T.C.
İSTANBUL SABAHATTİN ZAİM ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR BİLİMLERİ VE MÜHENDİSLİĞİ BİLİM DALI

GÖRSEL TABANLI HARİTALAMA İLE OTONOM ROBOTLARDA HEDEFE YÖNELİK OPTİMUM YOL BULMA

YÜKSEK LİSANS TEZİ

Tuğba FIÇICI

İstanbul
Ocak-2025

T.C.
İSTANBUL SABAHATTİN ZAİM ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR BİLİMLERİ VE MÜHENDİSLİĞİ BİLİM DALI

GÖRSEL TABANLI HARİTALAMA İLE OTONOM ROBOTLARDA HEDEFE YÖNELİK OPTİMUM YOL BULMA

YÜKSEK LİSANS TEZİ

Tuğba FIÇICI

Tez Danışmanı
Dr. Öğr. Üyesi Erdal ALİMOVSKİ

İstanbul
Ocak-2025

Lisansüstü Eğitim Enstitüsü Müdürlüğüne,

Bu çalışma, jürimiz tarafından Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Bilimi ve Mühendisliği Bilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Dr. Öğr. Üyesi Erdal ALİMOVSKİ

Üye: Doç. Dr. Emir SEYYEDABBASİ

Üye: Dr. Öğr. Üyesi Artrim KJAMILJI

Onay

Yukarıdaki imzaların, adı geçen öğretim üyelerine ait olduğunu onaylarım.

Prof. Dr. Erhan İÇENER

Enstitü Müdürü

BİLİMSEL ETİK BİLDİRİMİ

Yüksek lisans tezi olarak hazırladığım **"Görsel Tabanlı Haritalama ile Otonom Robotlarda Hedefe Yönelik Optimum Yol Bulma"** adlı çalışmanın öneri aşamasından sonuçlandığı aşamaya kadar geçen süreçte bilimsel etiğe ve akademik kurallara özenle uyduğumu, tez içindeki tüm bilgileri bilimsel ahlak ve gelenek çerçevesinde elde ettiğimi, tez yazım kurallarına uygun olarak hazırladığımı, bu çalışmamda doğrudan veya dolaylı olarak yaptığım her alıntıya kaynak gösterdiğimi ve yararlandığım eserlerin kaynakçada gösterilenlerden oluştuğunu beyan ederim.

Tuğba FIÇICI

ÖN SÖZ

Araştırmamdaki her aşamada bana yardımcı olan değerli tez danışmanım Dr. Öğr. Üyesi Erdal ALİMOVSKİ'ye, bu süreç boyunca desteklerini benden esirgemeyen değerli hocamız Doç. Dr. Gökhan ERDEMİR'e, beni hayatım boyunca koşulsuz şartsız destekleyen babam Selim FIÇICI ve annem Perihan FIÇICI'ya, bu süreç boyunca yanımda olan ve desteğini esirgemeyen nişanlım Buğra GÖRGÜN'e teşekkürlerimi sunarım.

Tuğba FIÇICI

İstanbul-2025

ÖZET

GÖRSEL TABANLI HARİTALAMA İLE OTONOM ROBOTLARDA HEDEFE YÖNELİK OPTİMUM YOL BULMA

Tuğba FIÇICI

Yüksek Lisans, Bilgisayar Bilimleri ve Mühendisliği

Tez Danışmanı: Dr. Öğr. Üyesi Erdal ALİMOVSKİ

Ocak, 2025 -88 Sayfa

Otonom robotların, bilinmeyen veya yarı bilinen ortamlarda kendi konumlarını belirlemesi, engellerden kaçınması ve etkili yol planlaması yapması kritik bir gereksinimdir. Bu tezde, otonom bir robotun, Gazebo simülasyon ortamında yer alan cadde ve işletmelerin bulunduğu bir mahalle ortamında keşif, haritalama ve yol planlama görevlerini gerçekleştirmesi incelenmiştir. Robot, keşif algoritması ile ortamı tarayarak kamera sensöründen aldığı görüntülerdeki işletme tabelalarında mevcut olan metinleri tespit edecek ve metin algılama algoritmasıyla bu tabelaların isimlerini tanıyacaktır.

Metin tespit etmek için EAST algoritması kullanılacaktır. Metin tanıma işlemi için ise Optik Karakter Tanıma (OCR) algoritmalarından EasyOCR, Keras ve Tesseract kullanılarak her işletme için koordinatlar belirlenecektir. Bu süreç sonunda, işletme isimlerini ve konumlarını içeren görsel bir harita oluşturulacaktır. Haritalama işlemi tamamlandıktan sonra, robota çeşitli görevler verilecektir. Örneğin, robota rastgele bir başlangıç noktasından seçilen X işletmesine gitme görevi verilecek ve robot bu görevi hem Dijkstra hem de A* algoritmalarını kullanarak gerçekleştirecektir. Farklı görevler aracılığıyla ve farklı parametreler göz önünde bulundurarak iki yol planlama algoritmasının performansı karşılaştırılacaktır. Bu çalışma, yol planlama algoritmaları ve OCR modellerinin güçlü ve zayıf yönlerini analiz ederek, otonom robot navigasyon sistemlerine yönelik önemli katkılar sağlamayı hedeflemektedir.

Anahtar Kelimeler: otonom robot, optik karakter tanıma, metin algılama, metin tespiti, görsel konumlandırma

ABSTRACT
OPTIMAL PATH PLANNING FOR AUTONOMOUS ROBOTS USING VISUAL-BASED MAPPING

Tuğba FIÇICI

Master, Bilgisayar Bilimleri ve Mühendisliği

Supervisor: Asst. Prof. Dr. Erdal ALİMOVSKİ

January, 2025 -88 Pages

The ability of autonomous robots to localize themselves, avoid obstacles, and perform efficient path planning in unknown or partially known environments is a critical requirement. This thesis explores the abilities of an autonomous robot to perform exploration, mapping, and path planning in a simulated neighborhood environment created using the Gazebo simulation platform, which includes streets and businesses. The robot uses an exploration algorithm to scan the created environment, detects business signboards with its camera sensor, and identifies the text on these signboards through a text detection algorithm. Optical Character Recognition (OCR) models, including EasyOCR, Keras, and Tesseract, are employed to recognize the business names. After the text detection and recognition of businesess in the environment coordinates are assigned to each identified business. This process culminates in the generation of a visual map containing business names and their respective locations.

Following the mapping phase, the robot is assigned various tasks, such as navigating from a randomly chosen starting point to a specified business (e.g., X). The navigation tasks are executed using both Dijkstra and A* algorithms, enabling a comparative analysis of the performance of these path planning methods under different scenarios. This study provides a comprehensive evaluation of the strengths and limitations of path planning algorithms and OCR models, offering significant contributions to the development of robust autonomous robot navigation systems.

Keywords: autonomous robot, optical character recognition, text detection, text recognition, visual localization

İÇİNDEKİLER

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

SEMBOLLER LİSTESİ

D_{ro}	: İki dizi arasındaki benzerlik skoru
K_m	: Dizideki eş karakterlerin sayısı
$\|S1\|$	: Birinci metin girdisinin uzunluğu
$\|S2\|$	: İkinci metin girdisinin uzunluğu
$f(n)$	: Toplam tahmini mesafe
$g(n)$	: n düğümüne gelene kadar hesaplanan gerçek mesafe
$h(n)$	: n düğümünden hedef düğüme olan tahmini mesafe
n	: Mevcut düğüm
x_1	: İki mesafe arası ölçümler için kullanılan ilk mesafe konumu
x_2	: İki mesafe arası ölçümler için kullanılan ikinci mesafe konumu
w	: Robotun açısal hızı
v	: Robotun doğrusal hızı

KISALTMALAR LİSTESİ

AMCL	: (Adaptive Monte Carlo Localization) Adaptif Monte Carlo Konumlandırması
API	: (Application Programming Interface) Uygulama Programlama Arayüzü
CNN	: (Convolutional Neural Network) Evrişimsel Sinir Ağları
CPU	: (Central Processing Unit) Merkezi İşlem Birimi
CRAFT	: (Character-Region Awareness for Text detection) Metin Algılama için Karakter-Bölge Farkındalığı
CRNN	: (Convolutional Recurrent Neural Network) Evrişimsel Tekrarlayan Sinir Ağları
CTC	: (Connectionist Temporal Classification) Bağlantısal Zamansal Sınıflandırma
DDS	: (Data Distribution Service) Veri Dağıtma Servisi
EAST	: (Efficient and Accurate Scene Text Detector) Etkili ve Doğru Sahne Metin Dedektörü
FOV	: (Field of View) Görüş Alanı
FPS	: (Frame Per Second) Saniyedeki Kare Sayısı
GPS	: (Global Positioning Systems) Küresel Konumlama Sistemi
GPU	: (Graphics Processing Unit) Grafik İşlemci Birimi
LiDAR	: (Light Detection and Ranging) Işık Algılama ve Mesafe Ölçme Sensörü
LSTM	: (Long Short Term Memory) Uzun Kısa Süreli Bellek
NMS	: (Non-Maximum Suppression) Maksimum Olmayanı Bastır
OCR	: (Optical Character Recognition) Optik Karakter Tanıma
POI	: (Points of Interest) İlgi Noktası
RAM	: (Random Access Memory) Rastgele Erişimli Bellek
RNN	: (Recurrent Neural Network) Tekrarlayan Sinir Ağları
ROS	: (Robot Operating System) Robot İşletim Sistemi
RRT	: (Rapidly-Exploring Random Trees) Hızlı Keşfedilen Rastgele Ağaçlar
RViz	: (ROS Visualization) ROS Görselleştirmesi
SDF	: (Simulation Description Format) Simülasyon Tanım Formatı

SLAM	: (Simultaneous Localization and Mapping) Eş Zamanlı Konum Belirleme ve Haritalama
SM	: (Sequence Matcher) Dizi Eşleştirme
Sn	: Saniye
TCPROS	: (Transmission Control Protocol for ROS) ROS için İletim Kontrol Protokolü
TF	: (Transform Frame) Dönüşüm Çerçevesi
UDPROS	: (User Datagram Protocol for ROS) ROS için Kullanıcı Veri Bloğu Protokolü
VIRT	: (Virtual Memory) Sanal Bellek
YAML	: (YAML Aint Markup Language) YAML Bir Biçimlendirme Dili Değildir

BİRİNCİ BÖLÜM
GİRİŞ

Robotik sistemlerde, son yılların popüler araştırma konularından biri otonom mobil robotlardır. Bu robotlardan, bilinmeyen ya da kısmen bilinmeyen bir ortamda çalışıp dinamik olarak değişen engellerden ve sabit engellerden kaçınarak hareketine devam etmesi beklenir. Hem kapalı alanlarda hem açık alanlarda yolu takip etmesi ve hedefe ulaşması için tasarlanmıştır. Otonom robotun işlevlerini yerine getirebilmesi için robotun üzerine entegre olan sensör verilerinden yararlanması gerekir (Alatise & Hancke, 2020). Hangi sensörun hangi amaçla kullanılacağı ortama, robotun görevine ve senaryoya göre değişiklik gösterebilir. Örneğin radyasyonlu bölgelerdeki kaynakların saptaması için geiger sayacı yardımıyla tespit edilmesi verilebilir (Lin & Tzeng, 2014). Diğer bir örnek olarak bir yangın söndürme robotunun yangın çıkan bir ortamda sıcaklık sensörü ve kamera verileriyle yangının olduğu bölgeyi saptaması beklenir (P Anantha Raj, 2018). Otonom sistemleri, insan sağlığını koruyan ve işini kolaylaştıran uygulamalar ile her alanda kullanılmaya başlanmıştır.

Dinamik ortamlar başta olmak üzere, engellerden kaçınmak robotik sistemlerde en önemli konulardan biridir. Tüm mobil robotlar, çarpışmayı önlemek için bir tür engelden kaçınma algoritmasına sahiptir. Bu algoritmalar, bir engeli algılayıp çarpışmayı önlemek için robotu durdurmaya dayanan basit algoritmalardan, robotun engellerin etrafından dolanmasını sağlayan karmaşık algoritmalara kadar farklılık gösterir (Borenstein & Koren, 1989).

Bir mobil robotun en önemli görevlerinden biri bir hedefe ulaşırken yol planlaması yapmasıdır. Yıllar içinde farklı türlerde algoritmalar geliştirilmiş ve kullanılmıştır. Algoritmaların etkinliği, doğrudan ortamın statik ve dinamik oluşundan etkilenmektedir. Statik ortamlar için geleneksel algoritmalar (Dijkstra, A*) başarılı bir performans gösterseler de dinamik ortamlardaki değişkenler algoritmaların performansını büyük ölçüde etkilemektedir. Çalışmada kullanılan algoritmalar ele alındığında, geleneksel A* algoritması, global yol planlamasında yaygın olarak kullanılsa da ürettiği yolların her zaman en kısa yol olmaması, bu yöntemin sınırlamaları arasında yer alır

(Ju, Qinghua, & Yan, 2020). Ortam karmaşıklığından oluşan sorunlara çözüm olarak algoritmaların beraber kullanılması etkili yöntemlerden biridir. Modern robotik uygulamalarda A* algoritmasının yerel ve global planlama için D* veya diğer yöntemlerle kombinasyonu, karmaşık ve dinamik ortamlarda önemli başarılar sağlamıştır. Örneğin, hünnap bahçesi gibi yoğun ortamlarda, hibrit bir yöntemle yol optimizasyonu ve engel kaçınma başarıyla gerçekleştirilmiştir (Z. Liu & Zhang, 2023). Yöntemlerin sınırlamaları için çeşitli çalışmalar yapılarak algoritmaların iyileştirilmesi hedeflenir. Çalışmada kullanılan algoritmalardan biri Dijkstra algoritmasıdır. Bu algoritma en uygun çalıştığı ortamlar olarak verilerin önceden tanımlandığı statik ortamlar gösterilebilir. Sürekli değişen dinamik ortamlarda da kullanıldığı uygulamalar vardır. Dinamik ortamlar için yerel planlama kullanılır. Ortam kısmen ya da hiç bilinmediği durumda engellerle ilgili düğüm bilgileri anında hesaplanarak en kısa yolu değerlendirebilmesi için Dijkstra algoritmasına verilir. Ancak, dinamik ortamlar için Dijkstra tek başına yeterli gelmemektedir (Karur, Sharma, Dharmatti, & Siegel, 2021).

Dinamik ortamlar için, günümüzde yapay zekâ destekli algoritmaların (RRT gibi) kullanımı daha çok görülmektedir. Mobil robotlarda otonom hareket yeteneklerini daha da artırma potansiyeline sahip olmasıyla beraber gelecekteki araştırmalar, dinamik ve karmaşık çevrelerde robotların daha özerk hareket etmesini sağlayacak tekniklerin geliştirilmesine odaklanması beklenmektedir (Tian et al., 2021). Otonom mobil robotlar için yol planlama ve navigasyon algoritmalarına artan talep kapsamlı bir şekilde makaleler tarafından ele alınmaktadır. Bu algoritmaların sadece engelden kaçınmayı sağlamanın ötesinde, robotun başlangıç noktasından hedefe giden yolu optimize etmesi gerektiğini ve belirli kısıtlamalara uyması gerektiğini vurgulamaktadır (Zhang, 2021).

Otonom robotların ortam içerisinde konumlandırma ve harita çıkarması için Eş Zamanlı Konumlandırma ve Haritalama (SLAM) teknolojisi kullanılır. Temel işleyişinde ilk adım robottan sensör verilerinin toplanmasıdır. Bu sensörler LiDAR, kamera, radar ya da GPS olabilir. Görüntü verilerden duvar köse veya nesne gibi özellikler çıkartılarak harita oluşturulurken ve robotun konumu tespit edilirken referans olarak kullanılır. Çıkartılan harita ızgara ya da graf tabanlı bir yapıda olabilir. Robotun kendi konumunu tespit etmesi özelliklerin referanslarının dışında Bayes filtreleri (Kalman filtresi,

Parçacık filtresi) kullanılarak saptanır. Bu işlemlerin tamamı eş zamanlı olarak hesaplanır ve konum, harita bilgileri güncellenir (Chen et al., 2023).

Optik Karakter Tanıma (OCR), girdi olarak metin içeren bir görüntüyü alır ve çıktı olarak yazılı formatta metne dönüştürür. Temel olarak özelliklerin çıkartılması ve sınıflandırılmasına dayanır. OCR kendi içinde, dinamik bir ortamda belirli hız ve açıda alınan görüntü verilerinin kullanıldığı sistemler çevrimiçi ve statik ortamlarda taranmış görüntü verilerinin kullanıldığı sistemler çevrimdışı sistem olarak ikiye ayrılır (Memon, Sami, Khan, & Uddin, 2020). Son yıllarda yapay sinir ağlarına dayanan modeller metin tanımada kullanılmaktadır. Bu modellere örnek olarak Tesseract, Keras ve Easy OCR verilebilir.

Bu tez çalışmasında, gezgin robotların simülasyon ortamında otonom bir şekilde hareket ederek görüntü algılama ve yol bulma süreçlerini entegre bir yapıda nasıl gerçekleştirdiğini ve bu süreçte kullanılan yol bulma algoritmalarının performanslarını karşılaştırmayı amaçlamaktadır. Araştırma, ROS 2 işletim sistemi içerisinde modüller arası haberleşmesi sağlanan mobil robotun hazırlanan simülasyon ortamında oluşturulmuştur. Çalışma ortamı, işletmelerin yoğunlukta olduğu birbirine bağlı caddeler üzerinde belirlenen senaryolara göre gerçekleştirilecektir. Ortamda belirlenen bir noktadan başlayan robot, işletme tabelalarından metinleri tespit ederek harita üzerinde işletmenin konumunu kaydedilip, POI verileriyle beraber tutulan ortam haritası ile robotun verilen hedef işletmelere hızlı ve en kısa yoldan ulaşması hedeflenmiştir. Metin tespiti için üç farklı model (EasyOCR, Keras, Tesseract) ve iki farklı yol bulma algoritması (Dijkstra, A*) kullanılmıştır. Deney ortamı işletim sistemi olarak Ubuntu 22.04 ve ROS 2 Humble sürümü kullanılmıştır. Simülasyon ortamı olarak Gazebo, robot olarak Turtlebot3 Waffle modeli kullanılmıştır.

1.1. Problem

Gezgin robotların otonom bir şekilde hareket edebilmesi için sensörlerinden aldıkları verileri işleyerek üç temel işlevi yerine getirmesi beklenirken bilinmeyen bir ortam içerisinde robotun karşılaşabileceği engellerden kaçınması, ortam içerisinde kendi konumunu tespit edebilmesi ve hedef bir koordinata ulaşırken yol planlaması yapması bek-

lenir. Bahsedilen işlevlerin gerçek zamanlı olarak yapılması bir problemdir. Ek olarak kamera verisinden okunan metinin doğruluğu ve gerçek zamanlı bir robot için yeterli sürede tespiti de deney ortamında denenecek diğer önemli problemdir. Hedef koordinata daha hızlı ulaşılması ve yapılan hesaplamaların gerçek zamanlı olarak işlenmesi problemi için kamera sensörü ile entegre yol bulma algoritmalarının performansları belirli ölçütlere göre değerlendirilecektir.

1.2. Amaç

Problem bölümünde de belirtildiği gibi gezgin robotların otonom hareket edebilmesi için belirli temel işlevleri yerine getirmesi gerekir. İşlevler içinde tezin ortamı yorumlandığında, hareketini sürdürürken ortamın haritasını çıkarmak, yazılı olarak verilen işletme adını harita üzerinde bulmak ve en kısa yoldan bu işletmeye ulaşmak. Hedef işletmelerin konumlarını tespit etmek ve gezinirken keşfettiği işletmeleri kaydederek yeni hedefler için algoritmanın cevap verme süresini azaltmak hedeflenmektedir. Gezgin robotların otonom yeteneklerini geliştirmek için kritik olan görüntü algılama ve yol bulma süreçlerini optimize etmeyi hedeflemektedir.

1.3. Varsayımlar

Tez için hazırlanan deney ortamı simülasyon ortamı içerisinde tasarlanmıştır. Bu ortam ile belirlenen varsayımlardan ilki kullanılacak olan robotun verilen görevleri yerine getirebilecek sensör ve işlemci kapasitesi olduğudur. Ortam hazırlanırken, işletme tabelaları robotun algılayabileceği konum, boyut ve okunabilirlik standartlarına sahiptir. Robotun hareketine devam ederken yol üzerinde kalması için sınırlar yaya kaldırımları ile belirlenmiştir. Kaldırımın yüksekliği, robotun LiDAR sensörü tarafından algılanabilecek boyut ve uzaklıkta konumlandırılmıştır. Belirlenen ortamda caddeler ve sokak bitimleri belirli bir yükselti ile sınırlandırılmıştır. Bu sayede robot ortam içerisinde hareketine devam etmesi beklenmektedir. Algoritmaların performansı değerlendirilirken deney, aynı simülasyon ortamı ve aynı girdiler ile gerçekleştirilecektir.

1.4. Tezin Katkıları

Bu tez çalışmasında, hem metin tespiti hem de optimum yol bulma süreçlerini inceleyen bir simülasyon ortamı geliştirilmiş ve çeşitli algoritmaların performansları de-

taylı biçimde değerlendirilmiştir. Aşağıda, tez kapsamında gerçekleştirilen katkılar madde madde sıralanmaktadır:

- EAST yönteminin metin tespit etme görevinde simülasyon ortamında performansını analiz etmek.
- Simülasyon ortamında Üç farklı OCR algoritmasını metin tanıma görevinde performans karşılaştırılması yapmak.
- Gerçek zamanlı metin tanıma süreçlerinde CPU(Central Processing Unit) ve GPU (Graphical Processing Unit) kullanımı açısından OCR algoritmalarının performansını (FPS, tabela okuma süresi, Rastgele Erişimli Bellek (Random Access Memory-RAM) kullanımı, sanal bellek kullanımı ve CPU kullanımı gibi parametreler bakımından) detaylı bir şekilde karşılaştırmak.
- Kamera sensöründen elde edilen bilgilerle simülasyon ortamında oluşturulan ortamdaki market, mağaza gibi işletmelerin koordinatlarını belirlemek.
- İki farklı optimum yol bulma algoritmasının (A* ve Dijkstra) farklı parametreler açısından performansını karşılaştırmak.
- Kamera sensöründen elde edilen bilgilerle belirlenen koordinatların optimum yol bulma algoritmalarına etkisini analiz etmek.

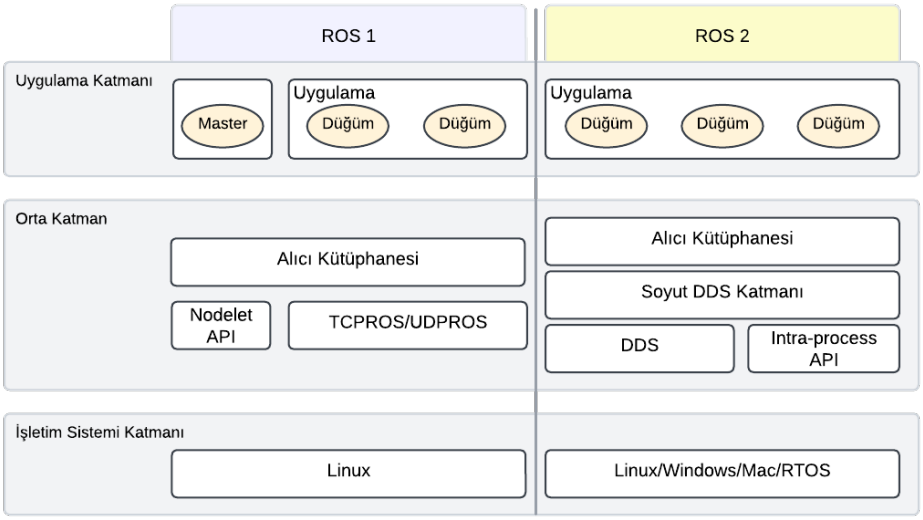
İKİNCİ BÖLÜM
MATERYAL VE YÖNTEM

2.1. Robot İşletim Sistem (ROS)

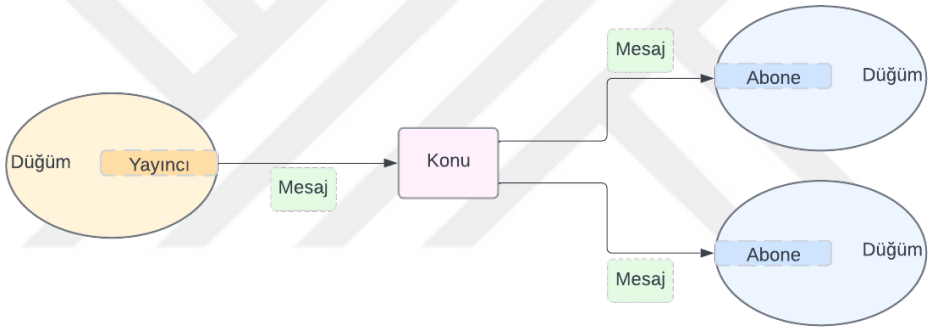
Robot İşletim Sistemi (ROS), robotik sistemleri için programlama süreçlerini kolaylaştıran, açık kaynaklı ara yazılım ya da yazılım çerçevesi olarak adlandırılabilir. Yazılım çerçevesi, yazılımın geliştirilmesinde kolaylık sağlayan, projeler ile alakalı kütüphaneleri barındıran ve yazılımın genel yapısını ve akışını belirleyen bir ortamdır (Yektek, 2024). Bahsedildiği gibi ROS bir işletim sistemi değildir. ROS, donanım soyutlama, düşük seviyeli aygıtlar için kontrol mekanizması sağlayan ve süreçler arası veri iletişimi gibi işlevleri yerine getiren bir yazılım çerçevesidir (Lee, Yoon, Jang, & Park, 2021).

Gelişen teknolojiyle beraber zamanla daha farklı isterler ortaya çıkmıştır. ROS'un kullanım sınırlarını genişletmek, ROS topluluğunun başta olmak üzere endüstrilerde ve araştırma alanlarındaki isterleri karşılamak ve bu yazılımın geliştirilmesini sağlamak amacıyla 2018 yılında Open Robotics tarafından ROS 2 (Robotics, 2018) yayımlanmıştır (Bonci, Gaudeni, Giannini, & Longhi, 2023).

ROS'un çalışma mantığı ele alındığında, ilk olarak düğümler ve veri (mesaj) iletim sisteminden bahsedilebilir. ROS 1 mimarisinden farklı olarak mesajlaşma sistemi değiştirilmiştir. ROS 1 için mesajların iletimi TCPROS/UDPROS ile sağlanır. Bu katman master adı verilen düğüme bağlı olarak iletişimi gerçekletirir. ROS 2 ise iletişimin sağlanması için Veri Dağıtma Servisi (DDS) katmanını kullanır (Maruyama, Kato, & Azumi, 2016). İki versiyon arasındaki fark Şekil 2.1'de verilmiştir. Mesaj iletimi uygulama katmanında incelendiğinde düğümler arasındaki iletişimi yayıncı-abone modeli baz alınarak gerçekleştirilir (Lee et al., 2021). Yayıncı düğüm bir konu üzerinden düzenli olarak veri yayını yapar. Abone düğüm ise bu konuyu dinleyerek veriyi kendi sürecine dahil edip kullanabilir. Yayıncı-abone modeli Şekil 2.2'de verilmiştir.



Şekil 2.1: ROS 1 ile ROS 2 mimari açıdan karşılaştırma



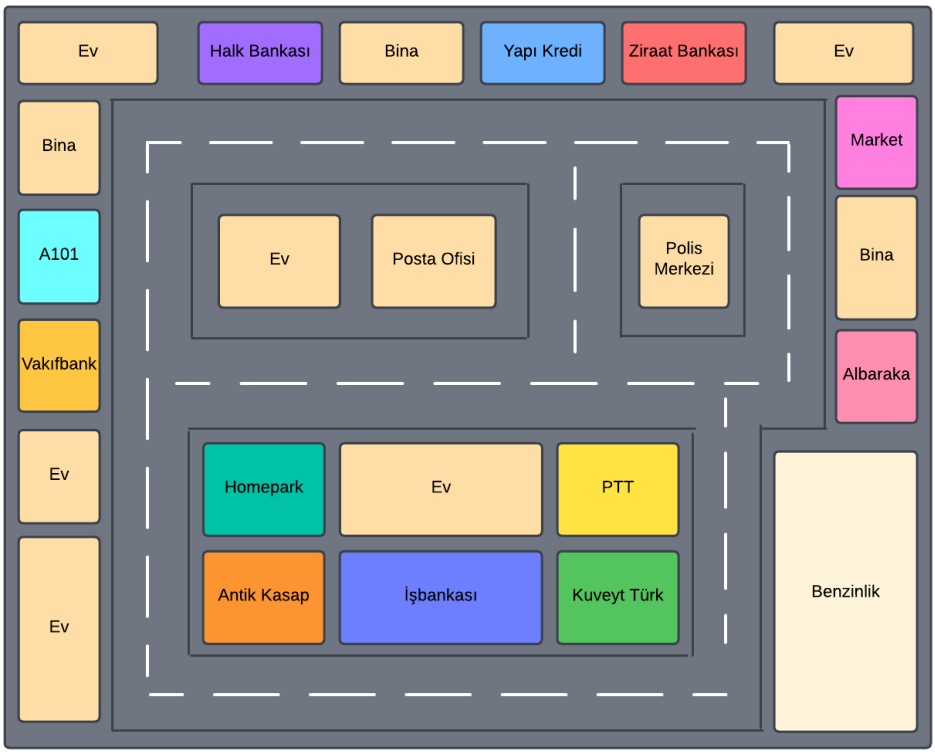
Şekil 2.2: Yayıncı-abone modeli

ROS'un için önemli olan diğer bir konsept parametreler ve parametre sunucusudur. Parametre sunucusu düğümler için robotun yapılandırma bilgilerini saklayan merkezi bir depo olarak düşünülebilir. Düğümler, çalışma zamanı içerisinde ilgili yapılandırma parametrelerini depolar ve alır. Amaç, yapılandırma değişikliklerinin tüm sistem tarafından görünür olmasıdır (Wiki, 2024).

Bu tez için çalışma ortamında ROS 2 Humble versiyonu ve algoritmaların geliştirilmesinde Python programlama dili kullanılmıştır.

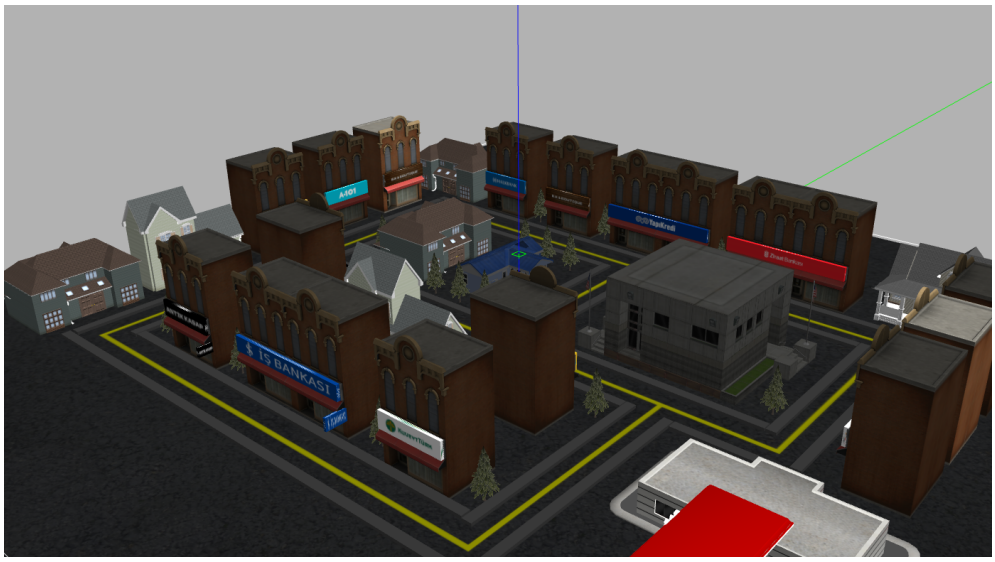
2.2. Gazebo Simülasyon Ortamı

Gazebo simülasyon ortamı, özellikle robotik sistemlerin gerçek dünyada uygulanmasının getirdiği zorluklar başta olmak üzere gerçekte denenmeden önce simüle edilerek eksiklerin ve sorunlar tespiti gibi isterlerin artması ile beraber geliştirilmiştir. Simülasyon ortamında istenilen modeller tasarlanabilir, hazır Gazebo tarafından sunulan modeller kullanılabilir, aydınlatma, yer çekimi, sürtünme katsayıları gibi değerler değiştirilerek istenilen ortamlar oluşturulabilir durumdadır (Koenig & Howard, 2004). Günümüzde Gazebo, Open Robotics tarafından bakımı yapılan ve açık kaynak kodlu olmasıyla beraber topluluk tarafından desteklenmektedir (Robotics, 2024a).

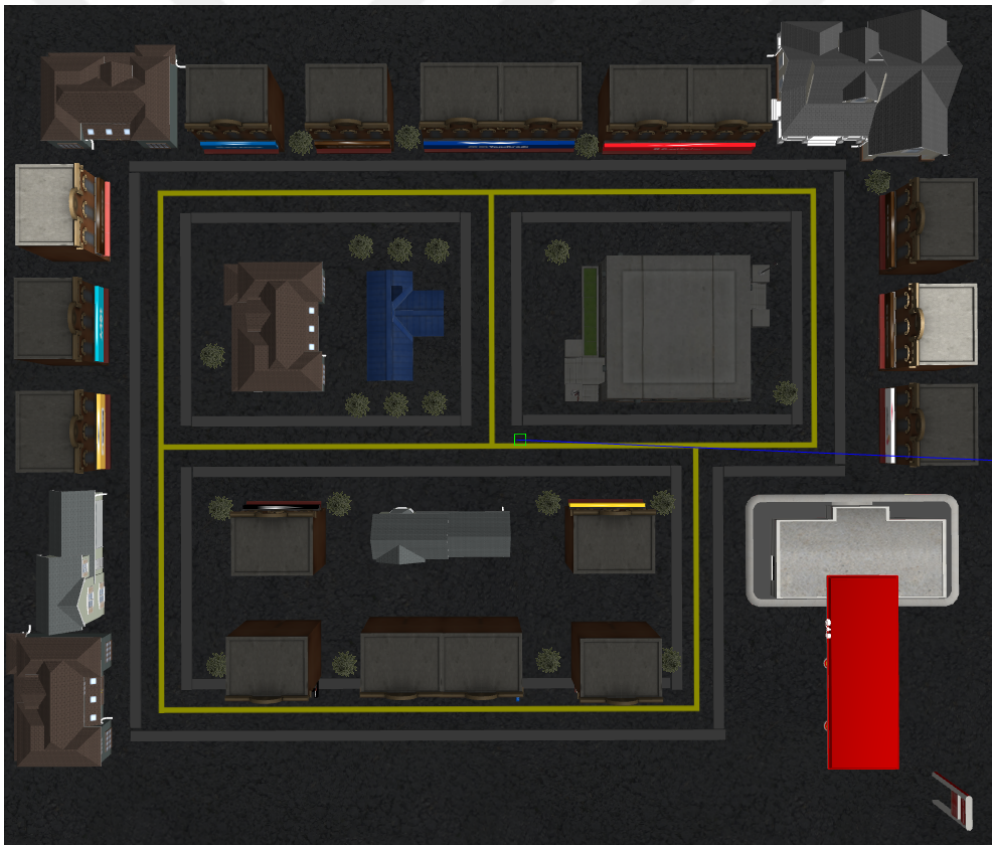


Şekil 2.3: Deney ortamının krokisi

Bu tez çalışmasında, simülasyon ortamı, küçük bir mahalle göz önüne alınarak çoğunluğu işletmelerden oluşan caddelerden oluşması planlanmıştır. Ortam krokisi Şekil 2.3'te verilmiştir. Sokakların genişliği LiDAR sensörünün mesafesi içerisinde olacak şekilde planlanmıştır. Kullanılan asfalt, ev, benzinlik ve ağaç modelleri Gazebo'nun resmi olarak yayınladığı (Robotics, 2024b)'dan kullanılmıştır. Ortamın görüntüleri Şekil 2.4 ve Şekil 2.5'te verilmiştir.



Şekil 2.4: Simulasyon ortamı



Şekil 2.5: Simulasyon ortamının kuş bakışı modeli

İşletme Modelleri

İşletmeler, dükkân modeli üzerine tabela kutuları bağlanarak oluşturulmuştur. Bağlanan tabelalara materyal tanımları eklenerek istenilen işletme adları ve logoları modellerin üzerine eklenmiştir. Eklenen materyal tanımı ve modelin SDF dosyasına eklenen Kod Bloğu 2.1'de verilmiştir. İki farklı tabela tipi kullanılmıştır. İlki dükkâna paralel ve yapışık olarak tasarlanmış ve daha büyük ölçekli olarak yerleştirilmiştir. İkinci tabela yaya kaldırımına bakan daha küçük ölçekli olarak tasarlanmıştır. Tasarlanan iki tabela Şekil 2.6'da verilmiştir.

Kod Bloğu 2.1: Model SDF kod bloğu

```
# /antik/materials/scripts
material antik/Diffuse
{
    receive_shadows off
    technique
    { pass
        { texture_unit
            { texture antik.png }
        }
    }
}
# model.sdf
<material>
    <lighting>1</lighting>
    <script>
        <uri>model://antik/materials/scripts</uri>
        <uri>model://antik/materials/textures</uri>
        <name>antik/Diffuse</name>
    </script>
    <shader type='pixel'/>
</material>
```

Şekil 2.6: İşletme tabela modelleri

2.3. RViz

RViz, ROS için geliştirilmiş bir görselleştirme aracıdır. Robotların sensör verilerini, haritalarını, planlarını ve diğer önemli bilgilerini gerçek zamanlı olarak görselleştirilip sunulmasını sağlar. Görselleştirme araçlarının kullanılması robotik sistemlerin durumunu anlamak, hataları tespit etmek ve robotların çevresiyle etkileşimlerini analiz etmek ve görsel olarak takip etmek açısından önemli bir yere sahiptir.

RViz'de oluşturulan ve takip edilen parametreler bir konfigürasyon dosyasına kaydedilerek her RViz açıldığında yeniden ortamın hazırlanmasına ihtiyaç kalmaz. Bu tez çalışmasında istenilen konuların seçildiği konfigürasyon dosyası, Kod Bloğu 2.2'de verilmiştir. Kod bloğundaki anahtar kelimeler kullanım sırasına göre açıklandığında, **Panels** arayüzdeki bölümler ve düzenini tanımlar. **Displays** içerisinde görselleştirilecek parametreler seçilebilir ve durumları gösterilir. **Selection** sahne içerisinde nesnelerin seçilmesini sağlar. **Tool Properties** bölümünde araçlar ile ilgili değişkenler düzenlenir. **Time** ile simülasyon ile zaman senkronizasyonu ve kaynakları kontrol edilebilir. **Visualization Manager** görselleştirme yöneticisidir. Bu anahtarın altında bulunan **Displays** bölümünde veri yayını yapan konulara abone olunarak verinin görselleştirilmesi sağlanır. Kullanılan konfigürasyon dosyasında **/camera/image_raw**, **/scan** ve **/map** konuları kullanılmıştır. Son olarak **Transformation** robotun koordinat dönüşümleri

ile sensörler arasındaki verileri işler.

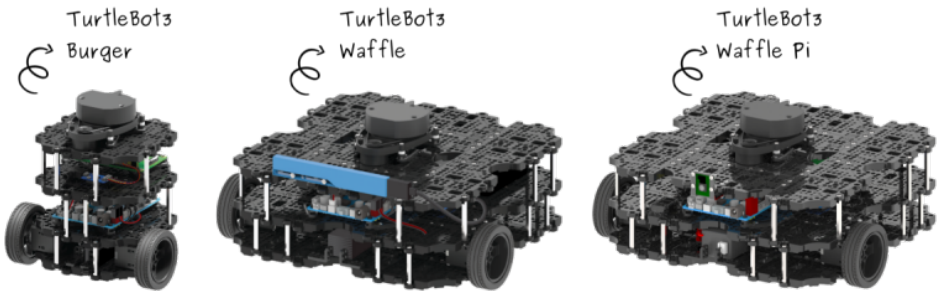
Kod Bloğu 2.2: RViz YAML konfigürasyonu

```
Panels:
  - Class: rviz_common/Displays
    Help Height: 209
    Name: Displays
    Property Tree Widget:
      Expanded: /Global Options1,/Status1,/LaserScan1,/Map1
      Splitter Ratio: 0.5
    Tree Height: 349
  - Class: rviz_common/Selection
    Name: Selection
  - Class: rviz_common/Tool Properties
    Expanded:/2D Goal Pose1, /Publish Point1
    Name: Tool Properties
    Splitter Ratio: 0.5886790156364441
  - Class: rviz_common/Views
    Expanded:/Current View1
    Name: Views
    Splitter Ratio: 0.5
  - Class: rviz_common/Time
    Experimental: false
    Name: Time
    SyncMode: 0
    SyncSource: LaserScan
Visualization Manager:
  Class: ""
  Displays:
    - Class: rviz_default_plugins/Camera
      Name: Camera
      Topic: /camera/image_raw
      ...
      Class: rviz_default_plugins/LaserScan
      Name: LaserScan
      Topic: /scan
      ...
      Class: rviz_default_plugins/Map
```

```
      Name: Map
      Topic: /map
      Update Topic: /map_updates
      ...
  Transformation:
    Current:
      Class: rviz_default_plugins/TF
```

2.4. Mobil Robot: Turtlebot3

Bu çalışmada, simülasyon ortamında oluşturulacak otonom robot olarak Turtlebot3 Waffle modeli seçilmiştir. Turtlebot3 hem simülasyon hem de gerçek hayat senaryolarında kullanılabilen bir mobil robottur (Niu et al., 2021). ROS tabanlı olması sebebiyle robotik sistemler üzerine olan çalışmalarda kolaylık sağlamaktadır. Robotis firması tarafından geliştirilen açık kaynak kodlu Turtlebot3, ROS 1 ve ROS 2'yi destekler. Çeşitli amaçlara hizmet etmesi için 3 temel model geliştirilmiştir. Modeller Şekil 2.7'de verilmiştir. Başlangıç seviyede uygulamalarda kullanılmak üzere daha küçük ve kompakt olarak geliştirilen Burger modeli, gelişmiş SLAM ve otonom navigasyon uygulamaları için geliştirilen daha büyük ve güçlü bir tasarımla daha fazla taşıma kapasitesi sunan Waffle modeli ve son olarak iki model arasında performans sunan Waffle Pi modeli geliştirilmiştir (Robotis, 2024). Tez kapsamında kullanılacak olan Waffle modelinin detaylı ozellikleri Tablo 2.1'de yer verilmiştir.

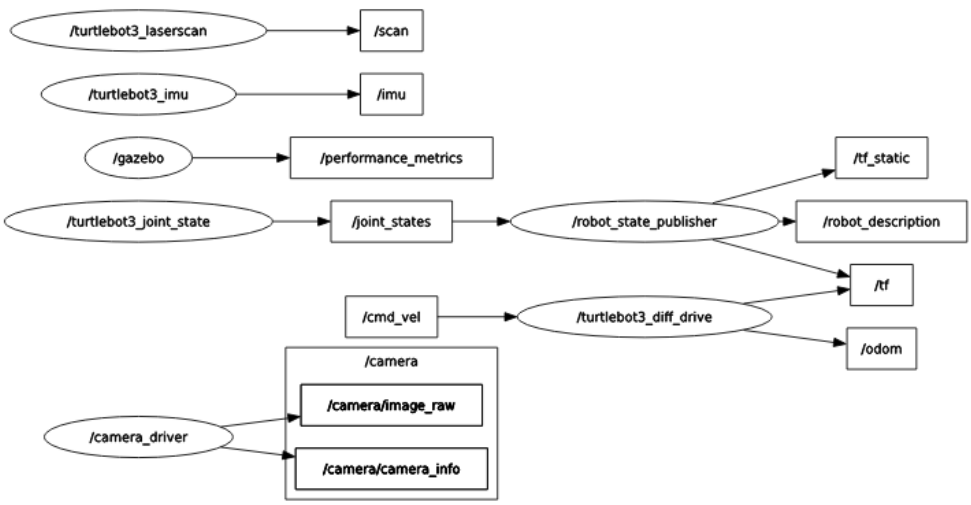


Şekil 2.7: Turtlebot3 örnek modeller

Tablo 2.1: TurtleBot3 Waffle teknik özellikleri

Özellik	**Değer**
Boyutlar	281 mm x 306 mm x 141 mm
Ağırlık	2.0 kg
İşlemci (SBC)	Intel Joule (veya eşdeğer)
Sensörler	LiDAR, IMU, çarpışma sensörleri
Batarya	11.1V 1800mAh LiPo
Kontrol Kartı	OpenCR
Maksimum Hız	0.26 m/s
ROS Desteği	ROS 1 ve ROS 2
Simülasyon	Gazebo, RViz

TurtleBot3 çevresel algılama ve hareket kabiliyetini sağlamak için birçok farklı konu yayınlamaktadır. Örneğin, **/camera/camera_info** konusu, robotun kamera sensörünün iç ve dış parametrelerini içererek, görsel algılamayı sağlar. Görüntü verileri, **/camera/image_raw** konusu aracılığıyla alınır. Hareket kontrolü ise, **/cmd_vel** konusu üzerinden yapılan doğrusal hız ve açısal hız komutları ile sağlanır. Robotun konum ve yönelim bilgileri, **/odom** konusu ile sağlanırken, çevresel engellerin mesafelerini ölçmek için **/scan** konusundan gelen LiDAR verileri kullanılır. **/imu** konusu ise robotun ivmeölçer ve jiroskop verileri için kullanılır ve robotun hareketini ve eğimini izler. Bu konulara ek olarak, robotun kinematik yapısını ve fiziksel modelini tanımlayan **/robot_description** konusu, robotun tasarımını ve yapısını açıklığa kavuşturur. Robotun iç durumu ve performansı, **/rosout** ve **/parameter_events** gibi konularla takip edilir. Robotun işlemci ve bellek kullanımı **/performance_metrics** konusu ile izlenir. **/tf** ve **/tf_static** konuları robotun sensörleri ve parçaları arasındaki dönüşüm bilgilerini sağlar, böylece robotun çeşitli bileşenlerinin dünya koordinatlarına göre konumlandırılması sağlanır. Son olarak, **/joint_states** konusu robotun hareketli parçalarının durumunu, her bir eklem için açı, hız ve kuvvet bilgilerini içererek robotun hareket kabiliyetini izler. Konuların birbiri ile bağlantısı Şekil 2.8'de verilmiştir.



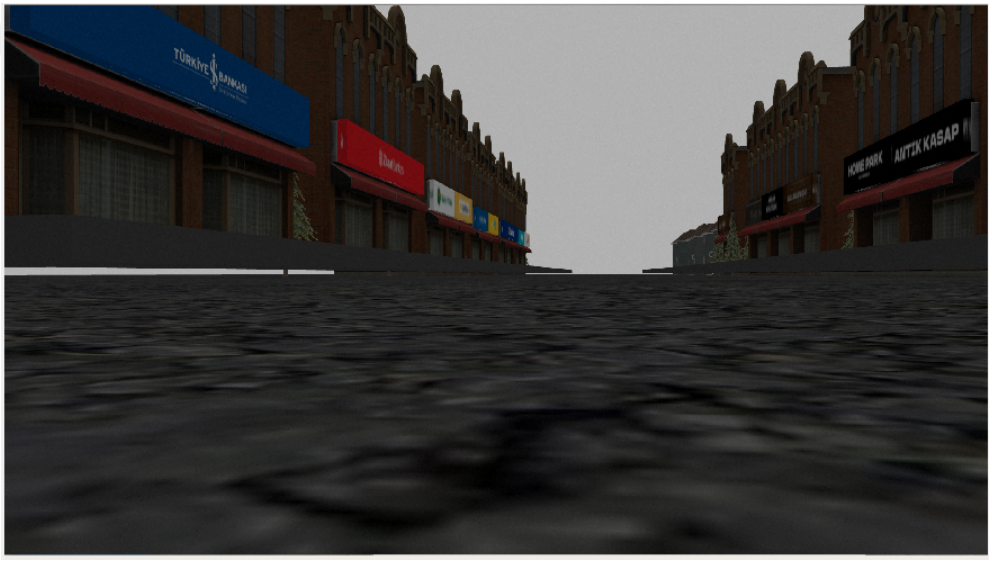
Şekil 2.8: Turtlebot3 Waffle konu grafı

Çevrenin algılanmasında iki ana sensor kullanılır. LiDAR sensörü, robotun çevresini lazer ışınları ile ölçerek etrafındaki engellerin taranmasını sağlar. Aynı zamanda engellerle beraber ortamın haritası çıkarılırken duvarların, kenarların ya da kaldırımlar gibi belirleyiciler kullanılarak haritanın sınırlarının bulunmasını sağlar. Diğer bir sensor ise kamera sensörüdür. Otonom bir robotun, başlıca etrafındaki nesnelerin ve çevrenin tanınmasında, hedef bir nesnenin saptanmasında kullanılır. Aynı şekilde bu tez çalışmasında LiDAR sensörü kaldırımların takibi için, kamera sensörü ise işletme tabelalarının bulunarak işletme adlarının tespiti için kullanılmaktadır. Kamera sensörünün görüş açısı 120 dereceye çıkartılmıştır. Güncellenen bölüm Kod Bloğu 2.3'te verilmiştir. Bu şekilde caddenin sağı ve solu taranabilir konuma gelmiştir. Kamera açısından caddenin görüntüsü Şekil 2.9'da verilmiştir.

Kod Bloğu 2.3: Robotun model.sdf dosyasındaki kamera açısı konfigürasyonu

```
<sensor name="camera" type="camera">
        <always_on>true</always_on>
        <visualize>true</visualize>
        <update_rate>30</update_rate>
        <camera name="intel_realsense_r200">
          <horizontal_fov>2.094</horizontal_fov> % Camera FOV
          <image>
            <width>1920</width>
            <height>1080</height>
            <format>R8G8B8</format>
```

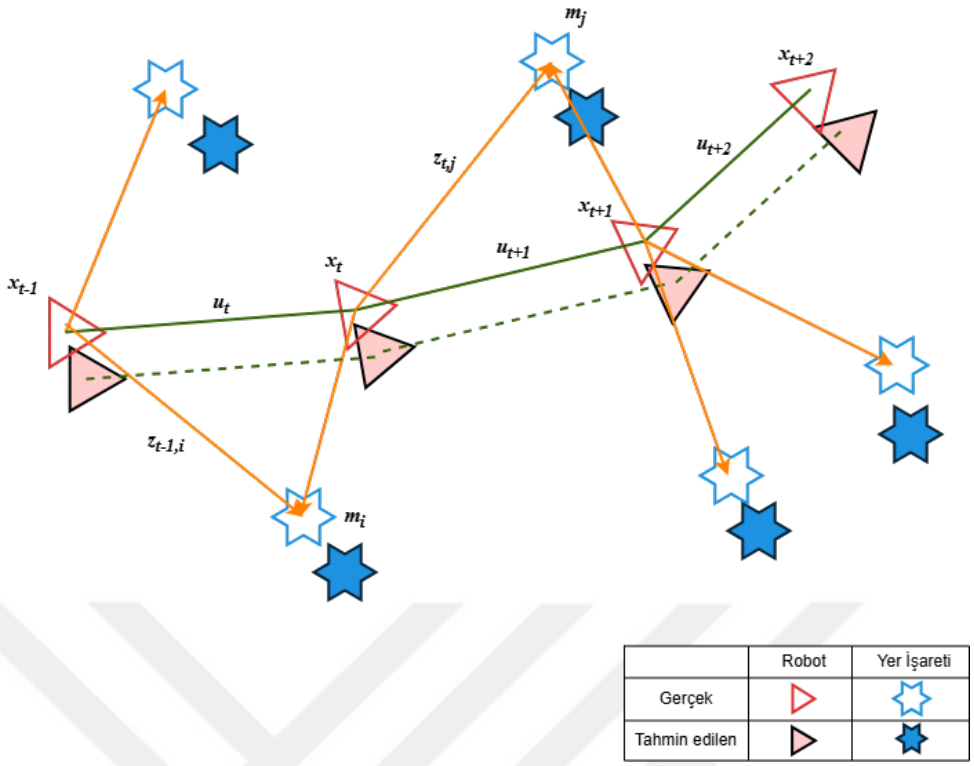
```
          </image>
          <clip>
            <near>0.02</near>
            <far>300</far>
          </clip>
          <noise>
            <type>gaussian</type>
            <mean>0.0</mean>
            <stddev>0.007</stddev>
          </noise>
        </camera>
```



Şekil 2.9: Turtlebot3'ün kamera açısı

2.5. Eş Zamanlı Konumlama ve Haritalama (SLAM)

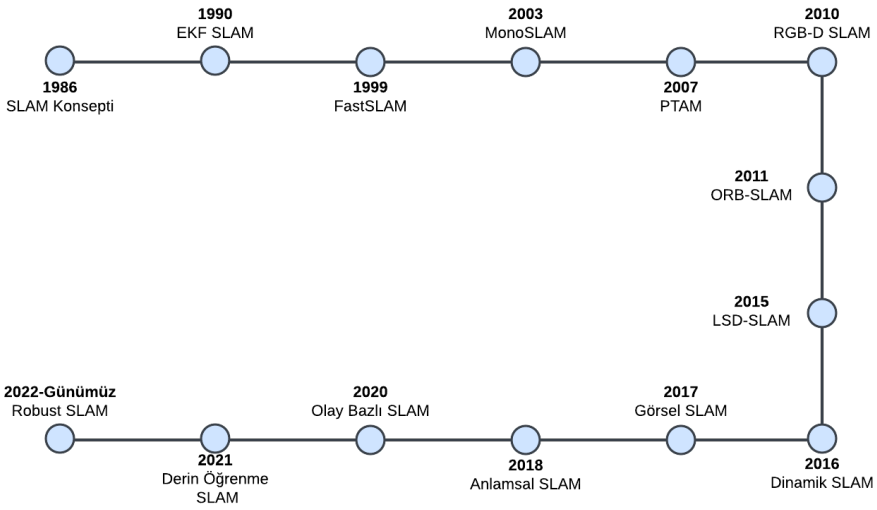
Bilinmeyen bir ortamda, otonom olarak hareket eden bir robotun konumunu tespit etmesi ve çıkan engelleri aşarak ortamın haritasını çıkarabilmesi popüler araştırma konularından biridir. Bu probleme çözüm olarak tasarlanan yöntem Eş zamanlı Konum Belirleme ve Haritalama (Simultaneous Localization and Mapping-SLAM)'dır (Büyükkelek, Dağadası, Yusefi, Türkmenoğlu, & Durdu, 2020).



Şekil 2.10: SLAM yönteminin çalışma prensibi

SLAM yöntemini kendi içinde üç kategoriye ayırabiliriz. Bayes filtre tabanlı algoritmalar, sensor verileri ve hareket tahminleri üzerinden robotun konumunu olasılıksal olarak tespit eder. Alt türleri olarak Kalman filtresi ve Parçacık filtresi verilebilir. İkinci olarak optimizasyon tabanlı yöntemlerde, robotun konumu hareket modeli ve sensör verilerinin matematiksel olarak optimize edilerek doğru pozisyonun bulunması hedeflenir. Graf tabanlı SLAM ve Demet Düzeltimi (Bundle Adjustment) bu kategoriye örnek olarak verilebilir. Son olarak sensörlerin kullanımına göre SLAM LiDAR tabanlı, Görüntü tabanlı ve Radar tabanlı olarak alt kategorilere ayrılabilir (Sankalprajan, Pendyala, Perur, & Pagala, 2020).

1980'lerin ortasında SLAM konsepti olarak ilk çalışma yayımlanmıştır (Smith, Self, & Cheeseman, 1986). SLAM Şekil 2.11'de ki versiyonlarından görüldüğü gibi zamanla değiştirilerek ve geliştirilerek günümüzdeki versiyonuna ulaşmıştır (Stewart & Church, 2024).



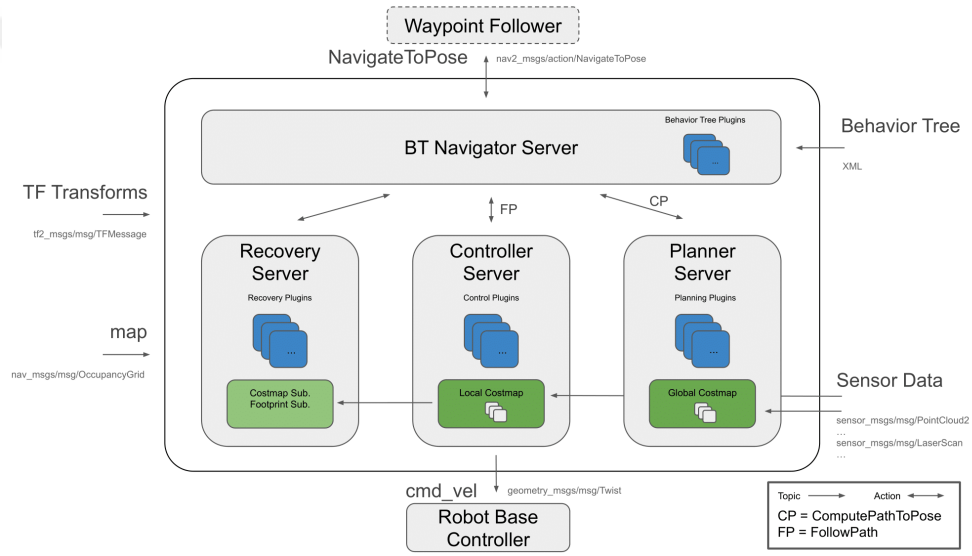
Şekil 2.11: SLAM yönteminin evrimi
Kaynak: Stewart and Church,2024

2.6. Yol Planlama

Mobil robotların bir ortamda üç işlevi yerine getirmesi beklenir. Bunlar robotun kendi konumunu saptaması, ortamın haritasını oluşturması ve hedefe gideceği yolu planlamasıdır (L. Liu et al., 2023). Hedefe doğru giderken engellerden kaçarak yol güzergahını güncelleyen planlama yöntemine yerel (lokal) yol planlama denir. Bu yöntemin daha büyük ölçekli ortamlarda yapılması ise küresel (global) yol planlama olarak adlandırılır. İki planlama türü içinde yıllar boyu çeşitli algoritmalar ve yöntemler geliştirilmiştir. Bu tezde genellikle tercih edilen A* ve Dijkstra algoritmaları ile ROS 2 ortamı için yol planlama algoritmaları, SLAM uygulamaları ve robotun ortam üzerindeki hareketlerinin kontrolünü sağlayan Navigasyon 2 (nav2) (ROS2-Workshop, 2024) ve nav2-navfn-planner (Index, 2024) paketleri kullanılmıştır.

Navigation2, ROS Navigasyon Yığınınının profesyonel olarak desteklenen ROS2 için geliştirilen devam versiyonudur. Ancak Nav2 mimarisi değiştirilerek modern teknolojilere uygun olarak geliştirilmiştir (ROS2-Workshop, 2024). Mimarinin temel yapı taşı yapılan işlevlerin kontrollerini ve yürütme işini sağlayan Davranış Ağaçları (Behavior Tree)'dır. Küresel Planlama (Global) ve Yerel Planlama (Local) robotun hedefe doğru ilerleyebilmesi için yol planlamasını sağlar. Yerel planlama, daha küçük alanlar için mevcut konuma göre planlama sunarken küresel planlama hedefe kadar olan yolun

planlanmasını sağlar. Robotun fiziksel hareketlenin kontrolü Hız Kontrol Sunucusu (Controller Server) tarafından yapılır. Yerel plan doğrultusunda doğrusal ve açısal hız hesaplamaları yapılır. Maliyet Haritası (Costmap) ortamın hem küresel ve yerel planlarına göre alanın maliyetini hesaplar. Statik ortam haritalarının yüklenerek kullanılmasını Harita Sunucusu (Map Server) sağlar. Adaptiv Monte Carlo Konumlandırması (AMCL) robotun harita üzerindeki konumunu belirlemek için kullanılır. Hareket sırasındaki başarısız durumlardan kaçınma senaryolarını Kurtarma Sunucusu (Recovery Server) üstlenir. Belirlenen yolun takip işi Konum Nokta Takipçisi (Waypoint Follower) tarafından kontrol edilir (Hansen, 2019). Bileşenlerin birbiri ile bağlantısı Şekil 2.12'de verilmiştir.



Şekil 2.12: Navigasyon 2 mimarisi
Kaynak: ROS2-Workshop,2024

2.6.1. Dijkstra Algoritması

Bilgisayar bilimci Edsger Dijkstra tarafından 1959 yılında yayınlanan en kısa yol bulma algoritmasıdır. Algoritma ağırlık ve yönlü graflar için başlangıç ve hedef düğümleri arasındaki yolun bulunması için geliştirilmiştir (Özdemir, Sacar, & Evrencan, 2021). Çalışma prensibi olarak açgözlü algoritmalar gibi çalışmaktadır. Her bir düğümden gidilebilecek düğümler arasındaki mesafe hesaplanarak en kısa yol seçilir ve ilerleme sağlanır(Zhou et al., 2023).

Dijkstra algoritması zamanla çeşitli amaçlar için değişitirilerek kullanılmış olsa da geleneksel olarak temel mantığı aynıdır. Graf üzerinde bir başlangıç düğümü seçilir. Her düğüm gezildiğinde hedef düğüm olup olmadığı kontrol edilir. Seçili düğüme komşu olan düğümler arasındaki mesafe ölçülerek en kısa yola sahip komşu düğüm ziyaret edilecek bir sonraki düğüm olarak seçilir. Ziyaret edilen ve hedef olamayan düğümler bir listede, daha ziyaret edilmemiş düğümler bir listede toplanarak tekrarın önüne geçilir. Adımların tamamı hedef düğüm bulunana kadar devam eder (DIJKSTRA, 1959). Algoritma 1'de Dijkstra algoritmasına ait sözde (pseudo) koduna yer verilmiştir.

Algoritma 1 Dijkstra Algoritması

Data: Harita, başlangıç noktası S, hedef noktası G
Result: En kısa yol SP

```
Init: openlist[], closedlist[]
while openlist[] boş değil do
    Dugum ← openlist[] içinde en kısa mesafeye sahip nokta
    Dugum'ı openlist[]'ten closedlist[]'e taşı
    if Dugum = G then
        Yolu çıkar ve dur
    else
        komşu düğümler üzerinden tekrarla
        for Her komşu düğüm do
            if (düğüm closedlist[] içinde veya bir engel ile karşılaşıldıysa) then
                Bu noktayı atla
            else
                Yolu kümülatif olarak hesapla  if düğüm openlist[] içinde ise then
                    if (Kümülatif mesafe < Önceden kayıtlı mesafe) then
                        Veriyi güncelle
                    else
                        Düğümü openlist[]'e ekle
        openlist[] içindeki düğümleri küçükten büyüğe sırala
```

Temel mantığından yola çıkılarak, robotik sistemler için bir ortam üzerinde önüne çıkan engellerden kaçınarak, hedefe giden en kısa yolu düğümler yerine harita koordinatlarını kullanarak hesaplar.

2.6.2. A* Algoritması

A* algoritması, sezgisel arama algoritmalarından biridir. Genel olarak küresel (global) yol planlamasında kullanılır. Hedef düğüm bulunana kadar geniş çaplı olarak düğüm başına tahmini ve gerçek maliyetleri hesaplar. Ek olarak yol planlama aşamasında bir rolü olmayan düğümler kapalı listede tutularak arama yolları kısaltılır (Tang, Tang, Claramunt, Hu, & Zhou, 2021). Bu yöntem ile algoritma hız performansı artı yönde etkilenir. A* algoritmasının zaman karmaşıklığı $O(E + N \log N)$'dir.

Tahmini maliyetlerin hesaplanmasında maliyet hesaplama formülü kullanılır. Fonksiyon açıklamaları Denklem 2.1'de verilmiştir (X. Liu & Gong, 2011).

$$f(n) = g(n) + h(n) \tag{2.1}$$

$f(n)$: Toplam tahmini mesafe
$g(n)$: n düğümüne gelene kadar hesaplanan gerçek mesafe
$h(n)$: n düğümünden hedef düğüme olan tahmini mesafe

Tahmini fonksiyon ($h(n)$), algoritmaya sezgisellik katmakla beraber hesaplanırken birbirinden farklı yöntemler tercih edilebilir. Genellikle Manhattan Denklem 2.2'de ve Öklid Denklem 2.3'te verilen mesafe denklemleri kullanılır. Tahmini fonksiyonun etkili olması istenmediği durumlarda 0 ya da 1 alınarak kullanılır.

$$|x_1 - x_2| + |y_1 - y_2| \tag{2.2}$$

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2.3}$$

Algoritmanın işleyişi ilk olarak başlangıç düğümün seçilmesi ve başlangıç düğümle beraber bütün komşu düğümlerinin tahmini maliyet hesabının ($f(n)$) yapılmasıyla başlar. Açık ve kapalı olmak üzere iki liste oluşturur. Açık olan ziyaret edilecek ve yol maliyeti hesaplanmamış düğümler için, kapalı olan ise ziyaret edilen ve hesaplamaları tamamlanmış düğümler için tutulur. Her bir komşu düğüm için; $f(n)$ değeri mevcut düğümden düşük değilse ve kapalı listedeyse komşu düğüm atlanır. Eğer daha iyi bir $f(n)$

değeri varsa bu değer güncellenir. Açık liste boş duruma gelene kadar veya hedef düğüm bulunana kadar algoritma tekrarlanır. Algoritma 2'de A* algoritmasına ait sözde (pseudo) koduna yer verilmiştir.

Algoritma 2 A* Algoritması

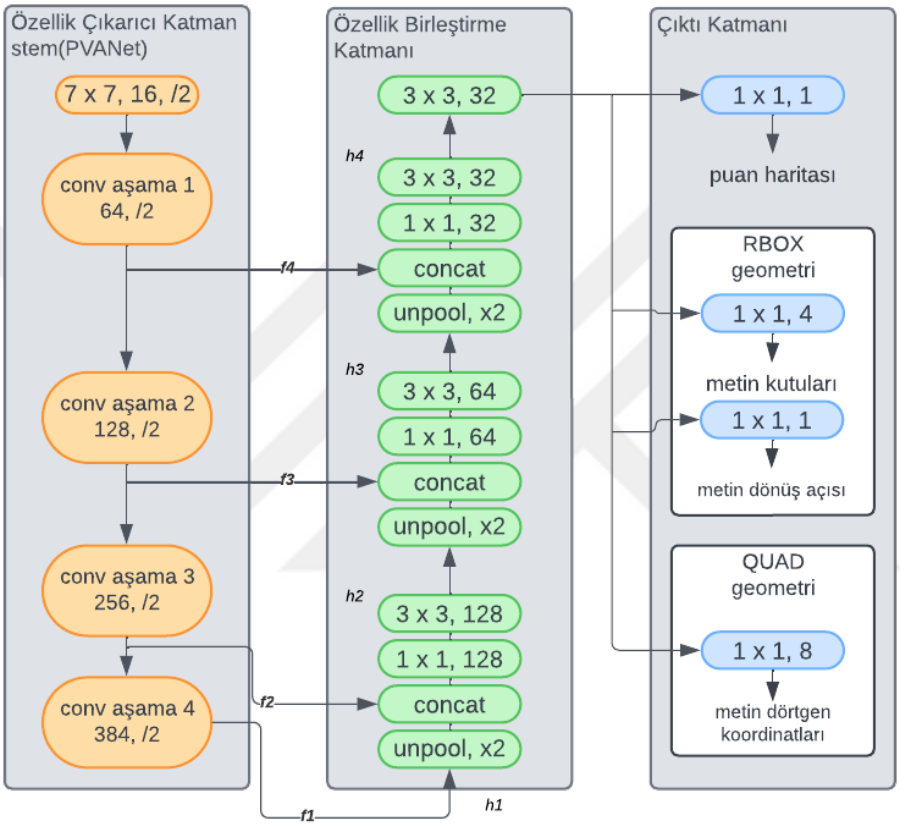
Data: Açık liste ($openlist$), kapalı liste ($closedlist$), başlangıç noktası (q)
Result: Başlangıçtan hedefe en kısa yol

```
openlist ← {}, closedlist ← {} Başlangıç noktasını openlist'e ekle, f değerini 0 olarak ayarla
while openlist boş değil do
    q ← openlist içindeki f değeri en küçük olan düğüm
    q'yu openlist'ten çıkar
    q'nun komşularını oluştur ve her birine q'yu ebeveyn olarak ayarla
    for Her komşu (successor) do
        if successor hedef noktasıysa then
            Aramayı durdur, yolu çıkart
        else
            successor.g ← q.g + mesafe(q, successor)
            successor.h ← mesafe(hedef, successor)
            successor.f ← successor.g + successor.h
            if successor openlist içinde ve aynı pozisyondaysa then
                if successor.f daha düşük değilse then
                    Bu komşuyu atla
            if successor closedlist içinde ve aynı pozisyondaysa then
                if successor.f daha düşük değilse then
                    Bu komşuyu atla
            Aksi takdirde successor'ı openlist'e ekle
    q'yu closedlist'e ekle
```

2.7. EAST Algoritması ve Maksimum Olmayanı Bastırma Yöntemi

Etkili ve Doğru Sahne Metin Dedektörü (Efficient Accurate Scene Text Detector-EAST), bir görüntüden metin olan bölgenin tespiti için kullanılan yapay zekâ teknikleri ile geliştirilmiş bir sinir ağı modelidir. Diğer kullanılan yöntemlerdeki ara katmanları kaldırarak direkt eşik değere göre sınıflandırma ve maksimum olmayanı bastırma (Non-Maximum Suppression-NMS) adımlarına geçiş sağlar (Zhou et al., 2017). Bu şekilde daha hızlı ve etkili sonuçlar elde edilir.

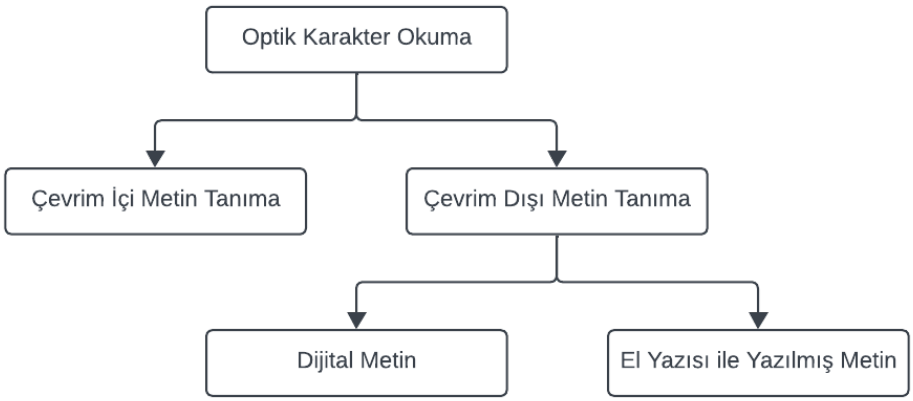
Basitleştirilmiş işleyişi ele alındığında ilk olarak karakteristik olarak çok ölçekli görüntüler evrişimli ağ kullanılarak elde edilir. Ardından karakter haritası oluşturulur ve metnin bulunduğu bölgenin konumu tahmin edilir. Doğru tahmini elde edebilmek için NMS algoritmasıyla daha düşük tahmin puanı olan ya da tekrarlayan tahminler elenir. Adımlar uygulandıktan sonra doğru sayılan bölgesi çıktı olarak elde edilir (Sun et al., 2022). EAST algoritmasının yapısı Şekil 2.13'te verilmiştir.



Şekil 2.13: EAST algoritma yapısı
Kaynak: Zhou et al.,2017

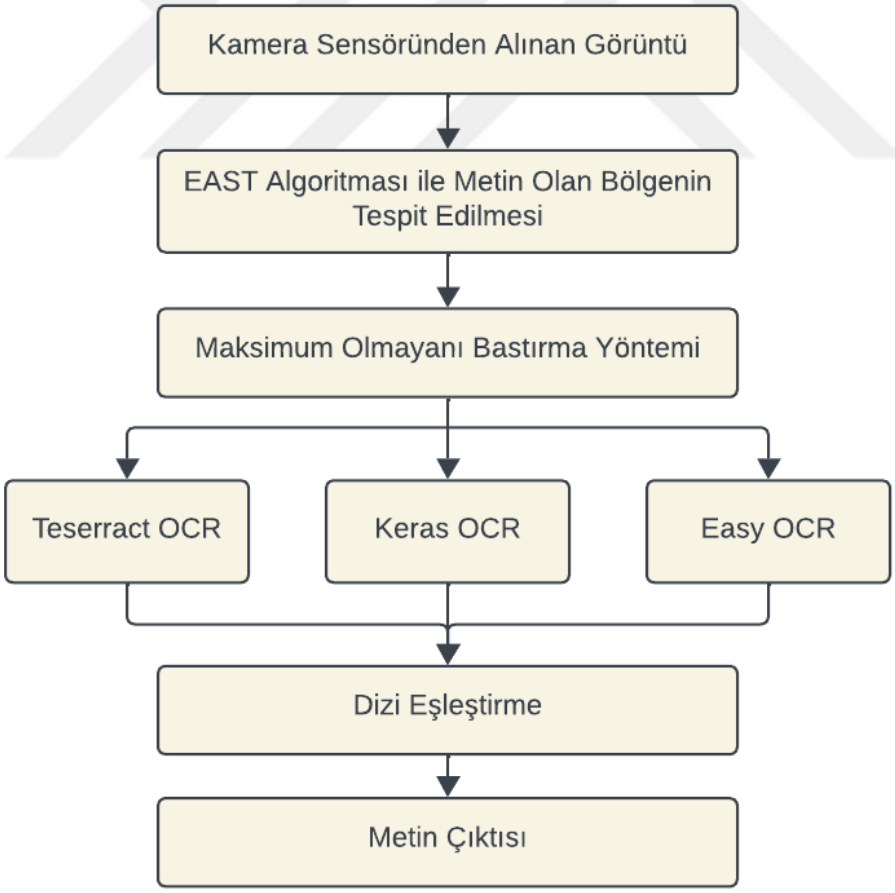
2.8. Optik Karakter Tanıma

Optik karakter tanıma (Optical Character Recognition-OCR), birbirinden farklı görüntü kaynaklarından metin karakterlerinin tanınmasının sağlayan teknolojidir. Kaynak olarak dijital görüntüler, videolar, yayınlar ve taranmış belgeler olabilir (Faizullah, Ayub, Hussain, & Khan, 2023). OCR iki kategoride sınıflandırılır. Bunlar çevrimiçi metin tanıma ve çevrimdışı metin tanımadır. Alt başlıkları ile beraber Şekil 2.14'te yer verilmiştir (Sonkusare & Sahu, 2016).



Şekil 2.14: OCR kategorileri

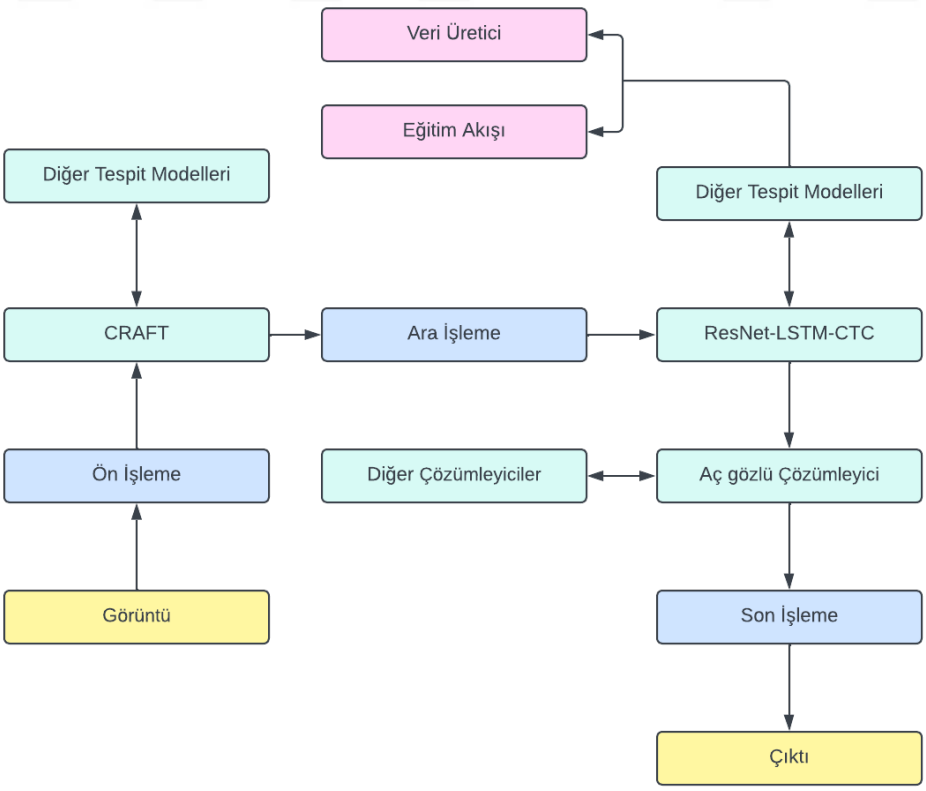
OCR için izlenecek adımlar her bir problem ve her bir ortam için değişiklik gösterebilir. Bu çalışmada sırasıyla Şekil 2.15'te yer verilmiştir.



Şekil 2.15: Çalışmada uygulanan OCR adımları

2.8.1. EasyOCR Modeli

Jaiden AI tarafından geliştirilen EasyOCR (AI, 2020), görüntü üzerinde metnin yerinin bulunarak karakter tanıma için kullanılan bir kütüphanedir. Metnin yerini bulmak için Metin Algılama için Karakter-Bölge Farkındalığı (Character-Region Awareness for Text detection-CRAFT) algoritmasını kullanır. Karakter tanıma için Evrişimsel Yinelenen Sinir Ağı (Convolutional Recurrent Neural Network-CRNN) modelini kullanır. Bu modelin için üç ana işlevi vardır. İlk olarak özellik çıkarma olarak ResNet derin ögrenme modelini kullanır (Mei, Wang, & Chen, 2024). Dizi etiketleme, bulunan özelliklerini eşleştirilerek anlamlandırılması için Uzun Kısa Süreli Bellek (Long Short Term Memory-LSTM) ağlarını kullanır (Pham, Lam, Duy, Bao, & Trinh, 2024). Son olarak çözümleme işlevinde, etiketlenmiş diziler Bağlantısal Zamansal Sınıflandırma (Connectionist Temporal Classification-CTC) algoritmasıyla gerçek metne çevrilir (Hou, Kong, Wang, & Li, 2018). Görüntü içindeki metnin tanınması için uygulanan akış Şekil 2.16'da verilmiştir. Türkçe tabelaların okunmasında dil desteği model seçimlerinde önemli kritere sahiptir. Bununla beraber EasyOCR 80 dili destekleyerek bu kritere uyum sağlar. Python ve Pytorch ile geliştirilmiştir.

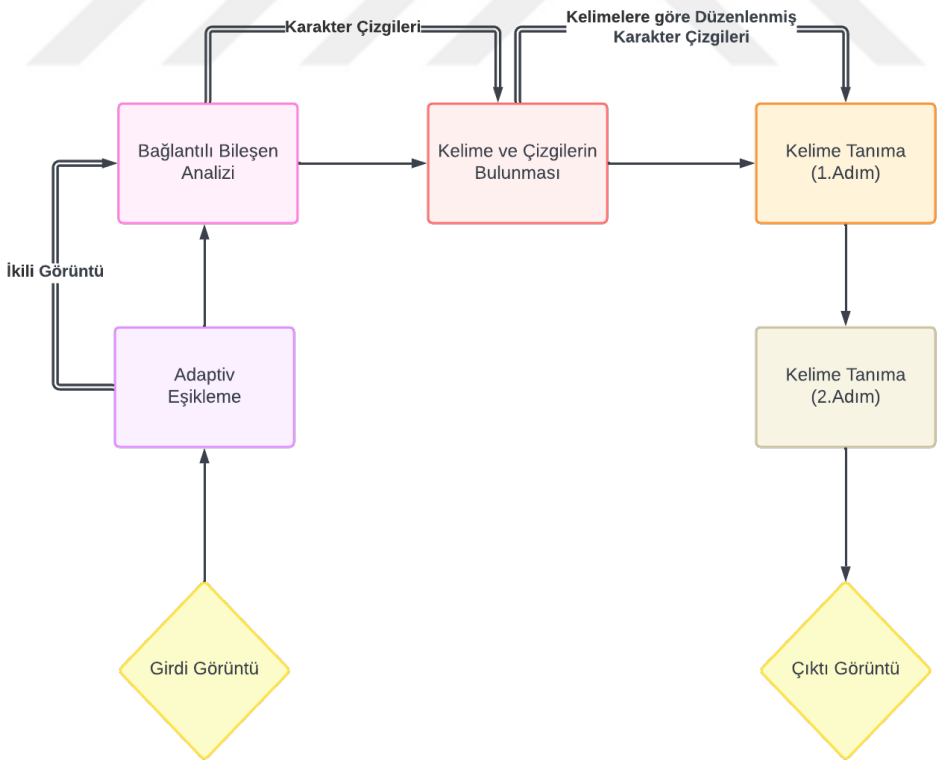


Şekil 2.16: Easy OCR akış yapısı

2.8.2. Tesseract OCR Modeli

1984 ile 1994 arasında HP tarafından geliştirilen açık kaynak olarak sunulan Tesseract bir OCR modelidir (Smith, 2007). Günümüzde Tesseract, Google tarafından geliştirilmektedir (Patel, Patel, & Patel, 2012). Zamanla model ilk çıktığı haliyle kalmamış ve değiştirilmiştir. Bu değişiklikler arasında göze çarpan önemli değişiklik Uzun Kısa Süreli Bellek (LSTM) sinir ağının eklenmesidir (Sporici, Cusnir, & Boiangiu, 2020).

Model mimarisi ele alındığında, ilk adımda adaptiv eşikleme işlevi kullanılarak görüntü ikili (binary) görüntüye dönüştürülür. Oluşturulan görüntüden bağlantılı bileşen analizi ile karakterlerin dış kenarları saptanır. Dış hatlardan karakterlerin bulunabilmesi için karakterler ayrılarak dış hatları yeniden düzenlenir. Son adım olarak 2 aşamalı kelime tanıma işlevleri uygulanır. İlk aşamada metindeki her bir kelime bulunur. Kelimeler eğitim verisi olmak için adaptiv sınıflandırıcıya aktarılır. Sınıflandırıcı kelime verilerinden ögrendikleri ile metni daha doğru tanınmasını sağlar (Smith, 2007). Mimari akışı Şekil 2.17'de verilmiştir.

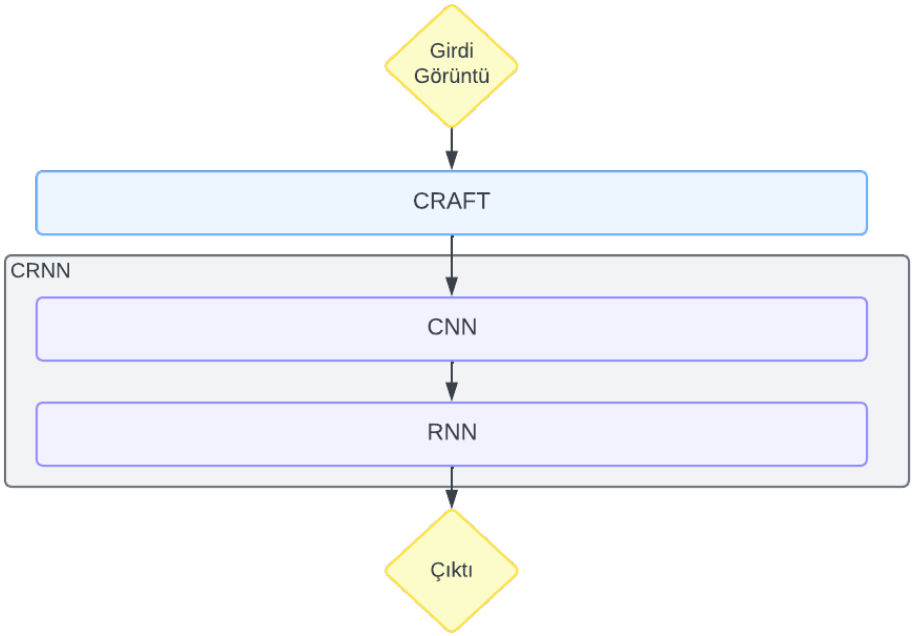


Şekil 2.17: Tesseract OCR akış yapısı

2.8.3. Keras OCR Modeli

Keras-OCR kütüphanesi, Keras Evrişimsel Yinelenen Sinir Ağı (CRNN) ile Metin Algılama için Karakter-Bölge Farkındalığı (CRAFT) algoritmasının paketlenmiş halidir (Clovaai, 2019). CRNN modeli temel olarak Evrişimsel Sinir Ağı (Convolutional Neural Network-CNN) ve Yinelenen Sinir Ağı (Recurrent Neural Network-RNN) katmanlarının birleştirilmesi ile oluşturulmuştur. Keras CRNN için iki farklı model kullanmaktadır. İlk model aşama bakımından orijinal CRNN'nin aşamalarını kullanırken ikinci model ek olarak bir katman daha içerir. Bu katman metni düzenlemek için kullanılan Uzamsal Dönüştürücü Ağ (Spatial Transformer Network) katmanıdır.

Çalışma akışı ele alındığında görüntü CRAFT algoritmasına verilerek metin olan bölgelerin bulunması sağlanır. Çıktı olarak alınan metin bölgesi CRNN modeline girdi olarak verilir. CNN katmanları ilk olarak görüntü üzerindeki özellikleri çıkartarak genel yapıyı anlamlandırır. Ardından RNN katmanları, karakterleri ve kelimeleri tanınmasını sağlar, bağlantılarını çıkarır (Jain et al., 2023). Keras OCR modeline ait genel akış Şekil 2.18'de verilmiştir.



Şekil 2.18: Keras OCR akış yapısı

2.8.4. Dizi Eşleştirme

Dizi Eşleştirme (Sequence Matcher) Python "difflib" kütüphanesinde bir sınıftır (Difflib, 2023). Verilen iki girdi karşılaştırarak benzerliklerini tespit etmek için kullanılır. Girdiler arasında bir benzerlik skoru hesaplar. Bu hesaplama Gestalt Örüntü Eşleme (Ratcliff, Metzener, et al., 1988) algoritmasındaki skorun hesaplanmasında Denklem 2.4'te bulunan formül kullanılır.

$$D_{ro} = \frac{2 * K_m}{|S1||S2|} \tag{2.4}$$

K_m : Dizideki eş karakterlerin sayısı

$|S1|$: İlk girdinin uzunluğu

$|S2|$: İkinci girdinin uzunluğu

Dizi Eşleştirme, tabelalardan okunan metinlerin önceden hazırlanan etiket listesiyle eşleştirilerek OCR tarafından okunamayan karakterler yüzünden tamamlanamayan işletme adlarının tamamlaması için kullanılır.

2.9. Entegrasyon Yazılımı

Entegrasyon yazılımı görevlere göre düzenlenmiştir. Robotun ilk görevi bilinmeyen ortamın keşfedilerek işletmelerin kaydedilmesidir. Ortamın keşfi için duvar takibi yöntemi kullanılmıştır. Bu yöntemde engel ile karşılaşan robota açısal hız verilerek robotun dönmesi sağlanır. Eğer LiDAR sensöründen alınan veri de robotun önünde herhangi bir engel yoksa açısal hız kesilerek doğrusal hız verilir ve robotun ilerlemesi sağlanır. Bu algoritma ile ortam içerisinde caddeler ve sokaklar gezilirken alınan kamera görüntüleri metin tespiti için sıralı işlemlere takip edilerek metin elde edilir. Metin elde edildiğinde, robotun konumu ve kaldırım uzaklığı hesaba katılarak işletmelerin konumları bulunur ve kaydedilir. Aynı zamanda tanınan metinler Kod Bloğu 2.4'te ki gibi ayrı bir YAML dosyasında tutularak daha sonra ki karşılaştırmalar için kullanılır. Metin tespiti için East yöntemi, belirlenen OCR modelleri ve dizi eşleştirme kullanılır. Algoritmanın akış şeması Şekil 2.20'de verilmiştir. Ortam haritası tamamlandıktan ve her bir cadde gezildikten sonra algoritma durdurulur.

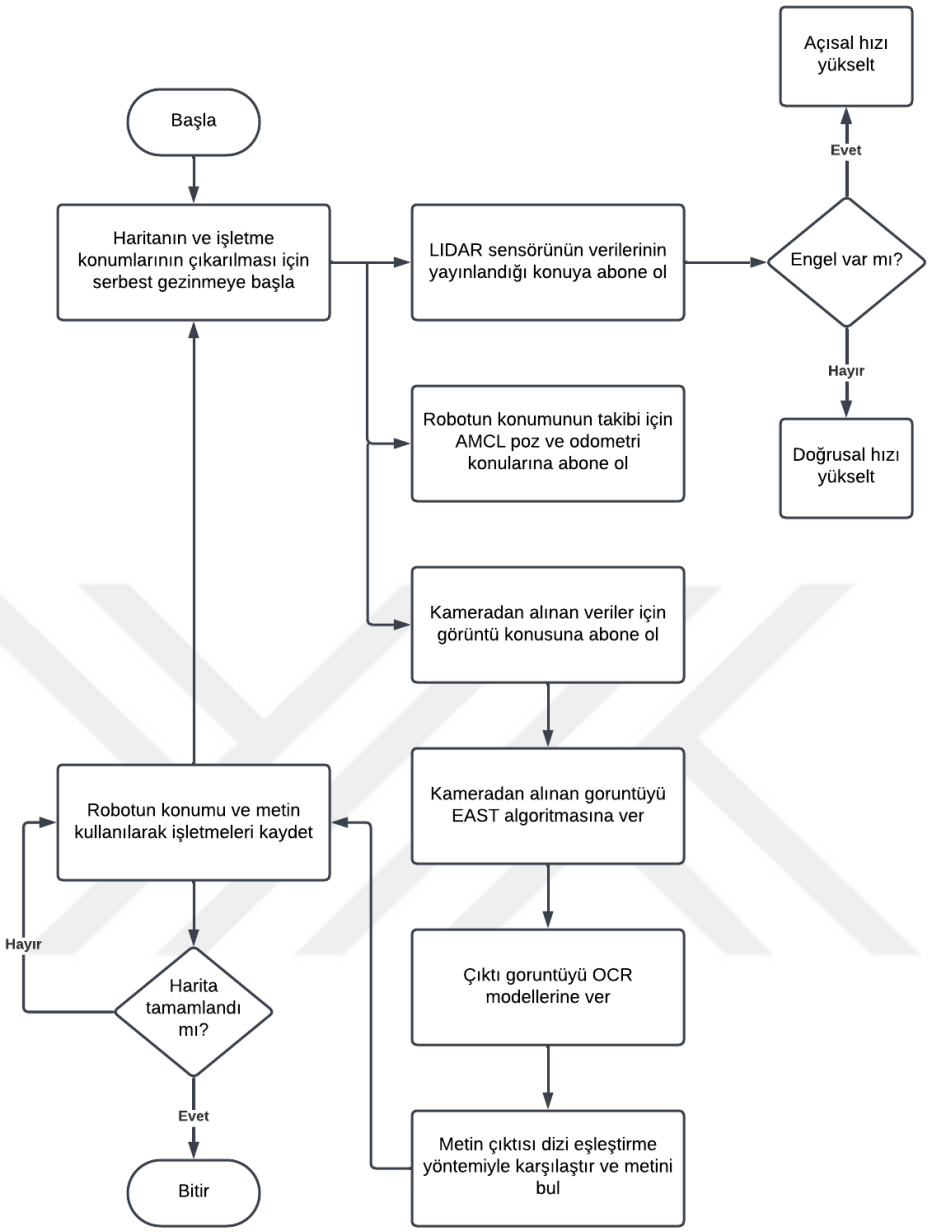
Kod Bloğu 2.4: Metinlerin tutulduğu YAML formatı

```
[
    {
        "robot2CameraLabel": "Ziraat"
    },
]
```

Şekil 2.19: EAST yöntemi ile metin olan bölgenin tespiti

OCR modelleri için tabela okuma sürelinin karşılaştırılması keşif algoritmasıyla beraber test edilerek sonuçlar kaydedilmiştir.

Harita ve konum bilgileri çıkarıldıktan sonra robota verilen isterlere göre Dijkstra ve A* algoritmalarının kullanılarak bilinen işletmelerin hedeflerine en kısa yoldan ulaşılır veya bilinmeyen işletmeler için kamera verilerinin gerçek zamanlı OCR modellerinde işlenerek hedef bulunur. Hedefe gidilirken daha önceden tanınmayan bir işletmeye denk gelinirse işletme kaydedilir.



Şekil 2.20: Gezinme algoritması akış şeması

ÜÇÜNCÜ BÖLÜM
DENEYSEL ÇALIŞMALAR

Bu bölümde gerçekleştirilen deneylerden ve toplanan sonuçlardan bahsedilecektir. Deneyler, 16 GB RAM, Nvidia 1660TI ekran kartı ve Ryzen 9 4000 Series 8 çekirdek işlemcisi ile belirtilen donanımlara sahip dizüstü bilgisayarda gerçekleştirilmiştir.

Belirlenen simülasyon ortamında robot modeli belirli bir noktada oluşturulmuştur. Projenin başlatma dosyası çalıştırıldıktan sonra robot keşif için kaldırımları takip ederek haritayı gezinmeye başlar. Ortam İçerisinde robotun LiDAR sensörünün taradığı alan ve kameradan alınan görüntü Şekil 3.1'de verilmiştir. Bu keşif gezisinde amaç ortam haritasının ve işletme konumlarından oluşan verinin oluşturulmasıdır. Örnek olarak işletme konumları için tutulan değerler Tablo 3.1'de verilmiştir.

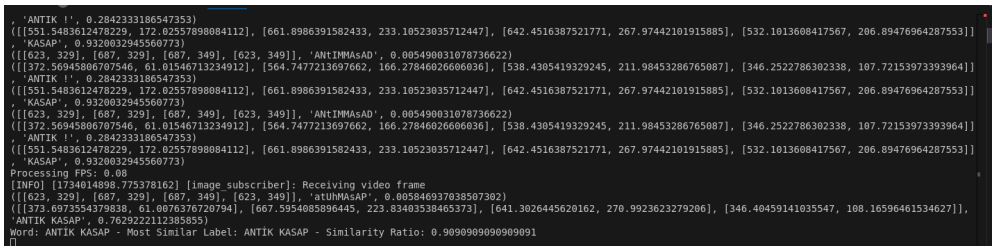
Şekil 3.1: Robotun ortam içerisindeki ekran görüntüsü

Tablo 3.1: İşletme konumları için tutulan veriler

İşletme Adı:	Antik Kasap
Konum X:	-14.6716
Konum Y:	23.5168
Tabela Okuma Süresi:	0.051 Sn
İşletme Bulma Süresi:	50 Sn

Robotun ortam içinde gezinmesi sırasında etrafta bulunan ve metin içeren tabloları tespit ederek metni tanıması gerekir. Ortam örneği baz alındığında "Antik Kasap" işletmesinin tabelası robotun kamera açısına girmiştir. Kamera açısından görüntü Şekil 3.2'de verilmiştir. Metin bölgesi tespit edilerek OCR modellerine tanım için girdi olarak verilir. Robottan görüntü verisi aktarım FPS'ine göre, her bir kare için model metin tespiti yapmaya devam eder. Metin tespit denemeleri Şekil 3.4'te verilmiştir. Doğruluk oranı en yüksek eşleştirme sağlanırsa işletme adı kaydedilir.

Şekil 3.2: Kameradan alınan görüntü



Şekil 3.3: Metin tanıma ile görüntüden tespit edilen işletme adı

Deney ortamından toplanan veriler Tablo 3.2'de verilmiştir. Bu verilerden yola çıkılarak test edilen senaryolar karşılaştırılacaktır. Algoritmaların ve modellerin çeşitliliğinden kaynaklı olarak genel kriter kullanımlarının yüksek olması beklenmektedir. Özellikle görüntüden metin tanıma sistemleri ağırlık olarak GPU kullanımını arttırırken, yol bulma algoritmaları ve robotun hareketini sağlarken hesaplanan değişkenler CPU kullanımını arttırır.

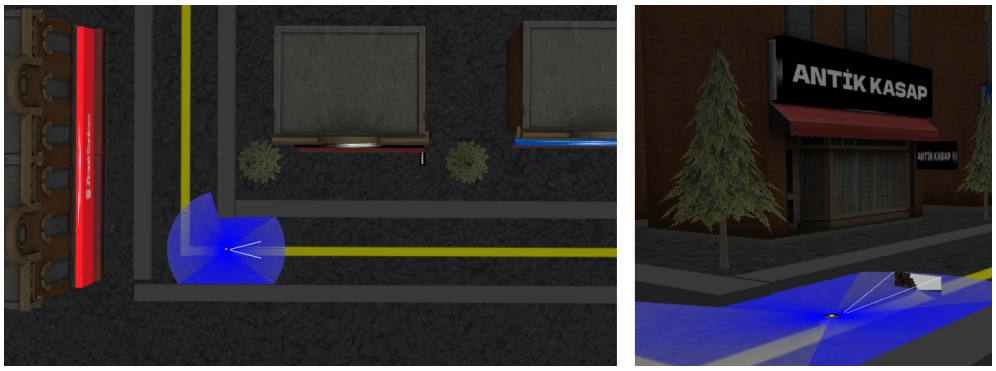
Tablo 3.2: Performans kriterleri tablosu

OCR Modelleri için	Tabela okuma süresi
Yol Planlama Modelleri için	Hedef noktaya ulaşma süresi
Genel Kriterler için	CPU kullanımı
	GPU kullanımı
	RAM

3.1. OCR Modellerinin Performans Analizi

3.1.1. Senaryo 1: Açılı Görüş Açısında Model Performansları

İlk senaryoda, robotun yol üzerinde hareket ettiğinde işletmelerin tabelalarının açılı olarak görüntülenmesi durumunda OCR modellerinin performansları karşılaştırılmıştır. Robotun açısı ve ortam üzerindeki konumu Şekil 3.4'te verilmiştir. Tesseract OCR ve Keras OCR modelleri için, alınan görüntü verisinden herhangi bir metin çıktısı elde edilmemiştir. Bu yüzden değerlendirme kriterleri kaydedilmemiştir.



Şekil 3.4: Senaryo 1: Robot konumu ve görüş açısı

EasyOCR modelinde kullanılan yapay sinir ağı, üzerinde çalıştığı donanıma göre iki farklı deney olarak test edilmiştir. Deneylerde ölçülen ve toplanan performans kriterleri Tablo 3.3'te verilmiştir. İlk deneyde hesaplama yükü işlemci üzerine verilmiştir. Bu yük, CPU kullanımını artırarak 8 çekirdeğin tam performansta çalışmasını sağlamıştır. İkinci deneyde hesaplama yükü GPU üzerine alınarak CPU kullanımı tam performansta 2 çekirdeğe kadar düşmüştür. Hesaplama yükü CPU üzerine verildiğinde FPS büyük ölçüde düşmektedir. Temel sebebi simülasyon ortamı ile koşan kodun aynı donanımı kullanması söylenebilir. Aynı şekilde tabela okuma süreleri arasında 3 kat fark, FPS açısından bakıldığında 4 kat fark ile GPU üzerinde çalışan EasyOCR modelinin bu senaryo daha iyi sonuç verdiği görülmüştür. RAM ve Sanal Bellek(Virtual Memory-VIRT) kullanımı değerleri, iki deney içinde birbirine yakın sonuçlar vermiştir.

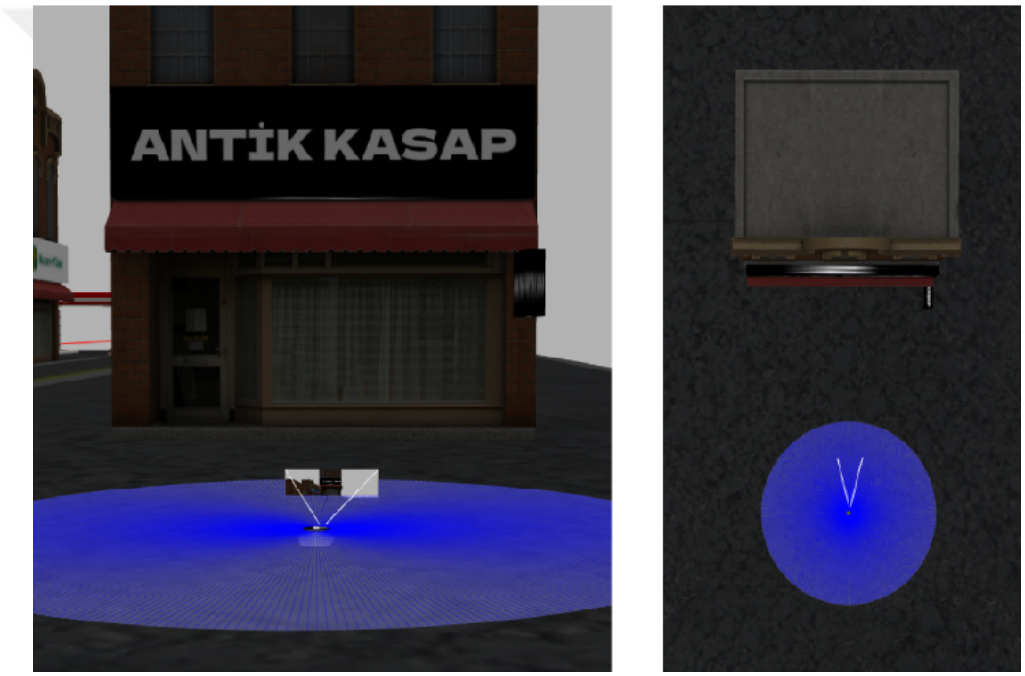
Tablo 3.3: İlk senaryo için EasyOCR modelinin performans sonuçları

EasyOCR	**GPU**	**CPU**
Saniyelik Görüntü Sayısı(FPS)	16.19	3.93
Tabela Okuma Süresi(sn)	0.0713	0.2109
CPU(%)	230	870
RAM(%)	9.1	9.3
VIRT(GB)	15.3	15.2

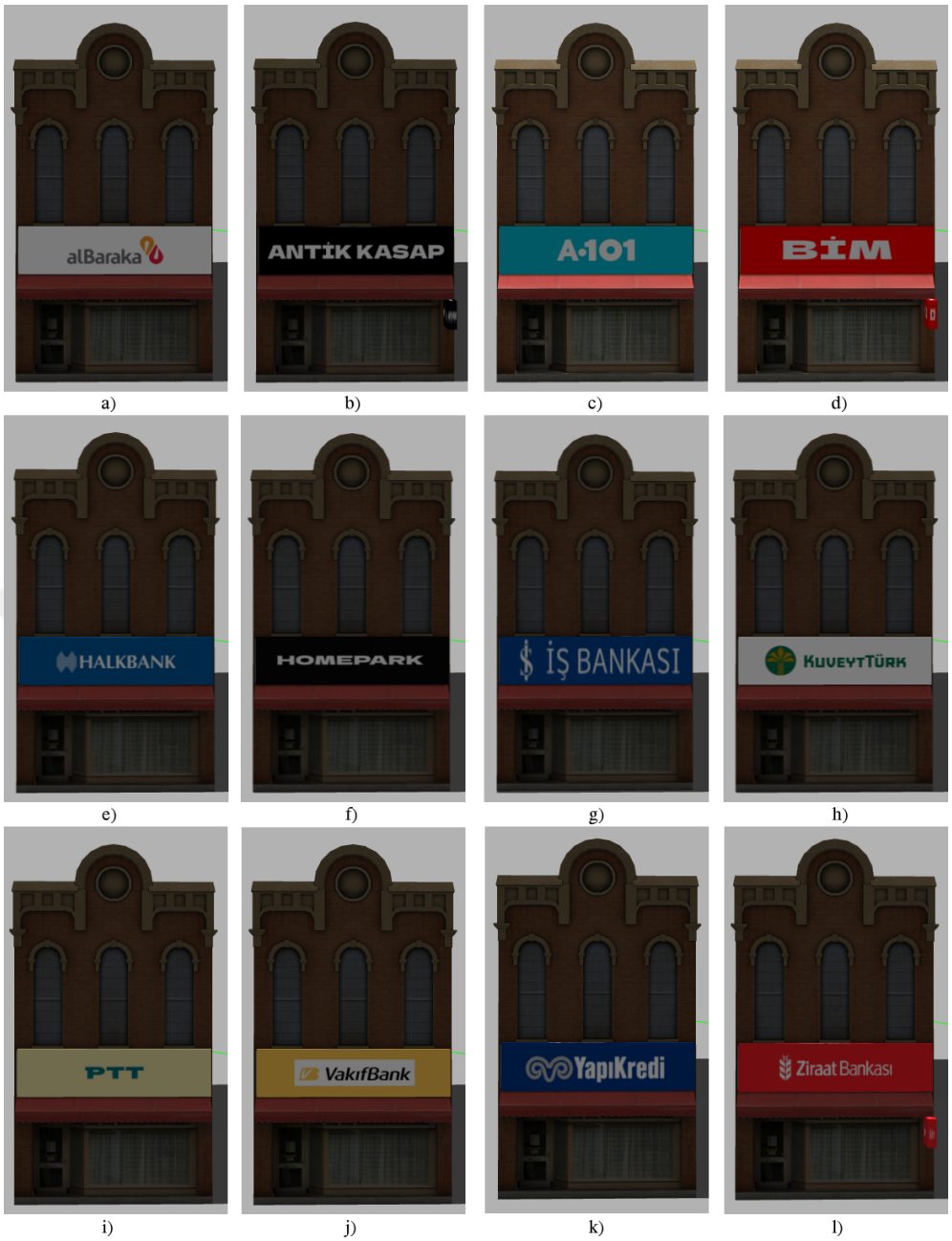
3.1.2. Senaryo 2: Dik Görüş Açısında Model Performansları

İkinci senaryo, ilk senaryoda açılı görüntüden metin tespitini 2 OCR modelinin yapamamasından dolayı ikinci bir senaryo oluşturulmuştur. Bu senaryoda robotun işletmenin tam karşısına, tabelanın kamera açısında olacak şekilde yerleştirilmiştir. Ortamda robotun konumu ve açısı Şekil 3.5'de verilmiştir.

Dik görüş açısında üç modelde metin tespiti yapabilmesinden dolayı her bir işletme tabelası için bu senaryo kullanılmıştır. Modeller tabelalar için test edilerek her bir modelin ortalama performansları hesaplanmıştır. Test edilen işletme tabelalarına Şekil 3.6'da yer verilmiştir. Toplamda 12 işletme tabelası kullanılmıştır.



Şekil 3.5: Senaryo 2: Robot konumu ve görüş açısı



Şekil 3.6: İşletme tabelaları

EasyOCR modeli, GPU ve CPU üzerinde çalıştırılarak iki farklı donanım üstündeki performansı karşılaştırılmıştır. İki farklı donanım testi için toplanan ve hesaplanan performans kriterleri Tablo 3.4'de ve Tablo 3.5'te verilmiştir.

İki test içinde performans sonuçlarının ortalaması Tablo 3.6'da verilmiştir. Her iki test içinde okunan tabela sayısı eşittir. RAM ve VIRT kullanımı birbirine yakın sonuçlar vermiştir. CPU versiyonunda işlemci kullanımının donanımlara verilen hesaplama

yükünden dolayı GPU versiyonundan 5 kat daha fazla olduğu görülmüştür. Hesaplama yükünün GPU'ya aktarılması ise saniyelik görüntü sayısında neredeyse 4 kat artışa ve tabela okuma süresinin ortalamada 0.175 sn azalttığı hesaplanmıştır.

Tablo 3.4: EasyOCR(GPU) modelinin tabela tanımada performans sonuçları

İşletmeler	Saniyelik Görüntü Sayısı(FPS)	Tabela Okuma Süresi(sn)	CPU (%)	RAM (%)	VIRT (GB)
Albaraka	19.35	0.8905	150	9.1	15.7
Antik Kasap	18.21	0.0566	230	9.2	16.1
A101	Tespit edilmedi				
BİM	18.52	0.7552	149	9.1	15.9
HalkBank	19.55	0.7783	142	9.2	15.9
Homepark	16.86	0.0406	162	9.1	15.9
İş Bankası	7.49	0.7599	160	9.2	15.9
KuveytTürk	Tespit edilmedi				
Ptt	18.52	0.8317	142	9.2	15.9
Vakıfbank	15.12	0.0069	157	9.1	15.9
Yapıkredi	11.9	0.0454	180	9.1	15.9
Ziraat Bankası	17	0.97	165	9.2	15.9

Tablo 3.5: EasyOCR(CPU) modelinin tabela tanımada performans sonuçları

İşletmeler	Saniyelik Görüntü Sayısı(FPS)	Tabela Okuma Süresi(sn)	CPU (%)	RAM (%)	VIRT (GB)
Albaraka	6.28	0.1522	900	10.4	15.6
Antik Kasap	4.86	0.8634	876	9.2	15.8
A101	Tespit edilmedi				
BİM	3.94	0.9986	784	9.3	15.8
HalkBank	4.64	0.9130	847	9.3	15.7
Homepark	4.67	0.2223	891	9.3	15.8
İş Bankası	3.31	1.3376	642	9.3	15.7
KuveytTürk	Tespit edilmedi				
Ptt	4	0.2204	862	9.3	15.8
Vakıfbank	4.22	1.0154	904	9.3	15.8
Yapıkredi	4.30	0.2252	876	9.3	15.7
Ziraat Bankası	6.10	0.9995	944	9.3	15.7

Tablo 3.6: EasyOCR modelinin performans sonuçlarının ortalaması

EasyOCR	GPU	CPU
Saniyelik Görüntü Sayısı(FPS)	16.25	4.63
Tabela Okuma Süresi(sn)	0.5197	0.6947
CPU(%)	163.7	852.6
RAM(%)	9.15	9.4
VIRT(GB)	15.9	15.74
Okunan Tabela Sayısı	10	10

Tesseract OCR'da Türkçe için resmi olarak yayınlanan üç model test edilmiştir. Bu modeller normal, best ve fast olarak adlandırılmıştır. Üç model içinde 12 işletmede tabela tespit performansları toplanarak ve hesaplanarak Tablo 3.7, Tablo 3.8 ve Tablo 3.9'da verilmiştir.

Tablo 3.7: Tesseract OCR(Normal) modelinin tabela tanımada performans sonuçları

İşletmeler	Saniyelik Görüntü Sayısı(FPS)	Tabela Okuma Süresi(sn)	CPU (%)	RAM (%)	VIRT (GB)
Albaraka	3.17	0.3132	22	8.4	15.7
Antik Kasap	2.3	0.3205	27.1	8.4	15.7
A101	Tespit edilmedi				
BİM	Tespit edilmedi				
HalkBank	2.64	0.3710	22.9	8.4	15.7
Homepark	1.99	1.1116	20.9	8.4	15.7
İş Bankası	Tespit edilmedi				
KuveytTürk	Tespit edilmedi				
Ptt	Tespit edilmedi				
Vakıfbank	2.79	0.3452	24	8.4	15.7
Yapıkredi	Tespit edilmedi				
Ziraat Bankası	2.55	0.7508	23.6	8.4	15.7

Tablo 3.8: Tesseract OCR(Best) modelinin tabela tanımada performans sonuçları

İşletmeler	Saniyelik Görüntü Sayısı(FPS)	Tabela Okuma Süresi(sn)	CPU (%)	RAM (%)	VIRT (GB)
Albaraka	7.15	0.1461	39.8	8.4	15.7
Antik Kasap	Tespit edilmedi				
A101	Tespit edilmedi				
BİM	Tespit edilmedi				
HalkBank	5.10	0.7754	36.4	8.4	15.7
Homepark	4.51	0.2452	38	8.5	15.7
İş Bankası	Tespit edilmedi				
KuveytTürk	Tespit edilmedi				
Ptt	7.31	0.1453	36.8	8.4	15.7
Vakıfbank	5.77	0.1648	40.6	8.4	15.8
Yapıkredi	3.17	0.1978	36	8.9	15.7
Ziraat Bankası	3.76	0.1348	32	8.5	15.7

Tablo 3.9: Tesseract OCR(Fast) modelinin tabela tanımada performans sonuçları

İşletmeler	Saniyelik Görüntü Sayısı(FPS)	Tabela Okuma Süresi(sn)	CPU (%)	RAM (%)	VIRT (GB)
Albaraka	8.28	0.5764	42	8.5	15.7
Antik Kasap	5.6	0.1636	33	8.4	15.8
A101	6.57	0.1761	50.6	8.5	15.8
BİM	Tespit edilmedi				
HalkBank	6.39	0.1486	47	8.4	15.7
Homepark	5.24	0.2393	33.1	8.4	15.7
İş Bankası	Tespit edilmedi				
KuveytTürk	Tespit edilmedi				
Ptt	10.07	0.1086	94.9	8.4	15.7
Vakıfbank	7.09	0.1402	43	8.4	15.8
Yapıkredi	Tespit edilmedi				
Ziraat Bankası	7.19	0.1402	43	8.4	15.7

Modellerin performans sonuçlarının ortalamasına Tablo 3.10'da yer verilmiştir. RAM ve VIRT kullanımı üç model içinde birbirine yakın sonuçlar vermiştir. Aralarında ayırt edici özelliklere bakıldığında genel anlamda fast modelinin okunan tabela sayısı ve saniyelik görüntü sayısı ile daha iyi sonuçlar verdiği söylenebilir. Normal modelde CPU kullanımı düşükken diğer kriterlerde en düşük performansı gösterdiği görülmektedir. Best modeli 0.03 sn farkla fast modelinden daha hızlı tabela okuduğu hesaplanmıştır.

Tablo 3.10: Tesseract OCR modelinin performans sonuçlarının ortalaması

Tesseract OCR	**Normal**	**Best**	**Fast**
Saniyelik Görüntü Sayısı(FPS)	2.57	5.25	6.23
Tabela Okuma Süresi(sn)	0.535	0.2584	0.2882
CPU(%)	23.42	37.08	48.38
RAM(%)	8.4	8.5	8.42
VIRT(GB)	15.7	15.71	15.72
Okunan Tabela Sayısı	6	7	8

Son OCR modeli olarak Keras modeli işletme tabelalarında test edilmiştir. 12 İşletmeden 10 tanesini tespit edebilmiştir. Her tabela için performans sonuçları ve sonuçların ortalaması Tablo 3.11 ve Tablo 3.12'de verilmiştir. Diğer modellere göre kıyaslandığında ortalama saniyelik görüntü sayısı diğer modellerin en düşük performansından 6 kat daha düşük olarak hesaplanmıştır. Aynı şekilde tabela okuma süreleri diğer modeller için 1 saniyenin altındayken Keras OCR için ortalama 5 saniye olarak hesaplanmıştır. RAM ve VIRT kullanımları diğer modeller ile karşılaştırıldığında daha fazla kaynak ihtiyacı olduğunu göstermektedir. Genel olarak bakıldığında okunan tabela sayısı kriteri dışında diğer kriterlerde en düşük performansı sergilemektedir.

Tablo 3.11: Keras OCR modelinin tabela tanımada performans sonuçları

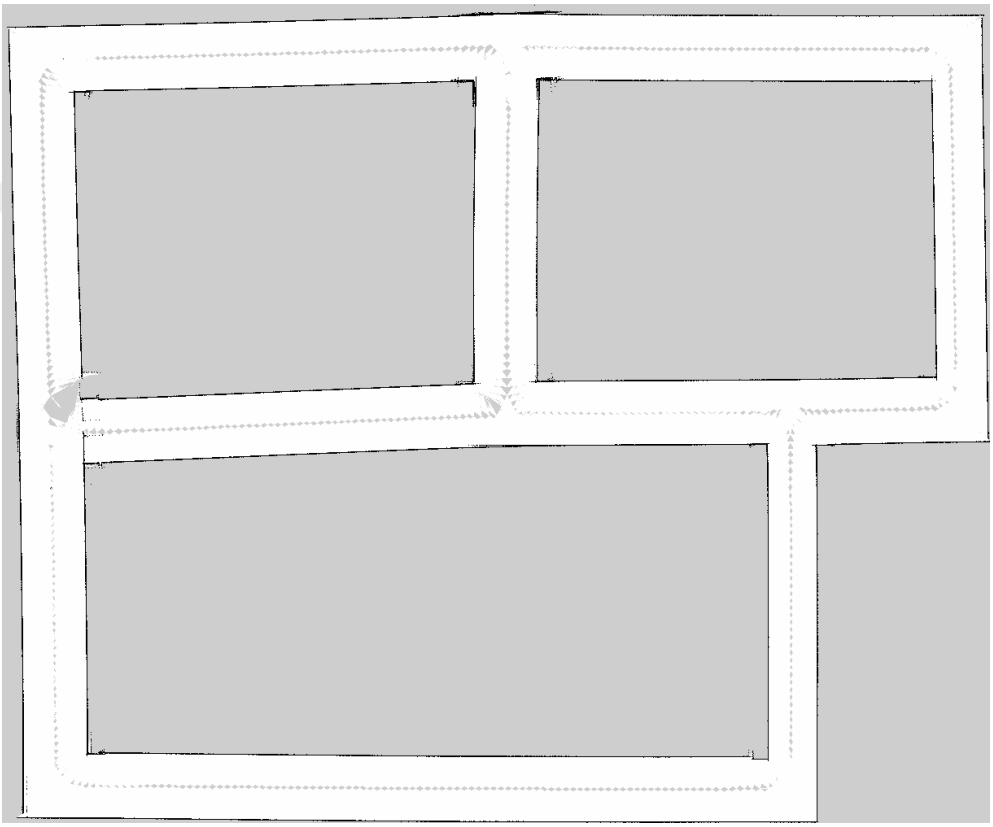
İşletmeler	Saniyelik Görüntü Sayısı(FPS)	Tabela Okuma Süresi(sn)	CPU (%)	RAM (%)	VIRT (GB)
Albaraka	0.69	3.81	940	14	17.2
Antik Kasap	0.25	4.99	112	10.2	18
A101	0.45	5.12	946	12.7	17
BİM	0.42	5.19	275	11.5	17.2
HalkBank	0.47	5.37	1086	12.6	17
Homepark	0.37	7.76	1026	11.5	17.2
İş Bankası	0.12	6.57	233	11.2	17.9
KuveytTürk	Tespit edilmedi				
Ptt	0.68	3.67	1099	14	17
Vakıfbank	0.45	2.38	813	11.5	17.2
Yapıkredi	0.22	5.05	885	13.2	17
Ziraat Bankası	Tespit edilmedi				

Tablo 3.12: Keras OCR modelinin performans sonuçlarının ortalaması

Keras OCR	
Saniyelik Görüntü Sayısı(FPS)	0.412
Tabela Okuma Süresi(sn)	4.991
CPU(%)	741.5
RAM(%)	12.24
VIRT(GB)	17.27
Okunan Tabela Sayısı	10

3.2. Yol Bulma Algoritmalarının Performans Analizi

Yol bulma algoritmaları olarak Dijkstra ve A* algoritmaları kullanılmıştır. Tabela tespiti ile konumları kaydedilen işletmelere başlangıç noktasından itibaren iki farklı algoritma ile gidilerek performansları karşılaştırılmıştır. Konumlara ulaşılmasında gezinme algoritması ve SLAM paketi ile çıkartılan haritaya Şekil 3.7'de verilmiştir. Her bir işletme için performans sonuçları ve sonuçların ortalamaları Tablo 3.13 ve Tablo 3.14'te verilmiştir.



Şekil 3.7: SLAM Toolbox ile oluşturulan ortamın haritası

Performans sonuçlarının ortalamasına bakıldığında iki algoritma için CPU kullanımı dışında sonuçlar birbirine yakın hesaplanmıştır. Genel olarak ele alındığında A* algoritması, Dijkstra algoritmasına göre simülasyon ortamında daha performanslı çalışmaktadır.

Tablo 3.13: Dijkstra algoritması ile hesaplanan performans sonuçları

İşletmeler	Gidilen Mesafe (m)	Geçen Süre (sn)	Enerji (j)	CPU (%)	RAM (%)
Albaraka	90.15	388.98	22.05	126	2.1
Antik Kasap	13.10	53.50	2.75	110	2.4
A101	28.43	120.43	8.35	102.1	2.5
BİM	110.32	568.34	26.63	113.2	2.3
HalkBank	52.40	258.32	12.83	111.5	2.2
Homepark	32.76	172.93	8.39	103	2.4
İş Bankası	24.55	113.66	5.72	99.6	1.8
KuveytTürk	42.87	172.72	9.85	102.4	1.7
Ptt	75.66	340.56	19.23	104	4.2
Vakıfbank	23.89	96.31	5.70	133	2.2
Yapıkredi	76.60	398.78	18.89	126.5	3.2
Ziraat Bankası	102.52	503.80	24.21	117	2.4

Tablo 3.14: A* algoritması ile hesaplanan performans sonuçları

İşletmeler	Gidilen Mesafe (m)	Geçen Süre (sn)	Enerji (j)	CPU (%)	RAM (%)
Albaraka	86.85	360.85	21.74	106	1.8
Antik Kasap	10.08	48.03	2.48	95	2.1
A101	32.42	132.37	8.15	95	2.1
BİM	109.99	565.54	26.96	105	2.1
HalkBank	49.98	238.03	12.21	90.2	2.3
Homepark	35.96	195.91	8.93	88.9	2.5
İş Bankası	22.63	103.64	5.60	86.4	1.4
KuveytTürk	38.44	167.05	9.57	85.6	1.4
Ptt	71.18	305.57	17.73	91.6	3.6
Vakıfbank	22.34	5.56	5.56	111	2.3
Yapıkredi	77.53	400.99	18.67	107	2.2
Ziraat Bankası	98.68	496.63	23.94	86.4	2.1

Tablo 3.15: Yol bulma algoritmalarının performans sonuçları

	Dijkstra	**A***
Hedefe Ulaşırken Gidilen Mesafe(m)	56.10	54.67
Geçen Süre(sn)	265.66	259.56
Harcanan Enerji(j)	13.71	13.43
CPU(%)	112.35	95.54
RAM(%)	2.45	2.15

Harcanan gücün hesaplanmasında ilk olarak robotun anlık hızı hesaplanır. Bu hesaplama için Denklem 3.1 kullanılmıştır. Anlık hız değeri elde edildikten sonra gidilen mesafe değeri çarpılarak robotun yol boyunca harcadığı güç hesaplanmıştır.

$$\sqrt{w^2 + v^2} \tag{3.1}$$

w : Açısal Hız

v : Doğrusal Hız

SONUÇ

Bu tezde, otonom bir robotun Gazebo simülasyon ortamında keşif, haritalama ve yol planlama görevlerini gerçekleştirme kapasitesi analiz edilmiştir. Robot, keşif algoritması ve kamera sensörü kullanarak işletme tabelalarını tespit etmiş, bu tabelalardaki metinleri algılama ve OCR algoritmalarıyla başarılı bir şekilde tanımıştır. EasyOCR, Keras ve Tesseract gibi OCR modelleri kullanılarak işletme adları okunmuş, işletmelere ait koordinatlar belirlenmiş ve işletme isimlerini ve konumlarını içeren görsel bir harita oluşturulmuştur.

Haritalama işlemi sonrası, robotun farklı görev senaryolarında Dijkstra ve A* algoritmalarını kullanarak yol planlama performansı değerlendirilmiştir. Elde edilen sonuçlar, iki algoritmanın güçlü ve zayıf yönlerini ortaya koyarken, belirli durumlarda bir algoritmanın diğerine kıyasla avantaj sağlayabileceğini göstermiştir. EasyOCR, tanıma oranı ve hız (FPS) açısından diğer yöntemlere göre daha iyi sonuçlar vermiştir. Ayrıca, yol planlama sürecinde A* algoritması, Dijkstra algoritmasına kıyasla daha iyi performans sergilemiştir.

Bu çalışma, OCR modelleri ve yol planlama algoritmalarının otonom robot navigasyon sistemlerindeki etkinliğini detaylı bir şekilde incelemiş ve bu sistemlerin geliştirilmesi için değerli bir katkı sunmuştur. Gelecekteki çalışmalarda, daha karmaşık çevrelerde farklı algoritmalar ve sensör sistemleri kullanarak bu yaklaşımları genişletebilir ve daha ileri seviye robot navigasyon sistemleri geliştirebilir.

KAYNAKÇA

AI, J. (2020). Easyocr. Retrieved from https://github.com/JaidedAI/EasyOCR (Erişim Tarihi: 05.12.2024)

Alatise, M. B., & Hancke, G. P. (2020). A review on challenges of autonomous mobile robot and sensor fusion methods. IEEE Access, 8, , 39830-39846. doi: 10.1109/ACCESS.2020.2975643

Bonci, A., Gaudeni, F., Giannini, M. C., & Longhi, S. (2023). Robot operating system 2 (ros2)-based frameworks for increasing robot autonomy: A survey. Applied Sciences, 13, 23, . Retrieved from https://www.mdpi.com/2076-3417/13/23/12796 doi: 10.3390/app132312796

Borenstein, J., & Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots. IEEE Transactions on Systems, Man, and Cybernetics, 19, 5, 1179-1187. doi: 10.1109/21.44033

Büyükkelek, A. F., Dağadası, M., Yusefı, A., Türkmenoğlu, Y., & Durdu, A. (2020). Dji tello ile ros tabanlı haritalandırma simülasyonu. Avrupa Bilim ve Teknoloji Dergisi, , , 504–508. doi: 10.31590/ejosat.820154

Chen, W., Wang, X., Gao, S., Shang, G., Zhou, C., Li, Z., … Hu, K. (2023). Overview of multi-robot collaborative slam from the perspective of data fusion. Machines, 11, 6, . Retrieved from https://www.mdpi.com/2075-1702/11/6/653 doi: 10.3390/machines11060653

Clovaai. (2019). CRAFT-pytorch. https://github.com/clovaai/CRAFT-pytorch. ((Erişim Tarihi: 05.12.2024))

Difflib. (2023). Difflib Library Documentation. https://docs.python.org/3/library/difflib.html. ((Erişim Tarihi: 05.12.2024))

DIJKSTRA, E. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1, , 269-271. Retrieved from http://eudml.org/doc/131436

Faizullah, S., Ayub, M. S., Hussain, S., & Khan, M. A. (2023). A survey of ocr in arabic language: Applications, techniques, and challenges. Applied Sciences, 13, 7, . Retrieved from https://www.mdpi.com/2076-3417/13/7/4584 doi: 10.3390/app13074584

Hansen, M. (2019). Navigation2 Overview. https://roscon.ros.org/2019/talks/roscon2019_navigation2_overview_final.pdf. ((Erişim Tarihi: 06.12.2024))

Hou, Y., Kong, Q., Wang, J., & Li, S. (2018, 07). Polyphonic audio tagging with

sequentially labelled data using crnn with learnable gated linear units..

Index, R. (2024). nav2_navfn_planner. https://index.ros.org/p/nav2_navfn_planner/. ((Erişim Tarihi: 05.12.2024))

Jain, P. H., Kumar, V., Samuel, J., Singh, S., Mannepalli, A., & Anderson, R. (2023). Artificially intelligent readers: An adaptive framework for original handwritten numerical digits recognition with ocr methods. Information, 14, 6, . Retrieved from https://www.mdpi.com/2078-2489/14/6/305 doi: 10.3390/info14060305

Ju, C., Qinghua, L., & Yan, X. (2020, 10). Path planning using an improved a-star algorithm. In (p. 23-26). doi: 10.1109/PHM-Jinan48558.2020.00012

Karur, K., Sharma, N., Dharmatti, C., & Siegel, J. E. (2021). A survey of path planning algorithms for mobile robots. Vehicles, 3, 3, 448–468. Retrieved from https://www.mdpi.com/2624-8921/3/3/27 doi: 10.3390/vehicles3030027

Koenig, N., & Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In 2004 ieee/rsj international conference on intelligent robots and systems (iros) (ieee cat. no.04ch37566) (Vol. 3, p. 2149-2154 vol.3). doi: 10.1109/IROS.2004.1389727

Lee, H., Yoon, J., Jang, M.-S., & Park, K.-J. (2021). A robot operating system framework for secure uav communications. Sensors, 21, 4, . Retrieved from https://www.mdpi.com/1424-8220/21/4/1369 doi: 10.3390/s21041369

Lin, H.-I., & Tzeng, H. J. (2014). Search strategy of a mobile robot for radiation sources in an unknown environment. In 2014 international conference on advanced robotics and intelligent systems (aris) (p. 56-60). doi: 10.1109/ARIS.2014.6871492

Liu, L., Wang, X., Yang, X., Liu, H., Li, J., & Wang, P. (2023). Path planning techniques for mobile robots: Review and prospect. Expert Systems with Applications, 227, , 120254. Retrieved from https://www.sciencedirect.com/science/article/pii/S095741742300756X doi: https://doi.org/10.1016/j.eswa.2023.120254

Liu, X., & Gong, D. (2011). A comparative study of a-star algorithms for search and rescue in perfect maze. In 2011 international conference on electric information and control engineering (p. 24-27). doi: 10.1109/ICEICE.2011.5777723

Liu, Z., & Zhang, S. (2023). Robot path planning based on improved a* algorithm. In 2023 international conference on computer engineering and distance learning (cedl) (p. 75-79). doi: 10.1109/CEDL60560.2023.00022

Maruyama, Y., Kato, S., & Azumi, T. (2016, 10). Exploring the performance of ros2. In (p. 1-10). doi: 10.1145/2968478.2968502

Mei, R., Wang, Z., & Chen, X. (2024, 08). Crnn-resnet: Combined crnn and resnet networks for ofdm receivers. IEEE Transactions on Cognitive Communications and Networking, PP, , 1-1. doi: 10.1109/TCCN.2024.3378225

Memon, J., Sami, M., Khan, R. A., & Uddin, M. (2020). Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr). IEEE Access, 8, , 142642-142668. doi: 10.1109/ACCESS.2020.3012542

Niu, H., Ji, Z., Arvin, F., Lennox, B., Yin, H., & Carrasco, J. (2021, 01). Accelerated sim-to-real deep reinforcement learning: Learning collision avoidance from human player. In (p. 144-149). doi: 10.1109/IEEECONF49454.2021.9382693

P Anantha Raj, M. S. (2018). Internet of robotic things based autonomous fire fighting mobile robot. In 2018 ieee international conference on computational intelligence and computing research (iccic) (p. 1-4). doi: 10.1109/ICCIC.2018.8782369

Patel, C., Patel, A., & Patel, D. (2012, 10). Optical character recognition by open source ocr tool tesseract: A case study. International Journal of Computer Applications, 55, , 50-56. doi: 10.5120/8794-2784

Pham, H., Lam, H., Duy, L., Bao, P., & Trinh, D. (2024, 09). An improved convolutional recurrent neural network for stock price forecasting. IAES International Journal of Artificial Intelligence (IJ-AI), 13, , 3381. doi: 10.11591/ijai.v13.i3.pp3381-3394

Ratcliff, J. W., Metzener, D., et al. (1988). Pattern matching: The gestalt approach. Dr. Dobb's Journal, 13, 7, 46.

Robotics, O. (2018). Ros2: Robot operating system 2. Retrieved from `https://github.com/ros2/ros2` (Erişim Tarihi: 03.12.2024)

Robotics, O. (2024a). About Gazebo. https://gazebosim.org/about. ((Erişim Tarihi: 12.12.2024))

Robotics, O. (2024b). Gazebo Models. http://models.gazebosim.org/. ((Erişim Tarihi: 12.12.2024))

Robotis. (2024). Turtlebot3 Features. https://emanual.robotis.com/docs/. ((Erişim Tarihi: 11.12.2024))

ROS2-Workshop. (2024). ROS 2 Navigation. https://ros2-industrial-workshop.readthedocs.io/en/latest/_source/navigation/ROS2-Navigation.html. ((Erişim Tarihi: 05.12.2024))

Sankalprajan, P., Pendyala, T. S., Perur, H. D., & Pagala, P. (2020, 06). Comparative analysis of ros based 2d and 3d slam algorithms for autonomous ground vehicles. In (p. 1-6). doi: 10.1109/INCET49848.2020.9154101

Smith, R. (2007). An overview of the tesseract ocr engine. In Ninth international conference on document analysis and recognition (icdar 2007) (Vol. 2, p. 629-633). doi: 10.1109/ICDAR.2007.4376991

Smith, R., Self, M., & Cheeseman, P. (1986, 01). Estimating uncertain spatial relationships in robotics. In (Vol. 1, p. 435-461). doi: 10.1109/ROBOT.1987.1087846

Sonkusare, M., & Sahu, N. (2016, 03). A survey on handwritten character recognition (hcr) techniques for english alphabets. Advances in Vision Computing: An International Journal, 3, , 1-12. doi: 10.5121/avc.2016.3101

Sporici, D., Cusnir, E., & Boiangiu, C.-A. (2020). Improving the accuracy of tesseract 4.0 ocr engine using convolution-based preprocessing. Symmetry, 12, 5, . Retrieved from https://www.mdpi.com/2073-8994/12/5/715 doi: 10.3390/sym12050715

Stewart, J., & Church, M. (2024, 10). Ros 2 robot with slam. doi: 10.13140/RG.2.2.15510.56643

Sun, W., Wang, H., Lu, Y., Luo, J., Liu, T., Lin, J., … Zhang, G. (2022). Deep-learning-based complex scene text detection algorithm for architectural images. Mathematics, 10, 20, . Retrieved from https://www.mdpi.com/2227-7390/10/20/3914 doi: 10.3390/math10203914

Tang, G., Tang, C., Claramunt, C., Hu, X., & Zhou, P. (2021). Geometric a-star algorithm: An improved a-star algorithm for agv path planning in a port environment. IEEE Access, 9, , 59196-59210. doi: 10.1109/ACCESS.2021.3070054

Tian, L., Zhang, Z., Zheng, C., Tian, Y., Zhao, Y., Wang, Z., & Qin, Y. (2021). An improved rapidly-exploring random trees algorithm combining parent point priority determination strategy and real-time optimization strategy for path planning. Sensors, 21, 20, . Retrieved from https://www.mdpi.com/1424-8220/21/20/6907 doi: 10.3390/s21206907

Wiki, R. (2024). Parameter server. https://wiki.ros.org/Parameter%20Server. (Erişim Tarihi: 04.12.2024)

Yektek, K. (2024). Framework nedir, ne İşe yarar? Retrieved from https://www.patika.dev/blog/framework-nedir-ne-ise-yarar (Erişim Tarihi: 04.12.2024)

Zhang, J. (2021, 10). Ai based algorithms of path planning, navigation and control for

mobile ground robots and uavs. doi: 10.48550/arXiv.2110.00910

Zhou, X., Yan, J., Yan, M., Mao, K., Yang, R., & Liu, W. (2023). Path planning of rail-mounted logistics robots based on the improved dijkstra algorithm. Applied Sciences, 13, 17, . Retrieved from https://www.mdpi.com/2076-3417/13/17/9955 doi: 10.3390/app13179955

Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). East: An efficient and accurate scene text detector. In 2017 ieee conference on computer vision and pattern recognition (cvpr) (p. 2642-2651). doi: 10.1109/CVPR.2017.283

Özdemir, S., Sacar, O., & Evrencan, O. (2021). Dijkstra algoritması kullanılarak İpek yolu koridorları arasında en kısa ulaştırma güzergâhının belirlenmesi. Demiryolu Mühendisliği, , 13, 97–105. doi: 10.47072/demiryolu.811572

EKLER

Tez çalışmasında kullanılan bazı dosyalara aşağıda yer verilmiştir.

EK1: Robot Simülasyon Başlatma Dosyası

```
import os

from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import
    PythonLaunchDescriptionSource
from launch.substitutions import LaunchConfiguration


def generate_launch_description():
    turtle_file_dir = os.path.join(get_package_share_directory('
        turtlebot3_gazebo'), 'launch')
    launch_file_dir = os.path.join(get_package_share_directory('
        robot_project'), 'launch')
    pkg_gazebo_ros = get_package_share_directory('gazebo_ros')
    #map_file = '/home/bugra/maps/wall_following_map3.yaml'

    use_sim_time = LaunchConfiguration('use_sim_time', default='true
        ')
    #x_pose = LaunchConfiguration('x_pose', default='-31.8042')
    #y_pose = LaunchConfiguration('y_pose', default='-31.1429')
    x_pose = LaunchConfiguration('x_pose', default='-33.3135')
    y_pose = LaunchConfiguration('y_pose', default='-26.7086')

    #world = os.path.join(
    #    '/home/bugra/yuksek-ws/src/robot_project',
    #    'worlds',
    #    'worldagacli''nw5a.world'
    #)
    world = os.path.join(
        '/home/bugra/yuksek-ws/src/robot_project',
        'worlds',
```

```
        'nw5a.world'
    )

    gpu_rendering_args = {
        'render_engine': 'ogre',  # GPU ıııhzlandrmas için ogre
           render motoru
        'verbose': '',  # ıDetayl ııçkt al
        'use_sim_time': 'true'  # Simülasyon saati
    }

    gzserver_cmd = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
            os.path.join(pkg_gazebo_ros, 'launch', 'gzserver.launch.
               py')
        ),
        launch_arguments={**gpu_rendering_args,'world': world}.items
           ()
    )

    gzclient_cmd = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
            os.path.join(pkg_gazebo_ros, 'launch', 'gzclient.launch.
               py')
        )
    )

    robot_state_publisher_cmd = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
            os.path.join(turtle_file_dir, 'robot_state_publisher.
               launch.py')
        ),
        launch_arguments={'use_sim_time': use_sim_time}.items()
    )

    spawn_turtlebot_cmd = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
            os.path.join(turtle_file_dir, 'spawn_turtlebot3.launch.
               py')
        ),
```

```
        launch_arguments={
            'x_pose': x_pose,
            'y_pose': y_pose
        }.items()
    )

    #map_file = LaunchConfiguration('map', default='/home/bugra/maps
        /wall_following_map3.yaml')
    #params_file = LaunchConfiguration('params_file', default='/home
        /bugra/yuksek-ws/src/robot_project/config/nav2_params.yaml')
    map_file= '/home/bugra/maps/map.yaml'
    params_file='/home/bugra/yuksek-ws/src/robot_project/config/
        nav2_params.yaml'
    # TurtleBot3 ıdosyalarnn konumunu ıtanmlayn (örnek bir yol)
    nav2_launch_dir = '/opt/ros/humble/share/nav2_bringup/launch'

    # Nav2 bringup launch ıdosyasn dahil et
    nav2_bringup_cmd = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
            os.path.join(nav2_launch_dir, 'bringup_launch.py')  #
                Nav2'nin kendi launch ıdosyas
        ),
        launch_arguments={
            'map': map_file,
            'params_file': params_file,     # Nav2 parametre
                ıdosyasn belirt
            'use_sim_time': 'true',         # Simülasyon ızaman
                ıkullanm
        }.items()
    )
#
    ld = LaunchDescription()

    # Add the commands to the launch description
    ld.add_action(gzserver_cmd)
    ld.add_action(gzclient_cmd)
    ld.add_action(robot_state_publisher_cmd)
    ld.add_action(spawn_turtlebot_cmd)
    ld.add_action(nav2_bringup_cmd)
```

```
96
97      return ld
```

EK2: Simülasyon Ortamında Model Tanımlarının Dünya Dosyası

```
1  <sdf version='1.7'>
2    <world name='default'>
3      <light name='sun' type='directional'>
4        <cast_shadows>1</cast_shadows>
5        <pose>0 0 10 0 -0 0</pose>
6        <diffuse>0.8 0.8 0.8 1</diffuse>
7        <specular>0.2 0.2 0.2 1</specular>
8        <attenuation>
9          <range>1000</range>
10         <constant>0.9</constant>
11         <linear>0.01</linear>
12         <quadratic>0.001</quadratic>
13       </attenuation>
14       <direction>-0.5 0.1 -0.9</direction>
15       <spot>
16         <inner_angle>0</inner_angle>
17         <outer_angle>0</outer_angle>
18         <falloff>0</falloff>
19       </spot>
20     </light>
21     <model name='ground_plane'>
22       <static>1</static>
23       <link name='link'>
24         <collision name='collision'>
25           <geometry>
26             <plane>
27               <normal>0 0 1</normal>
28               <size>100 100</size>
29             </plane>
30           </geometry>
31           <surface>
32             <friction>
33               <ode>
34                 <mu>100</mu>
```

```
              <mu2>50</mu2>
            </ode>
            <torsional>
              <ode/>
            </torsional>
          </friction>
          <contact>
            <ode/>
          </contact>
          <bounce/>
        </surface>
        <max_contacts>10</max_contacts>
      </collision>
      <visual name='visual'>
        <cast_shadows>0</cast_shadows>
        <geometry>
          <plane>
            <normal>0 0 1</normal>
            <size>100 100</size>
          </plane>
        </geometry>
        <material>
          <script>
            <uri>file://media/materials/scripts/gazebo.material</
               uri>
            <name>Gazebo/Grey</name>
          </script>
        </material>
      </visual>
      <self_collide>0</self_collide>
      <enable_wind>0</enable_wind>
      <kinematic>0</kinematic>
    </link>
  </model>
  <gravity>0 0 -9.8</gravity>
  <magnetic_field>6e-06 2.3e-05 -4.2e-05</magnetic_field>
  <atmosphere type='adiabatic'/>
  <physics type='ode'>
    <max_step_size>0.001</max_step_size>
```

```
      <real_time_factor>1</real_time_factor>
      <real_time_update_rate>1000</real_time_update_rate>
    </physics>
    <scene>
      <ambient>0.4 0.4 0.4 1</ambient>
      <background>0.7 0.7 0.7 1</background>
      <shadows>1</shadows>
    </scene>
    <audio>
      <device>default</device>
    </audio>
    <wind/>
    <spherical_coordinates>
      <surface_model>EARTH_WGS84</surface_model>
      <latitude_deg>0</latitude_deg>
      <longitude_deg>0</longitude_deg>
      <elevation>0</elevation>
      <heading_deg>0</heading_deg>
    </spherical_coordinates>
    <model name='House 1'>
      <static>1</static>
      <link name='link'>
        <collision name='collision'>
          <geometry>
            <mesh>
              <uri>model://house_1/meshes/house_1.dae</uri>
              <scale>1.5 1.5 1.5</scale>
            </mesh>
          </geometry>
          <max_contacts>10</max_contacts>
          <surface>
            <contact>
              <ode/>
            </contact>
            <bounce/>
            <friction>
              <torsional>
                <ode/>
              </torsional>
```

```
            <ode/>
          </friction>
        </surface>
      </collision>
      <visual name='visual'>
        <geometry>
          <mesh>
            <uri>model://house_1/meshes/house_1.dae</uri>
            <scale>1.5 1.5 1.5</scale>
          </mesh>
        </geometry>
        <material>
          <script>
            <uri>model://house_1/materials/scripts</uri>
            <uri>model://house_1/materials/textures</uri>
            <name>House_1/Diffuse</name>
          </script>
          <shader type='normal_map_tangent_space'>
            <normal_map>House_1_Normal.png</normal_map>
          </shader>
        </material>
      </visual>
      <self_collide>0</self_collide>
      <enable_wind>0</enable_wind>
      <kinematic>0</kinematic>
    </link>
    <pose>30.1864 -36.0466 0 0 -0 0</pose>
  </model>
  <model name='House 2'>
    <static>1</static>
    <link name='link'>
      <collision name='collision'>
        <geometry>
          <mesh>
            <uri>model://house_2/meshes/house_2.dae</uri>
            <scale>1.5 1.5 1.5</scale>
          </mesh>
        </geometry>
        <max_contacts>10</max_contacts>
```

```
        <surface>
          <contact>
            <ode/>
          </contact>
          <bounce/>
          <friction>
            <torsional>
              <ode/>
            </torsional>
            <ode/>
          </friction>
        </surface>
      </collision>
      <visual name='visual'>
        <geometry>
          <mesh>
            <uri>model://house_2/meshes/house_2.dae</uri>
            <scale>1.5 1.5 1.5</scale>
          </mesh>
        </geometry>
        <material>
          <script>
            <uri>model://house_2/materials/scripts</uri>
            <uri>model://house_2/materials/textures</uri>
            <uri>model://house_1/materials/textures</uri>
            <name>House_2/Diffuse</name>
          </script>
          <shader type='normal_map_tangent_space'>
            <normal_map>model://house_1/materials/textures/
                House_1_Normal.png</normal_map>
          </shader>
        </material>
      </visual>
      <self_collide>0</self_collide>
      <enable_wind>0</enable_wind>
      <kinematic>0</kinematic>
    </link>
    <pose>31.5406 -18.0664 0 0 -0 0</pose>
  </model>
```

```
    <model name='antik-tabela'>
      <link name='link'>
        <self_collide>0</self_collide>
        <enable_wind>0</enable_wind>
        <kinematic>0</kinematic>
        <pose>0 0 0 0 -0 0</pose>
        <visual name='visual'>
          <pose>0 -0.912491 0 0 -0 0</pose>
          <geometry>
            <mesh>
              <uri>model://thrift_shop/meshes/thrift_shop.dae</uri>
            </mesh>
          </geometry>
          <material>
            <shader type='pixel'/>
          </material>
          <transparency>0</transparency>
          <cast_shadows>1</cast_shadows>
        </visual>
        <collision name='collision'>
          <laser_retro>0</laser_retro>
          <max_contacts>10</max_contacts>
          <pose>0 0 5.69469 0 -0 0</pose>
          <geometry>
            <box>
              <size>7.21297 5.43165 11.3894</size>
            </box>
          </geometry>
          <surface>
            <friction>
              <ode>
                <mu>1</mu>
                <mu2>1</mu2>
                <fdir1>0 0 0</fdir1>
                <slip1>0</slip1>
                <slip2>0</slip2>
              </ode>
              <torsional>
                <coefficient>1</coefficient>
```

```
                <patch_radius>0</patch_radius>
                <surface_radius>0</surface_radius>
                <use_patch_radius>1</use_patch_radius>
                <ode>
                  <slip>0</slip>
                </ode>
              </torsional>
            </friction>
            <bounce>
              <restitution_coefficient>0</restitution_coefficient>
              <threshold>1e+06</threshold>
            </bounce>
            <contact>
              <collide_without_contact>0</collide_without_contact>
              <collide_without_contact_bitmask>1</
                  collide_without_contact_bitmask>
              <collide_bitmask>1</collide_bitmask>
              <ode>
                <soft_cfm>0</soft_cfm>
                <soft_erp>0.2</soft_erp>
                <kp>1e+13</kp>
                <kd>1</kd>
                <max_vel>0.01</max_vel>
                <min_depth>0</min_depth>
              </ode>
              <bullet>
                <split_impulse>1</split_impulse>
                <split_impulse_penetration_threshold>-0.01</
                    split_impulse_penetration_threshold>
                <soft_cfm>0</soft_cfm>
                <soft_erp>0.2</soft_erp>
                <kp>1e+13</kp>
                <kd>1</kd>
              </bullet>
            </contact>
          </surface>
        </collision>
      </link>
      <link name='link_1'>
```

```
        <inertial>
          <mass>4.00858</mass>
          <inertia>
            <ixx>1.03745</ixx>
            <ixy>0</ixy>
            <ixz>0</ixz>
            <iyy>17.2923</iyy>
            <iyz>0</iyz>
            <izz>16.3283</izz>
          </inertia>
          <pose>0 0 0 0 -0 0</pose>
        </inertial>
        <pose>0.022328 -3.05607 4.80447 0 -0 0</pose>
        <self_collide>0</self_collide>
        <enable_wind>0</enable_wind>
        <kinematic>0</kinematic>
        <visual name='visual'>
          <pose>0 0 0 0 -0 0</pose>
          <geometry>
            <box>
              <size>6.98357 0.331569 1.73082</size>
            </box>
          </geometry>
          <material>
            <lighting>1</lighting>
            <script>
              <uri>model://antik/materials/scripts</uri>
              <uri>model://antik/materials/textures</uri>
              <name>antik/Diffuse</name>
            </script>
            <shader type='pixel'/>
          </material>
          <transparency>0</transparency>
          <cast_shadows>1</cast_shadows>
        </visual>
        <collision name='collision'>
          <laser_retro>0</laser_retro>
          <max_contacts>10</max_contacts>
          <pose>0 0 0 0 -0 0</pose>
```

```
304             <geometry>
305               <box>
306                 <size>6.98357 0.331569 1.73082</size>
307               </box>
308             </geometry>
309             <surface>
310               <friction>
311                 <ode>
312                   <mu>1</mu>
313                   <mu2>1</mu2>
314                   <fdir1>0 0 0</fdir1>
315                   <slip1>0</slip1>
316                   <slip2>0</slip2>
317                 </ode>
318                 <torsional>
319                   <coefficient>1</coefficient>
320                   <patch_radius>0</patch_radius>
321                   <surface_radius>0</surface_radius>
322                   <use_patch_radius>1</use_patch_radius>
323                   <ode>
324                     <slip>0</slip>
325                   </ode>
326                 </torsional>
327               </friction>
328               <bounce>
329                 <restitution_coefficient>0</restitution_coefficient>
330                 <threshold>1e+06</threshold>
331               </bounce>
332               <contact>
333                 <collide_without_contact>0</collide_without_contact>
334                 <collide_without_contact_bitmask>1</
                      collide_without_contact_bitmask>
335                 <collide_bitmask>1</collide_bitmask>
336                 <ode>
337                   <soft_cfm>0</soft_cfm>
338                   <soft_erp>0.2</soft_erp>
339                   <kp>1e+13</kp>
340                   <kd>1</kd>
341                   <max_vel>0.01</max_vel>
```

```
              <min_depth>0</min_depth>
            </ode>
            <bullet>
              <split_impulse>1</split_impulse>
              <split_impulse_penetration_threshold>-0.01</
                  split_impulse_penetration_threshold>
              <soft_cfm>0</soft_cfm>
              <soft_erp>0.2</soft_erp>
              <kp>1e+13</kp>
              <kd>1</kd>
            </bullet>
          </contact>
        </surface>
      </collision>
    </link>
    <link name='link_1_0'>
      <inertial>
        <mass>0.505154</mass>
        <inertia>
          <ixx>0.145823</ixx>
          <ixy>0</ixy>
          <ixz>0</ixz>
          <iyy>0.046456</iyy>
          <iyz>0</iyz>
          <izz>0.108087</izz>
        </inertia>
        <pose>0 0 0 0 -0 0</pose>
      </inertial>
      <pose>3.37683 -3.47967 2.55398 0 -0 0</pose>
      <visual name='visual'>
        <pose>0 0 0 0 -0 0</pose>
        <geometry>
          <box>
            <size>0.321821 1.56973 1</size>
          </box>
        </geometry>
        <material>
          <lighting>1</lighting>
          <script>
```

```
              <uri>model://antik/materials/scripts</uri>
              <uri>model://antik/materials/textures</uri>
              <name>antik/Diffuse</name>
            </script>
            <shader type='pixel'/>
          </material>
          <transparency>0</transparency>
          <cast_shadows>1</cast_shadows>
        </visual>
        <collision name='collision'>
          <laser_retro>0</laser_retro>
          <max_contacts>10</max_contacts>
          <pose>0 0 0 0 -0 0</pose>
          <geometry>
            <box>
              <size>0.321821 1.56973 1</size>
            </box>
          </geometry>
          <surface>
            <friction>
              <ode>
                <mu>1</mu>
                <mu2>1</mu2>
                <fdir1>0 0 0</fdir1>
                <slip1>0</slip1>
                <slip2>0</slip2>
              </ode>
              <torsional>
                <coefficient>1</coefficient>
                <patch_radius>0</patch_radius>
                <surface_radius>0</surface_radius>
                <use_patch_radius>1</use_patch_radius>
                <ode>
                  <slip>0</slip>
                </ode>
              </torsional>
            </friction>
            <bounce>
              <restitution_coefficient>0</restitution_coefficient>
```

```
            <threshold>1e+06</threshold>
          </bounce>
          <contact>
            <collide_without_contact>0</collide_without_contact>
            <collide_without_contact_bitmask>1</
                collide_without_contact_bitmask>
            <collide_bitmask>1</collide_bitmask>
            <ode>
              <soft_cfm>0</soft_cfm>
              <soft_erp>0.2</soft_erp>
              <kp>1e+13</kp>
              <kd>1</kd>
              <max_vel>0.01</max_vel>
              <min_depth>0</min_depth>
            </ode>
            <bullet>
              <split_impulse>1</split_impulse>
              <split_impulse_penetration_threshold>-0.01</
                  split_impulse_penetration_threshold>
              <soft_cfm>0</soft_cfm>
              <soft_erp>0.2</soft_erp>
              <kp>1e+13</kp>
              <kd>1</kd>
            </bullet>
          </contact>
        </surface>
      </collision>
      <self_collide>0</self_collide>
      <enable_wind>0</enable_wind>
      <kinematic>0</kinematic>
    </link>
    <static>1</static>
    <allow_auto_disable>1</allow_auto_disable>
    <pose>15.4434 -42.4998 0 0 -0 0</pose>
  </model>
  <model name='ziraat'>
    <link name='link'>
      <self_collide>0</self_collide>
      <enable_wind>0</enable_wind>
```

```
        <kinematic>0</kinematic>
        <pose>0 0 0 0 -0 0</pose>
        <visual name='visual'>
          <pose>0 -0.912491 0 0 -0 0</pose>
          <geometry>
            <mesh>
              <uri>model://thrift_shop/meshes/thrift_shop.dae</uri>
            </mesh>
          </geometry>
          <material>
            <shader type='pixel'/>
          </material>
          <transparency>0</transparency>
          <cast_shadows>1</cast_shadows>
        </visual>
        <collision name='collision'>
          <laser_retro>0</laser_retro>
          <max_contacts>10</max_contacts>
          <pose>0 0 5.69469 0 -0 0</pose>
          <geometry>
            <box>
              <size>7.21297 5.43165 11.3894</size>
            </box>
          </geometry>
          <surface>
            <friction>
              <ode>
                <mu>1</mu>
                <mu2>1</mu2>
                <fdir1>0 0 0</fdir1>
                <slip1>0</slip1>
                <slip2>0</slip2>
              </ode>
              <torsional>
                <coefficient>1</coefficient>
                <patch_radius>0</patch_radius>
                <surface_radius>0</surface_radius>
                <use_patch_radius>1</use_patch_radius>
                <ode>
```

```
                <slip>0</slip>
              </ode>
            </torsional>
          </friction>
          <bounce>
            <restitution_coefficient>0</restitution_coefficient>
            <threshold>1e+06</threshold>
          </bounce>
          <contact>
            <collide_without_contact>0</collide_without_contact>
            <collide_without_contact_bitmask>1</
               collide_without_contact_bitmask>
            <collide_bitmask>1</collide_bitmask>
            <ode>
              <soft_cfm>0</soft_cfm>
              <soft_erp>0.2</soft_erp>
              <kp>1e+13</kp>
              <kd>1</kd>
              <max_vel>0.01</max_vel>
              <min_depth>0</min_depth>
            </ode>
            <bullet>
              <split_impulse>1</split_impulse>
              <split_impulse_penetration_threshold>-0.01</
                 split_impulse_penetration_threshold>
              <soft_cfm>0</soft_cfm>
              <soft_erp>0.2</soft_erp>
              <kp>1e+13</kp>
              <kd>1</kd>
            </bullet>
          </contact>
        </surface>
      </collision>
    </link>
    <link name='link_0'>
      <inertial>
        <mass>8.18069</mass>
        <inertia>
          <ixx>1.8493</ixx>
```

```
            <ixy>0</ixy>
            <ixz>0</ixz>
            <iyy>136.001</iyy>
            <iyz>0</iyz>
            <izz>134.331</izz>
          </inertia>
          <pose>0 0 0 0 -0 0</pose>
        </inertial>
        <pose>3.4816 -3.17386 4.82656 0 -0 0</pose>
        <self_collide>0</self_collide>
        <enable_wind>0</enable_wind>
        <kinematic>0</kinematic>
        <visual name='visual'>
          <pose>0 0 0 0 -0 0</pose>
          <geometry>
            <box>
              <size>14.0326 0.362972 1.60653</size>
            </box>
          </geometry>
          <material>
            <lighting>1</lighting>
            <script>
              <uri>model://ziraat/materials/scripts</uri>
              <uri>model://ziraat/materials/textures</uri>
              <name>ziraat/Diffuse</name>
            </script>
            <shader type='pixel'/>
          </material>
          <transparency>0</transparency>
          <cast_shadows>1</cast_shadows>
        </visual>
        <collision name='collision'>
          <laser_retro>0</laser_retro>
          <max_contacts>10</max_contacts>
          <pose>0 0 0 0 -0 0</pose>
          <geometry>
            <box>
              <size>14.0326 0.362972 1.60653</size>
            </box>
```

```
          </geometry>
          <surface>
            <friction>
              <ode>
                <mu>1</mu>
                <mu2>1</mu2>
                <fdir1>0 0 0</fdir1>
                <slip1>0</slip1>
                <slip2>0</slip2>
              </ode>
              <torsional>
                <coefficient>1</coefficient>
                <patch_radius>0</patch_radius>
                <surface_radius>0</surface_radius>
                <use_patch_radius>1</use_patch_radius>
                <ode>
                  <slip>0</slip>
                </ode>
              </torsional>
            </friction>
            <bounce>
              <restitution_coefficient>0</restitution_coefficient>
              <threshold>1e+06</threshold>
            </bounce>
            <contact>
              <collide_without_contact>0</collide_without_contact>
              <collide_without_contact_bitmask>1</
                  collide_without_contact_bitmask>
              <collide_bitmask>1</collide_bitmask>
              <ode>
                <soft_cfm>0</soft_cfm>
                <soft_erp>0.2</soft_erp>
                <kp>1e+13</kp>
                <kd>1</kd>
                <max_vel>0.01</max_vel>
                <min_depth>0</min_depth>
              </ode>
              <bullet>
                <split_impulse>1</split_impulse>
```

```
              <split_impulse_penetration_threshold>-0.01</
                 split_impulse_penetration_threshold>
              <soft_cfm>0</soft_cfm>
              <soft_erp>0.2</soft_erp>
              <kp>1e+13</kp>
              <kd>1</kd>
            </bullet>
          </contact>
        </surface>
      </collision>
    </link>
    <model name='thrift_shop'>
      <static>1</static>
      <link name='link'>
        <collision name='collision'>
          <pose>0 0 5.69469 0 -0 0</pose>
          <geometry>
            <box>
              <size>7.21297 5.43165 11.3894</size>
            </box>
          </geometry>
          <max_contacts>10</max_contacts>
          <surface>
            <contact>
              <ode/>
            </contact>
            <bounce/>
            <friction>
              <torsional>
                <ode/>
              </torsional>
              <ode/>
            </friction>
          </surface>
        </collision>
        <visual name='visual'>
          <pose>0 -0.912491 0 0 -0 0</pose>
          <geometry>
            <mesh>
```

```
                <uri>model://thrift_shop/meshes/thrift_shop.dae</uri
                    >
              </mesh>
            </geometry>
          </visual>
          <self_collide>0</self_collide>
          <enable_wind>0</enable_wind>
          <kinematic>0</kinematic>
        </link>
        <pose>7.07781 -0.021152 0 0 -0 0</pose>
      </model>
      <static>1</static>
      <allow_auto_disable>1</allow_auto_disable>
      <pose>6.64234 -29.4603 0 0 -0 0</pose>
    </model>
    <model name='pine_tree'>
      <static>1</static>
      <link name='link'>
        <collision name='collision'>
          <geometry>
            <mesh>
              <uri>model://pine_tree/meshes/pine_tree.dae</uri>
            </mesh>
          </geometry>
          <max_contacts>10</max_contacts>
          <surface>
            <contact>
              <ode/>
            </contact>
            <bounce/>
            <friction>
              <torsional>
                <ode/>
              </torsional>
              <ode/>
            </friction>
          </surface>
        </collision>
        <visual name='branch'>
```

```
685           <geometry>
686             <mesh>
687               <uri>model://pine_tree/meshes/pine_tree.dae</uri>
688               <submesh>
689                 <name>Branch</name>
690               </submesh>
691             </mesh>
692           </geometry>
693           <material>
694             <script>
695               <uri>model://pine_tree/materials/scripts/</uri>
696               <uri>model://pine_tree/materials/textures/</uri>
697               <name>PineTree/Branch</name>
698             </script>
699           </material>
700         </visual>
701         <visual name='bark'>
702           <geometry>
703             <mesh>
704               <uri>model://pine_tree/meshes/pine_tree.dae</uri>
705               <submesh>
706                 <name>Bark</name>
707               </submesh>
708             </mesh>
709           </geometry>
710           <material>
711             <script>
712               <uri>model://pine_tree/materials/scripts/</uri>
713               <uri>model://pine_tree/materials/textures/</uri>
714               <name>PineTree/Bark</name>
715             </script>
716           </material>
717         </visual>
718         <self_collide>0</self_collide>
719         <enable_wind>0</enable_wind>
720         <kinematic>0</kinematic>
721       </link>
722       <pose>12.7869 -19.368 0 0 -0 0</pose>
723     </model>
```

```
    <state world_name='default'>
      <sim_time>34 345000000</sim_time>
      <real_time>34 872872941</real_time>
      <wall_time>1734015350 388627032</wall_time>
      <iterations>34345</iterations>
      <model name='House 1'>
        <pose>30.1864 -36.0466 0 0 -0 0</pose>
        <scale>1 1 1</scale>
        <link name='link'>
          <pose>30.1864 -36.0466 0 0 -0 0</pose>
          <velocity>0 0 0 0 -0 0</velocity>
          <acceleration>0 0 0 0 -0 0</acceleration>
          <wrench>0 0 0 0 -0 0</wrench>
        </link>
      </model>
      <model name='House 2'>
        <pose>31.5406 -18.0664 0 0 -0 0</pose>
        <scale>1 1 1</scale>
        <link name='link'>
          <pose>31.5406 -18.0664 0 0 -0 0</pose>
          <velocity>0 0 0 0 -0 0</velocity>
          <acceleration>0 0 0 0 -0 0</acceleration>
          <wrench>0 0 0 0 -0 0</wrench>
        </link>
      </model>
      <model name='antik-tabela'>
        <pose>15.4434 -42.4998 0 0 -0 0</pose>
        <scale>1 1 1</scale>
        <link name='link'>
          <pose>15.4434 -42.4998 0 0 -0 0</pose>
          <velocity>0 0 0 0 -0 0</velocity>
          <acceleration>0 0 0 0 -0 0</acceleration>
          <wrench>0 0 0 0 -0 0</wrench>
        </link>
        <link name='link_1'>
          <pose>15.4657 -45.5559 4.80447 0 -0 0</pose>
          <velocity>0 0 0 0 -0 0</velocity>
          <acceleration>0 0 0 0 -0 0</acceleration>
          <wrench>0 0 0 0 -0 0</wrench>
```

```
        </link>
        <link name='link_1_0'>
          <pose>18.8202 -45.9795 2.55398 0 -0 0</pose>
          <velocity>0 0 0 0 -0 0</velocity>
          <acceleration>0 0 0 0 -0 0</acceleration>
          <wrench>0 0 0 0 -0 0</wrench>
        </link>
      </model>
      <model name='ground_plane'>
        <pose>0 0 0 0 -0 0</pose>
        <scale>1 1 1</scale>
        <link name='link'>
          <pose>0 0 0 0 -0 0</pose>
          <velocity>0 0 0 0 -0 0</velocity>
          <acceleration>0 0 0 0 -0 0</acceleration>
          <wrench>0 0 0 0 -0 0</wrench>
        </link>
      </model>
      <model name='pine_tree'>
        <pose>12.7869 -19.368 0 0 -0 0</pose>
        <scale>1 1 1</scale>
        <link name='link'>
          <pose>12.7869 -19.368 0 0 -0 0</pose>
          <velocity>0 0 0 0 -0 0</velocity>
          <acceleration>0 0 0 0 -0 0</acceleration>
          <wrench>0 0 0 0 -0 0</wrench>
        </link>
      </model>
      <model name='ziraat'>
        <pose>6.64234 -29.4603 0 0 -0 0</pose>
        <scale>1 1 1</scale>
        <link name='link'>
          <pose>6.64234 -29.4603 0 0 -0 0</pose>
          <velocity>0 0 0 0 -0 0</velocity>
          <acceleration>0 0 0 0 -0 0</acceleration>
          <wrench>0 0 0 0 -0 0</wrench>
        </link>
        <link name='link_0'>
          <pose>10.1239 -32.6342 4.82656 0 -0 0</pose>
```

```
          <velocity>0 0 0 0 -0 0</velocity>
          <acceleration>0 0 0 0 -0 0</acceleration>
          <wrench>0 0 0 0 -0 0</wrench>
        </link>
        <model name='thrift_shop'>
          <pose>13.7202 -29.4815 0 0 -0 0</pose>
          <scale>1 1 1</scale>
          <link name='link'>
            <pose>13.7202 -29.4815 0 0 -0 0</pose>
            <velocity>0 0 0 0 -0 0</velocity>
            <acceleration>0 0 0 0 -0 0</acceleration>
            <wrench>0 0 0 0 -0 0</wrench>
          </link>
        </model>
      </model>
    </state>
  </world>
</sdf>
```

ÖZGEÇMİŞ

Tuğba FIÇICI

A. EĞİTİM

Yüksek Lisans: İstanbul Sabahattin Zaim Üniversitesi, Lisansüstü Eğitim Enstitüsü Bilgisayar Bilimi ve Mühendisliği Bölümü, 2024, İstanbul

Lisans: İstanbul Sabahattin Zaim Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Bilgisayar Mühendisliği Bölümü, 2021, İstanbul

B. MESLEKİ DENEYİM

Ocak 2022-Şubat 2023, Outsmart OOH- Yazılım Geliştiricisi

Mart 2023-Devam Ediyor, İstanbul Takas ve Saklama Bankası A.Ş.- Linux Sistem Yönetimi Uzman Yardımcısı

C. YAYINLARI

Tuğba Fıçıcı, Zehra Tüfekçi, Burak Ağgül, Seçkin Canbaz, Gökhan Erdemir, "Özerk Gezgin Bir Robot İçin Bilinmeyen Bir Ortamda Nesne Arama ve Haritalama Uygulaması", *Fırat Üniversitesi Uzay ve Savunma Teknolojileri Dergisi 1(1)*, 160- 167, 2022.