



**YILDIZ TEKNİK ÜNİVERSİTESİ
MAKİNE FAKÜLTESİ**

YAPAY ZEKA TABANLI OTONOM SÜRÜŞ

15067019 İsa Karaböcek

15067603 İsmail Alperhan Bay

15067009 Oğuzhan Polat

15067023 Onur Göksun

MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALINDA HAZIRLANAN

BİTİRME ÇALIŞMASI RAPORU

Proje Danışmanı: Doç. Dr. Aydın Yeşildirek

İSTANBUL, 2020

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, yıl boyunca değerli bilgilerini bizlerle paylaşan, verdiği her tavsiyenin hayatımıza kattığı önemini asla unutmayacağımız saygıdeğer danışman hocamız; Doç. Dr. Öğretim Üyesi Aydın Yeşildirek'e, çalışmamız boyunca bizden yardımlarını esirgemeyen asistan hocamız; Mehmet İşcan'a, bizden desteklerini esirgemedikleri için sonsuz teşekkürlerimizi sunuyoruz.

İçindekiler Tablosu

Semboller	4
ÖZET	8
1.GİRİŞ	10
1.1 TANIM.....	10
1.2 AMAÇ.....	11
1.3 MOTİVASYON	11
1.4 KAPSAM	12
1.5 LİTERATÜR TARAMASI.....	13
1.5.1 TANIMLAMALAR.....	13
1.5.2 OTONOM ARAÇ SEVİYELERİ.....	14
1.5.3 ALAKALI ÇALIŞMALAR.....	15
2. GEREKSİNİMLER.....	18
2.1 PAYDAŞ ANALİZİ	19
2.2 TEKNİK GEREKSİNİMLER	19
2.3 TASARIM PARAMETRELERİ	21
3. TEORİK ALTYAPI	22
3.1 GELİŞTİRME ORTAMLARI	22
3.2 LOKALİZASYON VE HARİTALAMA.....	25
3.3 NESNE TESPİTİ	26
3.3.1 YOLOV3 ALGORİTMASI.....	27
3.4 HAREKET PLANLAMA.....	28
3.5 ARAÇ DİNAMİĞİ VE KONTROL	29
4. EKİPMAN SEÇİMİ	39
4. 1 NVIDIA Jetson TX2	39
4.2 RP LIDAR A3	41
4.3 Intel T265 Takip Kamerası	42
4.6 Ublox Neo 6 – M GPS Modülü	45
5. TASARIM VE SİMÜLASYON	46
5.1 Genişletilmiş Kalman Filtresi ile Lokalizasyon ve Sensör Füzyonu	46
5.1.2 Sensör Füzyonu ile Lokalizasyon (GPS + IMU)	48
5.1.3 Genişletilmiş Kalman Filtresi ile Lokalizasyon	50
5.2 SLAM (Simultaneously Localization and Mapping) ile Çevre Haritalama.....	53
5.3 RRT Algoritması ile Yörünge Planlama.....	56
5.3.1 RRT_EXP ROS Uygulaması	57

5.5 OpenCV ve YOLO ile Nesne Tanıma.....	60
5.5.1 OpenCV ile Nesne Tanıma	60
5.5.2 YOLOv3 ile Nesne Tanıma	62
5.5.2.1 Veri Seti Nasıl Eğitilir?	63
5.6 RTAP – MAP (Real-Time Appearance-Based Mapping).....	65
5.7 Pure Pursuit Algoritması ile Lateral Kontrolcü Tasarımı (Path Tracking)	68
6. BÜTÇE VE TAKVİM	73
7. RİSK ANALİZİ	74
8. SONUÇ VE GELECEK İŞLER (FUTURE WORKS).....	75
REFERANSLAR	77

Semboller

D_{rel}	Göreceli Mesafe	(m)
D_{safe}	Güvenli Mesafe	(m)
$D_{default}$	Güvenli Fren Mesafesi	(m)
V_{set}	Sürücünün Belirlediği Sabit Hız	(m/s)
V_{rel}	Göreceli Hız	(m/s)
V_{ego}	Ego Aracın Hızı	(m/s)
T_{gap}	İki Araç Arası Zaman	(s)
m	Kütle	(kg)
\dot{v}	İvme	(m/s ²)
b	Direnç Kuvvetleri	(N)
v	Hız	(m/s)
u	Kuvvet	(N)
p	Konum	(m)
F	$t - 1$ den t ye geçiş matrisi	
u	Gürültü	
Q	Gürültü içeren matris	
y	Gerçek ölçüm ve tahmin arasındaki fark, hata	
S	Tahmini sistem hatası	
H	Geçiş matrisi	
R	Sensör gürültüsü ile ilgili kovaryans matrisi	
K	Kalman kazancı	
δ	Direksiyon açısını gösterir.	
ψ	Yaw (yalpa) açısını gösterir.	
v_x	Aracın boyuna hızını gösterir.(CG'nin orijin olduğu eksen takımına göre)	
v_y	Aracın yanal hızını gösterir.(CG'nin orijin olduğu eksen takımına göre)	
CG	Aracın ağırlık merkezi konumunu gösterir.	
L_f	Ağırlık merkezinin ön aks merkezine olan uzaklığını gösterir.	
L_r	Ağırlık merkezinin arka aks merkezine olan uzaklığını gösterir.	
F_{yf}	Ön aks üzerindeki yanal kuvvet vektörüdür.	

F_{yr}	Arka aks üzerindeki yanal kuvvet vektörüdür.
F_{xf}	Ön aks üzerindeki boyuna kuvvet vektörü(bu model için sıfır kabul edildi, yani aracın boyuna ivmesi sıfır kabul edildi.)
I_z	Aracın yaw(yalpa) durumuna karşı z eksenindeki ataleti gösterir.
α_f	Ön lastiğin kayma açısıdır.
α_r	Örka lastiğin kayma açısıdır.
C_f	Ön lastik için viraj sertlik katsayısıdır.
C_r	Arka lastik için viraj sertlik katsayısıdır.

Kısaltmalar Listesi

LIDAR	Light Detection and Ranging – Işık veya Lazer Dalgası ile Tespit Etme ve Menzil Belirleme
DC	Direct Current – Doğru Akım
PWM	Pulse With Modulation – Darbe Genişlik Modülasyonu
COVID-19	Yeni Koronavirüs Hastalığı
IEEE	Elektrik ve Elektronik Mühendisleri Enstitüsü
SLAM	Eş Zamanlı Konum Belirleme ve Haritalama
SAE	Amerikan Otomotiv Mühendisleri Birliği
TÜİK	Türkiye İstatistik Kurumu
ADAS	Sürücü Asistan Sistemleri
RADAR	Radyo Dalgası ile Tespit Etme ve Menzil Belirleme
GPS	Küresel Konumlama Sistemi
NHTSA	Karayolu Trafiği Güvenliği İdaresi (ABD Federal Hükümeti Ajansı)
IMU	Ataletsel Ölçüm Birimi
ESC	Hız Kontrol/Kontrolcüsü
HIL	Gerçek zamanlı gömülü sistemlerin geliştirilmesinde ve test edilmesinde kullanılan bir tekniktir.
2B/2D	İki Boyutlu
3B/3D	Üç Boyutlu
FSM	Durum Makinesi
FWD	Önden Çekişli Araç
RWD	Arkadan Çekişli Araç
AWD/4WD	Dört Çekişli Araç
CG	Ağırlık Merkezi
GPU	Grafik İşlemci Birimi
I2C	Seri ve senkron bir haberleşme türüdür.
EKF	Kalman Filtresi
RRT	Rapidly Exploring Random Trees Algoritma
ID	Kimlik Numarası
CPU	İşlem Birimi
RAM	Erişimli Hafıza
RTAP-MAP	Gerçek Zamanlı Görünüm Tabanlı Haritalama
RGB	Kırmızı Yeşil Mavi renklerin farklı oranda bileşimleriyle diğer renklerin elde edilme tekniğidir.
BOW	Kelime Çantası Modeli, bilgisayarlı görme ve doğal dil işleme konularında kullanılan bir modeldir.

Şekiller Tablosu

Şekil 1 Otonom araç benzetimi	10
Şekil 2 2018 TÜİK İstatistik verileri.....	12
Şekil 3 Otonom Araç Görevleri	13
Şekil 4 Otomotiv Mühendisleri Topluluğu otonom araç sınıflandırması.....	14
Şekil 5 Otonom araçlar için otomasyon seviyeleri (SAE sınıflandırmasına göre)	15
Şekil 6 Stanford Yarış Ekibinin Darpa Grand Challenge (2005) yarışmasını kazanan araç.....	16
Şekil 7 MIT RACECAR Platformu ve Sensör Konfigürasyonu.....	17
Şekil 8 F1/10 yarışmasının tanıtım afisi.....	17
Şekil 9 ROS bilgi akış diyagramı	24
Şekil 10 SLAM Örnekleri	26
Şekil 11 Yapay Sinir Ağı Benzetimi	26
Şekil 12 YOLOv3 kıyaslama sonuçları.....	27
Şekil 13 Sonlu Durum Makinesi	29
Şekil 14 Önden Çekişli Otomobil için güç aktarma organlarının yerlesimi	30
Şekil 15 Arkadan İtişli Otomobil için güç aktarma organlarının yerlesimi	31
Şekil 16 Dört Çekişli Otomobil için diferansiyel yerleşimlerinin iki farklı konfigürasyonu.....	31
Şekil 17 İçten yanmalı motora sahip bir otomobilin güç aktarma modeli şeması	32
Şekil 18 Önden çekişli bir aracın güç aktarma organları patlatılmış montaj gösterimi	32
Şekil 19 Elektrik motoruyla tahrik edilen elektrikli otomobil için güç aktarma modeli şeması...33	33
Şekil 20 Hibrit motorlu bir otomobilin güç aktarma modeli.....	33
Şekil 21 Basitleştirilmiş Ackermann Geometrisi	34
Şekil 22 Ackermann Geometrisi için önemli açı ve uzunlukların şematize edilmesi	34
Şekil 23 Bisiklet Modeli Genel Gösterimi	35
Şekil 24 Geometrik Bisiklet Modeli.....	35
Şekil 25 Kinematik Bisiklet Modeli.....	36
Şekil 26 Dinamik Bisiklet Modeli Gösterimi.....	37
Şekil 27 Diferansiyel Tahrik Modeli Şematik Gösterimi.....	39
Şekil 28 Nvidia Jetson TX2	39
Şekil 29 Jetson TX2 Serisi Modül Blok Şeması [8].....	40
Şekil 30 Lidar Çalışma Prensibi	41
Şekil 31 RPLidar A3	41
Şekil 32 Kamera görüşü ve IMU sensörlerinin temsil edilmesi.....	42
Şekil 33 Intel Realsense programı ara yüzü ve alınan verilerin görselleştirilmesi	42
Şekil 34 PCA9685 [sol - sağ üst] Ardnuio bağlantısı [sol alt] kontrolü [sağ alt].....	43
Şekil 35 PCA9685 Modülü Giriş/Çıkış Pinleri ve Özellikleri	44
Şekil 36 LSM9DS1 IMU.....	45
Şekil 37 GPS Modülü ve Anten Birimi.....	45
Şekil 38 Sensör Füzyonu I/O	46
Şekil 39 Otonom Araç Blok Diyagramı	47
Şekil 40 Örnek Senaryo.....	47
Şekil 41 Kalman Filtresi.....	49
Şekil 42 EKF Simülasyonu Başlangıcı	52
Şekil 43 Matematiksel Modelden Gelen Hata (siyah çizgi).....	52
Şekil 44 EKF ile Sensör Füzyonu ve Matematiksel Modelin Karşılaştırılması.....	53
Şekil 45 SLAM iş akış diyagramı	53
Şekil 46 Temel SLAM Problem Tanımı	54
Şekil 47 Gazebo Ortamı ve Aracın İlk Konumu	55

Şekil 48 RVIZ Ortamında Aracın İlk Konumu	55
Şekil 49 Aracın Haritalamaya Devam Etmesi.....	55
Şekil 50 RVIZ ortamında aracın haritalamayı bitirmesi	55
Şekil 51 Python ortamında RRT Algoritması Simülasyonu	56
Şekil 52 RRT algoritması düğüm akış diyagramı [11].....	57
Şekil 53 Mekatronik mühendisliği bölüm binası benzetimi.....	58
Şekil 54 Başlangıç noktasından sınır ve bitiş noktalarının belirlenmesi.....	59
Şekil 55 Bitiş noktası için en kısa yolun bulunması ve haritanın anlık olarak güncellenmesi.....	59
Şekil 56 Bitiş noktasına varılması	59
Şekil 57 Gazebo ortamında aracın görüş alanına nesnelerin eklenmesi	61
Şekil 58 Grafik arayüzü aracılığıyla nesnelerin etiketlenmesi.....	61
Şekil 59: Nesne Tanımlama – Karar Verme Ros akış diyagramı	62
Şekil 60 Verilerin etiketlenmesi	63
Şekil 61 YoloV3 eğitim çıktısı	64
Şekil 62 RTAPMAP ROS akış diyagram	65
Şekil 63 Başlangıç konumundaki aracın kamera, lidar ve odometriden aldığı verilerin görüntülenmesi	66
Şekil 64 Teleop ile aracın sürülmESİ	66
Şekil 65 Lokalizasyon, haritalama ve sensör verilerinin görüntülenmesi.....	66
Şekil 66 Stereo kameralı 3B haritalama ve SLAM	67
Şekil 67 Lokalizasyon [sağ kısım] ve araç kamerasının görüş alanı [sol altta]	67
Şekil 68 Odometri ve kamera verisinin füzyonlanması	67
Şekil 69 Stanley Metodunun Geometrik Açıklaması	68
Şekil 70 Pure Pursuit yönteminin çalışma prensibi.....	69
Şekil 71 Pure Pursuit Metodunun Geometrik Gösterimi	69
Şekil 72 İleriye Bakma Mesafesinin Sürüş Performansına Etkisi.....	72
Şekil 73 Pure Pursuit Kontrolcü ile Yörünge Takibi	73
Şekil 74 Risk Analizi.....	74
Şekil 75 Gelecek işler.....	76

Tablolar

Tablo 1 Teknik Gereksinimler	20
Tablo 2 Ekipman Özellikleri	20
Tablo 3 Bütçe	74
Tablo 4 Takvim	74

Revizyon Tarihçesi

TARİH	Revize No	Tanım	Yazarlar
14/06/2020	1.0	Başlangıç dökümanı	İsa Karaböcek Oğuzhan Polat Onur Göksun İsmail Alperhan Bay
21/06/2020	1.1	Düzenlemeler yapıldı	İsa Karaböcek Oğuzhan Polat Onur Göksun İsmail Alperhan Bay

ÖZET

Yol istatistikleri, tüm araba kazalarının %95'inin insan hatasıyla ilişkili olduğunu ve Avrupa Birliği sınırları içerisinde 2011 yılında 30.000 kişinin hayatının kaybettiğini raporlamıştır. Otonom sürüsü, insan hatasının ortadan kaldırarak binlerce hayatı kurtarma potansiyeline sahiptir. Otonom sürüsü sağlayabilmek için bir aracın yeterli sayıda sensöre ihtiyacı vardır. Bunlar, hareketli nesneleri takip etmek, anlık konum bilgisi sağlamak, haritalandırma yapmak gibi görevler için gerekli olan sensörlerdir. Sensörler sayesinde araç çevresini algılar ve güvenli bir sürüsü sağlamak için bu veriler anlamlandırılarak işleme tabi tutulur. Buradaki önemli etkenlerden birisi sensör verileridir.

Bu projede, uzun zamandır konuşulan otonom kara araçları üzerine literatür taraması yapılmış, otonom bir kara aracının gerçekleştireceği görev algoritmalarının uzaktan kumandalı bir araç üzerinde test edilmesi amaçlanmıştır. Nesne algılama, lokalizasyon ve haritalama, hareket planlama görevleri bu projenin kapsamındadır.

Aracın kinematik denklemleri üzerinde durularak Saf Takip ve Stanley kontrolcülerini karşılaştırılmış ve model araç için daha uygun olan Saf Takip kontrol yöntemi benimsenmiştir. Araştırma kapsamında LIDAR ve Kamera gibi sensörlerden veri okunması ve bu verilerin işlenmesi incelenmiştir. Yapay zekâ ile nesne tanıma algoritması test edilmiş ve %95 doğruluk oranında sonuç alınmıştır. Yapay zekâ ile nesnelerin sınıflandırılması ve tespit edilmesi için algoritma geliştirilmiştir. Aracın kontrolü için üzerinde bulunan DC tahrik motoru ve Servo dümen motoru darbe genişlik modülü (PWM) yöntemi ile kontrol edilerek istenilen boylamsal hızlar ve dönüş açıları elde edilmiştir.

ROS (Robotic Operating System) araçları(toolları) kullanılarak model aracımız, sensör füzyonu olacak şekilde gerçek ortamda test edilmiştir. Yapılan tüm çalışmalar ROS sayesinde birleştirilerek simüle edilmiş, teknik şartnameye hedeflenen çıktılar alınmıştır.

Aracın elektronik parçaları temin edilmiş ve araç üzerine montaj süreci COVID-19 sebebiyle kesintiye uğramıştır ve devam edecktir. Bu aşamadan sonra test sürüşleri trafiğe kapalı ortamda gerçekleştirilecek gerçek zamanlı performans ölçümleri yapılacaktır. IEEE ve TÜBİTAK 2242 Lisans Bitirme Çalışması yarışmalarında projemiz sergilenecektir.

1.GİRİŞ

Gelişen teknoloji ile birlikte otomotiv sektörü hızla büyümektedir. Otomotiv şirketleri daha güvenli ve daha hızlı ulaşım araçları üretmek için rekabet etmekte ve büyük yatırımlar yapmaktadır. Yapılan araştırmalar göstermektedir ki geleceğin ulaşım araçlarında otonom sistemler güvenliği %90 oranında artırarak can ve mal kaybını azaltacaktır.



Şekil 1 Otonom araç benzetimi

1.1 TANIM

Yıllar geçtikçe insanlık yeni sorunlarla karşılaşmakta ve bu sorunları çözmek için teknolojinin gücünü kullanmaktadır. Ulusal Karayolu Trafik Güvenliği İdaresi'nin 2016 yılında yayınladığı rapora göre dünyada meydana gelen trafik kazalarının %94'ü insan hataları sebebi ile gerçekleşmektedir. Bu sorunu çözmek için otomotiv şirketleri devletlerden aldığı destek ile rekabet içine girmiştir. Tesla, Ford, Audi gibi lider şirketler son yıllarda yarı otonom denilebilecek araçlar üretmeye başlamıştır. Bu araçlarda adaptif hız sabitleyici, park asistanı, gece görüşü gibi gelişmiş fonksiyonlar bulunmaktadır. Yapılan milyar dolarlık yatırımlar henüz şirketlere istenen geri dönüşü (Return of Investment) sağlamamıştır.

Bu projede lidar ve takip kamerası kullanarak 1:10 boyutlarındaki bir model aracın planlanan yörüngeyi yol alması, nesnelerden kaçınması ve tabelaları tespit etmesi gerçekleştirilecektir. Bu bağlamda nesne tespiti için YOLO algoritması ve haritalama, lokalizasyon için SLAM algoritması kullanılacaktır.

1.2 AMAÇ

Yapılan çalışmalar sayesinde sensör teknolojisi hızla gelişmekte ve sensörler daha hassas ölçümler yapabilmektedir. Bu proje kapsamında model aracın sahip olduğu iç ve dış sensörlerden alınan veriler ile statik ve dinamik nesneleri tespit ederek çarpışmalardan kaçınması amaçlanmaktadır. Bu amaç doğrultusunda aracın dış sensörleri lidar ve takip kamerası olarak belirlenmiştir. Sensör sayısı ve çeşidi arttıkça işlemciye binen yük artsada gelişmiş çok çekirdekli yapay zekâ bilgisayarları bu yükün altından kalkabilmektedirler. Sensör çeşidinin getirdiği avantajlardan biri de gerçekleştirilecek olan sensör füzyonu bu sayede daha yüksek doğrulukta çıktılar üretebilecektir. Bu bağlamda lidarın üreteceği 3 boyutlu nokta bulutları bir haritalama verisi üretirken 25 metreye kadar çevresinde bulunan statik ve dinamik nesneleri tespit edebilecektir.

Otomotiv Mühendisleri Derneği (SAE) tam otonom araçların üretilmesi içinümüzde 20-30 yıllık bir zamanın olduğunu öngörmektedir. Bu bağlamda, bu çalışmanın sonucunda her koşulda ve her ortamda olmasa da belirli koşullar altında ve belirli ortamlarda otonom sürüs gerçekleştirebilecek bir algoritma geliştirilmesi amaçlanmaktadır. Mekatronik Sistem Tasarımı dersi kapsamında model aracın dinamik denklemleri hesaplanmıştır, yanal ve boylamsal motor kontrolcüsü tasarlanmıştır. Bu hesaplar sonunda aracın simülasyon ortamında çıktıları incelenmiştir. Ayrıca kameradan alınan verilerle YOLO algoritmasında eğitilen veri setleri ile tabela tespiti gerçekleştirilmiştir. Yapılan işlemler NVIDIA tarafından yapay zekâ çalışmaları için özel olarak üretilen çok çekirdekli Jetson TX2 işlemci kartı üzerinde denenmesi planlanmaktadır. Ancak global kriz sebebiyle çalışmalar simülasyon ortamıyla kısıtlandırılmıştır.

1.3 MOTİVASYON

Son on yılda, Gelişmiş Sürücü Yardım Sistemi ve otonom sürüs konusunda muazzam bir ilerleme kaydedilmiştir. Bu sistemleri çarpışma önleyici, şerit takip asistanı, trafik levha tespiti olarak sıralayabilir. Bu sistemlerin geliştirilmesinde ki en büyük motivasyon kaynağı ulaşımı daha güvenli ve ekonomik hale getirmektir. Türkiye İstatistik Kurumu'nun (TÜİK) 2018 yılında yaptığı araştırmaya göre Türkiye'de ölümlü ve yaralanmalı trafik kazalarının %96'sının yaya ve sürücü kaynaklı hatalar yüzünden meydana geldiği bildirilmiştir. Yol, araç ve hava şartları yüzünden meydana gelen kazaların oranı %3'tür. Bu bağlamda TÜİK verilerine göre trafik kazalarındaki mal ve can kayıpları çok büyük oranda insan hataları sebebiyle gerçekleşmektedir.

**TRAFİK KAZALARINA (Ölümlü-Yaralanmalı)
NEDEN OLAN UNSURLAR**

KUSUR UNSURLARI	2018 YILI
SÜRÜCÜ	194.928
YAYA	18.394
YOL	1.300
ARAÇ	1.360
YOLCU	1.916
TOPLAM	217.898

Şekil 2 2018 TÜİK İstatistik verileri

Bu projede ana motivasyon kaynağı olarak trafik kazalarındaki insan hatalarını azaltmak ve buna bağlı can ve mal kaybını minimuma indirmek gösterilmektedir. Gelişmiş Sürücü Yardım Sistemine (ADAS) sahip araçların en büyük avantajı sahip oldukları yazılım ve donanım ile gerçekleşmesi mümkün kazaları öngörmeleridir. Radar, Lidar, kamera gibi sensörler sürücülerin aracı kontrol etmekte zorlandığı sisli hava, karanlık hava yoğun trafik gibi senaryolarda oluşabilecek kaza durumlarını ön görmede yardımcı olmaktadır. Özellikle Lidar ve Radar sensörleri görüş alanına ihtiyaç duymadan doğru veriler üretebilmektedir. Bahsedilen gelişmiş görüş sistemleri, sensörler ve çok çekirdekli güçlü işlemciler bu projenin geliştirilmesinde dolaylı olarak bizlere motivasyon kaynağı olmuştur.

1.4 KAPSAM

Otonom araçlar sürüs için gerekli verileri çevreyi algılayarak ve yorumlayarak elde etmektedir. Çevre algılaması için gelişmiş sensör teknolojisinden faydalaniılmaktadır. Başlıca otonom araçlarda kullanılan sensörleri lazer, Radar (Radio Detection and Ranging), LIDAR (Light Detection and Ranging), GPS(Global Position System), Odometri ve istenilen görevde uygun kamera seçimleri ile sınıflandırabiliriz. Radarlar radyo dalgaları kullanılarak objelerin mesafe, yön ve hızlarını belirlemek için kullanılır. Uzak nesneleri algılamak için daha kullanışlıdır. Lidarlar ışık ışınları yardımıyla önünde bulunan nesneleri algılayabilir. Sensörlerden alınan verilerin yorumlanması için gelişmiş kontrol sistemleri kullanılır ve yorumlanan veriler ile çevre algılanması, yörunge planlanması ve tespit edilmek istenen dinamik/statik nesneler algılanır. Aynı zamanda sensör verileri füzyonlanarak, doğruluk oranı yüksek veriler elde etmek içinde kullanılır. Haritalama ve lokalizasyon çalışmalarında kamera +Lidar, Lidar + Odometri, kamera + Odometri gibi sensör verileri eşlenerek. Aracın hem konum bilgisini hemde görüş alanındaki nesnelerin bilgisi elde edilebilir. Bu sayede otonom sistemler için daha güvenli bir sürüs deneyimi sağlar.

Projede hedeflenen otonom araç kapsamında ise 4 temel görev hedeflenmiştir. Bunlar

- Sensör verileri füzyonlanarak eş zamanlı çevre algılanması ve olası engellerin tespit edilmesi,
- Trafik levhalarını tespit ederek güvenli bir sürüş sağlanması,
- Çevre haritalanması ve lokalizasyon yapılması,
- Çevreyi algılayarak bir yörünge planlanması yapabilmesi görevleri belirlenmiştir.



Şekil 3 Otonom Araç Görevleri

Bu proje kapsamında 1:10 ölçekli bir model araç üzerinde lidar ve kamera kullanarak statik ve dinamik nesnelerin tespiti, haritalama, lokalizasyon ve çarpışmalardan kaçınabilecek bir yörünge planlama sistemi gerçekleştirilecektir.

1.5 LİTERATÜR TARAMASI

Bu kısımda sürücüsüz araç hakkında bilinmesi gereken bazı terimleri açıklanacak. Daha sonra sektördeki otonom araç seviyelerinden bahsedilecektir. Son kısımda ise bu çalışmaya referans olabilecek nitelikte yapılmış akademik çalışmalar ve projelere yer verilecektir.

1.5.1 TANIMLAMALAR

Otonomi: Bir sistemin kendi kendine yüksek doğruluk oranı ile çalışması otonomi olarak bilinir. Bu konsept çeşitli otomobillerde, uçaklarda ve yük taşıma araçlarında uygulanmaya başlamıştır.

Otonom Araç: Gelişmiş bir otonom araç herhangi bir sürücü müdahalesi olmaksızın herhangi bir noktadan başka bir noktaya kendi kendine gidebilen araçtır. Otonom araç üzerinde bulundurduğu iç ve dış sensörlerle çevredeki nesneleri algılar, şertileri tespit eder ve 3 boyutlu haritalandırma yaparak doğru referans yörüngeyi üreterek bu yörüngeyi takip eder. Bu sayede otonom araçlar sürücünün hata yapabileceği sürüş senaryolarında hata oranını minimuma indirerek kaza oranlarını azaltabilir.

1.5.2 OTONOM ARAÇ SEVİYELERİ

Sektörde otonom araç seviyeleri 2 ana sınıflandırmaya ayrılmıştır. Bunlardan ilki US National Highway Traffic Safety Administration (NHTSA) diğer ise Otomotiv Mühendisleri Topluluğu (SAE). Bu ikisi arasındaki ana farklılık ise SAE 6 seviye sınıflandırma kullanırken NHTSA 5 seviyelik sınıflandırma kullanmaktadır. NHTSA SAE'nin yayınladığı 6 seviyelik sınıflandırmayı kabul etmiştir. Bu sebeple çalışmamızda SAE'nin 6 seviyelik otonom araç sınıflandırması benimsenmiştir. Otomotiv Mühendisleri Topluluğu (SAE)'na göre otonom araçları 5 seviyede incelemektedir:

Seviye 0: Bu seviyede sürücü tüm kontrolü ele almalıdır ve çevreyi sürekli olarak gözlemlidir. Araç hiçbir otonom yazılıma ve donanıma sahip değildir.

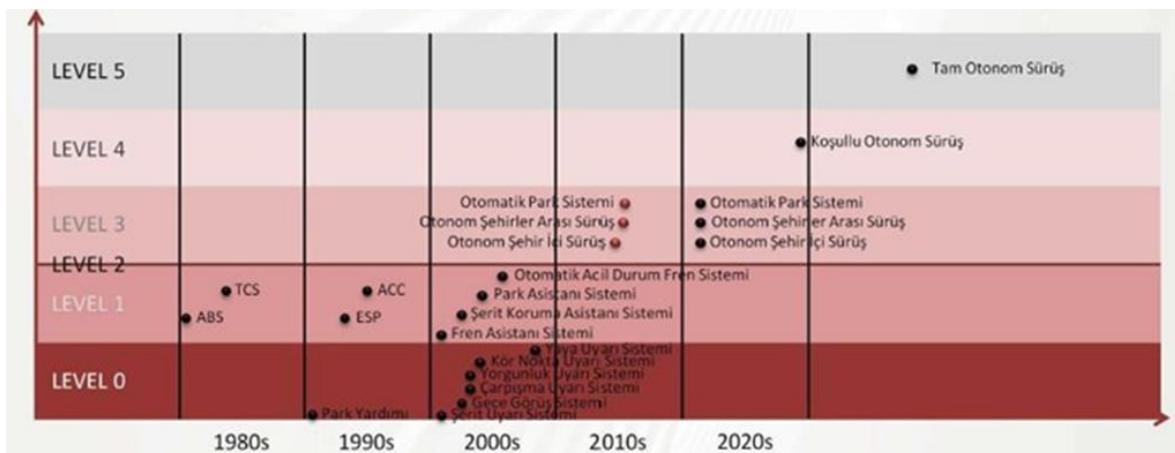
Seviye 1: Araç, temel sürüs asistanına ve sensörlere sahiptir. Sürücü sürekli olarak çevreyi gözlemeli ve kontrolü sürekli elinde tutmalıdır. Örnek vermek gerekirse gece görüş asistanı, şerit ya da yay uyarı sistemi bunlardan bazalarıdır.

Seviye 2: Araç, ivmelenme yönlendirme gibi fonksiyonlara sahiptir. Sürücü her zaman tetikte olmalıdır ve çevreyi gözlemlemelidir. Park asistanı, şerit koruma asistanı ya da adaptif hız sabitleyici bu fonksiyonlardan bazalarıdır.

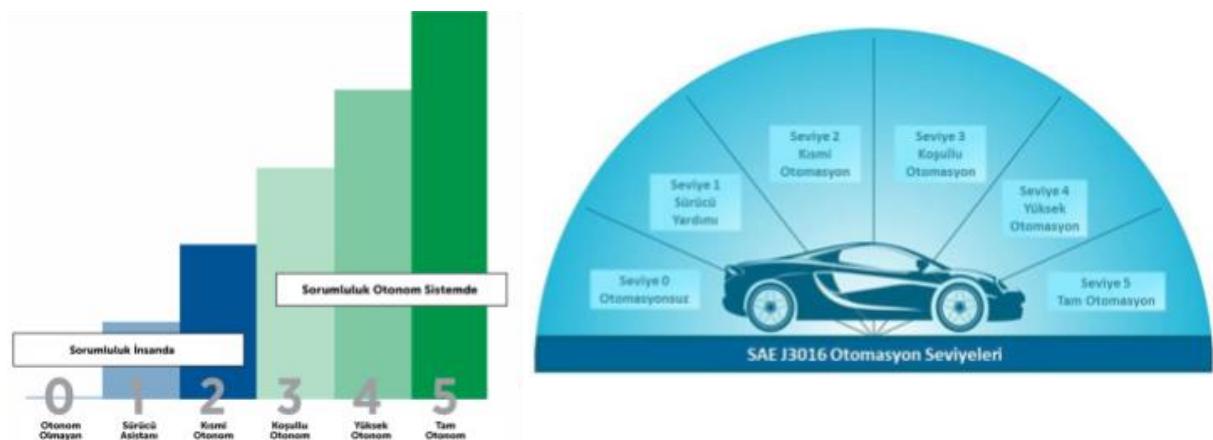
Seviye 3: Araç, otonom sistemleri barındırır. Sürücü çevreyi gözlemlemek zorunda değildir ancak tetikte olmalıdır.

Seviye 4: Araç, sürücüsüz olarak belirli koşullar altında bütün sürüs fonksiyonları yerine getirebilecek donanım ve yazılıma sahiptir. Sürücü sadece opsiyondur.

Seviye 5: Araç, tüm koşullar altında bütün sürüs fonksiyonlarını yerine getirebilecek teknolojiye sahiptir. Sürücüye ihtiyaç yoktur. Bu seviye tam otonom sürüs olarak bilinmektedir.



Şekil 4 Otomotiv Mühendisleri Topluluğu otonom araç sınıflandırması



Şekil 5 Otonom araçlar için otomasyon seviyeleri (SAE siniflendirmesine göre)

1.5.3 ALAKALI ÇALIŞMALAR

Stanley

Stanford Üniversitesi'nin *Stanford Yarış Ekibi* tarafından yaratılan Stanley, 2005 DARPA Grand Challenge yarışmasını kazanan otomobildir. DARPA Grand Challenge, 240 km süren Mojave çölünde özerk(otonom) bir araba yarışmasıydı.

Araç üstü otonom sistem, 3B haritalama için beş Sick AG LİDAR ünitesi bulunmaktadır. Pozisyon tahmini için IMU'lارla birlikte küresel harita lokalizasyonu için bir GPS sistemi, kilometre sayacı için tekerlek sensörleri ve yol koşullarını tespit etmek için bir video kamera kullanılmıştır.

Bu projeyi ilgi çekici yapan asıl özellik ise yörüngede takibinde kullanılmak üzere “Stanley Direksiyon (Steering) Kontrolörü” isimli yeni bir kontrol yaklaşımının önerilmesidir. Kinematik bisiklet modeline dayanan bu kontrolcü hakkında teorik bilgi Tasarım bölümünde verilecektir.

Stanford Yarış Takımı, daha sonra otomobiller için otonom sürüş yazılımı üreten Google'a ait Waymo şirketi ve çevrimiçi öğrenme sitesi Udacity'yi kuran Doçent Sebastian Thrun tarafından yönetildi. [1]

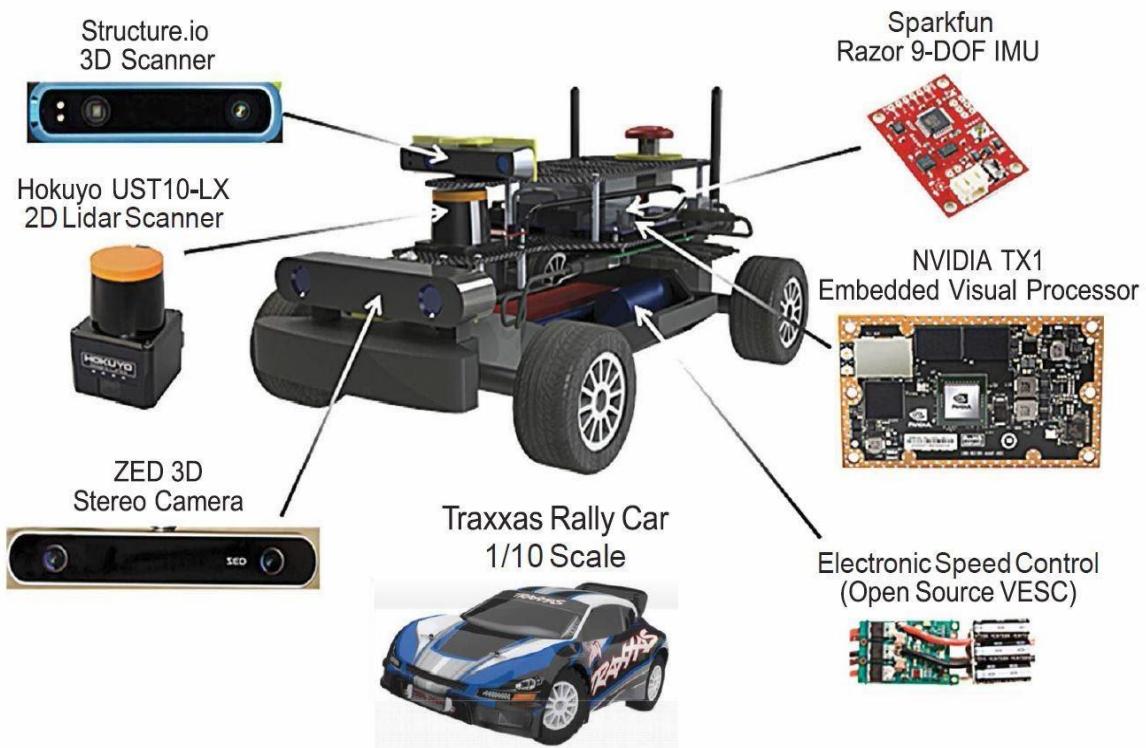


Şekil 6 Stanford Yarış Ekibinin Darpa Grand Challenge (2005) yarışmasını kazanan araç

MIT RACECAR

MIT RACECAR (Rapid Autonomous Complex-Environment Competing Ackermann-Steering Robot) Massachusetts Teknoloji Enstitüsü'nden robotik araştırma ve eğitim için oluşturulan açık kaynaklı bir donanım platformudur. Platform, öğrencilere karmaşık ortamlarda hızlı bir şekilde dolaşabilen otomobiller için perception(algilama) ve planning(planlama) ve kontrol algoritmaları hakkında bilgi vermek için ilgili lisans dersinde kullanılır. Öğrencilerin okul içerisinde platform için tasarladığı algoritmanın başarısını test edebilmesi için ayrıca bir yarışma pisti de vardır. Platformda kullanılan yarış aracına, sensörlerle ve geliştirme ortamına sırasıyla değinelim:

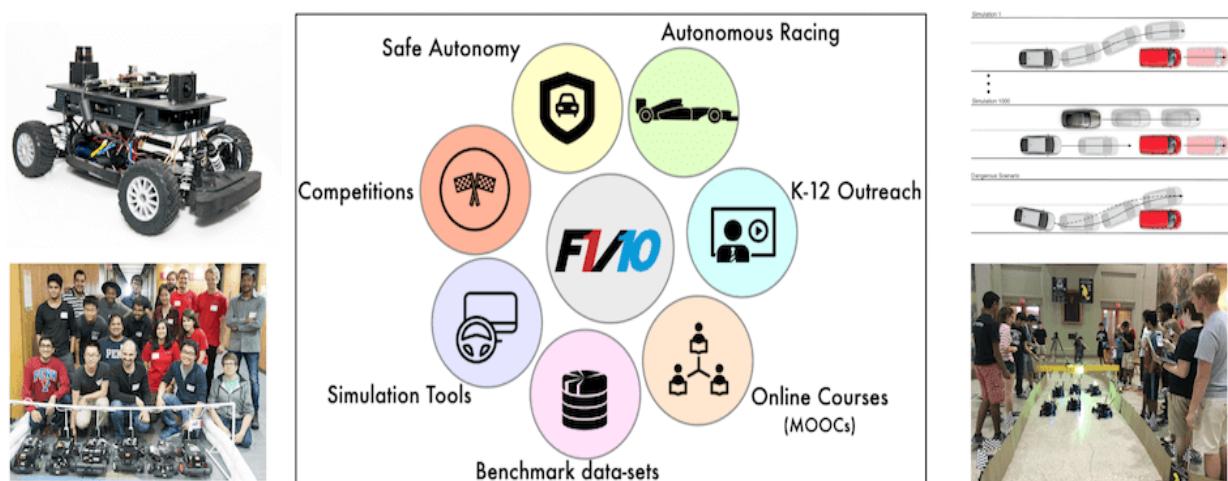
- RACECAR platformu, 1/10 ölçekli “Traxxas Rally” aracını temel alır.
- Merkezi işlem birimi olarak NVIDIA Jetson TX1 kullanılmaktadır.
- Pozisyon tahmini için Sparkfun IMU ve görsel kilometre sayacı kullanılmaktadır.
- Araç, bir lazer tarayıcı (laser scanner) olan Hokuyo UST-10LX ve bir stereo kamera (Stereolabs ZED) ile donatılmıştır.
- Platformu sanal ortamda test edebilmek için ve geliştirilen farklı algoritmaların ortak çalışabildiği ve sensör füzyonunun da yapılabildiği ROS yazılımı kullanılıyor. [2]



Şekil 7 MIT RACECAR Platformu ve Sensör Konfigürasyonu

F1/10

F1/10, öğrenciler için uluslararası otonom araç yarışmasıdır. Yarışma, 60 km /saat üzerinde hız yapabilen özerk 1/10 ölçekli F1 yarış otomobilini tasarlama, inşa etmeyi ve test etmeyi içeriyor. Öğrencilerin yarışma şartnamesinde verilen görevleri yaparken perception(algilama), planning (hareket planlama) ve kontrol yaklaşımlarını öğrenmesi amaçlanır. Yarışma pisti MIT RACECAR projesine benzemektedir. [3]



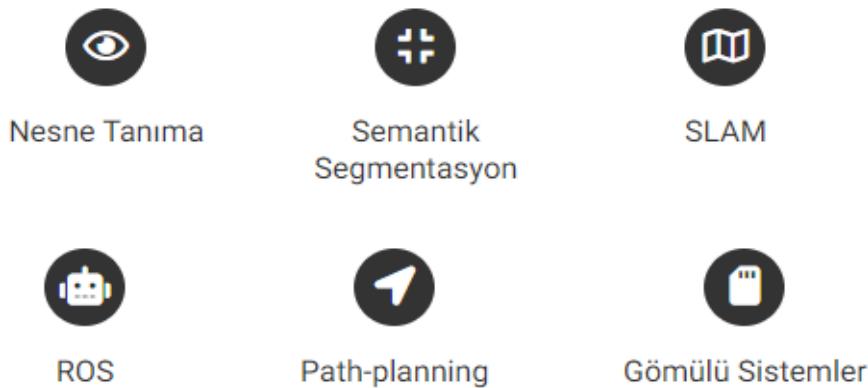
Şekil 8 F1/10 yarışmasının tanıtım afişi

2. GEREKSİNİMLER

Proje raporunun giriş kısmında da belirtilen OpenZeka MARC yarışması gereksinimlerin belirlenmesindeki en etkili paydaşlardan biri olmuştur. Kurallar kitapçığına göre verilen görevlerin yerine getirilebilmesi için **figürdeki** ana görevler belirlenmiştir.



Proje teslimi için kararlaştırılan 8 aylık süre ve proje ekibinin kabiliyeti göz önünde bulundurularak **figür 1** de verilen 8 ana görevden **figür 2**'deki 6 ana görev seçilmiştir. Seçilen bu görevler detaylandırılarak alt başlıklar oluşturulmuştur. Bunun sonucunda bir gereksinimler ağacı ortaya çıkarılarak projenin gereksinimleri net olarak ifade edilmiştir.



Otonom bir aracı düşündüğümüzde, hareket etmeden önce yapması gereken ilk görev çevresindeki nesneleri algılamak ve sürülebilir alanları belirlemek olacaktır. Bu nedenle *Nesne Tanıma* görevinin altında *Nesne Algılama* ve bu nesnelerden faydalı bilgiler (konum, hız, çarpışma zamanı) çıkarma gibi görevleri de yerine getirmesi gerekmektedir. Nesneler hakkında bilgi edindikten sonra aracın hareket edebileceği serbest bölgeleri belirlemek için kamera üzerinden *Semantik Segmentasyon* görevini yerine getirmesi gerekmektedir.

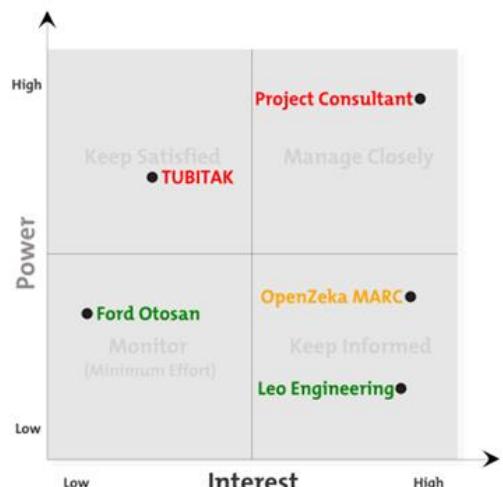
Araç harekete başlamadan önce ve hareketi süresince diğer nesneler ile çarpışmayı önlemek için sürekli olarak kendi konumunu güncellemesi gerekmektedir. Bu güncelleme görevini yerine getirmek adına SLAM (Simultaneous Localization and Mapping) görevi seçilmiştir. Araç yukarıdaki görevleri sırası ile gerçekleştirdikten sonra nereye gideceğine karar vermelidir. Bu projede lokal *Hareket Planlamaya* odaklanılmıştır. Harita seviyesinde yol planlama kapsam dışıdır. Hareket planlama görevini en sade hali ile iki nokta arasında gidilecek yolu planlaması olarak tanımlayabiliriz.

Araç planlanan yörüngede hareket etmek için boylamsal ve yanal eksenlerde hareket edecektir. Bu iki boyutlu hareket için iki farklı kontrolcüye ihtiyacı vardır. Bu kontrolcülerin tasarlanması ve araca entegrasyonu *Gömülü Sistemler* görevi altında inceleneciktir.

2.1 PAYDAŞ ANALİZİ

Proje sürecinde paydaşların tespiti ve önem sırası aşağıdaki gibi belirlenmiştir.

- Proje Danışmanı
- OpenZeka MARC
- TÜBİTAK
- Leo Mühendislik
- Ford Otosan



2.2 TEKNİK GEREKSİNİMLER

Sistem tasarımını belirginleştirmek, basitleştirmek ve yarışma konseptine uygun duruma getirmek için projenin teknik gereksinimleri belirlendi. Yapılacak otonom sürüş görevleri, tanınacak trafik işaret ve levhaları, test yapılacak parkurun ölçülerini belirlerken “OPENZEKA-MARC MİNİ OTONOM ARAÇ YARIŞMASI KURAL KİTAPÇIĞI” referans alındı. Diğer gereksinimleri ve kısıtlamaları belirlerken sahip olunan donanımların çalışma aralığı ve kapasitesi dikkate alındı. Teknik Şartnamede belirtilen kriterler aşağıdaki gibidir:

Teknik Gereksinimler	Nitel veya Nicel Karşılıkları
Çalışılacak eksen takımları	(x,y) ve (x',y') global ve araç koordinat eksenleri
Ek donanım montajlarıyla birlikte aracın toplam kütlesi	m=2.2 kg
Aracın düz yoldaki maksimum hız büyülüğu	v_max=10 m/s
Aracın kararlı viraj alma esnasındaki maksimum hız büyülüğu	v_(c,max)=5 m/s
Aracın hızlanma ivmesi büyülüğu	a_acc=2 m/s^2
Aracın yavaşlama ivmesi büyülüğu	a_dec=3 m/s^2
Minimum güvenli takip mesafesi	3 metre
Maksimum kabul edilebilir yörünge (çapraz yol) hatası	150 milimetre
Tanınacak Trafik İşaret ve Levhaları	DUR – Azami Hız – Asgari Hız

Tablo 1 Teknik Gereksinimler

Ekipman Özellikleri	Nitel veya Nicel Karşılıkları
Stereo Kamera'nın maksimum nesne algılama menzili	10 metre
LIDAR'ın maksimum nesne algılama menzili	Açık renkli objeler için 25 metre Koyu renkli objeler için 10 metre
DC Motor hız performansı	Özel üretim Titan 12T- 550 Motor ile 1,85 kg ağırlığındaki model araç (18m/s) maksimum hızına çıkabiliyor.
Servo Motor açı değişim aralığı	6 Volt gerilimle beslenince 60° açıyı 0,23 saniyede kat ediyor. (Açı aralığı = 60°)
DC Motor hız kontrol yöntemi	PWM Sinyali ile
Servo Motor açı kontrol yöntemi	PWM Sinyali ile
ESC'nin Motorlar ve Kontrolcü Kart ile haberleşme yöntemi	I2C PWM Modülü
Batarya'nın bu tasarım için çalışma ömrü	*8.4 Volt-7 Hücre 3000 mAh Batarya ve XL-5 ESC için *Sürekli sabit 1.4 A BEC Akımı çekildiğinde (yani başlatma, ivmelenme, frenleme durumları ihmal edildiğinde) Maksimum Batarya Ömrü = 45 dakika

Tablo 2 Ekipman Özellikleri

2.3 TASARIM PARAMETRELERİ

Otonom araçlar için yapay zekâ tabanlı bir otonom sürüş sistemi tasarlanırken aşağıdaki parametreler göz önünde bulundurulmuştur:

Operasyonel gereksinimler: Aracın hızı, düz yolda 10 m/s, virajlarda 5 m/s olacak şekilde tasarlanmıştır. Öndeki araç ile kontrol edilen ego aracın arasındaki güvenli takip mesafesi minimum 3 metre olacak şekilde belirlenmiştir. Aracın, virajlarda takip edeceği yörüngeden maksimum 0,15 metre hata ile yoluna devam etmesi ve düz yolda bu hatanın 0,01 metreye kadar düşmesi beklenmektedir.

Güç gereksinimleri: Aracın minimum 6 dakika boyunca 2,5 kg'a kadar faydalı yük taşıması beklenmektedir. Araç için tahrik elemanları ve bataryaların özellikleri bu parametrelere göre seçilecektir.

Çalışma Koşulları: Araç kapalı bir mekânda çalışacak olup, olumsuz hava koşulları göz ardı edilecektir. Aracın düz zeminde hareket edeceği kabul edilerek eğimden kaynaklanan yanal kuvvetler ihmali edilmiştir.

Nesne Tanımlama ve Tepki: Aracın yol üzerindeki statik ve dinamik nesneleri minimum %85 doğruluk oranı ile tespit etmesi beklenmektedir. Bu nesnelerden kaçınmak için manevra yapması veya güvenli bir şekilde durması için tepki süresinin 35 ms'nin altında olması gerekmektedir.

Enerji Tüketimi: Otonom araç denince akla gelebilecek ilk konu araç üzerindeki sensörler ve işlemciler olacaktır. Seviye 5 otonom bir araçta sensör teknolojilerinin ve araçlararası haberleşme, bulut tabanlı veriler gibi yeni bilgi kaynaklarının kullanılmasını zorunlu kılmaktadır. Bunun yanında kendi kendine öğrenebilen yapay zekâ algoritmaları, gelişmiş çevre algılama, planlama ve sensör verilerini işlemek çok daha fazla işlemci gücü gerektirir. Enerji tüketimini artıran bu sensörler ve kontrol elektronünün enerji kullanımını ise geliştirilen yazılımlar daha verimli hale getirecektir.

3. TEORİK ALTYAPI

3.1 GELİŞTİRME ORTAMLARI

OpenCV

OpenCV (Open Source Computer Vision) 1999 yılında Intel tarafından geliştirilmeye başlanan açık kaynak kodlu bir görüntü işleme kütüphanesidir. Willow, NVDIA, AMD, Google gibi şirketlerin desteği ile günümüzde geliştirilmeye devam etmektedir. İlk zamanlar C dili ile geliştirilmeye başlanmış daha sonra birçok algoritması C++ dili ile geliştirilmiştir. OpenCV platformdan bağımsız bir kütüphanedir dolayısıyla Windows, Linux Mac OS, Android gibi platformlarda sorunsuz çalışmaktadır.

OpenCV kütüphanesi içerisinde görüntü işlemeye (image processing) ve derin öğrenmeye (deep learning) yönelik 2500'den fazla algoritma içermektedir. Bu algoritmalar yüz tanıma, nesneleri ayırt etme, plaka tanıma ve daha birçok işlemi rahatlıkla yapabilmektedir.

CARLA Simülatörü (Car Learning Act)

CARLA otonom sürüş sistemlerin geliştirilmesi, eğitilmesi ve test edilmesi açısından Epic Games tarafından Unreal Engine oyun motoru üzerinden geliştirilmiş, ücretsiz ve açık kaynak kodlu bir programdır. Simülasyon içerisinde kentsel düzenlemeler, araç sayısı, yaya insanlar, trafik işaretleri vb. eklenmesine olanak sağlamaktadır. Araç üzerine eklenebilen sensörler, çevre koşulları, statik ve dinamik aktörlerin kontrolü ve harita oluşturma gibi birçok imkana sahiptir.

MATLAB (Automated Driving Toolbox)

Gerçek hayattaki fiziksel sistemleri simüle etmek için sıkça kullanılan programlardan biri de Matlab'dır. Bu projede Matlab'ın otonom sürüş için geliştirdiği Automated Driving araç kutusundan faydalانılmıştır.

Automated Driving Toolbox, [4] ADAS (Advanced Driver – Assistance System) ve otonom sürüş sistemlerinin tasarımını, simülasyonu ve testi için algoritmalar ve araçlar sağlar. Bu araç kutusu sayesinde kamera ve lidar algılama sistemlerinin yanı sıra sensör füzyonu, yol planlama ve araç kontrolörleri tasarlanabilir ve test edilebilir. Ayrıca tasarımın hızlı prototipleme ve HIL (Hardware In the Loop) testi için kullanılmasına olanak sağlayan C/C++ kod üretimini de desteklemektedir.

ROS (Robot Operating System)

ROS (Robot Operating System yani Robot İşletim Sistemi) mobil robot, drone, endüstriyel robot kolları vb. gibi robotik sistemler için geliştirilmiş açık kaynaklı bir işletim sistemidir.

ROS ortamının bazı avantajlarından dolayı Bitirme Çalışması kapsamında tercih ettiğimiz gibi bu avantajlara sırasıyla değinirsek:

- Süreçler arası, mesajlar ve paketler ile iletişimini sağlaması nedeniyle çok sayıdaki farklı algoritmaların ve görevlerin yapıldığı ayrı kod scriptleri haberleşerek kendi içinde hiyerarşik ve senkron çalışabiliyor.
- ROS, gömülü bilgisayarlar ve mikrodenetleyiciler dahil olmak üzere bir veya birden çok platformda kod çalıştırılmak için araçlar ve kütüphanelerle birlikte gelir. ***Bu özellik sayesinde çalışmamızın başarısını ROS ortamında hazır olan ROS Bot 2.0 üzerinde masrafsız bir şekilde test edebildik.***
- Gazebo ve Rviz gibi görselleştirme araçları sayesinde yapılan çalışmayı 3B ve 2B ortamda test edilebiliyor. (Örnek: Hareket Planlama için Rviz ortamında planlanan yörüngeyi ve haritalanan çevreyi görmemiz mümkün değildir. Gazebo ortamında ise gerçek çevre/dünya amaca uygun bir şekilde modellenebilir. ***Bu sayede özellikle drone yarışmalarında, yapılabilecek görevler için ortamın modellenebilmesi ve uçuş algoritmaların önceden tasarlanması test edilmesi mümkündür.***

ROS kurulumu için Linux işletim sisteminin olması yeterlidir. ROS kullanım alanları ve kullanılan işletim sistemlerine göre farklı dağıtımlara sahiptir:

- Ubuntu 16.04(LTS)+ROS Kinetic (Kullanılan)
- Ubuntu 18.04(LTS)+ROS Melodic

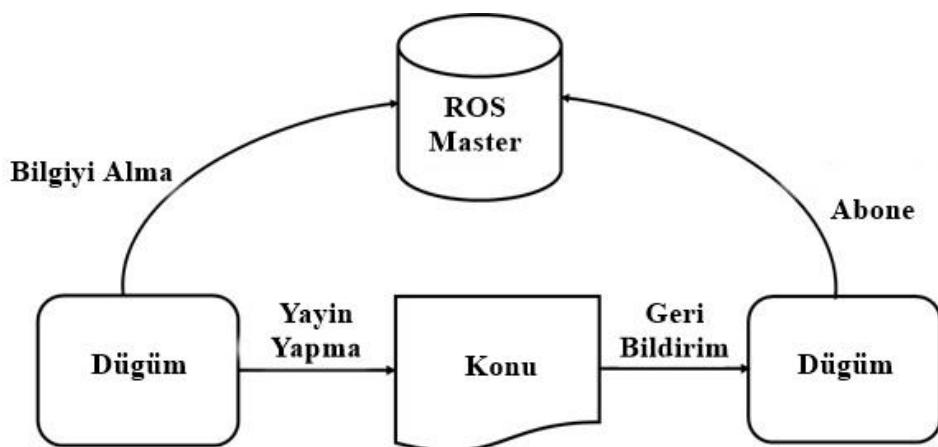
ROS kütüphaneleri sayesinde mobil robotların için aşağıdakileri işlemleri gerçekleştirmesi kolaylaştırılmıştır.

- Motor Sürücüsü
- Hareket Planlaması
- Kamera Sürücüsü
- PID Kontrol Döngüsü
- 3 Boyutlu görselleştirme

Bitirme Çalışması kapsamında Gazebo ve Rviz programları kullanılacaktır. Gazebo hem kapalı hemde açık mekanlar için geliştirilmiş açık kaynak kodlu ücretsiz bir robot deney ortamıdır. Birden fazla robotu 3 boyutlu olarak simüle edebilme özelliğine sahiptir. Gazebo içerisinde bir ortam oluşturarak robotun bu ortam içerisinde sürülmesi ile toplanan kamera ve diğer verilerin Rviz programında görüntülenmesi yapılır. Rviz temel olarak yaptığı işlem mobil robot verilerinin point cloud oluşturarak 2B ve 3B haritalandırma oluşturmaktır. Gazebo üzerinden aracın SolidWorks çiziminin eklenmesi, araç üzerinde sensörlerin eklenmesi, kontrolücünün eklenmesi ve yapay zeka algoritmalarının eklenip test edilmesi mümkündür.

ROS simülasyon sırasında alt kısımda birçok yürütür. Bu alt yazılım ve programlar simülasyon üzerindeki robotun kontrol, veri aktarımı, algoritma işleyişini anlamamızı kolaylaştırır. Bu yazılımların açıklaması aşağıdaki gibidir;

- **Packages (Paketler)**: ROS işlemlerini, kütüphaneleri ve yapılandırmaları içerir
- **Nodes (Düğüm)**: Görev işlemlerinin gerçekleştirildiği birimdir.
- **Master**: Çalışan tüm paket ve düğümleri haberleştiren kontrolü sağlayan birimdir.
- **Publisher (Yayıncı) / Subscriber (Abone)**: Yayıncı görevleri yollar, abone bu görevleri düğüm üzerinde gerçekleştirir.
- **Messages (Mesajlar)**: Düğümler arasındaki standart iletişim için kullanılır. Veri haberleşmesini sağlar.
- **Topics (Konular)**: Farklı aboneler ve yayıcılar arasındaki haberleşme birimidir.
- **Rosbags**: Konulardaki mesajları kayıt almak için kullanılan araçtır.
- **Rviz**: ROS işlemlerinde kullanılan veriyi görselleştirmek için kullanılan programdır.
- **Gazebo**: 3B tasarım yapmak ve simülasyon ortamı hazırlamak için kullanılan programdır.
- **Rqt**: Veri akış diyagramlarını görüntülemek için kullanılan araçtır.



3.2 LOKALİZASYON VE HARİTALAMA

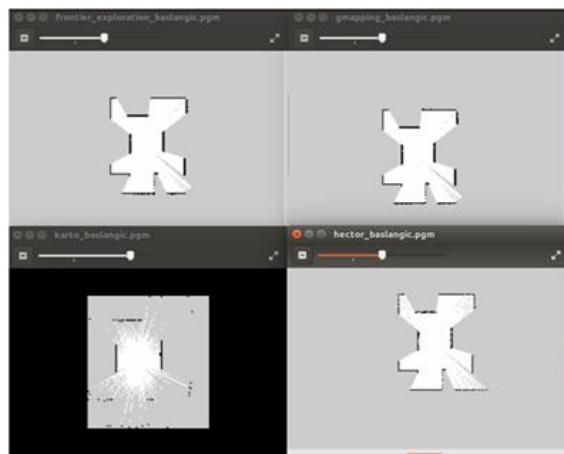
Son zamanlarda geliştirilen otonom araçların iki önemli problemi haritalandırma ve lokasyonunu araç tarafından tanımlanmasıdır. Günümüzde sürücüler, araçlar üzerinde tam kontrol sahibidirler. Sürüş eyleminin, tamamen görsel bilgiye dayanan bir iş olduğu açıktır. Sürücü etrafına bakarak kafasında o anki duruma göre nerede olduğunu, gitmek istediği yere nasıl gidebileceğini çözümleyebilmektedir. Peki, otonom araç hiç bilmediği bir ortama bırakıldığında ortamın haritasını ve kendi konumunu çıkararak istenilen yere gidebilir mi? Bilimsel literatürde Eş Zamanlı Lokalizasyon ve Haritalama yöntemi olarak bilinen SLAM bu probleme bir çözüm önerisi zolabilmektedir. İlk olarak 1986 yılında, Durrant-Whyte, Crovley ve Cheeseman'ın yapmış olduğu SLAM çalışmaları olasılıksal yöntemler içeren bir çalışmadır.

Otonom araçlar ve robotlar bulundukları ortamı taramak ve haritalandırmak için konum servislerini ve çeşitli algoritmaları kullanır. Bu algoritmaların biri de yukarıda bahsedilen SLAM (Simultaneous Localization and Mapping) yani eş zamanlı konum belirleme ve haritalama algoritmasıdır. SLAM kendi kendini kontrol edip karar verme ve verdiği kararları uygulama yeteneğine sahip olan bir aracın/robotun hiç bilinmeyen bir ortama bırakıldığında eş zamanlı olarak kendi lokasyonunu hesaplamasını, ortamı tanıması ve bu doğrultuda işlem yapmasını konu alır. Dış mekânda otonom sürüş yapan robotlar, Küresel Konumlandırma Sisteminin (GPS) kullanarak zamanlama sinyallerine göre konumlarını boylam ve enlem açısından hesaplayabilir. Fakat iç mekânda bulunan robotlar GPS sinyallerini alamaz ve mesafe sensörlerinin yardımına ihtiyaç duyarlar.

Aşağıda örnek olması açısından bazı SLAM algoritmaları verilmiştir:

- Hector SLAM
- Gmapping
- Karto
- Frontier Exploration

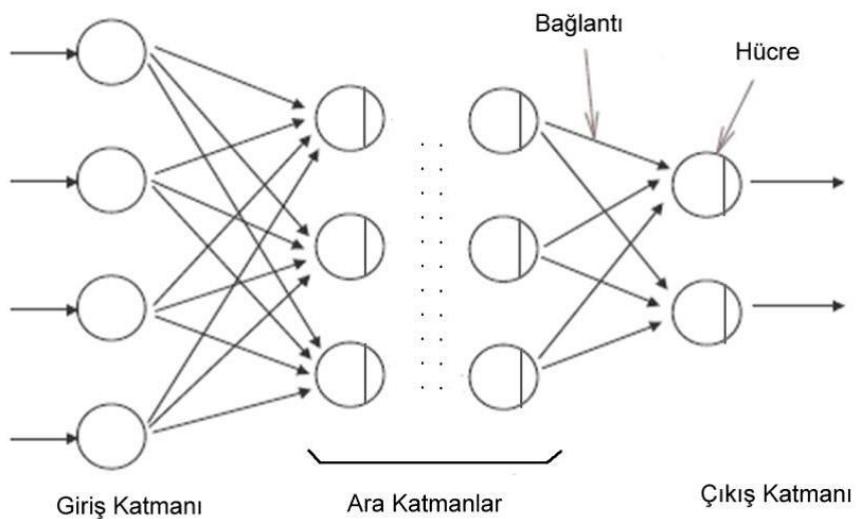
Bu algoritmalar farklı metodlarla geliştirilmiştir kullanım yer ve alanlarına göre değişiklik göstermektedir. Bunun için doğruluk oranı ya da hız için testler gerçekleştirdiğimizde farklı sonuçlar elde edilebilmektedir.



Şekil 10 SLAM Örnekleri

3.3 NESNE TESPİTİ

Proje kapsamında belirlediğimiz nesne algılama görevi için yapay zeka algoritmaları kullanılmasına karar verilmiştir. Yapay zeka algoritmalarında derin öğrenme yöntemi ile nesne algılama yapılacak ve belirlenen özel nesnelere göre sınıflandırılarak karar verme işlemi yapılacaktır. Derin Öğrenme bir makine öğrenme yöntemidir. Verilen bir veri kümesi ile çıktıları tahmin edecek yapay zekayı eğitmemize olanak sağlar. Yapay zekayı eğitmek için hem denetimli hem de denetimsiz öğrenme kullanılabilir. Yapay sinir ağları tipki insan beyni gibi nöronlardan oluşur. Tüm nöronlar birbirine bağlıdır ve çıktıyı etkilemektedir.



Şekil 11 Yapay Sinir Ağlı Benzetimi

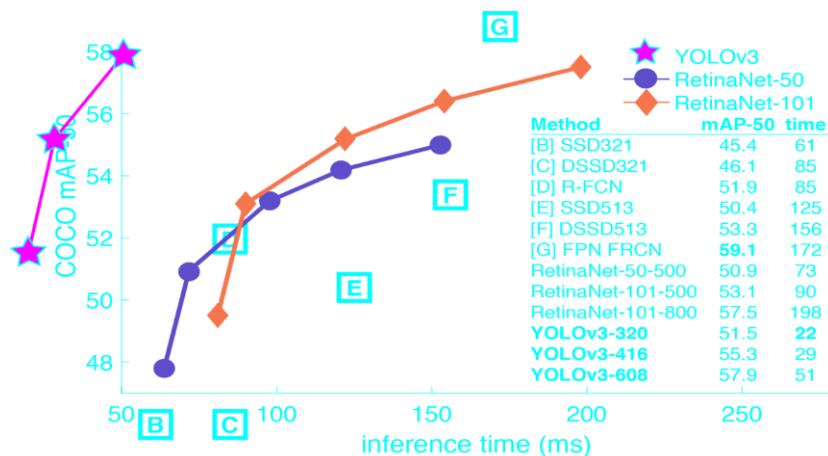
Nöronlar üç farklı katmana ayrıılır:

1. Giriş Katman
2. Gizli Katman
3. Çıkış Katmanı

Giriş Katmanı giriş verilerini alır. Bizim projemizde etiketlenen nesnelere bu katmandan Yapay Sinir Ağına giriş yapar. Gizli katmanda matematiksel işlemlerle işlenir ve Çıkış katmanında bize verilen giriş bilgisinin tahmin edilmiş çıktısı verilir.

3.3.1 YOLOV3 ALGORİTMASI

Proje kapsamında hazır bir algoritma olan YOLOV3 algoritması kullanılmıştır. YOLOv3 algoritması diğer dedektörlere kıyasla daha yüksek doğrulukta ve doğruluk oranına sahiptir. Otonom araçların kısa sürede karar vermesi göz önünde bulundurarak bu seçim yapılmıştır. YOLOv3 darknet altyapısını kullanır ve Şekil XX ‘de diğer dedektörler ile kıyaslanması verilmiştir.



Şekil 12 YOLOv3 kıyaslama sonuçları

YOLO algoritmasına etiketlenmiş veriler giriş katmanına verilir. Bunlar “DUR – Azami Hız – Asgari Hız- Yaya Geçidi” tabelalarıdır. Bu tabelalar Yapay Sinir Ağında eğitmek için sınıflandırmalar belirlenir. Belirlenen sınıflar veri setleri oluşturulur ve işleme başlanır. Algoritma eğitimi tamamlandıktan sonra veri setindeki fotoğraf sayısı, öğrenme oranı, doğru etiketlenmiş fotoğraflar, test ve eğitim verisinin oranına göre belirli bir doğruluk oranında çıktılar verir. Ağırlık dosyasına yazılan veriler, nesne tanımlama algoritmasının olduğu programa giriş olarak verilir ve kontrol algoritmasının olduğu kodlara da burdan alınan çıkış verileri giriş olarak verilir. Ve

tanımlama işlemine göre karar verme görevi bu şekilde tamamlanır.

3.4 HAREKET PLANLAMA

Otonom sürüste güvenlik ve verimlilik açısından en kritik nokta hareket planlamasıdır. Lokalizasyon, haritalama ve obje tanıma görevlerinde elde edilen bilgileri kullanarak hareketi planlar. Bu adımdan sonra, kontrolcüye giriş sinyali verilerek aracın otonom olarak hareketi başlamış olur.

Hareket planlama görevi, harita seviyesinde planlama ve yerel planlama olarak ikiye ayrılır. Harita seviyesinde GPS verisinden faydalananlarak gidilecek konum ve güncel konum arasında optimum bir yol haritası çizilir. Bu projede yerel seviyede olan planlamaya odaklanılacaktır.

Yerel planlamada sürüs senaryosu ve aracın davranışını düşünülerek bir algoritma kurulur. Sürüs senaryosu ise yol yapısına ve yoldaki engellere göre planlanır. Yol yapısı, şerit senaryoları ve düzenleyici unsurlar (trafik kuralları, karayolu, özel otoyol kuralları vs.) yoluyla sürüs senaryosunu etkiler. Yol üzerindeki engeller statik ve dinamik engeller olmak üzere iki farklı sınıfta incelenir. Park halindeki araçlar, yol kenarındaki ağaçlar ve çukurlar statik engeller olarak tanımlanabilirken; diğer araçlar, bisikletli yolcular ve yayalar ise dinamik engellerdir.

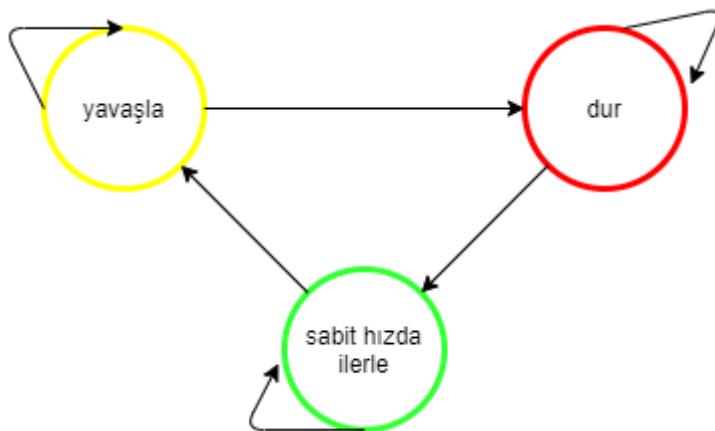
Davranışsal planlama, aracın ne zaman ilerlemesinin güvenli olduğuna karar verir. Yayaları, araçları ve bisikletlileri dikkate alarak yapacağı manevraları belirler. Ayrıca trafik ışıkları ve dur işaretleri gibi düzenleyici unsurlara da bakar.

Davranışsal planlama algoritmasının karar verebilmesi için Sonlu Durum Makinesi (Finite State machine: FSM) kullanılır. Sonlu durum makinası, sınırlı sayıda durumdan, durumlar arası geçişlerden ve eylemlerin birleşmesiyle oluşan davranışların bir modelidir. [5]

Aracın davranışları belirlemede 3 durum ön plana çıkmaktadır.

- Belirli bir referans hızı takip etme (speed tracing)
- Durmak için yavaşlama (decelerate to stop)
- Durma (stay stopped)

Bu üç durum ve geçiş şartları, *figür 1* 'deki gibi Sonlu Durum Makinesine yerleştirilebilir.



Şekil 13 Sonlu Durum Makinesi

Otonom halde seyir eden ego aracın, önündeki araçla (lead car) arasında güvenli mesafe olduğunda, araç sürücü tarafından ayarlanabilen *sabit hızda* devam eder. Herhangi bir durum değişikliği olmadığı sürece araç bu durumu korur.

Ego aracı, bir kavşağa yaklaştığında veya öndeki araç ile arasındaki mesafe güvenli sınırın altına düştüğünde, araç *yavaşla* durumuna geçer. Bu durum kavşağa girene kadar veya öndeki araç ile güvenli takip mesafesinde olana kadar devam eder.

Ego aracı, bir kavşağa girdiğinde eğer trafik lambasında kırmızı ışık yanıyorsa veya kavşağı kullanan başka bir araç var ise bu durum ortadan kalkana kadar *dur* durumunda bekler. Kavşağa yaklaşan bir araç olmadığı ve yeşil trafik ışığı yandığında araç *dur* durumundan çıkararak verilen referans hızı takip etmeye başlar.

Farklı senaryolarda aracın yapacağı manevralar da olacağından, birden fazla Sonlu Durum Makinesi hiyerarşik olarak kullanılabilir. Hareket planlama, ile ilgili detaylar, raporun Tasarım kısmında verilecektir.

3.5 ARAÇ DİNAMİĞİ VE KONTROL

Kontrol yaklaşımı hakkında bilgi vermeden önce üzerinde çalışacağımız araç modelini iyi tanıtmamız gerekiyor. Aracın elektrik motoruyla mı yoksa içten yanmalı motorla mı tahrik edildiğini bilmek ve ona göre power train (güç aktarımı) modelini çıkarabilmek gereklidir. Ayrıca aracın viraj dinamiklerini de iyi modelleyebilmek ve oluşturduğumuz modele göre yörünge izleme problemi için uygun kontrolcü tasarımları yaklaşımını seçmeliyiz. Bu kapsamda otonom sürüs yapmasını isteyebileceğimiz araç tipini dört kategoriye ayıralım:

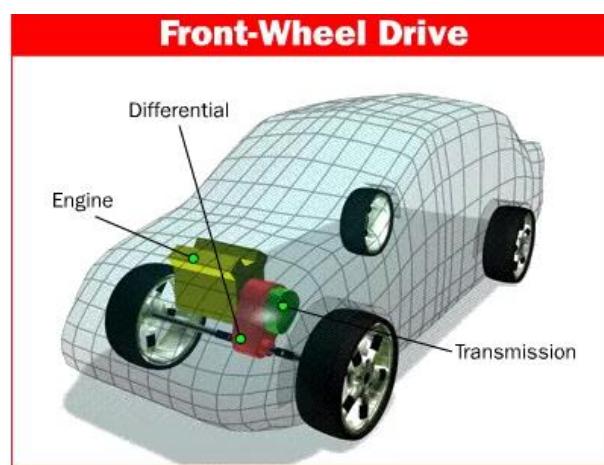
- İçten yanmalı motorla tahrik edilen binek araçlar (Ackermann Steering)

- Elektrik motoruyla tahrik edilen binek araçlar (Ackermann Steering)
- Elektrik motoruyla tahrik edilen ölçeklendirilmiş (1/10) araçlar (Ackermann Steering)
 - ✓ *Model aracımız bu kategoriye uyyor. Güç aktarma modelini oluştururken ve direksiyon açısı kontrolünü yaparken; aracın FWD (önden çekişli olması), DC Motor tahriki ile boyuna hız değerinin atanması,sahip olduğu diferansiyel ile viraj alması ve Servo Motor tahriki ile viraj alması dikkate alınmıştır. Ackermann Steering ve Basitleştirilmiş Bisiklet Modeli yaklaşımlarını bu kapsamda incelemiştir.*
- Elektrik motoruyla tahrik edilen mobil robotlar (Differential Drive)
 - ✓ *ROS Bot 2.0 isimli mobil robot bu kategoriye uymaktadır. Mobil robotun RWD (arkadan itişli) olması ve 2 farklı DC Motordan (sol ve sağ motor) üretilen moment farkı sayesinde viraj alması dikkate alındı. Mobil robotumuz, ROS platformu kütüphanesinde kayıtlı olduğu için direksiyon açısı kontrolcüsü hariç tüm algoritmalarımızı ROS Bot 2.0 üzerinde test ettik.*

Dört ana kategorideki araçlar da kendi içinde farklı özellikler barındırabilir. Önemli olan sahip olduğumuz aracın özelliklerini iyi tanımak ve modellemeyi başarılı yapabilmektir. Sahip olduğumuz aracı daha iyi tanıyalım için sırasıyla: FWD, RWD,AWD,Power Train, Ackerman Steering, Simplified Bicycle Model ve Differential Drive Model kavramlarına bir göz atalım.

Front-Wheel Drive (Önden Çekişli Araçlar)

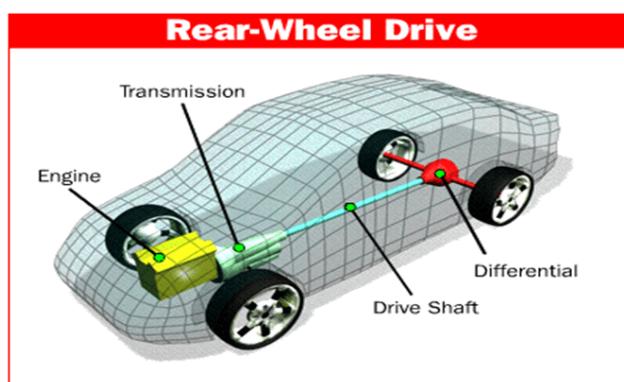
Önden çekişli araçlarda motorda üretilen güç, ön tekerlekler aracılığıyla yola aktarılır. Motor sadece ön akslar ile bağlantılıdır. Önden çekişli araçlarda motorda üretilen güç öncelikle şanzımandan diferansiyele gider ve diferansiyel üzerinden hareket akslardan tekerleklerile iletilir. FWD tipi arabaların çalışma prensibi, ön tekerlerin hareketi ile tüm aracın gitmesini sağlamaktır.



Şekil 14 Önden Çekişli Otomobil için güç aktarma organlarının yerlesimi

Rear Wheel Drive (Arkadan İtişli Araçlar)

Arkadan itişli araçlarda, motor şanzımana doğrudan bağlı olsa da arka tekerleklerde motorda üretilen güç şaft ile iletilir. Yani, arka tekerlere bağlı diferansiyel gücünü şafttan alır. Performansı yüksek motorlarla kullanılan RWD konfigürasyonunda araç içi ağırlık dağılımı dengelidir. Yön veren tekerlerin konumu (arka tekerler), motorun yakın olduğu güç aktaran tekerlekler(ön tekerler) aynı olmadığından dolayı,farklı sürüs teknikleri denenebilir. Bu yüzden profesyonel yarışma sürücülerinin kullandığı veya performans odaklı lüks araçlarda tercih edilir. RWD sistemlerin dezavantajlarından biri ise güç aktarımını sağlayan şaft yüzünden araç ağırlığının artmasıdır.



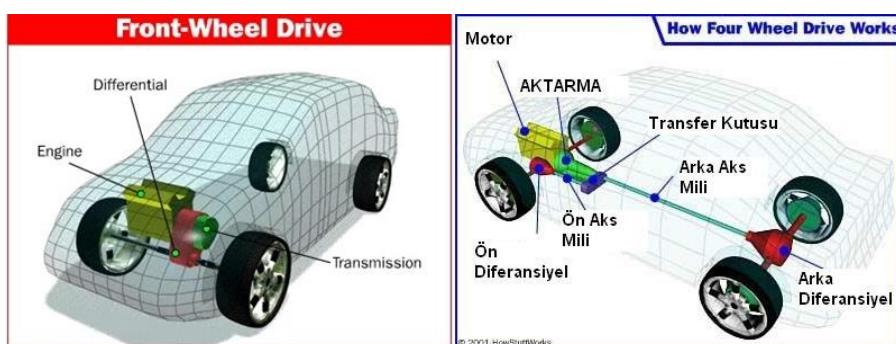
Şekil 15 Arkadan İtişli Otomobil için güç aktarma organlarının yerlesimi

Dört Çekişli Araçlar (All Wheel Drive and Four Wheel Drive)

Önden çekişli ve arkadan itişli sistem özelliklerinin aynı araçta yer aldığı bu sistemde tüm tekerlekler motora bağlıdır. Aracın merkezinde ve arkada olmak üzere iki diferansiyel bulunur. Arazi araçlarında ve yüksek performanslı otomobillerde (Örneğin: Jeep marka) bulunur ve zorlu yol koşulları için tasarlanmıştır.

AWD tipi araçların başlıca avantajları ve dezavantajları şunlardır:

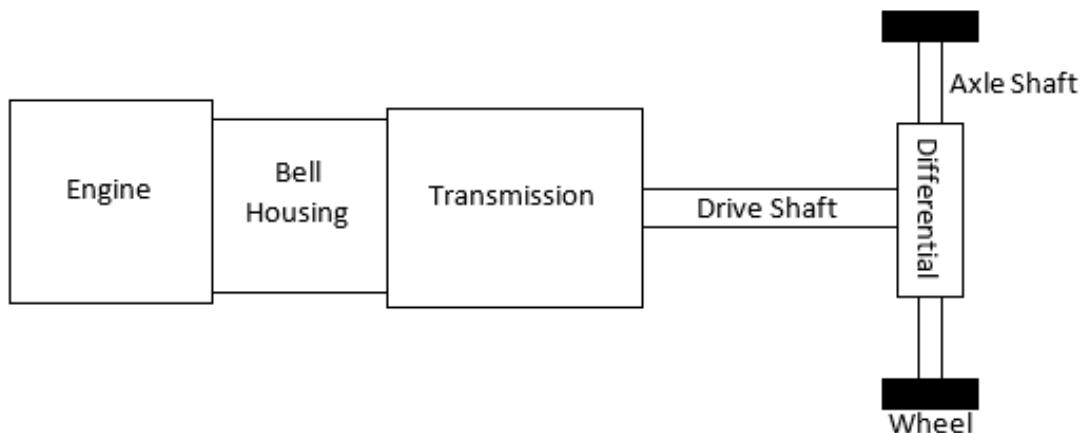
- Sürücü gaz pedalına bastığında dört tekerle de aynı anda güç iletilebilediği için çok çabuk ve seri kalkış yapılabilir.
- Zorlu virajlarda veya kaygan zemin içeren yol koşullarında tüm tekerlerin sürücü tarafından kontrol edilmesinden dolayı yol tutuş özelliği çok yüksektir.
- Diğer çekiş konfigürasyonlarına göre AWD tipi araçlar daha fazla sayıda mekanik parça içerir ve yapısı daha karmaşıktır, bu yüzden bakım maliyeti ve araç fiyatları yüksektir.



Şekil 16 Dört Çekişli Otomobil için diferansiyel yerleşimlerinin iki farklı konfigürasyonu

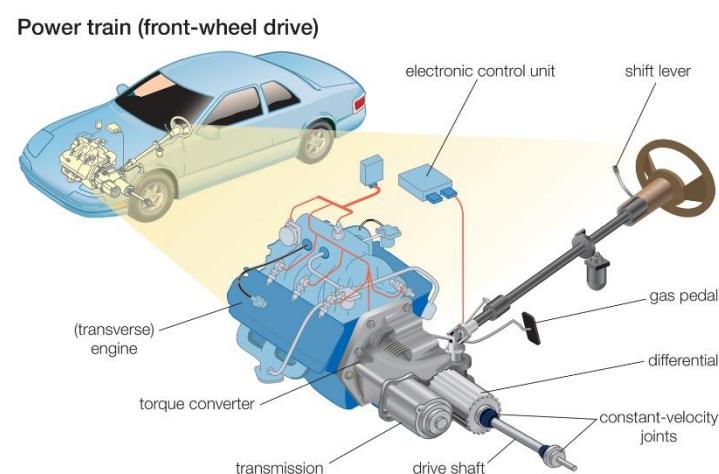
Power (Drive) Train Model

Power veya Drive Train Model olarak bilinen model, motorda üretilen gücün tekerler üzerine aktarılmasını her güç aktarma organını ayrı ayrı modelleyerek ifade eder. İçten yanmalı motorla tahrik edilen, elektrik motoruyla tahrik edilen veya hibrit motorla tahrik edilen her araç için güç aktarma modeli farklıdır. **Bitirme Çalışması kapsamındaki 1/10 model RC aracımız için, hız değeri Fırçasız DC Motor tahrikleri ile belirlenirken direksiyon açısı ise özel tasarım yüksek tork değerlerine sahip steering servo özelliğine sahip DC Servo Motorla kontrol edilir.**

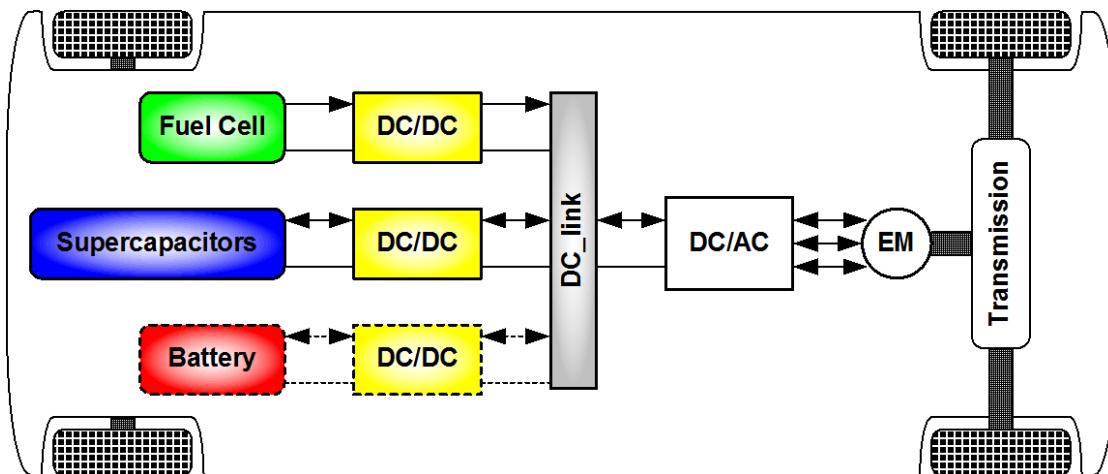


Şekil 17 İçten yanmalı motora sahip bir otomobilin güç aktarma modeli şeması

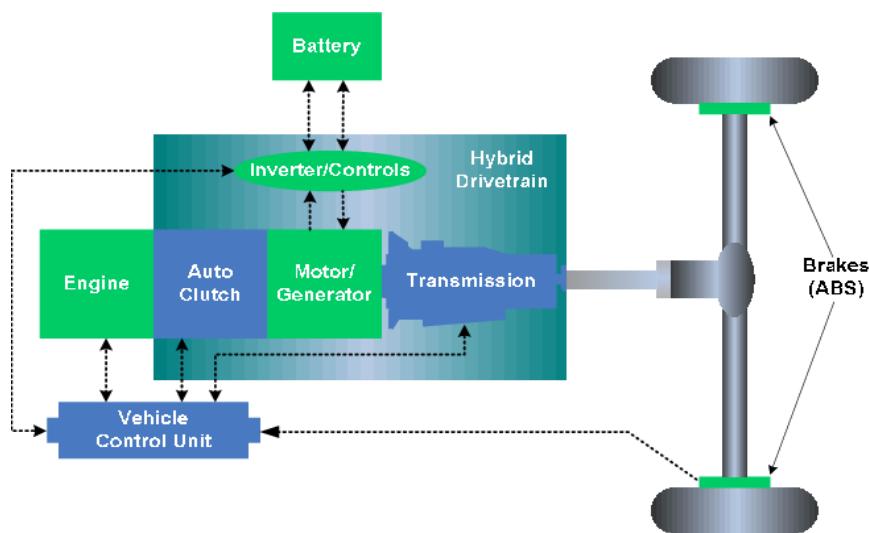
Dolayısıyla her aracın kendine özgü güç aktarma modeli vardır ve genel yaklaşımıları öğrenerek her türlü aracın güç aktarma modelini elde edebiliriz. Power Train Modeli yaklaşımını bilmek özellikle model aracın tasarımına odaklanan bir proje için her elemanın amaca uygun ayrı ayrı tasarlanmasında önemli bir çözümüdür. **BÇ kapsamında Power Train modeline benzer şematik yaklaşımıları motor kontrolü ve uzaktan kumandanın devre dışı bırakılması amacıyla kullandık.**



Şekil 18 Önden çekişli bir aracın güç aktarma organları patlatılmış montaj gösterimi



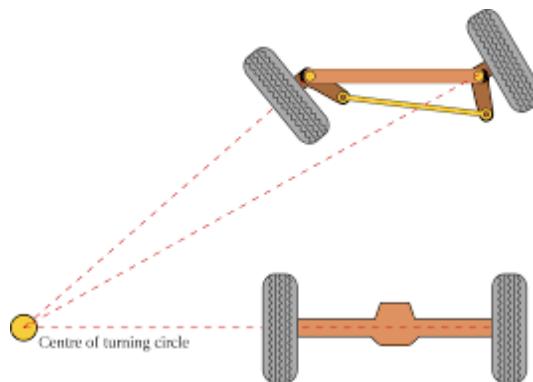
Şekil 19 Elektrik motoruyla tıhrik edilen bir elektrikli otomobil için güç aktarma modeli şeması



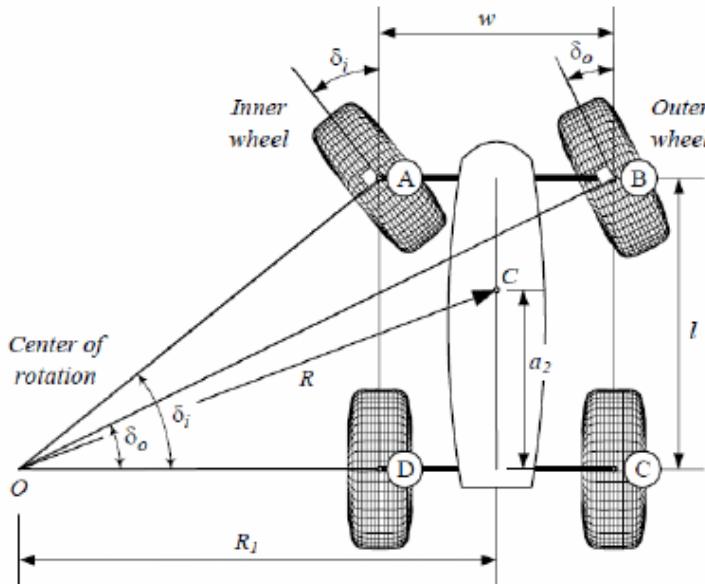
Şekil 20 Hibrit motorlu bir otomobilin güç aktarma modeli

Ackermann Direksiyon (Steering) Geometrisi

Ackermann Direksiyon Geometrisi, aracın viraj alırken tüm lastik/teker/akslarının aynı dönme merkezi etrafında (IC:ani dönme merkezi- yaklaşımına benzer şekilde) döndüğünü varsayıarak oluşturulan geometrik bir modeldir. Ackermann geometrisinin amacı, bir eğri etrafındaki yolu takip ederken, lastiklerin yanlara kaymasını önlemektir. Ackermann geometrisi, aracın viraj aldığı taraftaki tekerleğin daha fazla çekiş ve viraj alırken daha az lastik kaymasına sahip olması için dış tekerleğe göre daha küçük bir dönüş yarıçapına sahip olması için çalışır.



Şekil 21 Basitleştirilmiş Ackermann Geometrisi



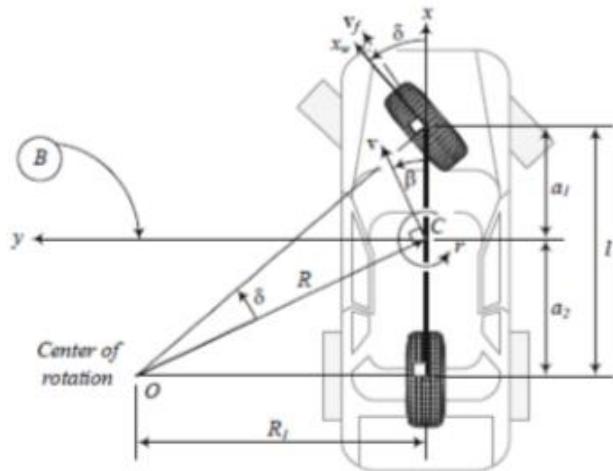
Şekil 22 Ackermann Geometrisi için önemli açı ve uzunlukların şematize edilmesi

Aracın Matematiksel Modeli ve Durum Uzayı Gösterimi

Yörünge takibi yapabilmemiz için aracımızın matematiksel modelini çıkarmamız gereklidir. Aracımız için oluşturduğumuz matematiksel modele göre çeşitli kontrolcü tasarımları yaklaşımları mevcuttur. Aracımızı dört tekerleğin hepsiyle modellemek yerine, Basitleştirilmiş Bisiklet Modelini kullanarak modellemek mümkündür. Bisiklet Modeli de kendi içinde iki çeşide ayrılır ve her birinin tercih sebebi farklıdır. Basitleştirilmiş Bisiklet Modeli kendi içinde üçe ayrılır:

- Geometrik Bisiklet Modeli
- Kinematik Bisiklet Modeli
- Dinamik Bisiklet Modeli

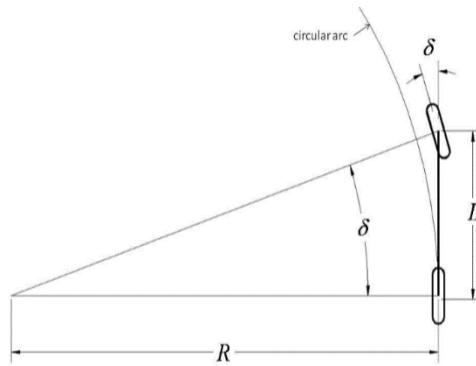
Aracın yanal dinamiklerini modellemek için, aracın ön ve arka tekerlekleri şekil-1'deki gibi bir tekerleğe indirgenmiş olarak sunulabilir. Bu yönteme **bisiklet modeli** denir. Bir bisiklet modeli kinematik, dinamik, doğrusal veya doğrusal olmayan türlerde olabilir.



Şekil 23 Bisiklet Modeli Genel Gösterimi

Geometrik Bisiklet Modeli

Geometrik Bisiklet Modeli, aracı sadece uzunluk ve açı değerleriyle ifade eden basitleştirilmiş bisiklet modelidir. Geometrik yol izleme amacıyla, sadece bisiklet modelinin, iki ön tekerleği ve iki arka tekerleği bir bisiklet gibi iki tekerlekli bir model oluşturmak üzere birleştirerek dört tekerlekli arabayı basitleştirdiğini belirtmek gerekir. İlkinci basitleştirme, aracı sadece bir düzlemdede hareket edebilmesidir.



Şekil 24 Geometrik Bisiklet Modeli

$$\tan(\delta) = \frac{L}{R}$$

δ: Direksiyon açısını gösterir.

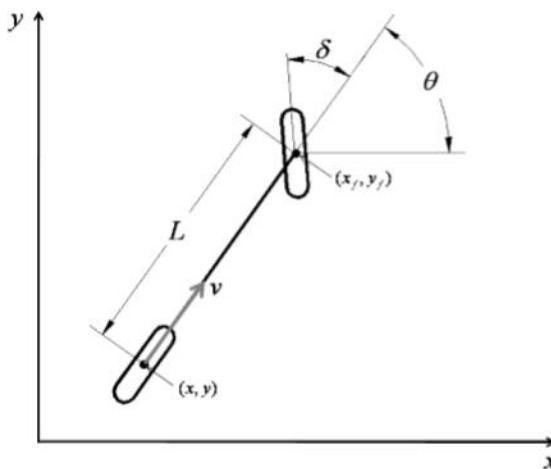
L: Aracın ön aks merkezi ile arka aks merkezi arasındaki dikey mesafedir.

R: Aracın dönme merkezine olan radyal uzaklığını gösterir.

Bu model, düşük hızlarda ve orta direksiyon açıllarında bir otomobilin hareketine oldukça iyi yaklaşır.

Kinematik Bisiklet Modeli

Kinematik Bisiklet Modeli, geometrik bisiklet modelinden farklı olarak aracın boyuna hızı ve yaw (yalpa) açısını da dikkate alarak bir model yaklaşımı geliştirir. Bu modelde teker kayması sıfır kabul edilmiştir ve yanal kuvvetler dikkate alınmamıştır. Kinematiklerin bu şekilde basitleştirilmesi, her bir ön tekerleği ayrı ayrı kontrol etmek yerine artık sadece bir direksiyon çıkışısı (kontrolcüden atanmış direksiyon sinyali-direksiyon açısı ve yönünü içeren komut) ile kontrol etmek mümkün olduğu anlamına gelir.



Şekil 25 Kinematik Bisiklet Modeli

(x_f, y_f) aracın ön teker merkez koordinatlarını gösteriyor.

(x, y) aracın arka teker merkez koordinatlarını gösteriyor.

L aracın ön aksı ile arka aksı arasındaki uzaklığı gösterir.

δ aracın direksiyon açısını gösterir.

θ aracın yaw (yalpa) açısını gösterir.

$$\dot{x} = v * \cos(\theta)$$

$$\dot{y} = v * \sin(\theta)$$

$$\dot{\theta} = \frac{v * \tan(\delta)}{L} = \frac{v}{R}$$

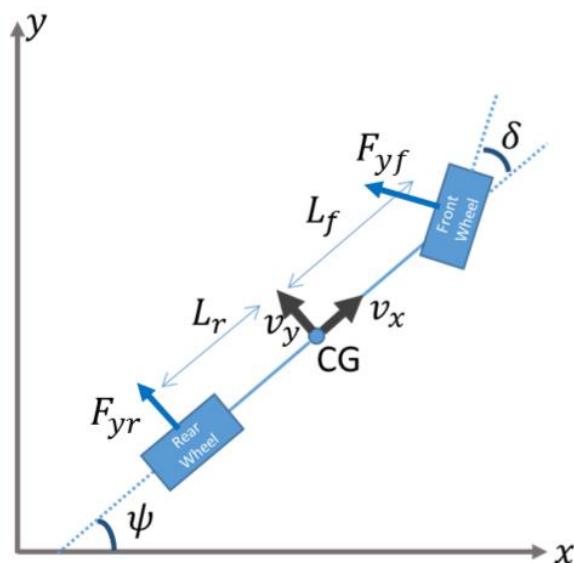
$$\tan(\delta) = \frac{L}{R}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ \tan(\delta) \\ \frac{L}{v} \end{bmatrix} * v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} * \dot{\delta}$$

Dinamik Bisiklet Modeli

Dinamik Bisiklet Modeli, Kinematik-Geometrik Bisiklet Modelinden farklı olarak teker kaymasını sıfırdan farklı kabul eder ve yanal kuvvet etkilerini, atalet değerlerini ve tekerlerin viraj sürtünme katsayısını dikkate alır. Yüksek hızlarda, eğimli yollarda, direksiyon açısı aralığının büyük olduğu keskin virajlı durumlarda Kinematik Bisiklet Modeline göre daha başarılı sonuç verir. Kontrolcü tasarımda istenirse Kinematik Bisiklet Modeli yerine Dinamik Bisiklet Modeli referans alınabilir ve ona göre kontrolcü tasarlanabilir.

BÇ (Bitirme Çalışması) kapsamında aracın boyuna hızının maksimum değerini 5 m/s olarak belirledik ve test parkuru olarak da eğimsiz yolu tercih ettiğimiz için kontrolcümüzü Kinematik Bisiklet Modelini referans alan bir yaklaşımına göre tasarlamayı tercih ettim. Ayrıca, Yapay Zeka ile üretilen referans yörüğünün içeriği waypoint aralıkları sık olacağı için aracımız sürekli bir yörüğe yerine smooth (yalın, pürüzsüz) bir yörüğe izleyecektir.



Şekil 26 Dinamik Bisiklet Modeli Gösterimi

δ direksiyon açısını gösterir.

ψ yaw (yalpa) açısını gösterir.

v_x aracın boyuna hızını gösterir.(CG'nin orijin olduğu eksen takımına göre)

v_y aracın yanal hızını gösterir.(CG'nin orijin olduğu eksen takımına göre)

CG aracın ağırlık merkezi konumunu gösterir.

L_f ağırlık merkezinin ön aks merkezine olan uzaklığını gösterir.

L_r ağırlık merkezinin arka aks merkezine olan uzaklığını gösterir.

F_{yf} ön aks üzerindeki yanal kuvvet vektördür.

F_{yr} arka aks üzerindeki yanal kuvvet vektördür.

F_{xf} ön aks üzerindeki boyuna kuvvet vektörü(bu model için sıfır kabul edildi, yani aracın boyuna ivmesi sıfır kabul edildi.)

I_z aracın yaw(yalpa) durumuna karşı z eksenindeki ataleti gösterir.

α_f ön lastiğin kayma açısıdır.

α_r arka lastiğin kayma açısıdır.

C_f ön lastik için viraj sertlik katsayısıdır.

C_r arka lastik için viraj sertlik katsayıdır.

$$m * \dot{v}_y = -C_f * \left[\tan^{-1} \left(\frac{v_y + L_f * \dot{\psi}}{v_x} \right) - \delta \right] * \cos(\delta)$$

$$-C_r * \left[\tan^{-1} \left(\frac{v_y - L_r * \dot{\psi}}{v_x} \right) \right] + m * v_x * \dot{\psi}$$

$$I_z * \ddot{\psi} = -L_f * C_f * \left[\tan^{-1} \left(\frac{v_y + L_f * \dot{\psi}}{v_x} \right) - \delta \right] * \cos(\delta)$$

$$+ L_r * C_r * \left[\tan^{-1} \left(\frac{v_y - L_r * \dot{\psi}}{v_x} \right) \right]$$

Küçük açılar için $\tan(\theta) \approx \theta$ ve $\cos(\theta) \approx 1$ yaklaşımını kullanırsak lineerleştirilmiş dinamik modeli elde ederiz.

$$m * \dot{v}_y = \frac{-C_f * v_y - C_f * L_f * \dot{\psi}}{v_x} + C_f * \delta$$

$$+ \frac{-C_r * v_y - C_r * L_r * \dot{\psi}}{v_x} - m * v_x * \dot{\psi}$$

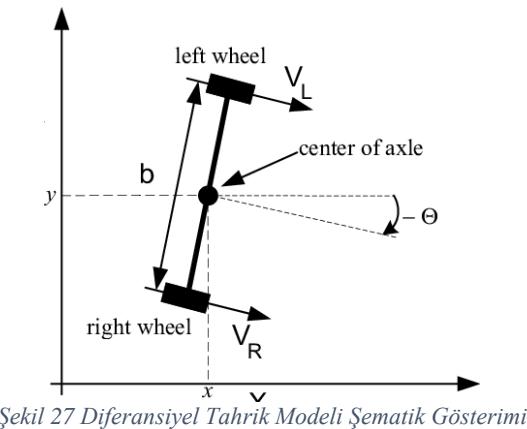
$$I_z * \ddot{\psi} = \frac{-L_f * C_f * v_y - L_f^2 * C_f * \dot{\psi}}{v_x} + L_f * C_f * \delta$$

$$+ \frac{L_r * C_r * v_y - L_r^2 * C_r * \dot{\psi}}{v_x}$$

Lineerleştirilmiş dinamik model genelde simülasyonlar için kullanılır, gerçekçi sonuç elde etmek için lineer olmayan model kullanılmalıdır.

Differential Drive Model (Diferansiyel Tahrik Modeli)

Genellikle mobil robotlarda kullanılan bu model, sol ve sağ tekere bağlanmış iki farklı DC Motorun farklı tahrik değerleri üretmesi prensibiyle yani diferansiyel tahrikle robotun viraj alması prensibine dayanır.



Şekil 27 Diferansiyel Tahrik Modeli Şematik Gösterimi

4. EKİPMAN SEÇİMİ

4. 1 NVIDIA Jetson TX2

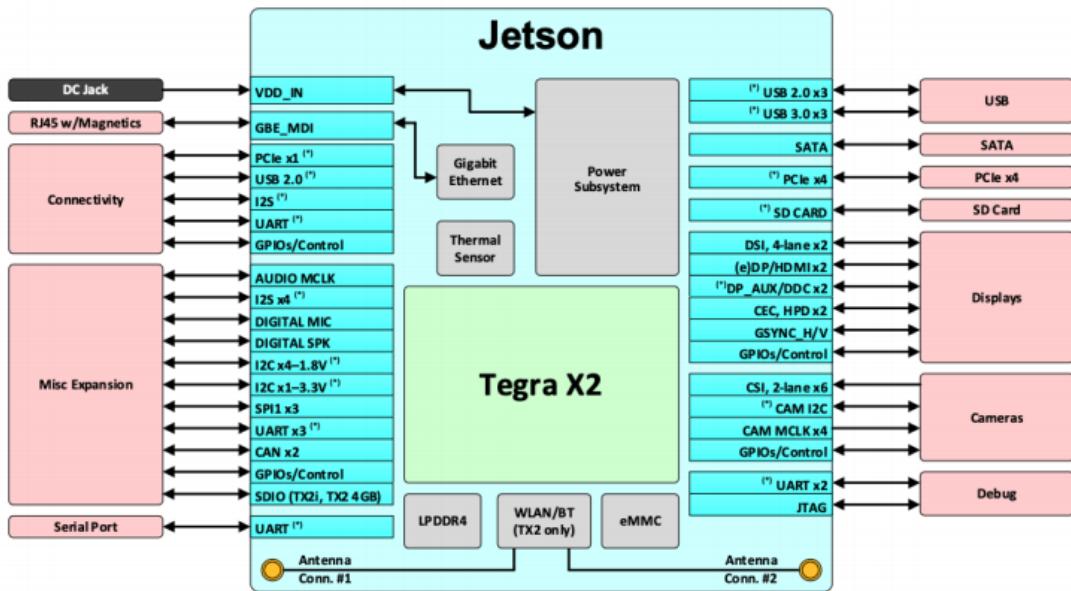
Otonom sürüş, yüksek miktarda paralel işlem gücü gerektirdiğinden, daha verimli sinir ağları ve görüntü işleme için çok sayıda grafik işlem birimi (GPU) çekirdeği kullanılması gereklidir. [6]



Şekil 28 Nvidia Jetson TX2

GPU'lar, büyük miktarda veriyi, özellikle görüntülerini hızlı bir şekilde işlemek için etkin yetenekleriyle bilinir. Bu, otonom otomobiller geliştirirken faydalı ve gerekli bir araçtır çünkü hesaplama sisteminin, aracın duruma uygun şekilde tepki vermesi için otomobilin ortamındaki binlerce görüntüyü hızlıca filtrelemesi gereklidir. Projemiz için NVIDIA'nın Jetson TX2'sini seçildi.

Bu tümleşik bilgi işlem kartı, hem grafik işlemci birimi (GPU) hem de merkezi işlemci birimi (CPU) aynı çip üzerinde barındıran “high performance AI at the edge” için özel olarak tasarlanmıştır. [7] Çipin sistem yapısı üzerinden hızlı bir şekilde veri aktararak hem GPU hem de CPU kullanarak paralel işleme gerçek zamanlı olarak elde edilebilir. Bu iş birliği, otonom otomobil için derin sinir ağının geliştirilmesine yardımcı olacak daha hızlı görüntü işleme sağlar.



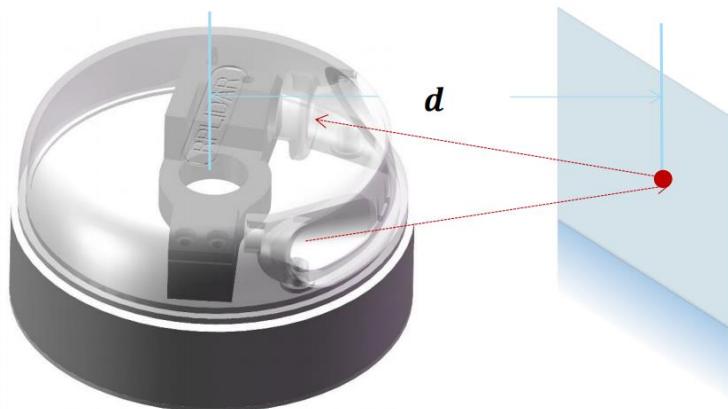
Şekil 29 Jetson TX2 Serisi Modül Blok Şeması [8]

Jetson TX2, Nvidia tarafından Nvidia Jetson serilerinin bir parçası olarak geliştirilen ikinci nesil gömülü bilgi işlem kartıdır. Tegra X2'yi kullanan Jetson TX2, öncekinden daha yüksek işlem gücüne sahipken daha iyi güç verimliliğine sahiptir. Bu, gücün sınırlı olduğu ortamlarda yüksek güçlü hesaplamalar gerektiren uygulamalar için idealdir. Bu tür uygulamalara örnek olarak otonom araçlar, dronlar ve sanal gerçeklik verilebilir.

Bu projede Jetson TX2 kartı görüntü işlemenin yanı sıra, aracın kontrolcüsünün de koşturulduğu bir ortam içermektedir. Model aracın kumanda devresini by-pass ederek, boylamsal hız ve dümen açısını belirleyen yazılım parçalarını çalıştırarak bir otopilot (otonom sürüş) görevi üstlenir.

4.2 RP LIDAR A3

Lazer ışınları yardımıyla 3 boyutlu tarama yapabilen elektronik sensörlerdir. Çalışma prensibi yollanan ışının bir cisimle çarparak geri dönme süresinin hesaplanarak geçen süre ile orantılı olarak uzaklığın tespit edilmesini sağlar.



Şekil 30 Lidar Çalışma Prensibi

Yukarıdaki şekilde görüldüğü gibi gönderilen lazer ışını yansımaları optikle yakalanır ve ışık hızının geçen zamana oranı ile d mesafesi hesaplanır. Dolayısı ile anlık olarak gönderilen lazer ışını sayısı ne kadar fazla ise elde edilen doğruluk oranı o kadar yüksektir. SLAMTEC firmasının ürettiği RPLidar serisinin A3 modeli saniyede 16000 lazer ışını gönderemektedir ve 25 metreye kadar nesneleri tespit edebilmektedir.



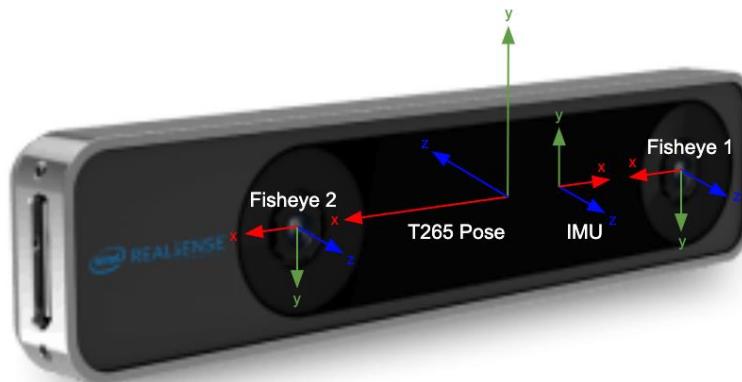
Şekil 31 RPLidar A3

RPLidar A3 modeli fırçasız motor kullanır bu sayede titreşimi, sürtünmeyi azaltır ayrıca daha gürültüsüz çalışır. Bu model düşük güç tüketimi ile sistemin pil ömrünü uzatmaktadır.

4.3 Intel T265 Takip Kamerası

Araç temel çevre görüşü için kamera sensörü kullanılacaktır. Aracın çevre algılamasında alınan verilerde yüksek doğruluk oranı aranmaktadır. Proje kapsamında Intel D435 derinlik kamerası ve Intel T265 takip kamerası arasında kıyaslama yapılmıştır. Derinlik verisini Lidardan alınan veri ile Intel T265 kamerasından alınan görüntü ile füzyonlayarak iki veri elde edilebilinmektedir. Buna ek olarak Intel T265 takip kamerası odometri veriside sağlamaktadır. Yani Intel T65 ve RP Lidar A3 sensörleri ile sadece sensör özelliklerini sayesinde daha fazla veri elde edilebilmektedir. Bu sebeplerden dolayı Intel T265 takip kamerası kullanılmasına karar verilmiştir.

Intel T265 kamerası 6 serbestlik derecesine sahiptir. Üzerinde bulunan iki farklı kameradan alınan veriler sayesinde kod üzerinden bu veri işlenerek füzyonlanmış görüntü ve IMU verisi elde edilebilmektedir.



Şekil 32 Kamera görüşü ve IMU sensörlerinin temsil edilmesi

T265 kamerası 848 X 800 çözünürlük ile 163 derecelik geniş görüş açısına sahiptir. Kamera ayrıca 1.5 Watt güç ile çalışarak pil tasarrufu sağlamaktadır. Kamera içerisinde bulunduğu imu (inertial measurement unit) sensörü sayesinde harici sensörler gerekmeksizin eş zamanlı lokalizasyon ve haritalama işlemlerini gerçekleştirebiliyor.



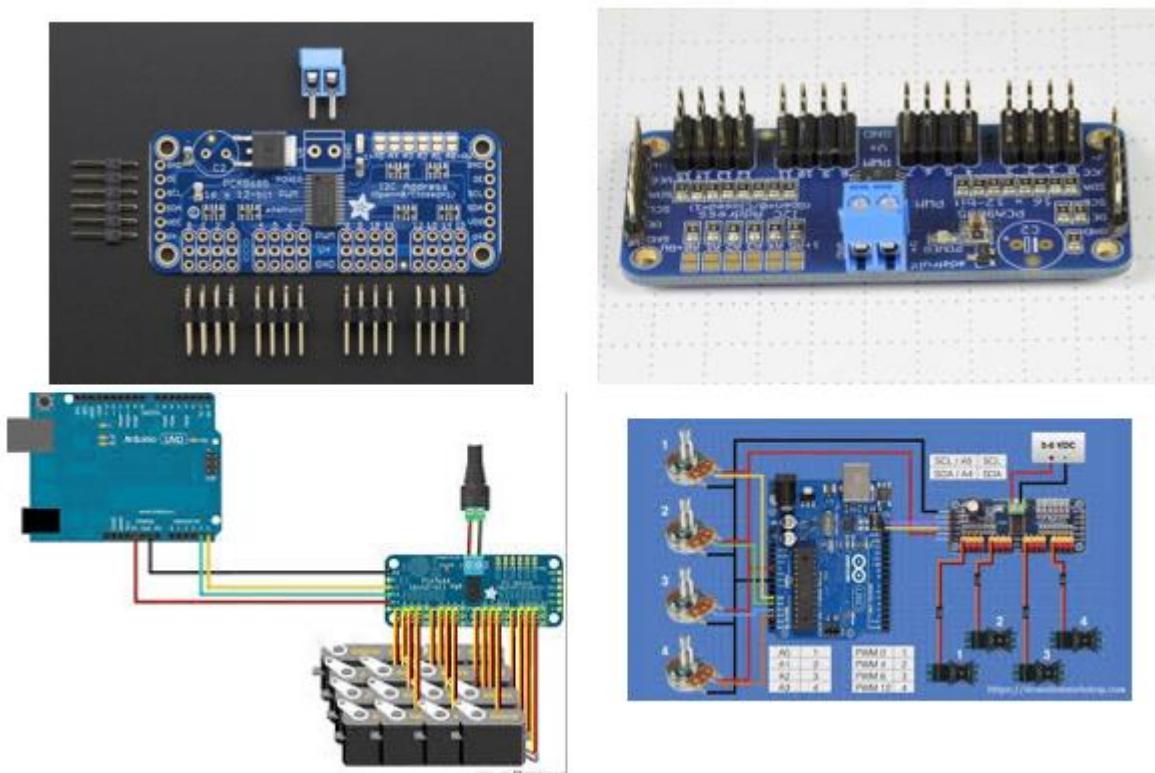
Şekil 33 Intel Realsense programı ara yüzü ve alınan verilerin görselleştirilmesi

4.4 PCA9685 PWM Modülü

Adafruit firmasının bir ürünü olan PCA9685 I2C-PWM Modülü, temelde iki tip uygulamalarda kullanılmak üzere üretilmiştir:

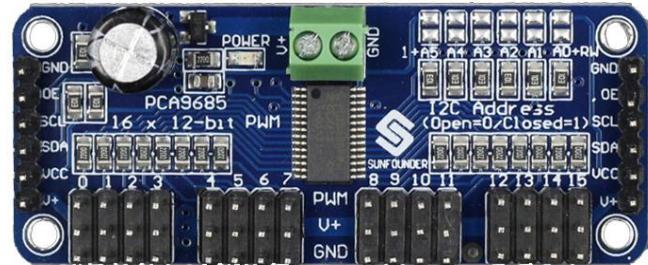
- RGBA(Red-Green-Blue-Amber) renkli aydınlatma uygulamaları için optimize edilmiştir. PCA9685, 16 kanallı I2C veri yoluna sahip bir LED denetleyicisidir.
- Her LED çıkışı, sabit bir frekansa sahip (tüm çıkışlar aynı PWM frekansına ayarlanmıştır.) 12 bit çözünürlüklü PWM denetleyicisine sahiptir. *Bu sayede PCA9685, 16 kanallı 12 bitlik bir servo sürücüsü olarak kullanılabilir. PCA9685, tek başına 16 servoyu kontrol edebilir, ayrıca en fazla 62 sürücüyü kartına da kademeli olarak bağlanabildiği için 16*62=992 servoyu PCA9685 kullanarak kontrol edebileceğimiz anlamına gelir.*

Bitirme Çalışması kapsamında, aracımızın steering servosunu kumanda devresinin iptal olduğu(kumandasız) konfigürasyonda yani NVIDIA JETSON TX2 denetleyicisi üzerinden kontrol edebilmemiz gerekiyor. Kontrolcünün atadığı direksiyon(steering) açısı sinyali, yönü ve büyüğlüğü ile NVIDIA JETSON TX2 merkezi işlem biriminden PCA9685 PWM modülüne iletilmelidir. PCA9685 PWM modülü üzerinden de servo motor kontrol edilecek ve servo motorun uygun açıda doğru yöne dönmesiyle aracımız güvenli bir şekilde viraj alabilecektir.



Şekil 34 PCA9685 [sol - sağ üst] Ardnuio bağlantısı [sol alt] kontrolü [sağ alt]

Son olarak PCA9685 modülünün güç kaynağı, mikrodenetleyici ve servo motorlara hangi pinler üzerinden bağlanacağından bahsedelim ve ilgili pinlerin fonksiyonlarına değinelim:



Şekil 35 PCA9685 Modülü Giriş/Çıkış Pinleri ve Özellikleri

Bağımsız Güç Kaynağı	16 Kanallı PWM Servo Sürücü-PCA9685
VCC	V+
GND	GND

Mikro denetleyici	16 Kanallı PWM Servo Sürücü-PCA9685
5V	VCC
GND	GND
SDA	SDA
SCL	SCL
Vin	V+

Servo Motor	16 Kanallı PWM Servo Sürücü-PCA9685
Turuncu Kablo	PWM
Kırmızı Kablo	V+
Kahverengi Kablo	GND

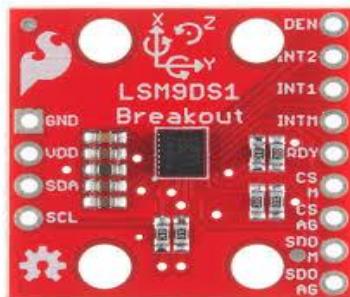
PCA9685 pinlerinin temel fonksiyonları şöyledir:

- GND:** Güç ve sinyal topraklama pimi, bağlı olmak zorundadır.
- VCC:** Mantık güç pimidir, çıkış lojik-1 iken 3-5 V olacak şekilde ayarlanabilir, pull-up dirençleriyle de kullanımı mümkündür.
- V+:** Servo Motorlara miktarını ayarlabilmemizin mümkün olduğu güç dağıtan pindir.
- SDA:** I2C saat pinidir ve kontrol pinidir.
- SCL:** I2C veri pinidir ve kontrol pinidir.
- OE (Output Enable):** Çıkış etkin. Tüm çıkışları hızlı bir şekilde devre dışı bırakmak için kullanılabilir. Bu pin düşük olduğunda tüm pinler etkinleştirilir. Pin yüksek olduğunda

çıkışlar devre dışı bırakılır. Varsayılan olarak düşük seçilmiştir, bu yüzden istege bağlı bir pindir.

- Cıktı Portları:** 16 çıkış portu vardır. Her bağlantı noktasının 3 pini vardır: V +, GND ve PWM çıkışları. Her PWM tamamen bağımsız olarak çalışır, ancak hepsinin aynı PWM frekansına sahip olması gereklidir.

4.5 SparkFun IMU



Şekil 36 LSM9DS1 IMU

IMU (Inertial Measurement Unit), sensörden algılanan açısal hız ve doğrusal ivme verisini tek bir modül ile toplayarak işlenmesini sağlayan elektronik birimdir. IMU'nun görevi aracın hızını, dönme açısını ve yer değiştirme miktarlarını belirlemektedir. IMU içerisinde ivme ölçer, jiroskop ve bazlarında manyetik alan ölçer bulunur.

4.6 Ublox Neo 6 – M GPS Modülü



Şekil 37 GPS Modülü ve Anten Birimi

Mobil robotlar, insansız hava araçları ve otonom araçlarda ortak olan lokalizasyon problemini çözmek için sıkılıkla kullanılan GPS modülü, dünya yörüngeindeki 24 uydudan en az üçüne bağlanarak enlem boylam ve yükseklik bilgisi verir. Bu projede iki boyutlu düzlemede koordinat bilgisi elde etmek için GPS modülünden faydalanaacaktır.

5. TASARIM VE SİMÜLASYON

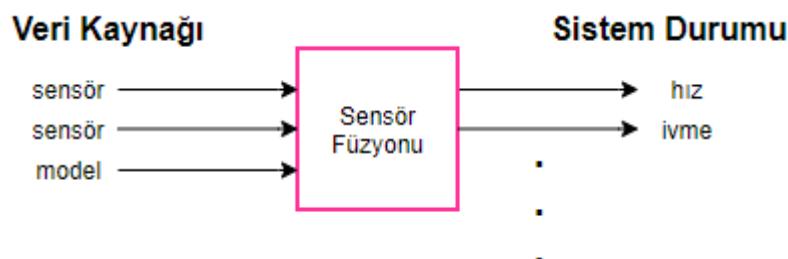
2. bölümde belirtilen gereksinimleri karşılayacak ekipmanlar seçildikten sonra, proje kapsamında verilen görevleri yerine getirecek sistemlerin tasarımları bu bölümde detaylı olarak verilecektir. Tasarım yapılırken, yüksek doğruluk içeren hesaplamalar yapılması ile yazılımın hızlı bir şekilde çalışarak cevap vermesi arasında bir optimizasyon yapılmıştır. Buradaki temel görüş, aracın kendisine ve çevresine zarar vermeden hareket etmesini ve bu harekete dair yapacağı hesaplamaları kabul edilebilir hızlarda gerçekleştirmesidir.

5.1 Genişletilmiş Kalman Filtresi ile Lokalizasyon ve Sensör Füzyonu

Otonom bir araç için en temel gereksinimlerden biri, sensör verilerini kullanarak bir koordinat sistemine göre kendi konumunu belirlemesidir. Bu amaçla, GPS (Global Positioning System) ve IMU (Inertial Measurement Unit) sensörleri tarafından sağlanan veriler, genişletilmiş Kalman Filtresi (Extended Kalman Filter: EKF) ile füzyon edilir. Filtrenin performansı, uygun şekilde tanımlanmış bir tahmin algoritması ile elde edilen giriş ve ölçüm gürültü kovaryanslarının çevrimiçi ayarlanmasıyla geliştirilir. [9]

5.1.1 Sensör Füzyonu

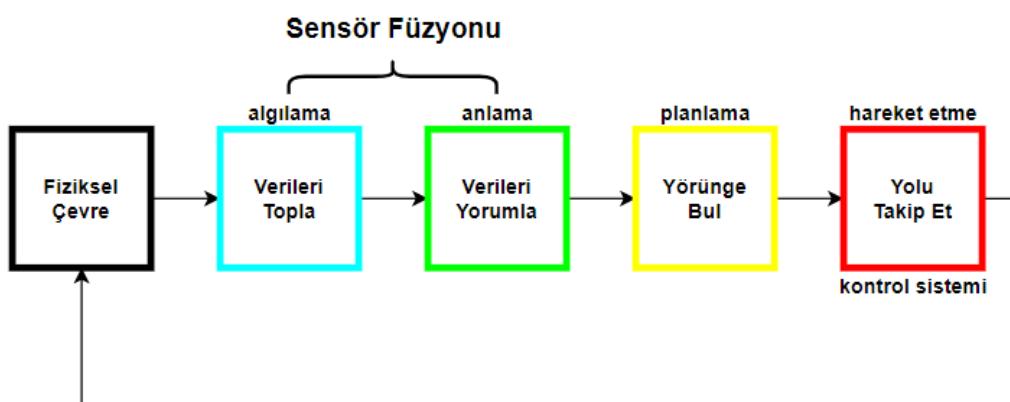
Sensör Füzyonu otonom sistemlerin tasarımının ayrılmaz bir parçasıdır. Otonom sürüs, radar takip istasyonları ve nesnelerin interneti (IoT) gibi teknolojilerin hepsi bir sensör füzyonuna dayanır. Temelde sensör füzyonunun amacı, iki veya daha fazla veri kaynağını birleştirerek fiziksel çevreyi “daha iyi” anlamaktır. Burada “**daha iyi**”, çözümün zaman içinde daha tutarlı ve tek bir veri kaynağından daha güvenilir olmasını ifade eder. Çoğunlukla verilerin sensörlerden geldiğini düşünebiliriz ve yaptıkları ölçüm, sistemin anlaşılmasını sağlar. Örneğin ivme veya bir nesneye olan uzaklık gibi verileri sağlarlar. Ancak, matematiksel bir model de bir veri kaynağı olabilir.



Şekil 38 Sensör Füzyonu I/O

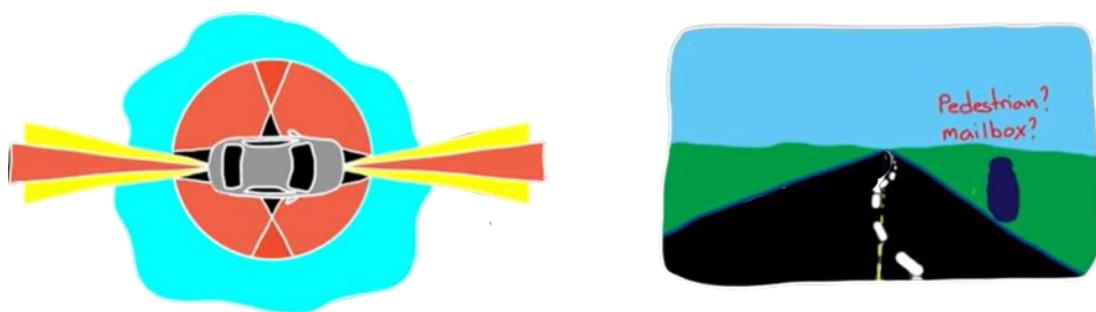
Otonom sistemlerin çevrelerindeki dünyaya etkileşime girebilmesi gereklidir ve bunda başarılı olabilmek için sistemin sahip olması gereken bazı yetenekler vardır. Bunları dört ana alana ayıralım: algılama, anlama, planlama ve hareket etme.

Algılama; çevreyi, sistemden ve dış dünyadan sensörler ile topladığı bilgiler ile doğrudan ölçmeyi ifade eder. Otonom sürüş yeteneğine sahip bir araba için, bu sensör paketi radar, lidar, kameralar ve daha fazlasını içerebilir. Ancak sadece sensörlerle veri toplamak yeterli değildir. Çünkü sistemin verileri yorumlayabilmesi ve otonom sistem tarafından anlaşılabilecek ve üzerinde çalışılabilecek bir bilgiye dönüştürebilmesi gereklidir.



Şekil 39 Otonom Araç Blok Diyagramı

Anlama adının rolü, algılanan verilerin anlamlandırılmasıdır. Aşağıdaki görselde temsil edilen otonom aracın, kamera sensöründen alınan verinin *Şekil 13*'deki gibi olduğunu varsayıyalım. Araç görseldeki piksel bloğunu, şerit çizgileri olan bir yol olarak yorumlamak zorundadır. Yolun kenarında görünen mavi leke, caddeden karşıya geçecek bir yaya veya bir posta kutusu olabilir. Sistemin daha sonra ne yapacağını belirlemesi için bu anlayış düzeyi kritiktir.



Şekil 40 Örnek Senaryo

Planlama adımı, otonom aracın ne yapmak istedığını anladığı ve hedefe ulaşmak için bir yörünge oluşturduğu adımdır.

Son olarak sistem, aracın bu yolu izlemesini sağlayan en uygun eylemleri hesaplar. Bu son adım, kontrolcünün gerçekleştirdiği **hareket etme** adımdır.

Sensör füzyonunun, otonom sistemlere sağladığı katkıları üç başlıkta toplayabiliriz:

- Verinin kalitesini artıtabilir
- Güvenilirliği artıtabilir
- Ölçülmemiş durumları tahmin edebilir.

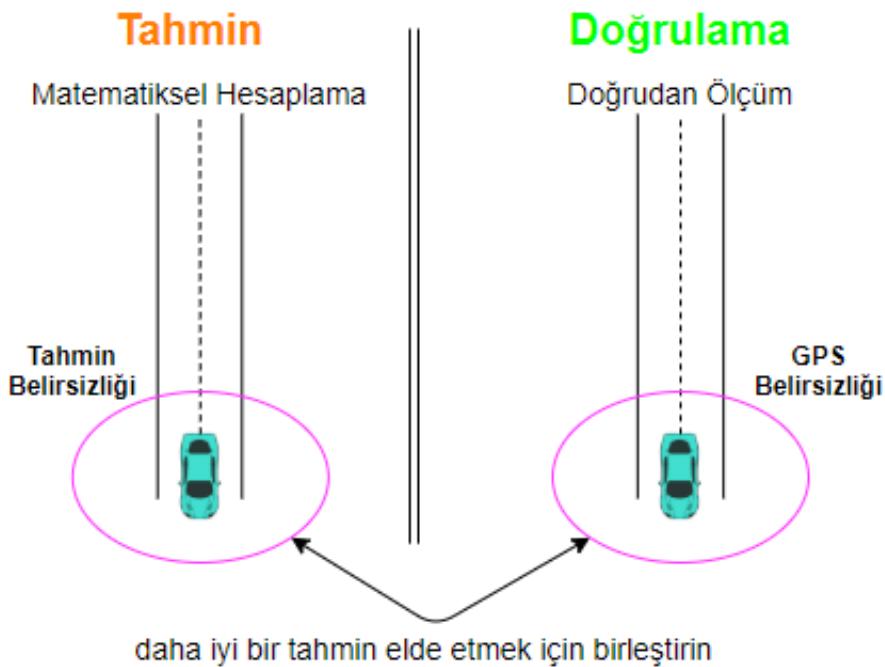
5.1.2 Sensör Füzyonu ile Lokalizasyon (GPS + IMU)

Dünyanın yüzeyine göre bir nesnenin konumunu bilmek için sadece GPS kullanmak yeterlidir. GPS enlem, boylam ve yükseklik bilgilerini vererek bir nesnenin 3 boyutlu düzlemede konum bilgisini verir. GPS'in üzerinde bulunduğu sistem (araç) nispeten yavaş yön değiştirdiği ve hızlandığı durumlarda ve birkaç metreye kadar konum doğruluğuna ihtiyaç duyulduğunda, GPS tatmin edici bir sonuç verir. Ancak sistem hızla yön değiştirdiğinde ve konum bilgisinin doğruluğundaki sapmaya ancak 30 cm'e kadar izin verilen durumlarda GPS tek başına yetersiz kalmaktadır. Bu nedenle GPS sensörü, IMU sensörü ile genişletilmiş kalman filtresinden geçirilerek konum bilgisinin doğruluğundaki sapma en aza indirilir.

Kalman Filtresi, temelde iki adımlı bir işlemden oluşur: tahmin etme (**predict**) ve doğrulama (**correct**). Tasarlanan sistemin, aracın nerede olduğunu veya ne kadar hızlı gittiğini tahmin etmesini istiyorsak, bunu yapmanın genelde iki yolu vardır. Doğrudan bir sensör yardımı ile ölçülebilir veya dinamik ve kinematik bilgilerini kullanarak bir tahminde bulunabilir.

Yolda giden ego aracının nerede olduğunu tahmin etmeye çalıştığımız **şekildeki** senaryoyu düşünelim. Birinci yol GPS'i kullanarak konum bilgisini almak olacaktır. Ancak, aracın başlangıç konumunu ve ortalama hızını biliyorsak belirli bir süre sonra aracın nerede olacağını bilebiliriz. Bu da ikinci tahmin yoludur. Bu iki yolu bir arada kullanmak bize doğruluğu daha yüksek bir tahmin sağlayabilir.

KALMAN FİLTRESİ



Şekil 41 Kalman Filtresi

Otonom araç, çarpışmalardan kaçınmak ve optimum manevralar ile yolu takip edebilmek için sürekli olarak kendi konumunu hesaplamalıdır. Durgun halde iken GPS verisinden aldığı ilk konum bilgisi ile matematiksel olarak konumunu hesaplayabilir. Ancak bu matematiksel model çevre şartlarındaki bozucu etkenlere karşı duyarsızdır. (Teker kayması, rüzgâr etkisi, eğim vs.) Bu nedenle GPS verisinden belirli periyotlar ile hesapladığı konumu karşılaştırarak daha doğru bir tahminde bulunması gerekebilir.

Şehir içi kullanımda GPS sinyali binalardan ve yol kenarında bulunan engellerden geçemez ve GPS' den alınan konum bilgisinde birkaç metreye kadar sapma görülebilir. Güvenlik açısından aracın yol üzerindeki diğer dinamik nesnelere karşı konumunu hassasiyetle bilmesi ve bu bilgiyi güncellemesi hayatı önem taşır. Bu nedenle matematiksel model ve GPS verisinin yanı sıra IMU sensöründen alınan yönelik verileri ile yapılan konum tahminindeki hata miktarı santimetre seviyelerine indirilebilir.

5.1.3 Genişletilmiş Kalman Filtresi ile Lokalizasyon

Yukarıda sözü geçen bilgiler ışığında genişletilmiş bir Kalman (EKF)滤器 ile sensör füzyonu yapılarak bir lokalizasyon probleminin çözümü gösterilecektir. Simülasyona geçmeden önce kısaca filtre tasarımına, hareket modeline ve gözlem modeline degeinilecektir. Ardından genişletilmiş kalman滤器 bu modeller ile tasarlanarak simülasyon yorumlanacaktır.

Filtre Tasarımı

Bu simülasyonda, ego aracı, t zamanında 4 durum içeren bir durum vektörüne sahiptir.

$$\mathbf{X}_t = [x_t, y_t, \emptyset_t, v_t]$$

x, y pozisyon; **Φ** oryantasyon ve **v** hızı temsil etmektedir.

X_{est} durum vektörü, **P_t** kovaryans matrisi, **Q** proses gürültüsünün kovaryans matrisi ve **R**, t zamanında gözlem gürültüsünün kovaryans matrisidir

Araçta bir hız sensörü ve cayro sensörü bulunur. Böylece, giriş vektörü her zaman adımda aşağıdaki denklemdeki gibi kullanılabilir:

$$\mathbf{U}_t = [v_t, \emptyset_t]$$

Ayrıca, aracın bir GPS sensörü vardır, bu da aracın her seferinde x-y konumunu gözlemleyebileceği anlamına gelir.

$$\mathbf{Z}_t = [x_t, y_t]$$

Giriş ve gözlem vektörü sensör gürültüsünü içerir. Kodun içerisinde, "observation" fonksiyonu giriş ve gözlem vektörünü üretir.

Hareket Modeli

Araç modeli $\dot{\emptyset} = \omega$ olmak üzere hareket modeli aşağıdaki gibi elde edilir.

$$\mathbf{x}_{t+1} = F\mathbf{x}_t + B\mathbf{u}_t$$

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} \cos(\phi)dt & 0 \\ \sin(\phi)dt & 0 \\ 0 & dt \\ 1 & 0 \end{bmatrix},$$

$$J_F = \begin{bmatrix} \frac{dx}{dx} & \frac{dx}{dy} & \frac{dx}{d\phi} & \frac{dx}{dv} \\ \frac{dy}{dx} & \frac{dy}{dy} & \frac{dy}{d\phi} & \frac{dy}{dv} \\ \frac{d\phi}{dx} & \frac{d\phi}{dy} & \frac{d\phi}{d\phi} & \frac{d\phi}{dv} \\ \frac{dv}{dx} & \frac{dv}{dy} & \frac{dv}{d\phi} & \frac{dv}{dv} \\ \frac{dv}{dx} & \frac{dv}{dy} & \frac{dv}{d\phi} & \frac{dv}{dv} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -v\sin(\phi)dt & \cos(\phi)dt \\ 0 & 1 & v\cos(\phi)dt & \sin(\phi)dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ Jacobian matris}$$

Gözlem Modeli

Araç GPS'ten x-y konum bilgisini alabilir. O halde GPS gözlem modeli $\mathbf{z}_t = H\mathbf{x}_t$ olur.

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad J_F = \begin{bmatrix} \frac{dx}{dx} & \frac{dx}{dy} & \frac{dx}{d\phi} & \frac{dx}{dv} \\ \frac{dy}{dx} & \frac{dy}{dy} & \frac{dy}{d\phi} & \frac{dy}{dv} \\ \frac{d\phi}{dx} & \frac{d\phi}{dy} & \frac{d\phi}{d\phi} & \frac{d\phi}{dv} \\ \frac{dv}{dx} & \frac{dv}{dy} & \frac{dv}{d\phi} & \frac{dv}{dv} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Genişletilmiş Kalman Filtresi (EKF)

Genişletilmiş Kalman Filtresi ile lokalizasyon işlemi:

=====Tahmin=====

$$\mathbf{x}_{tahmin} = F\mathbf{x}_t + B\mathbf{u}_t$$

$$\mathbf{P}_{tahmin} = J_F \mathbf{P}_t J_F^T + Q$$

=====Güncelleme=====

$$\mathbf{z}_{tahmin} = H\mathbf{x}_{tahmin}$$

$$\mathbf{y} = \mathbf{y} - \mathbf{y}_{tahmin}$$

$$S = J_H \mathbf{P}_{tahmin} J_H^T + R$$

$$K = \mathbf{P}_{tahmin} J_H^T S^{-1}$$

Sonuç olarak

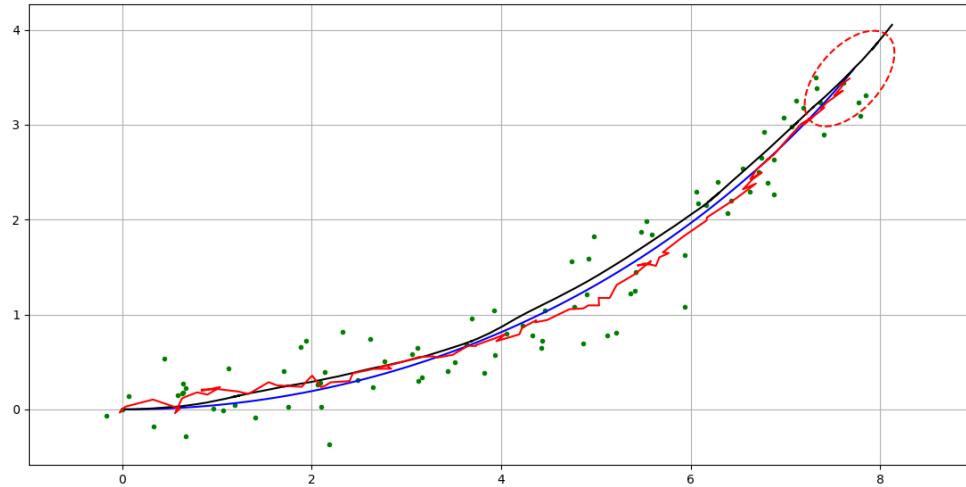
$$\mathbf{x}_{t+1} = \mathbf{x}_{tahmin} \mathbf{P}_t + K\mathbf{y}$$

$$\mathbf{P}_{t+1} = (\mathbf{I} - KJ_H) \mathbf{P}_{tahmin}$$

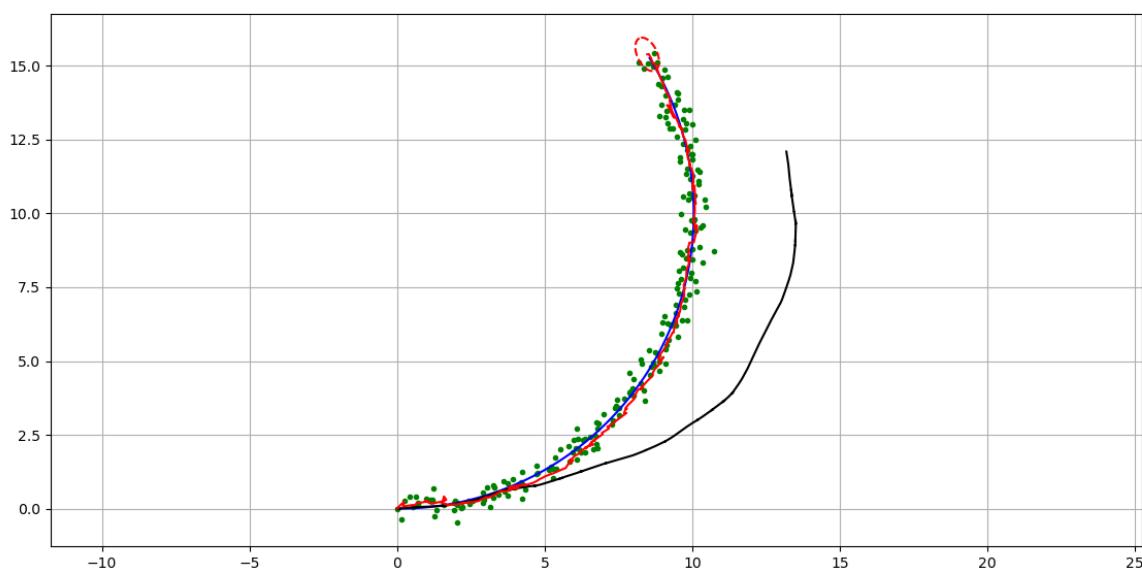
denklemleri elde edilir.

5.1.4 Simülasyon

Yukarıdaki denklemelere göre Genişletilmiş Kalman filtresini tasarlayarak dairesel bir yörunge üzerinde hareket etmesi beklenen bir aracı simüle edebiliriz.

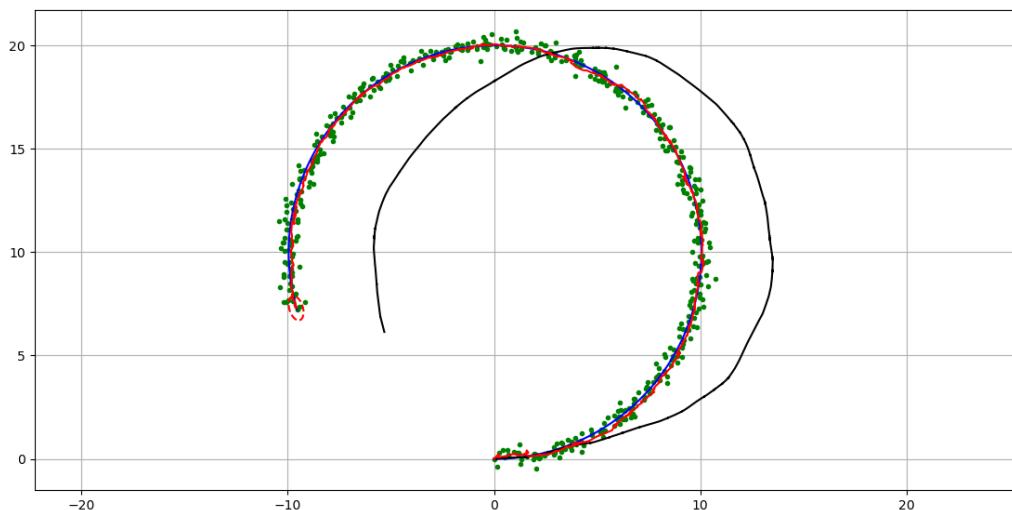


Bu simülasyonda **mavi** çizgi gerçek yörüngeyi gösterirken, **siyah** çizgi matematiksel modelden gelen hatalı yörungedir. **Yeşil** noktalar sensörlerden (GPS, IMU) gelen konum bilgisi olurken, **kırmızı** çizgi Genişletilmiş Kalman Filtresi ile tahmin edilen yöringedir.



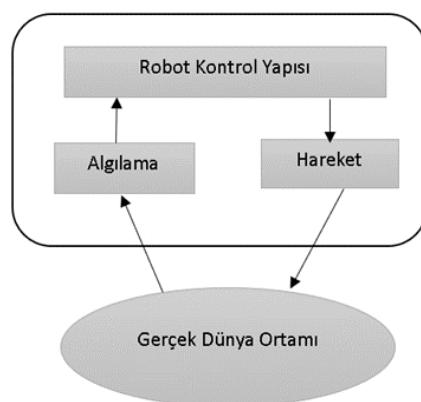
Şekil 43 Matematiksel Modelden Gelen Hata (siyah çizgi)

- Bu simülasyon sonucunda, sadece matematiksel modelden yola çıkararak elde edilen geri beslemesiz konum tahmini, aracı gerçek yörüngeye uzaklaştırdığını görmekteyiz.
- Ayrıca yalnızca GPS verisine bakılarak hareket edildiği takdirde referans olarak verilen dairesel yörüngeyi takip etmekte aracın zorlandığını ve kusurlu bir yörünge çizdiğini görebiliriz.
- Genişletilmiş Kalman Filtresi ile sensör füzyonu simülasyonu sonucunda teknik gereksinimler kısmında belirtilen maksimum 15 cm kabul edilebilir yörünge hatasının altında bir değer elde edilmiştir.



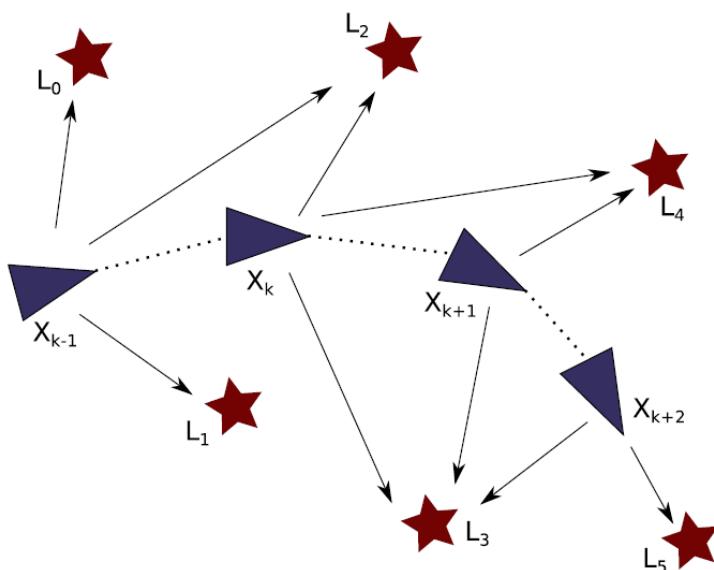
5.2 SLAM (Simultaneously Localization and Mapping) ile Çevre Haritalama

Otonom araçlar fiziksel ortamlarda hareket edebilmek için çevresinden topladıkları verileri içinde bulunan hareket sistemleri, karar mekanizmaları, ortamı algılamak amacıyla algılayıcı sistemleri ve algoritmalar sayesinde yorumlayıp kendi kendine hareket eden mekanizmalardır.



Şekil 45 SLAM iş akış diyagramı

Eş Zamanlı Lokalizasyon ve Haritalama aynı zamanda Eş Zamanlı Haritalama ve Yerelleştirme (ConCurrent Mapping and Localization) olarak da bilinir ve robotun bir haritadaki yerini belirlerken bir çevre haritası oluşturma zorunluluğunu inceler. SLAM algoritması bir robotun, lazer tarama verisi ile çevre haritası oluşturabileceği ve aynı zamanda atadığı nokta bulutları ile anlık olarak konumunu belirleyebileceği bir süreçtir. Başlangıçta aracın konumu ve çevrenin haritası belli değildir, kinematik modeli belirlenen araç hareket etmeye başlar. Bu süreçte hem aracın hem de yer işaretlerinin konumu anlık olarak tahmin edilmelidir. Bu eylemi gerçekleştirmek için araç, yer işaretleri ile aracın arasındaki konumun ölçülerini alabilecek bir duyusal sistemle ile donatılmalıdır. Bu sistemin görevini aracımızda Lidar sensörü görecektir.

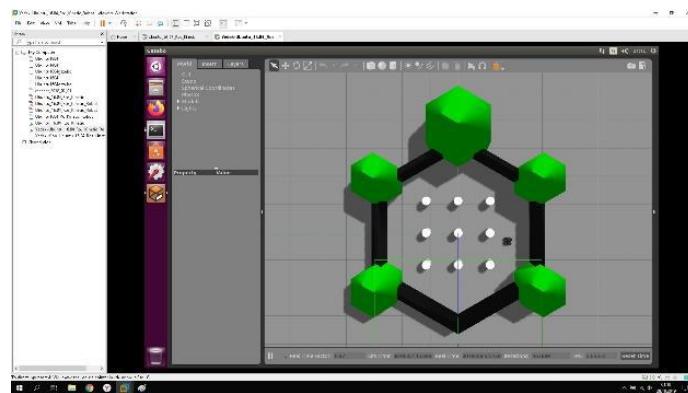


Şekil 46 Temel SLAM Problem Tanımı

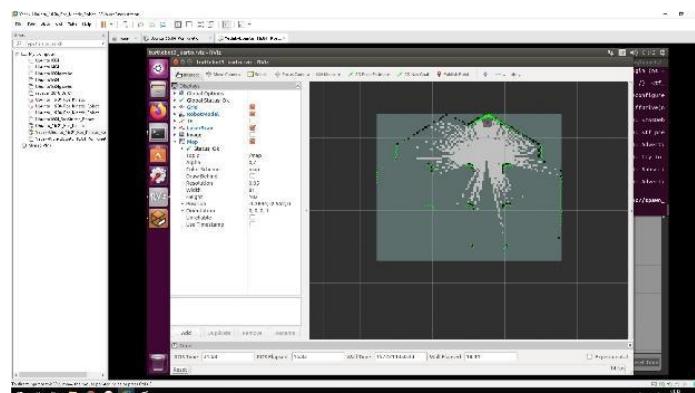
Yukarıdaki şekilde mavi üçgen hareket eden nesneyi temsil ederken, yerel konumu olan X_k 'yı hesaplarken L_n statik yer işaretleri (ve bir önceki yer işaretleri olan L_{n-1}) bilgisini kullanır.

Aşağıda ki SLAM simülasyonu Ubuntu 16.04 işletim sistemine kurulan ROS kinetic paketi ile RVIZ ortamında gerçekleştirılmıştır. Bu simülasyon için Construct Sim tarafından oluşturulan Gazebo sanal haritası kullanılmıştır. RVIZ ortamında bu işlem eş zamanlı olarak izlenebilmektedir.

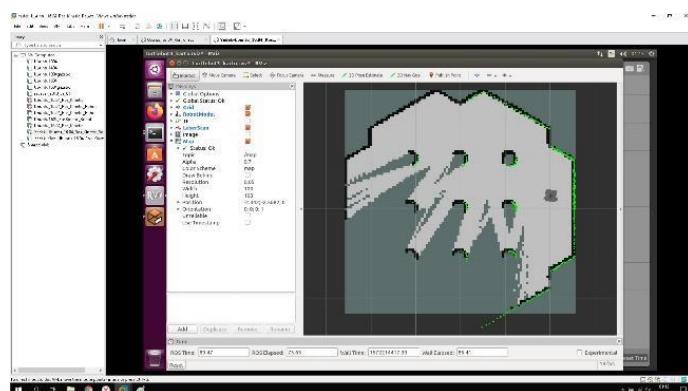
SLAM işlem aşamaları şekillerde adım adım verilmektedir.



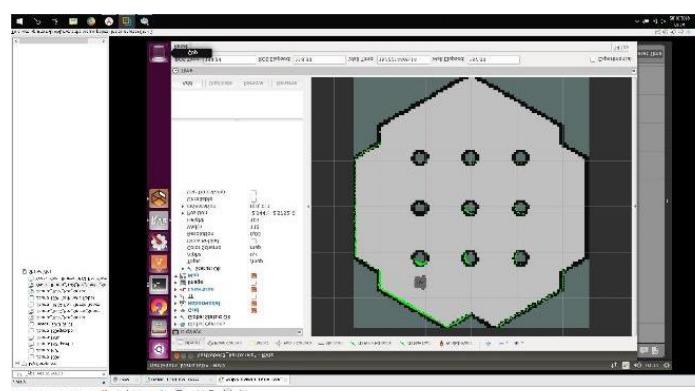
Şekil 47 Gazebo Ortamı ve Aracın İlk Konumu



Şekil 48 RVIZ Ortamında Aracın İlk Konumu



Şekil 49 Aracın Haritalamaya Devam Etmesi

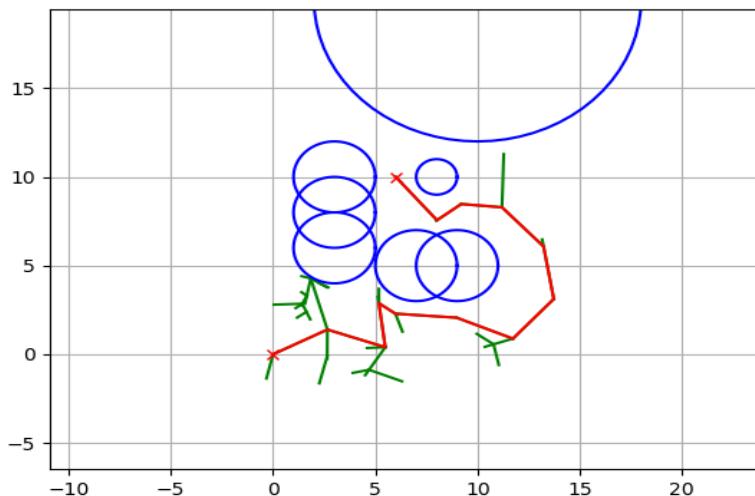


Şekil 50 RVIZ ortamında aracın haritalamayı bitirmesi

5.3 RRT Algoritması ile Yörünge Planlama

Navigasyon sorununu ele alan çok sayıda yol planlama algoritması vardır. Hızla keşfedilen rastgele ağaçlar (RRT), hem bir grafik oluşturan hem de bir yol bulan yaygın bir seçenektedir. Yol mutlaka ideal olmayacağından, Dr. Karaman ve Dr. Frazzoli tarafından popüler hale getirilen RRT *, [10] mesafe veya diğer metriklere göre en kısa yolu elde etmeyi amaçlayan optimize edilmiş bir modifiye algoritmadır. Her ikisi de python içinde uygulanır ve bu makalede gözlemlenmiştir. Fikri göstermek için, algoritmalar 2B alanda sınırlarla uygulanacaktır. Bununla birlikte, her iki algoritma da herhangi bir gerçek sürekli boyutlu uzaya yerleştirilebilir.

Python ortamında, RRT algoritmasını test etmek için aşağıdaki Simülasyon hazırlanmıştır. Bu simülasyonda harita büyülü, engellerin şekli ve konumu, önceden belirlenmiştir. Algoritma defalarca çalıştırılarak bulunan yollar gözlemlenmiştir. RRT 'nin doğası gereği zaman zaman uzun yollar planlarken genel bakıldığında en kısa yolu hızlıca tespit etmektedir.



Şekil 51 Python ortamında RRT Algoritması Simülasyonu

- **Mavi** daireler tespit edilen engelleri temsil etmektedir.
- **Yeşil** renkli ağaç dalları algoritma tarafından uygun yörüngeyi arama sırasında oluşturulan alternatif yörüngelerdir.
- **Kırmızı** çizgi iki nokta arasındaki olası yörüngeyi temsil etmektedir.
- **Kırmızı** renkli (0,0) koordinatındaki **x** noktası başlangıç noktasını; (6,10) koordinatındaki diğer **x** noktası ise hedef noktayı temsil etmektedir.

RRT'nin mantığı aslında oldukça basittir. Noktalar rastgele oluşturulur ve mevcut en yakın düğüme bağlanır. Bir tepe noktası her oluşturulduğunda, tepe noktasının bir engelin dışında olup olmadığını kontrol etmek gereklidir. Ayrıca, tepe noktasını en yakın komşusuna zincirleyerek engellerden kaçınmalıdır. Hedef bölge içinde bir düğüm oluşturulduğunda veya bir sınıra ulaşıldığında algoritma sona erer.

Rastgele bir pozisyon belirleme yöntemi bir tasarım kararıdır. Yerleşik rasgele sayı üreteçleri gibi basit yöntemler kullanılabilir. Temel uygulamalar için bu tür yaklaşım yeterlidir. Ancak, geliştiriciler, rastgele sayı üreteçlerinin gerçekten rastgele olmadığını ve bir dereceye kadar önyargı (bias) içerdigini farkındadır.

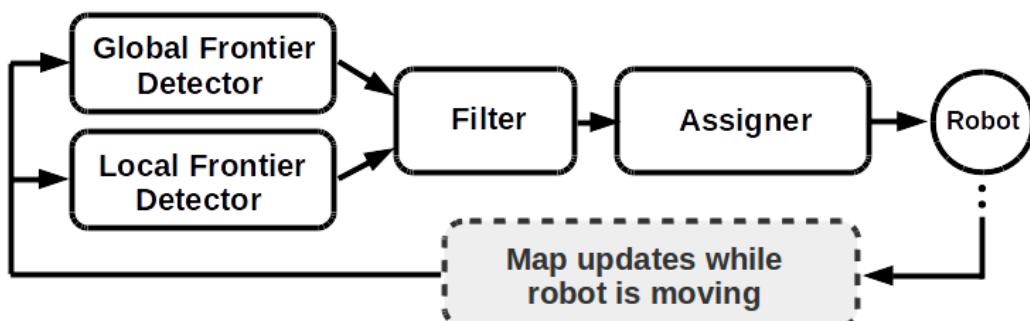
Ek olarak, rastgele üretilen tepe noktasını zincirleme yöntemi özelleştirilebilir. Bir yöntem, yeni tepe noktası ile en yakın kenar arasındaki en kısa mesafeyi oluşturan vektörün hesaplanması içerir. Kavşak noktasında, kenara yeni bir düğüm eklenir ve rastgele oluşturulan tepe noktasına bağlanır. Alternatif olarak, tepe noktası, kendisine ayrık düğümlerin bir bağlantısını zincirleyerek en yakın düğüme bağlanabilir. Bu yöntem daha az hesaplama gerektirir ve uygulanması daha kolaydır, ancak daha fazla noktanın depolanmasını gerektirir.

Algoritmanın avantajı, hızı ve uygulama kolaylığıdır. Diğer yol planlama algoritmaları ile karşılaştırıldığında, RRT oldukça hızlıdır. Algoritmanın en fazla işlem gerektiren (cost) kısmı, en yakın komşusunu bulmaktadır. Çünkü bu süreç üretilen köşe sayısına bağlı olarak büyür.

5.3.1 RRT_EXP ROS Uygulaması

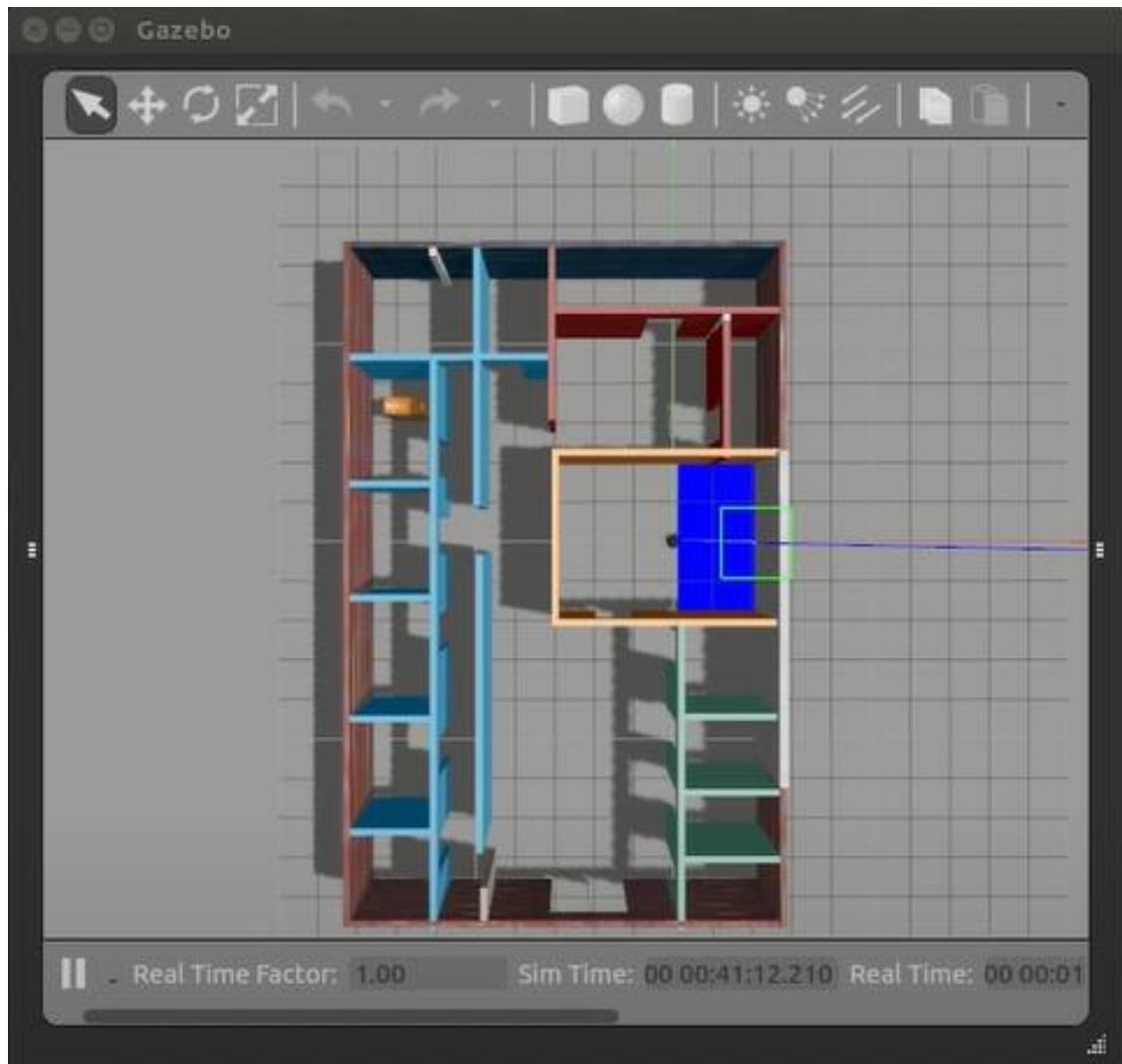
RRT_Exp algoritması açık kaynak kodlu olup, farklı robotlara entegre edilebilmektedir. Algoritma, araç başlangıç konumundaki iken, araca taranacak alanın sınırları girilir ve bir hedef noktası tanımlanır. Araç Lidar ile çevre haritalandırması yaparak hedef noktası için ağaç dalları şeklinde arama yapar. En çok noktanın kesiştiği noktaları birleştirerek bir yörüngelere planlaması belirler. Bu yörungede ilerlerken kendi konumunu ve haritayı sürekli güncelliyor.

Böylelikle yörungede ilerledikçe görüş alanını değiştirir ve yeni alan taraması yapar. Bu işlem hedef noktasına varincaya kadar tekrar eder. Algoritma düğüm akış diyagramı aşağıdaki gibidir.



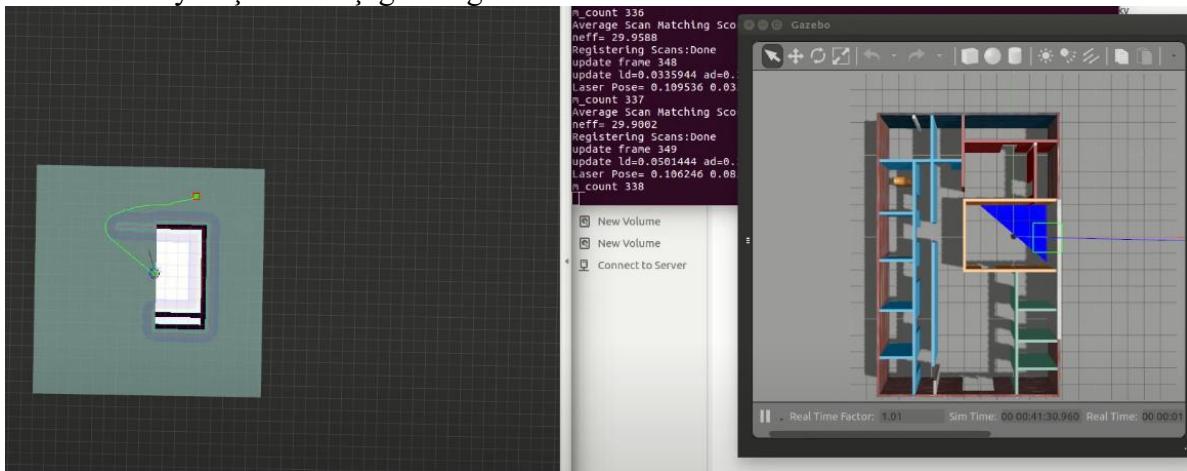
Şekil 52 RRT algoritması düğüm akış diyagramı [11]

RRT algoritması simülasyonunu yapmak için örnek bir çevre oluşturulmuştur. Taranacak çevre Gazebo üzerinden modellenmiştir. Çevre tasarıımı için Yıldız Teknik Üniversitesi Mekatronik Mühendisliği binası benzetim olarak modellenmiştir. Model RRT algoritması çalışma alanına tanımlanmış olup simülasyon bu model üzerinden yapılmıştır.

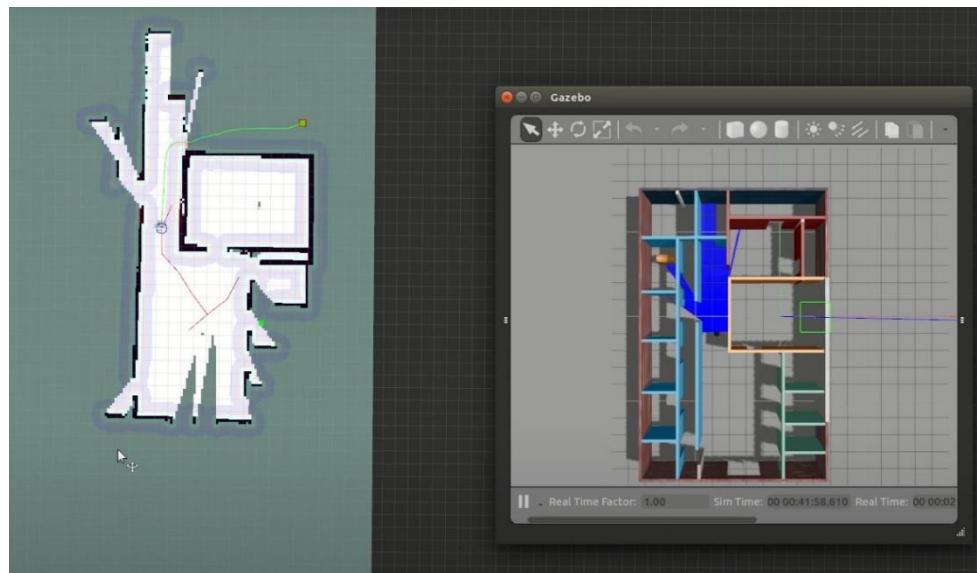


Şekil 53Mekatronik mühendisliği bölüm binası benzetimi

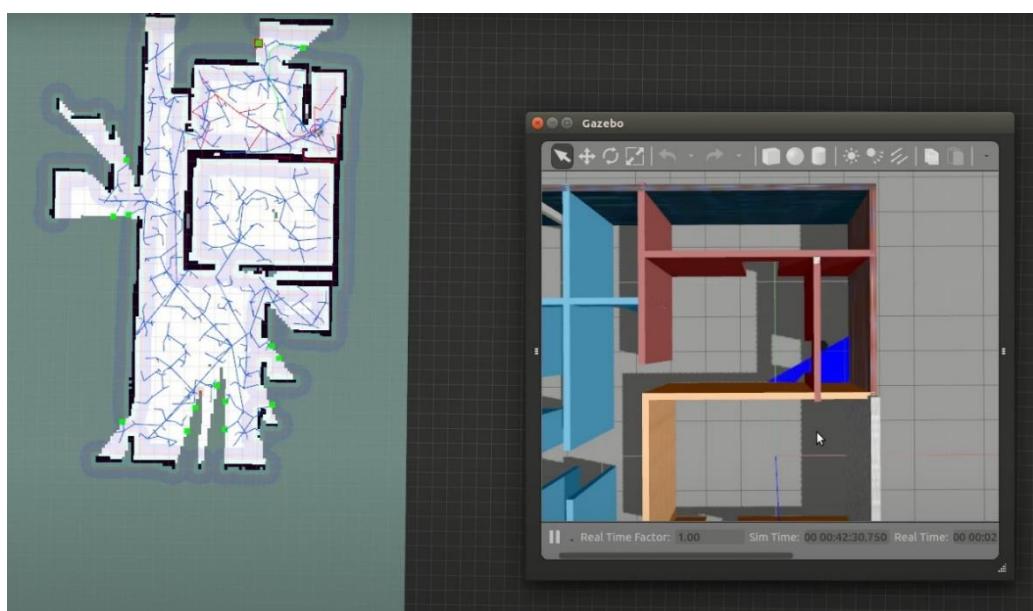
Simülasyon çıktıları aşağıdaki gibidir.



Şekil 54 Başlangıç noktasından sınır ve bitiş noktalarının belirlenmesi



Şekil 55 Bitiş noktası için en kısa yolu bulunması ve haritanın anlık olarak güncellenmesi



Şekil 56 Bitiş noktasına varılması

5.5 OpenCV ve YOLO ile Nesne Tanıma

5.5.1 OpenCV ile Nesne Tanıma

Mobil robotun çevresinin algılaması ve buna uygun olarak yörunge planlaması yapabilmesi için simülasyon ortamında OpenCV ile nesne tespiti çalışması yapılmıştır. Proje isterlerinde yapay zeka ile nesne tespiti yapılcagından bahsetmişlik ancak Covid-19 salgını nedeniyle proje fiziksel uygulamasından vazgeçilip simülasyon ortamında görevlerin yapılmasına karar verilmiştir. Simülasyon ortamında kullanılan robotun üzerinde Asus Tinker Board kartı kullanılmaktadır ve YOLOv3 alogritmasını çalıştırabilecek işlemci gücüne sahip değildir. Bunun yanı sıra yapay zeka algoritmalarını çalıştırabilmek için bir takım hızlandırıcı yazılımlar ve paralel işleme yapılabilmesi için kütüphaneler mevcuttur. Asus Tinker Board'da grafik işlemcisinin zayıf kalması ve CUDA kütüphane desteğinin bulunmamasından dolayı nesne tanımlama işlemi OpenCV kütüphanesi üzerinden yapılacaktır.

ROS üzerinde yapılan OpenCv simülasyonunda uygulama iki parçadan oluşmaktadır.

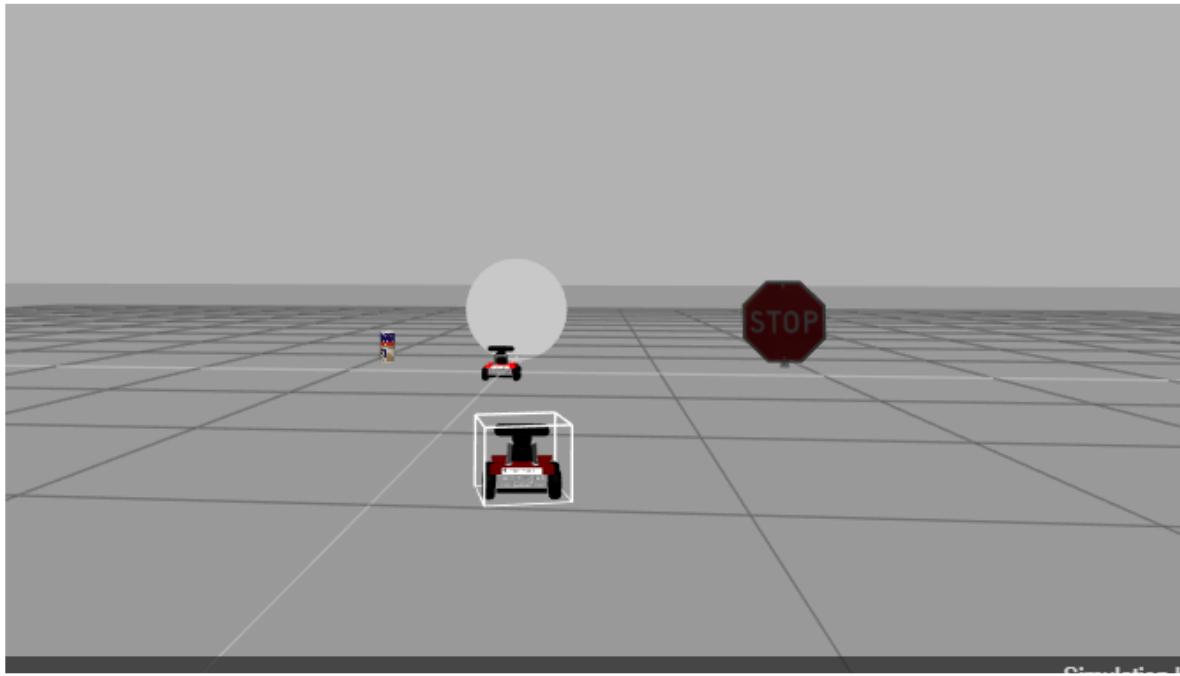
Bunlar;

- Nesneleri Öğretmek ve Test Etmek
- Tanımlanan Nesnelere Göre Karar Vermektir

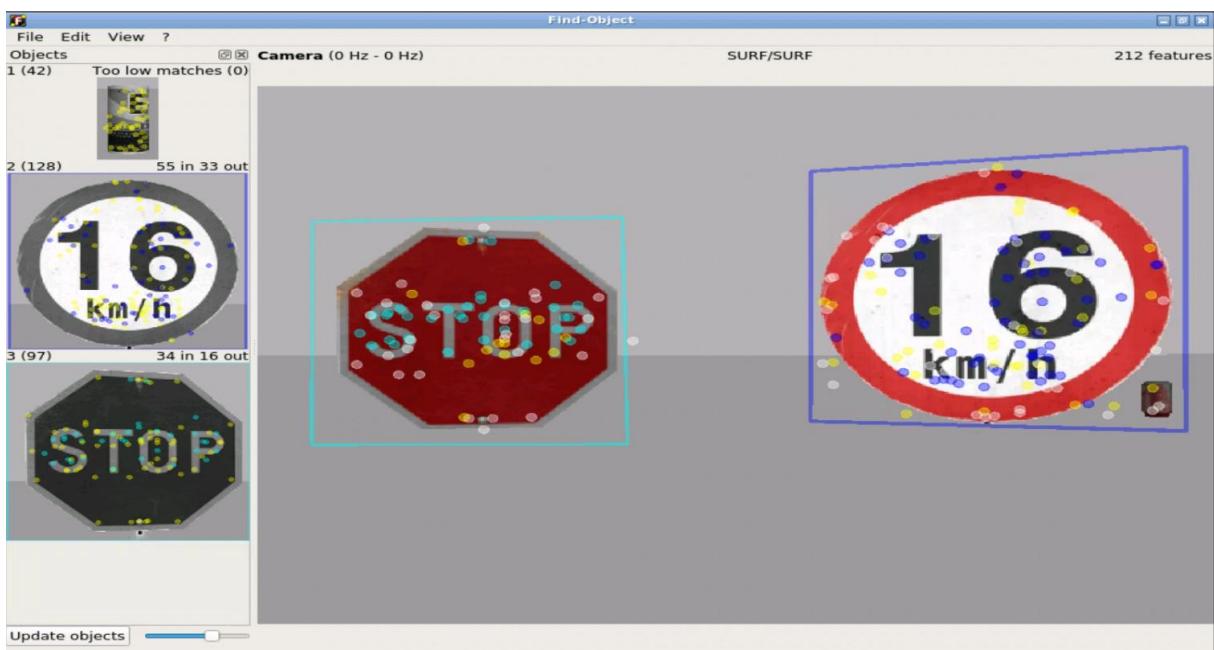
Nesneleri tanıma işlemi; çizgiler, kenar renkleri ve bunların görelî konumları gibi görüntü özelliklerine dayanır.

Nesneleri Öğretmek ve Test Etmek İşlemi adımları aşağıdaki gibidir.

- İlk olarak ROS üzerindeki öğretme *launch* dosyaları çalıştırılır ve *Gazebo* ve *Rviz* ortamında Rosbot aracı çalıştırılır.
- Kameradan alınan görüntü *Rviz* ortamında görüntülenir.
- Aracın görüş mesafesinden olacak şekilde aracın önüne tanıtılmazı istenilen nesneler eklenir.
- Tanıtılmak istenilen nesneler *Find-Object* grafik arayüzünden kamera görüşünde ortalanır.
- Objeler kutu içerisinde olacak şekilde etiketlenir ve bir class olarak tanımlanır
- Etkitelinen nesne OpenCV verisetindeki nesne ile eşleştirilir.
- Bu işlem eklenmek istenen obje sayısı kadar tekrarlanır.
- ID numarası ile nesneler tanımlanır ve *launch* dosyaları kapatılır.



Şekil 57 Gazebo ortamında aracın görüş alanına nesnelerin eklenmesi



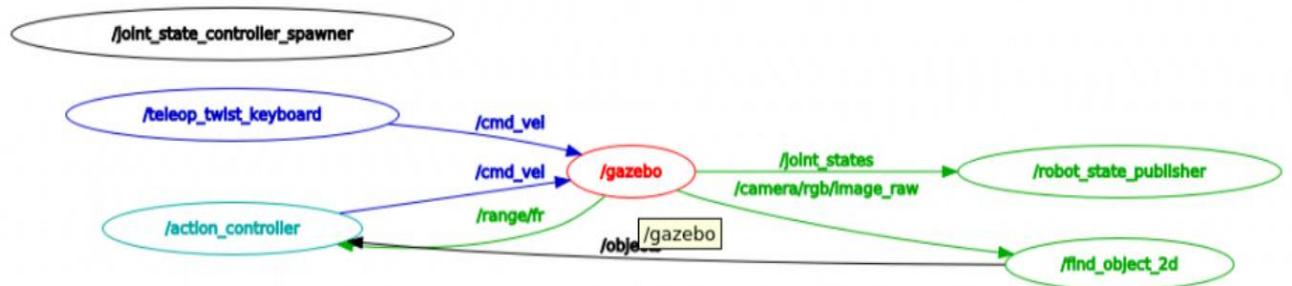
Şekil 58 Grafik arayüzü aracılığıyla nesnelerin etiketlenmesi

Nesneleri Test Etmek

- İlk kısımda öğreten nesneler launch dosyasındaki veritabanına tanımlanır.
- Tekrardan ikinci bir terminal açılarak test etme launch dosyaları çalıştırılır.
- *Gazebo* ve *Rviz* programları tekrardan çalıştırılır ve test ortamı hazırlanır.
- Test ortamında öğreten nesneler eklenir ve aracın görüş alanına giren nesneler ile test edilir.

- Tanımlamada zayıf kalan nesneler için öğretme ortamı tekrardan açılarak doğruluk oranını arttırlıncaya kadar tekrarlanır.

Bu işlem sonucunda tanımlanan nesneler */object topic* altında veritabanında tutulur ve en son kararlı sürümü baz alarak bir sonraki işlemde kullanılmak üzere hazırda tutulur.



Şekil 59: Nesne Tanımlama – Karar Verme Ros akış diyagramı

Tanımlanan Nesnelere Göre Karar Verme

Bu kısımda yukarıda tanımlanan nesnelere göre aracın hareket planlamasında karar verme süreci hakkında izlenen adımlar anlatılacaktır.

Tanımlanan nesnlere */object* topic altındaki nesneler mesaj yayımlamaya başlar. Yayınlanan mesajlar */cmd_vel* abonesine giriş olarak alınır. Aksiyon kontrolcüsünde, yukarıda tanımlanan nesnelere göre hareket planlaması kodlanır. Bunlar “Dur” tabelasını gördüğünde durma, “Azami Hız – Hız Arttır – Hız Azalt” tabelalarına göre hız belirleme, azaltma ve artırma işlemleri için kontrolcü tasarılanır. [12]

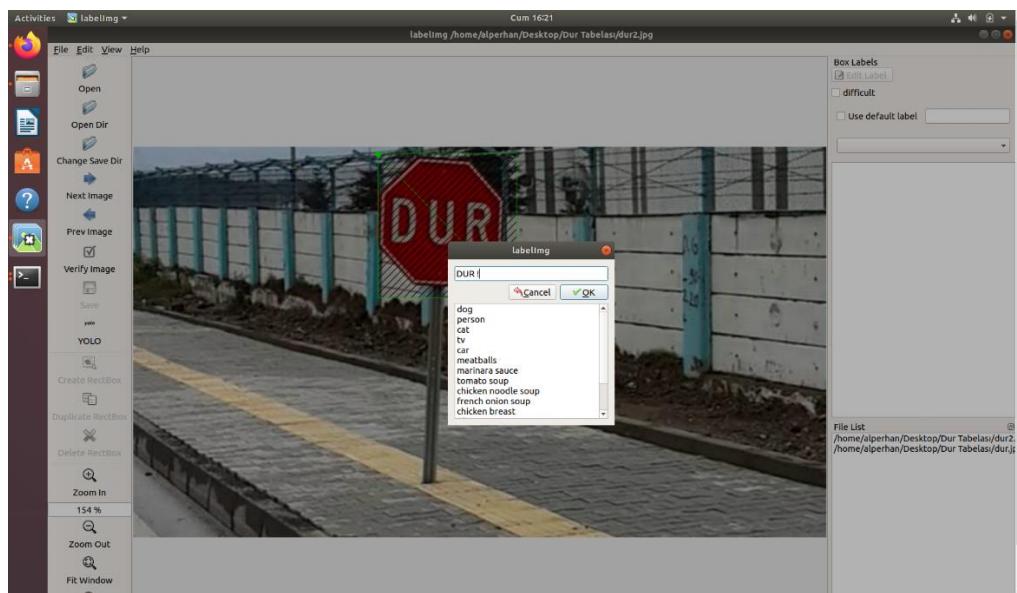
5.5.2 YOLOv3 ile Nesne Tanıma

Proje isterlerinde belirlenen YOLOv3 algoritması ile nesne tanımlama işlemi yapılmasına karar verilmiştir. Yolov3 yapay zeka tabanlı bir gerçek zamanlı nesne algılama dedektöridür. Bu kısımda YOLOv3 ile nesne tanımlama işleminden bahsedilecektir.

5.5.2.1 Veri Seti Nasıl Eğitilir?

YOLOv3 açık kaynak kodlu bir algoritmadır. Kendi veri setimizi eğiterek nesne tespiti yapmamıza izin vermektedir. Burada öğrenme transferi yöntemi (Transfer Learning) kullanılmaktadır. Nesne tanımlama işlemi adımları aşağıdaki gibidir;

- Açık kaynak kodlu Yapay Sinir Ağı temelli Darknet framework kurulur
- GPU kullanımına izin verilir, bu sayede CPU'ya kıyasla 500 kat daha hızlı çalışır.
- Tanıtılması istenilen nesneler için verisetleri oluşturulur.
- Veri setleri labelImg yazılımı yardımıyla etiketlenir.



Sekil 60 Verilerin etiketlenmesi

- Darknet içerisinde dönüştürme koduyla dosya içerisinde veriler tanımlanır.
- Dosya içerisinde test ve eğitim verileri ayrılır. Artık veriler eğitime hazırır.
- Ana dizine dönülüp konfigürasyon dosyası içerisinde girilir ve yeni konfigürasyonlar için *cfg* dosyası oluşturulur.
- Nesne sayısı *class* olarak tanımlanır ve sayısı girilir.
- Class sayısına göre filtre sayısı tanımlanır.
 - Yolov3 için filtre sayısı = (Class sayısı + 5) x 3 olması gerekmektedir.
- Darknet kendi ağırlık dosyası indirilir ve kendi konfigürasyon dosyamızın dizini tanıtılır.
- Darknet ağırlık dosyası altında veriler eğitime sokulur.

Bu işlem grafik işlemci gücüne göre süresi belirlenmektedir. Eğitimin daha hızlı sürmesi ve daha yüksek doğruluk oranı sağlamak ve tekrarlanan işlemleri hızlandırmak için çevrimiçi bir

geliştirme aracı olan Google Colab kullanılmıştır. [13]

Google Colab yüksek donanım gücüne sahiptir ve özellikleri aşağıdaki gibidir;

- GPU: Tesla K80i, 12GB GDDR5 VRAM, 2496 CUDA Çekirdeği
- CPU: Single Core Hyper Threaded Xeon Processors @2.3Ghz (1 core, 2 threads)
- RAM: ~12.6 GB
- Depolama: 33 GB belleğe sahiptir.

Etiketlenmiş verisetleri Googl Drive'da bir klasöre atılır. Bu veriler Google Colab'a giriş olarak verilir. Eğitim yaklaşık 3 Mbps indirme hızına sahip internet bağlantısı altında yaklaşık bir saat sürmüştür. Çıkış parametreleri aşağıdaki gibidir.

```

85 upsample           2x    13 x 13 x 256 -> 26 x 26 x 256
86 route  85 61
87 conv   256      1 x 1/ 1    26 x 26 x 768 -> 26 x 26 x 256 0.266 BF
88 conv   512      3 x 3/ 1    26 x 26 x 256 -> 26 x 26 x 512 1.595 BF
89 conv   256      1 x 1/ 1    26 x 26 x 512 -> 26 x 26 x 256 0.177 BF
90 conv   512      3 x 3/ 1    26 x 26 x 256 -> 26 x 26 x 512 1.595 BF
91 conv   256      1 x 1/ 1    26 x 26 x 512 -> 26 x 26 x 256 0.177 BF
92 conv   512      3 x 3/ 1    26 x 26 x 256 -> 26 x 26 x 512 1.595 BF
93 conv   18       1 x 1/ 1    26 x 26 x 512 -> 26 x 26 x 18 0.012 BF
94 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
95 route  91
96 conv   128      1 x 1/ 1    26 x 26 x 256 -> 26 x 26 x 128 0.044 BF
97 upsample          2x    26 x 26 x 128 -> 52 x 52 x 128
98 route  97 36
99 conv   128      1 x 1/ 1    52 x 52 x 384 -> 52 x 52 x 128 0.266 BF
100 conv  256      3 x 3/ 1    52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
101 conv  128      1 x 1/ 1    52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
102 conv  256      3 x 3/ 1    52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
103 conv  128      1 x 1/ 1    52 x 52 x 256 -> 52 x 52 x 128 0.177 BF
104 conv  256      3 x 3/ 1    52 x 52 x 128 -> 52 x 52 x 256 1.595 BF
105 conv  18       1 x 1/ 1    52 x 52 x 256 -> 52 x 52 x 18 0.025 BF
106 yolo
[yolo] params: iou loss: mse (2), iou_norm: 0.75, cls_norm: 1.00, scale_x_y: 1.00
Total BFLOPS 65.304
avg_outputs = 516723
Allocate additional workspace_size = 52.43 MB
Loading weights from darknet53.conv.74...
seen 64, trained: 0 K-images (0 Kilo-batches_64)
Done! Loaded 75 layers from weights-file
Learning Rate: 0.001, Momentum: 0.9, Decay: 0.0005
Resizing, random_coef = 1.40

608 x 608
Create 6 permanent cpu-threads

```

Sekil 61 YoloV3 eğitim çıktısı

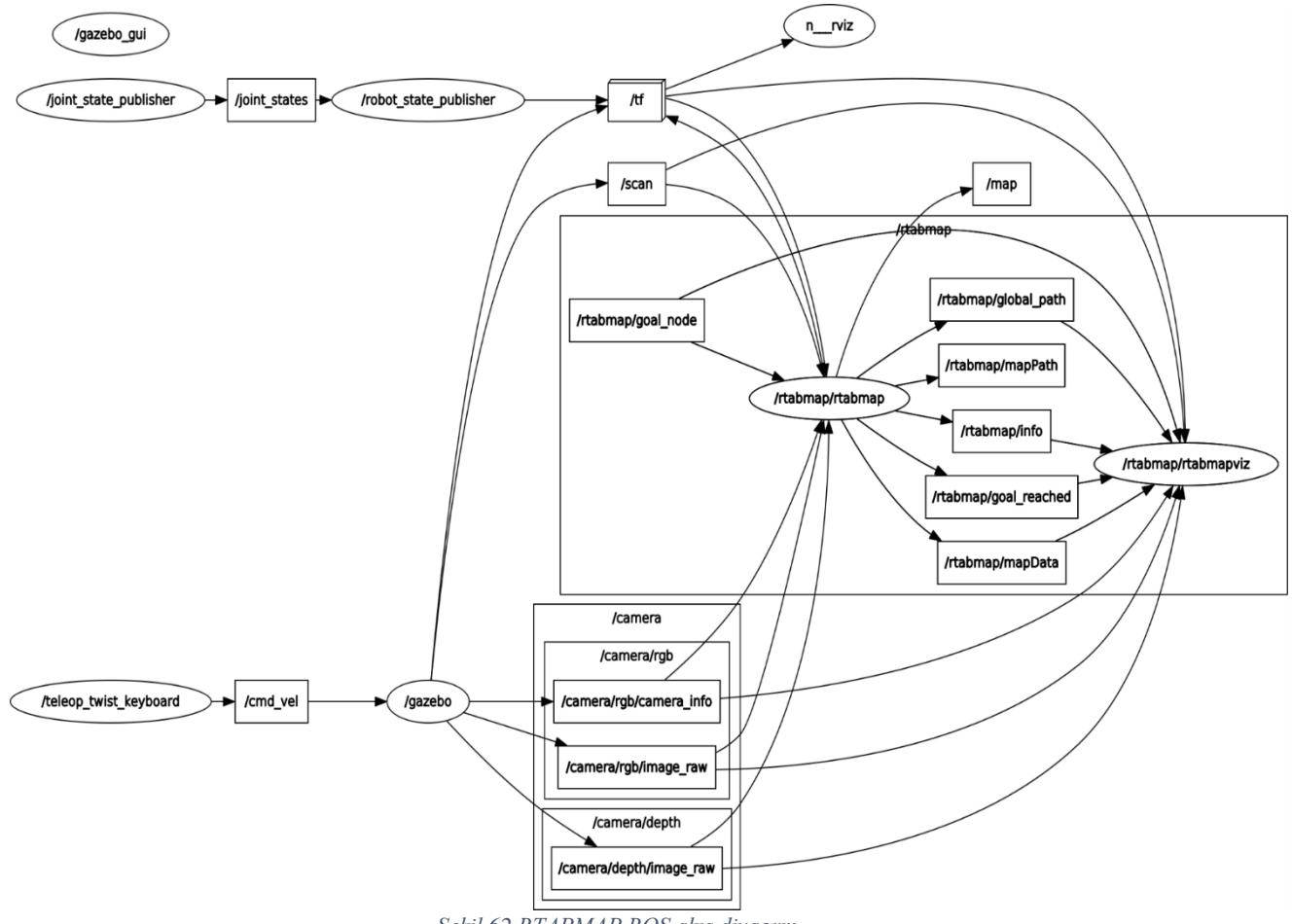
Son olarak eğitilen ağırlık dosyası indirilir ve YoloV3 dedektör koduna giriş olarak verilir. Tanımlama işlemi yapılır. [14]

5.6 RTAP – MAP (Real-Time Appearance-Based Mapping)

Bu kısımda tasarım bölümünde yapılan çalışmaları kapsayan, tüm konuların birlikte çalışması sağlayan gerçek zamanlı görünüşe dayalı haritalama çalışmasından bahsedilecektir. RTAPMAP tek bir simülasyon üzerinde sensör füzyon, lokalizasyon, SLAM, nesne tespiti ve yörünge planlamasının yapılmasına olanak sağlamaktadır.

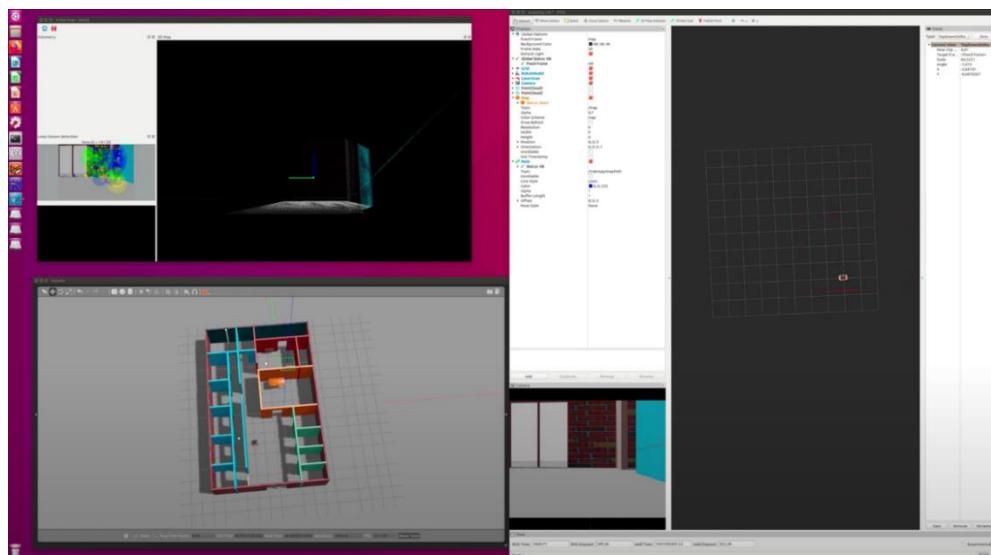
RTABMAP, gerçek zamanlı görünüşe dayalı haritalama, artımlı görünümeye dayalı döngü kapatma dedektörüne dayanan RGB-D, stereo ve Lidar grafik tabanlı SLAM yaklaşımıdır. Döngü kapatma dedektörü, yeni bir görüntüden önceki bir konumdan veya yeni konumdan gelme olasılığını belirlemek için bir BOW(BAG of Words) yaklaşımıdır. Döngü kapatma hipotezi kabul edildiğinde, haritanın grafiğine yeni bir kısıtlama eklenir ve ardından bir grafik optimize edici haritadaki hataları en aza indirir. Bir bellek yönetimi yaklaşımı, döngü kapatma, alıgilama ve grafik optimizasyonu için kullanılan konumların sayısını sınırlamak için kullanılır. Böylece büyük ölçekli ortamlarda gerçek zamanlı kısıtlamalara her zaman saygı gösterilir. [15]

ROS üzerinde yapılan simülasyonun iş akış diyagramı aşağıdaki gibidir.

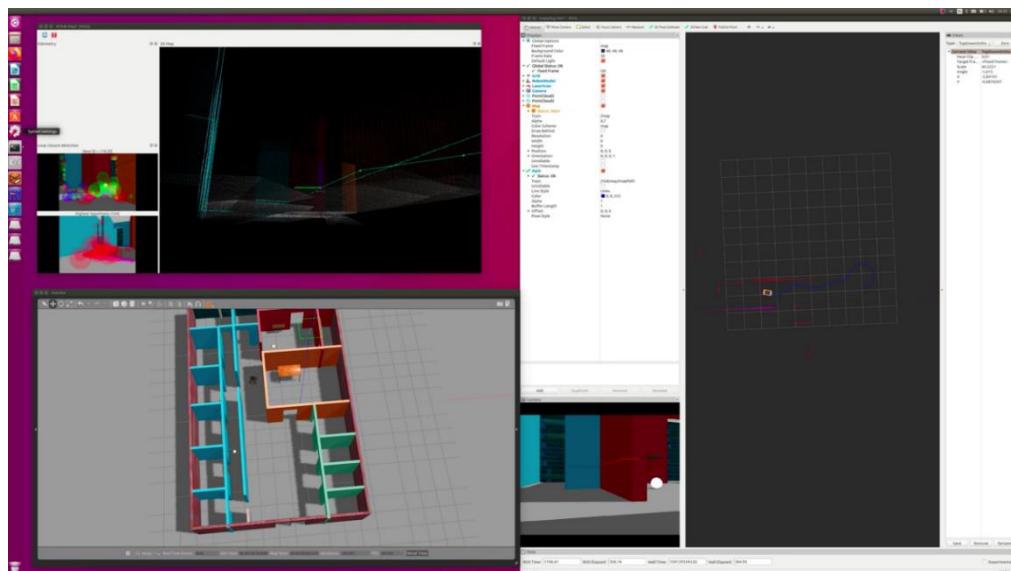


Şekil 62 RTAPMAP ROS akış diyagram

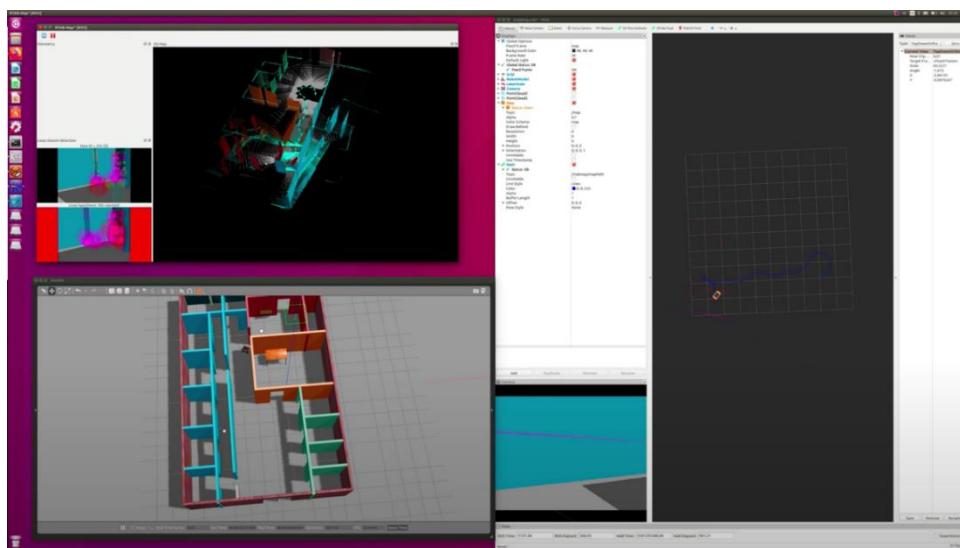
RTAPMAP simülasyonları çıktıları aşağıdaki görsellerde adım adım olarak gösterilmiştir.



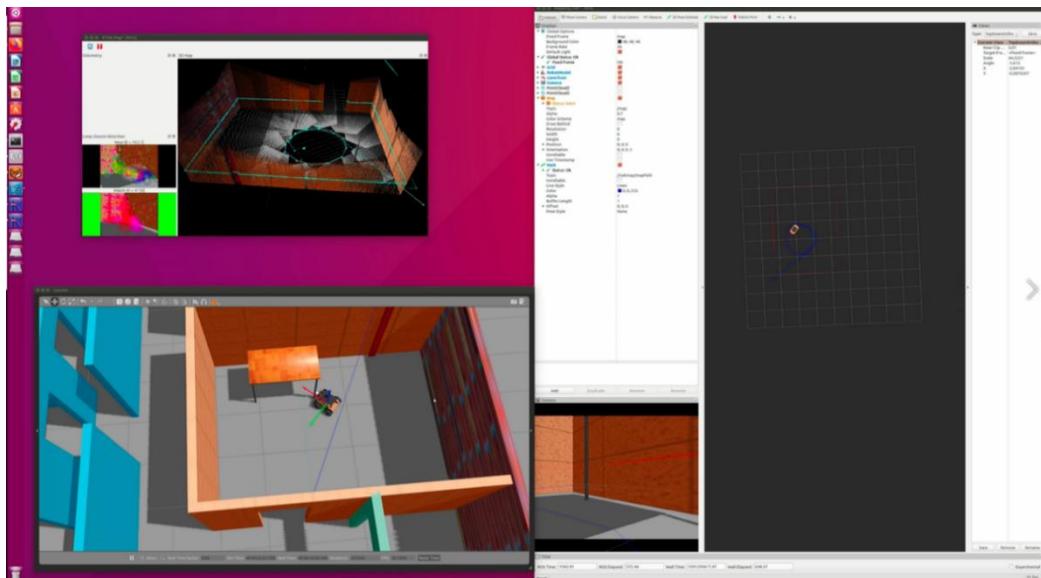
Şekil 63 Başlangıç konumundaki aracın kamera, lidar ve odometriden aldığı verilerin görüntülenmesi



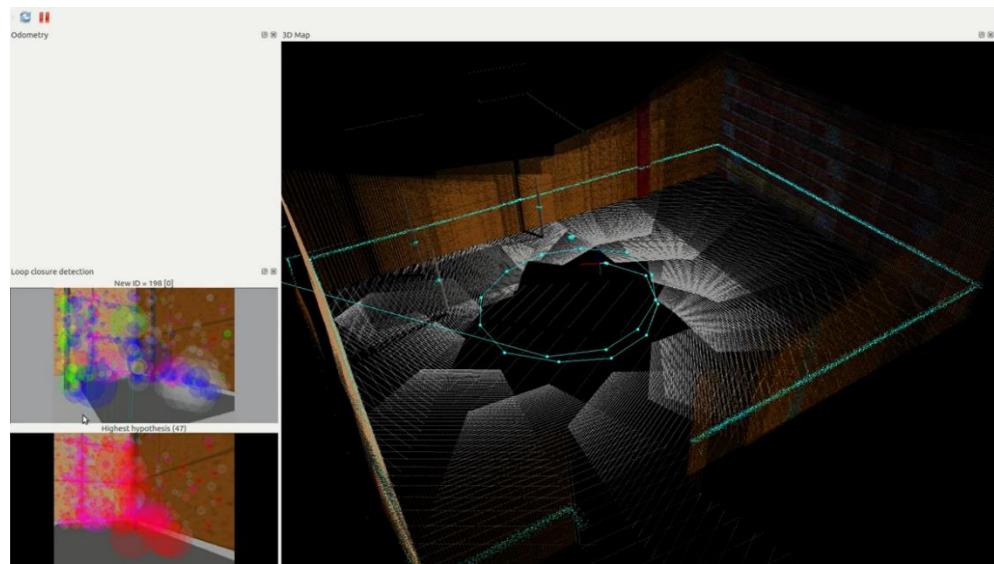
Şekil 64 Teleop ile aracın sürülməsi



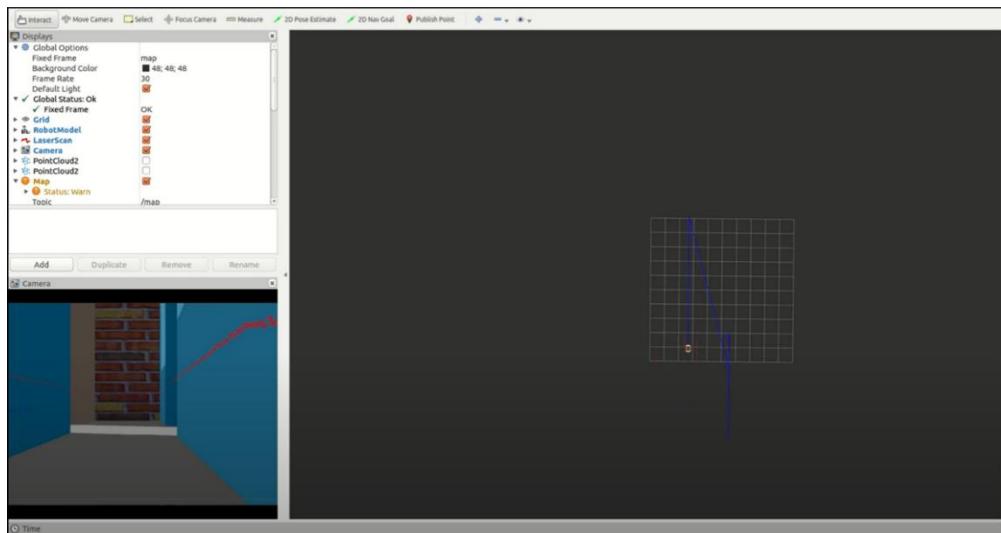
Şekil 65 Lokalizasyon, haritalama ve sensör verilerinin görüntülenmesi



Şekil 66 Stereo kameralı 3B haritalama ve SLAM



Şekil 67 Lokalizasyon [sağ kısım] ve araç kamerasının görüş alanı [sol alta]



Şekil 68 Odometri ve kamera verisinin füzyonlanması

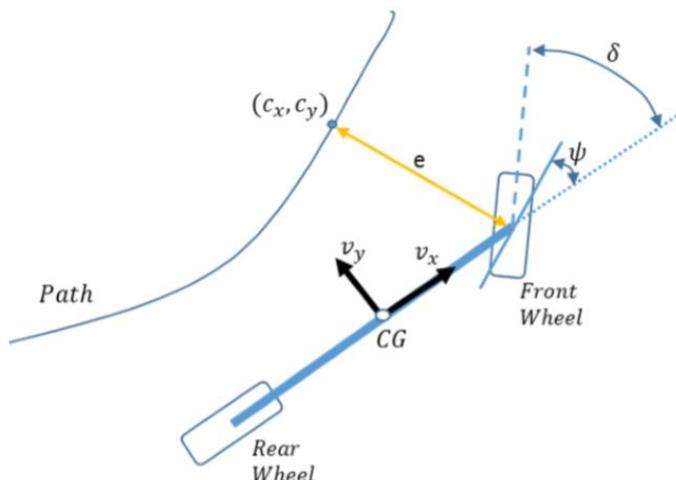
5.7 Pure Pursuit Algoritması ile Lateral Kontrolü Tasarımı (Path Tracking)

Bu bölümde aracın referans yörüngeyi en az yanal hata ile izleyebilmesi için iki yaklaşımı ele alacağız. İki kontrolcü de kinematik bisiklet modeli referans alınarak tasarlanmıştır. İnceleyeceğimiz kontrol metotları şunlardır:

- 1-) Stanley Kontrol Yöntemi
- 2-) Pure Pursuit (Saf Takip) Kontrol Yöntemi

Stanley Kontrol Yöntemi

Stanley yöntemi ilk olarak 2005 yılında Stanford Üniversitesi tarafından Stanley adlı özerk araçlarında DARPA Grand Challenge'da kullanılmıştır. Bu yöntemde ileriye bakmak(look-ahead) yerine yöringedeki en yakın noktaya göre çapraz yol hatası (cross-track error), açısal ve oryantasyon hatalarıyla ilgilenen bir kontrol stratejisi kullanılır. Pure Pursuit kontrol metodundan bir farkı da arka aks merkezi yerine ön aks merkezini referans almasıdır.



Sekil 69 Stanley Metodunun Geometrik Açıklaması

Stanley yöntemine bağlı direksiyon açısı denklemde aşağıdaki gibi ifade edilir:

$$\delta_{sta}(t) = \psi(t) + \tan^{-1} \left(\frac{k * e(t)}{v_x(t)} \right)$$

Burada ψ oryantasyon hatası, k iz hatası kazancı, $e(t)$ çapraz yol hatası ve $v_x(t)$ araç boyuna hızıdır.

c_x aracın ön aks merkezine en yakın noktanın x koordinatıdır.

c_y aracın ön aks merkezine en yakın noktanın y koordinatıdır.

Son olarak çapraz-yol hatasını (cross-track error) tanımlayalım.

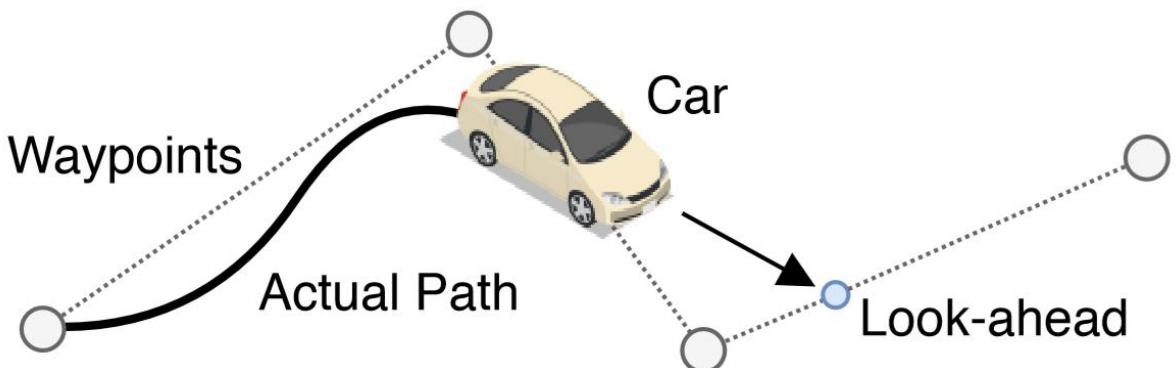
$$e = sign(x'_f - x'_c) * sqrt((x_f - x_c)^2 + ((y_f - y_c)^2))$$

Stanley metodu, cross-track error'un doğrudan kullanımı nedeniyle istenen yola daha hızlı bir şekilde yaklaşır. Yol, pürüzsüz değilse, başka bir deyişle yolu eğriliği sürekli değişmiyorsa (sivri uç durumu) bu yöntem büyük izleme hataları verir. Stanley metodu ön tekerlege en yakın noktayı kullandığı için stratejinin “ileriye dönük” bir davranışını yoktur.

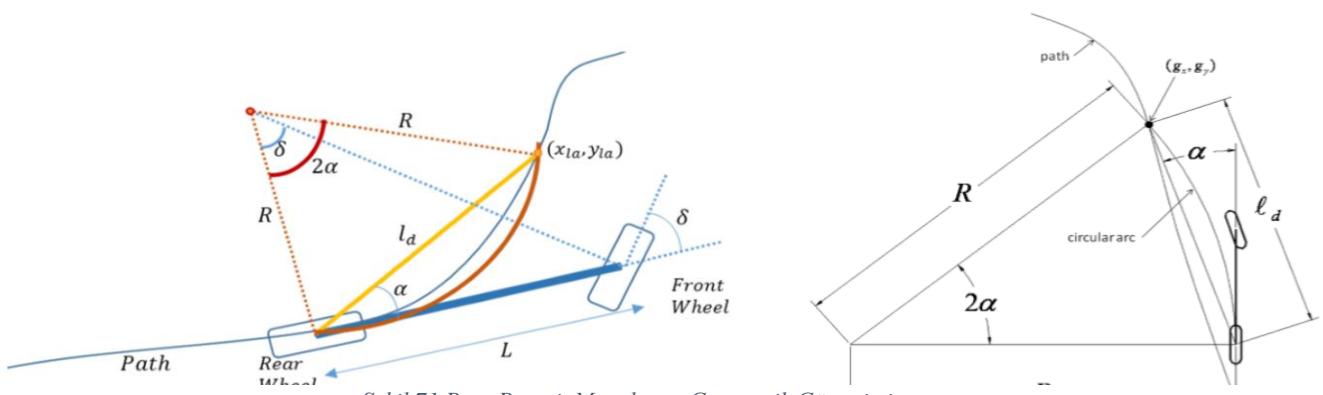
Pure Pursuit (Saf Takip) Kontrol Yöntemi (Tercih edilen yöntem)

Pure Pursuit yönteminin arkasındaki ana fikir, yöringedeki bir hedef noktası için her zaman ileriye bakmak ve aracın o hedef noktadan geçmesini sağlayan direksiyon açısını bulmaktadır. Lastik kaymasının sıfır olduğu varsayılar. Pure Pursuit yöntemi ve varyasyonları, mobil robotlar için yol izleme problemine en yaygın yaklaşılardan biridir. *Bu açıdan bakarsak 1/10 oranında küçültülmüş ve kinematik model ile ifade edilen model aracımızın yöringesini Pure Pursuit Metodu ile kontrol edebiliriz.*

Saf takip yöntemi, arka aks konumunu aracın önündeki yoldaki bir hedef noktasına bağlayan dairesel bir yayın eğriliğinin geometrik olarak hesaplanmasıından oluşur. Hedef noktası (goal point veya look ahead point, (x_{la}, y_{la}) veya (g_x, g_y) olarak ifade edilebilir.) mevcut arka aks konumundan istenen yola ileriye doğru bir mesafe vektörü (look-ahead distance) göre belirlenir.



Şekil 70 Pure Pursuit yönteminin çalışma prensibi



Şekil 71 Pure Pursuit Metodunun Geometrik Gösterimi

δ direksiyon açısını gösterir.

L aracın ön aksı ile arka aksı arasındaki uzaklığını gösterir.

δ aracın direksiyon açısını gösterir.

(x_{la}, y_{la}) veya (g_x, g_y) hedef veya ileriye bakma noktasını gösterir. Aynı anlamdadırlar.

R , aracın dönme merkezine olan radyal uzaklığını gösterir.

l_d , aracın arka aks merkezini referans alarak belirlenen hedef veya ileriye bakma vektörünün uzunluğudur. Pure Pursuit metodu yanal hata, hedef noktası ve ileriye bakma vektörünü belirlerken aracın arka aks merkezini referans alır.

Sinüs teoremini uygularsak:

$$\frac{l_d}{\sin(2 * \alpha)} = \frac{R}{\sin\left(\frac{\pi}{2} - \alpha\right)}$$

$$\frac{l_d}{2 * \sin(\alpha) * \cos(\alpha)} = \frac{R}{\cos(\alpha)}$$

$$\frac{l_d}{2 * \sin(\alpha)} = R$$

Daha önce $\tan(\delta) = \frac{L}{R}$ bulmuştuk. Buna göre

$$\tan(\delta) = \frac{L}{\frac{l_d}{2 * \sin(\alpha)}} = \frac{2 * L * \sin(\alpha)}{l_d}$$

Denklemden δ yi yalnız bırakıp Pure Pursuit direksiyon açısını hesaplayabiliriz.

$$\delta_{pp}(t) = \tan^{-1}\left(\frac{2 * L * \sin(\alpha(t))}{l_d}\right)$$

$\delta_{pp}(t)$ yi ifade eden kontrol yasasını başka bir şekilde de yazabiliriz:

$$\kappa = \frac{2\sin(\alpha)}{l_d} = \frac{1}{R}$$

burada κ , dairesel yayın eğriliğidir.

Buna göre direksiyon açısını şöyle de ifade edebiliriz:

$$\delta = \tan^{-1}(\kappa L)$$

Bu kontrol yasasının daha iyi anlaşılmasına yeni bir değişken tanımlanarak elde edilebilir.

e_{ld} hedef noktanın x bileşeni ile aracın arka aksının x bileşeni arasındaki farktır.(işaretlidir)

$$\kappa = \frac{2}{l_d^2} * e_{ld} \quad \text{bulunur.}$$

Bu denklem, saf izlemenin bir çapraz yol hatası üzerinde çalışan direksiyon açısının oransal bir kontrolörü olduğunu ve aracın önünde ileriye dönük bir mesafe olduğunu ve

kazancının $\frac{2}{l_d^2}$ olduğunu göstermektedir. Uygulamada, kazanım (ileriye dönük mesafe) bağımsız olarak birkaç sabit hızda sabit olacak şekilde ayarlanır ve sonuçta araç hızının bir fonksiyonu olarak atanır.

Pure Pursuit (Saf Takip) Kontrolörünü Ayarlama (Tuning)

Ayarlamayi basitleştirmek için, kontrol yasası yeniden yazılabilir. İleriye doğru olan mesafeyi aracın uzunlamasına hızı ile ölçeklendirebiliriz. İleriye dönük mesafeyi bu şekilde ölçeklendirmek yaygın bir uygulamadır.

Ayrıca, ileriye dönük mesafe (look-ahead distance) genellikle minimum ve maksimum değerlere doyurulur. Buna göre direksiyon açısını ifade eden kontrol yasasını tekrar yazalım:

$$\delta(t) = \tan^{-1} \left(\frac{2 * L * \sin(\alpha(t))}{k * v_x(t)} \right)$$

k Pure Pursuit (Saf Takip) ayarlama (Tuning) kazancıdır.

Pure Pursuit Algoritması ve Değerlendirmesi

1.adım: Global koordinat sisteminde aracın mevcut konumunu ve yönünü bulun.

2.adım: Aracın arka aksına yörüngegedeki en yakın noktayı bulun.

3.adım: Aracın arka aksından l_d kadar uzak olan ve yörünge üzerinde bulunan hedef noktayı arayın. ((x_{la}, y_{la}) veya (g_x, g_y))

4.adım: Global koordinatlarda tanımlanan hedef noktayı araç koordinatlarına dönüştürün.

5.adım: $\alpha(t)$ yi hesaplayın $\sin(\alpha) = \frac{e_{ld}}{l_d}$ ve kontrolü denklemini kullanarak direksiyon açısını bulun.

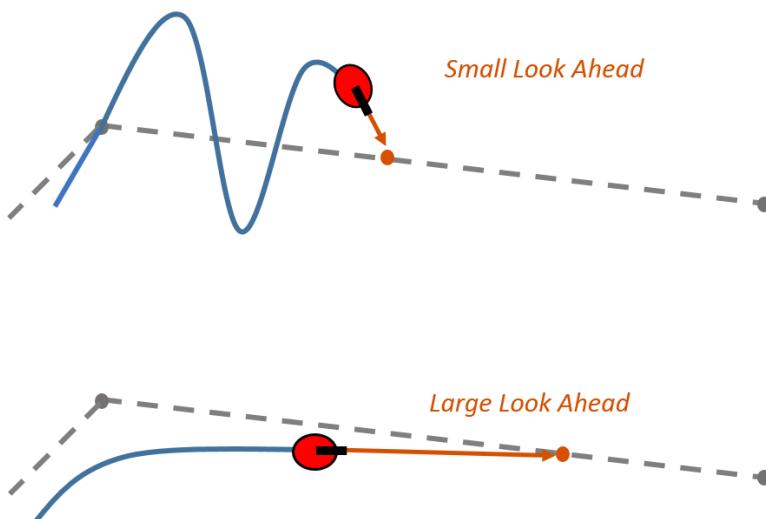
6.adım: 5.adımda bulunan direksiyon açısını uygulayın ve tekrar 1.adıma dönün.

Görülebileceği gibi, Pure Pursuit yönteminin ayarlanması gereken tek parametresi vardır, bu da ileriye bakma mesafesi l_d dir. Bu anlamda ileriye bakma mesafesi (l_d) orantılı bir kazanç(K) gibi davranışır.

l_d , küçük tutulursa araç yörüngeyi daha hassas bir şekilde izlemeye meyillidir ancak kontrol sinyali yani direksiyon açısı hızlı bir şekilde değişir ve bu da yanıtta salınımlara neden olur.

l_d , büyük tutulursa yanıt daha yumuşak hale gelir ancak bazı durumlarda izleme kalitesini ve güvenliğini azaltan büyük köşe kesimleri görülebilir.

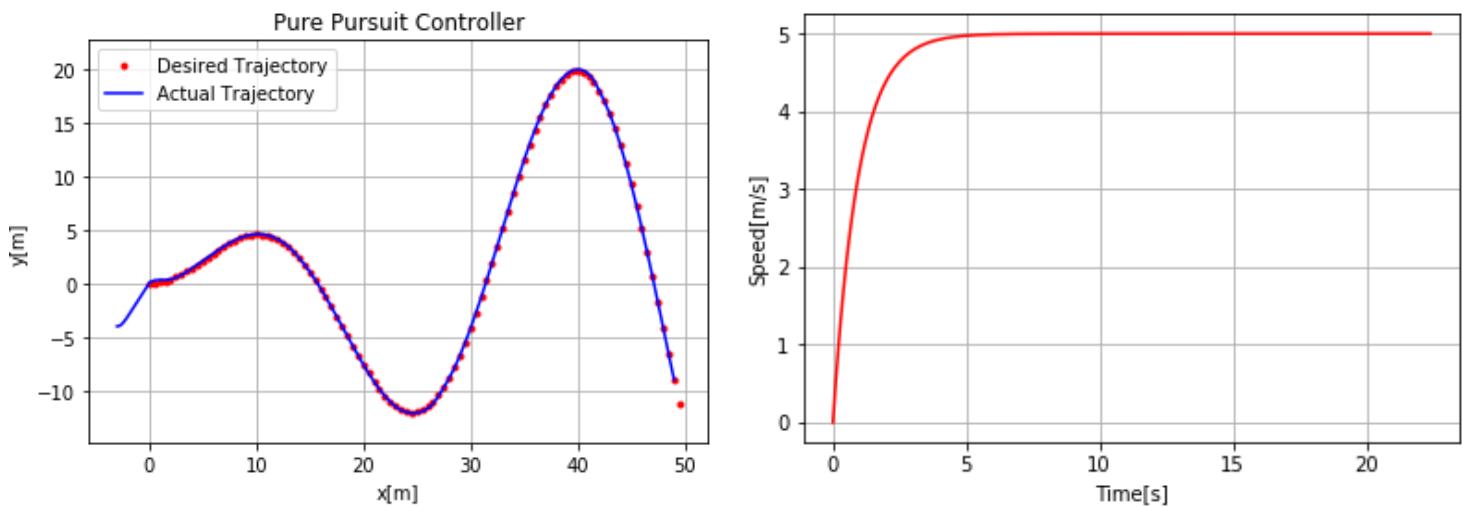
Sonuç olarak yol eğriliği (κ) arttıkça l_d azalır, yol eğriliği (κ) azaldıkça l_d artar. Bu yüzden l_d mesafesi yol geometrisine ve aracın hızına göre ayarlanabilir.



Şekil 72 İleriye Bakma Mesafesinin Sürtüş Performansına Etkisi

Pure Pursuit (Saf Takip) Kontrolcü Çıktıları

Pure Pursuit Algoritmasıyla yazılmış Python script tabanlı kodun başarısını “Matplotlib” kütüphanesine ait görselleştirme araçlarıyla inceledik. İzlenecek yörüngeyi sinüs eğrisi ile modelleyerek gittikçe keskinleştirdik. Böylece zorlu virajlar için kontrolcümüzü test etmiş olduk ve istenen sonucu elde ettik. Ayrıca aracımızın istenen yola yaklaşma hızı ve boyuna hızının oturma zamanının nasıl değiştigini görmek için aracımızın X ve Y başlangıç koordinatlarını izlenecek yolun dışında simülasyonu başlattık. X koordinatı için -3 metre Y koordinatı için -4 metre orijinden uzakta simülasyon başlatıldı. Aracımız hedeflenen maksimum boyuna hız değerine yaklaşık 5 saniyede ulaşmaktadır.



Şekil 73 Pure Pursuit Kontrolcü ile Yörünge Takibi

6. BÜTÇE VE TAKVİM

Bu projenin gerçekleştirilmesi için seçilen komponentlerden grafik işlemci kartı, Lidar sensörü ve kamera kilit rol oynar. Aynı zamanda bu üç komponent toplam bütçenin % 95'ini oluşturur. Algoritmaların geliştirilmesi, nesne tespiti ve haritalama gibi önemli görevleri yerine getiren bu 3 komponente diğer sensörler ve motor sürücü kartı destek olur. Projeye dahil olan komponentlerin satın alma işlemleri için kaynak arayışına gidilmiştir.

Bu amaçla TÜBİTAK 2209 desteği, TÜBİTAK 2242 desteği ve IEEE organizasyonuna başvuru yapılmıştır. Ayrıca özel sektörde yer alan ilgili şirketlere de sponsorluk başvuruları yapılarak hedeflenen bütçeye ulaşımaya çalışılmıştır.

Ekipman	Fiyat
NVIDIA Jetson TX2	7112 ₺
RPLidar A3	4080 ₺
Intel T265	1355 ₺
PCA9685 I2C-PWM Modülü	133 ₺
Sparkfun IMU	370 ₺
Ublox Neo – 6M GPS Modülü	60 ₺
TOPLAM	13,110 ₺

Tablo 3 Bütçe

İş Paketleri	2019										2020				
	Eylül	Ekim	Kasım	Aralık	Ocak	Şubat	Mart	Nisan	Mayıs	Haziran					
Proje Kapsamının Belirlenmesi	10/16														
Ekipmanların Seçimi															
Yapay Zeka Algoritmalarının Geliştirilmesi															
Araç Modellemesi										01/05					
Kontrolcü Tasarımı															
Algoritmaların Simülasyon Ortamında Test Edilmesi															
Elektronik Komponentlerin Montajı															
Test Sürüsü ve Analiz															
Revizyonlar															
Teslim										06/26					

Tablo 4 Takvim

Tablo 4'de yer alan iş paketlerinden “elektronik komponentlerin montajı” ve “Test Sürüsü” görevleri Covid – 19 sebebi ile daha sonraki birtarihe ertelenmiştir.

7. RİSK ANALİZİ



GPU Performansı

Kamera ve LIDAR'dan gelen verilerin işlenmesinde yaşanacak gecikme riskine karşı Jetson GPU kartının geliştirilmesi

Haberleşme

Motorlar ve sensörler gibi bileşenlerin farklı yazılım dillerinde çalışırken birbirleri ile haberleşme problemi karşın ara yazılımların eklenmesi

Güç Tüketimi

Araç üzerindeki bileşenlerin güç tüketiminin, aracın çalışma süresini etkilemesine karşın yedek bataryaların bulundurulması

Entegrasyon

Geliştirilen algoritmanın araç performansında sorun çıkmasına karşılık, algoritmanın öncesinde simülasyon ortamında test edilerek iyileştirilmesi

Şekil 74 Risk Analizi

8. SONUÇ VE GELECEK İŞLER (FUTURE WORKS)

Gelişen sensör teknolojisi ve hesaplama gücünün yıllar geçtikçe olağanüstü artması, bununla birlikte özel teşebbüs (örn: Tesla, Zoox, Waymo vs.) ve devlet yatırımlarının (örn: TOGG:Türkiye'nin Otomobili Girişim Grubu) yapay zeka ve elektrikli otonom araç sistemlerine yönelikmesi bizi "***Yapay Zeka ve Sensör Füzyonu Tabanlı Otonom Sürüs***" projesine yönelmeye teşvik etti.

Bu proje kapsamında 1/10 oranında küçültülmüş model aracımıza çeşitli otonom sürüş görevlerini başarıyla yaptırmayı hedefledik. Bu anlamda Waymo şirketinin çalışma prensibini model almış olduk. Waymo şirketi de sipariş üzerine otonom olmayan araçlara çeşitli seviyelerde otonom sürüş özelliklerini ekliyor. Proje kapsamını ve tasarım kriterlerini belirlerken bazı şartları "OPEN ZEKA-MARC MİNİ OTONOM ARAÇ YARIŞMASI KURAL KİTAPÇIĞI" referans aldık. Diğer tasarım kriterlerimizi belirlerken bütçemizi ve sahip olduğumuz donanımların çalışma aralıklarını ve kapasiteleri göz önünde bulundurduk. Tasarım kriterlerimizi ve donanım ihtiyaçlarını "OPEN ZEKA-MARC MİNİ OTONOM ARAÇ YARIŞMASI" göre belirledikten sonra donanım vb. ihtiyaçlarımıza karşılayabilmek için kaynakbütçepaydaş aramalarımıza devam ettik. Daha önceden teorik açığımızı kapatmak için Coursera platformunda eğitim veren Toronto Üniversitesi'nden Self-Driving Cars Specialization isimli kurs için finansal destek aldık. Tüm kurs içerikleri, sınavları ve sertifika erişimi ekibimize ücretsiz sağlanmıştır. Son olarak projemizi IEEE ve Tübitak gibi saygın kurumların düzenlediği Lisans Bitirme Projesi yarışmasında sergileyeceğiz. Paydaşlarımızı ve tasarım kriterlerini belirledikten sonra BÇ kapsamında görevleri beş ana başlığa böldük:

- Lokalizasyon
- Çevre Haritalama
- Nesne Tanıma ve Sınıflandırma
- Hareket Planlama
- Yörünge Takibi(Kontrol)

ve kaynak taraması yapıp bilgi altyapımızı güçlendirdik. Dört başlık altında incelenen görevler bittikten sonra sensör füzyonu, motor kontrolü, haberleşme, tüm sistemin montajı ve test sürüsü-optimizasyon sonrası projenin tamamlanması hedeflenmiştir.

Belirlenen beş ana başlık ve yol haritasına göre teknik şartnamenin isterlerini karşılayacak birçok simülasyon ROS Ortamı ve Python Görselleştirme Araçları kullanılarak yapılmıştır.

Elde edilen simülasyon sonuçlarına göre modelimiz:

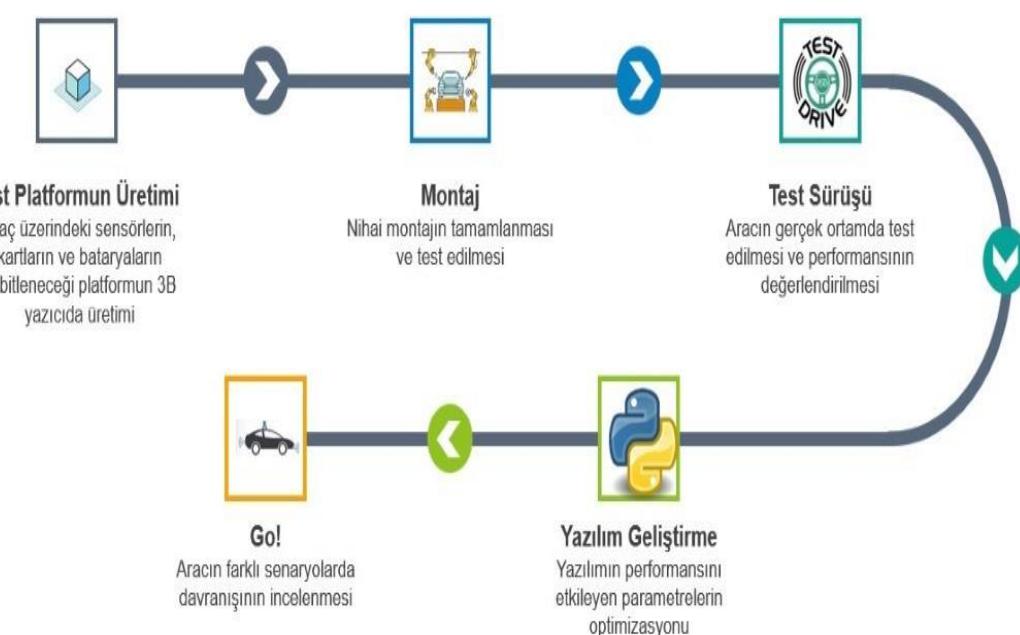
- GPS ve IMU verilerini kullanarak konumunu tahmin edebiliyor.

- Kamera ve LIDAR verilerini kullanarak çevre haritalama yapabiliyor.
- Kameradan alınan verileri OpenCv kütüphanesi yardımıyla işleyerek nesneleri tanıyor ve sınırlandırabiliyor.
- RRT Algoritması ile takip edilecek en doğru yörüngeyi planlayabiliyor.
- Bisiklet modeli temel alınarak oluşturulmuş Pure Pursuit kontrolcü ile hedef yörüngeyi en az hata ile takip ediyor.

Bu sonuçlara ek olarak COVID-19 salgını yayılmadan önce prototip çalışması kapsamında:

- Tüm sensörlerin montajlanması ve elektronik bağlantılarının yapılması için katı model olarak platform tasarlanmıştır.
- Jetson TX2 merkezi işlem birimi , hem DC Motor hem de Servo Motor ile PCA9685 PWM Modülüyle haberleşmiştir. Bu sayede aracımızın sırasıyla hızını ve viraj açısını belirleyen DC Motor ve Servo Motor kontrolü yapılmıştır.
- Farklı görevleri ve ana başlıklarını sağlamak için tasarlanan algoritmaların başarısı, YTÜ Mekatronik E2 Blok ortamı ROS ortamında modellenerek ROS Bot 2.0 ile test edilmiştir ve nihai montaj sonrası test için de hazırır. ROS Bot 2.0 diferansiyel tıhrik ile çalışan bir mobil robot olduğu için Pure Pursuit kontrolcüsü hariç tüm algoritmalar ortak çalışacak şekilde simülle edilmiştir.

Ağustos ayında yapılması planlanan IEEE ve Tübitak 2242 Lisans Bitirme Çalışması Yarışmasına hazırlık kapsamında virus tedbirleri azaltıldıktan hemen sonra ilk iş olarak yapılması gereken bazı gelecek çalışmalar görseldeki gibidir:



Şekil 75 Gelecek işler

REFERANSLAR

- [1] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., . . . Stang, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 661-692.
- [2] RACECAR. (2019). MIT RACECAR: <https://mit-racecar.github.io/> adresinden alındı
- [3] O'Kelly, M., Zheng, H., Jain, A., Auckley, J., Luong, K., & Mangharam, R. (2019). *Tech Report: TUNERCAR: A Superoptimization Toolchain for Autonomous Racing*. Pennsylvania: University of Pennsylvania.
- [4] Mathworks. (tarih yok). *Automated Driving Toolbox*. MathWorks: <https://www.mathworks.com/products/automated-driving.html> adresinden alındı
- [5] Gladyshev, P., & Patel, A. (2004). Finite state machine approach to digital event reconstruction. *Digital Investigation*, 130 - 149.
- [6] DUSTIN, F. (2017, Mart 7). *NVIDIA Jetson TX2 Delivers Twice the Intelligence to the Edge*. NVIDIA Developer Blog: <https://devblogs.nvidia.com/jetson-tx2-delivers-twice-intelligence-edge/> adresinden alındı
- [7] Nvidia. (2020). *OTONOM MAKİNELER*. nvidia: <https://www.nvidia.com/tr-tr/autonomous-machines/embedded-systems/jetson-tx2/> adresinden alındı
- [8] *DATA SHEET [PRELIMINARY], NVIDIA Jetson TX2 System-on-Module*. (2017). nvidia: <https://download.kamami.pl/p569306- DATA%20SHEET%20-%20NVIDIA%20Jetson%20TX2%20System-on-Module.pdf> adresinden alındı
- [9] Jetto, L., Longhi, S., & Venturini, G. (1999). Development and experimental validation of an adaptive extended Kalman filter for the localization of mobile robots. *IEEE Transactions on Robotics and Automation*, 219-229.
- [10] Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 846-894.
- [11] Umari, H. (2019, Haziran 7). *rrt_exploration*. Ros.org: http://wiki.ros.org/rrt_exploration adresinden alındı
- [12] Mitka, Ł. (2019). *Visual object recognition*. HUSARIAN DOCS: <https://husarion.com/tutorials/ros-tutorials/4-visual-object-recognition/> adresinden alındı
- [13] Ibañez, D. (tarih yok). *How to train YOLOv3 using Darknet on Colab notebook and optimize the VM runtime load times*. Tutorial DarknetToColab.ipynb: https://colab.research.google.com/drive/1lTGZsfMaGUpBG4inDIQwIJVW476ibXk_#scrolledTo=46LRTt-5Pr52 adresinden alındı
- [14] Redmon, J. (2013-2016). *Darknet: Open Source Neural Networks in C*. pjreddie.com: <https://pjreddie.com/darknet/> adresinden alındı
- [15] IntRoLab. (2019). *rtabmap_ros*. Ros.org: http://wiki.ros.org/rtabmap_ros