T.C
SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

OTONOM ARAÇ

Bitirme Çalışması

B150918037 Osman SARIGÖZ

Danışmanı : Öğr. Gör. Dr. Gökhan ATALI

Mayıs 2019

T.C
SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

OTONOM ARAÇ

Bitirme Çalışması

B150918037 Osman SARIGÖZ

Danışmanı : Öğr. Gör. Dr. Gökhan ATALI

Bu rapor 27.05.2019 tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

Öğr. Gör. Dr.	**Prof. Dr.**	**Doç. Dr.**
Gökhan ATALI	**Osman ELDOĞAN**	**Faruk YALÇIN**
Danışman	**Üye**	**Üye**

BEYAN

Rapor içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, raporda yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir çalışmada kullanılmadığını beyan ederim.

Osman SARIGÖZ

27.05.2019

TEŞEKKÜR

Lisans eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Öğr. Gör. Dr. Gökhan ATALI 'ya teşekkürlerimi sunarım.

Ayrıca lisans eğitimim boyunca Sakarya Üniversitesi Teknoloji Fakültesinde görev yapmakta olan tüm hocalarıma verdiği eğitimlerden dolayı teşekkür ederim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

A	:Sistem matrisi
B	:Giriş matrisi
C	:Çıkış matrisi
D	:İleri besleme matrisi
DC	:Doğru akım
EMK	:Elektro motor kuvvet
EKM	:Elektronik kontrol ünitesi
RC	:Radio controlled
TF	:Transfer fonksiyonu
UZM	:Uzay zaman modeli
RC	:Radio controlled
$\dot{x}_{D1}$	:Aracın lineer yol alırken durum uzay modeli matrisi
$\dot{x}_{D2}$	:Aracın direksiyon için durum uzay modeli matrisi
y_{C1}	:Aracın lineer yol alırken çıkış vektörü matrisi
y_{C2}	:Aracın direksiyon için çıkış vektörü matrisi

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar sözcükler: Görüntü işleme, tabela tanımlama, trafik lambası tanımları, yörünge takibi, otonom araç

Bir arabanın, otonom araç olabilmesi için, aracın kamera görüntüleri ile görüntü işleme yöntemlerini kullanarak şerit takibi, tabela tanımlama, trafik ışıklarını algılayan ve kırmızı renk algılamada aracın durması yeşil renkte aracın harekete geçmesi beklenen bir sisteminin oluşturulması hedeflenmiştir.

Aracımız şerit takibi yaparken sağ şerit ile sol şeridin eğimi pozitif ve negatif değerler aldığından aracımız şeritleri böyle tanımaktadır. Aracımız her iki şeridi algıladığın da araç düz ilerlemektedir. Eğer şeritlerin her ikisinin de eğimleri negatif eğimli ise araç sağa dönmektedir. Eğer pozitif eğimli ise de aracımız sola dönmektedir.

Aracımız şerit takibi yaparken mesafe sensörü engel algılamaktadır. Böylece tabela algılama ile trafik lambasını algılamak daha kolay olmaktadır. Çünkü aracımızın şerit takibi yaparken önünde bir cisim belli bir mesafede algılanınca durmaktadır. Tabela algılama ve trafik lambalarının algılanması daha da kolay olmaktadır. Aracımız durgun halde iken raspberry pi görüntü işlemede daha iyi sonuçlar vermiştir.

Aracımız tabela algılamada mavi renklidir. Mavi renkli bir yüzeyi tarayıp alan sınırları algılanmaktadır. Alan sınırları algılandıktan sonra mavi yüzeyin içinde bulunan işareti bazı kabuller yapılarak matematiksel yöntemlerle ele alınıp, tabelanın sağa dönmek, sola dönmek, ileri gitmek, geri dönme algılanma yapılmaktadır. Tabela algılandıktan sonra şerit seçimi yapmaktadır. Böylece aracımız tabela algılanıp, istenilen şeritte döndürüldükten sonra araç normal şerit takibine devam etmektedir.

Bu çalışmanın amacı görüntü işleme yöntemleri ile şerit takibi, trafik tabelasını algılanmasını, trafik lambaları için renk filtreleme ve görüntü işleme yönteminden faydalanılacaktır. Bu tasarımın, herhangi yoldaki bir şeritte hareket ederken dışardan bağımsız kararlar alarak, trafik ışıklarına uyan, tabela tanımlarına göre hareket eden tasarım hedeflenmiştir.

SUMMARY

Keywords : Image processing, signage identification, traffic light definitions, trajectory tracking, autonomous vehicle

In order for a car to be an autonomous vehicle, it is aimed to create a system that is expected to start the vehicle by using the vehicle image and image processing methods.

When our vehicle is following the lane, the right lane and the sliver of the left lane have positive and negative values, so our vehicle recognizes the lanes. The vehicle is moving straight when you detect both lanes. If the slopes of both strips are negative, the vehicle rotates to the right. If it has a positive inclination, our vehicle turns left.

The distance sensor detects an obstacle while our vehicle is following a lane. Thus, it is easier to detect the traffic light with signage detection. Because our vehicle lane tracking when an object is detected at a certain distance stops. Signage detection and traffic lights are even easier to detect. Raspberry pi gave better results when processing.

Our vehicle is blue in signage detection. The area boundaries are detected by scanning a blue surface. After the boundaries of the field are detected, the sign in the blue surface is considered by some mathematical methods and the right of the sign, turning left, going forward, turning back is perceived. After the signage is detected, the ribbon is selected. Thus, after the vehicle is detected and rotated in the desired lane, the vehicle continues to follow normal lane follow-up.

The aim of this study will be to follow the methods of image processing, strip tracking, traffic sign detection, color filtering for traffic lights and image processing. This design is targeted at the design of the signage that adapts to traffic lights and takes externally independent decisions while moving in a lane on any path.

BÖLÜM 1. GİRİŞ

Bir aracın otonom yol izleyebilmesi için görüntü işleme yöntemleri ile tasarımı amaçlanmıştır. Otonom aracın şerit algılaması ve bu iki şeridin eğimlerinden faydalanarak sağ ve sol şerit olarak tanımlanması yapılmıştır. Trafik lambasını algılamada renklerin belli bir aralıkta değerler verilerek kırmızı ve yeşilin algılanması yapılmıştır. Trafik tabela tanımlama ise mavi bir yüzey alanında ve o yüzey alanındaki içinde bulunan işaretleri tanıyan ve bu işlemleri görüntü işleme yöntemleri ile yapan böylece aracın otonom olması hedeflenmiştir. Günümüzde bilim ve teknoloji, her insan için vazgeçilmez seçenekler sunmaktadır. Bu seçenekler arasında otonom sistemlerde yer almaktadır. Otonom sistemlerinde önemli yeri olan otonom araçlarda yer almaktadır. Yapılan bu çalışmada uzun yolculuklarda şoförlerin bir anlık dikkatsizliği nedeniyle otonom aracın müdahalede bulunması veya aracı otonom bir şekilde kendi kendini sürebilmesi ile insanların can ve mal güvenliğini sağlamak için geliştirilmeye çalıştırılmıştır. Aracımızın kararlı bir şekilde otonom olması, işlemcinin iyi olması gerekmektedir. İşlemci ani durumlarda ani kararlar alması için bu önemli bir husustur. Yapılan bu çalışmada raspberry pi'nin işletim sistemi kullanılmıştır. Raspberry pi yüksek çözünürlükte görüntü işlemede geç yanıt vermektedir. Düşük çözünürlükte görüntü işleme yapılmada daha başarılı kararlar algıladığı gözlemlenmiştir.

BÖLÜM 2. KAYNAK ARAŞTIRMASI

Otonom araç için yapılan araştırmalar, bir aracın otonom olabilmesi için çevresinden yeterince bilgi alması ile olur. Çevreden alınan bilgiler ile kaynak araştırması aşağıdaki gibidir.

2.1. Otonom Araç

Otonom araçlar günümüzde her geçen gün önemi artmaktadır. Bu alanda çeşitli çalışmalar yer almaktadır. Anlık alınan verilere göre, aracın otonom olması için sistemin hızlı karar alması alanında yapılan çalışmalar bulunmaktadır. Bunlar; çevrenin algılanması, görüntü işleme yöntemi önemli yerlere sahiptir. Otonom araçlar için yapay zeka ile yapay sinir ağları ile yapmak önemlidir.

Bir aracın otonom araç olabilmesi için bulunduğu çevreden bilgi sahibi olması gerekmektedir. Bunun için sensör ve kamera ile çevremizi algılarız. Sensörler sınırlı bilgi vermektedir. Bunun için gerekli şerit takibi veya çizgi izleme gibi çalışmalar yapılabilir ama aracın otonom bir şekilde yol izleme yapamaz. Trafik ışıklarını algılama veya tabela tanımlama yapamamaktadır. Bunun için kamera kullanmak çevreden daha iyi bilgi almamızı sağlar. Kamera ile görüntü işleme yöntemleri ile tabela tanımlama, trafik ışığını algılamada bize olanak sağlamaktadır.

2.1.1. Sensör ile kızıl ötesi ışınlar

Alpay'ın yaptığı çalışmada sensör kullanımı ile kızıl ötesi ışınlar ile siyah zemin üzerine beyaz şerit tespitinde bulunmuştur. Ultrasonik mesafe sensörü ile engelleri algılayan ve çarpışmaları önlemektedir. Renk algılama kartı ile siyah, beyaz, mavi, yeşil renkleri algılama ile hız kontrolü yapılmıştır. Bluetooth modülü ile android telefon veya tablet ile kontrol edilmektedir [1].

2.1.2. Görüntü işleme yöntemi

Görüntü işleme yöntemleri ile kamera üzerinde yapılan çalışmalar yaygındır. Hareket edilecek yoldaki şeritleri alınan görüntüyü filtreleyerek yapılan çalışmalar sıkça gözlemlenmiştir.

Mustafa'nın yapmış olduğu çalışmada CCD kamerasından aldığı görüntü ile önündeki aracı tanımlayan, uzaklığını hesaplayan ve hız ölçümü yapmaktadır. Kamera gelen görüntü ile Gaussian filtresi uygulayıp gri renk dönüşümü yapmıştır. Bu verilere görüntü eşikleme ve Canny kenar belirleme ile kenar algılanmıştır. Harris ve Stephens köşe belirleme ve Hough algoritması ile görüntülerdeki nesneleri dikdörtgen şeklini ayırmıştır. Trafik tabelası boyutlarına göre araç mesafe uzaklığını belirlenmiştir [2].

Ayhan ve ark.'nın 2014'de yaptığı çalışmada, güvenli yolculuk için şerit takip sistemleri üzerinde çalışmışlardır. Gömülü bir platform üzerine kamera ile gerçek zamanlı şerit algılayabilen sistem görüntüyü, görüntü işleme yöntemleri ile algoritmalar tasarlayarak şeritleri algılamaktadır. Görüntü işlemede şerit takibi için belirli bir alanda şerit takibini yapmışlardır. Dış etkenlerden daha az etkilenmek için görüntü işleme yöntemlerini kullanarak şerit algılama üzerinde durmuşlardır. Şerit algılamak için görüntü işleme ile renk filtresi uygulandıktan sonra çıkan görüntüye

Hough dönüşümü uygulanmıştır. Hough dönüşümü ile görüntüdeki tüm çizgileri tespit edip şerit sayılacak açıdaki çizgileri eşik değerine göre tespit etmektedir. Şeritleri algıladıktan sonra şerit değiştirmeyi şerit çizgilerine göre algılayarak uyarı verir [3].

Ayhan ve Oguzhan'nın 2017'de geliştirdikleri çalışmada ise önceki projelerini geliştirilmiştirler. Görüntü işleme filtrelerini değiştirip yenileri eklenerek daha iyi sonuçlar elde edilmiştir. Şerit filtresi sonucu oluşan görüntüye çizgi algılaması için Gauss korelasyonu uygulanmıştır. Dış etkenlerden (ışık ve titreşim gibi) etkilenmemesi için RANSAC ile çizgi uydurma uygulanarak çizgilerin algılanmasını kolaylaştırmışlardır. Şeritlerin algılanmasının güvenliğini arttırmak için Kalman filtresi uygulanarak şerit algılamada kesikler olsa bile şeritlerin konumunu tahmin ederek devamlılığı sağlar. Şerit değişikliği tespit edilince sürücüye uyarı verilir [4].

Ferhat tarafından yapılan çalışmada otonom araç tasarımında hareket edilecek yörüngeyi kamera ile algılama yapmaktadır. Ultrasonik mesafe sensörü engelleri algılama yapmıştır. Aracın belli bir bölgeyi inceleyerek hareket yönünü belirlemiştir. Kameradan aldığı görüntüleri, görüntü işleme yöntemleri ile HSV renk uzayı, erozyon ve daire bulma algoritmalarını uygulayarak işaret tabelalarını tanıması sağlanmıştır. Tabeladaki ifadeye göre araç yönünü ayarlamaktadır. Dışardan herhangi bir bilgi akışı olmadan yol şeritlerine gerek duymadan hareket edebilmektedir. Görüntü işlemeyi Raspberry pi kullanarak Python dilinde ve SimpleCV kütüphanesini kullanarak yapmıştır. Motor kontrollerini Arduino Mega ile sensörden ve Raspberry pi'den gelen veri ile sağlamıştır. Wifi ile Raspberry pi arasında bağlantı kurarak hareketleri gözlemlenebilmektedir [5].

2.1.3. Yapay sinir ağları

Kameradan alınan görüntüyü görüntü işleme yöntemleri dışında yapay sinir ağlarını kullanarak da işleyebiliriz. Çoğu alanda kullanılan diğer yöntemlere göre büyük başarılar sağlayan derin öğrenme otonom araçlarda da farkını göstermektedirler.

Yapay sinir ağlarını kullanan David yaptığı çalışmada ilk adım olarak otom sürüş için kullanılan yöntemleri inceleyerek içinden yapay zeka ile derin öğrenmeyi seçmiştir. Yöntemi belirledikten sonra aracın tasarımı ile çalışmalarına devam etmiştir. Donanımsal olarak kullanılacak motor, sürücü, kamera ve işletim sistemi gibi malzemelerin seçimini yapmıştır. Aracın tasarımı bitince kontrol algoritmalarını ve yöntemlerini belirlemiştir.

Aracın kontrolü için yapay sinir ağı yöntemlerini test etmiş ve en iyi sonuç vereni seçmiştir. Yapay sinir ağını tasarlayıp doğru sonuç verene kadar sinir ağını eğitmiştir. Sinir ağına kamera görüntülerini vererek yörünge kontrolünü yapmasını sağlamıştır [6].

Satyanarayana ve arkadaşlarının yaptığı çalışmada aracın makine öğrenme yöntemleri ile otonom bir araç tasarlamışlardır. Aracın L298 motor sürücü, ultrasonik mesafe sensörü ve pi kamera kontrol edebilmek için raspberry pi kullanmışlardır. Y:apılan bu çalışmada engel algılama, stop tabelasını tanıma, kırmızı ve yeşil ışıkları tanımlayan, yörünge takibi yapan bir RC araç içerisinde geliştirmişlerdir [7].

BÖLÜM 3. TASARIM KRİTERLERİ VE TERCİHLERİ

Otonom araç için kriterler, projenin oluşturması için gerekli koşullar, tercihler ise kullanılması istenilen birim veya malzemedir.

3.1. Kriterler

Tasarlanmak için araçların otonom biçiminde şerit takibi yapması hedeflenmiştir. Şerit takibi yaparken aracımızın tabela algılama, trafik ışıklarının algılanması içinde mesafe sensörü kullanmaktayız. Çünkü aracımız hareketli halde iken trafik ışığı ile tabela algılama bazen yapmamaktadır. Böylece aracımız önünde engel algıladığını varsayar aracımız durgun durumuna gelmektedir. Aracımız durgun halde iken raspberry pi görüntü işlemede daha doğru yanıtlar vermektedir. Tabela algılanınca istenilen yöne gitmektedir. Trafik ışığının ise kımızımda durmaktadır. Ama araç yeşil algılasa bile mesafenin giderek azaldığı bilgisi gelsede yeşil renk için araç hareketli hale gelmektedir. Aracın hareketleri otonom bir şekilde tabela tanımlama, trafik ışıklarına ve şerit takibi yaparak kendi tercihleri ile olmalıdır.

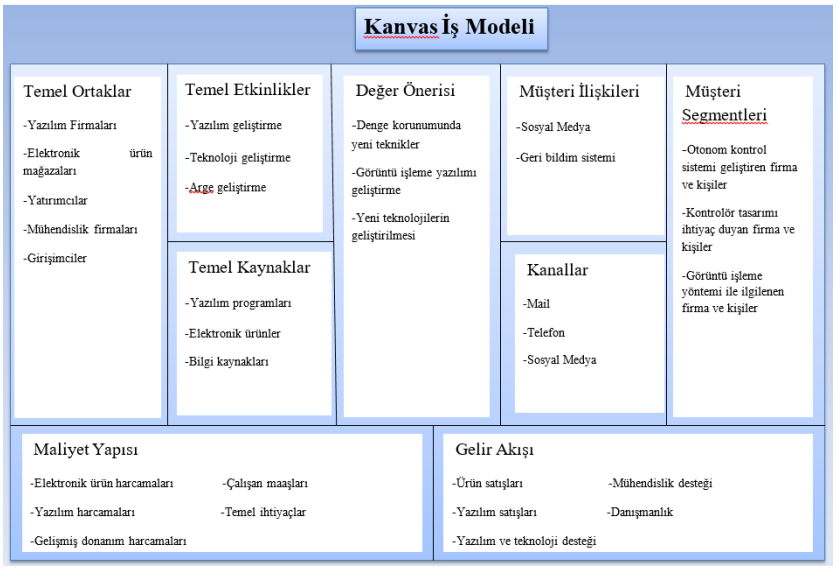
3.2. Tercihler

Otonom yol izlenmesi için şerit tesbit edilmesi gerekir. Belli bir şerit takibi, tabela tanımlama yaparak ve trafik ışıklarına algılayarak kendi içinde karar veren bir sistem amaçlanmıştır. Yol algılama için sensörler bulunsa da, trafik tabelası ve trafik lambası için yetersiz görülmektedir. Kamera kullanımı yaygın ve veri ihtiyacını karşılayan en iyi yöntemdir. Kameradan algılanan görüntüler işlenmesi için yüksek bir işlem gücü gereklidir. Görüntü işleme için kullanımı yaygın olan bir mini bilgisayar olan Raspberry pi uygun tercihtir. Kendisine ait kaliteli kamera modülü ile çalışma kolaylığı sağlamıştır. Raspberry pi Linux tabanlı bir bilgisayar olması ile de

kullanımında rahat ve OpenCV gibi görüntü işleme kütüphaneler ile destekleyen bir sistemdir.

3.3. Kanvas İş Modeli

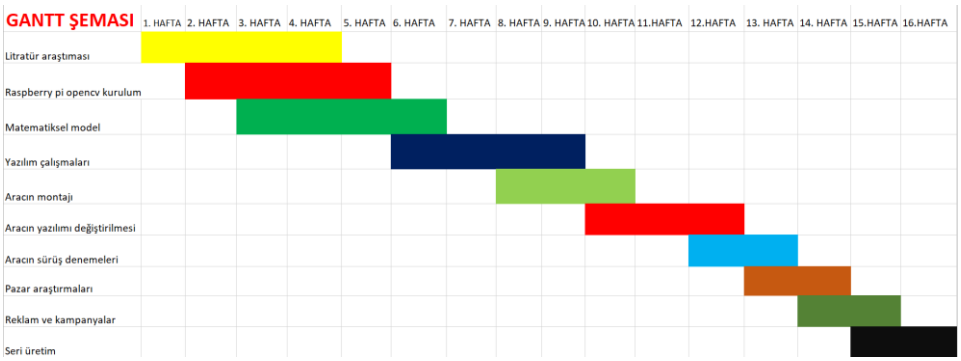
Şekil 3.1.'deki gibi kanvas iş modeli, yapılan ürünün maddi kaynaklarını, müşteri ilişkilerini, gelir akışı gibi temel özellikleri vardır.



Şekil 3.1. Kanvas iş modeli

3.4. Gantt Şeması

Gantt şemasıda şekil 3.2.'deki gibi ürünün oluşturma hedefleri görülmektedir



Şekil 3.2. Gantt şeması

BÖLÜM 4. OTONOM ARAÇ MATEMATİKSEL MODEL

Otonom aracın ileri yol denklemi ile direksiyon denklemine değineceğiz. Bu denklemler motorun içindeki elektrik diferansiyel denklemlerini ile yola çıkarak yapacağız.

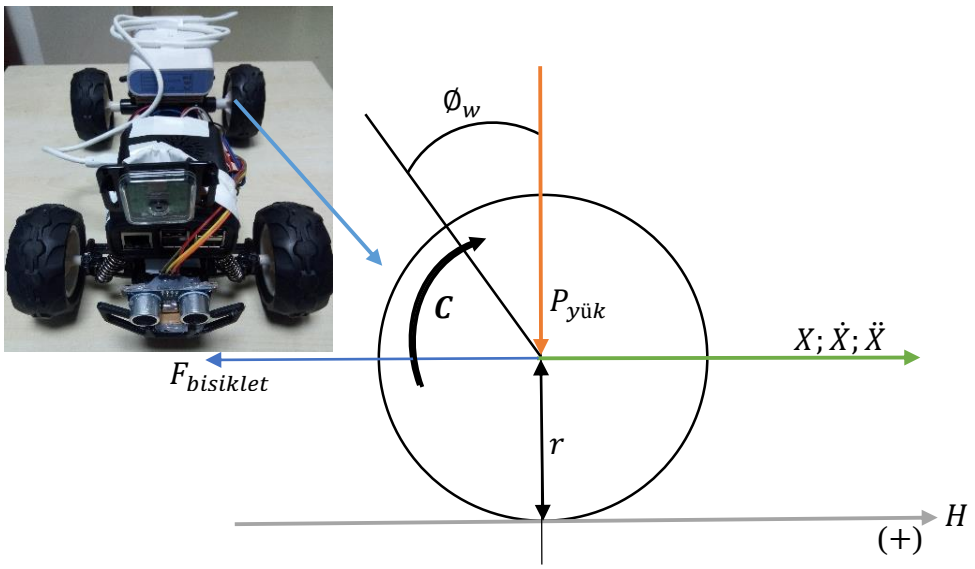
Tablo 4.1. Lineer yol denklemi için gerekli parametreler

Sembol-Değer	Açıklama-Birim
T_{motor}	:DC motor için oluşan tork arka tekerler için $[Nm]$
T_m	:DC motor için oluşan tork ön direksiyon için $[Nm]$
T_a	:DC motor ters yönde etki eden tork (Dış etkenler) $[Nm]$
I_R	:Rotor Ataleti $[kgm^2]$
w	:Motorun dönme açısal hızı $[rad/s]$
$\emptyset_w$	:Aracın tekerin dönme açısı $[rad]$
$\dot{\emptyset}_w$	: Aracın tekerin açısal dönme hızı $[rad/s]$
$\ddot{\emptyset}_w$	: Aracın tekerin açısal dönme ivmesi $[rad/s^2]$
$\emptyset_{viraj}$	:Aracın direksiyon dönme açısı $[rad]$
$\dot{\emptyset}_{viraj}$	: Aracın direksiyon açısal dönme hızı $[rad/s]$
$\ddot{\emptyset}_{viraj}$	: Aracın direksiyon açısal dönme ivmesi $[rad/s^2]$
V_a	:DC motor nominal terminal gerilimi $[Volt]$
V_e	:DC motor için ters EMK gerilimi $[Volt]$
K_m	:Motor tork sabiti $[Nms/rad]$
K_f	:Motor şaftında oluşan sürtünme $[Nms/rad]$
K_e	:Geri EMF sabiti $[Vs/rad]$
R_a	:Nominal terminal direnç arka teker $[Ohm]$
R	:Nominal terminal direnç ön direksiyon $[Ohm]$
$P_{yük}$	: Aracın arka tekere binen yük miktarı $[Newton]$

Tablo 4.1. (Devamı)

Sembol-Değer	Açıklama-Birim
L	:Rotor endüktansı $[H]$
H	: Aracın X ekseni yol alması için devir sayısı $[Radyan]$
C	:Motorun uyguladığı tork (Oluşan net tork) $[Nm]$
$F_{Araç}$	: Aracın teker (tahrik için) dışındaki kütlenin oluşturduğu reaksiyon kuvveti $[Newton]$
m_1	: Aracın toplam kütlesi $[kg]$
M_w	:Arka tekerin kütlesi $[kg]$
$a_{x1}(\ddot{x})$	: Aracın (x ekseninde) ivmelenme $[m/s^2]$
$\dot{x}$	: Aracın t (x ekseninde) hızı $[m/s]$
I_o	: Aracın tekerin atalet momenti [kgm^2]
M_o	: Aracın tekerinin oluşturduğu toplam tork $[Nm]$
r	: Aracın tekerinin yarıçapı $[m]$
r_{viraj}	: Aracın direksiyon dönmesini sağlayan mesnetin yarıçapı $[m]$
i	:Armatür aracılığıyla oluşan akım $[A]$

4.1. Otonom Araç Doğrusal Yol Denklemi



Şekil 4.1. Arabanın arka tekerin tahrik için serbest cisim diyagramı

Aracın x ekseninde, yol alırken motor torku ile açısal ivmelenmesi, açısal hızı gibi parametreler, tekerin yarıçapı çarpımı sonucu doğrusal ivmelenme ve doğrusal hız elde edilir. (Denklem 4.1) denklem ile x ekseninde oluşan kuvvetler şekil 4.1.'de serbest cisim diyagramında gösterilmiştir.

(Denklem 4.1) için aracın x ekseninde (pozitif yönde giderken) tekerin devir sayısı ile (eylemsizlik kuralına göre) robot (negatif yönde) kuvvetinin ilişkisi gösterilmiştir.

$$\sum F_{x1} = m_1 a_{x1};$$
$$m_1 \ddot{x} = H - F_{bisiklet} \quad (4.1)$$

Motor torkunun çektiği akım ile doğru orantılıdır. Motora zıt yönde uygulanan döndürme momenti ile motor akımının ilişkisi (Denklem 4.2)'da gösterilmektedir.

$$T_{motor} = I_R \frac{dw}{dt} + T_a \quad (4.2)$$

Şekil 4.1. için dönme torku "C" ile belirtmiştik. Net dönme momenti (motor torku için) (Denklem 4.3)'de gibi elde edilmiştir.

$$C = T_{motor} - T_a \quad (4.3)$$

Net motor torku, motorun çektiği akım (Denklem 4.4) gösterilmiştir.

$$C = I_R \frac{dw}{dt} \quad (4.4)$$

DC (Doğru akım) bir motorun terminal gerilimi tork ile doğru, ters EMK motor torkuna ters orantılıdır. Bu ilişki (Denklem 4.5)'de gösterilmektedir.

$$C = \frac{K_m V_a - K_m K_e \dot{\emptyset}_w}{R_a} \tag{4.5}$$

(Denklem 4.6) için tekerin tork denklemi gösterilmiştir. Elde edilen net tork ile robotun aldığı yol (Hr), durumu denklemde gösterilmiştir.

$$\begin{aligned} &\sum M_o = I_o \ddot{\emptyset}_w; \\ &I_o \ddot{\emptyset}_w = C - Hr; \end{aligned} \tag{4.6}$$

(Denklem 4.6)'de ile (Denklem 4.5) yerine konulması ile tekrar düzenlenirse (Denklem 4.7) elde edilmiştir.

$$I_o \ddot{\emptyset}_w = \frac{K_m V_a - K_m K_e \dot{\emptyset}_w}{R_a} - Hr \tag{4.7}$$

(Denklem 4.7) her iki tarafı (r) ile bölersek Denklem 4.8 elde edilmiştir. Tekerin devir adedini gösteren denklem elde edilmiştir.

$$\begin{aligned} &\frac{Hr}{r} = \frac{K_m V_a - K_m K_e \dot{\emptyset}_w}{R_a r} - \frac{I_o \ddot{\emptyset}_w}{r} \\ &H = \frac{K_m V_a - K_m K_e \dot{\emptyset}_w}{R_a r} - \frac{I_w \ddot{\emptyset}_w}{r} \end{aligned} \tag{4.8}$$

(Denklem 4.1) için H (devir sayısını) (Denklem 4.8) yerine konularak (Denklem 4.9) elde edilmiştir.

$$M_w\ddot{x} = \frac{K_m V_a - K_m K_e \dot{\emptyset}_w}{R_a r} - \frac{I_w \ddot{\emptyset}_w}{r} - F_{bisiklet} \tag{4.9}$$

(Denklem 4.9), aracın x ekseni denklemi elde etmemiz için (Denklem 4.10)'deki formülleri yazabiliriz.

$$\ddot{\emptyset}_w r = \ddot{x}, \ddot{\emptyset}_w = \frac{\ddot{x}}{r}$$
$$\dot{\emptyset}_w r = \dot{x}, \dot{\emptyset}_w = \frac{\dot{x}}{r} \tag{4.10}$$

Aracın oluşturduğu eylemsizlik kuvveti (Denklem 4.11)'deki gibi gösterilmektedir.

$$F_{Araç} = m_1 \ddot{x}; \tag{4.11}$$

(Denklem 4.9) için (Denklem 4.10)'deki formülleri ve robotun reaksiyon kuvvetini (Denklem 4.11) yerlerine yazılması ve gerekli işlemlerden sonra Denklem 4.12 elde edilmiştir [8].

$$M_w\ddot{x} = \frac{K_m V_a}{R_a r} - \frac{K_m K_e \dot{x}}{R_a r^2} - \frac{I_w \ddot{x}}{r^2} - m_1 \ddot{x}$$
$$(M_w + \frac{I_w}{r^2} + m_1)\ddot{x} + \frac{K_m K_e}{R_a r^2}\dot{x} = \frac{K_m}{R_a r} V_a \tag{4.12}$$

4.1.1. Lineer yol için TF

(Denklem 4.12) denkleminin uzun parametrelerini tek bir değişkenle aşağıdaki gibi gösterebiliriz.

$$(M_w + \frac{I_w}{r^2} + m_1) = a_1$$

$$\frac{K_m K_e}{R_a r^2} = b_1$$

$$\frac{K_m}{R_a r} = c_1$$

(Denklem 4.12) için 2.dereceden türevin katsayı değeri "1" yapalım ve gerekli işlemlerinden sonra (Denklem 4.13) elde edilmiştir.

$$\ddot{x} + \frac{b_1}{a_1}\dot{x} = \frac{c_1}{a_1}V_a \tag{4.13}$$

(Denklem 4.14) için laplace dönüşümü yaparak (Denklem 4.14) elde edilmiştir.

$$s^2X(s) + \frac{b_1}{a_1}sX(s) = \frac{c_1}{a_1}V_a(s) \tag{4.14}$$

(Denklem 4.14), ($X(s)$) ortak parantezi alınarak, sistemin TF için gerekli işlemlerden sonra Denklem 4.15 elde edilmiştir [9].

$$X(s)\left[s^2 + \frac{b_1}{a_1}s\right] = \frac{c_1}{a_1}V_a(s) \tag{4.15}$$

$$\frac{X(s)}{V_a(s)} = \frac{c_1/a_1}{s^2 + \frac{b_1}{a_1}s}$$

4.1.2. Lineer yol denklemi için UZM

Aracın lineer yol alırken durum UZM matematiksel olarak gösterebiliriz. Çıkış vektörü, x (alınan yol) çıktısını verir ve durum vektörünün zamana bağlı türevi $\dot{x}$ (robotun zamana bağlı hızı çıktıları), olarak gösterilir. Y vektörü çıkış vektörüdür. U,

giriş vektörüdür. A, sistem matrisi; B, giriş matrisi; C, çıkış matrisi; D, ileri besleme matrisidir.

(Denklem 4.13) için UZM oluşturmak için (Denklem 4.17)'deki robotun konumu(x), hızı($\dot{x}$), ivmesi($\ddot{x}$) değişkenlere atanmıştır.

UZM genel denklemi (Denklem 4.16)'de gösterilmektedir.

$$\dot{x}_{D1} = Ax + Bu$$
$$y_{C1} = Cx + Du \tag{4.16}$$

$$x_1 = x; \qquad \dot{x}_1 = \dot{x}$$
$$x_2 = \dot{x}; \qquad \dot{x}_2 = \ddot{x} \tag{4.17}$$

(Denklem 4.13) denklemini, (Denklem 4.17)'deki değişkenlerle ifade edersek (Denklem 4.18) elde edilmiştir.

$$\dot{x}_2 + \frac{b_1}{a_1} x_2 = \frac{c_1}{a_1} V_a \tag{4.18}$$

(Denklem 4.18)'deki gerekli düzenlemelerden sonra aracın ivmesi için (Denklem 4.19) elde edilmiştir.

$$\dot{x}_2 = -\frac{b_1}{a_1} x_2 + \frac{c_1}{a_1} V_a \tag{4.19}$$

(Denklem 4.19) elde edilen denklem, durum vektörü ile çıkış vektörü, (Denklem 4.20) ile (Denklem 4.21) gösterilmiştir [9,10].

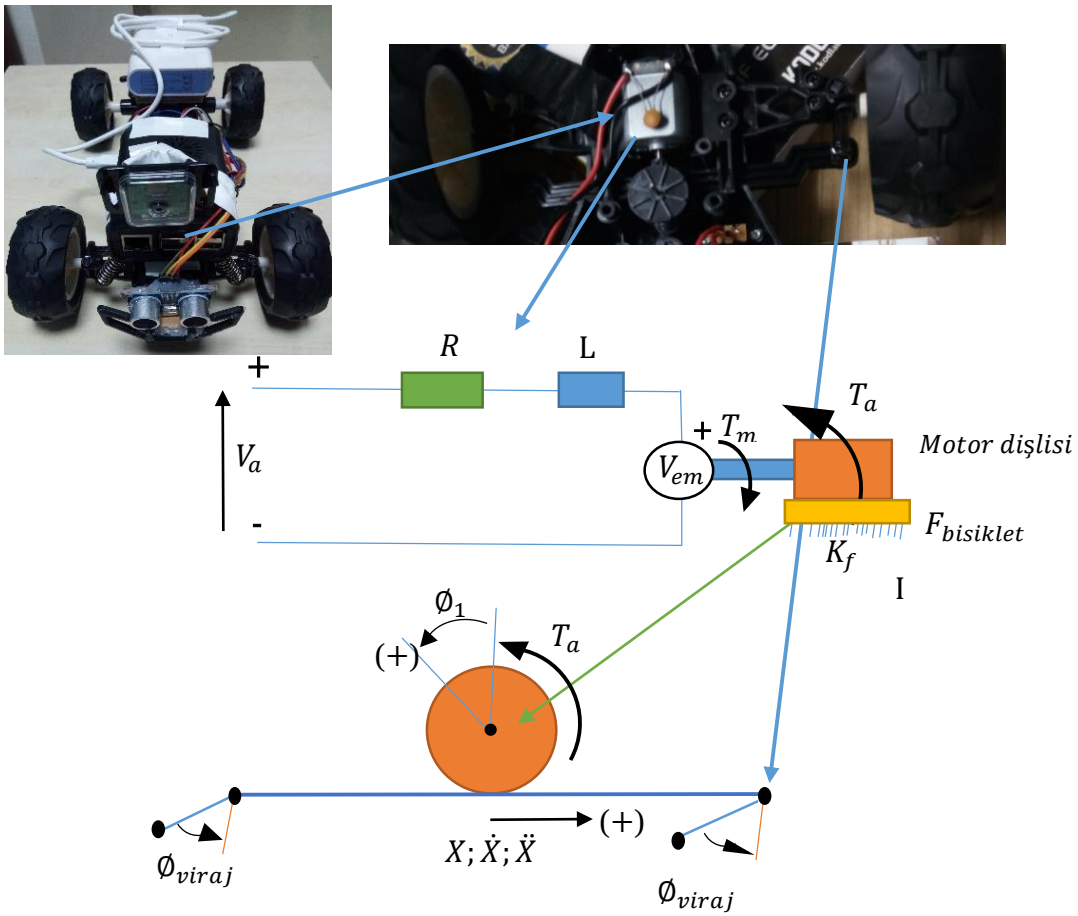
Durum vektörü,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{b_1}{a_1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{c_1}{a_1} \end{bmatrix} V_a \tag{4.20}$$

Çıkış vektörü,

$$y = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{4.21}$$

4.2. Aracın Direksiyon Hareket Denklemleri



Şekil 4.2. Aracın ön direksiyon matematiksel modeli

Araç direksiyon için DC motorunun dönmesi ile lineer hareket x ekseninde yapmıştır. Lineer olarak yol alırken motor torku ile açısal ivmelenmesi, açısal hızı gibi parametreler, aracın direksiyon açısı motor parametreleri ile diferansiyel denklem elde edilmiştir.

(Denklem 4.22) için motor terminal direnç değerinin, motorun çektiği akım ile motorun tork değeri elde edilmiştir. Motor torkunun çektiği akım ile doğru orantılıdır.

$$T_m = K_m i \tag{4.22}$$

Motora zıt yönde uygulanan zıt EMK voltajının, motor şaftını hızı ile zıt EMK sabiti ile orantılıdır. (Denklem 4.23)'da gösterilmektedir.

$$V_e = K_e \dot{\emptyset}_w \tag{4.23}$$

Şekil 4.2. için motorun elektrik devresinin formülü (Denklem 4.24)'de gibi elde edilmiştir.

$$V_a - R_i - L\frac{di}{dt} = 0 \tag{4.24}$$

Toplam moment, motorun çektiği akım ile motorun şaft hızının ilişkisi (Denklem 4.25) gösterilmiştir.

$$\sum M = T_m - K_f w - T_a = I_R \dot{\emptyset}_w \tag{4.25}$$

DC motorun akımının zamana göre türevi, denklem parametreleri ile bobinin ilişkisi (Denklem 4.26)'de gösterilmektedir.

$$\frac{d_i}{d_t} = \frac{Ri}{L} + \frac{K_e \dot{\emptyset}_w}{L} + \frac{V_a}{L} \tag{4.26}$$

DC motorun şaftının ani ivmelenmesi, (Denklem 4.27)'de motor denklem parametreleri ile motorun çektiği akımın arasındaki ilişkisi gösterilmiştir.

$$\ddot{\emptyset}_w = \frac{K_m i}{I_R} + \frac{-K_e \dot{\emptyset}_w}{I_R} - \frac{T_a}{I_R} \tag{4.27}$$

(Denklem 4.28)'de motorun akımı ile ilgili denklem gösterilmiştir.

$$i = \frac{-K_e \dot{\emptyset}_w}{R} + \frac{V_a}{R} \tag{4.28}$$

(Denklem 4.29) motorun şaftının ani ivmelenmesinin denklem gösterilmiştir. (Denklem 4.28) formülden (Denklem 4.29)'a yerine konularak (Denklem 4.30) elde edilmiştir [12].

$$\ddot{\emptyset}_w = \frac{K_m i}{I_R} - \frac{T_a}{I_R} \tag{4.29}$$

$$\ddot{\emptyset}_w = -\frac{K_m K_e}{I_R R} \dot{\emptyset}_w + \frac{V_a}{I_R R} - \frac{T_a}{I_R} \tag{4.30}$$

(Denklem 4.30)'da motorun yaptığı ani ivmelenmenin sonucu direksiyon lineer hareket edecektir. Direksiyon için x ekseni denklemi elde etmemiz için (Denklem 4.31)'deki formülleri yazabiliriz.

$$\begin{aligned} &\ddot{\emptyset}_w r = \ddot{x}, \ddot{\emptyset}_w = \frac{\ddot{x}}{r} \\ &\dot{\emptyset}_w r = \dot{x}, \dot{\emptyset}_w = \frac{\dot{x}}{r} \end{aligned} \tag{4.31}$$

(Denklem 4.30)'deki motorun dönme denklemi (Denklem 4.31)'deki değişiklikleri yaparak (Denklem 4.32)'de direksiyon lineer hareket denklemi gösterilmektedir.

$$\frac{\ddot{x}}{r} = -\frac{K_m K_e}{I_R R} \frac{\dot{x}}{r} + \frac{V_a}{I_R R} - \frac{T_a}{I_R} \tag{4.32}$$

(Denklem 4.32) lineer hareket denklemidir. Bu hareket sonucu aracın tekerleri açısal olarak dönme hareketi yapmaktadır.

$$\ddot{\emptyset}_{viraj} r_{viraj} = \ddot{x} \qquad (4.33)$$
$$\dot{\emptyset}_{viraj} r_{viraj} = \dot{x}$$

(Denklem 4.33)'de viraj için denklem parametreleri gösterilmiştir. (Denklem 4.32) denklemin gerekli değişiklikler yaparak (Denklem 4.34) elde edilir [6].

$$\frac{\ddot{\emptyset}_{viraj} r_{viraj}}{r} = -\frac{K_m K_e}{I_R R}\frac{\dot{\emptyset}_{viraj} r_{viraj}}{r} + \frac{V_a}{I_R R} - \frac{T_a}{I_R} \qquad (4.34)$$
$$\ddot{\emptyset}_{viraj} = -\frac{K_m K_e}{I_R R}\dot{\emptyset}_{viraj} + \frac{V_a}{I_R R}\frac{r}{r_{viraj}} - \frac{T_a}{I_R}\frac{r}{r_{viraj}}$$

4.2.1. Lineer yol için TF

Denklem 4.34 denkleminin uzun parametrelerini tek bir değişkenle aşağıdaki gibi gösterebiliriz.

$$\frac{K_m K_e}{I_R R} = a_2$$
$$\frac{r}{r_{viraj} I_R R} = b_2$$
$$\frac{r}{r_{viraj} I_R} = b_2 R = c_2$$

Denklem 4.34 için 2.dereceden türevin katsayı değeri "1" yapalım ve gerekli işlemlerinden sonra (Denklem 4.35) elde edilmiştir [9].

$$\ddot{\emptyset}_{viraj} + \frac{K_m K_e}{I_R R}\dot{\emptyset}_{viraj} = \frac{V_a}{I_R R}\frac{r}{r_{viraj}} - \frac{T_a}{I_R}\frac{r}{r_{viraj}}$$
$$\ddot{\emptyset}_{viraj} + a_2\dot{\emptyset}_{viraj} = V_a b_2 - T_a c_2 \quad (4.35)$$
$$\ddot{\emptyset}_{viraj} + a_2\dot{\emptyset}_{viraj} = V_a b_2 - T_a b_2 R$$

Denklem 4.35 için laplace dönüşümü yaparak Denklem 4.36 elde edilmiştir.

$$\ddot{\emptyset}_{viraj} + a_2\dot{\emptyset}_{viraj} = V_a b_2 - T_a b_2 R \quad (4.36)$$
$$s^2\emptyset_{viraj}(s) + a_2 s\emptyset_{viraj}(s) = b_2 V_a(s) - b_2 R T_a(s)$$

Denklem 4.36, ($\emptyset_{viraj}(s)$) ortak parantezi alınarak, sistemin TF için gerekli işlemlerden sonra Denklem 4.37 elde edilmiştir [9].

$$\emptyset_{viraj}(s)[s^2 + s] = b_2[V_a(s) - RT_a(s)] \quad (4.37)$$
$$\frac{\emptyset_{viraj}(s)}{V_a(s) - RT_a(s)} = \frac{b_2}{s^2 + s}$$

4.2.2. Lineer yol denklemi için UZM

Robotumuzun lineer yol alırken durum UZM matematiksel olarak gösterebiliriz. Çıkış vektörü, x (alınan yol) çıktısını verir ve durum vektörünün zamana bağlı türevi $\dot{x}$ (robotun zamana bağlı hızı çıktıları), olarak gösterilir. Y vektörü çıkış vektörüdür. U, giriş vektörüdür. A, sistem matrisi; B, giriş matrisi; C, çıkış matrisi; D, ileri besleme matrisidir.

(Denklem 4.34) için UZM oluşturmak için (Denklem 4.39)'deki robotun konumu(x), hızı($\dot{x}$), ivmesi($\ddot{x}$) değişkenlere atanmıştır. UZM genel denklemi (Denklem 4.38)'de gösterilmektedir.

$$\dot{x}_{D2} = Ax + Bu \tag{4.38}$$
$$y_{C2} = Cx + Du$$

$$x_1 = \emptyset_{viraj}; \quad \dot{x}_1 = \dot{\emptyset}_{viraj} \tag{4.39}$$
$$x_2 = \dot{\emptyset}_{viraj}; \quad \dot{x}_2 = \ddot{\emptyset}_{viraj}$$

(Denklem 4.34) denklemini, (Denklem 4.39)'deki değişkenlerle ifade edersek (Denklem 4.40) elde edilmiştir.

$$\dot{x}_2 + a_2 x_2 = V_a b_2 - T_a b_2 R \tag{4.40}$$

(Denklem 4.40)'deki gerekli düzenlemelerden sonra robotun ivmesi için (Denklem 4.41) elde edilmiştir.

$$\dot{x}_2 = -a_2 x_2 + V_a b_2 - T_a b_2 R \tag{4.41}$$

(Denklem 4.41) elde edilen denklem, durum vektörü ile çıkış vektörü, (Denklem 4.42) ile (Denklem 4.43) gösterilmiştir [9,11].

Durum vektörü,

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & -\frac{K_m K_e}{I_R R}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0 & 0\\ \frac{K_m K_e}{I_R R}\frac{r}{r_{viraj}} & -\frac{1}{I_R}\frac{r}{r_{viraj}}\end{bmatrix}\begin{bmatrix}V_a\\ T_a\end{bmatrix} \tag{4.42}$$

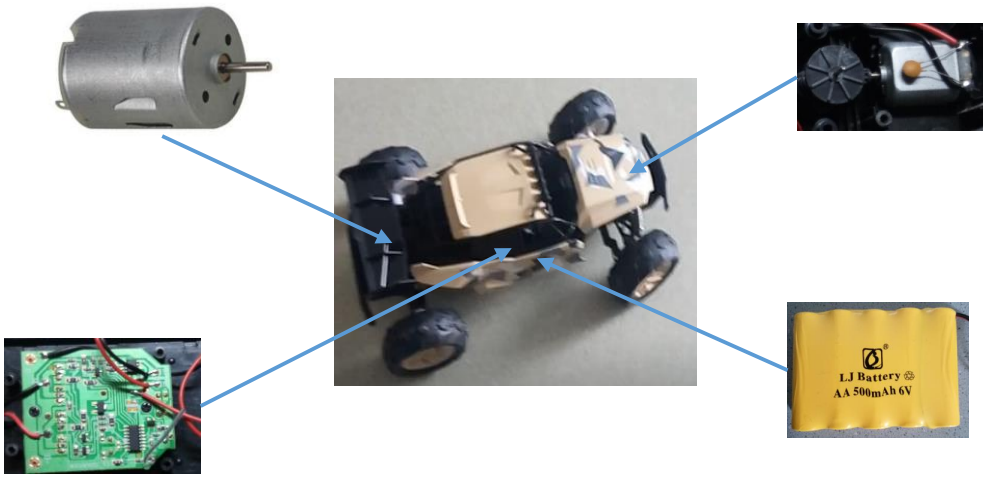
Çıkış vektörü,

$$y = [1 \quad 0]\begin{bmatrix}x_1\\ x_2\end{bmatrix} \tag{4.43}$$

BÖLÜM 5. FİZİKSEL MODEL

Otonom araç için donanımsal bilgiler aşağıdaki gibidir. Aracımız temel olarak bir hazır RC arabanın içindeki elektronik kontrol ünitesi çıkartılmış ve tercih ettiğimiz ürünler ile raspberry pi ve görüntü işleme yöntemleri kullanarak otonom araç yapımı hedeflenmiştir.

5.1. RC Araba



Şekil 5.1.RC araba

Otonom araba yapmak için hazır bir RC arabanın şekil 5.1.'deki gibi prototipini kullanacağız. Aracın arka tekerleri bir DC motor ile redüktör vasıtasıyla aynı yönde hareket etmektedir. Aracın direksiyon kontrolü bir DC motor ile sağa ve sola dönmesini sağlayan sistem bulunmaktadır. Motorları ve elektronik kontrol ünitesi besleyen LJ tipi 500 mAh, 6V, şarjlı bir batarya bulunmaktadır. Aracın içerisinde bulunan elektronik kontrol ünitesi (EKÜ) çıkarttık. Böylelikle motor bağlantılarını kendi sürücümüzü takacağız. Sürücümüz ise raspberry pi ile kontrol edeceğiz. Raspberry pi ile görüntü işleme yöntemleri ile otonom hareket eden araç yapacağız.

5.1.1. AA 500 mAh 6 volt şarjlı pil

Şekil 5.2. AA 500 mAh 6 volt pil

Şekil 5.2'daki gibi nominal gerilim değeri 6 volttur. Nominal kapasite değeri 500 mAh'dır. Düşük iç direnç ve uzun ömürlüdür [13].

5.1.2. DC motor 6 volt 250 mA (arka tekerler için)

Şekil 5.3. DC motor

Şekil 5.3.'deki gibi DC motor hız 14500 rpm (yüksüz) 0.21 A, 12530 rpm (maks. yüklü) 1.34 A, tork değeri 46.2g.cm, durma momenti 340g.cm özelliklere sahiptir [14].

5.1.3. Fırçalı DC motor 6 volt 250 mA

Şekil 5.4. Fırçalı DC motor

Şekil 5.4.'deki gibi fiçalı DC motor başlangıç torku 20 g.cm, yüksüz hızı 9100 rpm 0.25 A, maks. yüklü 1800 rpm 0.5 A özelliklerine sahiptir [15] .

5.2. Raspberry pi 3 B+ Mini Bilgisayar

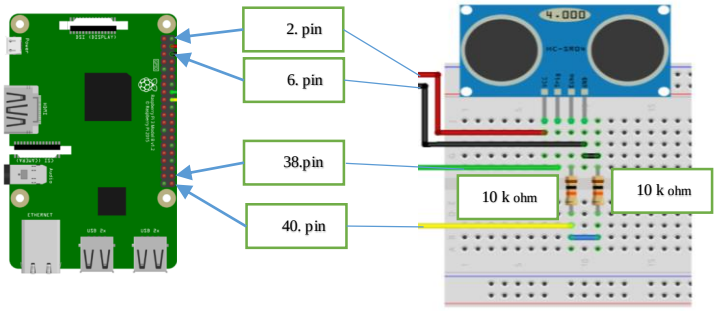
Şekil 5.5.Raspberry pi 3B+

Raspberry pi şekil 5.5.'deki gibi dört çekirdek, 64 bit, 1GB ram, 40 GPIO pine sahip, kredi kartı boyutlarında bir mini bilgisayardır. Raspberry pi Vakfı tarafından eğitim için geliştirilmiştir. 1981 BBC Micro'dan örnek alınarak herkesin ulaşabileceği, normal bilgisayardan daha düşük işletim sistemine sahip olsa da karışık olan işlemleri yapabilir [16].

5.3. HC-SR04 Ultrasonik Mesafe Sensörü

Şekil 5.6.HC-SR04

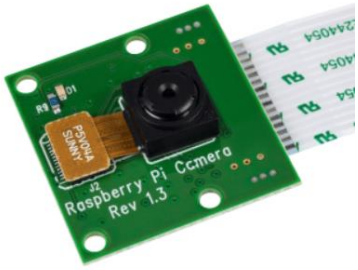
HC-SR04 ultrasonik mesafe sensörü şekil 5.6.'de, yarasaların etrafı görme tekniği olan ses sinyallerinin yankılanarak geri dönmesi ile çalışır. Belli aralıklarla ses sinyalini açılı bir şekilde yollar ve herhangi bir nesneye çarptığında ses sinyali çarpma açısına göre geri döner. Menzili 2cm ile 400cm arasında olduğu için mesafe uygulamalarında sıklıkla kullanılır [17].



Şekil 5.7. HC-SR04 ile raspberry pi devresi

Şekil 5.7.'de raspberry pi ile mesafe sensörü arasında gerilim bölücü yapmalıyız. Raspberry pi giriş voltu olarak 3.3 volt almaktadır. HC-SR04 sensörü Echo pini 5 Volt vererek raspberry pi kartımızı zarar görebilir. Böylece gerilim bölücü ile 2.5 volt vererek raspberry pi kartımız zarar görmez.

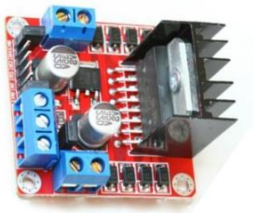
5.4. Raspberry Pi Kamera Modülü



Şekil 5.8. Raspberry pi kamera modülü

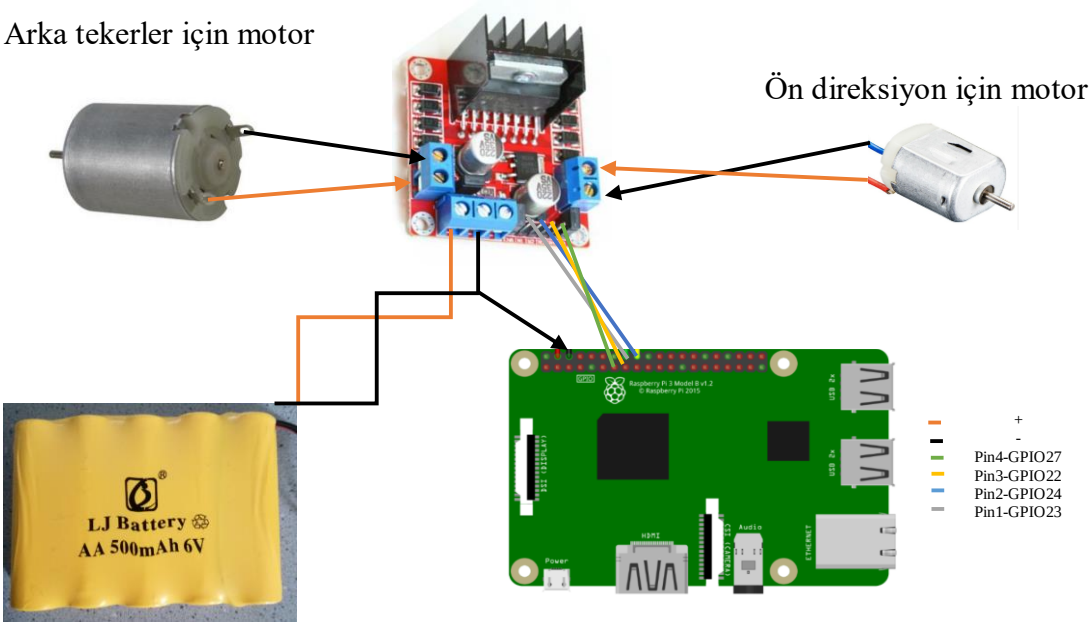
Raspberry pi kamera modülü şekil 5.8. gibi üzerindeki CSI konnektörüne direk bağlanabilen, yüksek çözünürlük bir video kamerasıdır. 5 MP çözünürlüklü kamera 1080p video ve sabit fotoğraf çekme kapasitesine sahiptir. Dahili ribbon kabloyu raspberry'mizin CSI (kamera seri arayüzü) portuna bağlanır.

5.5. L298N DC Motor Sürücü



Şekil 5.9. L298N

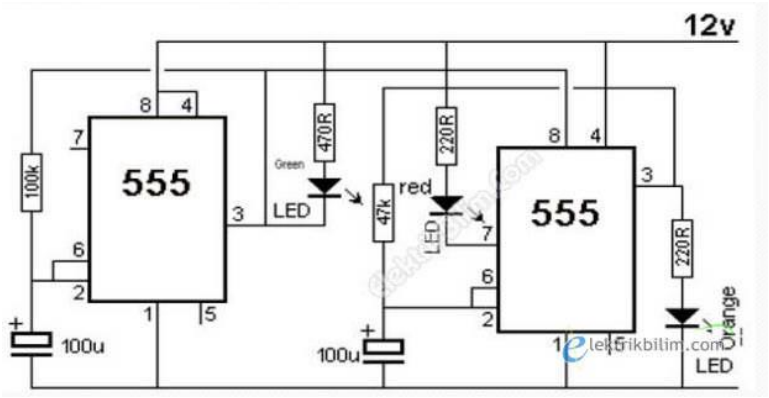
Şekil 5.9.'daki özellikleri birbirinden bağımsız iki motoru kontrol eder. Kanal başına 2 A akım vermektedir. Yüksek sıcaklık ve kısa devre koruması mevcuttur. Üzerinde dâhili regülatörü vardır. Kart üzerinde dâhili soğutucu vardır [19].



Şekil 5.10. Otonom araç motor bağlantılar

5.6. Mini Trafik Lambası

Trafik lambası devresi şekil 5.11.'deki gibi 2 adet 555 zamanlayıcı entegre yardımıyla diğer direnç ve kondansatör yardımı ile birbirinden farklı zamanlarda yanıp sönen sonsuz döngüde çalışan bir elektrik devresidir. Biz 3 voltluk led kullanmaktayız. Bundan dolayı biz devreye 6 volt ile kontrol edeceğiz [18].



Şekil 5.11. Trafik lambası devresi

5.7. Power Bank

Şekil 5.12. Raspberry pi besleme powerbank

Giriş değeri 5 volt, 2 amper olan; çıkış değeri 2 adettir. Birinci çıkış 5 volt 1 amper, ikinci çıkış 5 volt 2 amperdir. Kapasitesi 7800 mAh'dır. Raspberry pi ikinci çıkışa göre besleme yapacağız.

5.8. Otonom Araç Genel Şeması

Otonom aracın görünümü şekil 5.13.'deki gibidir.

Şekil 5.13. Otonom araç

5.8.1. Otonom araç genel maliyet şablonu

Tablo 5.1. Otonom araç genel maliyeti

Ürün Adı:	Fiyatı(KDV Dahil)
HC-SR04 Ultrasonik Mesafe Sensörü	6.61 TL
Raspberry pi 3 model B+	271 TL
RC araba	180 TL
Yedek şarjlı pil	63.72 TL
L298 motor sürücü	14.29 TL
Raspberry pi kamera modülü	78 TL
Raspberry pi kamera kutusu	27.86 TL
Powerbank	40 TL
Mini trafik lambası (Genel)	30 TL
Maket yol	30 TL
Toplam:	741.48 TL

BÖLÜM 6. OTONOM ARAÇ TASARIMI

Otonom bir aracın yolda kararlı bir şekilde gitmesi için şerit takibine göre gitmelidir. Bu durum yapay bir sistemin işletim sisteminde gerçekleştirmesi zordur. Otonom aracın bağımsız bir şekilde gitmesi için yeterli bir işletim sisteme ihtiyacı vardır. Otonom bir sistem içinde dışarıdan yeterli veri alınması gerekmektedir. Dışarıdan veri akışı için kamera en yaygın tercihtir. Otonom aracın kontrol etmek için en yaygın kullanılan mini bilgisayar olan raspberry pi ve OpenCV kütüpanesi ile dışarıdan gelen verilere göre karar veren bir işletim sistemi ve görüntü işleme yöntemleri ile gerçek zamanlı yol algılama, tabela tanımlama, trafik ışıklarına göre hareket eden ve bu sitemin yeterli işlem gücüne sahiptir. Görüntü işleme için raspberry pi için yüksek çözünürlüklü kamera modülüne sahiptir. Ancak raspberry pi kamera ile aracın seyir halinde iken yüksek çözünürlükte ani kararlar alamadığı gözlemlenmiştir. Bunun için raspberry pi için kamera yüksek çözünürlükleri algılama yapsa da biz raspberry pi'nin daha hızlı kararlar algılaması hedeflenmiştir.

6.1. Otonom Araç Programlama

Otonom aracın raspberyy pi'de python OpenCV ile görüntü işleme yöntemleri ile yapılmıştır.

6.1.1. Python

Python, ilk olarak Gudio van Rossum tarafından 1991'de hazırlanmış genel olarak bir programlama dilidir. Yorumlanabilen ve dinamik dil olan python, esas olarak nesne tabanlı programlama yaklaşımını ve belli oranda da fonksiyonel programlamayı desteklemektedir.

Python yazılım vakfı, ana geçimi C dili gerçekleşimini özgür ve açık kaynak kod mantığı altında yürütülmekte fikirsel hakları korunmaktadır. Python, görece kolaylığı ve sahip olduğu geniş standart kütüpanesi de oldukça yaygındır [30].

6.1.2. OpenCV kütüphanesi

Opencv kütüphanesi, resim ve videoları sanal ortamda işlemek için 1999 yıkında Intel tarafından geliştirilmeye başlanmış görüntü işleme kütüphanesidir. BSD lisansına sahip olması sayesinde açık kaynak kodludur. Programla dili olarak C ve C++ dili kullanılır. Windows, Linux, MacOS gibi pek çok platformda kullanılır. Kütüphane içinde beş temel kütüphane barındırır. CV(Computer Vision) görüntü işleme fonksiyonlarını içeren yüksek seviyeli algoritmalara sahip bir kütüphanedir. Makine öğrenmesi için kullanılan MLL(Machine Learning Library) istatistiksel verilere ulaşmak ve sınıflandırmak için kullanılan fonksiyonları içeren bir kütüphanedir. HighGUI grafik ara birim ile nesne yaratma, resim ve video kaydetme, yükleme, hafızadan silmek için gerekli giriş-çıkış fonksiyonlarını içerir. CXCore içinde IplImage, cvPoint, cvSize, cvHistogram vs. yapılarını içeren ve XML desteği sunun bir kütüphanedir. CvAux ise şablon eşleştirme(tamplate matching), şekil eşleştirme(shape matching), yüz tanıma(face recognition), nesnenin ana hatlarını bulma(finding skeletons), vücut hareketlerini tanıma(gesture recognition) ve kamera kalibrasyonu gibi deneysel algoritmaları içeren bir kütüphanedir [31].

6.2. Otonom Araç İçin Görüntü İşleme

Otonom aracın şerit takibi yapabilmesi için görüntünün sadece istenilen bölümü işlemek büyük fayda sağlamıştır. Bunu raspberry pi'de python ve OpenCV kütüpanesi kullanılarak yapılabilir. Kameranın görüntüsü (320,240) boyutlarında görüntü almaktayız. Raspberry pi'de küçük boyutlarda görüntü işlemek, işlemcinin görüntü kaçırmadan karar vermesini sağlanır. Aracın mesafe sensörü ile belli bir mesafede engel ile karşılaşıldığında aracımız durmaktadır. Böylece aracın karşısında trafik tabelasını ve trafik ışığını algıladığında araç durmaktadır. Böylece araç durgun halde

iken görüntünün daha iyi işleneceğinden dolayı aracın verdiği kararlarda daha doğru sonuçlara ulaşılıştır.

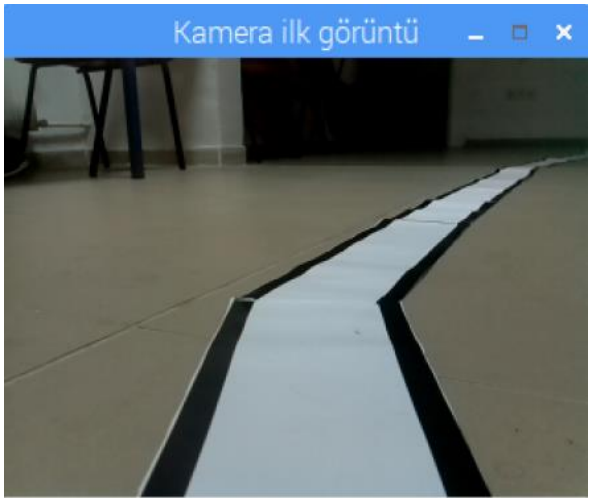
6.2.1. Şerit tespit edilmesi

Kameradan algılanan ilk görüntüleri şerit tesbitlerini yapabilmek için görüntü işleme yöntemlerini kullanacağız. Bu işlemler için renk filtreleme yapılarak şerit imgesi oluştururuz. Oluşan bu imgenin daha doğru kararlar almamızı sağlamaktadır. Görüntü işleme sonucu şerit çizgileri elde ettik. Şerit çizgilerinin algılamada işleminden sonra çizgilerin eğimlerinden faydalanarak sağ şerit sol şerit algılama gerçekleşmiştir.

Kullanılacak yöntemler:

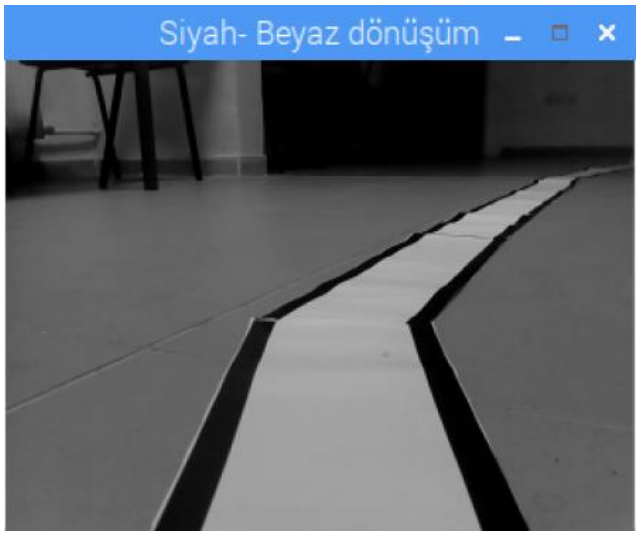
1. Gri Ton Dönüşümü
2. Beyaz Rengin Maskelenmesi
3. Gauss Filtresi
4. Canny Kenar Bulma
5. İstenilen Alan
6. Hough Dönüşümü ile Doğru Tespiti

Şekil 6.1'deki gibi kamerada alınan ilk görüntüdür. Kameradan alınan görüntülerin görüntü işleme yöntemi ile şerit takibi yapacağız.



Şekil 6.1. Kamera ilk görüntü

Gri ton dönüşümü en yaygın kullanılan bir görüntü işleme yöntemidir. Kamera görüntüleri RGB (kırmızı-mavi-yeşil) ile 24 bit boyutlarda algılamaktadır. Gri ton dönüşümü yapılarak 8 bit boyutlarda algılamaktadır. Gri ton dönüşüm ile 3 kat daha hızlı işlem yapar. Şekil 6.1.'de kameradan alınan görüntüleri şekil 6.2.'deki gibi kamera görüntülerini görüntü işleme yöntemleri olan gri tona dönüşüm yapılarak, görüntünün daha kolay işlemesi sağlanır.

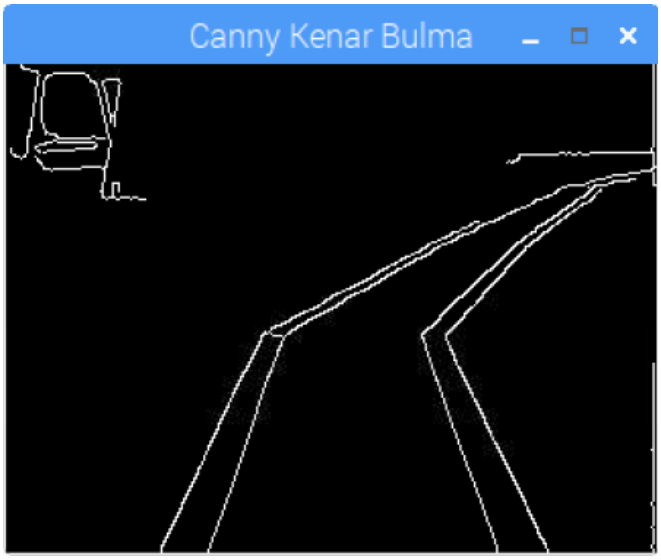


Şekil 6.2. Gri tona dönüşüm

imgGrayscale=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY) şekil 6.2. elde edilmiştir. Kamera görüntülerini gri ton dönüşümünün ardından piksellerin komşuluk ilişkilerine göre merkez piksellerin yakınındakilerini etki ederek kendisine benzetmesidir. Gauss filtresi kamera görüntülerinin düzleştirmek ve gürültüyü ortadan kaldırmak için uygulanan görüntü işleme yöntemidir. Şekil 6.2.'deki gri ton dönüşümü yapılmıştı. Şimdi şekil 6.3.'de siyah ve beyaz renklerin maskelenmesi yapılmıştır.

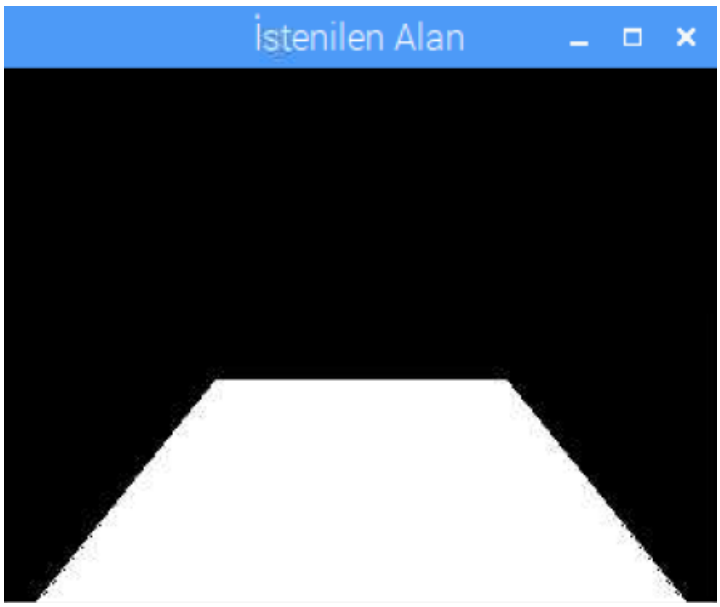
Şekil 6.3. Gauss filtresi

imgBlurred = cv2.GaussianBlur(imgGrayscale,(5,5),0) şekil 6.3. elde edilmiştir. Kenar algılamak için canny filtresi önemli bir görüntü işleme yöntemidir. Yatay yönde ilk türevi hesaplamak için yatay ve dikey yöndeki sobel çekirdeği ile filtrelenir. Kenarların gradyent büyüklüğü ve açıları hesaplanır ve kenar sayılmayan pikseller yok edilir. Görüntü taranırken yerel maksimum değerin etrafında olmadığına dikkat edilir. Asıl kenarlar algılandığında, seçilmesi gereken maksimum ve minimum değerlerini belirlediğimiz bir karşılaştırma ile sadece istenilen kenarların seçtiğimiz adımdır. Kameradan gelen görüntülerin şekil 6.3'de siyah ve beyaz renklerini maskelenmesi ardından, şekil 6.4.'de canny kenar bulma yöntemi ile şerit algılama yapılmıştır.



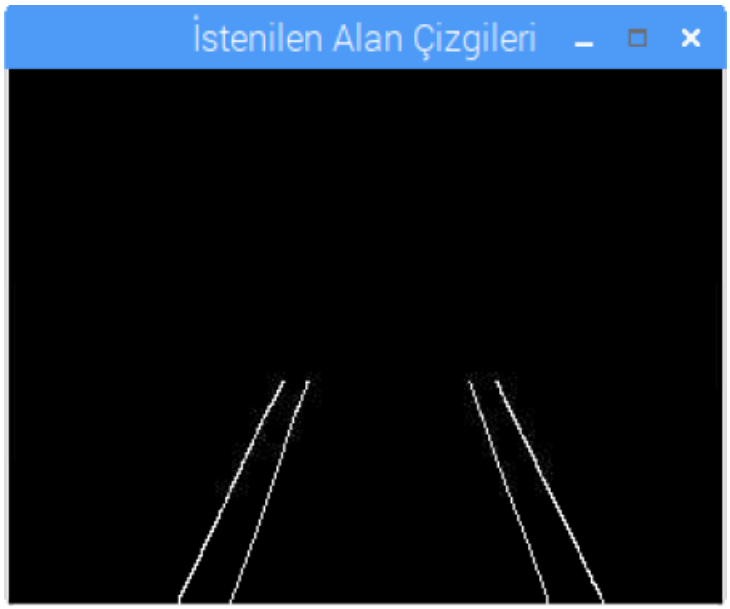
Şekil 6.4. Canny kenar bulma

imgCanny=cv2.Canny(imgBlurred,0,255) ile şekil 6.4. elde edilmiştir. Kamera görüntüsünün işlenmesi gereken bir alanı ham resimden ayırt ederek iş yükünün azaltmak için uygulanan görüntü işleme yöntemidir. Şekil 6.4.'de elde edilen görüntü işleme yöntemi ile canny kenar algılandı. Ama her kenar algılanmaması için belirli bir alan belirledik. Şekil 6.5.'de istenilen alan gösterilmektedir.



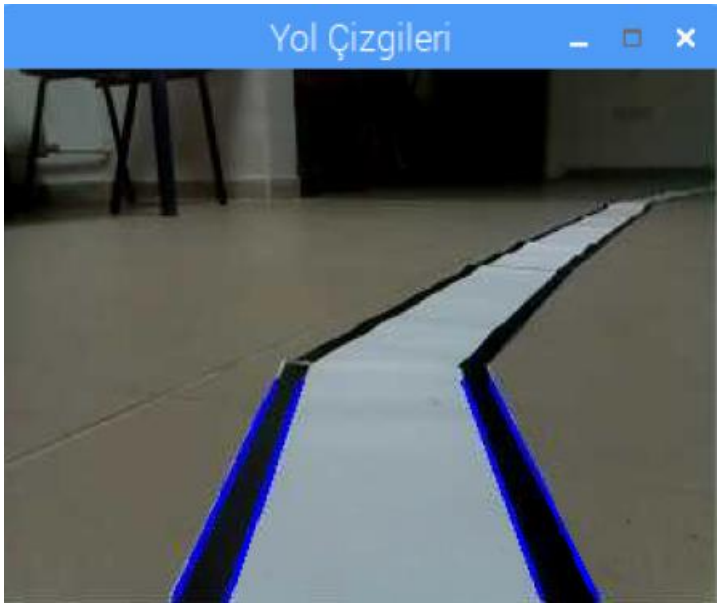
Şekil 6.5. İstenilen alan

vertices =np.array([[(15,240),(95,140),(225,140),(305,240)]],np.int32) yukardaki verilen aralıklar (320,240) boyutlarındaki görüntüye göre verilmiştir. Şekil 6.5.'de istenilen alan gösterilmiştir. Şekil 6.5.'de istenilen alanları belirlemiştik. Şekil 6.4.'de elde edilen kenarların istenilen alandaki çizgiler şekil 6.6.'da gösterilmiştir.

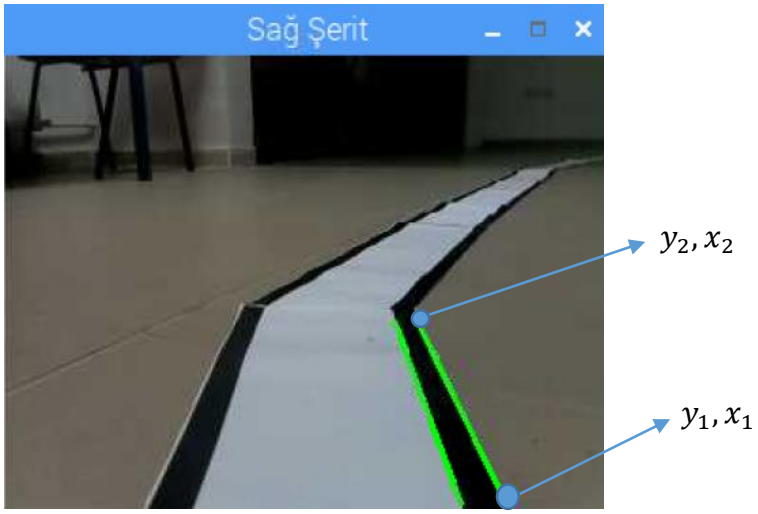


Şekil 6.6. İstenilen alan canny çizgisi

masked = cv2.bitwise_and(imgCanny, mask) kod yardımı ile şekil 6.6. elde edilmiştir. Şekil 6.6.'da istenilen alan ile elde edilen kenarları, görüntü işleme yöntemi ile kenarlar belirlenmiştir. cv2.line(frame, (x1, y1), (x2, y2), (0,255, 0), 2) ile şerit çizgileri elde ederiz.



Şekil 6.7. Şerit algılama

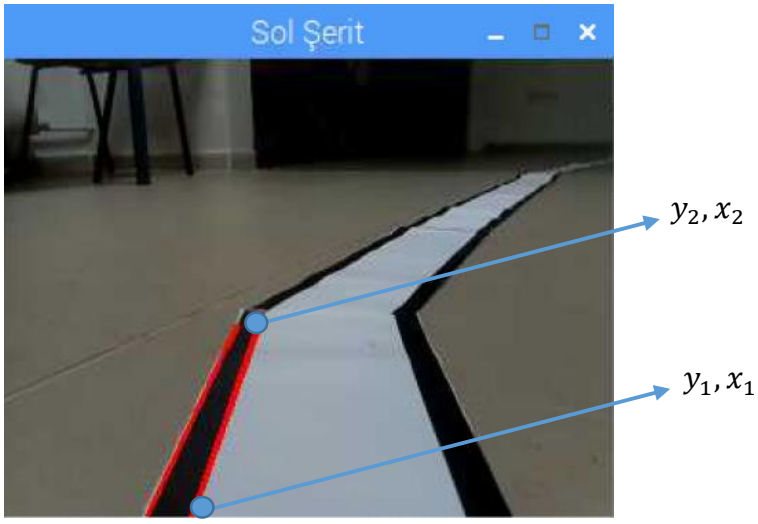


Şekil 6.8. Sağ şerit algılaması

Şekil 6.8.'deki gibi görüntü işleme yöntemi ile şeridin eğiminden faydalanılarak, şeridin sağ şerit olduğunu anlarız.

$$Sağ\ şerit\ eğimi = \frac{y_2 - y_1}{x_2 - x_1} \tag{6.1}$$

(Denklem 6.1)'de sağ şeridin eğimi sıfırdan büyüktür. Sağ şerit algılanmıştır.

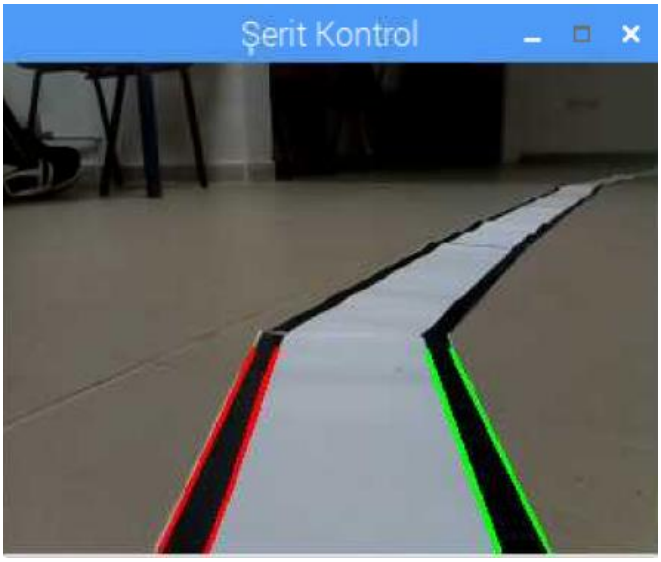


Şekil 6.9. Sol şerit algılaması

Şekil 6.9.'deki gibi görüntü işleme yöntemi ile şeridin eğiminden faydalanılarak, şeridin sol şerit olduğunu anlarız.

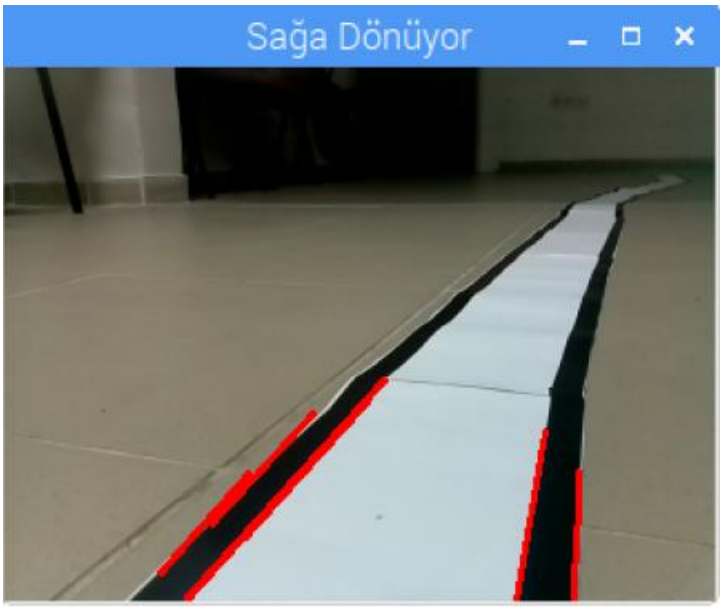
$$Sol\ şerit\ eğimi = \frac{y_2 - y_1}{x_2 - x_1} \tag{6.2}$$

(Denklem 6.2)'de sol şeridin eğimi sıfırdan küçüktür. Sol şerit algılanmıştır.



Şekil 6.10. Şerit algılaması ileri gitme

Şekil 6.10.'da görüntü işleme yöntemi ile sağ şerit ve sol şerit algılanmıştır. Bu iki şerit algılanması ile aracın düz gitmesi belirlenmiştir. Aracımızın görüntü işleme ile şekil 6.11'deki gibi her iki şeridinde eğimi (Denklem 6.2)'deki gibi sıfırdan küçük çıkarsa araç sağa dönecektir. Böylece araç şerit takibini yapmaya devam edecektir.



Şekil 6.11. Şerit algılaması sağa dönme

Aracımızın görüntü işleme ile şekil 6.12'deki gibi her iki şeridinde eğimi (Denklem 6.1)'deki gibi sıfırdan büyük çıkarsa araç sola dönecektir. Böylece araç şerit takibini yapmaya devam edecektir [20,21,22,27].

Şekil 6.12. Şerit algılaması sola dönme

6.2.2. Trafik ışıklarının algılanması

Trafik ışıklarını algılamak için rengin uyumlu aralıkları bilinmesi gerekmektedir. Bu uyumlu aralıklar elde edilince gauss filtresinden geçirilerek küçük bir alan yani gürültü oluşturan pikselleri ortadan kaldırmaktadır. Gauss filtresinin ardından morfolojik dönüşümler yapılmaktadır. Bu dönüşümler ise dilate komutu aşınma işlemi yapmaktadır. Bu işlem sayesinde görüntünün bir kez daha gürültüden arındırılma işlemi yapılmıştır. Gauss ile dilate komutu ile gürültülerden kurtulduk. Şimdi bizim için erode komutu ile genişleme işlemi yapacağız. Bu işlem gürültü olmayan asıl algılanması gereken yerlerde genişletme işlemi yapılmaktadır.

Kullanılacak yöntemler:

1. Renk Uzayının Belirlenmesi
2. Gauss Filtresi
3. Morfolojik Dönüşümler
4. Hough Dönüşümü ile Çember Tespiti

6.2.2.1. Yeşil rengin algılanması

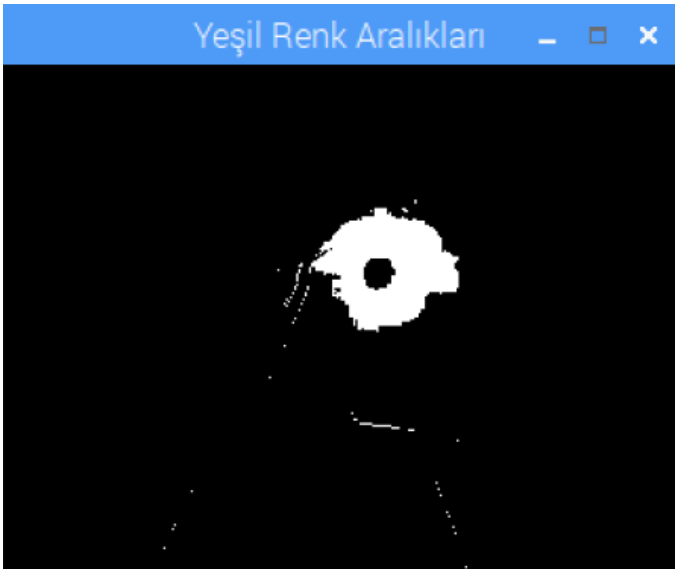
Trafik ışıkları yeşil renk için kameradan algılanan ilk görüntü (320,240) boyutlarında şekil 6.13.'de gösterilmiştir.

Şekil 6.13. İlk kamera görüntüsü yeşil için

Numpy kütüpanesi temel bir matematiksel veriler için python'da önemli bir kütüphanedir. Python'da import numpy as np yapılarak, numpy kütüphanesi np ile isimlendirilmiştir. Yeşil rengin belirlenmesi için belirli aralıklar girilmesi gerekmektedir.

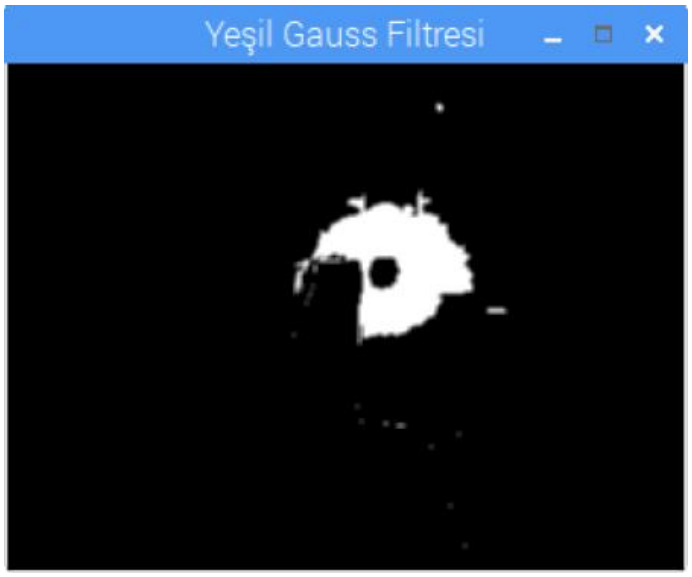
Yeşil rengin belirlenmesi için; Alt deger=np.array([0,120,0]), Üst deger=np.array([100,255,100])

cv2.inRange(frame,Alt deger,Üst deger) ile yeşil renk aralıklarını girdik.



Şekil 6.14. Yeşil rengin algılanması

Şekil 6.14. gauss renk filtresinden geçirtilerek resmin üzerindeki gürültüyü kaldırılarak şekil 6.15. elde edilmiştir.



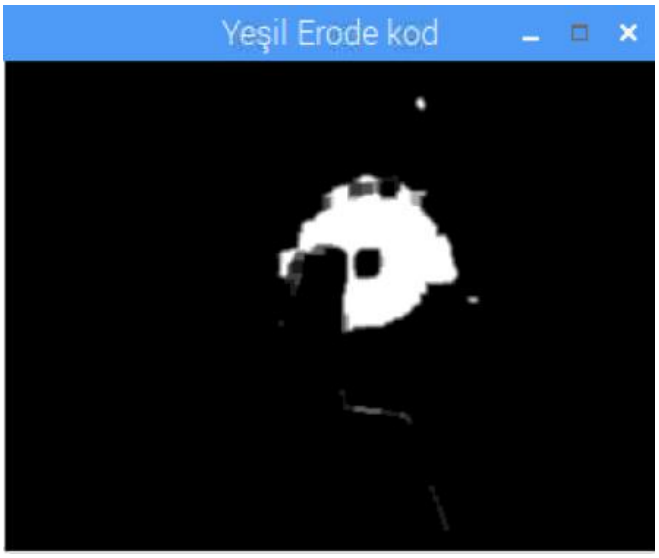
Şekil 6.15. Yeşil renk için gauss filtresi

Python'da cv2.GaussianBlur(filtre2,(3,3),2) komutu ile şekil 6.15. elde edilmiştir. Morfolojik dönüşümler, görüntü şekline dayalı kolay bir işlemdir. Normalde ikili görüntülerde gerçekleşir. İki verinin girişini ihtiyaç duyulur. İki temel morfolojik operatör ile aşındırıcı ve genişletme filtresi uygulanır. Aşındırma işlemi python'da

cv2.dilate(filtre2,np.ones((5,5),np.uint8)) ile gösterilmektedir. Genişleme işleminde python'da cv2.erode(filtre2,np.ones((5,5),np.uint8)) ile gösterilmektedir.

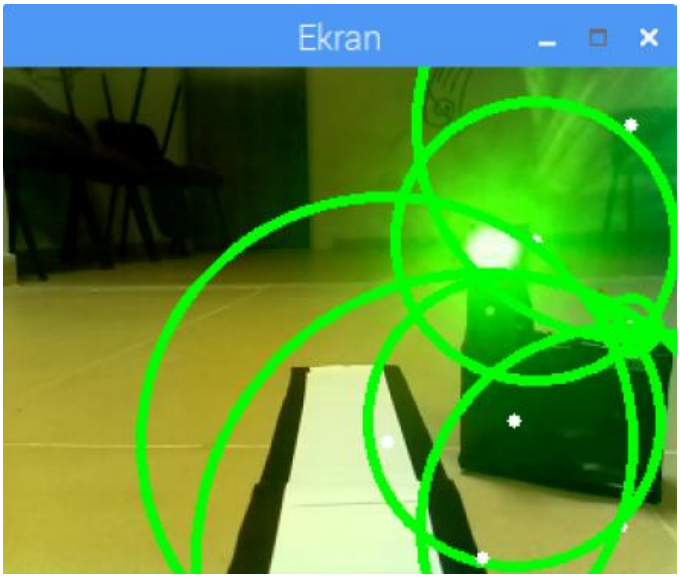
Şekil 6.16. Yeşil rengin algısı için morfolojik dönüşüm dilate

Python'da cv2.dilate(filtre2,np.ones((5,5),np.uint8)) komutu ile şekil 6.16. elde edilmiştir. Burada yeşil rengin algılanmasında küçük gürültüleri ordadan kaldırmak için aşıdırma işlemi yapılmıştır.



Şekil 6.17. Yeşil rengin algısı için morfolojik dönüşüm erode

Python'da cv2.erode(filtre2,np.ones((5,5),np.uint8)) komutu ile şekil 6.17. elde edilmiştir. Yeşil rengin algılanmasında genişletme yaparak rengin algılanması kolaylaştırılır. Ekranda yeşil rengin algılandığını belirtmek için python'da cv2.HoughCircles() komutu ile daire çizmektedir. Opencv'de zorlu bir yöntemidir.

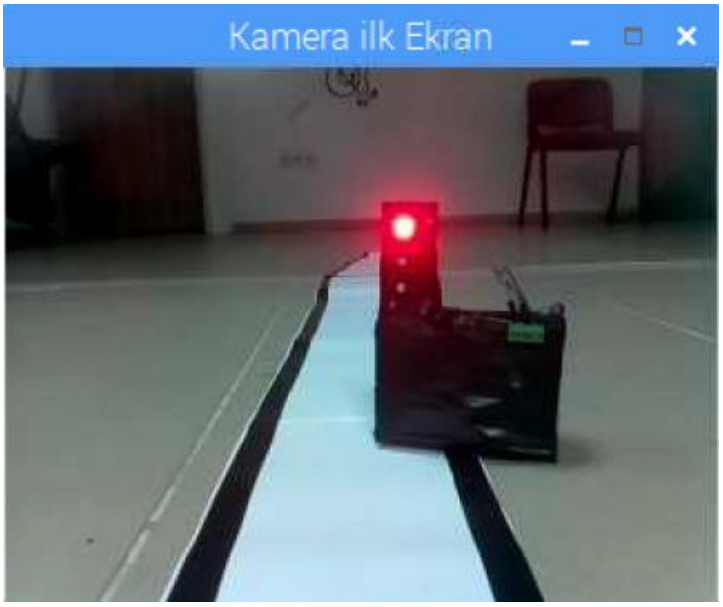


Şekil 6.18. Yeşil rengin algılanması

circless = cv2.HoughCircles(filtre2,cv2.HOUGH_GRADIENT,5,intRows/4) komutu ile şekil 6.18. elde edilmiştir.

6.2.2.2. Kırmızı ışığının algılanması

Kırmızı için kameradan algılanan ilk görüntü şekil 6.19'da gösterilmiştir. Python'da kırmızı rengin algılanması için alt ve üst değerleri girilmiştir.



Şekil 6.19. Kırmızı için ilk kamera görüntüsü

Alt deger=np.array ([60,60,100]), Üst deger=np.array ([95,95,255]

cv2.inRange(frame,alt deger,ust deger) ile kırmızı renk aralıklarını girdik. Verdiğimiz bu aralıklar ile şekil 6.20.'de kırmızının algılandığı alanlar gösterilmiştir.

Şekil 6.20. Kırmızı için renk aralıklarını belirlemesi

Şekil 6.20. algılanan alanın gürültüleri ortadan kaldırmak için gauss filtresi uyguladık. Sonuç olarak şekil 6.21. python'da cv2.GaussianBlur(filtre,(3,3),2) yazılarak elde edilmiştir.

Şekil 6.21. Kırmızı renk için gauss filtresi

Şekil 6.21.'deki görüntünün aşındırma yapılarak küçük alandaki piksellerin yok edilmesi şekil 6.22.'deki gibidir. Python'da cv2.dilate(filtre1,np.ones((5,5),np.uint8)) komutu ile yapılmıştır.

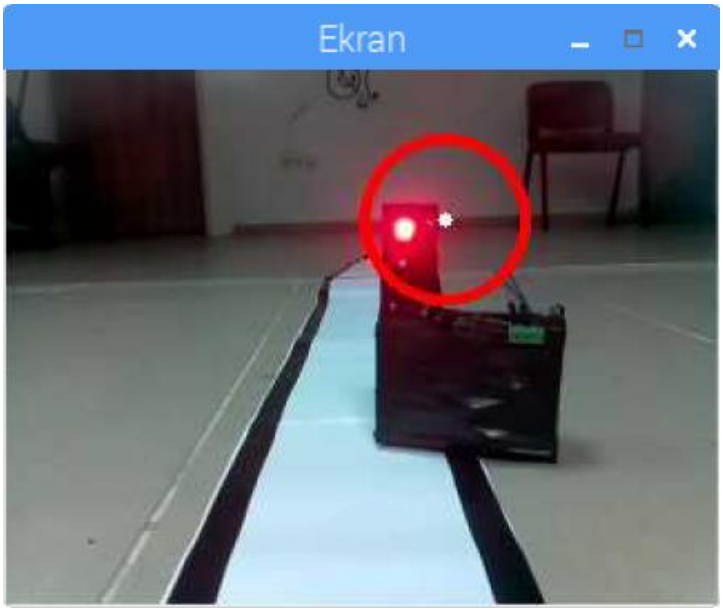
Şekil 6.22. Kırmızı renk için morfolojik dönüşüm dilate

Şekil 6.22'de aşındırma işlemi yaptık. Küçük çaptaki pikseller yok edildi. Şimdi bizim için asıl algılanma bölgesinin genişletme yapılarak kırmızı rengin daha iyi algılama yapılabilir. Şekil 6.23'de resmin genişletme işlemi yapılmıştır.

Python'da cv2.erode(filtre1,np.ones((5,5),np.uint8)) komutu ile elde edilir.

Şekil 6.23. Kırmızı renk için morfolojik dönüşüm erode

Ekranda kırmızı rengin algılandığını belirtmek için python'da cv2.HoughCircles() komutu ile daire çizmektedir.



Şekil 6.24. Kırmızı rengin algılanması

circles = cv2.HoughCircles(filtre1,cv2.HOUGH_GRADIENT,5,intRows/4) komutu ile kırmızı renk filtrelemeyi çember içine almıştır. Şekil 6.24.'de gibi kımızının algılandığı gözlenmiştir [23,26,27].

6.2.3. Trafik tabela tanımlama

Tabela tanımlama işlemi de mavi bir yüzeyde trafik işaretlerini tanımlamak için, tabelanın mavi renginin algılanması gerekmektedir. Bu algılanan alanın morfolojik dönüşüm ile algılanan tabela içi ile dışındaki gürültüleri yok etmektedir. Gürültünün yok edilmesi ile mavi yüzeyin minimum alanda algılama işlemi cv2.minAreaRect(cnt) ile yapılacaktır. Bu işlemin ardından belirlenen alanın köşeleri elde etmek cv2.boxPoints(rect) ile yapılmaktadır. Bu oluşturulan imgenin np.int0(box) kod yardımı ile oluşturularak bir dikdörtgen yüzey elde edilmiştir. Bu alanın ekranda göstermek içinse cv2.drawContours(img,[box],0,(0,0,255),2) kodunu kullanmaktayız. Bu elde edilen alanın içerisindeki trafik işaretini tanımlama işlemi yapmak için tabelanın uzaklığı ve yakınlığı ile ilgili imutils.perspective kütüpanesini kullanmaktayız. Bu alan ise değişen boyutlarda algılama işlemi gerçekleştirilmiştir. Bu elde edilen boyutların bazı matematiksel işlemler yapılmaktadır. Bu işlemleri bir

kabul üzerine yapılmaktadır. Bu işlemler Tabela sağa dönmede detaylı anlatım yapılmıştır. Yapılan bu işlemler ardından tabelanın algılanması durumunda şerit seçimi yapacaktır. Tabela tanımlandıktan sonra şerit takibi için gri tona dönüşüm yapmaktadır. Bunun ardından gauss filtresi ile gürültülerin ortadan kaldırılır. Gauss filtresinin ardından canny kenar bulma yapılmaktadır. Daha sonra algılanması için gerekli alanı belirleriz. Bu alanda kalan çizgiler algılandığında şerit için istenilen çizgiler elde edilir. Daha sonra tabelada istenilen yöne gitmesi tabelanın algılama süresi kadardır. Tabela görüş alanından çıkınca eski şerit takibine devam etmektedir.

Tabela tanımlama için kullanılan yöntemler:

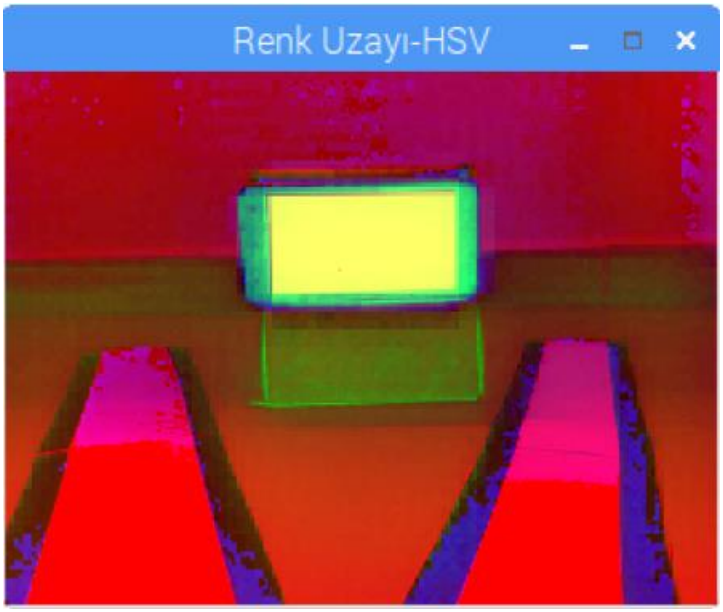
1. HSV Renk Uzayı
2. Mavi Rengin Maskelenmesi
3. Morfolojik Dönüşümler
4. Tabela Belirleme İşlemleri

6.2.3.1. Mavi trafik tabelası için renginin algılanması

Şekil 6.25.'de mavi renk için kamerada algılanan ilk görüntüdür.

Şekil 6.25. Kamera ilk görüntü mavi için

Python'da cv2.cvtColor(frame, cv2.COLOR_BGR2HSV) komutu ile şekil 6.26. elde edilmiştir. HSV renk uzayı için Heu aralığı (0:179), doygunluk aralığı (0,255) ve değer aralığı (0,255)'dir.



Şekil 6.26. Mavi renk HSV

Python'da mavi rengin algılanması için alt ve üst değerleri girerek mavi rengin algılanması sağlanmıştır.

Alt değer = np.array([90,200,50])
Üst değer = np.array([110,255,255])

Mavi_renk= cv2.inRange(hsv, Alt değer, Üst değer) bu kodlar ile şekil 6.27. elde edilmiştir.

Şekil 6.27. Mavi renk algılaması

Mavi rengin sınırlayıcı bir alana dikdörtgen ile minimum alana incelemek için python'da cv2.minAreaRect(cnt) komutu kullanmaktayız. Yalnız dört köşeli dikdörtgen olması için python'da cv2.boxPoints(rect) komutu kullanmaktayız. Algılanan bu dikdörtgeni bir imgeye dönüştürmek için python'da np.int0(box) komutu kullanmaktayız. Bu imgeyi ekranda görebilmek için cv2.drawContours(img,[box],0,(0,0,255),2) komutu kullanmaktayız. Bu durum şekil 6.28'de gösterilmiştir.

Şekil 6.28. Mavi renk çerçevelenmesi

6.2.3.2. Trafik tabela sağa dönme

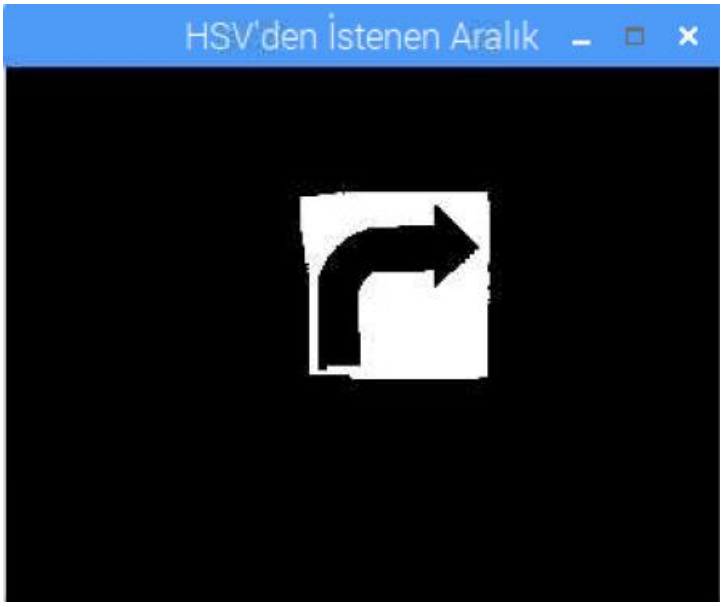
Tabela tanımlama için görüntü işleme yöntemleri ile yapacağız. Şekil 6.29.'deki tabela tanımlama için başlangıçta elde edilen görüntüdür.

Şekil 6.29. Tabela ilk görüntü

Kameradan algılanan görüntülerinin şekil 6.29.'daki gibi HSV renk uzayı görünümü şekil 6.30. elde edilmiştir. Python'da cv2.cvtColor(frame, cv2.COLOR_BGR2HSV) komutu ile elde edilmektedir.

Şekil 6.30. Tabela için HSV renk uzayı

Şekil 6.30.'daki görüntünün renk uzayı belirlenmişti. Bu renk uzayının istediğimiz mavi yüzeyin algılanması şekil 6.31.'deki gibidir.



Şekil 6.31. Tabela için mavi rengin maskelenmesi

Mavi rengin algılanması için alt ve üst değerlerini belirlemiştik. Şekil 6.31.'deki gibi algılanacaktır. Şimdi algılanan mavi rengin dışında bulunan alanın gürültüyü yok etmek için cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel) komutunu kullanırız. Daha sonra mavi rengin bulunduğu alandaki gürültülerin ortadan kaldırmak için cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel) komutunu kullanırız. Böylelikle tabela tanımlamak için daha kararlı sonuçlara ulaşırız. Bu durum şekil 6.32.'de tüm gürültüler ortadan kaldırılmış olur.

Şekil 6.32. Tabela için morfolojik dönüşümler

Mavi rengin sınırlayıcı bir alana dikdörtgen ile minimum alana incelemek için python'da cv2.minAreaRect(cnt) kullandık. Yalnız dört köşeli dikdörtgen olması için python'da cv2.boxPoints(rect) kullanmalıyız. Algılanan bu dikdörtgeni bir imgeye dönüştürmek için python'da np.int0(box) komutu kullandık. Bu imgeyi ekranda görebilmek için cv2.drawContours(img,[box],0,(0,0,255),2) komutu kullanmaktayız. Böylelikle tabela tanımlama için gerekli alanı elde ettik. Bu durum şekil 6.33.'de gösterilmiştir.

Şekil 6.33. Tabela algılama

Kameradan algılanan trafik tabelası mesafeye ve video boyutlarına bağlı olarak trafik tabelasının boyutlarını algılanması ve değişebilir olması için from imutils.perspective import four_point_transform kütüphanesini pythonda belirterek four_point_transform(mask, [largestRect][0]) komutu ile şekil 6.34. elde edilmiştir.

Şimdi şekil 6.34. elde edilen tabela görüntüsü mavi yüzeyi beyaz, beyaz trafik işareti siyah algılanmıştır. Şimdide tabela algılama için mavi yüzeyi siyah, beyaz trafik işaretini beyaz yapmak için cv2.bitwise_not(image) görüntü işleme yapıldı. Daha sonra tabelanın uzaklığına bağlı ve video akışının boyutlarına bağlı olarak (x,y)= np.divide(image.shape, 10) komutunu uygulayarak 2 tane veri (x,y) elde edilmektedir. Bu değerler tabelanın yaklaşmasıyla artmaktadır. Tabelanın uzaklaşmasıyla bu iki değer azalmaktadır. Bu elde edilen x ve y değerleri matris boyutlarını vermektedir. Resmin hangi boyutlarda algıladığını göstermektedir. Bu elde edilen iki değerin (x,y) için, dört farklı blokta inceleyeceğiz.

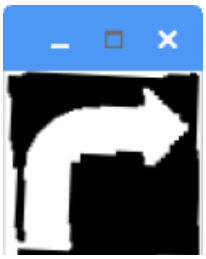
Sol blok =Tabela[4*x:9*x, y:3*y]
Merkez blok = Tabela [4*x:9*x, 4*y:6*y]
Sağ blok = Tabela [4*x:9*x, 7*y:9*y]
Üst blok = Tabela [2*x:4*x, 3*y:7*y]

Bu bölümdeki renk eşik değerlerine göre tabela işaretlerini yönünü anlamamızı sağlar. Şimdi sol blok, sağ blok, üst blok, merkez bloklarda oluşan matrislerin np.sum() ile içindeki (x,y) matrislerinin renk eşik değerlerini değerleri toplanır. Toplanan bu değerler sol blok.shape[0] ile sol blok.shape[1] değerlerini çarpımı yapılmaktadır. Bu değerler matris boyutlarıdır. Yani incelenen bölümlerin boyutlarıdır. Her bölümün matrislerinin renk eşik değerlerini toplarsak, toplanan bu değeri matrisin boyutlarına bölersek o bölgenin genel olarak renk eşik değerini elde ederiz. Bu işlemler pythonda aşağıdaki gibi yapılmaktadır.

Sol kısmı = np.sum(Sol blok)/(sol blok.shape[0]*sol blok.shape[1])
Merkez kısmı = np.sum(Merkez blok)/(Merkez blok.shape[0]* Merkez blok.shape[1])
Sağ kısmı = np.sum(Sağ blok)/(Sağ blok.shape[0]* Sağ blok.shape[1])
Üst kısmı = np.sum(Üst blok)/(Üst blok.shape[0]* Üst blok.shape[1])

Bu elde edilen değerler,
Segment=[Sol kısım, Merkez kısım, Sağ kısım, Üst kısım] ile bir dizi oluşturuldu. Oluşturulan bu dizinin içindeki ortalama renk eşik değerlerini değerleri 150'den büyük ise 1 değeri verilir. Eğer küçük ise 0 değeri verilmektedir. Bu işlemi yapan komut değeri aşağıdaki gibidir.

segments = tuple(1 if segment > 150 else 0 for segment in segments) sağ tabela için bu işlem gerçekleştirildiğinde segments=(1,0,0,1) değeri elde edilir [24].



Şekil 6.34. Tabela sağa algılandı

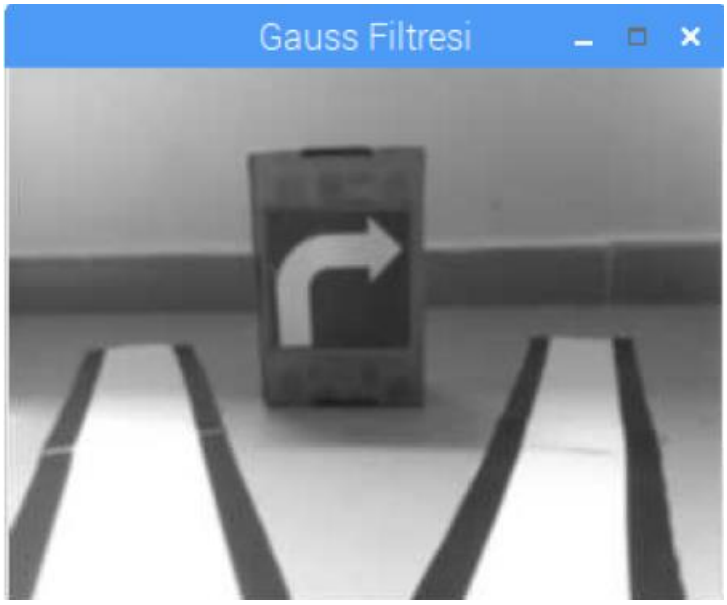
Tabela algılama fonksiyonunda, tabela.txt dosyası açılmaktadır. Açılan bu dosyada "1001" ifadesi yazılmaktadır. Yazılan bu ifade main fonksiyonunda tabela.txt dosyasının açılıp okunmaktadır. Tabela kontrol ifadesi aktifleşmektedir. Tabela

kontrol aktifleşince tabela okuma ifadesi aktifleşmektedir. Tabela okuma aktifleşince içerisinde while döngüsü yardımı ile tabela.txt içindeki bilgi alınır ama bu işlem tabelayı algıladıktan sonra sadece bir kere çalışan bir döngüdür. Main fonksiyonunda sağa dönme komutu “1001” ile anlamaktadır. Bu değeri için şerit belirleme yapacağız. Tabelanın sağa olduğunu algılandığında şerit belirlemek için gri ton dönüşüm yapılmaktadır.

Uygulanan yazılımda cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)’dir. Oluşan imgede şekil 6.35’de gösterilmiştir.

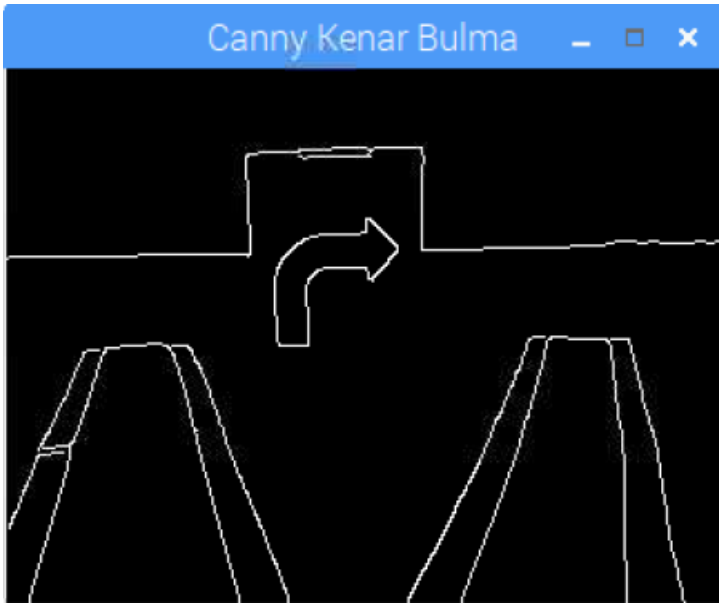
Şekil 6.35. Tabela için şerit belirleme gri ton

Şekil 6.35.’deki gri ton dönüşümü yapıldı. Şerit belirlemesi için gauss filtresi ile görüntünün gürültüleri yok edilir ve görüntünün düzleştirilmesi şekil 6.36.’de gösterilmiştir. Uygulanan yazılım cv2.GaussianBlur(imgGrayscale,(5,5),0) komutu kullanılmıştır.



Şekil 6.36. Tabela için şerit belirleme gauss filtresi

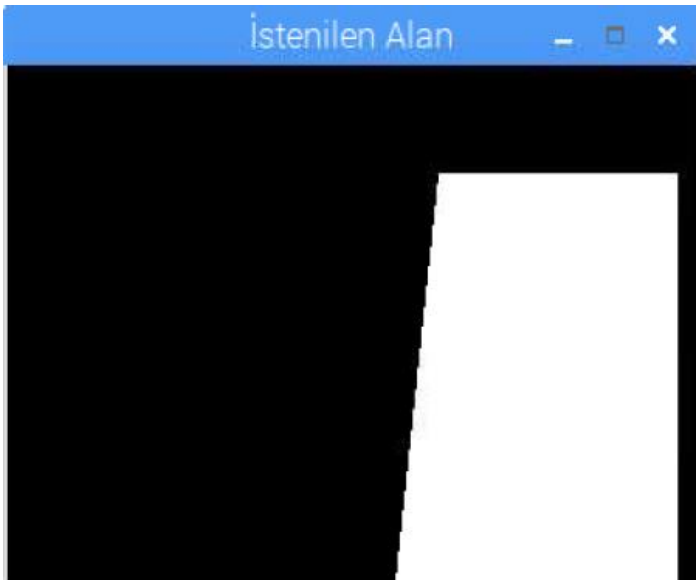
Şekil 6.36.'de şerit için gauss filtresi yapmıştık. Canny kenar bulma yöntemi ile şekil 6.37'de elde edilmiştir. Uygulan yazılımda cv2.Canny(imgBlurred,0,255)'dır.



Şekil 6.37. Tabela için şerit belirleme canny kenar bulma

Şekil 6.37'deki gibi canny kenar bulma yöntemi ile kenarları elde etmiştik. Elde edilen bu kenarlar, şekil 6.38.'daki istenilen alanda gösterilecektir. İstenilen alan ise (320,240) için geçerlidir.

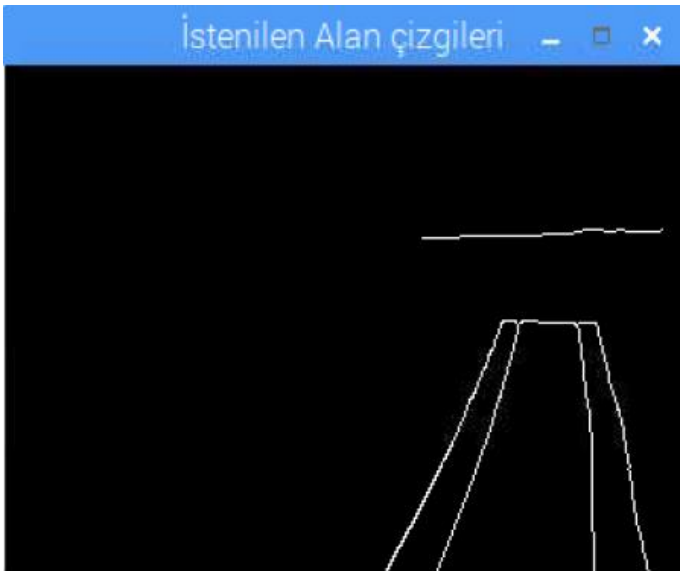
vertices = np.array([[(180,240),(200,50),(310,50),(310,240)]],np.int32)



Şekil 6.38. Tabela için şerit belirleme istenilen alan

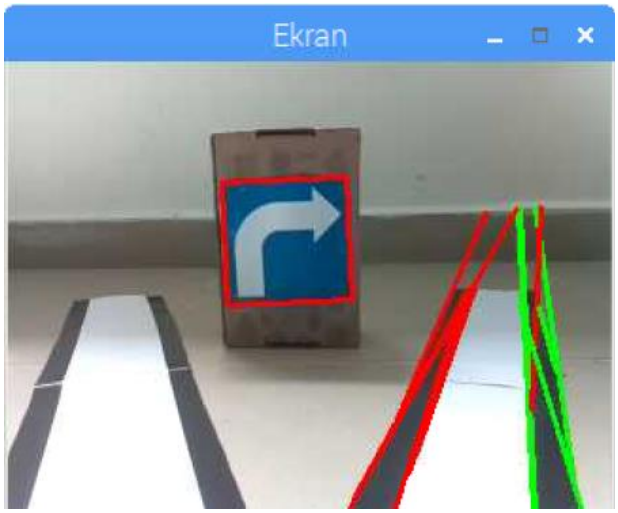
Canny kenarları şekil 6.37'de, şekil 6.38. istenilen alandan filtreleme yaparak şekil 6.39. elde edilmiştir. Cv2.bitwise_and komutu ile sadece belirlediğimiz o alan ile ve kapısından geçeceğinden belirlediğimiz alanda kalmaya devam eder.

masked = cv2.bitwise_and(imgCanny, mask) yazılım kodu ile elde edilmiştir.



Şekil 6.39. Tabela için şerit belirleme istenilen alan canny çizgileri

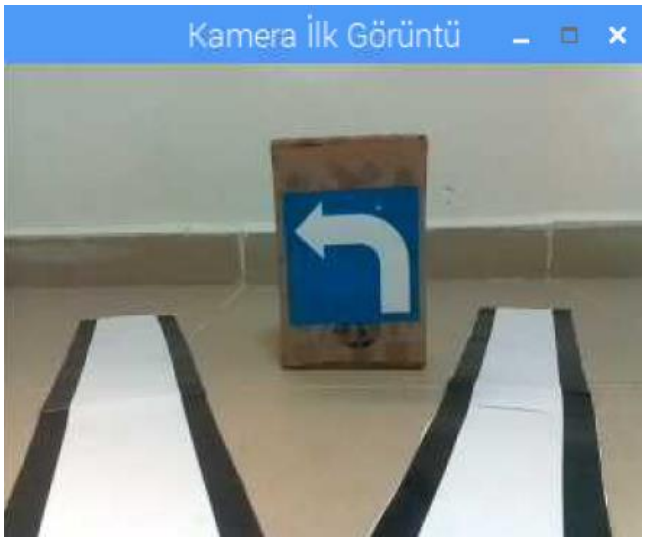
Şekil 6.39.'de istenilen alan ile elde edilen kenarları, şekil 6.40.'de gösterilmiştir. Şekil 6.40.'de elde edilen kenarları (Denklem 6.1) ile (Denklem 6.2.) ile filtreleyerek, aracın sağa dönmesi sağlanmıştır.



Şekil 6.40. Tabela için şerit belirleme sağa dönme

6.2.3.3. Trafik tabela sola dönme

Tabela tanımlama için görüntü işleme yöntemleri ile yapacağız. Şekil 6.41.'deki tabela tanımlama için başlangıçta elde edilen görüntüdür [20,21,22].

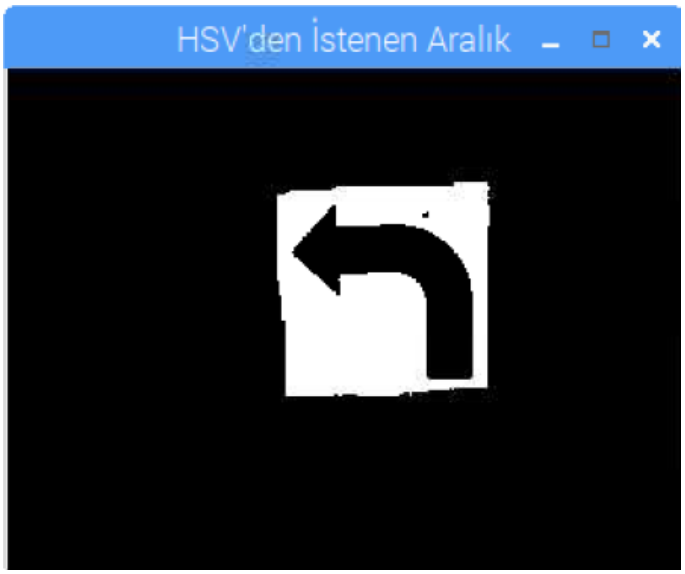


Şekil 6.41. Tabela ilk görüntü sol

Kameradan elde edilen ilk görüntü şekil 6.41.'daki HSV renk uzay görünümü şekil 6.42'deki gibi elde edilmiştir. Python'da cv2.cvtColor(frame, cv2.COLOR_BGR2HSV) komutu ile elde edilmektedir.

Şekil 6.42. Tabela HSV renk aralığı sol

Şekil 6.42.'daki görüntünün renk uzayı belirlenmişti. Bu renk uzayının istediğimiz mavi yüzeyin algılanması şekil 6.43.'deki gibidir.



Şekil 6.43. Tabela için mavi rengin maskelenmesi sol

Mavi rengin algılanması için alt ve üst değerlerini belirlemiştik. Şekil 6.43.'deki gibi algılanacaktır. Şimdi algılanan mavi rengin dışında bulunan alanın gürültüyü yok etmek için cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel) komutunu kullanırız. Daha sonra mavi rengin bulunduğu alandaki gürültülerin ortadan kaldırmak için cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel) komutunu kullanırız. Böylelikle tabela tanımlamak için daha kararlı sonuçlara ulaşırız. Bu durum şekil 6.44.'de tüm gürültüler ortadan kaldırılmış olur.

Şekil 6.44. Tabela için morfolojik dönüşümler sol

Mavi rengin sınırlayıcı bir alana dikdörtgen ile minimum alana incelemek için python'da cv2.minAreaRect(cnt) kullandık. Yalnız dört köşeli dikdörtgen olması için python'da cv2.boxPoints(rect) kullanmalıyız. Algılanan bu dikdörtgeni bir imgeye dönüştürmek için python'da np.int0(box) komutu kullandık. Bu imgeyi ekranda görebilmek için cv2.drawContours(img,[box],0,(0,0,255),2) komutu kullanmaktayız. Böylelikle tabela tanımlama için gerekli alanı elde ettik. Bu durum şekil 6.45.'deki gibi gösterilmiştir.

Şekil 6.45. Tabela algılama sol

Tabela sağa dönmede yapılan işlemler ve detaylı anlatım tabelanın sola dönmede için geçerlidir. Şekil 6.46. algılandıktan sonra segments=(0,0,1,1) değeri elde edilir.

Şekil 6.46. Tabela sola algılandı

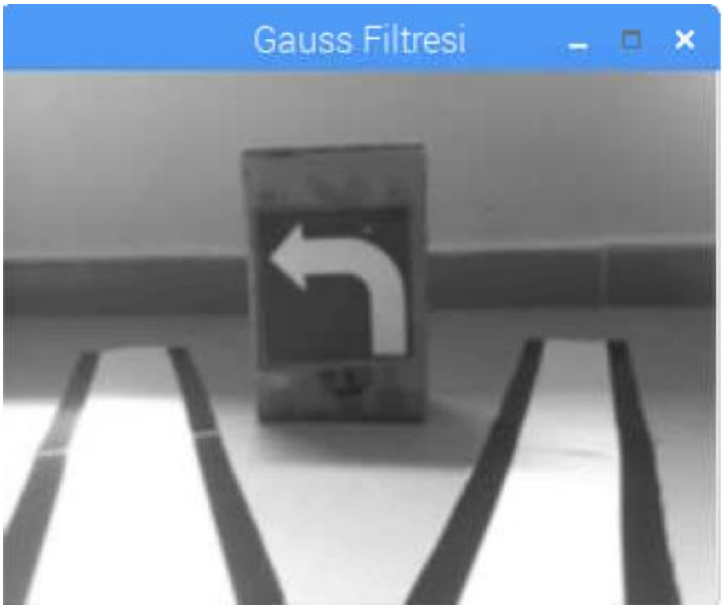
Tabela sağa dönme main fonksiyonunda gerçekleşen durumlar sola dönme içinde geçerlidir. Daha sonra main fonksiyonunda sola dönme komutu “0011” ile anlaşılmaktadır. Bu elde edilen değer ile şerit takibi yapacağız [24].

Tabelanın sola olduğunu algılandığında şerit belirlemek için gri ton dönüşüm yapılmaktadır.

Uygulanan yazılımda cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)'dir.

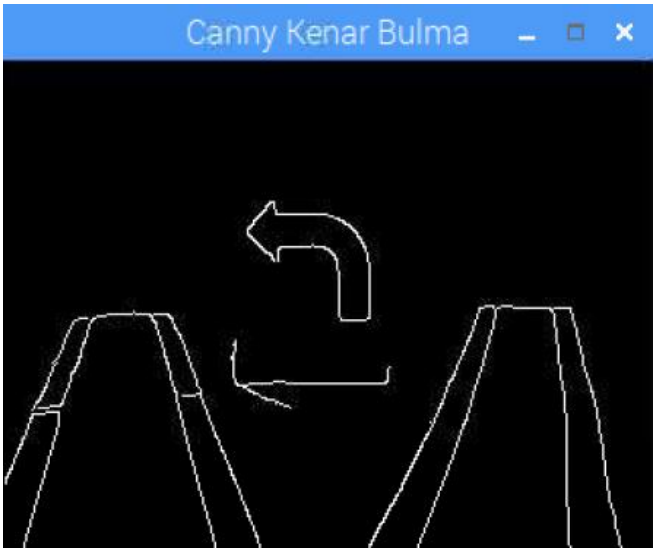
Şekil 6.47. Tabela için şerit belirleme gri ton sol tabela

Şekil 6.47.'deki gri ton dönüşümü yapıldı. Şerit belirlemesi için gauss filtresi ile görüntünün gürültüleri yok edilir ve görüntünün düzleştirilmesi şekil 6.48.'de gösterilmiştir. Uygulanan yazılım cv2.GaussianBlur(imgGrayscale,(5,5),0) komutu kullanılmıştır.



Şekil 6.48. Tabela için şerit belirleme gauss filtresi sol tabela

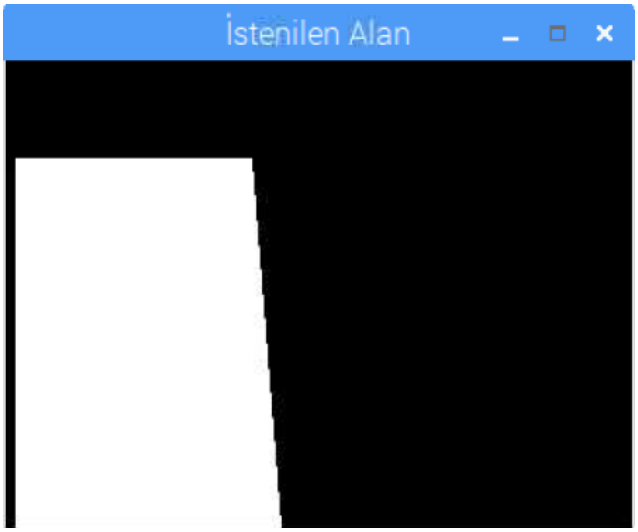
Şekil 6.47.'de şerit için gauss filtresi yapmıştık. Canny kenar bulma yöntemi ile şekil 6.49'de elde edilmiştir. Uygulan yazılımda cv2.Canny(imgBlurred,0,255)'dır.



Şekil 6.49. Tabela için şerit belirleme canny kenar belirleme

Şekil 6.49'deki gibi canny kenar bulma yöntemi ile kenarları elde etmiştik. Elde edilen bu kenarlar, şekil 6.50.'daki istenilen alanda gösterilecektir. İstenilen alan ise (320,240) için geçerlidir.

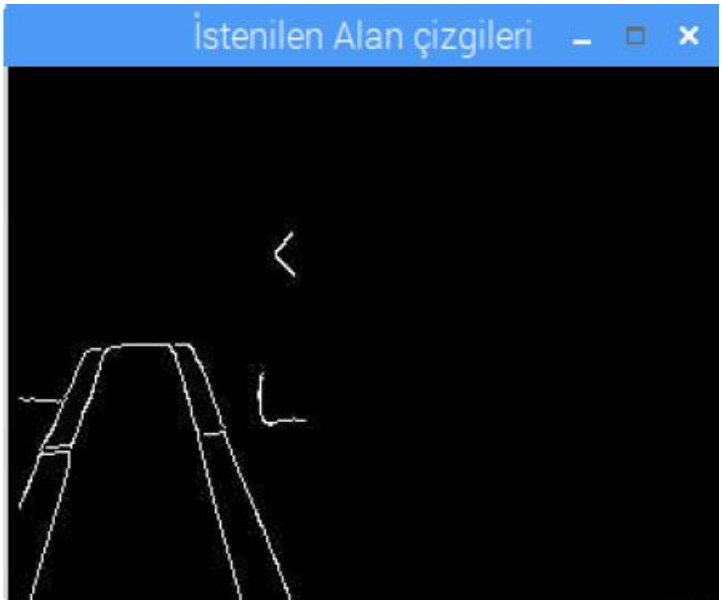
vertices =np.array([[(5,240),(5,50),(125,50),(140,240)]],np.int32)



Şekil 6.50. Tabela için şerit belirleme istenilen alan

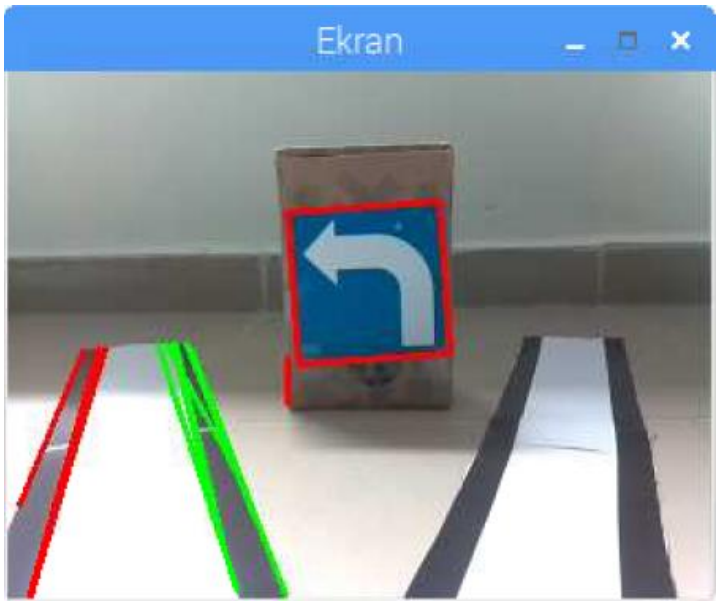
Canny kenarları şekil 6.49'de, şekil 6.50. istenilen alandan filtreleme yaparak şekil 6.51. elde edilmiştir. Cv2.bitwise_and komutu ile sadece belirlediğimiz o alan ile ve kapısından geçeceğinden belirlediğimiz alanda kalmaya devam eder.

masked = cv2.bitwise_and(imgCanny, mask) yazılım kodu ile elde edilmiştir.



Şekil 6.51 Tabela için şerit belirleme istenilen alan canny çizgileri

Şekil 6.51.'de istenilen alan ile elde edilen kenarları, şekil 6.52.'de gösterilmiştir. Şekil 6.52.'de elde edilen kenarları (Denklem 6.1) ile (Denklem 6.2.) ile filtreleyerek, aracın sağa dönmesi sağlanmıştır [20,21,22].



Şekil 6.52. Tabela için şerit belirleme sola dönme

6.2.3.4. Trafik tabela için düz ilerleme

Tabela tanımlama için görüntü işleme yöntemleri ile yapacağız. Şekil 6.53.'deki tabela tanımlama için başlangıçta elde edilen görüntüdür.

Şekil 6.53. Tabela ilk görüntü ileri gitme

Kameradan algılanan görüntülerinin şekil 6.53'daki gibi HSV renk uzayı görünümü şekil 6.54. elde edilmiştir. Python'da cv2.cvtColor(frame, cv2.COLOR_BGR2HSV) komutu ile elde edilmektedir.

Şekil 6.54. Tabela için HSV renk uzayı

Şekil 6.54.'daki görüntünün renk uzayı belirlenmişti. Bu renk uzayının istediğimiz mavi yüzeyin algılanması şekil 6.55.'deki gibidir.

Şekil 6.55. Tabela için mavi rengin maskelenmesi ileri gitme

Mavi rengin algılanması için alt ve üst değerlerini belirlemiştik. Şekil 6.55.'deki gibi algılanacaktır. Şimdi algılanan mavi rengin dışında bulunan alanın gürültüyü yok etmek için cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel) komutunu kullanırız. Daha sonra mavi rengin bulunduğu alandaki gürültülerin ortadan kaldırmak için cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel) komutunu kullanırız. Böylelikle tabela tanımlamak için daha kararlı sonuçlara ulaşırız. Bu durum şekil 6.56.'de tüm gürültüler ortadan kaldırılmış olur.

Şekil 6.56. Tabela için morfolojik dönüşümler iler gitme

Mavi rengin sınırlayıcı bir alana dikdörtgen ile minimum alana incelemek için python'da cv2.minAreaRect(cnt) kullandık. Yalnız dört köşeli dikdörtgen olması için python'da cv2.boxPoints(rect) kullanmalıyız. Algılanan bu dikdörtgeni bir imgeye dönüştürmek için python'da np.int0(box) komutu kullandık. Bu imgeyi ekranda görebilmek için cv2.drawContours(img,[box],0,(0,0,255),2) komutu kullanmaktayız. Böylelikle tabela tanımlama için gerekli alanı elde ettik. Bu durum şekil 6.56.'da gösterilmiştir.

Şekil 6.57. Tabela için morfolojik dönüşümler düz ilerleme

Tabela sağa dönmede yapılan işlemler ve detaylı anlatım tabelanın düz ilerleme içinde geçerlidir. Şekil 6.57.'de algıladığında segments=(0,1,0,1) değeri elde edilir.

Şekil 6.58. Tabela düz ilerleme algılandı

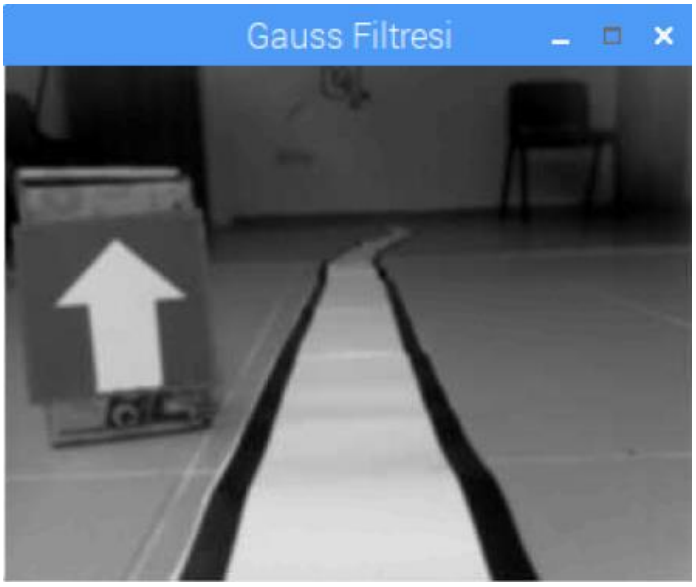
Tabela sağa dönme main fonksiyonunda gerçekleşen durumlar düz ilerleme içinde geçerlidir. Daha sonra main fonksiyonunda düz ilerleme komutu "0101" ile anlaşılmaktadır. Bu elde edilen değer ile şerit takibi yapacağız. Tabelanın geri dönme olduğunu algılandığında şerit belirlemek için gri ton dönüşüm yapılmaktadır. Şekil 6.58'deki gibidir [24] .

Uygulanan yazılımda cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)'dir.

Şekil 6.59. Tabela için şerit belirleme gri ton düz ilerleme

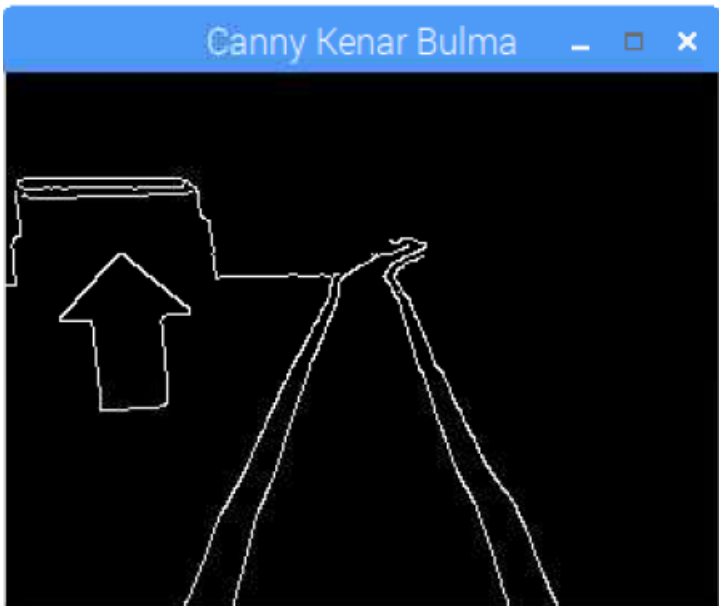
Şekil 6.58.'deki gri ton dönüşümü yapıldı. Şerit belirlemesi için gauss filtresi ile görüntünün gürültüleri yok edilir ve görüntünün düzleştirilmesi şekil 6.59.'da

gösterilmiştir. Uygulanan yazılım cv2.GaussianBlur(imgGrayscale,(5,5),0) komutu kullanılmıştır.



Şekil 6.60. Tabela için şerit belirleme gauss filtresi

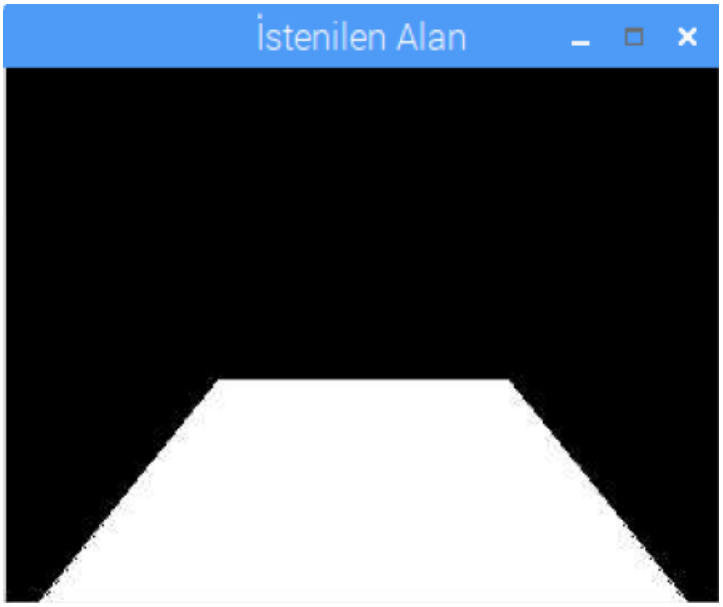
Şekil 6.59.'de şerit için gauss filtresi yapmıştık. Canny kenar bulma yöntemi ile şekil 6.60'de elde edilmiştir. Uygulan yazılımda cv2.Canny(imgBlurred,0,255)'dır.



Şekil 6.61. Tabela için şerit belirleme canny kenar belirleme

Şekil 6.60'deki gibi canny kenar bulma yöntemi ile kenarları elde etmiştik. Elde edilen bu kenarlar, şekil 6.61.'daki istenilen alanda gösterilecektir. İstenilen alan ise (320,240) için geçerlidir.

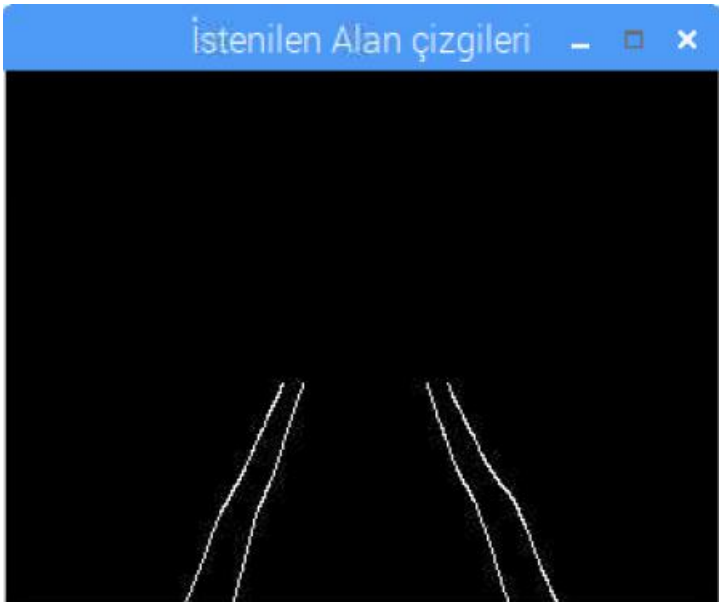
vertices = np.array([[(15,240),(95,140),(225,140),(305,240)]],np.int32)



Şekil 6.62. Tabela için şerit belirleme istenilen alan

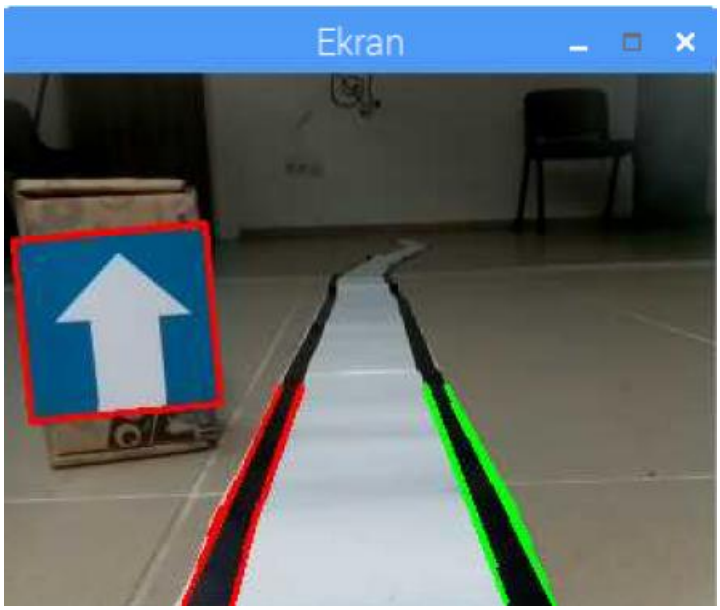
Canny kenarları şekil 6.60'de, şekil 6.61. istenilen alandan filtreleme yaparak şekil 6.62. elde edilmiştir. Cv2.bitwise_and komutu ile sadece belirlediğimiz o alan ile ve kapısından geçeceğinden belirlediğimiz alanda kalmaya devam eder.

masked = cv2.bitwise_and(imgCanny, mask) yazılım kodu ile elde edilmiştir.



Şekil 6.63. Tabela için şerit belirleme istenilen alan canny çizgileri

Şekil 6.62.'de istenilen alan ile elde edilen kenarları, şekil 6.63.'de gösterilmiştir. Şekil 6.63.'de elde edilen kenarları (Denklem 6.1) ile (Denklem 6.2.) ile filtreleyerek, aracın düz ilerlemesi sağlanmıştır [20,21,22].



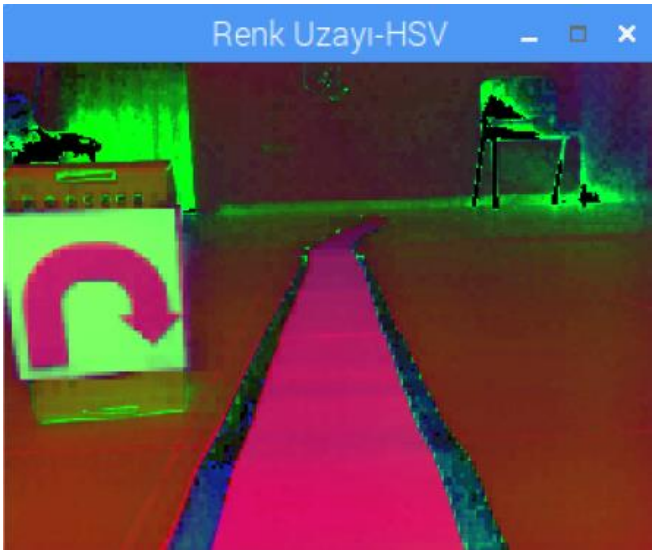
Şekil 6.64. Tabela için şerit belirleme sağa dönme

6.2.3.5. Trafik tabela için geri dönme

Tabela tanımlama için görüntü işleme yöntemleri ile yapacağız. Şekil 6.64.'deki tabela tanımlama için başlangıçta elde edilen görüntüdür.

Şekil 6.65. Tabela ilk görüntü geri dön

Kameradan elde edilen ilk görüntü şekil 6.64.'daki HSV renk uzay görünümü şekil 6.65'deki gibi elde edilmiştir. Python'da cv2.cvtColor(frame, cv2.COLOR_BGR2HSV) komutu ile elde edilmektedir.



Şekil 6.66. Tabela için HSV renk uzayı

Şekil 6.65.'daki görüntünün renk uzayı belirlenmişti. Bu renk uzayının istediğimiz mavi yüzeyin algılanması şekil 6.66.'deki gibidir.

Şekil 6.67. Tabela için mavi rengin maskelenmesi geri dön

Mavi rengin algılanması için alt ve üst değerlerini belirlemiştik. Şekil 6.66.'deki gibi algılanacaktır. Şimdi algılanan mavi rengin dışında bulunan alanın gürültüyü yok etmek için cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel) komutunu kullanırız. Daha sonra mavi rengin bulunduğu alandaki gürültülerin ortadan kaldırmak için cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel) komutunu kullanırız. Böylelikle tabela tanımlamak için daha kararlı sonuçlara ulaşırız. Bu durum şekil 6.67.'de tüm gürültüler ortadan kaldırılmış olur.

Şekil 6.68. Tabela için morfolojik dönüşümler geri dön

Mavi rengin sınırlayıcı bir alana dikdörtgen ile minimum alana incelemek için python'da cv2.minAreaRect(cnt) kullandık. Yalnız dört köşeli dikdörtgen olması için python'da cv2.boxPoints(rect) kullanmalıyız. Algılanan bu dikdörtgeni bir imgeye dönüştürmek için python'da np.int0(box) komutu kullandık. Bu imgeyi ekranda görebilmek için cv2.drawContours(img,[box],0,(0,0,255),2) komutu kullanmaktayız. Böylelikle tabela tanımlama için gerekli alanı elde ettik. Şekil 6.68'de bu durum göstermektedir.

Şekil 6.69. Tabela için morfolojik dönüşümler sol

Tabela sağa dönmede yapılan işlemler ve detaylı anlatım tabelanın geri dönmede için geçerlidir. Şekil 6.69.algılandıktan sonra segments=(1,0,1,1) değeri elde edilir.

Şekil 6.70. Tabela geri dön algılandı

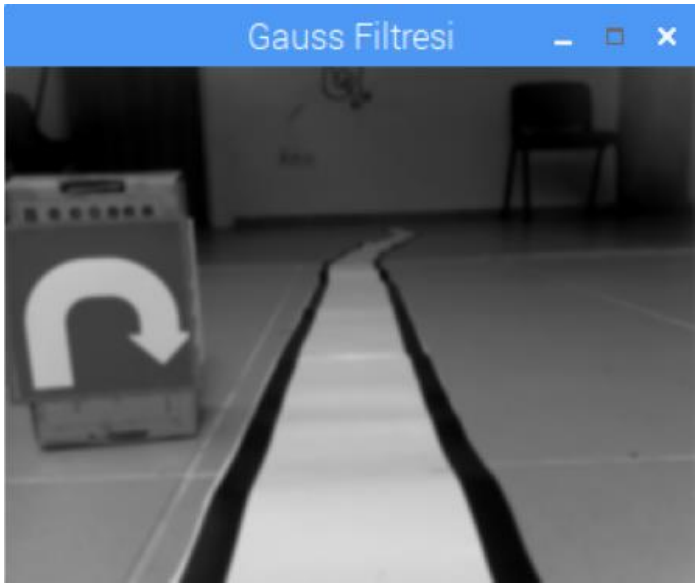
Tabela sağa dönme main fonksiyonunda gerçekleşen durumlar tabela geri dönme içinde geçerlidir. Daha sonra main fonksiyonunda geri dönme komutu “1011” ile anlaşılmaktadır. Bu elde edilen değer ile şerit takibi yapacağız [24].

Tabelanın geri dönme olduğunu algılandığında şerit belirlemek için gri ton dönüşüm şekil 6.70.’deki gibi yapılmaktadır.

Uygulanan yazılımda cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)’dir.

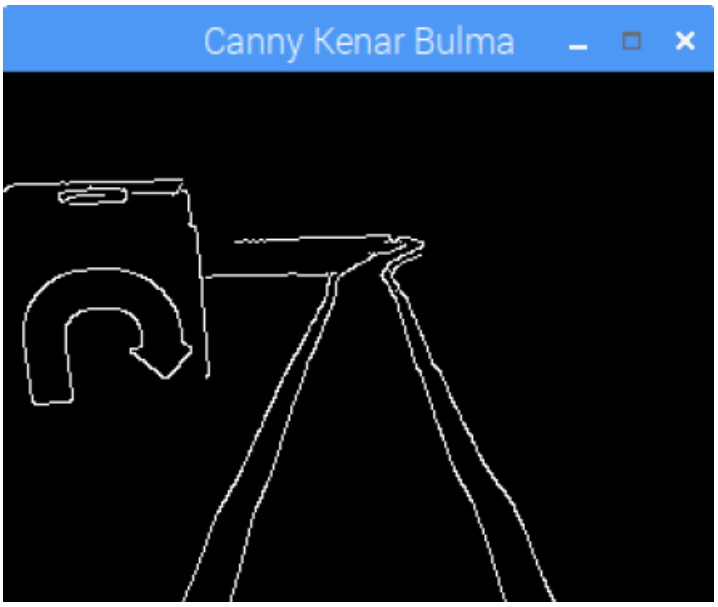
Şekil 6.71. Tabela için şerit belirleme gri ton geri dönme

Şekil 6.70.’deki gri ton dönüşümü yapıldı. Şerit belirlemesi için gauss filtresi ile görüntünün gürültüleri yok edilir ve görüntünün düzleştirilmesi şekil 6.71.’de gösterilmiştir. Uygulanan yazılım cv2.GaussianBlur(imgGrayscale,(5,5),0) komutu kullanılmıştır.



Şekil 6.72. Tabela için şerit belirleme gauss filtresi

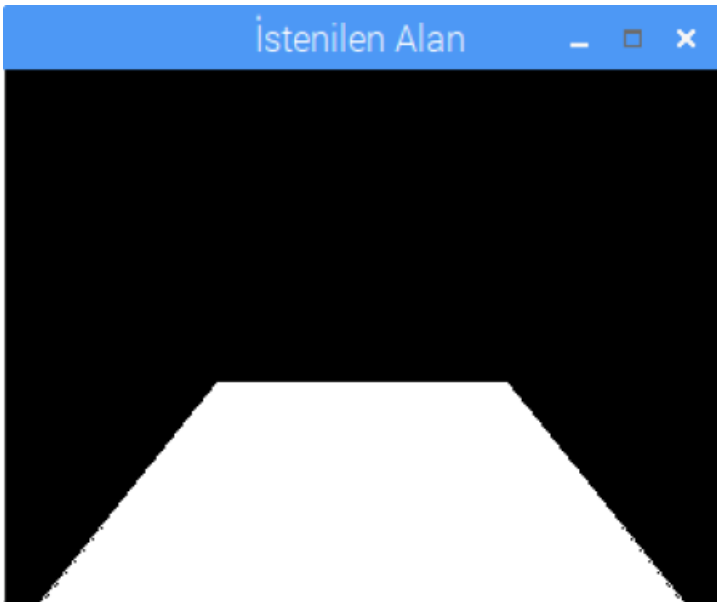
Şekil 6.71.'de şerit için gauss filtresi yapmıştık. Canny kenar bulma yöntemi ile şekil 6.72'de elde edilmiştir. Uygulanan yazılımda cv2.Canny(imgBlurred,0,255)'dır.



Şekil 6.73. Tabela için şerit belirleme canny kenar belirleme

Şekil 6.72'deki gibi canny kenar bulma yöntemi ile kenarları elde etmiştik. Elde edilen bu kenarlar, şekil 6.73.'daki istenilen alanda gösterilecektir. İstenilen alan ise (320,240) için geçerlidir.

vertices = np.array([[(15,240),(95,140),(225,140),(305,240)]],np.int32)

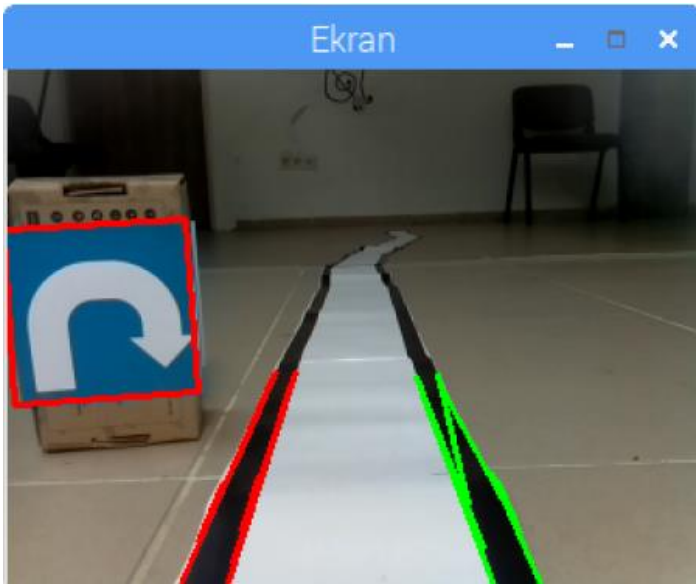


Şekil 6.74. Tabela için şerit belirleme istenilen alan

Canny kenarları şekil 6.72'de, şekil 6.73. istenilen alandan filtreleme yaparak şekil 6.74. elde edilmiştir. Cv2.bitwise_and komutu ile sadece belirlediğimiz o alan ile ve kapısından geçeceğinden belirlediğimiz alanda kalmaya devam eder.
masked = cv2.bitwise_and(imgCanny, mask) yazılım kodu ile elde edilmiştir.

Şekil 6.75. Tabela için şerit belirleme istenilen alan canny çizgileri

Şekil 6.74.'de istenilen alan ile elde edilen kenarları, şekil 6.75.'de gösterilmiştir. Şekil 6.75.'de elde edilen kenarları (Denklem 6.1) ile (Denklem 6.2.) ile filtreleyerek, aracın düz bir şekilde geri dönmesi sağlanmıştır.[20,21,22,26,27,28,29]



Şekil 6.76. Tabela için şerit belirleme

BÖLÜM 7. SONUÇ VE TARTIŞMALAR

7.1. Sonuç

Otonom araç için seçilen raspberry pi ile görüntü işleme yöntemleri şerit algılama yapılmıştır. Bu işlem için şerit algılamada çizgilerin görüntü işleme yöntemleri ile ekranda alınan bilgilerin gürültüleri ortadan kaldırılmıştır. Şerit takibi için elde edilen çizgilerin eğimlerine bakarak şeritleri tanımaktadır. Bu işlem sol şerit olması için çizgilerin eğimi sıfırdan küçük olması gerekmektedir. Şağ şerit içinde çizgilerin eğimleri sıfırdan büyük olması gerekmektedir. Araç sağ ve sol şeridi algıladığında araç şerit takibini düz gitme olarak yapacaktır. Eğer algılanan tüm çizgilerin eğimleri sıfırdan küçük olursa araç sola dönmesi gerekmektedir. Eğer algılanan tüm çizgilerin eğimi sıfırdan büyük olursa araç sağa dönecektir.

Aracımız şerit takibi yaparken tabela algılama veya trafik ışıklarını bazen algılayamamaktadır. Bu durumda mesafe sensörü yardımı ile aracımız şerit takibinde iken engel algılanınca aracımız durmaktadır. Bu nedenle tabela algılama ve trafik ışıklarını algılama aracın durgun halde iken yapması daha kolaydır. Aracımız tabela algıladıktan sonra aracın önünde tabelanın olması, engel olmaktan çıkar ve araç tabela algılamada başarılı olmuştur.

Aracımız aynı şekilde trafik lambasında engel olarak algılar ve aracımız durgun halde iken yeşil renk algılarsa aracımız hareketli hale geçecektir. Şerit takibini yapmaya devam edecektir. Araç trafik lambasını engel algılamadan dolayı dururşa, trafik lambasından kırmızı renk algılaması durumunda aracımız hareket etmeyi bırakır. Önündeki engel alışa bile aracımız durgundur. Tabela algılama yapmamaktadır. Şerit

takibi yapmamaktadır. Aracımız sadece yeşil renk algılarsa şerit takibi yapabilmekte ve tabela tanımlama yapabilmektedir.

7.2. Öneriler

Aracımız şerit takibi yaparken sadece çizgilerin eğimlerine göre karar veren bir araç tasarlanmıştır. Bu işlemi çizgilerin eğimlerinin tanjant değerlerine göre karar veren ve aracın direksiyonu servo motoru ile kontrol edilen bir araç yapılabilir. Bizim aracımız direksiyon kontrolü basit bir DC motor ile kontrol edilmektedir. Bundan dolayı DC motor boşta iken araç ileri gitmektedir. DC motorun aktifleşmesi durumunda sağa sola dönme yapmaktadır. Yani direksiyon için üç seçenemiz var. Bu durum donanısal bir sıkıntıdır.

Aracımız arka tekerleri bir DC motor ile kontrol edilir. Arka tekerler aynı anda aynı hareketi yapmaktadır. Bu nedenle aracımız keskin viraj almakta sıkıntı çekmektedir. İki farklı DC motor ile daha kontrollü olabilir. Aracımızın arka tekerleri redüktör oranı düşüktür redüktör oranı arttırılarak aracın daha yavaş gitmesini sağlayarak raspberry pi görüntü işleme esnasında daha kararlı sonuçlar alınır.

Aracımız görüntü işleme yöntemleri ile otonom araç tasarlanmıştır. Aracımız yapay sinir ağları, yapay zeka ile otonom araç yapılması daha güvenli, daha doğru sonuçlar alınması beklenir.

KAYNAKLAR

[1] Doruk, A., Şerit takibi ve hız ayarı yapabilen otonom robot araba, Conference Paper, 2016 .

[2] Kısa, M., Karayolunda seyreden araçların tanınması, Selçuk Üniversitesi, Fen Bilimleri Enstitüsü, Makine Mühendisliği Anabilim Dalı, Doktora Tezi, 2014.

[3] Küçükmanisa, A., Urhan, O., Güven, T., Gerçek zamanlı gömülü şeritten ayrılma tesbit sistemi, Conferece Paper, 2014.

[4] Küçükmanisa, A., Urhan, O., Gömülü bir platform üzerinde gerçek zamanlı şeritten ayrılma uyarı sistemi, Journal of the Faculty of Engineering and Architecture of Gazi University, 32:4,1287-1300, 2017.

[5] Oflezer, F., Akıllı araç yön tayini ile ilgili bir uygulama, Bozok Üniversitesi, Fen Bilimleri Enstitüsü, Mekatronik Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi, 2015.

[6] Ungurean, D.BC., DeepRcar an autonomous car model, Faculty of İnfermation Technology CTU Progue, Assıgnment of Masters Thesis, 2018.

[7] Satyanarayana, K.N.V., Tapasvi, B., KanakaRaju, P., RameshBabu G., Based on machine learning autonomous car using raspberry-pi, K.N.V. Satyanarayana, Int. Journal of Engineering Research and Application, ISSN : 2248-9622, Vol. 7, Issue 12, (Part -5) December 2017, pp.76-82,2017.

[8] Ooi, R.C., Balancing a two wheeled autonomous robot, The University of Western Australia School of Mechanical Engineering, 26-28, 2003.

[9] Derdiyok, A., Denetim sistemleri ders notları, Sakarya Üniversitesi Mekatronik Program, 2017.

[10] Ogata, K., Modern control engıneerıng fifth edition, İçinde :10. Bölüm Control Systems Design in State Space, Pearson Yayınevi, New York-London 746, 751,2010.

[11] Raut, P., Karjatwala, S., Kadam, S., Dynamic modelling, simulation & control design of drive & reaction wheel balancing bot, International Journal of Innovative Research in Science, Engineering and Technology, 2016.

[12] Ooi, R.C., Balancing a two-wheeled autonomous robot, The University of Western Australia School of Mechanical Engineering, 23-25, 2003.

[13] https://www.bataryam.com/nicd-5aa500-6v-500mah-sarj-edilebilir-batarya- Erişim tarihi:20.04.2019

[14] https://www.electrokit.com/en/product/dc-motor-6v-250ma-14500rpm-2/ Erişim tarihi:10.05.2019

[15] https://www.robotshop.com/en/6v-250ma-brushed-dc-motor.html Erişim tarihi:10.05.2019

[16] https://opensource.com/resources/raspberry-pi Erişim tarihi: 01.04.2019

[17] https://www.robotistan.com/hc-sr04-ultrasonik-mesafe-sensoru Erişim tarihi: 10.05.2019

[18] http://elektrikbilim.com/722-ne555-ile-trafik-lambasi-uygulamasi.html Erişim tarihi: 10.05.2019

[19] www.robotistan.com/l298n-voltaj-regulatorlu-cift-motor-surucu-karti Erişim tarihi: 10.05.2019

[20] https://medium.com/@sddkal/python-ve-opencv-%C5%9Ferit-tespiti-18bc4d4ad3b3 Erişim tarihi: 05.05.2019

[21] https://github.com/paramaggarwal/CarND-LaneLines-P1/blob/master/P1.ipynb Erişim tarihi: 01.05.2019

[22] https://medium.com/computer-car/my-lane-detection-project-for-the-self-driving-car-nanodegree-by-udacity-36a230553bd3 Erişim tarihi: 01.05.2019

[23] https://mrrol.wordpress.com/2016/12/24/raspberry-pi-3-ile-goruntu-isleme/ Erişim tarihi: 01.04.2019

[24] https://github.com/nikgens/TankRobotProject/blob/master/signRecognition/detectTrafficSign.py Erişim tarihi: 15.04.2019

[25] http://pklab.net/?id=392&lang=EN Erişim tarihi:15.03.2019

[26] https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html Erişim Tarihi:18.05.2019

[27] http://bilgisayarkavramlari.sadievrenseker.com/2007/11/25/duzlestirme-filitresi-gauss-filtresi-gaussian-filter-smoothing-filter-gaussian-blur/ Erişim Tarihi:18.05.2019

[28] https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html Erişim Tarihi: 18.05.2019

[29] https://www.geeksforgeeks.org/arithmetic-operations-on-images-using-opencv-set-2-bitwise-operations-on-binary-images/ Erişim Tarihi: 18.05.2019

[30] https://teknolojiprojeleri.com/programlar/python-nedir-ne-ise-yarar-nerelerde-kullanilir Erişim Tarihi:23.05.2019

[31] http://mesutpiskin.com/blog/opencv-nedir.html Erişim Tarihi: 23.05.2019

EKLER

Raspberry pi OpenCV Kurulum Adımları:

sudo apt-get install unzip
sudo apt-get install cmake
sudo aptitude install libglib2.0-dev
sudo apt-get install libglib2.0-dev
lsblk
sudo umonut /dev/your-dev-name
sudo apt-get update
sudo apt-get upgrade
sudo rpi-update
sudo reboot
sudo apt-get install build-essential cmake cmake-curses-gui pkg-config
sudo apt-get install wget
sudo apt-get install unzip
sudo apt-get install cmake
sudo aptitude install libglib2.0-dev
sudo apt-get install libglib2.0-dev
sudo apt-get install libjpeg-dev
sudo apt-get install libtiff5-dev
sudo apt-get install libjasper-dev
sudo apt-get install libpng12-dev
sudo apt-get install libavcodec-dev
sudo apt-get install libavformat-dev
sudo apt-get install libswscale-dev

```
sudo apt-get install libeigen3-dev
sudo apt-get install libxvidcore-dev
sudo apt-get install libx264-dev
sudo apt-get install libgtk2.0-dev
sudo apt-get -y install libv4l-dev v4l-utils
sudo modprobe bcm2835-v4l2
v4l2-ctl --list-devices
v4l2-ctl --set-fmt-video=width=800,height=600,pixelformat=3
sudo apt-get install libatlas-base-dev gfortran
sudo apt-get install python2.7-dev python2-numpy
sudo apt-get install python3-dev python3-numpy
sudo mount /dev/your-dev-name /home/pi/usbmem
mkdir /home/pi/usbmem
sudo mount /dev/your-dev-name /home/pi/usbmem
mkdir /home/pi/usbmem/opencv
```

ÖZGEÇMİŞ

Osman SARIGÖZ, 01.01.1997'de Balıkesir'de doğdu. İlk, orta ve lise Balıkesir'de tamamladı. 2015 yılında Balıkesir lisesi'nde mezun oldu. 2015 yılında Sakarya Üniversitesi Teknoloji Fakültesi Mekatronik Mühendisliği Bölümü'ne başladı. Mesleki ilgi alanları arasında mühendislik yazılımları, bilgisayar destekli tasarım ve dijital elektronik yer almaktadır.

STANDARTLAR VE KISITLAR FORMU

1. Çalışmanın amacını özetleyiniz.

Otonom araba tasarımı geçekleştirilmesi hedeflenmiştir. Bu araç şerit takibi yapabilen, trafik ışığını algılayan, kırmızı ışıkta duran ve yeşil ışıkta harekete geçen ve tabela tanımlama yapabilen, tabelayı okuyarak sağ, sola, ileri gidebilen bir araba yapılmıştır. Araç kırmızı ışık algılayınca tabela okumaz. Şerit takibi yapmaz. Ancak yeşil renk algıladığında araç hareket etmektedir.

2. Çalışmanın tasarım boyutunu açıklayınız.

Aracı hazır alındı. Katı model oluşturulmadığından elimizdeki denklemlerin birim değerlerini tam değerler elimizde olmadığından dolayı analiz ve simülasyon sonuçları bulunmamaktadır.

Otonom araç şuan bir araba kiti üzerinde yapılan bir çalışmadır. Bu çalışmalar bir RC arabanın içindeki elektronik kontrol ünitesi çıkartılıp raspberry pi kontrolünde kamera ile çevreden bilgi almaktadır. Alınan bu bilgiler görüntü işleme yöntemi ile otonom araç tasarlandı.

3. Bu çalışmada bir mühendislik problemini kendiniz formüle edip, çözdünüz mü?

Arabanın doğrusal yol denklemi için diferansiyel denklemi elde edildi. Direksiyon için diferansiyel denklem elde edildi. Elde edilen denklemler uzay zaman modeline ve transfer fonksiyonuna dönüştürüldü. Yalnızca diferansiyel denklemler elde edilmiştir.

4. Çalışmada kullandığınız yöntemler nelerdir ve önceki derslerde edindiğiniz hangi bilgi ve becerileri kullandınız? Açıklayınız.

Otonom araba için en önemli yazılım bilgisidir. Tüm gördüğüm yazılım dersleri öğrendiğim deneyimler kullandım. Bilgisayarlı görme dersinde görüntü işleme

öğrenmiştik. Bu ödevde görüntü işleme üzerinde oldu. Öğrendiğim diferansiyel denklem ile matematiksel model oluşturdum.

5. Kullandığınız veya dikkate aldığınız mühendislik standartları nelerdir?

İnsanlığın ilerlemesini ve faydalanması için çalışma yapılmaktadır. İzinsiz kaynak kullanımı ve ya çalmanın başka bir kişiden kopyalanması yapılamaz. Çalışma doğaya ve çevreye zararı en azda tutulmalıdır..

6. Kulandığınız veya dikkate aldığınız gerçekçi kısıtlar nelerdir? Lütfen çalışmanıza uygun yanıtlarla doldurunuz.

a) Ekonomi

Aracın üretiminde kullanılan parçalar uygun fiyatlı ve ulaşımı kolay olan ürünlerden seçilmiş.

b) Çevre sorunları:

Çevreye zararlı bir etkisi bulunmamamktadır.

c) Sürdürülebilirlik:

Çalışmada robotu geliştirmesi kolay olsun diye yaygın kullanılan ürünler ve geliştirme kitleri kullanılmıştır. Yapay zeka ilede bu ödev yapılabilir.

d) Üretilebilirlik:

Kullanılan parçalar temin kolaylığı açısından seçilmiştir.

e) Etik:

Aracın insanlık için faydalı bir biçimde kullanılmıştır. Kanunların içinde bir çalışmadır.

f) Sağlık:

İnsanları ağır işlerden kurtararak daha konforlu bir hayat kalitesi sağlamaktadır.

g) Güvenlik:

İstemsiz kazalardan korunmak için önlemler alınmıştır. Güvenlik ilk öncelikdir.

h) Sosyal ve politik sorunlar:

Ülkemizde yerli bir teknoloji olması ülkemizin gelişimi için büyük fayda sağlamaktadır.

Çalışmanın Adı	**Otonom Araç**
Çalışmayı Hazırlayan(lar)	**Osman Sarıgöz**
Danışman Onayı	**Öğr. Gör. Dr. Gökhan ATALI**