki değerler aracın ileri gitmesine sebep olacaktır. Eğer sadece 0 değerinde ileri gitmesi beklenirse araç sürekli sola veya sağa gitmeye çalışacak ve bu yüzden ilerleyemeyecektir. Bu yüzden iki taraftan da tolerans bırakılması aracın düz ilerlemesini sağlayacaktır. Şekil 8.3'te son görünüm gösterilmiştir.

8.1.2 Sağa Dönüş Testi

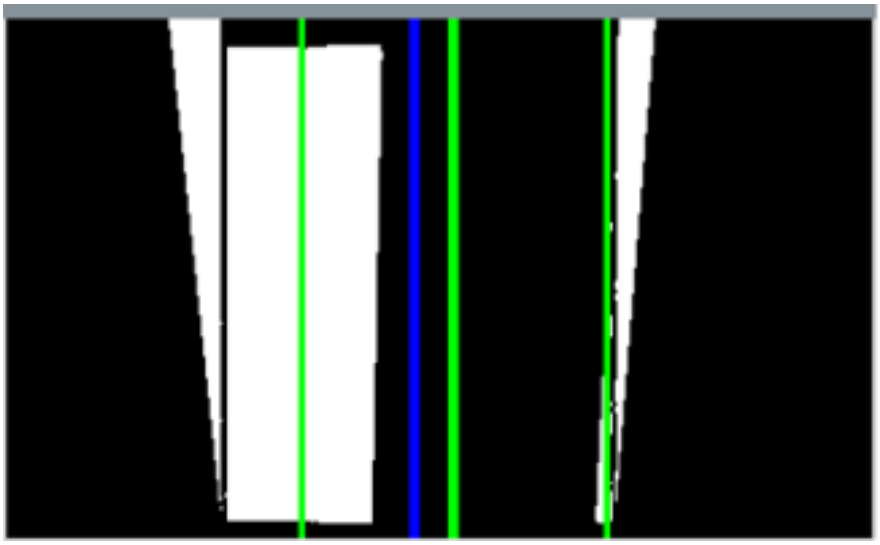
Şekil 8.4: Sağa dönüş testi

Şekil 8.4'te sağa dönüş testi gösterilmiştir. Teste devam ederek araç sola doğru çevrilmiştir. Aracı biraz sola çevirdiğimizde kameradan alınan orijinal görüntü yukarıdaki gibidir. Görüntüyü yatay eksende ortadan ikiye ayırdığımızı düşündüğümüzde görüntüdeki sol şeridin orta çizgiye daha yakın ve sağ şeride daha uzak olduğunu görülmektedir. Şekil 8.5'te sağa dönüş testi perspektif görünümü gösterilmiştir.



Şekil 8.5: Sağa dönüş testi perspektif görünümü

Şekil 8.5'te görüldüğü üzere perspektif görüntü kuşbakışı görüntüye çevrilerek hesaplamalara uygun hale getirilmiştir.



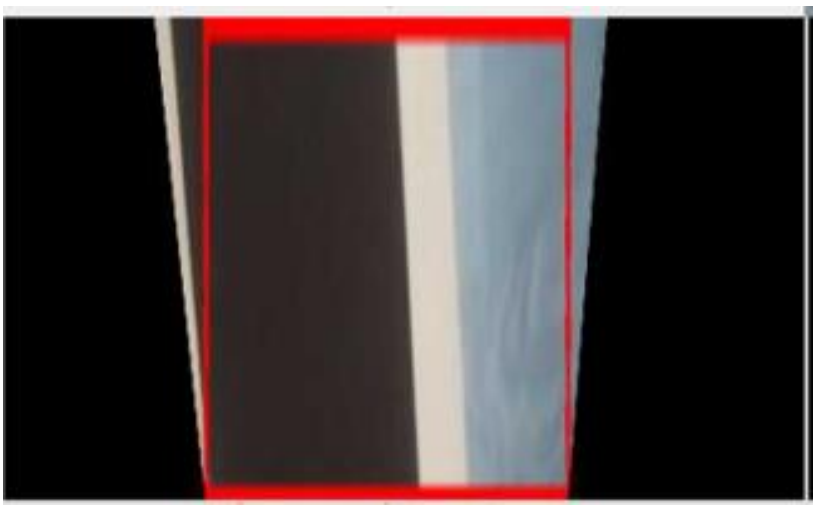
Şekil 8.6: Sağa dönüş testi son görünüm

Şekil 8.6'da görüldüğü üzere son görüntüde görüntü işleme tekniklerini uyguladığımızda aracın sol kısmında kalan beyaz alanın sağ kısımda kalan beyaz alana göre yoğunluk olarak daha fazla olduğu görülmüştür. Hesaplar sonucunda sonuç 18 çıkmış ve Arduino Uno Mikrodenetleyiciye sağa dön komutu gönderilmiştir. Sağa dön komutu ile araç sağa dönüşünü gerçekleştirmiştir.

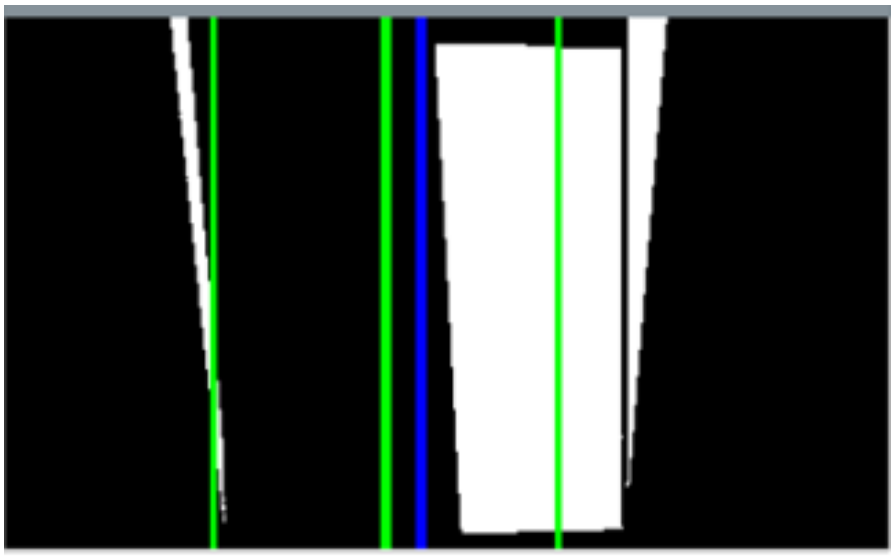
8.1.3 Sola Dönüş Testi

Aynı testler sola dönme komutu için de test edilmiştir. Aracı sağa çevirdikten sonra araçtan alınan perspektif, kuşbakışı ve maskelenmiş görüntüler şekil 8.7 ve şekil 8.8'de gösterilmiştir.

Şekil 8.8: Sola dönüş testi başlangıç görüntüsü



Şekil 8.7: Sola dönüş testi perspektif görüntüsü



Şekil 8.9: Sola dönüş testi final görüntüsü

Hesaplamalar sonucu -16 ile sola dön komutu hesaplanmıştır. Yukarıdaki şekil 8.9'da sola dönüş testi final görüntüsü gösterilmiştir.

Şekil 8.10: Terminal görüntüsü

Araç sola dönerek yoluna devam etmiştir. Şekil 8.10'da terminal görüntüsü gösterilmiştir.

8.1.4 Hemzemin Geçit Testi

Test için öncelikle 400 adet pozitif ve 1200 adet negatif görüntü makine öğrenimi algoritmasına yüklenerek çalıştırılmıştır. Hazırlanan xml dosyası C++ koduna entegre edilmiştir.

Araç sağ şeritte gördüğü yaya geçidini 15 cm mesafede tanır. Araç yaya geçidi tabelasını gördükten sonra 10 saniye durur. Ardından yoluna devam eder. Bu işlemi hemzemin geçidi her gördüğünde tekrar eder. Hemzemin geçidin olmadığı yerlerde hemzemin geçit görmemesi için negatif görüntü olarak hemzemin geçidin olmadığı görüntüler makine öğrenmesi algoritmasına yüklenmelidir. Ayrıca yaya geçidi tabelası ve park tabelası görüntüleri de negatif görüntülere eklenerek başarı oranı daha da arttırılmıştır. Yapılan testlerde araç hemzemin geçidi başarı ile tanımlamaktadır. Hemzemin geçidi tabelası tespiti şekil 8.11'de gösterilmektedir.

Şekil 8.11: Hemzemin geçidi görüntü tespiti

8.1.5 Yaya Geçidi Testi

Hemzemin geçitte uygulanan adımlar yaya geçidi tabelası için de uygulanmıştır. Yaya geçidi tespiti yapmak için görüntü işleme tekniklerini kullanmak, nesne tespiti ve sınıflandırma problemleriyle ilgilidir. İlk olarak, yaya geçidi içeren bir veri seti toplanır. Bu veri setine hem yaya geçitleri hem de bu geçitlerin bulunmadığı görüntüler dahil edilmiştir. Veri setini etiketlenerek, modeli eğitirken hangi bölümlerin yaya geçidi içerdiği belirlenir. Görüntüleri boyutlandırma, renk dönüşümleri ve normalizasyon gibi ön işleme adımları gerçekleştirilir. Veri seti eğitim, doğrulama ve test kümelerine ayrılarak başarı oranı test edilir. Yaya geçitleri tespiti için kullanabilecek birçok nesne tespiti modeli bulunmaktadır. YOLO (You Only Look Once), SSD (Single Shot Multibox Detector) ve Faster R-CNN gibi modeller popüler seçeneklerdir. Bu çalışmada Cascade GUI kullanılmıştır. Modeli gerçek zamanlı uygulamak için bir görüntü akışından veya bir kameradan gelen görüntüler sürekli olarak işlenir. Yeterli başarıya ulaşan modeller gerçek zamanlı teste tabii tutulur. Tez çalışması için en uygun model seçilir. Araç sağ şeritte gördüğü yaya geçidini 15 cm mesafede tanır. Araç yaya geçidi tabelasını gördükten sonra 10 saniye durur. Önündeki bölgeyi analiz eder. Eğer önüne bir engel çıkarsa durmaya devam eder. Ardından yoluna devam eder. Yaya geçidi tabelası tespiti şekil 8.12'de gösterilmiştir.

Şekil 8.12: Yaya geçidi görüntü tespiti

8.1.6 Park Testi

Şekil 8.13'te gösterilen park görevi için makine öğrenme algoritması ile tanıtılan park tabelası parkurun sağ tarafına yerleştirilmiştir. Araç park tabelasına 15 cm mesafeye yaklaştığında park tabelasını algılayarak mevcut görevini sonlandırır. Park etme algoritmasına geçen program artık önündeki beyaz dikdörtgen alanı taramaktadır. Beyaz park alanını gördüğünde dikdörtgenin içine girerek görevini tamamlar. Robot park bölgesine girdiğinde ana bilgisayar, motor sürücüye dur komutu yollar. Park görevini de tamamlayan robot başarılı bir şekilde parkuru bitirmiştir.

Şekil 8.13: Park tabelası tespiti

(Çetinkaya, 2023) Tezinde kullandığı araç ile bu tezde kullanılan araç karşılaştırıldığında sonuçlar şu şekildedir: Python yazılımı kullanarak, Jupyter Lab. üzerinde kodlamalar yapılmıştır. Kullanılan kamera 65fps, 224x224 piksel görüntüyle çalışmaktadır. Aracı eğitmek için Python kütüphanelerinden Pytorch kullanılmıştır.

Kullanılan yol parke zemin üzerine 150 cm x 150 cm uzunluğunda 30 cm genişliğindedir. Yol kenar çizgileri mavi, yol çizgileri kırmızı renklidir. Diğer yollara göre zemin değiştirilmiş, uzunluk arttırılmıştır. Dönüşlerde yine yoldan çıkmalar olmuştur. Jetson Nano duruma göre ideal hız 0,6 m/sn sabit ayarlanmıştır. Daha düşük hızlarda çok ağır ilerleme ve bazen durmalar görülmüştür. Yüksek hızlarda ise araç yeterli tepkiyi gösteremeden yoldan çıkmaktadır. (Çetinkaya, 2023)

Şekil 8.14: Çalışmada kullanılan yol (Çetinkaya, 2023)

Şekil 8.14'te gösterilen Çetinkaya'nın çalışmasında kullanıldığı araç 0.6 m/sn hızda sabit olarak yolu tamamlama görevini yerine getirmiş ve algıladığı nesnelere çarpmadan geçmeyi başarmıştır. Bu tezde kullanılan araç 1,5 m/sn maksimum hıza ulaşarak yolu başarılı bir şekilde tamamlamış ve algıladığı nesnelere çarpmadan geçmeyi başarmıştır.

Tüm modülleri birleştirmek için, Raspberry Pi'nin işlem gücünü en maliyetli algoritmalar arasında paylaşmak için verimli bir yöntem seçilmiştir. Böylece, hareket komutlarını göndermekten sorumlu kısım bir döngü içinde çalışmakta ve şerit algılama işleminin gerçekleştirildiği ayrı bir iş parçacığını başlatmaktadır. Ayrıca sensörlerden gelen tüm giriş sinyallerinin okunmasının yanı sıra trafik işaretlerini algılama algoritması da mümkün olduğunca az CPU kaynağı talep etmek için her 150 ms'de bir çalışmaktadır (Bianca ve diğ. 2018).

Bir başka çalışmada Raspberry Pi ile yapılan Otonom araçta elde edilen verilerin tablosu aşağıda Tablo 8.1'de belirtilmiştir. (Bianca ve diğ. 2018)

Tablo 8.1: Otonom araç tarafından yarış pistinde elde edilen değerler (Bianca ve diğ. 2018)

No	İş	Mesafe (m)	Süre (ms)
1	Şerit takibi	4.95 m	35.000 ms
2	Dur levhası algılama	-	100 ms
3	Dur	-	1.000 ms
4	Park etme	1.4 m	9.000 ms
5	Kavşakta sağa dönüş	1.35 m	9.500 ms
6	Kavşakta sola dönüş	1.8 m	12.850 ms

Tablo 8.2: Tezde kullanılan otonom araç tarafından yarış pistinde elde edilen değerler

No	İş	Mesafe (m)	Süre (ms)
1	Şerit takibi	4.95 m	27000 ms
2	Dur levhası algılama	-	100 ms
3	Dur	-	700 ms
4	Park etme	1.4 m	7000 ms
5	Kavşakta sağa dönüş	1.35 m	6000 ms
6	Kavşakta sola dönüş	1.8 m	9.000 ms

Yukarıda Tablo 8.1 ve Tablo 8.2'de görüldüğü üzere bu tez kapsamında geliştirilen otonom araç şerit takibi, dur levhası algılama, park etme ve kavşak dönüşlerinde olumlu bir yol kat etmiştir. Bu görevlere ek olarak yaya şeridi tespiti, hemzemin geçit tespiti de yapabilmektedir. Ayrıca trafik ışıklarını da başarılı bir şekilde algılayabilmektedir. Süre ölçümleri kronometre ile ölçülmüştür. Referans tezde yer alan mesafeler aynı şekilde katedilmiştir.

9. SONUÇLAR

Bu tez çalışması, görüntü işleme teknikleri üzerine odaklanarak otonom sürüş algoritması tasarlamayı amaçlamıştır. Yapılan araştırma ve deneylerin sonuçları aşağıda özetlenmiştir:

Görüntü İşleme Algoritmalarının Seçimi ve Optimizasyonu:

Tez kapsamında, çeşitli görüntü işleme algoritmaları incelenmiş ve otonom sürüş uygulaması için en uygun olanlar seçilmiştir (Canny kenar tespiti, Gaussian filtreleme, Histogram Çıkarma). Bu algoritmalar, görüntüden nesne algılama, yol takibi ve engel tespiti gibi temel görevleri gerçekleştirmek üzere optimize edilmiştir.

Simülasyon ve Gerçek Ortam Deneyleri:

Tasarlanan otonom sürüş algoritması hem simülasyon ortamında hem de gerçek dünya koşullarında test edilmiştir. Yapılan deneyler, algoritmanın başarıyla çeşitli yol durumlarına uyum sağladığını ve güvenilir bir şekilde otonom sürüş gerçekleştirebildiğini göstermiştir.

Gelecek Çalışma İmkanları:

Bu tez çalışması, otonom sürüş alanındaki tekniklerin bir sentezi olup (nesne tespiti, şerit tespiti, ışık tespiti vb.), gelecek araştırmalara ışık tutacak bir temel oluşturmuştur. Özellikle, çevresel koşullara daha etkili uyum sağlayabilen ve gerçek zamanlı hata düzeltmeye odaklanan çalışmalar, bu alandaki potansiyel geliştirmeleri açığa çıkarabilir.

Bu sonuçlar, görüntü işleme teknikleri ile tasarlanan otonom sürüş algoritmasının başarıyla uygulanabileceğini ve gelecekteki otomasyon teknolojilerinin gelişimine katkı sağlayabileceğini göstermektedir.

10. KAYNAKLAR

Adrian Kaehler; Gary Bradski. “Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library.”(2016).

Avrupa Komisyonu. “Advanced Driver Assistance Systems in Europe.”(1998).

Road Safety Statistics: What is Behind The Figures? Brüksel, Belçika: Avrupa Komisyonu, (2022).

Bainbridge, L. “Ironies of automation. Automatica, 19, 775-779.”(1983).

Bekiaris, E., Petica, S., Brookhuis, K.A. *Driver needs and public acceptanceregarding telematic in-vehicle emergency control aids.* Brüksel, (1997).

Bernd Jähne; Horst Haußecker. “Computer Vision and Applications, A Guide for Students and Practitioners. Academic Press.”(2000).

Bianca-Cerasela-Zelia Blaga, Mihai-Adrian Deac, Rami Watheq Yaseen Al-doori, Mihai Negru, Radu Dnescu. “Miniature Autonomous Vehicle Development on Raspberry Pi .”(2018).

Boden, Margaret Ann. “Mind as Machine: A History of Cognitive Science.”(2006).

Broggi, A., Bertozzi, M., & Fascioli, A. “Argo and the millemiglia in automatico tour. IEEE Intelligent Systems and their Applications, 14(1), 55–64.”(1999).

Broggi, A., Cerri, P., Debattisti, S., Laghi, M. C., Medici, P., Molinari, D., Panciroli, M., &. “ Proud—public road urban driverless-car test. IEEE Transactions on Intelligent Transportation Systems, 16(6), 3508–3519.”(2015).

Broggi, A., Cerri, P., Felisa, M., Laghi, M. C., Mazzei, L., & Porta, P. P. “The vislab intercontinental autonomous challenge: an extensive test for a platoon of intelligent vehicles. International Journal of Vehicle Autonomous Systems, 10(3), 147–164.”(2017).

Broggi, A., M. Buzzoni, S. Debattisti, P. Grisleri, M. C. Laghi, P. Medici. “Extensive tests of autonomous driving technologies, IEEE Transactions on Intelligent Transportation Systems, 14 (3) 1403.”(2013).

Brookhuis, K.A., De Waard, D. “The use of psychophysiology to assess driver status.Ergonomics, 36, 1099-1110.”(1996).

Buller, W.; Wilson, B., Garbarino, J., Kelly, J.; Subotic, N.; Thelen, B.; Belzowski, B. *Radar Congestion Study*. Washington, DC, USA, (2018).

Bundesministerium für Verkehr und digitale Infrastruktur. “Info- Papier ”automatisiertes Fahren”, Technical Report, Bundesministerium für Verkehr und digitale Infrastruktur, Berlin.”(2014).

Cardoso, V., Oliveira, J., Teixeira, T., Badue, C., Mutz, F., Oliveira-Santos, T.,. “A model-predictive motion planner for the iara autonomous car. In 2017 IEEE international conference on robotics and automation (ICRA) (pp. 225–230). IEEE.”(2017).

Cerri, P., Soprani, G., Zani, P., Choi, J., Lee, J., Kim, D., Yi, K., & Broggi, A. “Computer vision at the hyundai autonomous challenge. In 2011 14th international IEEE conference on intelligent transportation systems (ITSC) (pp. 777–783). IEEE.”(2011).

Chang, W.; Chen, C.; Hung, Y. “Tracking by parts:A bayesian approach with component collaboration. IEEE Trans. Syst. Man Cybern. Part B Cybern. 2019, 39, 375–388. [.”(2019).

Chellappa. “Digital Image Processing.”1992.

Chitradevi, B., ve P. Srimathi. “An overview on image processing techniques." International Journal of Innovative Research in Computer and Communication Engineering 2.11.”(2014).

Chong, Z. J., B. Qin, T. Bandyopadhyay, T. Wongpiromsarn, E. S. Rankin,. *Autonomous personal vehicle for the first-and last-mile transportation services, paper presented at the IEEE 5th International Conference*. Qingdao, (2011).

Congress, N. “The Automated Highway System: an idea whose time has come.”1994.

Çetinkaya, M. “Otonom araçlar için GPU kullanarak gerçek zamanlı yapay zeka temelli engel algılama sistem tasarımı .”(2023).

DARPA. “Grand Challenge .”(2004).

De Lima, D. A., & Pereira, G. A. S. “Navigation of an autonomous car using vector fields and the dynamic window approach. Journal of Control, Automation and Electrical Systems, 24(1–2), 106–116.”(2013).

De Lima, D. A., & Pereira, G. A. “Um sistema de visao estéreo para navegaçao de um carro autônomo em ambientes com obstáculos. In XVIII congresso brasileiro de automática (pp. 224–231).”(2010).

Dias, J. E. A., Pereira, G. A. S., & Palhares, R. M. “Longitudinal model identification and velocity control of an autonomous car. IEEE Transactions on Intelligent Transportation Systems, 16(2), 776–786.”(2014).

Dünya Sağlık Örgütü. *Global status report on road safety*. Cenevre, İsviçre, (2015).

Engelking, C. “The 'Driverless' Car Era Began More Than 90 Years Ago. .”(2017).

Englund, C., Chen, L., Ploeg, J., Semsar-Kazerooni, E., Voronov, A., Bengtsson, H. “he grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles. IEEE Wireless Communications, 23(4), 146–152.”(2016).

Erbs, F.; Barth, A.; Franke, U. “Moving vehicle detection by optimal segmentation of the dynamic stixel world. In Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, 5–9 June 2011; pp. 951–956.”(2011).

F. Mariut, C. Fosalau ve D. Petrisor. “Lane Mark Detection Using Hough Transform.”(2012).

Fenton, R. E. and R. J. Mayhan. “Automated highway studies at the Ohio State University.”(1991).

Gaurav Kumar ve Pradeep Kumar Bhatia. “A Detailed Review of Feature Extraction in Image Processing Systems.”(2014).

Geddes, N. B. *Magic Motorways*. New York, (1940).

Gregor, R., Lutzeler, M., Pellkofer, M., Siedersberger, K.-H., & Dickmanns, E. D. “Ems-vision: A perceptual system for autonomous vehicles. IEEE Transactions on Intelligent Transportation Systems, 3(1), 48–59.”(2002).

Guidolini, R., Badue, C., Berger, M., de Paula Veronese, L., & De Souza, A. F. “A simple yet effective obstacle avoider for the iara autonomous car. In 2016 IEEE 19th international conference on intelligent transportation systems (ITSC) (pp. 1914–1919). IEEE.”(2016).

Guidolini, R., De Souza, A. F., Mutz, F., & Badue, C. “Neural-based model predictive control for tackling steering delays of autonomous cars. In 2017 international joint conference on neural networks (IJCNN) (pp. 4324–4331). IEEE.”(2017).

Hancock, P.A. and Parasuraman, R. “Human Factors and safety in the design ofIntelligent Vehicle-Highway Systems (IVHS).”(1992).

Hata, A. Y., Ramos, F. T., & Wolf, D. F. “Monte carlo localization on gaussian process occupancy maps for urban environments. IEEE Transactions on Intelligent Transportation Systems, 19(9), 2893–2902.”(2017).

Hoedemaeker, M. “Needs, driving styles and opinions towards Automated Driving.”(1996).

Jain, K. “Fundamentals of digital image processing.”(1989).

Janssen, W.H., Wierda, M. Van der Horst, A.R.A. . *Automation of driving and userbehaviour*. Soesterberg, (1992).

Jurvetson, Steve. “Hands-free Driving.”(2009).

K. Ghazali, R. Xiao and J. Ma. “Road Lane Detection Using H-Maxima and Improved Hough Transform.”(2011).

Kalal, Z.; Mikolajczyk, K.; Matas, J. “ Tracking-learning-detection. IEEE Trans. Pattern Anal. Mach. Intell. 2012, 34, 1409–1422.”(2012).

Kenneth R. Castleman ve Prentice-Hall. “Digital Image Processing.”(1996).

Kim, Z. “Robust Lane Detection and Tracking in Challenging Scenarios.”(2008).

Klette, Reinhard. “Concise Computer Vision. Springer. ISBN 978-1-4471-6320-6.”(2014).

Kornhauser, A. “Smart Driving Cars, Mail Blog, http://smartdrivingcar.com/.”(2017).

L.Hal, Ernest. “Computer Image Processing And Recognition.”(1979).

Lamon, P., S. Kolski and R. Siegwart. *A vehicle for fully autonomous navigation and mapping in outdoor environments paper presented at the 9th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*. Brüksel, (2006).

Linda G. Shapiro; George C. Stockman. “Computer Vision. Prentice Hall.”(2001).

M. Mansourpour, M.A. Rajabi, J.A.R. Blais. “EFFECTS AND PERFORMANCE OF SPECKLE NOISE REDUCTION FILTERS ON.”(2006).

Massera Filho, C., Wolf, D. F., Grassi, V., & Osório, F. S. “Longitudinal and lateral control for autonomous ground vehicles. In 2014 IEEE intelligent vehicles symposium proceedings (pp. 588–593). IEEE.”(2014).

Michon, J.A. *Generic Intelligent Driver Support*. Londra, (1993).

Morris, Tim. “Computer Vision and Image Processing. Palgrave Macmillan.”(2004).

Mui, C. “Fasten your seatbelts: Google’s driverless car is worth trillions (Part 1).”(2013).

Mutz, F., Veronese, L. P., Oliveira-Santos, T., De Aguiar, E., Cheein, F. A. A., & De Souza, A. F. . “Large-scale mapping in complex field scenarios using an autonomous car. Expert Systems with Applications, 46, 439–462.”(2016).

N Dalal ve B Triggs. “Histograms of oriented gradients for human detection.”(2005).

Nagaoka, S.; Raihanul, I.; Okajima, K.; Ito, R.; Watanabe, K.; Ito, T. “Evaluation of Attenuation of Ultrasonic Wave in Air to Measure Concrete Roughness Using Aerial Ultrasonic Sensor. Int. J. GEOMATE 2018, 4, 158–163.”(2018).

National Highway Traffic Safety Administration (NHTSA). (2013).

Nilsson, L. & Alm, H. “Collision Avoidance Systems -Effects of different levels of taskallocation on driver behaviour.”(1991).

O.O. Khalifa ve A.H.A Hashim. “Vision-Based Lane Detection for Autonomous Artificial Intelligent Vehicles.”(2009).

Ouwerkerk, F. “Relationships between Road Transport Working Conditions,Fatigue, Health and Traffic Safety.”(1986).

Papert, Seymour. “The Summer Vision Project.”(1966).

P. Viola ve M. Jones. “Rapid object detection using a boosted cascade of simple features.”(2001).

Probst, T. “Die Benutzung (teil-)autonomer Motorfahrzeuge im Strassenverkehr aus haftpflichtrechtlicher Sicht, in T. Probst and F. Werro (eds.) Strassenverkehrsrechts.”(2016).

Pulli, Kari; Baksheev, Anatoly; Kornyakov, Kirill; Eruhimov, Victor. “Realtime Computer Vision with OpenCV.”(2012).

R. C. Gonzalez ve R. E. Woods. “Digital image processing.”(1992).

Rao, K.M.M. “OVERVIEW OF IMAGE PROCESSING" Readings in Image Processing Fundamentals Of Digital Image Processing - Anil K.”(1989).

Raymann, L. *Kapazität und Leistung versus Umwelt und Klima - Wirkungen des Einsatzes automatischer Fahrzeuge im Sinne einer Nachhaltigen Mobilität, Technical Report*. Zürih, (2016).

Rosebrock, Adrian. “An photo of a stop sign with two bounding boxes: ground-truth and prediction.”(2016).

Rosengren, L.G. *Driver assistance and co-operative driving.* Londra, (1995).

S. C. Pei, Y. C. Zeng, and C. H. Chang. “Virtual restoration of ancient Chinese paintings using color contrast enhancement and lacuna texture synthesis.”(2004).

S. Srivastava, R. Singal and M. Lumb. “Efficient Lane Detection Algorithm using Different Filtering Techniques.”(2014).

S.–D. Chen, and A. R. Ramli. “Contrast enhancement using recursive mean-separate histogram equalization for scalable brightness preservation.”(2003).

Sabbagh, V. B., Freitas, E. J., Castro, G. M., Santos, M. M., Baleeiro, M. F., da Silva, T. “Desenvolvimento de um sistema de controle para um carro de passeio autônomo. In XVIII congresso brasileiro de automática.”(2010).

SAE International. “Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, Warrendale, Sep. 2016, https://doi.org/10.4271/J3016_201609.”(2016).

“Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems, Warrendale, Jan.”(2014).

Schneider, F. E., & Wildermuth, D. “Results of the european land robot trial and their usability for benchmarking outdoor robot systems. In Conference towards autonomous robotic systems (pp. 408–409). Springer.”(2011).

Sejnowski, Terrence J. *The deep learning revolution.* Cambridge, Massachusetts London, England, (2018).

Shinzato, P. Y., dos Santos, T. C., Rosero, L. A., Ridel, D. A., Massera, C. M., Alencar, F.,. “Carina dataset: An emerging-country urban scenario benchmark for road detection systems. In 2016 IEEE 19th international conference on intelligent transportation systems (ITSC) (pp.41–46). IEEE.”(2016).

Singh, Sanjay Kumar. “Road traffic accidents in India: issues and challenges.”(2017).

Smiley, A., Brookhuis, K. A. “Alcohol, drugs and traffic safety. road users and traffic safety.”(1987).

SparkFun. “Autonomous vehicle competition.”(2018).

Stroustrup, Bjarne. “Bjarne Stroustrup's FAQ: When was C++ invented?”(2010).

“Evolving a language in and for the real world: C++.”(1991).

Szeliski, Richard. “Computer Vision: Algorithms and Applications.”(2010).

Tesla. “Tesla Autopilot.”(2017).

The Daily Ardmoreite. *The first remote-controlled vehicle*. USA, (1921).

The Houdina Radio Car. “American Wonder.”(1925).

The Milwaukee Sentinel. “'Phantom Auto' will tour city.”(1926).

Thorpe, C., Herbert, M., Kanade, T., & Shafter, S. “Toward autonomous driving:the cmu navlab. ii. architecture and systems. IEEE Expert, 6(4), 44–52.”(1991).

Türkiye İstatistik Kurumu. “Karayolu Trafik Kazaları.”(2022).

Vatavu, A.; Danescu, R.; Nedevschi, S. “Stereovision-based multiple object tracking in traffic scenarios using free-form obstacle delimiters and particle filters. IEEE Trans. Intell. Transp. Syst. 2015, 16, 498–511.”(2015).

Williams, Al. “Microcontroller projects using the Basic Stamp.”(2002).

Wojtanowski, J.; Zygmunt, M.; Kaszczuk, M.; Mierczyk, Z.; Muzal, M. “Comparison of 905 nm and 1550 nm semiconductor laser rangefinders’ performance deterioration due to adverse environmental conditions.”(2014).

Xin, J., Wang, C., Zhang, Z., & Zheng, N. “China future challenge: Beyond the intelligent vehicle. IEEE Intelligent Transportation Systems Society Newsletter, 16(2), 8-10.”(2014).

Z Zou, K Chen, Z Shi, Y Guo, J Ye. “Object detection in 20 years: A survey.”(2023).

EKLER

11. EKLER

```
#include <opencv2/opencv.hpp>

#include <raspicam_cv.h>

#include <iostream>

#include <wiringPi.h>

#include <thread>

#include <chrono>

using namespace std;

using namespace cv;

using namespace raspicam;

Mat goruntu, goruntu_kopya, hsv_goruntu, hsv_goruntu2, kirmizi_maske, yesil_maske, turuncu_maske, Matrix, Pers_goruntu, esik_goruntu;

Mat kenar_goruntu, final_goruntu, final_goruntu_kopya, final_goruntu_kopya2, serit_ilgi_alani, ilgi_alani_serit_sonu, gri_goruntu;

int sol_serit_pozisyonu, sag_serit_pozisyonu, merkez_goruntu, serit_merkezi, Sonuc, yol_sonu, yaya_mesafe, park_mesafe;

int hemzemin_mesafe, kirmizi_piksel_sayisi, turuncu_piksel_sayisi, yesil_piksel_sayisi;

int zemin_dusuk_deger = 100, zemin_yuksek_deger = 255;

int hue_min = 110, hue_max = 179, saturation_min = 138, saturation_max = 255, value_min = 78, value_max = 225;

RaspiCam_Cv Camera;
```

```
stringstream ss;

vector<int> histogram_yol;

vector<int> histogram_yol_sonu;

Point2f Kaynak[] = {Point2f(45, 160),Point2f(365, 160),Point2f(0,200), Point2f(400,200)};

Point2f Hedef[] = {Point2f(100,0),Point2f(280,0),Point2f(100,240), Point2f(280,240)};

//Makine Öğrenimi değişkenleri

CascadeClassifier yaya_Cascade, hemzemin_Cascade, park_Cascade;

Mat goruntu_yaya, yaya_ilgi_alani, gri_yaya, goruntu_hemzemin, hemzemin_ilgi_alani, hemzemin_gri, goruntu_park, ilgi_alani_park, gri_park;

vector<Rect> yaya;

vector<Rect> hemzemin;

vector<Rect> park;

void on_trackbar(int, void*){}// Bu fonksiyon, bir trackbar değeri her değiştiğinde çağrılacaktır

void Kamera_Kurulum ( int argc,char **argv, RaspiCam_Cv &Camera ){

   Camera.set ( CAP_PROP_FRAME_WIDTH,  ( "-w",argc,argv,400 ) );

   Camera.set ( CAP_PROP_FRAME_HEIGHT,  ( "-h",argc,argv,240 ) );

   Camera.set ( CAP_PROP_BRIGHTNESS, ( "-br",argc,argv,50 ) );

   Camera.set ( CAP_PROP_CONTRAST ,( "-co",argc,argv,50 ) );

   Camera.set ( CAP_PROP_SATURATION,  ( "-sa",argc,argv,50 ) );
```

```
    Camera.set ( CAP_PROP_GAIN,  ( "-g",argc,argv ,50 ) );
    Camera.set ( CAP_PROP_FPS,  ( "-fps",argc,argv,0));}
void Kamera_Yakalama(){
    Camera.grab();
    Camera.retrieve(goruntu);
    goruntu_kopya = goruntu.clone();
    cvtColor(goruntu, goruntu, COLOR_BGR2RGB);
    cvtColor(goruntu, goruntu_yaya, COLOR_BGR2RGB);
    cvtColor(goruntu, goruntu_hemzemin, COLOR_BGR2RGB);
    cvtColor(goruntu, goruntu_park, COLOR_BGR2RGB);}
void Perspektif(){
        line(goruntu,Kaynak[0], Kaynak[1], Scalar(0,0,255), 2);
        line(goruntu,Kaynak[1], Kaynak[3], Scalar(0,0,255), 2);
        line(goruntu,Kaynak[3], Kaynak[2], Scalar(0,0,255), 2);
        line(goruntu,Kaynak[2], Kaynak[0], Scalar(0,0,255), 2);
        Matrix = getPerspectiveTransform(Kaynak, Hedef);
        warpPerspective(goruntu, Pers_goruntu, Matrix, Size(400,240));}
void Esik(){
        cvtColor(Pers_goruntu, gri_goruntu, COLOR_RGB2GRAY);
        inRange(gri_goruntu,                    Scalar(zemin_dusuk_deger),
Scalar(zemin_yuksek_deger), esik_goruntu);
```

```
        Canny(gri_goruntu,kenar_goruntu, 900, 900, 3, false);

        add(esik_goruntu, kenar_goruntu, final_goruntu);

        cvtColor(final_goruntu, final_goruntu, COLOR_GRAY2RGB);

        cvtColor(final_goruntu, final_goruntu_kopya, COLOR_RGB2BGR);
//yalnızca Histogram işlevinde kullanılır

        cvtColor(final_goruntu, final_goruntu_kopya2, COLOR_RGB2BGR);}
//yalnızca Histogram işlevinde kullanılır

void Histogram(){

   histogram_yol.resize(400);

   histogram_yol.clear();

   for(int i=0; i<400; i++){

        serit_ilgi_alani =
final_goruntu_kopya(Rect(i,140,1,100));//goruntu.size().width = 400

        divide(255, serit_ilgi_alani, serit_ilgi_alani);

        histogram_yol.push_back((int)(sum(serit_ilgi_alani)[0]));}

   histogram_yol_sonu.resize(400);

   histogram_yol_sonu.clear();

   for (int i = 0; i < 400; i++){

            ilgi_alani_serit_sonu = final_goruntu_kopya2(Rect(i, 0, 1, 240));

            divide(255, ilgi_alani_serit_sonu, ilgi_alani_serit_sonu);

   histogram_yol_sonu.push_back((int)(sum(ilgi_alani_serit_sonu)[0]));}
```

```
      yol_sonu = sum(histogram_yol_sonu)[0];}

void Yol_Bulucu(){

   vector<int>:: iterator LeftPtr;

  LeftPtr = max_element(histogram_yol.begin(), histogram_yol.begin() + 150);

   sol_serit_pozisyonu = distance(histogram_yol.begin(), LeftPtr);

   vector<int>:: iterator RightPtr;

   RightPtr = max_element(histogram_yol.begin() +250, histogram_yol.end());

   sag_serit_pozisyonu = distance(histogram_yol.begin(), RightPtr);

   line(final_goruntu, Point2f(sol_serit_pozisyonu, 0), Point2f(sol_serit_pozisyonu,
240), Scalar(0, 255,0), 2);

   line(final_goruntu, Point2f(sag_serit_pozisyonu, 0), Point2f(sag_serit_pozisyonu,
240), Scalar(0,255,0), 2);}

void Cizgi_Merkezi(){

   serit_merkezi = (sag_serit_pozisyonu-sol_serit_pozisyonu)/2
+sol_serit_pozisyonu;

   merkez_goruntu = 188;

   line(final_goruntu, Point2f(serit_merkezi,0), Point2f(serit_merkezi,240),
Scalar(0,255,0), 3);

   line(final_goruntu, Point2f(merkez_goruntu,0), Point2f(merkez_goruntu,240),
Scalar(255,0,0), 3);

   Sonuc = serit_merkezi-merkez_goruntu;}

void yaya_tespit(){
```

```
if(!yaya_Cascade.load("//home//pi//Downloads//yaya_cascade.xml")){
        printf("Unable to open yaya cascade file");}
yaya_ilgi_alani = goruntu_yaya(Rect(0,0,400,240));
cvtColor(yaya_ilgi_alani, gri_yaya, COLOR_RGB2GRAY);
equalizeHist(gri_yaya, gri_yaya);
yaya_Cascade.detectMultiScale(gri_yaya, yaya);// Birden fazla yaya geçidi tespit edilip edilmediğini kontrol et
if (yaya.size() > 1) {
        int max_area_index = 0;
        int max_area = yaya[0].width * yaya[0].height;
        for (int i = 1; i < yaya.size(); i++) {
           int area = yaya[i].width * yaya[i].height;
           if (area > max_area) {
                max_area = area;
                max_area_index = i;}}
        yaya = vector<Rect>(1, yaya[max_area_index]);}
if (!yaya.empty()) {
   Point P1(yaya[0].x, yaya[0].y);// Dikdörtgen çiz ve en büyük yaya geçidi için mesafeyi hesapla
   Point P2(yaya[0].x + yaya[0].width, yaya[0].y + yaya[0].height);
   rectangle(yaya_ilgi_alani, P1, P2, Scalar(0, 0, 255), 2);
```

```
    putText(yaya_ilgi_alani, "yaya", P1, FONT_HERSHEY_PLAIN, 1,  Scalar(0, 0, 255, 255), 2);

    yaya_mesafe = (-0.318)*(P2.x-P1.x) + 46.164;

    ss.str(" ");

    ss.clear();

    ss<<"D = "<<yaya_mesafe<<"cm";

    putText(yaya_ilgi_alani, ss.str(), Point2f(1,130), 0,1, Scalar(0,0,255), 2);}}

void hemzemin_tespit(){

if(!hemzemin_Cascade.load("//home//pi//Downloads//hemzemin_cascade2.xml")){

        printf("hemzemin cascade dosyası açılamıyor");}

    hemzemin_ilgi_alani = goruntu_hemzemin(Rect(0,0,400,240));

    cvtColor(hemzemin_ilgi_alani, hemzemin_gri, COLOR_RGB2GRAY);

    equalizeHist(hemzemin_gri, hemzemin_gri);

    hemzemin_Cascade.detectMultiScale(hemzemin_gri, hemzemin);

    for(int i=0; i<hemzemin.size(); i++){

        Point P1(hemzemin[i].x, hemzemin[i].y);

        Point P2(hemzemin[i].x + hemzemin[i].width, hemzemin[i].y + hemzemin[i].height);

        rectangle(hemzemin_ilgi_alani, P1, P2, Scalar(0, 0, 255), 2);

        putText(hemzemin_ilgi_alani, "hemzemin", P1, FONT_HERSHEY_PLAIN, 1,  Scalar(0, 0, 255, 255), 2);
```

```
            hemzemin_mesafe = (-0.85)*(P2.x-P1.x) + 113,3;
            ss.str(" ");
        ss.clear();
        ss<<"D = "<<hemzemin_mesafe<<"cm";
        putText(hemzemin_ilgi_alani, ss.str(), Point2f(1,130), 0,1, Scalar(0,0,255), 2);}}

void kirmizi_tespit()
{     Scalar dusuk_kirmizi(82, 98, 143), yuksek_kirmizi(141, 250, 255);
        inRange(hsv_goruntu2, dusuk_kirmizi, yuksek_kirmizi, kirmizi_maske);
        kirmizi_piksel_sayisi = countNonZero(kirmizi_maske);}
void yesil_tespit()
{     Scalar dusuk_yesil(30, 10, 236);
        Scalar yuksek_yesil(89, 115, 255);
        inRange(hsv_goruntu2, dusuk_yesil, yuksek_yesil, yesil_maske);
        yesil_piksel_sayisi = countNonZero(yesil_maske);}
void park_levha(){
    if(!park_Cascade.load("//home//pi//Downloads//park_cascade2.xml")){
            printf("Park cascade dosyası acilamiyor");}
    ilgi_alani_park = goruntu_park(Rect(0,0,400,240));
    cvtColor(ilgi_alani_park, gri_park, COLOR_RGB2GRAY);
    equalizeHist(gri_park, gri_park);
```

```
    park_Cascade.detectMultiScale(gri_park, park);
    for(int i=0; i<park.size(); i++){
        Point P1(park[i].x, park[i].y);
        Point P2(park[i].x + park[i].width, park[i].y + park[i].height);
        rectangle(ilgi_alani_park, P1, P2, Scalar(0, 0, 255), 2);
        putText(ilgi_alani_park, "park", P1, FONT_HERSHEY_PLAIN, 1,  Scalar(0, 0, 255, 255), 2);
        park_mesafe = (-1.46)*(P2.x-P1.x) + 159,42;
        ss.str(" ");
      ss.clear();
      ss<<"D = "<<park_mesafe<<"cm";
      putText(ilgi_alani_park, ss.str(), Point2f(1,130), 0,1, Scalar(0,0,255), 2);}}
int main(int argc,char **argv){
    bool kosul = false;//zaman >=180 saniye olduğunda ve diğer koşullar sağlandığında park görevini başlatır
    bool deci10_yolla = false;
    bool kosul_yesil = false;
    bool kosul_turuncu= false;
    bool harekete_basla = false;
    bool kosul_yesil2= false;
    wiringPiSetup();
```

```
    pinMode(21, OUTPUT);

    pinMode(22, OUTPUT);

    pinMode(23, OUTPUT);

    pinMode(24, OUTPUT);

    Kamera_Kurulum(argc, argv, Camera);

    cout<<"Kameraya baglaniliyor"<<endl;

    if (!Camera.open()){

        cout<<"Baglanma basarisiz"<<endl;}

    cout<<"Camera Id = "<<Camera.getId()<<endl;

    namedWindow("Trackbars");

    createTrackbar("zemin_dusuk_deger", "Trackbars", &zemin_dusuk_deger,
zemin_yuksek_deger, NULL);

    createTrackbar("zemin_yuksek_deger", "Trackbars", &zemin_yuksek_deger,
zemin_yuksek_deger, NULL);

    auto startTime = chrono::steady_clock::now();// park icin

    auto start_time = std::chrono::high_resolution_clock::now();

    while(!kosul){

        auto suanki_zaman = chrono::steady_clock::now();

      auto gecen_zaman = chrono::duration_cast<chrono::seconds>(suanki_zaman -
startTime).count();

        Kamera_Yakalama();
```

```
    Perspektif();
    Esik();
    Histogram();
    Yol_Bulucu();
    Cizgi_Merkezi();
    yaya_tespit();
    hemzemin_tespit();
    //kirmizi_tespit();
    //yesil_tespit();
    park_levha();
    //cout<<"Kirmizi: = "<<kirmizi_piksel_sayisi<<endl;
    //cout<<"Yesil: = "<<yesil_piksel_sayisi<<endl;
    //trafik_bas
    cout << "Gecen Zaman: " << gecen_zaman << endl;
    //if (!harekete_basla) {
    //// Trafik ışığı yeşil yanıyorsa, kırmızıya dönmesini bekle
//if (kosul_yesil) {
    //if (kirmizi_piksel_sayisi > 1000) {
            //digitalWrite(21, 1);
            //digitalWrite(22, 1);    //decimal = 11
```

```
            //digitalWrite(23, 0);

            //digitalWrite(24, 1);

        //std::cout << "Trafik ışığı kırmızı" << std::endl;

        //kosul_yesil2 = true;}

    //else {

        //std::cout << "Trafik ışığının kırmızıya dönmesini bekleniyor" << std::endl;

            //digitalWrite(21, 1);

            //digitalWrite(22, 1);    //decimal = 11

            //digitalWrite(23, 0);

            //digitalWrite(24, 1);}}

// Trafik ışığı kırmızıysa yeşile dönmesini bekle

//else if (!kosul_yesil) {

    //if (yesil_piksel_sayisi > 1000) {

        //std::cout << "Trafik ışığı yeşil" << std::endl;

        //digitalWrite(21, 1);

            //digitalWrite(22, 1);    //decimal = 11

            //digitalWrite(23, 0);

            //digitalWrite(24, 1);

            //kosul_yesil = true;   //}

    //else {
```

```
            //std::cout << "Trafik ışığının yeşile dönmesini bekleniyor" << std::endl;
                  //digitalWrite(21, 1);
                  //digitalWrite(22, 1);    //decimal = 11
                  //digitalWrite(23, 0);
                  //digitalWrite(24, 1);   //}  //}
        //if (kosul_yesil2 && yesil_piksel_sayisi >1000){
            //harekete_basla = true;    //}    //}
//trafik_son
if (gecen_zaman <40){
            digitalWrite(21, 1);
            digitalWrite(22, 1);    //decimal = 3
            digitalWrite(23, 0);
            digitalWrite(24, 1); }
else if (gecen_zaman >40){ //trafik calismazsa harekete_basla yerine 1 yaz
        if (yol_sonu >= 30000 || Sonuc == -63){
            digitalWrite(21, 1);
            digitalWrite(22, 1);    //decimal = 3
            digitalWrite(23, 0);
            digitalWrite(24, 0);
            cout<<"yol sonu"<<endl; }
```

```
else if (Sonuc >=-8 && Sonuc <= 8){
   digitalWrite(21, 0);
   digitalWrite(22, 0);    //decimal = 0
   digitalWrite(23, 0);
   digitalWrite(24, 0);
   cout<<"ileri"<<endl; }
else if (Sonuc >8 && Sonuc <=63){
   digitalWrite(21, 1);
   digitalWrite(22, 0);    //decimal = 1
   digitalWrite(23, 0);
   digitalWrite(24, 0);
   cout<<"Sag"<<endl; }
else if (Sonuc <-8 && Sonuc >=-62){
   digitalWrite(21, 0);
   digitalWrite(22, 1);    //decimal = 2
   digitalWrite(23, 0);
   digitalWrite(24, 0);
   cout<<"Sol"<<endl; }
if (yaya_mesafe> 10 && yaya_mesafe < 40){
   digitalWrite(21, 1);
```

```
                digitalWrite(22, 0);    //decimal = 5
                digitalWrite(23, 1);
                digitalWrite(24, 0);
                cout<<"yaya"<<endl;
                yaya_mesafe = 0;}
              if (hemzemin_mesafe > 1 && hemzemin_mesafe < 1000){
                digitalWrite(21, 0);
                digitalWrite(22, 1);    //decimal = 6
                digitalWrite(23, 1);
                digitalWrite(24, 0);
                cout<<"hemzemin"<<endl;
                hemzemin_mesafe = 0; } }
//if (yol_sonu > 30000){
    //ss.str(" ");
    //ss.clear();
    //ss<<" Lane End";
    //putText(goruntu, ss.str(), Point2f(1,50), 0,1, Scalar(255,0,0), 2);}
  //else if (Sonuc == 0){
    //ss.str(" ");
    //ss.clear();
```

```
   //ss<<"Sonuc = "<<Sonuc<<" ileri git";
   //putText(goruntu, ss.str(), Point2f(1,50), 0,1, Scalar(0,0,255), 2);}
//else if (Sonuc > 0){
   //ss.str(" ");
   //ss.clear();
   //ss<<"Sonuc = "<<Sonuc<<" saga git";
   //putText(goruntu, ss.str(), Point2f(1,50), 0,1, Scalar(0,0,255), 2);}
//else if (Sonuc < 0){
    //ss.str(" ");
    //ss.clear();
    //ss<<"Sonuc = "<<Sonuc<<" sola git";
    //putText(goruntu, ss.str(), Point2f(1,50), 0,1, Scalar(0,0,255), 2);}
waitKey(1);
if (park_mesafe > 20 && park_mesafe < 100 && gecen_zaman >= 600)
{           digitalWrite(21, 1);
             digitalWrite(22, 0);    //decimal = 9
             digitalWrite(23, 0);
             digitalWrite(24, 1);
             cout<<"5 SANIYE BOYUNCA DUR"<<endl;
             kosul = true;
```

```
                cv::destroyAllWindows();

                Camera.release(); // Kamerayı serbest bırakır

                cout<<"KOSUL DOGRU"<<endl;   }}
```

// Koşul sağlandığında, sistem çağrılarını kullanarak park_test0.4.cpp dosyasını derle ve çalıştır

```
const char* kaynakDosya = "/home/pi/Desktop/park_test0.4.cpp";

const char* ciktiDosyasi = "park_test0.4";
```

// OpenCV include ve kütüphane dizinlerini belirt ve OpenCV kütüphanelerine bağla

```
const char* opencv_bayrak = "`pkg-config --cflags --libs opencv4`"; // OpenCV bayraklarını almak için pkg-config kullanın
```

// raspicam include ve kütüphane dizinlerini belirtir ve raspicam kütüphanelerine bağlar

```
const char* raspicam_bayrak = "-I/usr/local/include/raspicam -L/usr/local/lib -lraspicam -lraspicam_cv";

const char* wiringpi_bayrak = "-I/usr/local/include/ -L/usr/local/lib -lwiringPi -lwiringPiDev";

std::string komutuDerle = "g++ " + std::string(kaynakDosya) + " -o " + std::string(ciktiDosyasi) + " " + opencv_bayrak + " " + raspicam_bayrak + " " + wiringpi_bayrak;

int derleme_sonucu = std::system(komutuDerle.c_str()); // park_test0.4.cpp dosyasını derler

if (derleme_sonucu == 0) {
```

```
    int runResult = std::system(("./" + std::string(ciktiDosyasi)).c_str()); // Derlenmiş park_test0.4 dosyasını çalıştırır

    if (runResult == 0) {

        std::cout << "park_test0.4.cpp başarıyla derlendi ve çalıştırıldı." << std::endl;

    } else {

        std::cerr << "Yürütme hatası" << ciktiDosyasi << "." << std::endl;}

} else {

    std::cerr << "Derleme hatası" << kaynakDosya << "." << std::endl;}

Camera.release(); // Kamerayı serbest bırakır

if (!kosul)

{

        return 0; }  }
```