T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

Eş Zamanlı Konumlandırma, Haritalandırma Uygulamaları ve Nesne Tanıma Tabanlı Konumlandırma

Ahmet Murat ERTURAN

YÜKSEK LİSANS

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Aralık-2019
KONYA

TEZ KABUL VE ONAYI

Ahmet Murat ERTURAN tarafından hazırlanan "Eş Zamanlı Konumlandırma, Haritalandırma Uygulamaları ve Nesne Tanıma Tabanlı Konumlandırma" adlı tez çalışması 19/12/2019 tarihinde aşağıdaki jüri tarafından oy birliği / ~~oy çokluğu~~ ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Dr. Öğr. Üyesi Burak YILMAZ	.....
Danışman Doç. Dr. Seyfettin Sinan GÜLTEKİN	.....
Üye Doç. Dr. Ömer AYDOĞDU	.....

Yukarıdaki sonucu onaylarım.

Prof. Dr. Saadettin Erhan KESEN
Enstitü Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Ahmet Murat ERTURAN

Tarih: 19/12/2019

ÖZET

YÜKSEK LİSANS

EŞ ZAMANLI KONUMLANDIRMA, HARİTALANDIRMA UYGULAMALARI VE NESNE TANIMA TABANLI KONUMLANDIRMA

Ahmet Murat ERTURAN

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Seyfettin Sinan GÜLTEKİN

2019, 87 Sayfa

Jüri
Doç. Dr. Seyfettin Sinan GÜLTEKİN
Dr. Öğr. Üyesi Burak YILMAZ
Doç. Dr. Ömer AYDOĞDU

Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM) problemi, robotun bilinmeyen bir ortamın haritasını oluştururken aynı anda bu ortamda kendisini konumlandırması olarak tanımlanır. LIDAR (Light Detection and Ranging) sensörü gibi bir mesafe sensörü ve teker-açı değerlerinin alındığı odometri sensörü kullanılarak robot ortamın haritasını ve bu ortamda konumunu tahmin eder. Açık alanlarda konum bilgisi Küresel Konumlama Sistemi (GPS) ile yüksek doğruluk oranıyla belirlenebilirken kapalı ortamlarda GPS verilerinin alınamamasından dolayı konum bilgisi belirlenemez. Kapalı ortam konumlandırması özellikle savunma teknolojileri için önemli bir konudur.

Robot kapalı bir ortamda sensör verilerini girdi olarak alır ve konum tahmini çıktısı verir. Bu tahmini çeşitli Bayes temelli tahmin yöntemleri kullanarak yapar. En sık kullanılan tahmin yöntemlerinden biri Gauss Dağılımı aranmayan Parçacık Filtresi (PF)' dir. PF temelli Monte Carlo Localization (MCL) yöntemi kullanılarak ortamda konum tahmini yapılır. Robot odometri sensör verisinde hata veya giderilemeyen bir gürültü olmadığı takdirde başarılı tahminler ortaya koyar. Ancak ortamda teker kayması, robot kaçırılması, engele takılma gibi bir problemde robot teker açı verilerini kaybederek konum tahminini yanlış yapar. Bu tez çalışmasında kapalı ortamlarda meydana gelebilecek bir teker verisi kaybında robotun konumlandırılması konusunda nesne tanıma tabanlı, yenilikçi bir yöntem önerilmiştir.

Bir derin öğrenme modeli olan Faster R-CNN ile eğitilen veri seti sonucunda tanınan iki adet nesnenin konumuna göre robotun konumlandırılması amaçlanmıştır. Bu amaç doğrultusunda yapılan deneysel sonuçlar verilmiş ve robotun klasik tahmin yöntemlerinin başarısız olduğu hatalı teker verisi durumlarında başarılı konum tahmini gerçekleştirdiği görülmüştür. Tez içeriğinde ayrıca bir kapalı ortamın odometri verisi kullanılmadan haritalandırılması ve ölçeklendirilmesi uygulaması yapılmıştır. Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM) konusunda literatürde en çok kullanılan iki yöntem olan Gmapping ve HectorSLAM yöntemleri uygulamaları yapılmış ve bu yöntemlerin avantaj, dezavantajları belirtilmiştir. Savunma teknolojileri ve savunma sistemleri entegrasyonunda önemli bir konu olan kapalı ortam konumlandırması ve kapalı ortam haritalandırması konuları ele alınmış ve önerilen yöntemlerin başarılı sonuçlar verdiği görülmüştür.

Anahtar Kelimeler: Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM), Otonom Robotlar, Savunma Teknolojileri ve Savunma Sistemleri Entegrasyonu, Parçacık Filtresi (PF), Monte Carlo Lokalizasyonu (MCL), Derin Öğrenme, LIDAR (Light Detection and Ranging)

ABSTRACT

MS THESIS

SIMULTANEOUS LOCALIZATION, MAPPING APPLICATIONS AND OBJECT RECOGNITION BASED LOCALIZATION

Ahmet Murat ERTURAN

Konya Technical University
Institute of Graduate Studies
Department of Electrical and Electronics Engineering

Advisor: Assoc. Prof. Dr. Seyfettin Sinan GÜLTEKİN

2019, 87 Pages

Jury
Assoc. Prof. Dr. Seyfettin Sinan GÜLTEKİN
Asst. Prof. Dr. Burak YILMAZ
Assoc. Prof. Dr. Ömer AYDOĞDU

Simultaneous Localization and Mapping (SLAM) problem is defined as the robot positioning itself while simultaneously creating a map of an unknown environment. Using a distance sensor such as the LIDAR (Light Detection and Ranging) sensor and an odometry sensor from which the wheel-angle values are taken, the robot estimates the map of the environment and its location. Location information in outdoor environment can be determined with the Global Positioning System (GPS) with high accuracy, while indoor environment cannot be determined due to the lack of GPS data in indoor environments. Indoor environment localization is an important issue especially for defense technologies.

In a indoor environment, the robot receives sensor data as input and outputs a position estimate. It makes this prediction using various Bayes-based prediction methods. One of the most commonly used estimation methods is the Particle Filter (PF), which is not required for Gaussian Distribution. PF based Monte Carlo Localization (MCL) method is used to estimate the location of the environment. Robot odometry makes successful predictions if there is no error or irreversible noise in the sensor data. However, in the event of a problem such as wheel slippage, robot hijacking and obstruction, the robot loses angle data and makes the position estimation incorrect. In this thesis, an innovative method based on object recognition has been proposed for positioning the robot in a loss of wheel data that may occur in closed environments.

The Faster R-CNN, a deep learning model, aims to position the robot according to the position of two recognized objects. For this purpose, experimental results are given and it is seen that the robot performs a successful position estimation in case of faulty wheel data where classical prediction methods fail. In the thesis, mapping and scaling of a closed environment without using the odometry data was performed. Gmapping and HectorSLAM methods, which are the two most commonly used methods in Simultaneous Localization and Mapping (SLAM) in the literature, have been applied and advantages and disadvantages of these methods have been indicated. Indoor positioning and indoor mapping, which is an important issue in the integration of defense technologies and defense systems, were discussed and the proposed methods proved to be successful.

Keywords: Simultaneous Localization and Mapping (SLAM), Autonomous Robots, Defense Technologies and Defense Systems Integration, Particle Filter (PF), Monte Carlo Localization (MCL), Deep Learning, LIDAR (Light Detection and Ranging)

ÖNSÖZ

Bu tez çalışmasında öncelikle değerli bilgi ve yardımlarını esirgemeyen ve bu süreçte desteklerini hep yanımda hissettiğim danışman hocam Sayın Doç. Dr. Seyfettin Sinan GÜLTEKİN' e teşekkür ederim. Kendisinden çok şey öğrendiğim ve bu tez çalışmasına büyük katkı sağlayan kıymetli hocam Dr. Öğr. Üyesi Akif DURDU' ya katkılarından dolayı çok teşekkür ederim. RACLAB ekip arkadaşlarıma ve katkılarından dolayı çalışma arkadaşım Sayın Abdullah YUSEFI' ye teşekkür ederim.

Hayatımın her döneminde desteklerini esirgemeyen annem, babam, ablama ve her konuda olduğu gibi bu çalışmada da desteğini hep hissettiğim sevgili eşim Elif Merve ERTURAN' a teşekkür ederim.

Ahmet Murat ERTURAN
KONYA- 2019

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Simgeler

$\hat{x}_k$	:k anındak durum kestirimi
$\widetilde{\omega}^{(i)}$	:i parçacığın normalize edilmiş ağırlığı
r_1, r_2	:1 ve 2. Trafik işaretlerinin robota olan uzaklıkları
$u_{1:t}$	:Kontrol matrisi
$z_{1:t}$	:t anına kadar ölçüm vektörü
x_1, y_1	:1. Trafik işaretinin genel haritadaki konumu
x_2, y_2	:2. Trafik işaretinin genel haritadaki konumu
x_k	:k anındaki durum vektörü
$x_t^{[p]}$	:Parçacık zaman adımı
x_t	:Robotun t anındaki konumu
Σ_t	:Koveryans
δ_t	:Ölçüm gürültüsü
ε_t	:Gauss gürültüsü
$bel(x_t)$	:t anındaki inanç
C	:Işık hızı
D	:Mesafe (m)
K	:Kalman Kazancı
M	:Harita
T	:Zaman (sn)
$M(s_i(\xi))$	:Tarama bitiş noktalarının global haritadaki koordinatları
Ne_{ff}	:Parçacık öncül yörünge ağırlığı
η	:Normalizasyon faktörü
μ	:Ortalama değer

Kısaltmalar

ADAS	:Autonomous Driving Assistant System (Otonom Sürüş Destek Sistemi)
AMCL	:Adaptive Monte Carlo Localization
AutoCAD	:Computer Aided Design (Bilgisayar Destekli Tasarım)
CNN	:Convolutional Neural Network (Evrişimsel Sinir Ağı)
GKF	:Genişletilmiş Kalman Filtresi
FastSLAM	:A Factored Solution to the SLAM
GMCF	:Generalized Monte Carlo Flux
GPS	:Global Positioning System (Küresel Konumlama Sistemi)
GPU	:Graphics Processing Unit (Grafik İşlem Birimi)
ICP	:Iterative Closest Point
IMU	:Inertial Measurement Unit
IR	:Infrared
İHA	:İnsansız Hava Aracı
İYA	:İnsansız Yer Aracı
KF	:Kalman Filtresi
LIDAR	:Light Detection and Ranging
mAP	:Mean Average Precission
MCL	:Monte Carlo Localization
PF	:Parçacık Filtresi
QR	:Quick Response

RACLAB	:Robotic Automation Laboratory (Robotik Otomasyon Laboratuvarı)
RADAR	:RAdio Detecting And Ranging
RBPF	:Rao-Blackwellization Particle Filter
ROS	:Robot Operating System
Rviz	:ROS visualization (ROS Görselleştirme)
SLAM	:Simultaneous Localization and Mapping (Eş Zamanlı Konumlandırma ve Haritalandırma)
SONAR	:Sound Navigation and Ranging
vSLAM	:Visual SLAM (Görsel SLAM)
YSA	:Yapay Sinir Ağları
ReLU	:Rectified Linear Unit

1. GİRİŞ

Gelişen teknolojiyle beraber robotik ve otonom sistemler, insan hayatında önemli bir konu haline gelmiştir. Savunma Sanayi başta olmak üzere üretim sanayi, tıp ve uzay bilimleri alanlarında robotik sistemlerin, insan gücünü azaltarak yüksek doğrulukta çalışması kullanım alanlarını yaygınlaştırmıştır. Tanım olarak robot; bilgisayar algoritmalarına bağlı olarak çalışan, mekanik ve programlanabilir cihazlardır. İnsanlık tarihinde uzun yıllardır olan bu kavram, yazılı olarak M.Ö. 800 yılında Homeros'un Ilyada eserinde hareket edebilen üç bacaklılardan bahsetmesine kadar dayanmaktadır. 13. Yüzyılda Türk mühendis El-Cezeri çeşitli mekanik araçlar tasarlamış ve insansız çalışabilen robotlar fikrini ortaya atmıştır (Kitab-ul Hiyel). 1800'lerin sonunda Nikola Tesla uzaktan kumanda ile Elektromanyetik Dalgaları kullanarak birçok aracı kontrol etmiştir. İlk defa Karel Čapek tarafından 1920 yılında "robot" kelimesi kullanılmıştır. 1947 yılında Alan Turing kendi kendine düşünebilen akıllı makineler fikrini ortaya atmıştır. 1960 da "Shakey" robotu kendisiyle aynı ortamda bulunan nesnelerden kaçarak onlara çarpmadan ilerleyebilmiştir. MIT' nin ürettiği Robot Cob (1993), NASA'nın "Sojouner" uzay robotu (1997) ve General Motors-NASA ortak üretimi olan "Robonaut-2" ile robotik alanındaki gelişmeler artarak devam etmiştir. 2000'li yıllardan günümüze kadar geçen sürede artan işlemci güçleri ve programlama yetkinliği ile robotik sistemlerde devrimsel gelişmeler yaşanmıştır. Otonom sistemlerin robotik alanına entegrasyonu sonrasında İnsansız Hava Araçları (İHA) başta olmak üzere Otonom Sürüş Destek Sistemleri (ADAS) gibi birçok alanda, insansız hareket edebilen ve karar verebilen mekanizmalar tasarlanmıştır.

Robotların ortamdaki nesneleri ve çeşitli çevresel değişkenleri algılayabilmesini sağlayan elektronik devre elemanlarına sensör denir. Sensörler sıcaklık, basınç, ivme-hız, nesne mesafesi gibi birçok fiziksel büyüklüğü elektriksel sinyallere dönüştürürler. Bu sayede robot, ortamdaki değişkenlerin verilerini toplayarak bulunduğu çevre hakkında bilgi sahibi olur. Robotik sistemlerde çoğunlukla LIDAR sensör, Sonar Sensör, IMU (Inertial Measurement Unit), Radar, RGB kamera ve GPS (Küresel Konumlandırma Sistemi) gibi sensör çeşitleri kullanılmaktadır. Sensörlerden alınan veriler ile robot bulunduğu ortamın haritasını oluştururken, konumunu belirler ve etrafında olan nesnelere çarpmadan ilerleyebilir. Otonom hareket eden robotlar, sensörlerden aldıkları verileri yapay zeka teknikleri ve çeşitli tahmin yöntemleri kullanarak daha doğru sonuçlara

ulaşmayı amaçlarlar. Otonom hareket eden robotlarda ortam sensörler aracılığıyla keşfedilir ve karar mekanizmaları robotların nasıl bir hareket gerçekleştireceğini belirler. Bunu belirlerken çoğunlukla Kalman Filtresi, Parçacık Filtresi gibi Bayes Teoremi temelli tahmin filtreleri kullanılır.

Robotik sistemlerin gelişmeleriyle beraber Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM), yol planlaması, nesne tanıma ve takip etme gibi problemler ortaya çıkmıştır. Bu problemlerin çözümü ışığında geliştirilen matematiksel modeller ve algoritmalar (Genişletilmiş Kalman Filtresi (GKF) ve Parçacık Filtresi (PF) tahmin yöntemleri) literatürde birçok çalışmada kullanılmıştır.

Bu tez kapsamında robotun otonom hareket sırasında konumunun belirlenmesi ve SLAM algoritmaları kullanılarak kapalı mekanların haritalandırılıp ölçeklendirilmesi amaçlanmıştır.

1.1. Tezin Amacı

Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM) konusu hem haritalandırma hem de lokalizasyon konularını beraber ele aldığından dolayı robotikte önemli bir problem olarak bilinmektedir. Bu tezin öncül amacı savunma sistemlerine entegre olabilecek bir kapalı ortam konumlandırma yöntemi sunmaktır. Buna ek olarak SLAM algoritmaları kullanılarak oluşturulan bir ortamda robotun konum tahminini yanlış yapması durumunda lokalizasyonu belirleyecek bir yöntem ortaya koymaktır. Robot, ortamda SLAM yöntemlerini kullanarak çevrenin haritasını oluşturur ve işaret eşleştirme ile konum tahmini yapar. Bu konum tahminiyle beraber robot yol planlaması yapar hedef noktasına hareket eder. Yol planlaması robotun önceden çıkarılmış ortam haritasına göre belirli bir noktadan başka bir noktaya en kısa yoldan, engel vb. bir nesneye takılmadan ilerlemesi konusudur. Herhangi bir kayma veya engele takılma olmadığı sürece robot otonom bir şekilde kalkış noktasından hedef noktasına varmaktadır. Ancak odometri ve teker kayması gibi problemlerde robotun konum tahmini doğru bir şekilde yapılamaz. Hata kümülatif olarak artarak konumun yanlış tahminine neden olur.

Savunma sistemlerinde kullanılacak bir otonom robot için konumlandırma önemli bir problemdir. Bu tez kapsamında parçacık filtresi tabanlı Gmapping algoritması kullanılarak oluşturulan ortam haritasında robotun konum tahmini öncelikle Monte Carlo Lokalizasyon (MCL) yöntemleri kullanılarak yapılır. MCL yöntemi herhangi bir teker kayması olmadığı takdirde çok başarılı konum tahmini gerçekleştirmektedir. Ancak

herhangi bir teker verisi kaybı olduğu takdirde ortamda konum tahmini hatalı olur. Bu nedenle önerilen matematiksel metot robotun teker kayması veya konum değişikliği gibi hatalarda ortaya çıkan yanlış konum bilgisini düzeltme amacı taşımaktadır. Deneysel çalışmalarda haritada bulunan ve konumları bilinen iki nesneyi tanıyarak robotun gerçek konumunu belirlemesi amaçlanmıştır. Nesneleri tanımak ve hangi nesnenin haritada hangi konumda olduğunu belirlemek amacıyla bir Derin Öğrenme ağı kullanılmıştır. Bu sayede robot gördüğü nesnenin haritadaki konumunu bilmekte ve bu konuma göre matematiksel modeli kullanarak gerçek konumunu hesaplamaktadır. Önerilen bu yöntemle yapılan deneysel çalışmalarda robotun teker verisi kaybında, gerçek konumunu MCL yöntemine nazaran daha doğru bir şekilde belirlediği görülmüştür.

Tezin bir diğer amacı SLAM algoritmaları kullanılarak haritalandırılan bir iç mekânın ölçeklendirilmesidir. Encoder teker verileri kullanılmadan sadece LIDAR sensör yardımıyla geleneksel yöntemlere göre daha kısa bir sürede kapalı mekânın haritalandırılması ve ölçeklendirilmesi amaçlanmıştır.

1.2. Tezin Önemi

Robotik sistemler birçok alanda yaygın bir şekilde kullanıldığı gibi savunma sistemleri alanında da kullanılmaktadır. Savunma sistemlerinde robotların kullanılmasıyla beraber hem insan hayatı korunabilmekte hem de insan gücünün yetersiz kaldığı durumlarda daha efektif sonuçlar alınmaktadır. Özellikle insansız hava ve yer araçları sayesinde keşif, nesne/insan tespiti ve takibi sistemleri gelişmiş ve kullanımı yaygınlaşmıştır.

Bu tezin önemi savunma sistemlerinde insansız olarak hareket eden ve takip ettiği yolu üzerindeki nesneleri tanıyarak bu nesnelere göre gerçek konumunu belirleyebilen bir robot ortaya koymaktır. Her mekanik sistem gibi otonom sistemler de enerjiye ihtiyaç duyar. Bu enerjinin en verimli şekilde kullanılması amaçlanmalıdır. Bir noktadan başka bir noktaya otonom olarak ilerleyecek bir robotun en kısa yolu bulması ve ilerlemesi pil sarfiyatını en aza indirerek hedefe varması anlamına gelmektedir. Buradan hareketle tezin bir diğer amacı da önemli bir güç tasarrufu sağlamaktadır.

Kapalı ortamlarda GPS (Küresel Konumlandırma Sistemi) verilerini alamayan robotların konumlandırılması son derece önemli bir konudur. Otonom olarak ilerleyen robotun tahmin filtreleri sayesinde konumu haritaya göre tahmin edilebilir. Ancak sensör verileri hatalarından dolayı yanlış tahminler oluşabilmektedir. Özellikle savunma

sanayide, insansız bir mekanizmanın hata yapması büyük problemlere yol açabilmektedir. Dolayısı ile yapılan çalışmalarla bu hataların en aza indirilmesi amaçlanmıştır.

Tez çalışmasında insansız bir yer aracı başlangıç noktasından hedef noktasına, bir yol planlaması yaparak ulaşmıştır. Bu hedefe giden yolda oluşabilecek sensör hatalarında robotun "Ben neredeyim?" sorusuna cevap verebilmesi gerekmektedir. Tezde önerilen yöntem sayesinde robot iki nesneyi tanıyarak bu nesnelere göre kendi konumunu belirlemektedir. Bu iki nesnenin ortamdaki konumunun bilinmesi durumunda robotun kendi konumunun, önerilen bir yöntem ile tahmin etmesi amaçlanmıştır. Ayrıca kapalı bir ortamın haritalandırılmasının yanı sıra ölçeklendirilmesi işlemi de yapılmıştır. Bu işlem geleneksel yöntemlerle uzun süre ve iş gücü gerektirirken önerilen yöntem ile az iş gücüyle ve kısa bir zaman diliminde gerçekleştirilmiştir.

1.3. Tezin Organizasyonu ve Bölümleri

Tez çalışması kapsamında; SLAM hakkında bilgiler (Bölüm 1), konu ile ilişkili literatür taraması (Bölüm 2), kullanılan materyal ve yöntemlerin tanıtıldığı SLAM, Yapay Sinir Ağları ve Derin Öğrenme, ROS (Robot İşletim Sistemi) gibi sistemlerin anlatıldığı kısım (Bölüm 3), yapılan araştırma ve deneysel çalışmalar ile önerilen yöntemin ortaya koyduğu sonuçlar (Bölüm 4), sonuçlar ve öneriler (Bölüm 5) bölümleri bulunmaktadır.

Bölüm 3.1'de SLAM konusu ele alınmıştır. SLAM'in nerede kullanıldığı ve matematiksel ifadelerine yer verilmiştir. SLAM problemi literatürde önemli bir yer edinmektedir. Otonom bir robot hareketi sağlamak için ortam haritasının ve etraftaki nesnelerin konumlarının bilinmesi gerekmektedir. Bu bağlamda SLAM ile robota aslında bir otonomi kazandırılmış olur. Ancak bu otonomi, odometri verisine bağlıdır. Odometri verisi olmadığı zaman robotun ortamda ne kadar ilerlediği ve nerede olduğu yanlış tahmin edilebilir. Modern SLAM yöntemlerinden görsel SLAM konusuna değinilmiş ve kullanılan sensör çeşitlerine örnekler verilmiştir. SLAM yöntemlerinden Gmapping ve HectorSLAM algoritmaları üzerinde durulmuş ve bu algoritmaların nasıl sonuçlar verdiği, hangisinin tercih edildiği ve nedeni detaylı olarak anlatılmıştır. Bölüm 3.2'de robotik sistemlerde yaygın olarak kullanılan Bayes tahmin filtreleri ele alınmıştır. Bayes Filtrelerinin çeşitleri ve hangisinin daha başarılı sonuçlar araştırılmıştır. Ayrıca konum tahmini için tez kapsamında kullanılan Monte Carlo Localization (MCL) algoritması da detaylandırılmıştır. Bölüm 3.3'te derin öğrenme ile nesne tanıma yöntemleri ele

alınmıştır. Derin öğrenmenin tarihsel gelişimi ve tez kapsamında hangi modelin neden kullanıldığı detaylandırılmıştır. Altı farklı nesneden alınan görüntülerin Faster R-CNN modeli ile eğitildiği ve yüksek doğrulukla tanınabildiği bilgileri verilmiştir. Bölüm 3.4'te önerilen matematiksel yöntem gösterilmiştir. İki adet çemberin kesişim noktalarının bulunmasını baz alan yöntem ile robotun, iki nesnenin konumuna göre kendisini konumlandırabilmesi amaçlanmıştır. Bölüm 3.5'te otonom hareketi oluşturan ve belli bir başlangıç noktasından hedef bitiş noktasına en kısa yol planlaması algoritmasından bahsedilmiştir. Bölüm 3.6'da ise tez kapsamında gerçek zamanlı olarak deneylerin yapılmasını sağlayan Robot İşletim Sistemi (ROS) anlatılmıştır. Neden ROS tercih edilmiştir ve ne gibi katkılar sunmuştur sorularına cevap verilmiştir.

Bölüm 4'te tez kapsamında yapılan deneysel çalışmalar anlatılmıştır. Nesne tanıma tabanlı bir yöntem kullanılarak kapalı ortamda robot konumlandırması ve iç mekân haritalandırılıp ölçeklendirilmesi üzerine deneysel çalışmalar yapılmıştır. Öncelikle Bölüm 4.1'de nesne tanıma tabanlı bir konumlandırma yöntemi gösterilmiştir. Deney ortamının haritasının çıkarılması ve SLAM uygulamasının yapılması Bölüm 4.1.1'de gösterilmiştir. Gmapping algoritması kullanılarak ortam haritası çıkarılmış ve robotun otonom hareket ettirilmesi sağlanmıştır. Bölüm 4.1.2'de önerilen yöntemin nasıl uygulandığı ve algoritma akış şeması verilmiştir. Bölüm 4.1.3'te önerilen nesne tanıma yöntemi kullanılarak robotun konumu 3 farklı deneyde kaydedilmiştir. İlk deneyde odometri verisi kaybolmadan robot başlangıç noktasından bitiş noktasına kadar en kısa yoldan ulaşmış ve MCL yöntemi ile konum tahmini doğru yapılmıştır. Aynı şekilde nesne tanıma yöntemi ile konum tahmini de doğru yapılmıştır. İkinci deneyde ise odometri verisi kaybedilmiştir ve birinci nesne hizasında robot otonom ilerlemeye başlamıştır. Odometri bilgisi olmadığından dolayı robot MCL yöntemi ile konum tespitini yanlış yapmıştır. Önerilen nesne tanıma yöntemiyle ise konum tespiti yüksek doğruluk oranıyla gerçekleştirilmiştir. Deney sonunda robotun anlık olarak oluşturduğu konum çıktısı bölüm içerisindeki şekillerde görülmektedir. Üçüncü deneyde ise yine odometri verisi kaybedilmiştir ve robot bu kez sağda bulunan diğer bir nesnenin hizasında ilerlemiştir. MCL yöntemi ile odometri verisi yanlış olduğundan dolayı robot konum tahmini yine yanlış yapılmıştır. Ancak önerilen yöntem ile konum başarılı bir şekilde tespit edilmiştir. Bölüm içerisindeki şekillerde robotun konum bilgisi gösterilmiştir ve doğru tahmin oranları verilmiştir. Bölüm 4.2'de HectorSLAM algoritması kullanılarak kapalı bir mekânın haritalandırılması ve ölçeklendirilmesi çalışması yapılmıştır.

Son olarak Bölüm 5 de tez kapsamında yapılan çalışmanın sonuçları verilmiştir. Araştırmanın sonuçlarının literatüre katkısı anlatılmıştır. Ayrıca tez çalışmasının devamında yapılabilecek çalışmalar öneriler kısmında detaylandırılmıştır.

2. KAYNAK ARAŞTIRMASI

Robotik sistemlerde SLAM (Eş Zamanlı Konumlandırma ve Haritalandırma) konusu önemli bir problemdir. Özellikle kapalı ortamlarda GPS (Küresel Konumlandırma Sistemi) konum bilgisine erişilemediğinden dolayı haritalandırmanın yanında konumlandırma problemini çözmesi bu alanda çalışmaların artmasına neden olmuştur. Yirmi yıllık bir süre zarfında ortaya konan gelişmelerle beraber çeşitli sensörler ve algoritmalar kullanılmıştır. Bunun yanı sıra olasılıksal tahmin yöntemlerinin gelişimiyle beraber başarılı sonuçlar elde edilmiştir.

Literatürde bulunan araştırmaların çoğunda mesafe algılayıcı sensörler ve odometri teker verileri kullanılarak SLAM uygulamaları gerçekleştirilmiştir. Mesafe algılayıcı sensörler vasıtasıyla ortamdaki nesnelerin uzaklıkları algılanırken; teker dönüş ve açı verileriyle de robotun ortamdaki ilerleme bilgisi elde edilmiştir. Mesafe algılayıcı sensörlerden LIDAR sensörü lazer tabanlı bir sensördür ve en çok tercih edilen sensör tipidir. Bunun yanı sıra Sonar ve Radar sensörleri de mesafe algılama işlemleri için kullanılmıştır.

Son yıllarda yapılan çalışmalarda ise LIDAR sensörlerinin maliyetlerini azaltmak amacıyla Görsel SLAM (vSLAM) uygulamaları artış göstermiştir. vSLAM ile tek bir kamera kullanılarak görüntü üzerinden ortam haritası oluşturulurken aynı zamanda odometri verisi de alınmış olur. Böylelikle teker hatalarının önüne geçilirken tek sensör kullanılması ile maliyet de düşürülmüştür. Ancak bu yöntemin de ışık miktarı düşük olan veya karanlık ortamlarda çalışmaması gibi dezavantajları bulunmaktadır. Tez kapsamında mesafe algılama sensörleri ile SLAM yöntemleri üzerine yapılan araştırmalar incelenmiştir.

(Smith ve ark., 1988) tarafından yapılan çalışmada stokastik harita olarak adlandırılan önceki durumdan çıkarım yaparak bir sonraki durumu tahmin etme prosedürleri oluşturulmuştur. Haritadaki nesnelerin ilişkilerine bağlı olarak yapılan tahminler, durum tahmin filtresi teorilerine bağlı olarak geliştirilmiştir.

(Leonard ve Durrant-Whyte, 1991) mobil robotlar için açık bir konu olan haritalandırma ve konumlandırma konusunda ilk kez ele alınan önemli bir çalışma yaptılar. SLAM problemini zor bir problem olarak tanımladılar. Çünkü mobil robotun doğru bir hareket gerçekleştirebilmesi için doğru bir haritaya ihtiyacı vardır. Doğru bir harita için ise doğru bir konum bilgisine sahip olması gerekmektedir. Bu problemi

çözebilmek için servo monteli bir sonar sensör kullandılar ve ortam özelliklerinin bir alt kümesini robotun ilk konumundan öğrenebilecekleri, GKF yöntemlerinin kullanıldığı bir araç geliştirdiler.

(Thrun ve ark., 1998) yaptıkları çalışmada büyük ölçekli geometrik bir ortamda olasılıksal yöntemleri ve Kalman Filtresini (KF) kullanarak önemli bir konumlandırma ve haritalandırma uygulaması yapmışlardır. Bu yıldan sonra yapılan çalışmalarda Kalman Filtresi ve Genişletilmiş Kalman filtrelerinin SLAM problemi üzerindeki çözüm önerileri artmıştır.

(Zunino ve Christensen, 2001) tarafından yapılan çalışmada mobil robot üzerine yerleştirilen sonar sensörlerden alınan bilgiler Genişletilmiş Kalman Filtresi (GKF) kullanılarak işlenmiş ve oluşturulan güçlü algoritma ile bir ev ortamında konumlandırma ve haritalandırması gerçekleştirilmiştir. Kalman Filtresi (KF) ve Genişletilmiş Kalman Filtresi (GKF)'nin SLAM uygulamalarındaki kullanımının artmasının nedeni; büyük oranda yakınsama sağlaması ve belirsiz durumlarda doğru tahmin oluşturmasıdır. Ancak GKF'nin Gauss dağılımları yaklaşımı temelli olması ve doğrusallaştırma gibi mecburiyetlerinin olmasından dolayı işlem yükü ve süresi bakımından dezavantaja sahiptirler.

(Martinelli ve Siegwart, 2007) tarafından yapılan çalışmada ortaya koydukları yöntem sonucunda doğrusallaştırmanın meydana getirdiği tahmin süresindeki artış problemi aşılmış ve eşzamanlı gözlenmeyen özelliklerin korelasyon hesaplaması problemi çözülmüştür. Kalman Filtreleri yüksek doğrulukla kestirimler yapsa da sadece Gauss olasılıklarını modelleyebilmektedir. Bu durumda Gauss dağılımlı olmayan ve doğrusal olmayan durumların tahminini güçleştirmektedir. Parçacık Filtreleri ise Gauss dağılımlarına bağlı olmadan tahmin gerçekleştirebilmektedir ve bu yüzden SLAM uygulamalarında daha fazla tercih edilmiştir.

(Montemerlo ve ark., 2002) yılında yaptıkları çalışmada Eş Zamanlı Konumlandırma ve Haritalandırma sorununa yeni ve etkili bir çözüm olan FastSLAM algoritmasını sunmuşlardır. FastSLAM algoritması Parçacık Filtresine dayanan bir tahmin yöntemi kullanmaktadır. Sonraki yıllarda yapılan bir kısım SLAM çalışmasına öncülük eden bu algoritma Rao-Blackwellization metodunun kullanılması ile oluşturulmuştur. Rao-Balckwellizion Parçacık Filtresi (RBPF) SLAM problemini çözmek için etkin bir araç olmuştur. Ancak her parçacık ortamın ayrı bir haritasını taşıyan

parçacık filtreleri kullanmaktadır. Bu da parçacık sayısının fazla olması anlamına gelmektedir.

(Grisetti ve ark., 2007) tarafından yapılan çalışmayla RBPF ile artan parçacık sayısı, örnekleri doğru tahmin ederek azaltılmıştır. Bu algoritma ile parçacık filtresinin gereksiz yeniden örnekleme işlemi sayısı azalmış ve parçacık tükenme riski büyük ölçüde ortadan kaldırılmıştır.

SLAM algoritmalarında mesafe algılayıcı sensörler ve kilometre sayacı sensörleri büyük bir öneme sahiptir. Mesafe algılayıcı sensörler ile ortamdaki nesnelerin robota olan uzaklıkları belirlenirken; kilometre sayacı sensörleri ile de robotun ortamda ilerlediği mesafe açı değerleri bilgisi elde edilir. Mesafe algılayıcı sensörlerden en çok tercih edilen sensör LIDAR sensörüdür. Nokta bulutu şeklinde bir lazer ışın demeti saçarak etrafındaki nesnelerin uzaklıklarını bir bütün olarak belirleyebilir.

Robotun ilerlemesini ve teker açı değerlerini alabilmesini sağlayan odometri sensörleri de bulunmaktadır. Bu sensörler sayesinde robot ilerlediği mesafeyi ve tekerlerin hangi açılarda ilerlediği bilgisini edinir. Odometri bir kör sayımdır (Dead-Reckoning). Yani teker çevresi ve açısı kadar ilerleme robotun aldığı mesafe bilgisine denk gelir. Tekerin boşta dönmesi ya da kayması gibi bir problemi algılayamazlar ve robotun o mesafeyi aldığını varsayarlar. Sensörlerin hatalı veriler elde etmesi sonucunda SLAM problemleri ortaya çıkmaktadır. LIDAR sensörleri lazer ışın temelli çalıştıklarından dolayı toz, duman bulunan ortamlarda ışın kırılması yaşar ve yanlış sonuçlar elde edilebilir.

(Marck ve ark., 2013) tarafından yapılan çalışmada yoğun duman olan bir ortamda SLAM yapılmıştır. Ancak lazer ışınlarının dumanda kırılmasıyla haritalandırma ve konumlandırma hataları oluşmuştur. Bu problemi ortadan kaldırabilmek için bir sonar sensör çemberi tasarlanmış ve duman altındaki bölümde ses dalgalarıyla haritalandırma yapılmıştır. Gerçek zamanlı olarak bir okul kantininde yapılan deneylerde duman altındaki ortam haritalandırılmıştır. Iterative Closest Point (ICP) ve Generalized Monte Carlo Flux (GMCF) metodları kullanılmış ve ortamın yoğun duman altındaki kısmı da haritalandırılmıştır.

(Santos ve ark., 2015) tarafından yapılan bir çalışmada ise yine benzer bir konu ele alınmıştır. Duman altındaki bir ortamda Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM) yapılmıştır. LIDAR verilerinin dumandan etkilenmesinden dolayı bozulan harita sezgisel yöntemlerle ve bulanık mantık kullanılarak düzeltilmiştir.

LIDAR, sonar ve duman algılayan bir sensör kullanılarak ortamın dumanlı olan kısımlarında LIDAR dan alınan verileri azaltarak başarılı bir SLAM metodu oluşturulmuştur. Bu iki çalışmada da LIDAR nokta bulutu verilerinin alınamaması sonucunda oluşan SLAM problemi kısmen çözülmüştür. Ancak Sonar sensörün ses dalgaları yayması ve bazı nesnelerin kenar-köşe bölümlerini doğru okuyamamasından dolayı çok başarılı sonuçlar elde edilememiştir.

(Deißler ve Thielecke, 2013) yaptıkları çalışmada dumanlı bir ortamda SLAM yapılması konusunda radar kullanarak daha başarılı bir konumlandırma ve haritalandırma yapmışlardır. Odometri verisini kullanarak ortamda ilerleyen, iki alıcı ve bir vericiden oluşan yarasa tipi dizi anten tasarlayarak ortamda bulunan nesne köşelerini net bir şekilde belirleyen bir SLAM uygulaması gerçekleştirmişlerdir. Cam gibi saydam nesnelerin LIDAR lazer ışınlarını emmesi sonucunda da bir SLAM problemi oluşmaktadır.

(Zhang ve ark., 2014) tarafından yapılan çalışmada robotun, ortamda SLAM yapılırken cam bölmelerin haritasını oluşturamadığı ve bu bölümlerin tanımlanamadığı gösterilmiştir. Sonar sensörleri ile 180 derecelik bir yarım çember oluşturulmuş ve ses dalgaları kullanılarak camlı bölgenin haritası oluşturulmuştur. Adaptif Monte Carlo Localizasyonu (AMCL) ve Genişletilmiş Kalman Filtresi yöntemleri kullanılmıştır. Alınan sonuçlarda robot haritayı oluşturmuş ve otonom hareket etmeyi başarmıştır.

SLAM probleminde mesafe ölçüm sensöründen alınan veri kadar kilometre sayacı verileri de büyük öneme sahiptir. Odometri sensörleri teker dönme ve açı bilgilerini alırlar. Ancak bu bilgi, önceden belirtildiği gibi bir kör sayım sonucunda elde edilmektedir. Robotun tekerlerinin kayması, yumuşak bir zeminde boşa dönmesi gibi problemlerde robot ilerlemediği halde odometri sensör verisi robotun ilerlediği bilgisini verir. Bu problemin çözülmesi üzerine yapılan çalışmalarda tahmin filtreleri, görüntü işleme teknikleri ve derin öğrenme metotları kullanılmışır.

(Kim ve ark., 2012) tarafından yapılan çalışmada dört bölümden oluşan bir barkod sistemi geliştirilmiştir. Bu barkodlar, görüntü işleme metotları kullanılarak tespit edilmiş ve robota doğru konum bilgisi verilmiştir. Otonom hareket boyunca robotun barkodları gördüğü noktaları ve lokalizasyon bilgisini Parçacık Filtresi kullanarak tahmin algoritması üzerine yerleştirmişlerdir. Doğru konum bilgileri elde etmeyi başarmışlardır. Ancak dezavantajı, çok büyük alanlarda çok miktarda barkod bulunması gerekmektedir.

(Xiong ve ark., 2016) tarafından yapılan çalışmada QR kodlar kullanılmıştır. Robot üzerine bir kamera yerleştirilmiştir ve bu kamera ile tavanda 2.35 m yüksekliğe

konulmuş kare kodlar okunarak konum bilgisi elde edilmiştir. Odometri verisinin hatası kümülatif olarak toplanmış ve bir eşik değerini aşması durumunda QR kod ile alınan konum bilgisi tercih edilmiştir. LIDAR, kamera ve odometri sensörleri kullanılmış ve SLAM uygulaması gerçekleştirilmiştir. Otonom hareket sırasında yol planlaması A Star algoritması tarafından gerçekleştirilmiştir.

(Nazemzadeh ve ark., 2017) çalışmalarında engelliler için otonom ilerleyen bir tekerlekli sandalye üzerine çalışmalar yapmıştır. Bu sandalyenin hatalı bir odometri bilgisi aldığında yerde bulunan QR kodları görüntü işleme ile tanıyarak doğru konumlandırma yapması amaçlanmıştır. H filtreleri kullanılmış ve GKF, GKF, PF ye oranla daha doğru tahmin sonuçlarının yanı sıra daha hızlı bir tahmin elde edilmiştir.

SLAM probleminde haritalamanın yanında konumlandırmanın da olması problemi zor bir hale getirmektedir. Lokalizasyon problemindeki ana konular üç bölüme ayrılır:

- Küresel Konumlama
- Konum İzleme
- Robot Kaçırma

(Weiss ve ark., 1994) yaptıkları çalışmada birinci problemi ele almış ve odometri (kilometri sayacı) hatalı bilgilerinin Gauss tahmin yöntemleri ve Kalman Filtreleri kullanarak düzeltmişlerdir.

(Scaramuzza ve Siegwart, 2008) tarafından yapılan çalışmada, ikinci problem ele alınmış ve robotun başlangıç noktası hakkında bilgi olmadan yolun göreceli tahminini yapan bir algoritma geliştirilmiştir. Üçüncü problem robotik lokalizasyondaki en önemli problemdir. Robotun yol alırken konumunu kaçırması, kaldırılıp başka bir noktaya koyulması gibi durumlarda robot kilometre sayacı geri bildirimini alamaz. Kilometre sayacı sensörleri için hata kümülatif olarak artar ve konum bilgisi yanlış verilmektedir.

(Scaramuzza ve Fraundorfer, 2011) tarafından yapılan çalışmada robot kaçırılma durumunda stereo kamera kullanılarak görüntü işleme teknikleri ile görsel odometri bilgisi elde edilmiştir. Literatürde robot lokalizasyonu çalışmalarında tahmin filtreleri, Bluetooth, Wifi, GPS veya özellik çıkarma ve eşleme yöntemleri kullanılmıştır. Bu çalışmada, literatüre katkı olarak yeni bir matematiksel yöntem önerilmiştir. Bu yöntemi kullanarak; Robot, Faster R-CNN modeli ile eğitilmiş nesnelerin koordinatlarına göre konumlandırılmıştır.

Yapay Zekâ üzerine yapılan çalışmaların popülerliği son yıllarda artış göstermiş olsa da ilk araştırmalar daha eskilere dayanmaktadır. 1950'de Alan Turing tarafından yapılan "Bilgisayar Mekanizması ve Zekâ" isimli çalışmada makinelerde yapay zeka fikri ortaya atılmıştır. Bu tarihten 2000'li yılların başına kadar makine öğrenmesi üzerine yapılan çalışmalar işlem kapasitesi darlığından dolayı yavaş ilerlemiştir. Yapay Sinir Ağlarındaki (YSA) artan gizli katman sayılarıyla beraber işlem yükü daha da artış göstermiştir. Ancak 2000'li yıllardan sonra artan grafik işlem gücü ve GPU destekli işlem kabiliyet gücünden ve hesaplama maliyetlerinin düşmesinden sonra YSA üzerine yapılan çalışmalar hızlı bir artış göstermiştir.

Günümüzde yapay zekanın uzay, havacılık, hastalık tespiti gibi tıbbi alanlar ve nesne sınıflandırma gibi alanlarda kullanımı giderek artmaktadır. Yapay zeka konusundaki gelişmeler, yapay sinir ağları, makine öğrenmesi ve derin öğrenme gibi kavramlar ortaya çıkarmıştır. Derin öğrenme bir makine öğrenmesi sınıfıdır. Derin öğrenme, özellik çıkarma ve dönüştürme için birçok doğrusal olmayan işlem birimi katmanını kullanır (Şeker ve ark., 2017). Her ardışık katman, önceki katmandaki çıktıyı girdi olarak alır (Deng, 2014). Algoritmalar denetimli (sınıflandırma gibi) veya denetimsiz (desen analizi gibi) olabilir.

(Fukushima, 1980) tarafından yapılan çalışmada canlıların sinir ağlarından esinlenilerek ilk derin öğrenme mimarisi fikri ortaya atılmıştır. Neocognitron olarak adlandırılan mimari, bir öğretici olmadan öğrenme yeteneği kazanmıştır. Modelin uzamsal ağa giriş sinyali olarak verilmesi durumunda konuşma tanıma ve işitsel bilgi işlenmesi gibi uygulamaların yapılabileceği belirtilmiştir.

(LeCun ve ark., 1990) tarafından el yazısını tanıma için geri yayılım ağlarının bir uygulaması sunulmuştur. Yöntem ABD Posta Servisi posta kodu basamaklarında kullanılmış ve %1'lik düşük bir hata oranı yakalamıştır.

2000'li yıllardan sonra gelişen grafik işlem gücü sayesinde yapay zekâ gizli katmanları işlem yükü azaltılmış ve çalışmalar artmıştır. (Hinton, 2007) tarafından yapılan çalışmada öğrenme kısıtlamaları yukarıdan aşağıya bağlantılar içeren çok katmanlı bir sinir ağı kullanılarak sınıflandırmak yerine veri üretmek için eğiterek çözülebileceğini göstermiştir.

(Hinton ve ark., 2012) tarafından yapılan çalışmada eğitimde ezberleme "overfitting" problemi ele alınmıştır. Bu ezber rastgele özellik seçilmesi ile büyük oranda

azaltılmıştır. Çalışma sonucunda rastgele “dropout” ’un konuşma ve nesne tanıma için önemli eğitim başarısı sağladığı görülmüştür.

Evrişimsel Sinir Ağları (CNN) nesneyi tanımak için görüntüyü çeşitli katmanlarda işler. Evrişimsel Sinir Ağları sınıflandırma problemini çözmek için yapay sinir ağları kullanır, ancak verileri belirlemek ve bu verilerin özelliklerini çıkarabilmek için diğer katmanları kullanır. Bunun sonucunda da işlem yükü ve süresi artar. (Krizhevsky ve ark., 2012) tarafından yapılan çalışmada 1,2 miyon yüksek çözünürlüklü görüntü 1000 farklı sınıfa ayrılıp geniş bir derin sinir ağı eğitilmiştir. Test verileriyle önceden yapılan çalışmalara oranla daha başarılı olan %37,5 ve %17,0 oranları elde edilmiştir. Sonuç olarak ortaya atılan derin evrişimsel sinir ağı tamamen denetimli öğrenme kullanarak zor bir veri setinin eğitiminde başarı göstermiştir.

(Girshick ve ark., 2014) tarafından yapılan çalışma ile R-CNN modeli ortaya atılmıştır. Pascal VOC dataseti ile yapılan eğitimde önceki çalışmaya oranla Mean Average Precision (mAP) değeri %30 daha fazla artırılmış ve %53,3 oranında Mean Average Precision (mAP) yüzdesi elde edilmiştir.

(Girshick, 2015) tarafından yapılan çalışmada yeni bir model olarak Fast R-CNN fikrini öne sürdü. Fast R-CNN, R-CNN'den 9 kat daha hızlı eğitim, 213 kat daha hızlı test sonuçları ve daha yüksek mAP değeri elde etti.

(Ren ve ark., 2016) tarafından yapılan çalışmada Faster R-CNN modeli tanıtıldı. Önerilen metot ile nesne tanıma konusunda derin öğrenme için resim çerçevesini tanımada gerçek zamana yakın sonuçlar elde edilmiştir. Bu çalışma sonuçları Faster R-CNN modelini gerçek zamanlı uygulamalarda bir adım öne çıkarmaktadır.

Bu kaynak araştırması sonucunda, tez kapsamında gerçek zamanlı trafik işaretleri tanınacak olmasından dolayı hızlı ve doğru sonuçlar almak amacıyla Faster R-CNN modeli tercih edilmiştir. Faster R-CNN ağı ile eğitilen veriler başarılı bir şekilde tanınmış ve gerçek zamanlı olarak hızlı bir şekilde sisteme entegre edilmiştir.

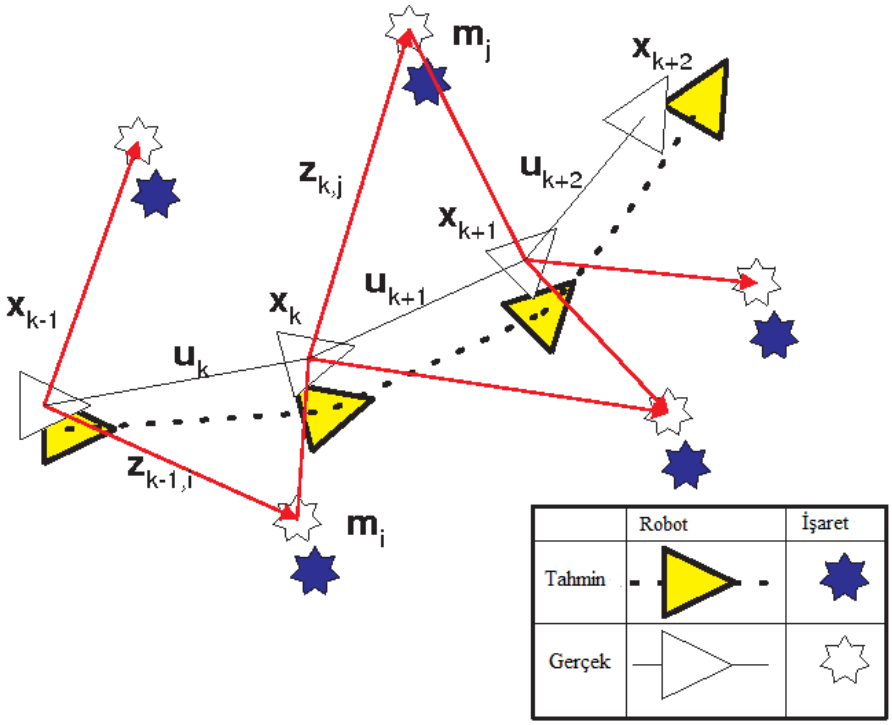
3. MATERYAL VE YÖNTEM

Tez kapsamında Bölüm 3 içerisinde kullanılan materyal ve yöntemler detaylandırılmıştır. Bölüm 3.1'de Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM) konusu tanımlanmış ve yöntemleri hakkında bilgi verilmiştir. Bölüm 3.2'de Bayes Tahmin Filtreleri ve tez kapsamında kullanılan Monte Carlo Lokalizasyonu (MCL) yöntemi hakkında bilgiler verilmiştir. Bölüm 3.3'te Yapay Sinir Ağları konusunda genel bilgiler verilip nesne tanıma için kullanılan Derin Öğrenme ağı ve modeli detaylandırılmıştır. Bölüm 3.4'te önerilen matematiksel yöntem ve formülasyonları gösterilmiştir. Bölüm 3.5 de kalkış noktasından hedef noktasına otonom harekette yol planlaması yapılmasına değinilmiştir. Son olarak Bölüm 3.6'da deneysel çalışmaların yapıldığı Robot İşletim Sistemi (ROS) hakkında detaylı bilgiler verilmiştir.

3.1. Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM)

Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM) problemi robotun bilinmeyen bir ortamın haritasını oluşturması ve aynı zamanda bu ortamda kendi konumunu belirlemesidir (Thrun ve Leonard, 2007). Burada iki ana konu vardır. Birincisi ortamın haritası oluşturulacak, ikincisi bu harita içerisinde robotun konumu belirlenecektir. Ayrıca bu konu, robotun kısmen otonom bir hareket sağlaması için de önemlidir. Haritaya ve çevreye bağlı olarak robot bir konumdan başka bir konuma otonom olarak hareket ettirilebilir. Bir yol planlaması algoritması sayesinde robotun başlangıç konumundan hedeflenen bir noktaya otonom olarak gönderilmesi konusunda SLAM algoritmalarının önemini göstermektedir.

Harita oluşturma ve konum belirleme konuları literatürde ayrı ayrı ele alınmıştır. Özellikle robotun anlık konumunun belirlenmesinde Bayes tabanlı tahmin filtreleri yoğunlukla kullanılmıştır. Bunun yanı sıra haritalama konularında ortamda bulunan özelliklerin belirlenmesi ve eşleştirilmesi, robotun ilerlediği yol üzerinde doluluk boşluk ızgarası oluşturması gibi konular ele alınmıştır. Haritalama ve konumlandırma konularının beraber ele alındığı çalışmalar da bulunmaktadır. Şekil 3.1'de SLAM probleminde bir konum tahmin şekli gösterilmiştir (Durrant-Whyte ve Bailey, 2006).



Şekil 3.1 Robotun harita işaretçileri gösterimi (Durrant-Whyte ve Bailey, 2006)

Haritalandırma ve konumlandırma kavramlarının beraber ele alınmasıyla SLAM problemi oluşmuştur ve çeşitli algoritmalar yardımıyla bu problem çözülmeye çalışılmıştır. SLAM her iki konuyu da içerisinde barındırdığından dolayı zor ve zahmetli bir problemdir. SLAM probleminin temelinde sensör verileri ile alınan çevresel değişkenler girdi; harita ve konum bilgisi ise çıktı olarak alınır (Khairuddin ve ark., 2015). Olasılıksal Bayes bakış açısıyla SLAM probleminin 2 ana formu vardır. Birincisi; çevrimiçi SLAM olarak bilinir. Haritayla birlikte anlık konum üzerinde posterior (öncül) tahmini yapılır.

Denklem (3.1)'de x_t; robotun t anındaki konumu, m harita, $z_{1:t}$ ölçümler ve $u_{1:t}$ kontrol matrisini verir. Bu problemin çevrimiçi SLAM olarak adlandırılmasının nedeni sadece t anındaki tahminleri içermesidir. İkinci SLAM problemi de tam SLAM olarak adlandırılır. Burada ise sadece geçerli x_t yerine $x_{1:t}$ gibi bir posterior (öncül) hesaplanır.

$$p(x_t, m \mid z_{1:t}, u_{1:t}) \tag{3.1}$$

Çevrimiçi SLAM problemi, tüm SLAM problemleri ile geçmiş pozların bütünleştirilmesinin sonucudur. Genel olarak SLAM probleminin tahmin formülasyonu Denklem (3.2)'deki gibidir.

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \tag{3.2}$$

Denklem (3.3)' de x robotun konumu, m haritadaki işaretlerin konumu ve $Z_{0:t}$, $U_{0:t}$ sırasıyla t anındaki gözlemler ve kontrollerdir. Gözlemler eğer robotun konumu ve haritadaki işaretler biliniyorsa robot konumundan bağımsız olarak düşünülebilir (Kuleli, 2009).

$$p(x_t, m \mid Z_{0:t}, U_{0:t}, x_0) \tag{3.3}$$

Robotun bahsedilen hareket ve gözlem modelleri denklem (3.4) ve denklem (3.5)' deki gibi belirlenir. Burada robotun peş peşe vereceği tahmin ve düzeltme formülasyonu görülmektedir.

$$p(z_t \mid x_t, m) \tag{3.4}$$

$$p(x_t \mid x_{t-1}, u_t) \tag{3.5}$$

Denklem (3.6)'da zaman güncellemesi ve denklem (3.7)'de ölçüm güncelleme formüller verilmiştir. Yukarıdaki denklemlerden yola çıkılarak, haritalama probleminde koşullu olasılık yoğunluk fonksiyonu $p(m \mid x_{0:t}\, z_{0:t}, u_{0:t})$ kullanılarak bulunur. t anında robotun ilk konum bilgisinin alınmasıyla bütün konumları tahmin edilir. Lokalizasyon probleminde ise $p(x_t \mid z_{0:t}, u_{0:t}, m)$ fonksiyonu kullanılarak hesaplama yapılır.

$$p(x_t, m \mid Z_{0:t-1}, U_{0:t}, x_0) = \int p(x_t \mid x_t,\ x_{t-1}, u_t)\, p(x_{t-1}, m \mid Z_{0:t-1}, U_{0:t-1}, x_0) dX_{t-1} \tag{3.6}$$

$$p(x_t, m \mid Z_{0:t}, x_0) = \frac{P(z_t \mid x_t, m) P(x_t, m \mid Z_{0:t-1}, U_{0:t}, X_0)}{p(z_t \mid Z_{0:t},\ U_{0:t})} \tag{3.7}$$

SLAM probleminin çözümünde çeşitli tahmin filtreleri kullanılmaktadır. Kalman Filtresi, Genişletilmiş Kalman filtresi ve Parçacık Filtresi yaygın olarak kullanılan yöntemlerdir. Kalman Filtresi (KF); uzay, robotik ve tıp alanlarında uzun yıllardır tercih edilen bir tahmin yöntemidir. Gauss dağılımlarına bağlı, doğrusal (lineer) sistemlerde

oldukça başarılı tahmin sonuçları verir. Ancak gerçek zamanlı çalışan robotik sistemlerde lineer olmayan ortam koşullarından dolayı hatalı tahminlerde bulunurlar. KF' nin lokalizasyon konusunda bu dezavantajı Genişletilmiş Kalman Filtresi (GKF) ile çözülmüştür. GKF' de lineer olmayan sistem öncelikle doğrusallaştırılır ve sonra Kalman uygulanır. Bu doğrusallaştırma Gaussian denklemlerinin kullanılmasıyla mümkün olur. Jacobien Matrisleri kullanılarak GKF, SLAM de tahmin ve güncelleme denklemi elde edilir. Doğrusallaştırma, Kalman'ın uygulanabilirliği açısından fayda sağlarken, dört konuda da problem meydana getirir. (Durrant-Whyte ve Bailey, 2006). Bu dört problem;

- Yakınsama
- Hesaplama güçlüğü
- Veri tamamlayıcılığı
- Doğrusallığın olmamasıdır.

Doğrusallaştırma çabası ve Gauss denklemlerinin hesap karmaşıklığı süre açısından dezavantajlıdır.

Parçacık Filtreleri Gauss dağılımına bağlı olmadan çalışırlar. Lineer olmayan sistemlerde gürültüyü ifade edebildiklerinden dolayı robotik sistemlerde, konumlandırma konularında tercih edilmektedir. FastSLAM algoritmaları Rao-Blackweillezed parçacık filtre tabanlı algoritmaları robotun lokalizasyonunu tahmin etmede başarılı sonuçlar vermiştir. FastSLAM algoritmasının (Montemerlo ve ark., 2002) tarafından ortaya atılmasından sonra olasılıksal SLAM'in tasarımında temel bir kavramsal kayma meydana geldi. GKF ile Gauss varsayımlarına dayanan çözümler düşük performanslıydı. Ancak doğrusal olmayan Gauss dağılımlarının parçacık filtreleri ile doğrusallaştırmadan çözümleri önemli bir hesaplama karmaşıklığını ortadan kaldırmıştır (Montemerlo ve ark., 2002). Rao-Blackwellized temelli parçacık filtreleri uygulanarak örnek alanı azaltılmış ve işlem hızı artırılıp hesap yükü azaltılmıştır. Bu tez çalışmasında da Rao-Blackwellisation parçacık filtresi tabanlı Gmapping algoritması kullanılmış ve bu algoritma belirtilen nedenlerden dolayı tercih edilmiştir.

Sensörler yardımıyla robot, ortamdan veri toplayarak bu verileri anlamlandırır ve kullanır. Lazer mesafe ölçüm sensörleri, Sonar sensörler, RGB kameralar veya GPS (Küresel Konumlama Sistemi) gibi sensörler ile çevre etkileşimi sağlarlar. Konumlandırma konusunda dış ortamlarda GPS sensörleri yüksek doğrulukla sonuçlar verebilmektedir. Ancak kapalı ortamlarda GPS verileri alınamaz ve robot konumunu belirleyemez. Bu gibi durumlarda ortaya çıkan problemleri çözebilmek için çeşitli

yöntemler kullanılmaktadır. Odometri sensörü yardımıyla robotun teker çevre bilgisi elde edilip, çevrede ne kadar ilerlediği bilinebilmektedir. Kör sayım (Dead-Reckoning) ile robot farkındalığı olmadan ortamda ilerler ve teker çevresiyle ilerleme ve açı değer verisi elde eder. Daha sonra ortamda otonom hareket esnasında bu bilgiyi kullanır ve konumunu tahmin eder. Ancak "kör" bir sayım olduğundan dolayı hataya oldukça açıktır. Örneğin robotun bir engele takılması sonucunda tekerin boşa dönmesi veya kaygan-yumuşak bir zeminde tekerin boşa dönmesi gibi durumlarda robot; haritada sabit bir konumda kalırken tahminde ilerlediği varsayılır. Bu nedenle robotun konumu yanlış tahmin edilir. Hata kümülatif olarak artar ve devam eden ilerleme sonunda oldukça büyük bir hata ile karşılaşılır.

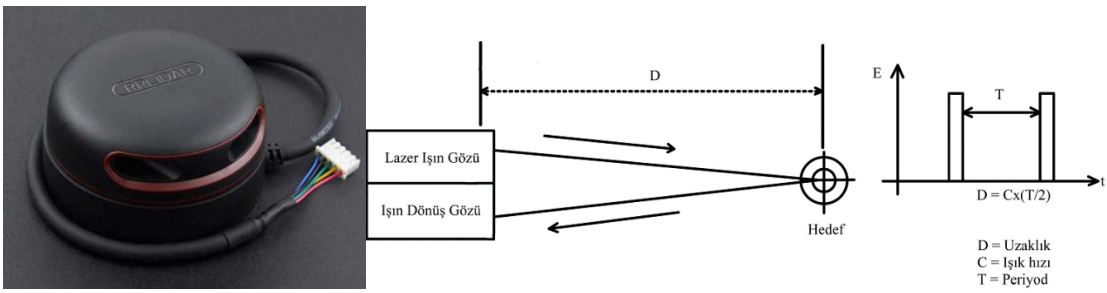
Bu tez kapsamında; kapalı ortamlarda odometri verisinin potansiyel hatasına karşı farklı bir yöntem üzerinde çalışılmıştır. Önerilen yöntemin kullanılmasıyla beraber robot, teker verisini kaybettiğinde farklı bir konum tespiti yaparken; gerçek konumu trafik işaretçileriyle belirlenmiştir. Bu sayede robot bilinen başlangıç noktasından hedef noktasına giderken tahmin ettiği konum bilgisi ve önerilen yöntemle tahmin edilen konum bilgileri karşılaştırılmıştır.

3.1.1. SLAM Yöntemleri

Günümüzde SLAM için klasik dönem ve modern dönem kavramları bulunmaktadır. Farklı sensörlerin ve yöntemlerin kullanımı bu ayrımı oluşturmaktadır. Klasik dönemde daha çok mesafe sensörleri kullanılarak SLAM uygulamaları yapılmıştır. LIDAR, Sonar, Radar, IR sensörler bunlara örnek verilebilir. En yaygın sensör çeşidi olarak LIDAR tercih edilmektedir. Ancak LIDAR sensörlerin yüksek maliyetleri ve diğer sensörlerin de yetersizliği farklı bir çözüm yolu ortaya çıkarmıştır.

Modern dönem SLAM yöntemleri bu dezavantajlar sonucunda ortaya çıkmıştır. Tek kamera ile görsel haritalandırma çalışmaları yapılmış ve hem sensör fazlalığı ortadan kaldırılmaya çalışılmış; hem de maliyetler aşağı çekilmiştir. Görsel SLAM konusunda hala çalışmalar yapılmaktadır ve henüz çözülemeyen bazı problemler mevcuttur. Örneğin; iki kare görüntü arası özellikler çıkarılırken eşleştirmenin yapılamaması veya karanlıkta kameranın çalışmaması ve SLAM yapılamaması gibi dezavantajları mevcuttur.

Bu tez çalışmasında LIDAR sensör kullanılarak SLAM çalışması yapılmıştır ve bu konuda detaylı bilgi verilmiştir. LIDAR lazer ışınının uçuş süresini hesaplayan bir mesafe ölçüm sensörüdür.



(a) (b)

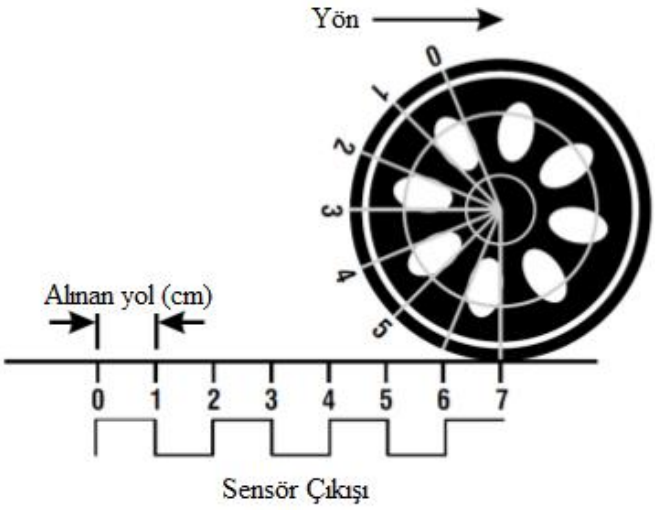
Şekil 3.2 LIDAR (a) ve çalışma mekanizması (b)

Şekil 3.2'de gösterildiği gibi LIDAR iki adet gözden oluşmaktadır. Bu gözlerden biri lazer ışınını gönderir. Bir engele çarpıp dönerek yansıyan ışın diğer gözden LIDAR'a tekrar girer. Lazer ışınının çıkış, engele çarpış ve engelden dönüş süresi hesaplanır. Işık hızı da bilindiğinden dolayı engelin LIDAR'a olan uzaklığı denklem 3.8 ile hesaplanır. D engele olan mesafe, C ışık hızı ve T lazer ışınının uçuş süresidir.

$$D = \frac{C.T}{2} \tag{3.8}$$

LIDAR, 360 derece dönerek tek lazer ışını gibi binlerce ışınların oluşturduğu bir ışın demeti gönderir. Bu ışın demeti topluluğuna nokta bulutu denir. Dolayısı ile LIDAR, bir nokta bulutu yayarak etrafındaki bütün nesnelerin uzaklıklarını belirler. Bundan dolayı SLAM uygulamalarında sıklıkla kullanılırlar ve yüksek doğrulukla haritalandırma sonuçları verirler.

Kullanılan bir diğer sensör ise teker sayacı enkoderlardır. Odometri verisini elde etmede kullanılırlar. Robotun ne kadar ilerlediğini sayarak ortamda bir kör sayım gerçekleştirirler. Şekil 3.3'te verilen örnekte görüldüğü gibi teker dönüşüyle beraber ne kadar ilerleme kaydedildiği bilgisi elde edilir. SLAM problemi için birkaç algoritma ve paket mevcuttur. Bu algoritmalar içerisinde en sık kullanılanı Gmapping ve HectorSLAM yöntemleridir.



Şekil 3.3 Odometri sensörü ile teker ilerlemesi (Spielmann ve ark., 2013)

3.1.2. Gmapping Yöntemi

Bu yaklaşım, her partikülün ortamın ayrı bir haritasını taşıyan bir parçacık filtresi kullanmaktadır (Grisetti ve ark., 2005). Rao-Blackwellized parçacık filtresi temelli bir SLAM algoritmasıdır. Mesafe sensörü ve odometri sensörü kullanılarak yapılabilmektedir. Robot ortamda gezerken işaretçiler ile örneklemeler yaparak ortamın haritasını oluştururken, odometri sensörü ile de ortamda ne kadar ilerlediği bilgisini elde eder. Her bir parçacık; robotun bir önceki pozisyon örneği ve haritada bilinen önceki pozisyon örneği toplamına eşittir. SLAM için Rao-Blackwellized parçacık filtresinin ana fikri; gözlemleri $z_{1:t}$ ve robotun odometri ölçümleri $u_{0:t}$ göz önüne alındığında, robotun potansiyel $x_{1:t}$ yörüngeleri hakkında bir $p(x_{1:t} \mid z_{1:t}, u_{0:t})$ tahmini yapmak ve öncül harita ve yörüngeler üzerinden bir öncül değeri Denklem (3.9) ile hesaplamaktır (Grisetti ve ark., 2005).

$$p(x_{1:t}, m \mid z_{1:t}, u_{0:t}) = p(m \mid x_{1:t}, z_{1:t})\, p(x_{1:t} \mid z_{1:t}, u_{0:t}) \tag{3.9}$$

Gmapping olasılıksal dağılım yöntemlerini kullanarak robotun son gözlem bölgesine göre bir yayılım örneği oluştururlar. Böylece belirsiz durumlar giderilir ve doğru bir harita oluşturulur. Bu yöntemde kullanılan Rao-Blackwellized ile her güncellemede yeniden örnekleme yapılır. Böylece robot yörüngesi ve haritayı temsil eden örnekler seti güncellenir. Sırasıyla Örnekleme, önem ağırlığına göre atanma, sonuçlara göre yeniden örnekleme ve her bir parçacık gözlemine karşılık gelen harita tahmini yapılır. Rao-Blackwellized parçacık filtresi ile odometri verisine ihtiyaç duyulmaktadır.

Bu yöntem ile daha doğru haritalama yapıldığı görülmüştür. Çünkü düşük ağırlıklı parçacıklar kaybolmuş ve yerine yüksek tahminli parçacıklar güncellenerek tahmin doğruluğu artırmıştır.

Gmapping yöntemi parçacıkların yeniden örneklenme işlemi için seçici bir uyarmalı teknik sunar. Yörünge ve haritaların iyi parçacıklarının yeniden örnekleme adımları için dikkatli bir şekilde kullanılması gerekir. (Doucet ve ark., 2001) tarafından önerilen yaklaşım ile yeniden örnekleme basamağı Gmapping algoritmasında kullanılmıştır. Bu yaklaşımla parçacıklar arasında önemli ağırlık değerine sahip olan parçacıklar dikkate alınmıştır. Denklem (3.10)'da Ne_{ff}, parçacığın öncül yörüngede ne kadar iyi bir ağırlığa sahip olduğunu belirtir. $\tilde{\omega}^{(i)}$, i parçacığın normalize edilmiş ağırlığıdır. Ne_{ff}'in parçacık sayısının, N/2 sayısının yarısının altına düştüğü her durumda bir yeniden örnekleme adımı gerçekleştirilir. Böylece parçacık tükenme riski azalırken daha doğru bir harita meydana gelir.

$$Ne_{ff} = \frac{1}{\sum_{i=1}^{N} \left(\tilde{\omega}^{(i)}\right)^2} \tag{3.10}$$

Bu tez kapsamında yapılan birinci araştırmada Gmapping algoritması kullanılarak SLAM uygulaması yapılmıştır.

3.1.3. HectorSLAM Yöntemi

SLAM uygulamalarında kullanılan en yaygın ikinci yöntem HectorSLAM algoritmasıdır. (Kohlbrecher ve ark., 2011) tarafından önerilen bu yöntem açık kaynaklı olup iki boyutlu bir SLAM yöntemidir. HectorSLAM yöntemi çevrenin ızgara haritasını çıkarmak için bir lazer tarama kullanmayı gerektirir. Harita SLAM yöntemlerinin birçoğuna nazaran kilometre sayacı (odometri) bilgisi gerektirmez. Bu nedenle robotun pozu sadece tarama eşleştirme yöntemlerine dayanarak tahmin edilir. Modern LIDAR sensörlerin hız ve doğruluğunun artmasından dolayı bu yöntemle, hızlı ve doğru tahminler elde edilmektedir (Kamarudin ve ark., 2014).

HectorSLAM'de kullanılan tarama eşleştirme algoritması Gauss-Newton yaklaşımına dayanmaktadır. Algoritma, $\xi = \left(P_x, P_y, \psi\right)^T$ dönüşümünü bularak, lazer taramasının uç noktalarının oluşturulmuş harita ile optimum hizalanmasını bulur.

Denklem (3.11)'deki $M(s_i(\xi))$ fonksiyonu tarama bitiş noktalarının global haritadaki koordinatları olan $(s_i(\xi))$ noktasını döndürür. Işınların bitiş noktalarının hizalanması ilkelerine dayanan bu yöntemle lazer ışınları eşleştirmesi baz alınmıştır.

$$\xi^* = argmin \ \sum_{i=1}^{n} [1 - M(s_i(\xi))]^2 \tag{3.11}$$

HectorSLAM' de teker odometri verisine ihtiyaç yoktur ve sadece tarama düzleminin eşiğindeki bitiş noktaları eşleştirilir. Dolayısı ile saydam nesnelerden çıkan ışınlar haritaya dahil edilir. Örneğin cam saydam olduğundan dolayı lazer ışınları nüfuz eder ve yansıtılmaz. HectorSLAM yönteminde ışınların eşleştirilmesi ilkesinden dolayı camlı bölgeler de haritaya dahil edilebilir. Bu nedenle bir dezavantaja sahiptirler. Ancak sadece LIDAR verilerinin alınmasına bağlı olarak hızlı ve doğru sonuçlar vermektedir. Özellikle İnsansız Hava Araçları (İHA) ve el ile haritalandırma konularında kullanılmaktadırlar.

Bu tez kapsamında yapılan ikinci çalışmada ortam LIDAR sensör kullanılarak HectorSLAM algoritmasıyla haritalandırılmıştır. Haritalandırılan iç mekânda ölçeklendirme yapılarak mekânın büyüklüğü ve planı çizilmiştir. Kullanılan bu yöntem ile süre ve iş gücü konusunda avantaj sağlanmıştır.

3.2. Bayes Filtreleri ve Monte Carlo Lokalizasyon Tekniği

SLAM çalışmalarında ve robotik sistemlerde Bayes tabanlı tahmin filtreleri kullanılmaktadır. En yaygın olarak kullanılanlar Kalman Filtresi, Genişletilmiş Kalman Filtresi ve Parçacık Filtreleridir. Kalman Filtresi tabanlı yapılan ilk SLAM çalışmalarında kısmi gözlemlenebilirlik etkileri araştırılmıştır (Andrade-Cetto ve Sanfeliu, 2004). Genişletilmiş Kalman Filtresi kullanılarak yapılan bir çalışmada ise odometri verisinin kümülatif poz hatası azaltılmış ve daha stabil bir sonuç ortaya koyulmuştur (Weingarten ve Siegwart, 2005). Monte Carlo tekniği kullanılarak parçacık filtresi tabanlı bir SLAM uygulamasında robot ve engel konumlarını tahmin edebilmek için çoklu genel parçacık filtreleri kullanılmıştır (Montemerlo ve Thrun, 2003).

Bayes Teoremi istatistiksel bir yaklaşımdır. İnançları hesaplamak için kullanılan en yaygın algoritma Bayes Filtre algoritmasıdır. Robotik sistemlerde de meydana gelen olasılıksal tahminleri oluşturmada kullanılır. Bu algoritma tahmin dağılım *bel* (inanç) değerini kontrol verilerinden elde eder. Bayes Teoreminde yapılan tahmin bir önceki

inanç değerine göre oluşturulur. Yani t anındaki inanç *bel(x_t);* t-1 zamanındaki inançtan *bel(x_{t-1})* hesaplanır. Giriş parametreleri; son alınan kontrol u_t ve son ölçüm z_t ile birlikte t-1 anındaki inanç *bel(x_t)* değeridir. Çıktı ise bir önceki inanç değeri ve parametrelere bağlı olarak t anındaki *bel(x_t)* değeridir. Algoritma 3.1, t-1 anında hesaplanan inançtan yola çıkarak t anındaki inanç *bel(x_t)* değerini hesaplamaya yarar (Thrun ve ark., 2005).

Algoritma 3.1. Bayes Teoremi Algoritması

1: Bayes_Teoremi Algoritması *(bel(x_{t-1}), u_t, z_t):*
2: **for** bütün x_t değerleri
3: $\overline{bel}(x_t) = \int p\ (x_t \mid u_t,\ x_{t-1}\ bel(x_{t-1})dx$
4: $bel(x_t) = \eta\ p\ (z_t \mid x_t\ )\ \overline{bel}(x_t)$
5: **endfor**
6: **return** *bel(x_t)*

Bayes Teoremi bir durum modelinde tahminler kullanarak sonuca ulaşır. Posterior (öncül) değerler ve gözlemler kullanarak bir tahmin sonucu üretir. Bu teorem kullanılarak gürültülü verilerden bir tahmin sonucunda doğru veri elde edilmeye çalışıldığından dolayı Bayes Filtresi olarak da bilinir. Robotun bilmediği bir ortamı haritalayıp verilen görevi bu çevrede gerçekleştirebilmesi için çeşitli tahmin yollarına başvurması gerekir. Konumunu belirleme konusunda Bayes yöntemleri oldukça başarılı sonuçlar vermektedir. Dış ya da iç mekânda hiçbir sensör konum belirleme için mükemmel çalışmaz. Sensör hatalarından ortaya çıkan hatalı sonuçlar Bayes Filtresi ile giderilir (Fox ve ark., 2003).

Denklem (3.12) kullanılarak meydana gelen her durum değişikliği için öncül inanç değerine göre yeni inanç değeri güncellenir.

$$\text{bel}(x_t) = \int p\ (x_t \mid u_t,\ x_{t-1}\ bel(x_{t-1})dx_t \tag{3.12}$$

Güncellenen her adımda odometri verilerine göre x_t tahmin olasılığı hesaplanır. Denklem (3.12)'deki bel(x_t) öncül inanç değerlerine göre sürekli güncellenir. Güncelleme sonucu oluşan inanç; denklem (3.13)'deki gibi olur ve tahmin sonucunda gözlenen ölçüm ile z_t çarpılır.

$$\text{bel}(x_t) = \eta\ p\ (z_t \mid x_t\ )\ \overline{bel}(x_t) \tag{3.13}$$

Denklem (3.14) kullanılarak Denklem (3.15) elde edilir ve burada gösterilen η ifadesi normalizasyon faktörünü ifade eder. Robotik sistemlerde kullanılan Bayes Tahmin algoritmaları bir sonraki bölümde detaylandırılmıştır.

$$p\,(x_t \,/\, z_{1:t}, u_{1:t}) = \frac{p(z_t \,/\, x_t, z_{1:t}, u_{1:t})\, p(x_t \,/\, z_{1:t-1}, u_{1:t})}{p(z_t \,/\, z_{1:t-1}, u_{1:t})} \tag{3.14}$$

$$p(x_t \,/\, z_{1:t}, u_{1:t}) = \eta\; p(z_t \,/\, x_t, z_{1:t}, u_{1:t}) p(x_t \,/\, z_{1:t} - 1\,, u_{1:t}) \tag{3.15}$$

3.2.1. Kalman Filtresi (KF) ve Genişletilmiş Kalman Filtresi (GKF)

Kalman Filtresi, Bayes Filtrelerinin en yaygın olarak kullanılan ve lineer sistem filtrelemesinde çok başarılı sonuçlar veren filtresidir. Kalman Filtresi 1950'lerde ilk olarak Rudolf Kalman tarafından filtreleme ve tahmin tekniği olarak icat edilmiştir. Genellikle sinyal işleme, ses işleme ve görüntü işleme gibi alanlarda tercih edilirken uzay robotik ve savunma sanayi alanlarında da yaygın bir şekilde kullanılmaktadır. Özellikle konumlandırma ve nesne takibi üzerine yapılan çalışmalarda tahmin gücü ve doğruluğu yüksek olduğundan dolayı tercih edilmektedir. Tahmin et ve güncelle işlemlerini yaparak sürekli bir güncel tahmin ortaya koyar.

KF sürekli durumlar için bir inanç hesaplaması yapar. Bu durum ayrık ve hibrit durum uzayları için geçerli değildir. Kalman Filtresi ile durum değerleri Gauss fonksiyonu olarak hesaplanır. Bu nedenle kontrol ve ölçüm değerleri de Gauss gürültüsü eklenmiş parametrelerle gösterilir. Kalman filtresi anlık durumların inanç değerlerini günceller. t zamanında inanç; ortalama değer μ ve koveryans Σ_t değerlerini temsil eder Denklem (3.16).

$$p(x) \;=\; \det(2\pi)^{-\frac{1}{2}} \exp\{\,(x-\mu)^{T-1}(x-\mu)\} \tag{3.16}$$

Bir sonraki durum olasılığı olan $p(x_t \mid x_{t-1}, u_t)$ ihtimali Gauss gürültüsü eklenmiş lineer bir doğrusal denklem olmalıdır. Bu denklem Denklem (3.17) ile gösterilir.

$$x_t \;=\; A_t x_{t-1} \;+\; B_t u_t \;+\; \varepsilon_t \tag{3.17}$$

Burada t anında u_t kontrol vektörü, x_t ve x_{t-1} durum vektörleri, ε_t Gauss gürültüsüdür. A_t ve B_t matrisleri ifade eder. A_t (nxn) boyutunda bir kare matristir. "n"

durum vektörü x_t'nin boyutudur. B_t (nxm) boyutundadır ve "m" kontrol vektörünün boyutudur. Durum ve kontrol vektörü ($x_t \; ve \; u_t$) sırası ile A_t ve B_t matrisleriyle çarpılarak durum geçiş fonksiyonu lineer hale getirilir. ε_t rastgele Gauss vektörüdür. Durum vektörü ile aynı boyutta ve ortalaması sıfırdır. Koveryansı ise R_t olarak ifade edilir.

Denklem (3.18)'de C_t (kxn) boyutunda bir matristir. δ_t ise ölçüm gürültüsüdür. Denklemler ile gösterilen adımlar uygulanarak Kalman Filtresi sonuç tahmini yapılır. Öncül değerler referans alınarak Kalman Filtresi ile bir dizi tahmin üretilir ve bu tahmin sürekli güncellenerek sonuç tahmini oluşturulur.

$$z_t = C_t x_t \; + \; \delta_t \tag{3.18}$$

Kalman Filtresi doğrusal sistemler için önemli bir yöntemdir. Ancak doğrusal olmayan sistemlerde hatalı tahmin sonuçları verirler. Bu yüzden doğrusal olmayan durumların öncelikle doğrusallaştırılıp sonra Kalman Filtresi uygulanması gerekir. Non-Lineer sistemler için Genişletilmiş Kalman Filtresi (GKF) kullanılmaktadır. Elde edilen denklem (3.19, 3.20, 3.21, 3.22 ve 3.23)'de gösterilmiştir (bkz Şekil 3.4).

Önceki Durum Tahmini:

$$x_k = Ax_{k-1} + BU_k \tag{3.19}$$

Hata Koveryans Tahmini:

$$P_k = AP_{k-1}A^T + \; Q \tag{3.20}$$

Kalman Kazancı:

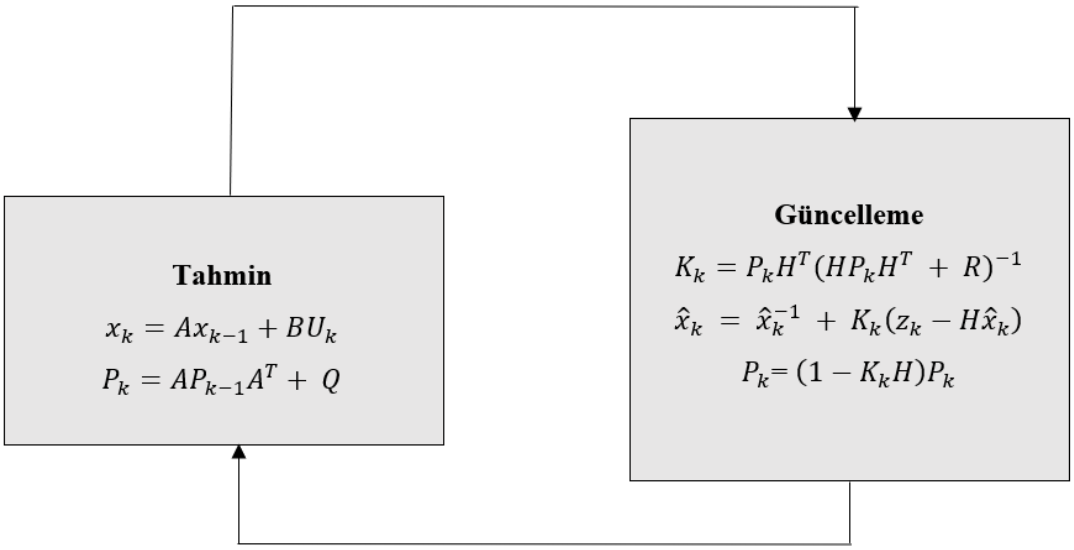
$$K_k = P_k H^T (HP_k H^T \; + \; R)^{-1} \tag{3.21}$$

Tahmin Güncellemesi:

$$\hat{x}_k \; = \; \hat{x}_k^{-1} \; + \; K_k(z_k - H\hat{x}_k) \tag{3.22}$$

Hata Koveryansını Güncelle:

$$P_k = (1 - K_k H)P_k \tag{3.23}$$



Şekil 3.4 Kalman Filtresi tahmin et-güncelle döngüsü.

3.2.2. Parçacık Filtresi (PF)

Kalman Filtresi (KF)'nin doğru tahmin sonuçlarında bulunabilmesi için bütün gürültüler ve sistem durumları doğrusal olmalıdır. KF'nin verimli bir çalışma gösterebilmesi için başlangıç tahmin hataları ve gürültüler Gauss dağılımı olarak ifade edilebilmelidir. Ancak gerçek dünyada Gauss dağılımında olmayan ve doğrusal olmayan birçok durum söz konusudur. Bu problemin çözümünde Genişletilmiş Kalman Filtresi (GKF) tercih edilmiştir. GKF ile f ve h fonksiyonlarının tahminleri sonucunda hesaplanan Jacobien matrisi çözülerek bu problemin üstesinden gelinmiştir (Orderud, 2005). Doğrusal olmayan sistemin, lineerizasyon yöntemleri kullanılarak doğrusal hale getirilip KF uygulanmıştır (Daum, 2005). Ancak bu yöntem birçok matematiksel işlem yükünü getirmiş ve sistemi kısıtlayıcı tahmin durumlarına sevk etmiştir.

Bayes Filtrelerinin parametrik olmayan alternatif bir yöntemi Parçacık Filtresidir (PF). PF, sistemin güncel inancını bir önceki inançtan rastgele bir dizi durum örneğiyle yakınsar. Nesnelerin hedef takibi, konum belirleme ve manevra tahminleri konularında sıklıkla tercih edilirler. PF'de rastgele oluşturulan parçacıklar dağılır. Durum ve ölçüm modellerine göre parçacıklar ağırlık dağılımı alır. Tahminin yüksek olduğu noktalara parçacıklar ağırlanarak toplanır. Bütün parçacıklar ağırlık değerlerini aldıktan sonra yeniden örnekleme işlemi uygulanır. Yeniden örnekleme sırasında ağırlık değerleri düşük olan parçacıklar yok olur. Yok olan parçacıklar yerine yeni parçacıklar örneklenir ve hedefe uzak olan parçacıklar tekrar yok edilir. Parçacıkların örnek sayısı (M), oluşan inancın doğru olabilmesi için bir değere kadar büyük olmalıdır. Fazla miktarda parçacık

tahminin doğruluk oranını artırırken, sistemin yavaşlamasına neden olacaktır. Az sayıda parçacık ise hızlı sonuç verirken, tahmin doğruluk oranını azaltır. t anında Parçacık sayısı denklem (3.24)'de gösterilmiştir (Atalı, 2018).

$$x_t = \{x_t^{[p]} \,/\, 1 \leq i \leq M\} \tag{3.24}$$

Denklem (3.24)'de x_t; t anındaki gerçek durum tahmini, $x_t^{[p]}$ parçacık zaman adımıdır. Parçacık sayısı yeteri büyüklükte verildiğinde sistemin gerçek durumu tahmin etmesi yüksek olasılıklıdır. Başlangıçta bütün parçacıklar eşit ağırlıkla rastgele dağıtılır. Zamanla nesnenin konumuna göre yakın olan parçacıklar daha yüksek ağırlık değeri alır. Düşük ağırlıklı parçacıklar yok olur ve yeniden örnekleme işlemi yapılır.

Algoritma 3.2. Parçacık Filtresi Algoritması

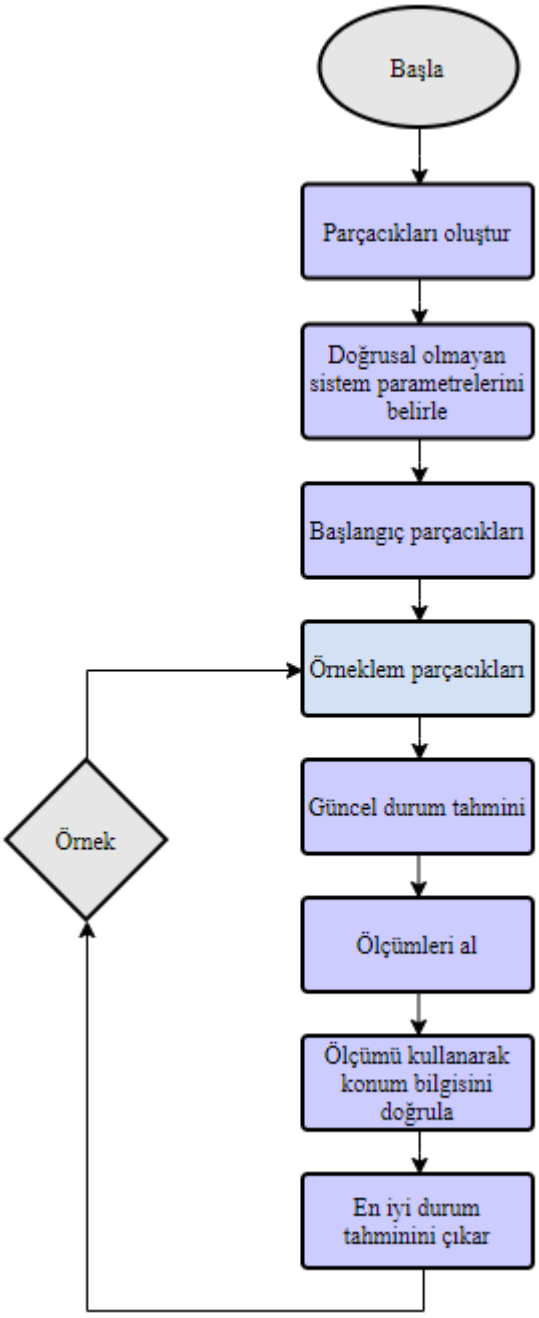
1: **Parcacık_Filtresi Algoritması** (X_{t-1} , u_t , z_t)
2: **for all** p ∈ 1'den M' ye **do**
3: **örnek** $x_t^{[p]}$ ~ p (x_t / u_t , $x_{t-1}^{[p]}$)
4: $w_t^{[p]}$ = p(z_t / $x_t^{[p]}$)
5: **endfor**
6: **for** p = 1'den M' ye **do**
7: p olasılıklarını bul ∝ $w_t^{[p]}$
8: $x_t^{[p]}$ ' yi X_t ' ye ekle.
9: **return** $\boldsymbol{X_t}$

Algoritma 3.2'de Parçacık Filtresinin temel algoritması gösterilmiştir. Belirtilen algoritmada son kontrol değeri u_t , ölçüm ölçüm değeri z_t ve parçacık seti $x_{t-1}^{[p]}$ sistemin girişidir. İlerleyen adımda algoritma öncelikle inanç değerine göre geçici bir parçacık seti oluşturur. 3. Adımda bir sonraki durum dağılımından örnekleme işlemi gerçekleştirilir. 4. Adımda her bir parçacık için $x_t^{[p]}$, önem faktörü olarak adlandırılan $w_t^{[p]}$ hesaplanır. Önem faktörleri z_t ölçümlerini parçacık setine dahil etmek için kullanılırlar. $w_t^{[p]}$ değerini parçacıkların ağırlıkları olarak alırsak, ağırlıklı parçacıklar kümesi Bayes Filtresinin öncül inanç değerini gösterir.

6 ve 9. Adımlar arasında yeniden örnekleme işlemi yapılır. Her bir parçacığın yeniden örneklenmesi önem ağırlığına göre belirlenir. Örnekleme ile parçacıkların ağırlığı değişir ve tahmin güncellenir (Thrun ve ark., 2005). Parçacık Filtresi

örneklemelerinin sonunda en güçlü olan hayatta kalır ilkesinden yola çıkılarak yüksek ağırlığa sahip parçacıklar tahminde gösterilirken, düşük ağırlıklar yok olur ve güncelleme devam eder. Parçacık Filtresi akış diyagramı Şekil 3.5'te verilmiştir.

Robotik sistemler gerçek dünyada doğrusal olmayan parametreler ile çalışırlar. Bundan dolayı PF robotik sistemlerde konum tahmininde tercih edilmektedir. Bu tez kapsamında da robotun konum tahmini parçacık filtresi ile yapılmış ve hatalı olan tahminlerin yeni bir yöntem önerilerek giderilmesi amaçlanmıştır.



Şekil 3.5 Parçacık filtresi akış diyagramı

3.2.3. Monte Carlo Lokalizasyonu (MCL)

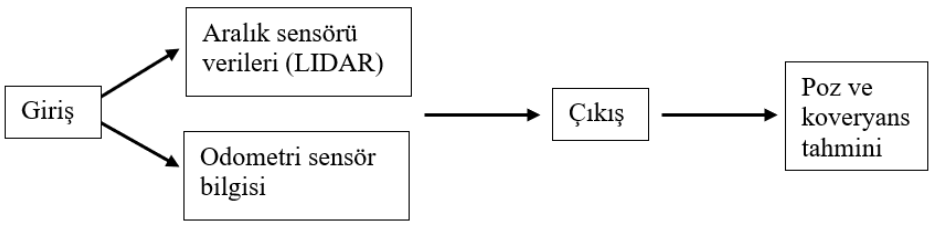
Monte Carlo Localization (MCL) tekniği robotun pozisyonunu ve oryantasyonunu tahmin etmede kullanılan bir yöntemdir. LIDAR ve odometri gibi sensör datalarını kullanarak bilinmeyen bir ortamda lokasyon ve oryantasyon tahmini yapar. Robotu lokalize etmek için; MCL algoritması robot konumunun tahmininde parçacık filtresi kullanır. Parçacıklar her parçanın olası bir robot durumunu temsil ettiği için muhtemel dağılımlarını temsil eder. MCL, LIDAR datalarını ve robot pozunu girdi olarak alır. LIDAR verileri kendi koordinatlarında alınır ve algoritma aracılığıyla dönüştürülür. Girdi pozu odometri sensörü verilerinin zamana entegre edilmesiyle hesaplanır. Robot konumundaki değişiklik güncelleme eşiğinden büyükse parçacıklar güncellenir ve algoritma parçacık filtresinden yeni bir durumu tahmin eder. Parçacıkların dağılımı üç adımda güncellenir ve yeni poz tahmini yapılır. Bu adımlar:

1- Lokalizasyondaki değişime göre parçacıklar yayılır.
2- Aralık sensörü (LIDAR, Sonar, Radar) okumasını alma olasılığına göre parçacıklara ağırlık verilir (Olasılık Ağırlıkları).
3- Parçacıklar dağılımlara göre yeniden örneklenir ve düşük ağırlıklı parçacıklar elemine edilir.

Robot konum tahmini en yüksek ağırlıklı parçacık kümesinin ortalaması ve koveryansıdır. Robotun en iyi konum tahmini için;

- Parçacıkları çoğaltma
- Olasılıkları değerlendirme
- Yeniden örnekleme süreçleri tekrarlanır ve tahmin yapılır.

MCL algoritması, parçacık filtresi kullanır ve konum tahmin edilmesinde kullanılır. Parçacıkların dağılımı robotun konum ihtimallerini gösterir. Her bir parçacık robot konumunun olasılığını temsil eder. En yüksek ağırlıklı parçacık grubu robot pozisyonunu tahmin eder. Algoritmanın girişi LIDAR aralık sensörü ve odometri verilerini temsil eder. Çıkışı ise pozu ve koveryansı tahmin eder (bkz Şekil 3.6).



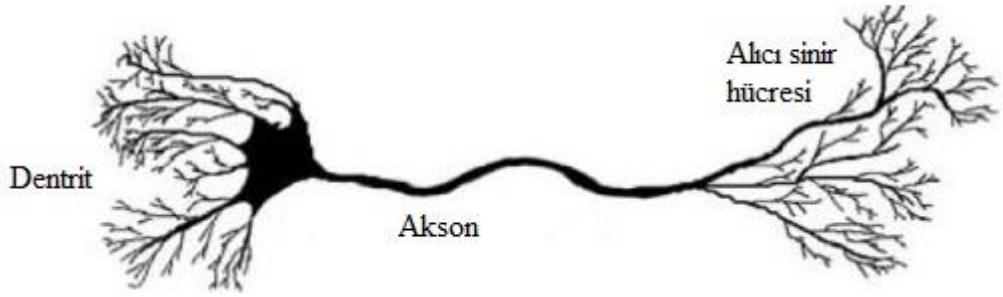
Şekil 3.6 Algoritmanın girdi ve çıktı bilgisi

Parçacık filtresi, tahmini durumun posterior dağılımına yaklaşmak için ayrı parçacıklar kullanan öz yinelemeli ve Bayesian bir durum tahmincisidir. PF durum tahminini yinelemeli olarak yapar ve iki adımı sürekli izler. Bu adımlar tahmin ve düzeltmedir. Tahminde mevcut durumu tahmin etmek için önceki durumu kontrol eder. Düzeltmede ise durumun tahminini düzeltmek için mevcut sensör ölçümlerini kullanır.

Algoritma tahmin edilen durumun önceki dağılımına uyması için parçacıkları periyodik olarak yeniden dağıtır ve yeniden örnekler. Bundan dolayı parçacık filtreleri rastgele doğrusal olmayan sistem modellerine uygulanabilir. MCL algoritmasında, parçacıkların dağılımına göre bir konum tahmini yapılmaktadır. Yüksek miktarda parçacık robotun doğru konum tahmini artırır. Ancak daha az parçacık ile de algoritmanın hızı artar. Bu yüzden parçacık sayısında doğruluk ve hız konuları göz önünde bulundurularak optimum bir seçim yapılması gerekir.

3.3. Yapay Sinir Ağları ve Derin Öğrenme

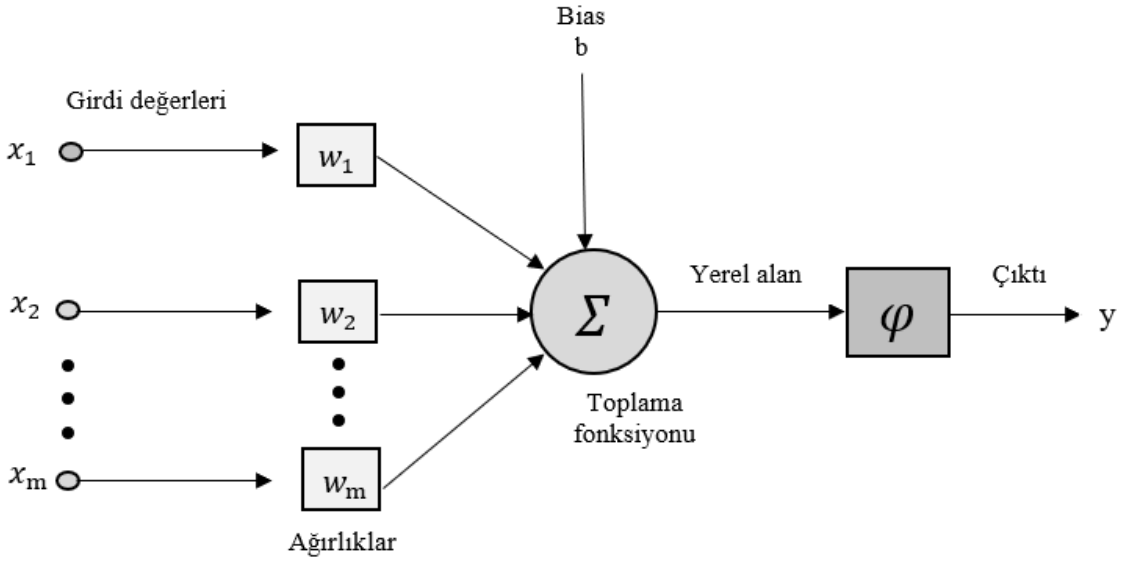
Yapay Sinir Ağları (YSA), insan beyninin yapısal ve işlevsel özelliklerinden ilham alan, çok katmanlı ağ yapıları olan sinir ağları üzerinde çalışan bir dizi algoritma ve modeldir. YSA, insan beyninin öğrenme biçimini taklit eden biyomimetik bir yapı olarak ortaya atılmıştır. Şekil 3.7'de biyolojik bir sinir hücresi gösterilmiştir. Sinir hücreleri akson, dentrit ve alıcı sinir ucu olmak üzere üç kısımdan oluşur. Sinir hücrelerine gelen sinyaller dentrit üzerinden aktarılırken; akson bilgiyi hücrelere dağıtır. Aksonlardan gelen bilgi sinapslardan geçirilerek diğer hücrelerin dentritlerine iletilir ve bu işlem bir sinir ucundan diğer bir sinir ucuna devam ederek ilerler.



Şekil 3.7 Biyolojik sinir ağının yapısı

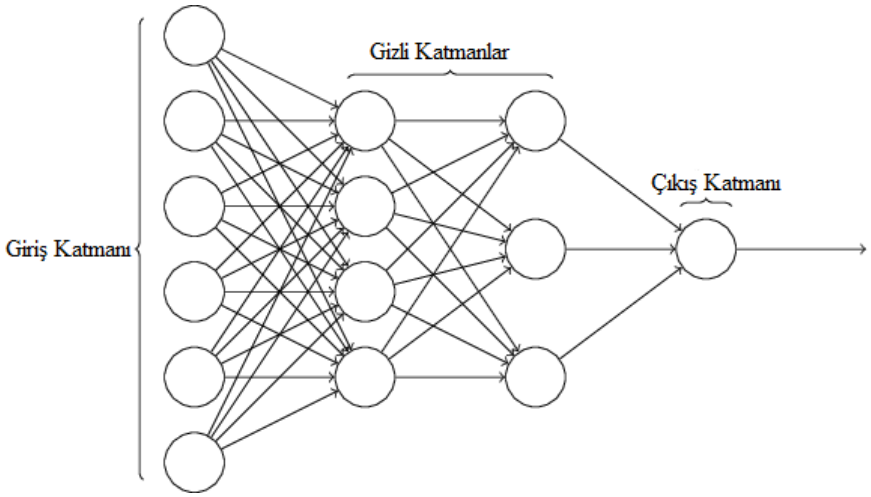
YSA ele alındığında biyolojik sinir hücrelerinde bulunan aksonlar çıkışı, sinapslar ağırlığı ve dentritler aktivasyon fonksiyonlarını temsil ederler. Şekil 3.8'de görüldüğü

üzere bir yapay sinir ağının yapısı verilmiştir. Girdi değerleri, ağırlıklar, aktivasyon fonksiyonu ve çıktı ile sonlanan bir sinir iletimi şekli görülmektedir.



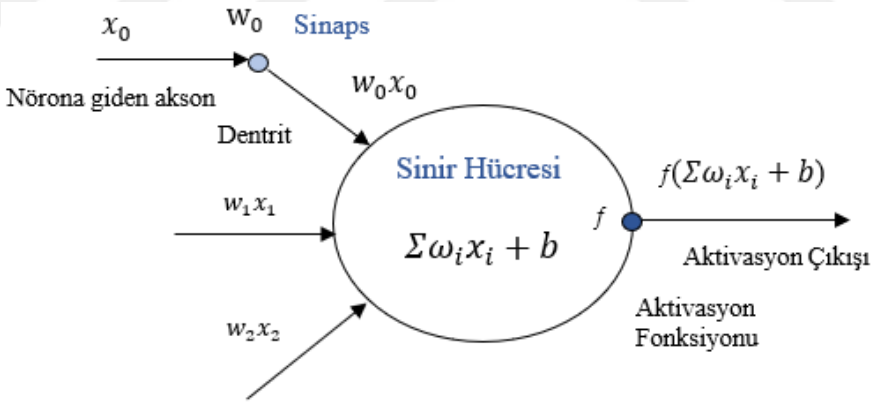
Şekil 3.8 Yapay sinir ağı yapısı

Yapay sinir ağları sinirsel sistemin çalışma mantığından hareketle ele alındığı için tek bir nöronla gerçekleştirilemezler. Beyin bilgileri işlerken anlamlandırma ve sınıflandırma aşamalarını uygular. Bundan dolayı YSA' da benzer şekilde birden çok nöron kullanarak bu işlemi gerçekleştirir. Bu nöronların bir araya gelerek çok katmanlı bir ağ oluşturması gerekir. Çok katmanlı yapay sinir ağlarının genel gösterimi Şekil 3.9'da verilmiştir. Burada ilk başta bilgilerin algılanabileceği giriş katmanı bulunur. Bu giriş katmanında bilgiler algılanarak gizli katmana aktarılır. Gizli katman bloğunda bir veya birden fazla katman bulunur. Gizli katman bloğundaki katman miktarı arttıkça ağ derinleşir. Artan gizli katman sayısıyla beraber öğrenme zorlaşır. Çünkü ağın bilgiyi ezberleme ihtimali artar. Ağın ezberlememesi için data miktarının artırılması yanı sıra aktivasyon fonksiyonları ve parametreler optimize edilmelidir. Giriş ve gizli katmanda işlenen veri çıkış katmanına aktarılarak sınıflandırma gerçekleştirilir. Çıkış katmanındaki nöron sayısı sınıflandırma çeşidine bağlı olarak değişir.



Şekil 3.9 YSA giriş katmanı, gizli katmanlar ve çıkış katmanı yapısı

Aktivasyon fonksiyonları YSA'da önemli bir rol oynamaktadır. Sinir hücrelerinde bilgilerin işlenmesi doğrusal olmayan denklemler ortaya çıkardığından dolayı aktivasyon fonksiyonuna ihtiyaç duyulur. Şekil 3.10'da görüldüğü gibi x girdileri, w ağırlıkları ve *f(x)* ağın çıkışına verilen aktivasyon fonksiyonunu temsil etmektedir. Girdiler aktivasyon fonksiyonuyla işleme sokularak ya çıktıyı ya da başka bir katmanın girdisini oluşturur.



Şekil 3.10 Nöron matematiksel modeli

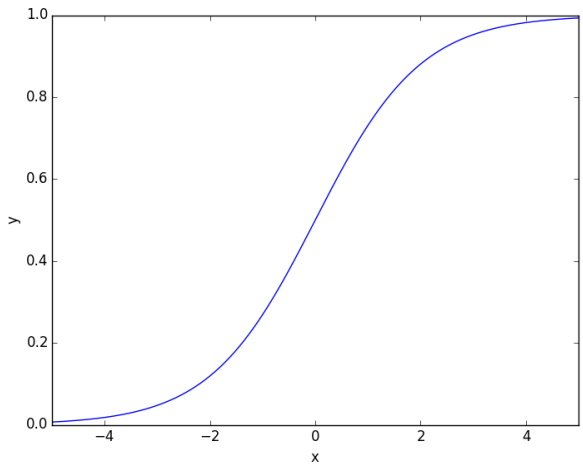
Aktivasyon fonksiyonları kullanılmazsa doğrusal çıkışlar elde edilir. Doğrusal çıkışlar tek dereceli denklemlerdir ve aktivasyon fonksiyonu uygulanmadan elde edilen ağ, sınırlı bir öğrenmeye sahip olur (Lineer Regresyon). Ancak video, resim veya ses gibi gerçek dünya ortamından alınmış doğrusal olmayan verilerin ağ tarafından öğrenilebilmesi için aktivasyon fonksiyonları kullanılır ve doğrusal olmayan çıkışlar da elde edilebilir. Doğrusal olmayan bir çıkış bir dereceden daha büyük dereceye sahip fonksiyonlar anlamına gelir. Ağın bu fonksiyonu hesaplayıp öğrenebilmesi aktivasyon

fonksiyonları sayesinde gerçekleştirilir. Literatürde genellikle kullanılan aktivasyon fonsiyonları; Sigmoid fonksiyonu, tanh fonksiyonu, ReLU (Rectified Linear Unit), doğrusal aktivasyonudur.

Sigmoid Fonksiyonu: Sigmoid fonksiyonu biyolojik nöronların çalışma mantığına yakındır. Giriş olarak aldığı değerleri 0 ve 1 aralığına dağıtarak bir çıktı verir. Matematiksel denklemi 3.25'te verilmiştir.

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{3.25}$$

Sigmoid fonksiyonu ağın eğitiminde yavaş bir aktivasyon fonksiyonudur. Çok katmanlı bir ağda özellikle ilerleyen eğitim aşamasında yavaşlar. Aktivasyon fonksiyonunun eğrisi Şekil 3.11'de görülmektedir.



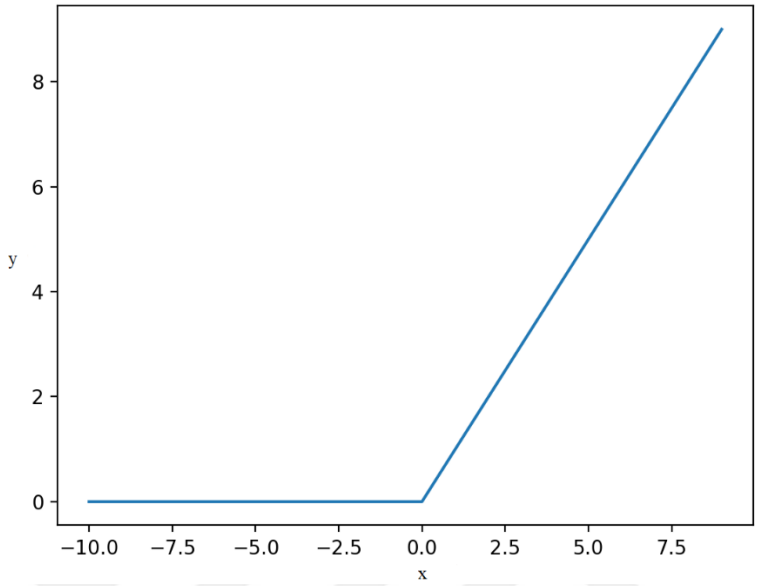
Şekil 3.11 Sigmoid fonksiyonu eğrisi

Tanh Fonksiyonu: Giriş olarak aldığı değerleri -1 ve 1 arasında dağıtır ve bir çıktı verir. Bundan dolayı negatif değerlerin öğrenilmesinde önemli bir rol oynar. Eğrisi Sigmoid fonksiyonuna benzerdir. Farklı olarak -1 ve 1 aralığında değişir.

ReLU Fonksiyonu: Literatürde son yıllarda en yaygın kullanılan aktivasyon fonksiyonudur. Özellikle evrişimsel sinir ağlarında bir ara katman olarak kullanılmaktadır. Ağın doğrusal yapısını bozmak ve ezberleme eğilimini engellemek amacıyla sıfırın altındaki giriş değerlerini sıfır yapar ve sıfırın üstündeki giriş değerleri için doğrusal davranır. Matematiksel eşitliği denklem 3.26'da verilmiştir.

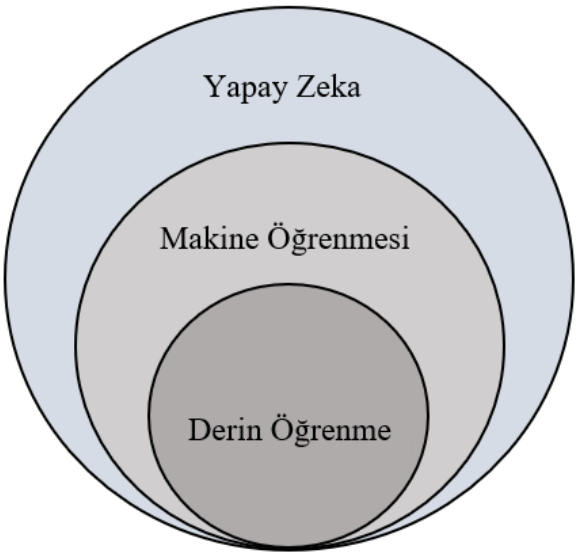
$$f(x) = max(0, x) \tag{3.26}$$

ReLU aktivasyon fonksiyonunun eğrisi Şekil 3.12' de gösterilmiştir.



Şekil 3.12 ReLU fonksiyonu eğrisi

Derin öğrenme bir makine öğrenmesi sınıfıdır (Şeker ve ark., 2017). YSA; makine öğrenmesi ve derin öğrenmeyi kapsayan bir kümedir (bkz Şekil 3.13). YSA'da bulunan gizli katmanlar derin ağlarda oldukça fazladır. Makine öğrenmesinde özellik çıkarma ve bu özelliklerin gizli katmandan çıkışa aktarılması temeldir.

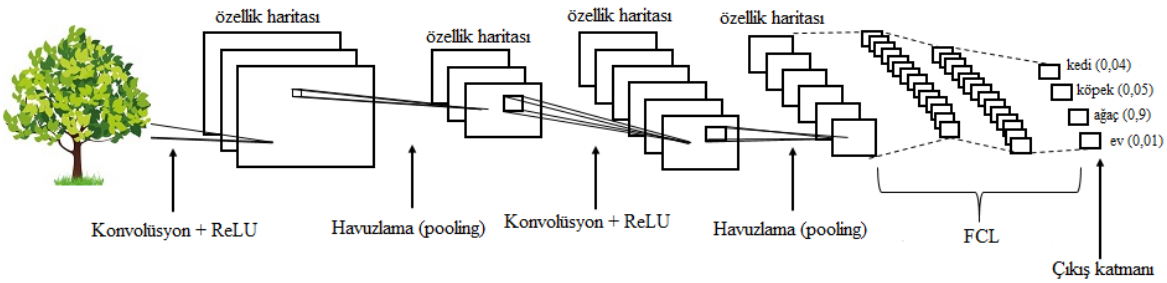


Şekil 3.13 Yapay zekâ, makine öğrenmesi ve derin öğrenme kümesi

Derin öğrenme de özellik çıkarma algoritma tarafından otomatik olarak gerçekleştirilir (LeCun, 2016). Görüntünün ham verisi bir ön işleme tabi tutularak özellik çıkarma işlemi uygulanması gerekir. Manuel olarak yapılan bu işlem hem uğraştırıcı hem de zaman harcayıcıdır. Bunun önüne geçmek amacıyla çok katmanlı yapay sinir ağları kullanılmıştır. Ancak yine de istenilen performans elde edilememiştir. Bu problemler Evrişimsel Sinir Ağlarının (CNN) ortaya çıkmasına neden olmuştur. Evrişimsel Sinir Ağları (CNN) görüntüyü bir girdi olarak alıp, farklı özellikleri birbirinden ayırt edebilen bir derin öğrenme algoritmasıdır. Evrişimsel Sinir Ağları görüntü sınıflandırma, benzerlik kümelerini tanıma ve nesneleri ayırt etmeye yarar. CNN mimarisinde belirli katmanlar bulunur ve bu katmanlar bloklar halinde bir araya gelir. Bloklar bilgi aktarımını sağlar ve amaç eski özellik çıkarma problemlerini ortadan kaldırmaktır. CNN'in en önemli avantajları;

I. Girdinin özellikleri ek bir işleme gerek kalmaksızın öğrenilir. Manuel özellik çıkarmanın zorluğu ortadan kalkar.
II. Filtreler sayesinde ağ derinleştikçe özellik boyutları azalır.
III. Ağın ezberleme eğilimi azalır. Konvolüsyon katmanında özellik matrisi görüntü üzerinde gezdirilerek tüm matrisin işlenmesi gerçeklenir.

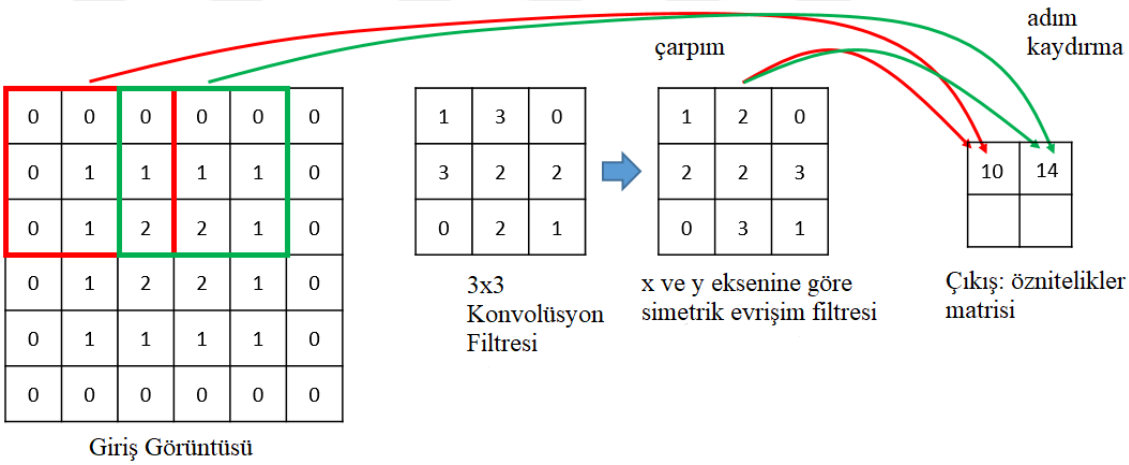
Şekil 3.14'te bir CNN yapısı görülmektedir. CNN mimarisi belirli katmanlardan oluşur. Bu katmanlar Giriş katmanı, çıkış katmanı, konvolüsyon katmanı, ReLU katmanı, softmax katmanı ve FCL (Fully Connected Layer) olarak kısaltılan tam bağlı katmandan oluşur.



Şekil 3.14 Evrişimsel Sinir Ağı (CNN) katmanları

Giriş katmanında veri (görüntü, video vb.) alınır ve konvolüsyon katmanına aktarılır. Konvolüsyon katmanında görüntüye çeşitli filtreler uygulanarak veri boyutlandırılır. Daha sonra diğer katmanlardan geçirilerek çıkış katmanına aktarılır. Bu katmanlar aşağıda detaylandırılmıştır.

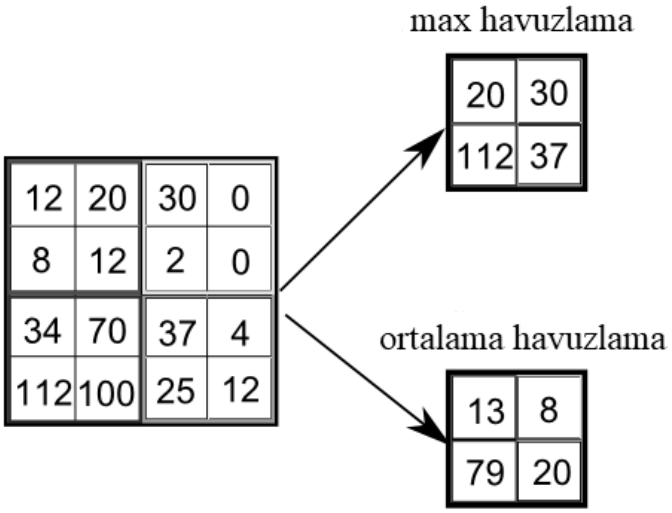
Konvolüsyon Katmanı: Ham görüntü giriş katmanından alındıktan sonra konvolüsyon katmanına alınır. Bu katman dönüşüm katmanı olarak da adlandırılır. Dönüşümden kasıt görüntünün üzerinde bir filtrenin dolaştırılıp yeni bir matris oluşturulmasıdır. Bu bölümde filtre önemli bir rol oynamaktadır. Her filtre görüntüyü dolaşırken özellikler çıkarılır ve çıkış katmanına aktarılır. Her bir adımda filtre katsayıları diğer katmana aktarılır ve ağ üzerinde eğitim yapılırken önemli kısımlar tespit edilmiş olur. Giriş resminin matrisi 6x6 boyutunda olduğu düşünülürse bu görüntü üzerinde 3x3' lük bir konvolüsyon kerneli dolaştırılarak en yüksek değerler alınır ve özellik haritası çıkarılır. Şekil 3.15'te görüldüğü gibi giriş görüntüsü üzerinde evrişim filtresi dolaştırılarak çarpım işlemi gerçekleştirilir ve özellik matrisi belirlenir. Bu aşamada evrişimsel katmanda işlem tamamlanır ve matris diğer bir katmanın girişine verilir.



Şekil 3.15 Konvolüsyon işlemi ve adım kaydırma

Havuzlama (Pooling) Katmanı: Havuzlama katmanında resmin en önemli özellikleri seçilerek matris boyutunun küçültülmesi amaçlanmaktadır. Görüntünün boyutunun azaltılması birçok avantaj sağlamaktadır. Öncelikle eğitilecek parametre sayısının azalmasıyla eğitim süresi kısaltılmaktadır. Ayrıca daha az özellikle beraber daha yüksek doğrulukla bir eğitim süreci geçirilmesi sağlanmaktadır. Havuzlama katmanı sayesinde azalan özellikle beraber ezberlemenin de önüne geçilmektedir. İki tip havuzlama tekniği sıklıkla tercih edilmektedir. Birinci teknik maksimum havuzlama yöntemi, ikinci teknik ise ortalama havuzlama yöntemidir. Bir önceki katmanda olduğu gibi bu tekniklerde de bir kutu gezdirme yöntemi kullanılır ve matrislerin maksimum değerleri ve ortalamaları alınarak çıkış parametresi oluşturulur. Şekil 3.16'da görüldüğü gibi maksimum havuzlamada kernelin boyutu kadar gezdirme gerçekleştirildiğinde maksimum matris

değerleri yazılır. Ortalama havuzlamada ise matrislerin kernel boyutunda ortalamaları alınarak çıkış parametreleri elde edilir.



Şekil 3.16 Maksimum havuzlama ve ortalama havuzlama

ReLU (Düzeltilmiş Doğrusal Birim): ReLU bir aktivasyon fonksiyonudur. Eğitimin ağ üzerinde genellikle ezberleme eğilim vardır. Bu ezberleme eğilimi doğrusal değerlerden kaynaklanmaktadır. Bu problemi ortadan kaldırmak için evrişimsel katmanın çıkışına veya havuzlama katmanının çıkışına ReLU fonksiyonu eklenerek ağın doğrusal yapısı bozulur. Katmanda negatif değerler sıfıra çekilirken pozitif değerler değişmeden katmanın dışına aktarılır ve ReLU kendinden önceki katmanda bulunan doğrusal yapıyı bozar. Dolayısıyla ağın daha hızlı öğrenmesini sağlar.

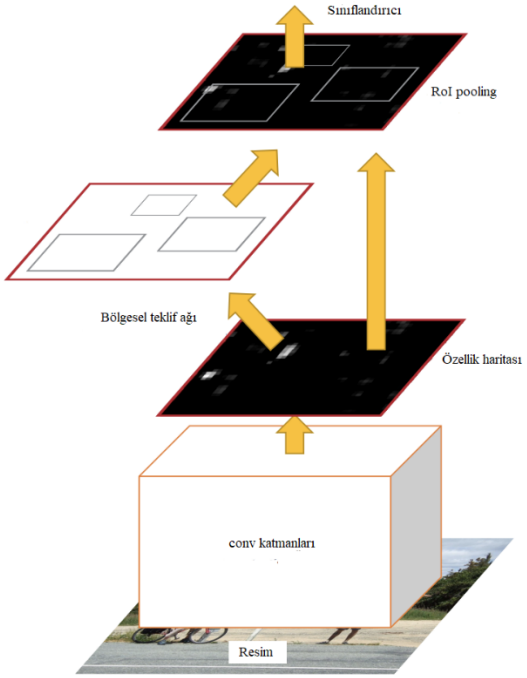
Softmax Katmanı: Softmax katmanında ağın hangi çıkış sınıfına ait olduğunu olasılık dağılımına göre üretir. Örneğin giriş verisine bağlı olarak çıkışta; %70 masa, %20 tahta, %10 sıra gibi bir olasılıksal dağılım sonucu üretir.

Tam Bağlı Katman (Fully Connected Layer, FCL): FCL katmanları CNN mimarisinde özellik vektörünün sınıflandırılmasını yapmaktadır. FCL katmanında önceki katmanlardan gelen matrisler dizilerek bir özellik vektörü oluşturulur ve çıkıştaki sınıf sayısına göre vektörler elde edilir.

Dropout Katmanı: CNN' de ağın eğitimi sırasında ortaya çıkacak ezberlemenin önüne geçilmesi için kullanılan katmandır. Bu katmanın amacı ağdaki bazı düğümleri kaldırarak bazı parametreleri kullanmamaktır. Ağ daha az parametre ile öğrenme işlemini devam ettirdiğinden dolayı ağ parametreleri rastgele değiştirilir ve başarılı bir öğrenme gerçekleştirilir.

Standart bir CNN mimarisinde bulunan bu katmanlar kullanılarak eğitim yapılır. 2000'li yıllara kadar pek popüler olmayan derin öğrenme bu tarihten sonra gelişen işlemci teknolojisiyle beraber büyük bir sıçrama yaşamıştır. Özellikle GPU teknolojisinin gelişimiyle derin öğrenme alnında birçok çalışma peş peşe ortaya çıkmıştır. 2012 yılında gerçekleştirilen ImageNet yarışmasıyla beraber derin öğrenme üzerine yapılan çalışmalar üzerine koyarak devam etmiştir. Bu çalışmalardan öncelikle, R-CNN derin öğrenme modeli ile başarılı eğitim sonuçları alınmış ve çok katmanlı bir ağın eğitim süresi kısaltılmıştır (Girshick ve ark., 2014). Daha sonra Fast R-CNN modeli tanıtılarak, R-CNN'den 9 kat daha hızlı eğitim, 213 kat daha hızlı test sonuçları ve daha yüksek Mean Average Precision (mAP) değeri elde etmiştir (Girshick, 2015). Son olarak, Faster R-CNN ağını tanıtıldı ve PASCAL VOC 2007 dataseti üzerinde güçlü sonuçlar elde edilmiştir. (Ren ve ark., 2016) .

Faster R-CNN: İlk olarak ortaya atılan R-CNN ağı nesne tanıma konusunda oldukça başarılı sonuçlar verse de yavaş çalışmaktadır. Çünkü R-CNN metodunda girişten alınan görüntünün tek tek CNN' den geçirilmesi işlem yükünü artırmaktadır. Bu problemi çözmek amacıyla ortaya atılan Fast R-CNN ağı ise bu bölgeleri tek seferde CNN ağından geçirerek RoIPool olarak adlandırılan bir işlem ortaya koymuştur. Bu da ağı hızlandırmıştır. Ancak yine de gerçek zamanlı bir uygulamada yavaş ve yetersiz kalmıştır. Son olarak Faster R-CNN ağı ortaya konmuş ve bölge öneri ağı yerine nesnenin yerinin tahmini CNN' in Evrişimsel katmanından çıkarılmış olan özellikler kullanılarak yapılmıştır (bkz Şekil 3.17). Faster R-CNN ile nesne tanıma konusunda derin öğrenme ağı başarılı bir sonuç ortaya koymuştur. Özellikle nesne takibi ve robotik nesne tanıma gibi gerçek zamanlı ve anlık çalışma konularında başarılı sonuçlar elde etmiştir. Ancak günümüzde hala eksikleri bulunan bu ağında geliştirilme çalışmaları devam etmektedir.



Şekil 3.17 Faster R-CNN nesne algılama için tek ve birleşik ağı (Ren ve ark., 2016).

Veri Seti: Bu tez çalışmasında Faster R-CNN ağı kullanılarak bir trafik işareti veri seti eğitilmiştir. Veri setinde 6 farklı sınıf oluşturulmuş ve görüntüler ağın giriş katmanına verilmiştir. Bu 6 farklı sınıf; ileri sağ, park, girilmez, yaya, gevşek zemin ve soldan trafik işaretlerinden oluşmaktadır. Şekil 3.18'de oluşturulan veri setinin yol üzerinde bulunan görüntüleri verilmiştir.

Şekil 3.18 Eğitimde kullanılan veri setinin deney ortamından alınan görüntüleri

Eğitimde 1798 adet görüntü veri seti oluşturularak kullanılmıştır. Bu görüntüler gerçek zamanlı olarak robotun sürülmesi esnasında alınmış ve etiketlenerek sınıflandırılmıştır. 1798 görüntünün 360 adeti test ve 1438 adeti eğitim için kullanılmıştır. Eğitim Asus marka bir bilgisayar üzerinde gerçekleştirilmiş ve GPU olarak Nvidia GTX 1050Ti ekran kartı kullanılmıştır. Eğitim sonunda iki levhanın (ileri sağ ve park) robot

gerçek zamanlı olarak ilerlerken tanınması sağlanmıştır. Şekil 3.19'da görüldüğü gibi park levhası %90 (şekil 3.19a), ileri sağ levhası ise %95 (şekil 3.19b) doğrulukla tespit edilmiştir.

(a) (b)

Şekil 3.19 Park (a) ve ileri sağ dönüş (b) levhalarının tanınması

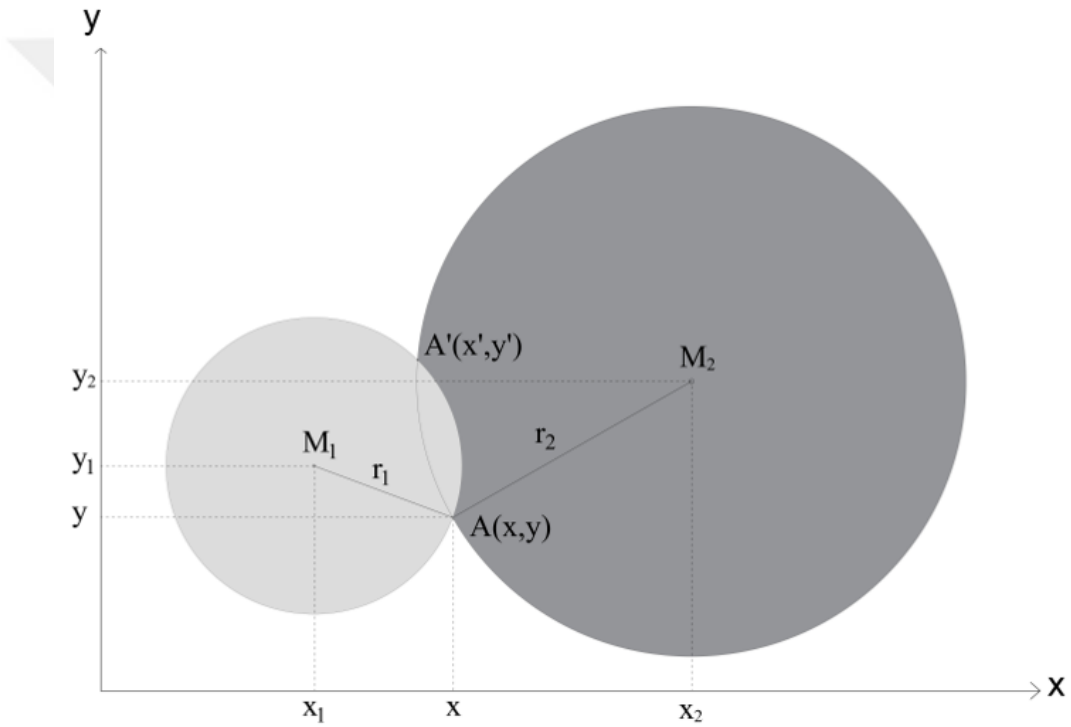
3.4. Nesne Tanıma Tabanlı Konumlandırma için Önerilen Matematiksel Yöntem

Bu tez çalışmasında derin öğrenme tabanlı nesne tanıma yöntemleri kullanılarak kapalı ortamlarda robot konumlandırılması yapılmıştır. İki adet nesnenin konumuna göre robotun konumu tahmin edilmiştir. Tanınan iki nesnenin haritadaki konumları bilinmektedir. Aynı zamanda robotun nesnelere olan uzaklığı da bilinmektedir. Bu nesnelerin koordinatları birer çember merkezi kabul edilir ve robota olan uzaklıkları da bu çemberlerin yarıçapları olarak alınırsa çember kesişim noktası denkleminden robotun konumu belirlenir (bkz Şekil 3.20).

Robot, odometri verisi bozulmadığı takdirde istenilen hedef noktasına yol planlaması yaparak otonom bir şekilde gidebilmektedir. Hedefe varış süresinde robot ortamda kendi konumunu Monte Carlo Localization (MCL) yöntemini kullanarak tahmin eder. Ancak robot kaçırma problemiyle odometri verisinin kümülatif hataları sonucunda robot konum tahmini yanlış yapar. Kilometre sayacı bilgisi bir kör sayım sonucunda oluştuğundan dolayı robot, veri kaybedilince gerçek konumunu bulamaz. Yapılan çalışmayla böyle bir durumda robotun gerçek konumunu tahmin etmesi amaçlanmıştır. Otonom olarak bir noktadan hedef bir noktaya giderken kaybolan lokalizasyon bilgisi iki

nesnenin tanınmasıyla bulunmuştur. Yapılan deneysel çalışmalar Bölüm (4.1)'de detaylı olarak anlatılmıştır.

Şekil 3.20'de önerilen yöntemin koordinat düzlemi üzerindeki gösterimi verilmiştir. İki nesne; global haritadaki koordinat merkezleri M_1 ve M_2 olan iki adet çember olarak kabul edilmiştir. Birinci nesne M_1 merkezlidir ve $(x_1\ ,\ y_1)$ koordinatlarında olduğu görülmektedir. İkinci nesne M_2 merkezlidir ve $(x_2\ ,\ y_2)$ koordinatlarında merkez noktası gösterilmiştir. Robotun iki nesneye olan uzaklıkları da çemberlerin $r_1\ ve\ r_2$ yarıçapları olduğu düşünülmüştür. Robotun bu iki çemberin kesişim noktalarından birinde olduğu kabul edilirse, nesnelere göre robot konumu tespit edilir.



Şekil 3.20 İki nesnenin merkez koordinatları ve kesişim noktaları. Birinci nesne x_1, y_1 , ikinci nesne ise x_2, y_2 koordinatlarındadır. İki çemberin kesişim noktaları A ve A′ robotun bulunabileceği konumlardır.

Birinci çemberin merkezi ve yarıçapı bilinmektedir. Denklem (3.27)'de M_1 merkezli nesnenin çember denklemi verilmiştir. Denklem (3.28)'da ise M_2 merkezli çember denklemi verilmiştir. Merkez noktaları ve yarıçapları bilinen iki çemberin kesişim noktalarının bulunması için denklemler birbirlerinden çıkarılır ve robot konumu olan A(x,y) veya $A'(x', y')$ noktaları tespit edilir (3.29). Bu iki bilinmeyenli denklemden iki farklı kök ortaya çıkar. Bu köklerden biri kesişim noktalarından A noktasını temsil ederken diğeri A' noktasını temsil eder (bkz Şekil 3.20). Denklem (3.40) ve Denklem

(3.41) kullanılarak robotun hangi konumda olduğu belirlenir. r_1 ve r_2 yarıçaplarının birbirlerine göre büyüklüğü karşılaştırılarak robotun hangi konumda nesnelere yaklaştığı tahmin edilir.

İki çemberin merkez noktaları ve yarıçapları biliniyorsa Denklem (3.27) ve Denklem (3.28)'deki eşitlikler çözülür ve x, y noktaları belirlenir. Bu eşitliği çözmek için iki denklem taraf tarafa çıkarılır ve;

$$r_1{}^2 = (x_1 - x)^2 + (y_1 - y)^2 \quad (3.27)$$
$$r_2{}^2 = (x_2 - x)^2 + (y_2 - y)^2 \quad (3.28)$$

Eşitliği elde edilir ve denklem (3.29) ile ifade edilir. Oluşan bilinmeyenlere parametreler atanarak denklemin çözümü kolaylaştırılmıştır.

$$r_1^2 - r_2^2 = x_1^2 - x_2^2 - 2x_1x + 2x_2x + y_1^2 - y_2^2 - 2y_1y + 2y_2y \quad (3.29)$$

Denklem (3.30-3.33)' de kareli ifadelerden oluşan yarıçaplar σ , çember merkez noktaları β , birinci çember noktaları δ , ikinci çember merkez noktaları Υ gibi sabitlere atanarak denklem kurulmuştur.

$$\sigma = r_1^2 - r_2^2 \quad (3.30)$$
$$\beta = x_1^2 - x_2^2 + y_1^2 - y_2^2 \quad (3.31)$$
$$\delta = 2x_1 - 2x_2 \quad (3.32)$$
$$\Upsilon = 2y_1 - 2y_2 \quad (3.33)$$

Denklemlerde belirtilen sabitler (3.29)'da yerine konularak aşağıdaki eşitlikler oluşturulmuştur.

$$\mathcal{E} = \beta - \delta x - \Upsilon y \quad (3.34)$$
$$g = \delta^2 - \Upsilon^2 \quad (3.35)$$
$$x = \frac{\beta - \mathcal{E} - \Upsilon_y}{\delta} \quad (3.36)$$

Robotun x noktasındaki konumu, denklem (3.36) ile bulunur. x noktası bulunduktan sonra (3.37)'de, x değeri yerine yazılarak robotun bulunduğu y konumu tespit edilir.

$$r_1 = x_1^2 - 2x_1\left(\frac{\beta - \mathcal{E} - \Upsilon_y}{\delta}\right) + \left(\frac{\beta - \mathcal{E} - \Upsilon_y}{\delta}\right)^2 + y_1^2 - 2y_1y + y^2 \quad (3.37)$$

Burada ikinci dereceden bir denklem oluşur. Denklemin kökleri bulunarak robotun iki çemberin kesişim noktalarından hangisinde olduğu tespit edilir.

$$Ay^2 + By - e = 0 \tag{3.38}$$

$$\Delta = B^2 - 4ae \tag{3.39}$$

İkinci dereceden bir bilinmeyenli denklemin iki adet kökü bulunmuştur (3.40-3.41). Çemberlerin kesişim noktası iki adet olduğundan dolayı bu iki kök kesişim noktalarını verir. r_1 ve r_2 yarıçap uzunluklarının birbirlerine olan durumlarına göre robotun konumu belirlenir.

Eğer $r_1 < r_2$;

$$y_1 = \frac{-B-\sqrt{\Delta}}{2A} \tag{3.40}$$

Eğer $r_1 > r_2$;

$$y_2 = \frac{-B+\sqrt{\Delta}}{2A} \tag{3.41}$$

Bu tez kapsamında gösterilen matematiksel yöntem kullanılarak robotun odometri verisinin hatalı olduğu durumlarda robotun konumunun iki nesnenin konumuna göre belirlenmesi amaçlanmıştır. Nesnelerin x ve y koordinatları sabittir. Robot ortamda bulunan nesneleri gördüğü anda tanır ve eğitilen derin ağı kullanarak hangi nesnenin hangi konumda olduğunu belirler. Böylece yukarıdaki matematiksel denklemleri kullanarak kendi konumunu doğru belirler. Deneysel çalışmalarda (Bölüm 4.1) sonuçlar ve başarı oranları daha detaylı bir şekilde verilmiştir.

3.5. Yol Planlaması

Haritası bilinen bir ortamda robotun konumunun belirlenmesi önemli bir problem olarak bilinmektedir. SLAM algoritmaları yardımıyla robot ortamı haritalandırırken aynı zamanda bu ortamda kendisini konumlandırır. Bu konumlandırma işlemi, sonradan gerçekleşebilecek olası bir yol planlama görevi için de önemlidir. Robotun belirli bir

başlangıç noktasından gösterilen bir hedef noktaya en kısa yoldan ulaşması işlemine yol planlaması denir. Yol planlamasında robot ortamda bir engelle karşılaşırsa bu engelden sakınarak tekrar bir yol planlama yapıp hedefe ulaşmak durumundadır. Yol planlaması probleminde birçok algoritma kullanılmaktadır. A* algoritması, Parçacık Filtresi, RRT ve Dijkstra en sık kullanılan yol planlama yöntemleridir.

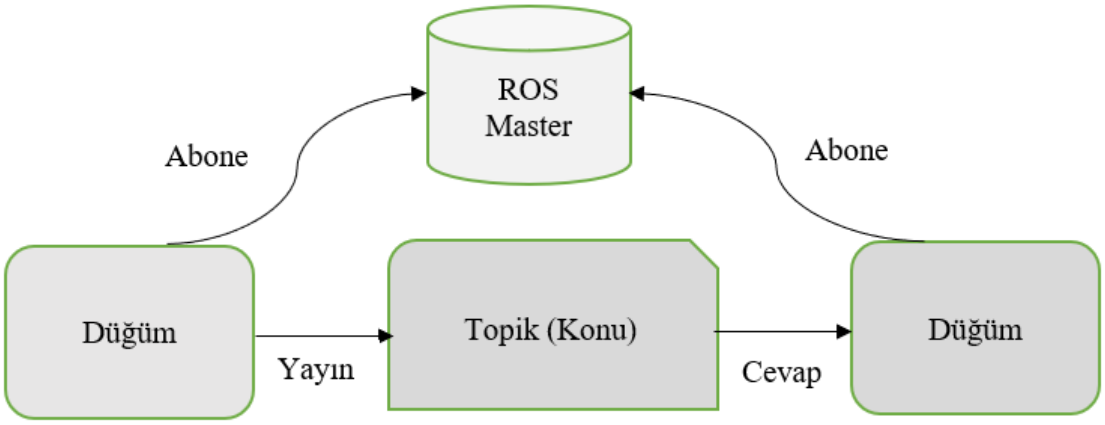
Bu tez çalışmasında sıklıkla tercih edilen A* algoritması ve parçacık filtresi ile parçacık dağılımını tayin eden AMCL yöntemi kullanılmıştır.

3.6. Robot İşletim Sistemi (ROS)

Robot İşletim Sistemi (ROS) olarak adlandırılan sistem aslında bir işletim sistemi değildir. Linux tabanlı bir işletim sistemine entegre olarak çalışabilen açık kaynak kodlu bir yazılımdır. ROS'un amacı açık kaynaklı bir yazılım ortamı oluşturup robotik sistemlerin kullanımını ve programlanmasını kolaylaştırmaktır. ROS dahilinde olan standart kütüphaneler ve desteklediği robotların yanı sıra geliştiriciler tarafından yazılmış kütüphanelere de destek verdiği için tercih edilmektedir. Hazır paketler ve kodlar üzerinde birkaç değişiklik yapılarak başka bir robot üzerinde kolaylıkla kullanılabilir.

ROS ilk olarak 2007 yılında Standford Yapay Zeka Laboratuvarın'da geliştirilmeye başlanmıştır. 2008 ve 2013 yıllarında Willow Garage enstitüsünde üzerinde çalışmalar yapılmış ve bu tarihten günümüze kadar gelişimini hızlı bir şekilde sürdürmüştür. Robotun donanımı ve yazılımı arasında optimum ilişki kurması amaçlanmıştır. ROS'un temel yapısında bir yayıncı-abone ilişkisi vardır. Robotun üzerinde bulunan sensörlerden birinin verisine ihtiyaç duyulduğunda kolaylıkla bu yayıncıya abone olunarak veriler alınabilir. Bu durumdan yola çıkarak sensörün yayın yaptığı ve sistemin sensöre abone olması durumunda verileri alabildiği görülür.

ROS'un ilk ortaya çıkışında robotik çalışmaların hızlandırılması amacı vardır. Çünkü geliştirilen algoritmalar başkalarıyla paylaşıldığı zaman daha da geliştirilebilir hal almaktadırlar. Bir araştırmacı bir problemi çözdüyse başka bir araştırmacı o problemi tekrar çözmek yerine, çözümü geliştirerek çalışmayı ilerletebilmektedir. ROS, paketler ve servisler özelinde çalışır. ROS MASTER bütün işletim sisteminin çalıştırılmasını sağlar. Bu işletim sistemine bağlı düğümler ve bu düğümlerden alınan bilgiler bulunmaktadır. Şekil de görüldüğü gibi ROS MASTER a bağlı olarak düğümlerde yayıncı ve çağırıcı konulu mesajlar bulunmaktadır (bkz Şekil 3.14).



Şekil 3.14 ROS master, düğümler ve konu döngüsü

Sistem, robotun üzerine bağlı birçok sensörden aldığı verileri ayrı ayrı tutar. Bir sensör verisine ihtiyaç olduğu takdirde o düğüme abone olunur ve veriler çekilir. Böylece hem sensörlere bir zarar gelmesi durumunda sistem etkilenmez hem de gereksiz veri akışının önüne geçilerek sistem hızlandırılmış olur. ROS sistemi yayıncı/abone mesajlaşma sistemini kullanır ve diğer düğümlerle haberleşir. Düğümler içerisinde hesaplamalar yapılabilmektedir. Örneğin bir düğüm LIDAR nokta bulutu verilerini alır. Diğer düğüm bu verileri işler ve son düğümde veriyi kullanır. ROS Master bu düğümlerin birbiriyle haberleşebilmesi ve veri aktarılabilmesi için gereklidir.

ROS'un bir diğer avantajı kullanıcıyı tek bir dil kullanmaya mecbur etmemesidir. ROS python ve C++ dilleriyle yazılmıştır. İçerdiği paketlerde de genellikle Python ve C++ dilleri kullanılmaktadır. ROS çoğunlukla Linux/Ubuntu işletim sistemi üzerinde kullanılmaktadır. Ancak son yıllarda MATLAB programında da ROS paketleri kullanılmaya başlanmıştır.

ROS görselleştirme programı olarak Gazebo ile tam uyumludur. Rviz arayüzü sayesinde Gazebo ortamındaki robot hareketleri ve sensör verileri gözlenir. Yapılan çalışmalar öncelikle bu simülasyon ortamında denenip sonuçları görülebilmektedir. İlk sürümü 2010 yılında "Box Turtle" adıyla piyasaya sürülmüştür. En yaygın kullanılan sürümü şu an aktif olan "Kinetic" sürümüdür ve son çıkan sürümü 2023 yılına kadar desteklenen "Melodic Morenia" sürümüdür. Tez kapsamında Ubuntu 16.04 ile desteklenen "Kinetic Kame" sürümü kullanılmıştır (bkz Tablo 3.1).

Tablo 3.1 ROS yayıncı versiyonları ve destek tarihleri

ROS Yayıncı	Tarih	Destek Tarihi
ROS Box Turtle	02.08.2010	-------
ROS Indigo Igloo	22.07.2014	05.2019
ROS Kinetic Kame	23.05.2016	05.2021
ROS Lunar Loggerhead	23.05.2017	05.2019
ROS Melodic Morenia	23.05.2018	05.2023

Robot işletim Sistemi 2010 yılından bugüne kadar sürekli bir gelişim göstermiştir. Birçok şirket ürettikleri robotları, sensörleri veya sürücüleri artık ROS a uyumlu olarak yapmaktadırlar. Bu da büyük bir avantaj sağlamaktadır. Çünkü yapılan birçok robotik çalışma bu platform altında yapılmaktadır. Hazır paketler kullanılarak daha önceden yapılmış olan çalışmaları hızlı bir şekilde geliştirebilecek ortam sunulmuştur.

4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Bu bölümde tez kapsamında yapılan araştırmalar ve sonuçları verilmiştir. İlk olarak Bölüm 4.1'de kapalı ortamda robot konumlandırması yapılmıştır. Otonom hareket eden bir robotun teker veri kaybı sonucunda meydana gelen konumlama hatası önerilen derin öğrenme tabanlı bir nesne tanıma yöntemiyle giderilmiştir.

Bölüm 4.2'de bir SLAM algoritması olan HectorSLAM kullanılarak robotun kapalı bir mekânı haritalandırması ve ölçeklendirmesi amaçlanmıştır. Bu haritalama işlemi tek sensör kullanılarak yapılmış ve teker odometri bilgisine ihtiyacı ortadan kaldırmıştır.

4.1. Derin Öğrenme Ağı ile Nesne Tanıma Tabanlı Kapalı Ortam Robot Konumlandırması ve Oluşturulan Araştırma Ortamı

Tez kapsamında yapılan ilk çalışmada nesne tanıma tabanlı bir konumlandırma yöntemi önerilmiştir. Kapalı ortamlarda robot konumlandırması GPS verilerinin alınamamasından dolayı önemli bir konudur. SLAM yöntemleri kullanılarak bilinmeyen bir ortamda robotun harita oluştururken kendisini konumlandırması sağlanmaktadır. Bu konumlandırma odometri teker sensörleri aracılığıyla çeşitli tahmin yöntemleri kullanılarak yapılabilir. Ancak sensör verilerinin gürültü veya başka bir dış etkene bağlı olarak bozulması sonucunda hatalı konum tahminleri meydana gelir. Bundan dolayı ortamda bulunan nesnelere bağlı bir konumlandırma çalışması ortaya koyulmuştur. Deneysel çalışmalar için Konya Teknik Üniversitesi'nde bulunan RACLAB (Robotic Automation Laboratory) laboratuvarında bir deney parkuru oluşturulmuştur. Şekil 4.1' de oluşturulan parkurun haritası oluşturularak robotun ortamda otonom hareket etmesi amaçlanmıştır. Ortamda konumu bilinen iki nesneye göre robotun konumlandırılması amaçlandığından dolayı iki nesne parkura Şekil 4.1' deki gibi yerleştirilmiştir. Bu nesneler 2 adet trafik işareti olarak belirlenmiştir. Derin öğrenme kullanılarak trafik işaretlerinin tanınması ve bu işaretlerin ortamdaki konumlarının alınarak robotun konumlandırılması sağlanmıştır.

Şekil 4.1 RACLAB laboratuvarında oluşturulan araştırma ortamı ve levhaların konumları.

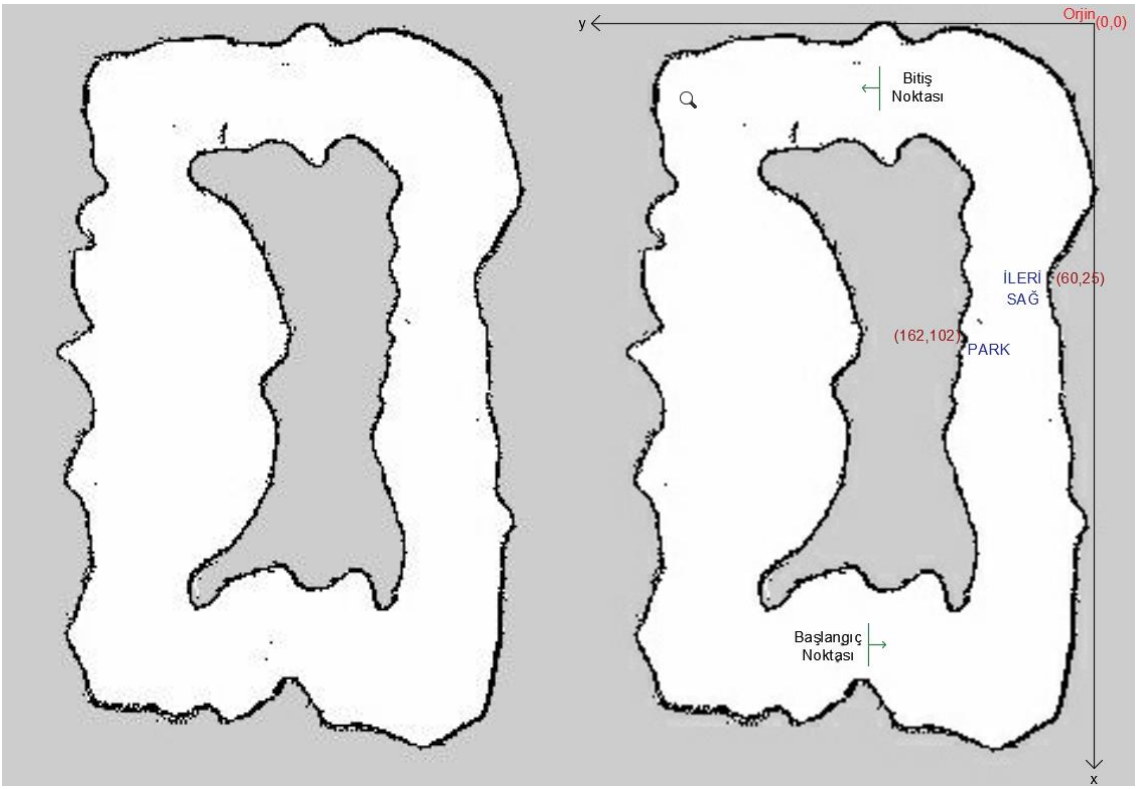
Monte Carlo Localization (MCL) yöntemi robotik konumlandırmada bilinen bir yöntemdir. Parçacık Filtresi temelli bu yöntem ile robot sensör verilerini kullanarak ortamda konum tahmini gerçekleştirir. Önerilen yöntemde ise robot yolu üzerinde bulunan iki trafik işaretine bağlı olarak konumunu belirlemektedir. Deneyde bu iki yöntem farklı senaryolarda ele alınmış ve doğruluk oranları karşılaştırılmıştır.

4.1.1. Araştırma Ortamının Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM) Yöntemi ile Haritalandırılması

ROS üzerinde hazır paketleri bulunan SLAM algoritmalarından en çok bilinenler HectorSLAM ve Gmapping algoritmalarıdır. HectorSLAM algoritması ışınların uç uca sıralanması yöntemini kullanarak, odometri verisi olmadan bir haritalama gerçekleştirir. Gmapping algoritmasında ise robotun hareketleri yanı sıra son gözlemleri de dikkate alarak doğru bir tahmin dağılımı hesaplama yöntemi önerilmiştir (Grisetti ve ark., 2005). Gmapping algoritmasıyla beraber tükenen parçacıkların yeniden örnekleme işlemleri ile geniş alanlarda avantajlı sonuçlar elde edilmiştir. Odometri teker verilerine bağlı olarak bir harita oluşturulur. Elde edilen odometri verisiyle birlikte robot, yol planlaması ve otonom olarak hedef noktaya varma gibi işlevler gerçekleştirir.

Gmapping algoritması odometri verisi avantajından ve daha yüksek doğrulukla harita oluşturmasından dolayı bu çalışmada tercih edilmiştir. Şekil 4.1'de verilen deney ortamında Turtlebot2 robotu ile Gmapping algoritması kullanılarak SLAM uygulaması yapılmıştır. Kullanılan SLAM yöntemi sonucunda oluşan deney ortamının haritası Şekil

4.2a'da verilmiştir. Şekil 4.2b' de oluşan haritanın koordinatları belirlenmiştir. Buna göre başlangıç noktası (orjin (0,0)) referans alınarak iki trafik işaretinin konumları belirlenmiştir.



(a) (b)

Şekil 4.2 Gmapping algoritması kullanılarak oluşturulan çalışma ortamının haritası (a), ortam haritası merkez koordinatları ve nesnelerin konumları (b).

Robot güzergahında bulunan levhalardan Park işaretinin global_map'e göre konumu x ekseninde 162 cm ve y ekseninde 102 cm (162,102) olarak belirlenmiştir. İleri Sağ işaretinin olduğu koordinatlar da x ekseninde 60 cm ve y eksenin 25 cm (60,25) olarak belirlenmiştir. Derin Öğrenme modeli olan Faster R-CNN ile eğitilen iki trafik işaretinin konum bilgileri algoritmaya verilmiştir. Dolayısı ile robot trafik işaretlerini tanıdığı anda bu işaretlerin hangi x ve y koordinatlarında olduğunu bilmektedir.

4.1.2. Nesne Tanıma Yöntemi ile Robot Konum Tahmini

Robotun otonom bir şekilde başlangıç noktasından hedefe giderken konum tahmini birçok tahmin filtresiyle yapılabilmektedir. Parçacık filtresi tabanlı Monte Carlo Lokalizasyon yöntemi, kullanılan bu konumlandırma yöntemlerinin en yaygınlarından biridir. Ancak bu yöntemin, herhangi bir odometri bozucu etkisi sonucunda hatalı

tahminlerde bulunabildiği bilinmektedir. Eğer odometri verisi bozucu bir etkiye maruz kalırsa konum tahmini kümülatif olarak hatalı tahmin edilir. Önerilen nesne tanıma yöntemi ile bu gibi durumlarda ortaya çıkacak konum tahmin hataları, iki trafik işaretinin konum bilgisi referans alınarak giderilebilir. Algoritma 4.1'de robotun iki trafik işaretinin konumunu referans alarak kendi konum bilgisini belirlediği algoritma verilmiştir.

Algoritma 4.1 Robotun nesne tanıma yöntemi ile konumunun belirlenmesi algoritması

Algoritma : Otonom İlerlerken Robot Konumu Belirleme Algoritması

Giriş: Kinect Sensör ile Alınan Resimler

İşlem:

(1) Kinect kameradan referans görüntüler alınır
(2) Faster R-CNN ağı kullanılarak trafik işaretleri tanınır
(3) **if** Trafik İşareti Bulundu **then**
(4) Trafik İşaretlerinin Mesafelerini Hesapla $\boldsymbol{r_1}$ ve $\boldsymbol{r_2}$
(5) $\boldsymbol{\beta = x_1^2 - x_2^2 + y_1^2 - y_2^2}$
(6) $\boldsymbol{\delta = 2x_1 - 2x_2}$
(7) $\Upsilon = \boldsymbol{2y_1 - 2y_2}$
(8) $\mathcal{E} = \boldsymbol{\beta - \delta x - \Upsilon y}$
(9) g = $\boldsymbol{\delta^2 - \Upsilon^2}$
(10) **if** $\boldsymbol{r_1 > r_2}$ **then**
(11) $\mathbf{x} = \frac{\boldsymbol{\beta - \mathcal{E} - \gamma . y}}{\boldsymbol{\delta}}$
(12) $\mathbf{y} = \frac{\boldsymbol{-f + \sqrt{\Delta}}}{\boldsymbol{2 * g}}$
(13) **else**
(14) $\mathbf{x} = \frac{\boldsymbol{\beta - \mathcal{E} - \gamma . y}}{\boldsymbol{\delta}}$
(15) $\mathbf{y} = \frac{\boldsymbol{-f - \sqrt{\Delta}}}{\boldsymbol{2 * g}}$

Deneysel çalışmalarda Şekil 4.3a'da gösterilen Turtlebot 2 robotu kullanılmıştır. Robotta bulunan LIDAR sensör ile SLAM yapılmış ve Kinect kamerası ile de görüntüler alınmıştır. Algoritma 4.1'de gösterildiği gibi giriş resimleri Kinect sensör ile alınıp önceden eğitilmiş olan ağa verir ve Bölüm 3.4'te anlatılan matematiksel adımlar izlenerek robot konumu iki işaretin konumunu bağlı olarak tahmin edilir.

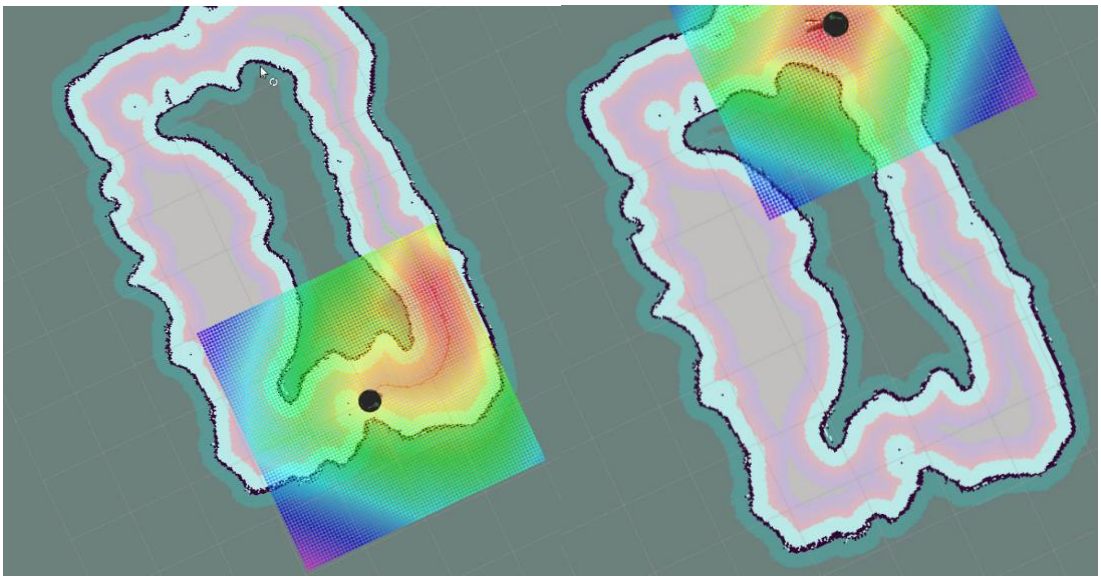
(a) (b)

Şekil 4.3 Turtlebot 2 robotu (a) ve Kinect Kamera (b)

Robot, Kinect derinlik kamerasını (bkz şekil 4.3b) kullanarak tanıdığı park ve ileri sağ levhalarına olan uzaklığını belirler. Ayrıca iki trafik işaretinin de x ve y koordinatları bilindiğinden dolayı önerilen matematiksel yöntemle iki işarete göre robot kendi konumunu hesaplar.

Deneysel çalışmalar üç senaryo üzerine kurularak gerçekleştirilmiştir. İlk senaryoda robotun başlangıç noktasından hedef noktasına otonom hareketi amaçlanmış ve bu otonom hareket sırasında herhangi bir odometri verisi kaybı oluşturulmamıştır. Bu noktadan hareketle klasik konumlandırma yöntemi olan MCL yöntemi ile konumlandırma sonuçları elde edilmiştir.

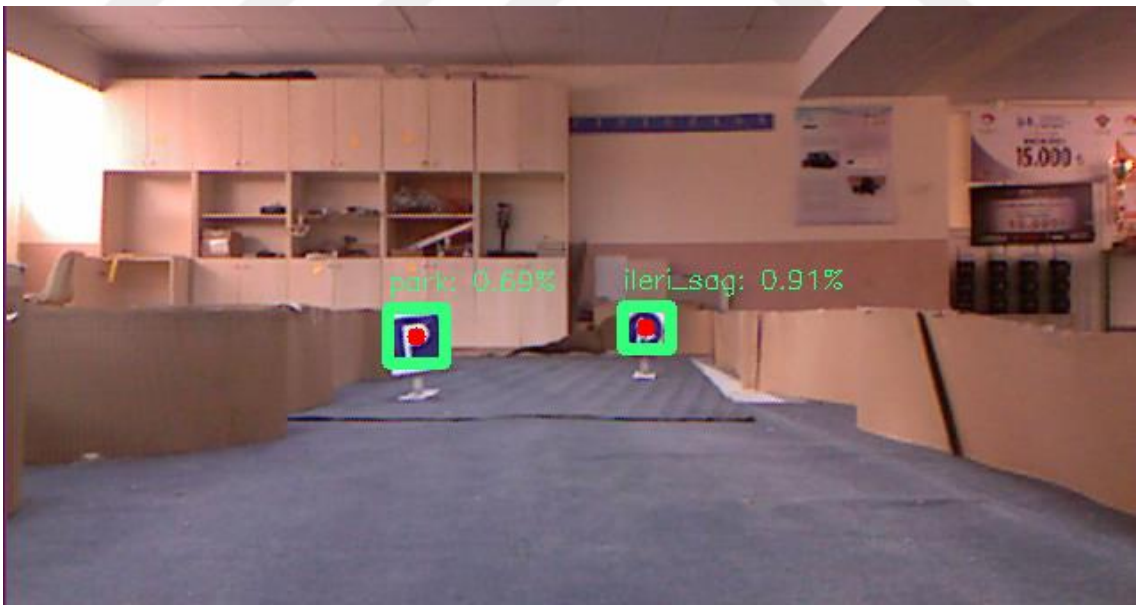
Şekil 4.4' te robotun teker odometri verisi kaybı olmadan ortamda ilerlemesi sırasında Kinect kameradan alınan bir görüntüsünü göstermektedir. Bu görüntüye göre robot levhalara yaklaşmakta ve bu levhaları tanımaktadır. Şekil 4.4' te ROS' un Rviz ara yüzünde robotun başlangıç noktasından hedef noktasına ilerlemesi gösterilmiştir. Robot gerçek konumda y ekseninde 60 cm' lik bir uzaklıkla ilerlemiş ve hedef noktasına varmıştır. MCL yöntemi herhangi bir teker açı değeri kaybı olmadığında yüksek doğrulukla konum tahmini gerçekleştirmiştir.



(a) (b)

Şekil 4.4 Rviz (ROS görselleştirme arayüzü (bkz Bölüm 3.6)) ortamında robotun başlangıç noktasından (a), hedef noktasına kadar izlediği yol (b).

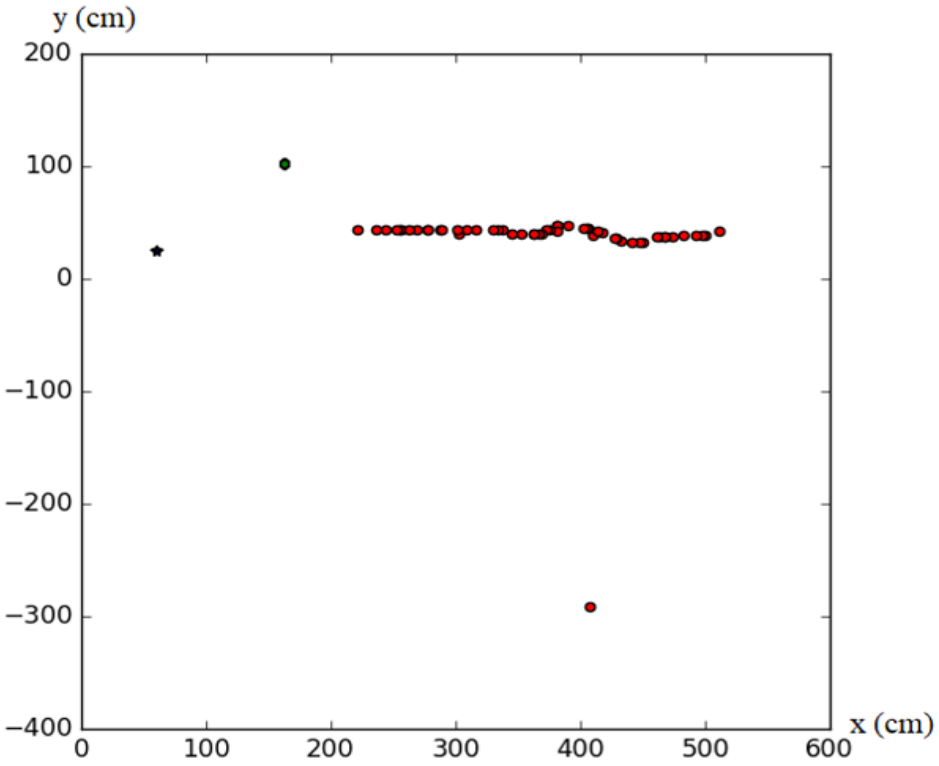
Şekil 4.5' te robotun Kinect kamerasından alınan bir görüntü verilmiştir. Bu görüntüde robotun hareketi sırasında iki trafik işaretini başarılı bir şekilde tanıdığı görülmektedir.



Şekil 4.5 Teker veri kaybı olmadığı durumda robotun otonom hareketi

Robotun tanıdığı iki trafik işareti Park ve İleri_sag trafik işaretleridir. Bu işaretlerin ortamdaki konumları bilinmektedir (Park (162,102) ve İleri_sag (60,25)). Robot yolu üzerinde bulunan iki trafik işaretini de tanıdığı anda bu konum bilgilerini alır

ve Bölüm 3.4’ te anlatılan matematiksel yöntemle kendi konumunu hesaplamıştır. Önerilen derin öğrenme ile nesne tanıma tabanlı yöntemle elde edilen konum tahmini Şekil 4.6’da verilmiştir. Şekildeki yıldız ve yuvarlak iki siyah nokta Park (x=162, y=102) ve İleri Sağ (x=60, y=25) levhalarının konumlarını vermektedir. Kırmızı noktalar ise robotun anlık x ekseninde 600 cm’den 200 cm’ ye konum bilgisini vermektedir. Önerilen yöntem ile gerçek konumu y ekseninde 60 cm olan robotun konum tahmini ortalama 56,93 cm olarak belirlenmiştir.

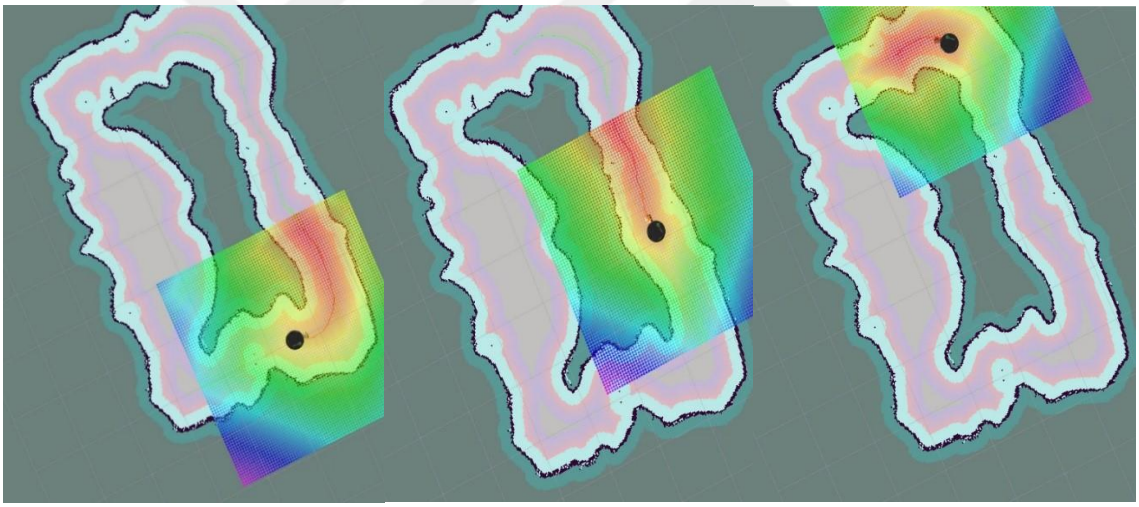


Şekil 4.6 Önerilen yöntem ile robotun konumu. Park levhası konumu (x=162, y=102) ve İleri Sağ levhası konumu (x=60, y=25).

Elde edilen sonuçlara göre birinci senaryoda önerilen yöntem ile %93,98 oranında bir doğruluk elde edilmiştir. Sonuç olarak eğer herhangi bir teker odometri verisi kaybı veya gürültüsü yoksa MCL yöntemi daha doğru sonuç vermiştir. Ancak önerilen yöntem de yüksek bir doğruluk oranıyla robotun konumunu belirlemeyi başarmıştır.

4.1.3. Teker Kayması ve Teker Açı Verilerinin Kaybolması Sonucunda Önerilen Yöntem ile Konumlandırma

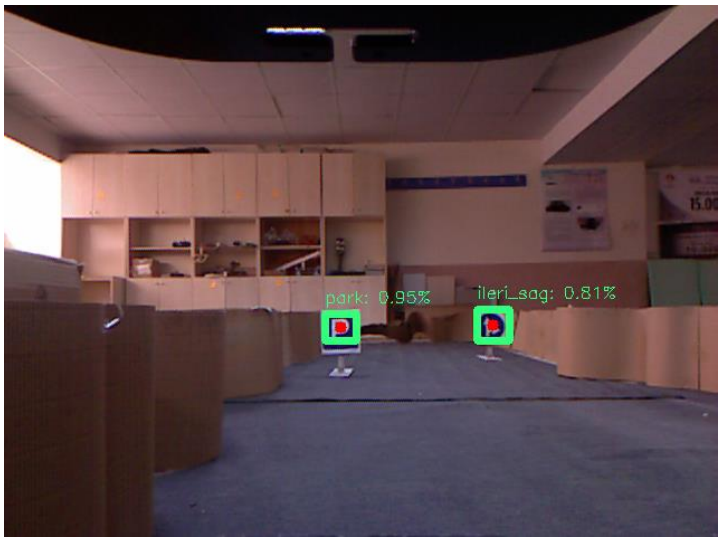
Bölüm 4.1.2' de birinci senaryo oluşturulmuş ve robotun herhangi bir teker odometri veri kaybı olmadığı durumda konum tahmini, MCL ve önerilen yöntemle belirlenmiştir. İkinci senaryoda ise robot kaçırma problemi ele alınarak teker verilerinin kaybolması durumunda konum tahmini ele alınmıştır. Referans bir konum bilgisi ele alınması açısından robotun gerçek konumu ilerlediği güzergahta y ekseninde 100 cm olarak belirlenmiştir. Park levhası hizasından robotun otonom olarak hedef noktasına hareketi sağlanmıştır. ROS' un Rviz ara yüzünde oluşan robot hareketi Şekil 4.7' de verilmiştir. Teker odometri veri kaybı olduğundan dolayı robot MCL yöntemiyle yanlış konum tahmininde bulunmuştur. Gerçek konumu y ekseninde 100 cm olan robotun konumu MCL ile 60 cm olarak belirlenmiştir. Gerçek konuma göre MCL yöntemi ile 40 cm' lik bir sapma meydana gelmiştir.



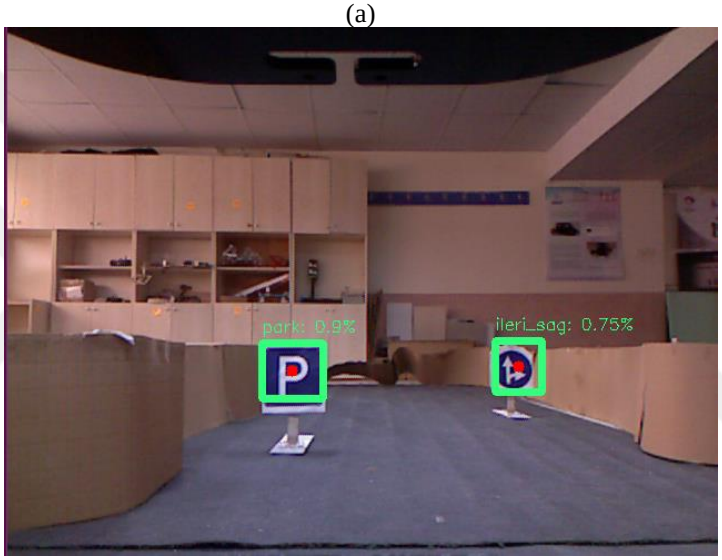
(a) (b) (c)

Şekil 4.7 Robotun başlangıç noktasından hedef noktasına otonom hareketi sırasında MCL ile konum tahmini. (a)-(b)-(c) başlangıç noktasından bitiş noktasına kadar sırayla izlenen yol.

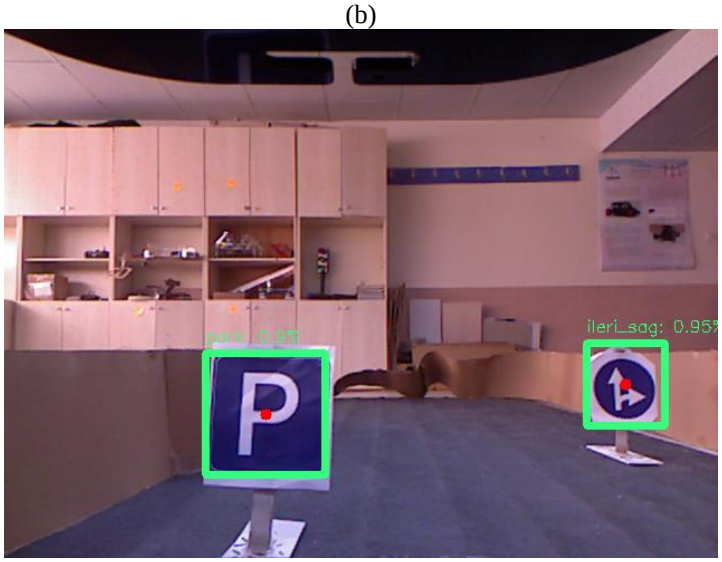
Şekil 4.8' de robotun otonom hareketi sırasında Kinect kameradan alınan görüntü verilmiştir. Robot hareketi esnasında yolu üzerinde olan iki trafik işaretini başarılı bir şekilde tanımıştır.



(a)



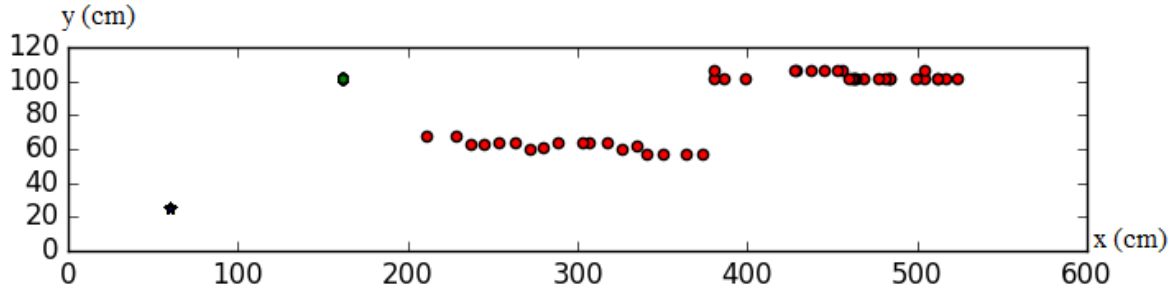
(b)



(c)

Şekil 4.8 Trafik işaretlerine göre robotun sol tarafta bulunan park levhası hizasından ilerlemesi durumu. (a)-(b)-(c) robotun otonom ilerlediği görüntüler ve levhalara yaklaşma durumu.

Derin öğrenme modeli olan Faster R-CNN kullanılarak eğitilen veri seti sonucunda robotun işaretleri tanıması sağlanmıştır. Her bir işaretin genel haritadaki konum bilgileri alınmıştır. Bu konum bilgileri kullanılarak Bölüm 3.4' te gösterilen matematiksel yöntemle robotun konum tahmini sağlanmıştır. Şekil 4.9' da gösterildiği gibi robot iki nesneye göre kendi konumunu belirlemiş ve her bir konum belirlemesi sırasında bir kırmızı nokta ile koordinatları işaretlemiştir. Ayrıca trafik işaretleri de siyah yıldız ve yuvarlak nokta olarak Şekil 4.9' da görülmektedir.



Şekil 4.9 Robotun solda bulunan Park levhası referans alınarak ilerlemesi (0-600 cm x ekseni ve 0-120 cm y ekseni)

Robotun y ekseninde 100 cm gerçek konumda ilerlemesi sırasında önerilen yöntemle konum tahmini ortalama y ekseninde 89,39 cm olarak hesaplanmıştır. MCL yöntemi ile 60 cm olan hatalı konum tahmini önerilen yöntem kullanılarak daha doğru bir sonuçla elde edilmiştir.

İkinci senaryoda robotun teker verisi kaybı sonrasında konum tahmini iki yöntemle de gerçekleştirilmiştir. Bu yöntemlere göre önerilen yöntem % 89,39 oranında bir doğruluk elde etmiştir. Teker veri kaybı sonucunda MCL yönteminin hatalı konum tahmininin önerilen derin öğrenme temelli nesne tanıma yöntemiyle başarılı bir şekilde giderildiği görülmüştür. Buna göre ortamda herhangi iki noktada bulunan nesnelerin konumlarının bilinmesi halinde robotun kendisini bu nesneleri tanıyarak ve konumlarını bilerek konumlandırabildiği sonucuna varılmıştır.

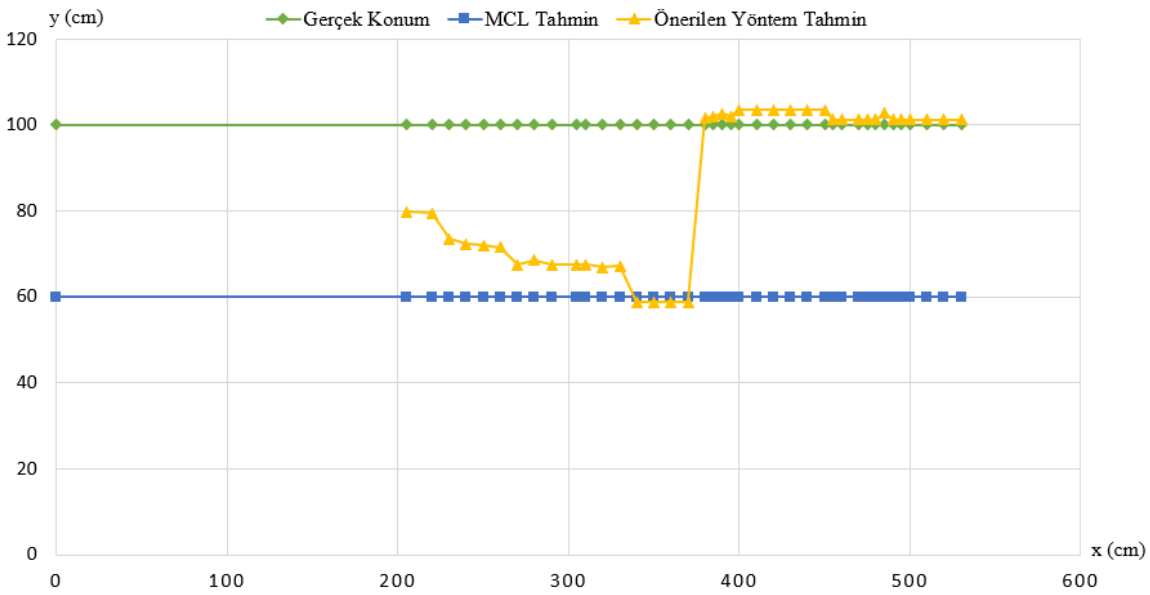
Tablo 4.1'de görüldüğü gibi robotun aldığı yol boyunca x ve y koordinatlarındaki konumlarının ortalama değerleri verilmiştir. Robot y ekseninde 102 cm de bulunan Park levhası hizasından hareketine başlamıştır. 100 cm, y ekseninde gerçek konumda ilerlemiş ve hedef noktasına otonom bir şekilde ulaşmıştır. Monte Carlo Localization ile robotun odometri verisi yanlış olduğundan dolayı 60 cm y ekseninde konum tahmini yapılmıştır. Ancak robot gerçekte 100 cm y ekseninde ilerlemiştir. Önerilen nesne tanıma tabanlı

yöntem ile iki levhaya göre robotun y ekseninde ortalama 89,39 cm ile ilerlediği görülmüştür. Dolayısı ile ilk yöntemle ortalama %60' lık bir doğruluk oranı elde edilirken; önerilen yöntem ile doğruluk oranının %89,39 olduğu görülmüştür.

Tablo 4.1 Park levha hizasında robotun ilerlemesi ile MCL yöntemi ve önerilen konum tahmini yöntemi doğruluk oranları

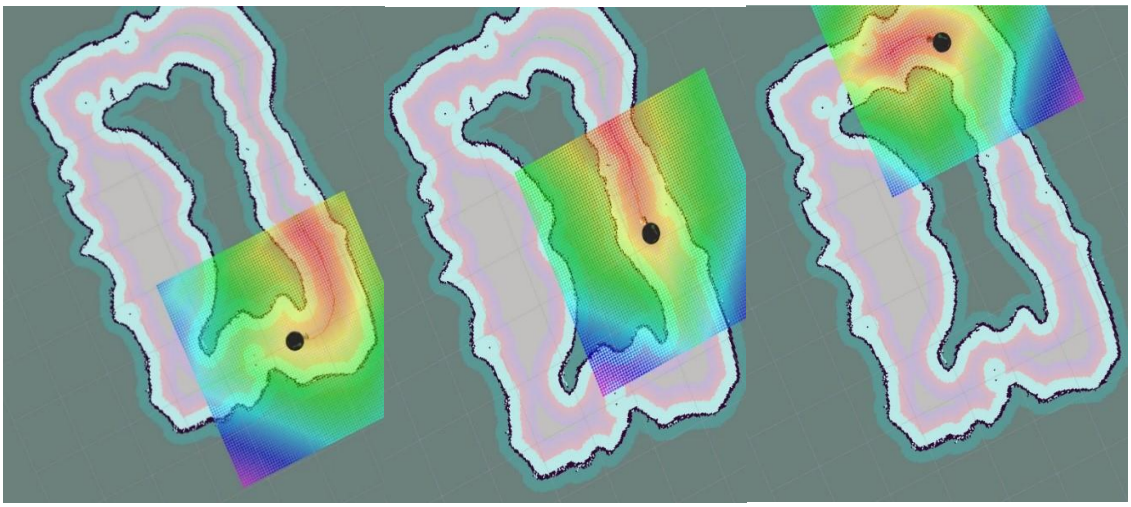
Koordinatlar	Y (cm)	X (cm)	Doğruluk Oranı (%)
Gerçek Konum	100 cm	600 cm'den 200 cm' ye kadar doğrusal ilerleme	
Monte Carlo Lokalizasyon ile Konum	60 cm	600 cm'den 200 cm' ye kadar doğrusal ilerleme	%60
Nesne Tanıma Yöntemi ile Konum	89,39 cm	600 cm'den 200 cm' ye kadar doğrusal ilerleme	%89,39

Şekil 4.10'da robotun gerçek konumu ve kullanılan iki yöntemin konum tahmin sonuçları grafikle gösterilmiştir. Yeşil çizgi robotun gerçek konumu olan 100 cm' yi, mavi çizgi MCL yöntemi ile tahmin edilen konum olan 60 cm' yi ve sarı çizgi ise nesne tanıma tabanlı önerilen yöntem konum tahminini vermektedir. Robot önerilen yöntemde konum tahmini yaparken iki levhanın konumuna ihtiyaç duymaktadır. Dolayısı ile iki levhayı tanıyıp hangisinin hangi koordinatta olduğu bilgisine erişmesi gerekmektedir. Bu bağlamda robot iki nesneyi tanıyana kadar herhangi bir konum belirlemesi yapamaz ve "none(hiç)" çıktısını verir. Bu nedenle Şekil 4.9'da bulunan sarı renkteki çizgi (önerilen yöntem kullanılarak yapılan konum tahmini) sıfır noktasından başlamıştır. Robot ilk başta konum tahminini 80 cm üzerinde yapmıştır. Ancak sonrasında yeşil çizgiye yaklaştığı ve oturduğu görülmüştür. Bazı sıçramaların nedeni Faster R-CNN ağıyla gerçek zamanlı olarak nesne tanımada hız problemidir. Sıçramalara rağmen doğru tahmin sonucu alınmış ve ortalama 89,39 cm y ekseninde konum tahmini yapılmıştır.



Şekil 4.10 Robotun park trafik işareti hizasında (gerçek konumu y ekseni 100 cm) ilerlemesi durumu. Yeşil çizgi robotun gerçek konumu, mavi çizgi MCL yöntemiyle tahmin, sarı çizgi önerilen yöntemle konum tahmini

Deneysel çalışmada gerçekleştirilen üçüncü senaryoda yine teker veri kaybı sonucunda robotun konum tahmini yapılmıştır. Referans bir konum bilgisi ele alınması açısından robotun gerçek konumu ilerlediği güzergahta y ekseninde 35,5 cm olarak belirlenmiştir. Bu kez İleri_sağ levhası hizasından robotun otonom olarak hedef noktasına hareketi sağlanmıştır. ROS' un Rviz ara yüzünde oluşan robot hareketi Şekil 4.11' de verilmiştir. Teker odometri veri kaybı olduğundan dolayı robot MCL yöntemiyle yanlış konum tahmininde bulunmuştur. Gerçek konumu y ekseninde 35,5 cm olan robotun konumu MCL ile 60 cm olarak belirlenmiştir. Gerçek konuma göre MCL yöntemi ile 25,5 cm' lik bir sapma meydana gelmiştir.

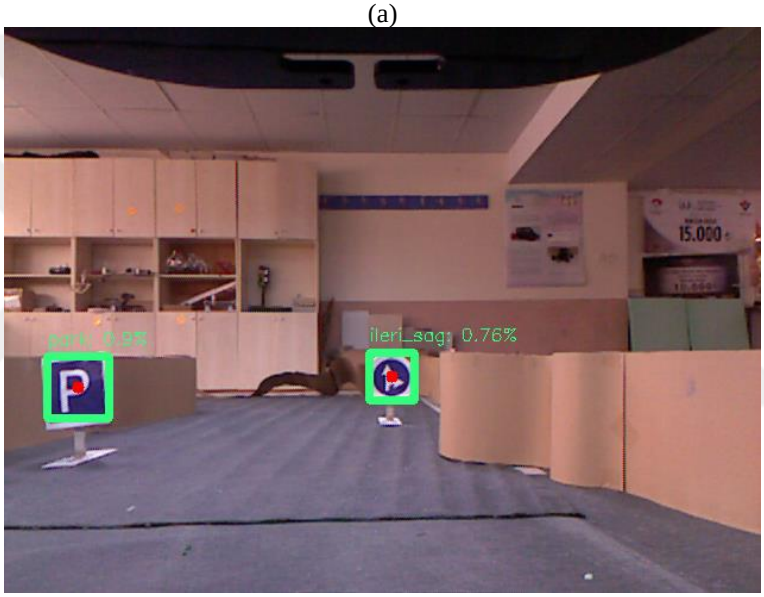


(a) (b) (c)

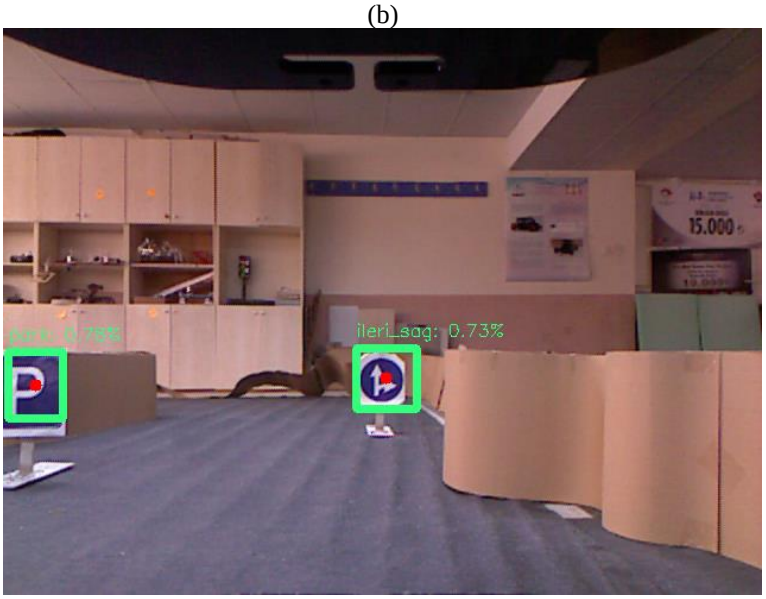
Şekil 4.11 Robotun başlangıç noktasından hedef noktasına otonom hareketi sırasında MCL ile konum tahmini. (a)-(b)-(c) başlangıç noktasından bitiş noktasına kadar sırayla izlenen yol.

Şekil 4.12’ de robotun otonom hareketi sırasında Kinect kameradan alınan görüntü verilmiştir. Robot hareketi esnasında yolu üzerinde olan iki trafik işaretini başarılı bir şekilde tanımıştır. Derin öğrenme modeli ve robot hızına bağlı olarak levhaların tanınma oranları robotun levhalara uzaklığına bağlı olarak değişmektedir.

(a)



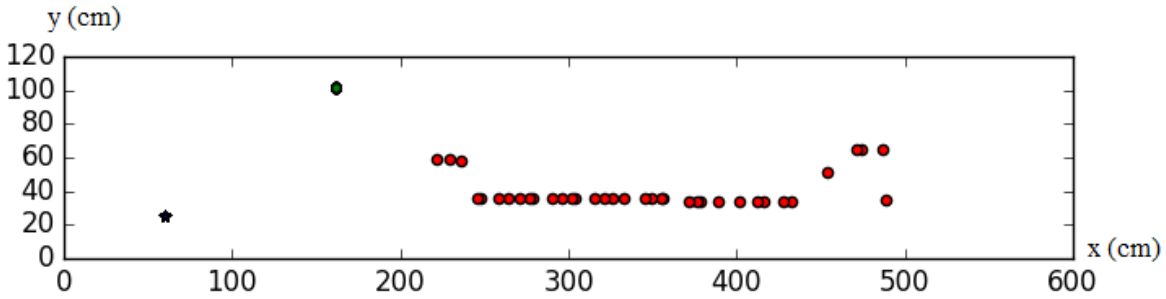
(b)



(c)

Şekil 4.12 Trafik işaretlerine göre robotun sağ tarafta bulunan ileri sağ levhası hizasından ilerlemesi durumu. (a)-(b)-(c) robotun otonom ilerlediği görüntüler ve levhalara yaklaşma durumu.

Derin öğrenme modeli olan Faster R-CNN kullanılarak eğitilen veri seti sonucunda robotun işaretleri tanıması sağlanmıştır. Her bir işaretin genel haritadaki konum bilgileri alınmıştır. Bu konum bilgileri kullanılarak Bölüm 3.4' de gösterilen matematiksel yöntemle robotun konum tahmini sağlanmıştır. Şekil 4.13' de gösterildiği gibi robot iki nesneye göre kendi konumunu belirlemiş ve her bir konum belirlemesi sırasında bir kırmızı nokta ile koordinatları işaretlemiştir. Ayrıca trafik işaretleri de siyah yıldız ve yuvarlak nokta olarak Şekil 4.13' de görülmektedir.



Şekil 4.13 Robotun İleri Sağ levhası hizasından ilerlerken tahmin edilen konumu (0-600 cm x ekseni ve 0-120 cm y ekseni)

Robotun y ekseninde 35,5 cm gerçek konumda ilerlemesi sırasında önerilen yöntemle konum tahmini ortalama y ekseninde 40,29 cm olarak hesaplanmıştır. MCL yöntemi ile y ekseninde 60 cm olan hatalı konum tahmini önerilen yöntem kullanılarak daha doğru bir sonuçla elde edilmiştir.

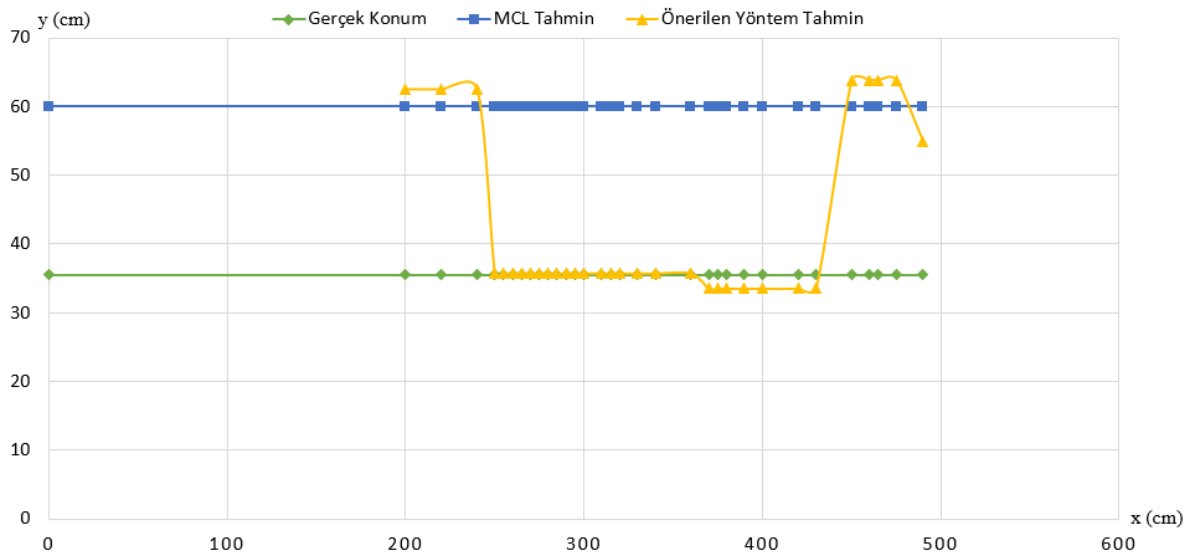
Üçüncü ve son senaryoda robotun teker verisi kaybı sonrasında konum tahmini iki yöntemle de gerçekleştirilmiştir. Bu yöntemlere göre önerilen yöntem %88,11 oranında bir doğruluk elde etmiştir. Teker veri kaybı sonucunda MCL yönteminin hatalı konum tahmininin önerilen derin öğrenme temelli nesne tanıma yöntemiyle başarılı bir şekilde giderildiği görülmüştür. Buna göre ortamda herhangi iki noktada bulunan nesnelerin konumlarının bilinmesi halinde robotun kendisini bu nesneleri tanıyarak ve konumlarını bilerek konumlandırabildiği sonucuna varılmıştır.

Tablo 4.2'de görüldüğü gibi robotun aldığı yol boyunca x ve y koordinatlarındaki konumlarının ortalama değerleri verilmiştir. Robot y ekseninde 102 cm de bulunan Park levhası hizasından hareketine başlamıştır. 35,5 cm y ekseninde gerçek konumda ilerlemiş ve hedef noktasına otonom bir şekilde ulaşmıştır. Monte Carlo Localization ile robotun odometri verisi yanlış olduğundan dolayı 60 cm y ekseninde konum tahmini yapılmıştır. Ancak robot gerçekte 35,5 cm y ekseninde ilerlemiştir. Önerilen nesne tanıma tabanlı yöntem ile iki levhaya göre robotun y ekseninde ortalama 40,29 cm ile ilerlediği görülmüştür. Dolayısı ile ilk yöntemle ortalama %60' lık bir doğruluk oranı elde edilirken; önerilen yöntem ile doğruluk oranının %88,11 olduğu görülmüştür.

Tablo 4.2 İleri Sağ levha hizasında robotun ilerlemesi ile MCL yöntemi ve önerilen konum tahmini yöntemi doğruluk oranları

Koordinatlar	Y (cm)	X (cm)	Doğruluk Oranı (%)
Gerçek Konum	35,5 cm	600 cm'den 200 cm'ye kadar doğrusal ilerleme	
MCL Yöntemi ile Konum Tahmini	60 cm	600 cm'den 200 cm'ye kadar doğrusal ilerleme	%59,16
Nesne Tanıma Yöntemi ile Konum	40,29 cm	600 cm'den 200 cm'ye kadar doğrusal ilerleme	%88,11

Şekil 4.14'te robotun gerçek konumu ve kullanılan iki yöntemin tahminleri grafik olarak verilmiştir. Yeşil çizgi robotun gerçek konumu olan 35,5 cm'yi, mavi çizgi MCL yöntemi ile tahmin olan 60 cm'yi ve sarı çizgi ise önerilen yöntem konum tahminini vermektedir. Robot önerilen yöntemde konum tahmini yaparken iki levhanın konum bilgisine ihtiyaç duymaktadır. Dolayısı ile iki levhayı tanıyıp hangisinin hangi koordinatta olduğu bilgisine erişmesi gerekmektedir. Bu bağlamda robot iki nesneyi tanıyana kadar herhangi bir konum belirlemesi yapamaz ve "none (hiç)" çıktısını verir. Bu nedenle Şekil 4.14'te sarı renkle önerilen konum tahmini sıfır noktasından başlamıştır. Robot ilk başta konum tahminini 60 cm üzerinde yapmıştır ancak sonrasında yeşil çizgiye yaklaştığı ve oturduğu görülmüştür. Sıçramalara rağmen iyi bir sonuç alınmış ve ortalama 40,29 cm y ekseninde konum tahmini yapılabilmiştir.



Şekil 4.14 Robotun ileri sağ trafik işareti hizasında (gerçek konumu y ekseni 35,5 cm) ilerlemesi durumu. Yeşil çizgi robotun gerçek konumu, mavi çizgi MCL yöntemiyle tahmin, sarı çizgi önerilen yöntemle konum tahmini

Bölüm 4.1'de gerçekleştirilen deneyler ışığında robot konumu MCL algoritması ve derin öğrenme tabanlı nesne tanıma yöntemleriyle tahmin edilmiş ve karşılaştırılmıştır. Teker veri kaybının olmadığı durumlarda klasik bir yöntem olan MCL yönteminin yüksek doğrulukla konum tahmini yaptığı bilinmektedir. Ancak teker veri kayıplarının veya robot kaçırılması gibi bir problemin ortaya çıktığı durumlarda konum tahmininin önerilen derin öğrenme tabanlı nesne tanıma yöntemiyle daha doğru sonuçlar verdiği görülmüştür. Böylelikle yapılan çalışmayla, özellikle savunma sistemlerinde önemli bir problem olan kapalı ortamlarda konumlandırma problemine bir çözüm sunulmuştur. Önerilen yöntem GPS verileri alınamadığı durumlarda veya askeri, arama kurtarma konularında dağ araları, tünel içleri gibi bölgelerde kullanılarak konumlandırma konusuna bir çözüm sunmaktadır.

4.2. Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM) Yöntemi Kullanılarak Kapalı Ortam Haritalandırma ve Ölçeklendirme

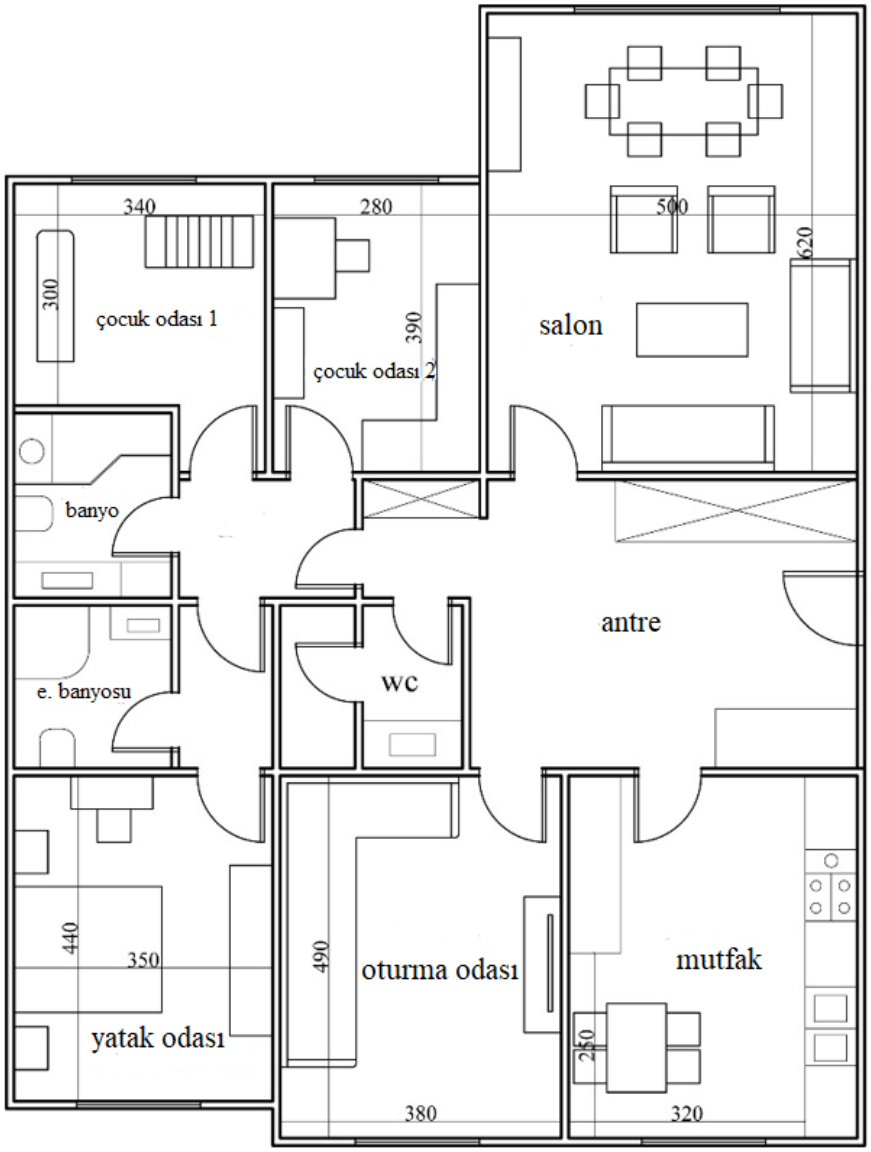
Kapalı ortamların haritalandırması ve robotların bu ortamlarda konumlanması konusunda yapılan çalışmalar sonucunda birçok alanda gelişmeler görülmüştür. Mimari yapılarda bu gelişmelerin görüldüğü alanlardan biridir. Kapalı ve dar iç mekanların ölçeklendirilmesi konusunda uygulanan geleneksel yöntemler vakit ve doğruluk

açısından verimli değildir. Özellikle tarihi yapıların restorasyonunda dokuya zarar vermeden ölçümler alınması gerekmektedir. Ayrıca tarihi binalarda ve yeraltı incelemelerinde insanların giremeyeceği ve ölçüm alınamayacağı alanlar bulunabilmektedir.

Günümüz gelişen teknolojisinin birçok alana entegre olarak çalışması sonucunda yapay zekanın kullanım alanı da artmıştır. Mimarlık alanında ve mekan dizaynı konusunda mekan ölçülerine bağlı olarak en uygun kullanım şekilleri oluşturulabilmektedir. Bu konuda da yine iç mekanların ölçülerinin ve boyutlarının bilinmesi gerekmektedir. Yapılan bu araştırmada iç mekanlar için SLAM algoritmaları kullanılarak haritalama ve ölçeklendirme uygulaması yapılmıştır. HectorSLAM algoritması kullanılarak teker odometri verisine ihtiyaç ortadan kaldırılmıştır. Bunun nedeni de aracın odometri verisine bağlı ölçeklendirme hatalarından kaçınmak ve hızlı bir haritalama sağlamaktır. LIDAR sensörü kullanılarak haritalandırılan ortam tek bir sensöre bağlı olmasından dolayı da avantaja sahiptir. Alınan sonuçlarda ortamın haritalandırması ile ölçeklendirilmesinin zaman maliyeti açısından büyük bir avantajı olduğu görülmüştür.

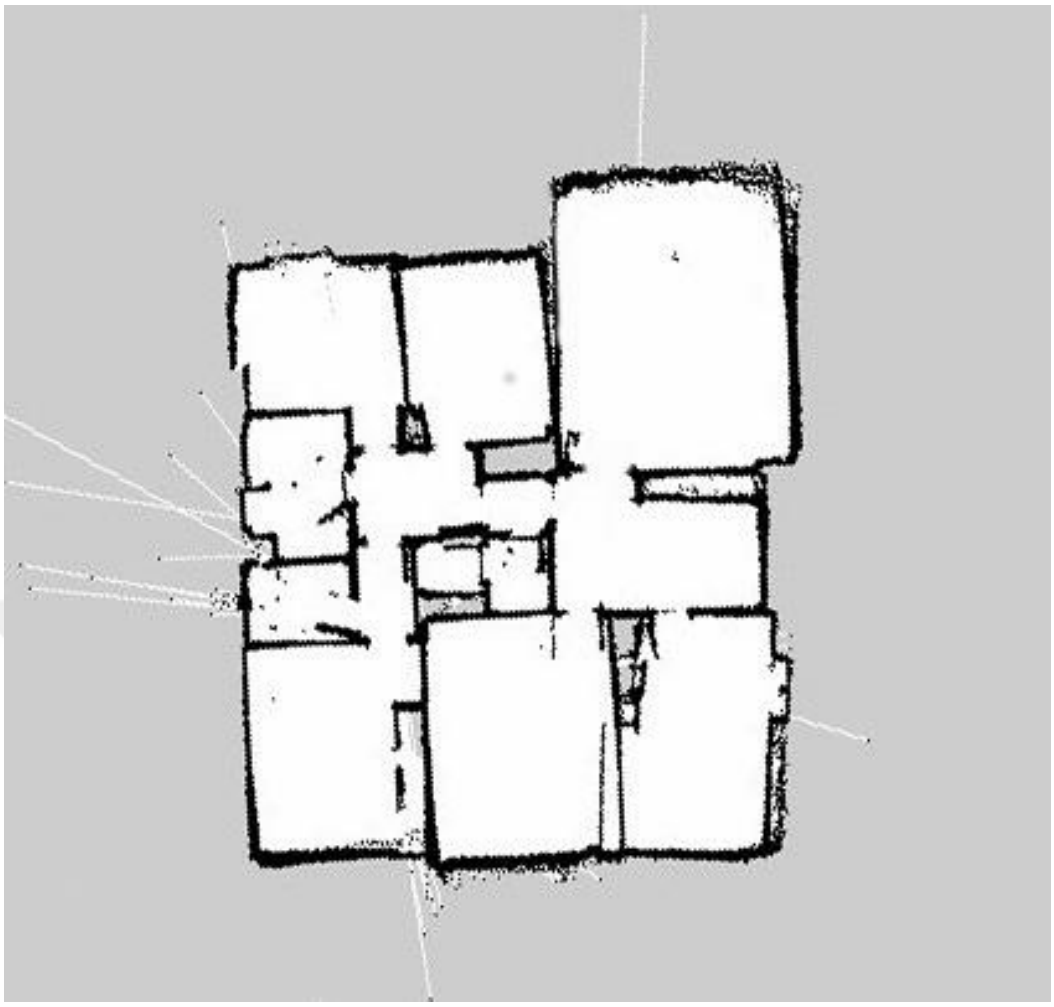
4.2.1. HectorSLAM Algoritması Kullanılarak İç Mekânın Haritalandırılması

Araştırma ortamı dört oda, bir salon, bir tuvalet, iki banyo, antre ve bir mutfaktan oluşmaktadır (bkz Şekil 4.15). Ortamın HectorSLAM algoritması kullanılarak haritası elde edilmiştir. Bu harita çıkarılırken LIDAR sensör el ile tüm alan içerisinde dolaştırılmıştır. Teker odometri verisine ihtiyaç duyulmayan bu yöntemde; odometri bilgisi LIDAR ışınlarının uç uca eşleştirilmesi yöntemi ile elde edilmiştir.



Şekil 4.15 Deney ortamının mimari planı

Gmapping algoritması kullanılsaydı daha başarılı bir harita oluşturulabilirdi. Ancak değişen iç mekân alanlarında robot teker açı değerlerinde hata ile karşılaşılma ihtimali artardı. Örneğin fayans gibi kaygan bir zemin üzerinde robotun tekerlerinin kayması ile odometri bilgisi hatalı olacaktır. Ayrıca HectorSLAM algoritması ile süre olarak da robotun tüm ortamı dolaşması sırasında meydana gelecek süre uzamasının önüne geçilmiş oldu. Robot İşletim Sistemi (ROS) üzerinden kontrol edilen SLAM algoritması ile mekânın haritası başarılı bir şekilde oluşturulmuştur.



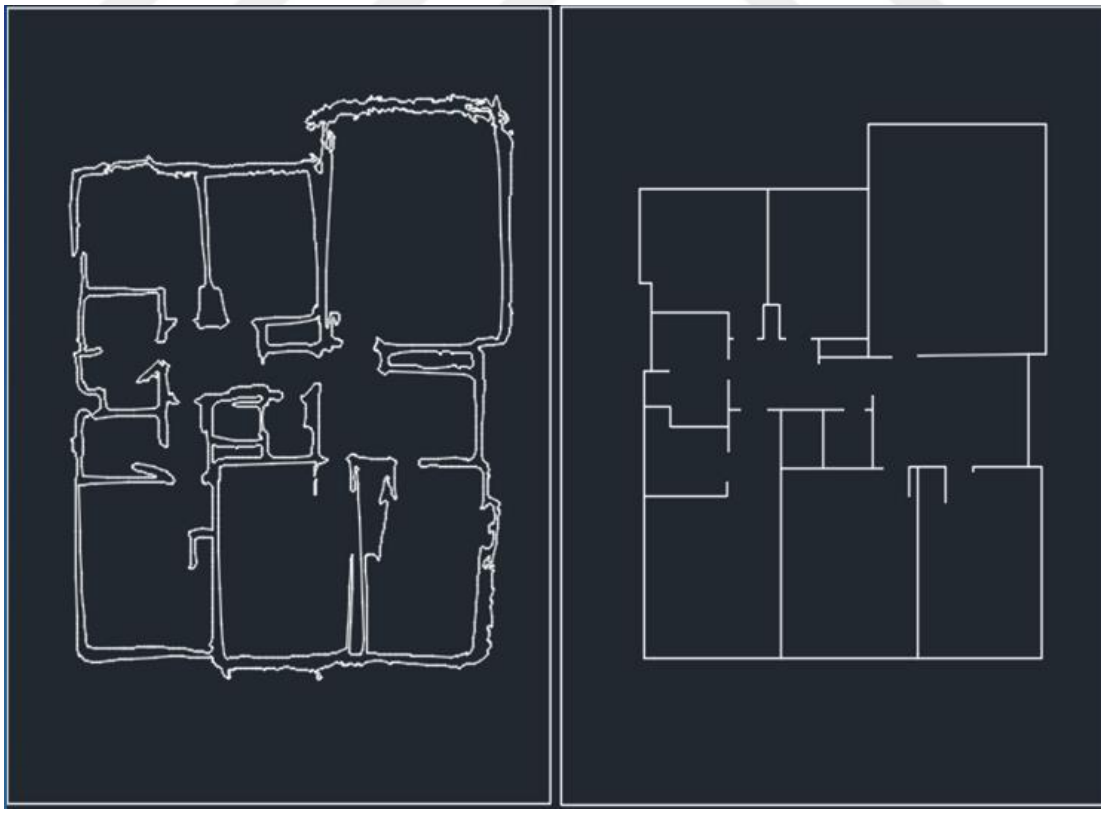
Şekil 4.16 HectorSLAM algoritması kullanılarak oluşturulan harita

Mekânın haritası oluşturulurken kullanılan HectorSLAM algoritması ışınların bitiş noktalarının hizalanmasının optimize edilmesi ile elde edilir. Teker odometri verisi kullanılmaz ancak LIDAR ışınlarının Iterative Closest Point (ICP) gibi yöntemler ile ışınların bitiş noktalarının eşleştirilmesi sonucunda harita elde edilir. HectorSLAM yöntemi ile haritalamanın dezavantajı cam gibi saydam bölmelerin de haritaya dahil edilmesidir. Şekil 4.16'da görüldüğü gibi odaların uç bölümlerinde bulunan cam pencereler de duvar olarak çizilmiş ve haritaya dahil edilmiştir. Ancak bu çalışmanın amacı iç mekânın ölçülerini almaktır. Bundan dolayı camların ayrıca bir sensör kullanılarak tespit ettirilmesine gerek kalmamıştır. Böylelikle maliyet ve işlem yükü minimum seviyede tutulmuştur.

4.2.2. Araştırma Ortamının Haritası ve Ölçeklendirilmesi Uygulaması

Geleneksel uygulamalarda mekân ölçeklendirilmesi için metre veya lazer metre kullanılmaktadır. Lazer metre, metreye oranla daha hızlı bir ölçüm gerçekleştirir. Ancak lazer metrenin de ölçüm esnasında yere paralel bir şekilde ölçüm alması ve doğru bir ölçüm yapılması gerekmektedir. Her iki yöntemde de harcanan süre miktarı fazladır. Ayrıca doğruluk oranı da ölçen kişinin hatalı ölçümüne göre değişmektedir. Alınan ölçümler sonrasında haritanın başka bir ara yüze alınması için tekrar çizilmesi gerekmektedir. Bu da ayrıca bir işlemdir ve proje çizme konusunda bir profesyonel gerektirir.

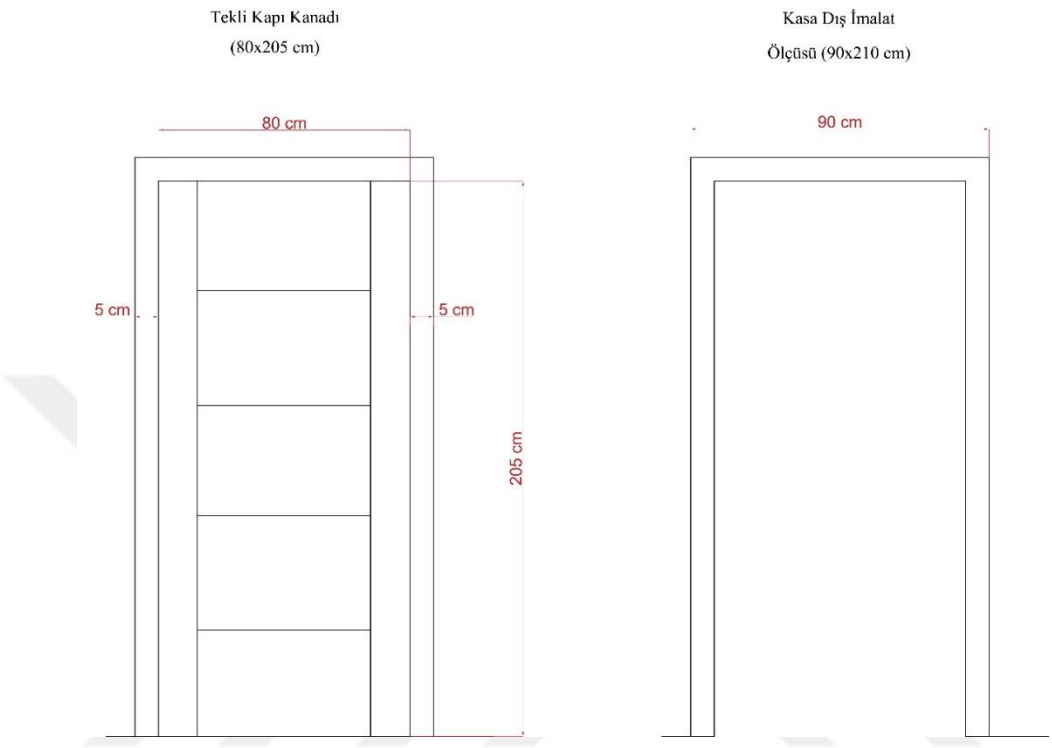
Bu tez kapsamında yapılan çalışmada geleneksel olarak iç mekanların ölçeklendirilmesi ve mimari projelendirilmesi konusunda bir yöntem önerilmiştir. Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM) algoritması kullanılarak ortam haritası kısa bir sürede oluşturulmuştur (bkz Şekil 4.16). Oluşan bu harita .yaml dosyası uzantısı şeklinden .dwg dosya uzantısına dönüştürülmüş ve Autocad programına aktarılmıştır (bkz Şekil 4.17). AutoCad programına aktarılan harita LIDAR taramasından dolayı oluşan girinti ve çıkıntılar temizlenerek düzenlenmiştir (Şekil 4.17b).



(a) (b)

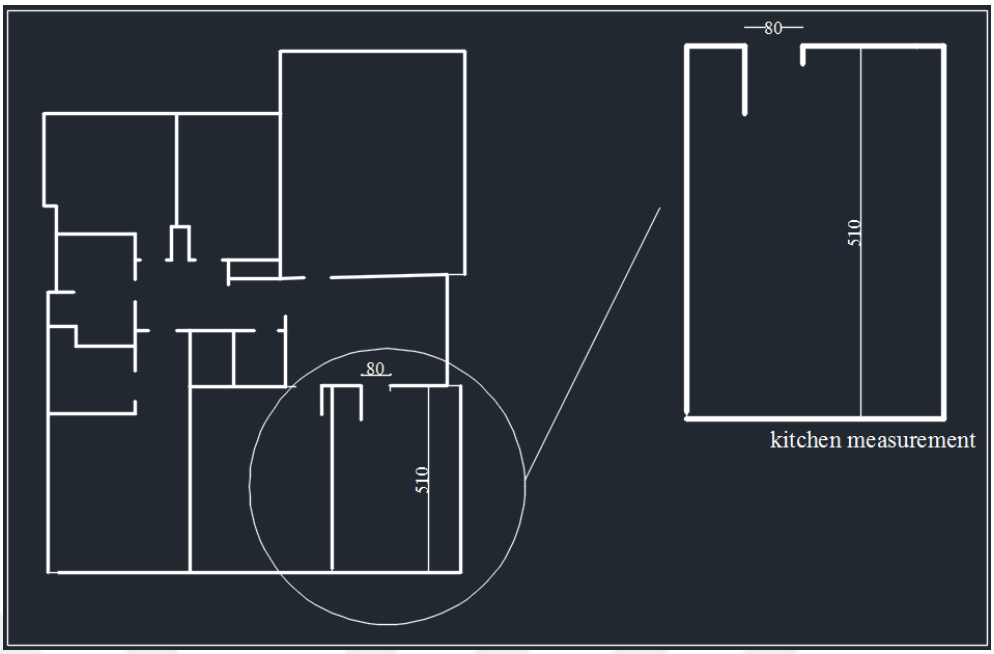
Şekil 4.17 AutoCad programına aktarılan harita. (a) aktarılan görüntü (b) temizlenen görüntü.

Şekil 4.18’de kapı boşluğu standart ölçüsü 80 cm olarak belirlenmiştir. Haritada görülen her bir kapı boşluğu mesafesi 80 cm baz alınarak, iç mekandaki bütün duvarların ölçüleri bulunmuştur.



Şekil 4.18 Tekli kapı genişlik ölçüsü

Belirlenen duvar uzunluğuna örnek olarak Şekil 4.18’de verilmiştir. Şekil 4.19’da kapı genişliği 80 cm referans alınmış ve mutfak uzun duvarı ölçüsü 510 cm olarak belirlenmiştir. Gerçek ölçüsü 490 cm olan mutfak duvar uzunluğunun %96,07 doğrulukla tahmini gerçekleştirilmiştir. Tüm mekânın 197m^2 olan gerçek alanı; önerilen yöntem ile 184,62m^2 olarak belirlenmiş ve %93,71 oranında bir doğrulukla tahmin edilmiştir.



Şekil 4.19 Mutfak duvar uzunluğu

Yapılan çalışma ile geleneksel yöntemlere alternatif bir metod önerilmiştir. Bu yöntemin en büyük avantajı sürenin kısa olmasıdır. Geleneksel yöntemlerle ortalama 40-50 dakikada ölçüm alınırken projenin çizilmesi ve ortama aktarılması da ayrı bir süre alacaktır. Ancak önerilen yöntem ile ortalama 8 dakika içerisinde ortam haritası SLAM algoritmaları sayesinde oluşturulup, çizim ortamına aktarılarak sonuçlar elde edilmektedir.

Tablo 4.3 Ölçeklendirme yöntemlerinin karşılaştırılması

	Metre	**Lazer Metre**	**Önerilen Yöntem**
Zaman	50-55 dakika	40-45 dakika	8-9 dakika
İç Mekân Haritası Oluşturulan Arayüz	Autocad	Autocad	RViz
Gerekli İnsan Gücü	3 Kişi	2 Kişi	1 Kişi
Doğruluk Oranı	95-98%	95-98%	%93,71

Tablo 4.3'te kullanılan geleneksel yöntemler ve önerilen yöntem için karşılaştırmalı bir analiz verilmiştir. Kümülatif olarak SLAM ile yapılan uygulamada hata oranı daha fazla olmuştur. Ancak çok daha kısa bir sürede ölçeklendirme tamamlanmış ve aynı zamanda kullanılan iş gücünden kazanım sağlanmıştır.

HectorSLAM yöntemi ile tek bir sensör kullanılarak ortam haritalandırılmıştır. Bu haritalandırmanın el veya bir İnsansız Hava Aracı (İHA) ile yapılabilir olması sistemin avantajıdır.

5. SONUÇLAR VE ÖNERİLER

5.1 Sonuçlar

Teknolojideki gelişmelerle beraber havacılık, uzay, savunma sistemleri entegrasyonları ve üretim sanayi gibi alanlarda robot kullanımı yaygınlaşmıştır. Bu gelişmeler ışığında robotik sistemlerin kontrol edilen ve komutla hareket eden sistemler olmasının haricinde, kendi kendine karar verebilen ve ortamda otonom hareket edebilen sistemler olması amaçlanmaktadır. Askeri uygulamalarda ve savunma sistemlerine yönelik çalışmalarda kapalı ortam konumlandırması önemlidir. Dış ortamda GPS verisi ile yüksek doğrulukla konum belirlenebilirken, kapalı ortamlarda GPS verileri alınamadığından dolayı konumlandırma için farklı yöntemler kullanılır. Robotun bilinmeyen bir ortamın haritasını oluştururken aynı zamanda bu ortamda kendisini konumlandırması Eş Zamanlı Konumlandırma ve Haritalandırma (SLAM) problemi olarak ele alınır. SLAM; robotun kapalı ortamda konumlandırılmasına yardımcı olurken aynı zamanda robota bir otonom hareket kabiliyeti de kazandırmış olur. Ancak robotun ortamda konumlandırılması çeşitli tahmin yöntemleri kullanılarak gerçekleştirildiğinden dolayı hatalı sensör verilerine bağlı olarak hatalı konum tahmini sonuçlar elde edilebilir. Teker verisini kullanan robot, tekerin yumuşak bir zeminde kayması, robotun engele takılması veya robot kaçırması gibi bir durumda odometri verisine bağlı olarak konumunu yanlış tahmin eder.

Bu tez çalışması kapsamında savunma teknolojilerinde önemli bir problem olan kapalı ortamlarda konumlandırma ve kapalı ortamların haritalandırılarak ölçeklendirilmesi konuları ele alınmıştır. İlk deneysel çalışmada kapalı ortamda robot konumunun kaybedilmesi sonucunda iki adet nesnenin konumuna bağlı olarak ortamda robotun konumlandırılması amaçlanmıştır. Deneysel ortam oluşturulup ölçeklendirilmiş ve iki nesne ortamda bilinen noktalara yerleştirilmiştir. Bir Derin Öğrenme modeli olan Faster R-CNN ile eğitilmiş 1798 veri kullanılarak ortamdaki iki nesnenin tanınması sağlanmıştır. Tanınan ve konumları bilinen iki nesneye göre önerilen matematiksel yöntem kullanılarak robotun konumlandırılması amaçlanmıştır. Deneysel sonuçlarda öncelikle robotun bir teker kaybı olmadığı durumu ele alınmış ve elde edilen sonuçlar klasik bir konum tahmin yöntemi olan Monte Carlo Localization (MCL) yöntemiyle karşılaştırılmıştır. Teker veri kaybı olmadığı takdirde MCL yönteminin önerilen yönteme göre daha başarılı bir sonuç verdiği görülmüştür. Ancak ikinci ve üçüncü deneylerde

robotun teker verileri kaybolmuş ve konum tahminleri tekrar yapılmıştır. Öncelikle birinci nesne hizasında 100 cm y ekseninde hareket eden robot MCL ile %60 doğru konum tahmini sağlarken önerilen yöntem ile %89,39 tahmin oranı elde etmiştir. İkinci nesne hizasında 35,5 cm y ekseninde hareket eden robot MCL ile %59,16 doğru konum tahmini sağlarken önerilen yöntem ile %88,11 tahmin oranı elde etmiştir. Sonuç olarak klasik yöntem olan MCL ile konum tahmini bir teker veri kaybı olmadığı takdirde çok başarılı sonuç verirken; teker veri kaybında önerilen yöntem MCL' ye oranla daha doğru konum tahmini gerçekleştirmiştir.

Tez kapsamında yapılan ikinci çalışmada ise kapalı ortamların haritalandırılması sonucunda kısa bir sürede ve daha az iş gücüyle ölçeklendirilmesi amaçlanmıştır. Bir SLAM yöntemi olan HectorSLAM algoritması kullanılmış ve teker verisine bağlı olmadan ortamın haritalandırılması LİDAR sensör kullanılarak yapılmıştır. Haritalandırılan ortam kapı boşlukları standart 80 cm referans alınarak bütün ölçeklendirme işlemi gerçekleştirilmiştir. Klasik yöntemlerde metre ve lazer metre kullanılarak 40-50 dakika arasında gerçekleştirilebilen ölçeklendirme problemi, önerilen bu yöntemle ortalama 8 dakikada gerçekleştirilmiştir. Ayrıca insan gücü ile maliyeti de azaltılmıştır. İnsansız olarak da gerçekleştirilebilen bu yöntem ile kapalı, bilinmeyen bir ortamın haritalandırılarak ölçeklendirilmesi amaçlanmıştır.

5.2 Öneriler

Tez kapsamında yapılan çalışmada savunma teknolojilerine yönelik ve savunma sistemlerine entegre edilebilir, otonom hareket eden bir robotun dış etkilere bağlı sensör veri kaybı sonucunda kapalı bir ortamda konumlandırılması ve kapalı bir ortamın haritalandırılarak ölçeklendirilmesi amaçlanmıştır. Deneysel çalışmada trafik işaretleri kullanılarak bir veri seti oluşturulmuş ve nesnelerin tanınması sağlanmıştır. Ancak nesne aralığı geniş tutularak daha büyük bir veri seti oluşturulup bir ortamda bulunan her bir nesnenin konumuna göre robotun konumlandırılması da sağlanabilir. Ayrıca LIDAR, Kamera ve teker odometri sensörü kullanılan bu çalışma da sadece kamera kullanılarak Visual SLAM yöntemleri ile maliyet de azaltılabilir.

İkinci çalışmada kapalı ortamların haritalandırılması ve ölçeklendirilmesi gerçekleştirilmiştir. Harita içerisinde kapı boşlukları standart uzunluğu referans alınarak ölçeklendirme işlemi yapılmıştır. Görüntü işleme yöntemleri kullanılarak bu ölçeklere bağlı bütün ortamın ölçeklendirilmesi daha doğru sonuçlarla elde edilebilir. Ayrıca drone kullanılarak tek bir LİDAR sensör aracılığıyla ortam haritalandırılması ve ölçeklendirilmesi insansız olarak gerçekleştirilebilir.

KAYNAKLAR

Andrade-Cetto, J. ve Sanfeliu, A., 2004, The Effects of Partial Observability in SLAM. Proceedings of the 2004 IEEE International Conference on Robotics Automation. New Orleans,LA.

Atalı, G., 2018, Dağıtık Mobil Robotlar için Yeni Bir Otonom Yol Planlama Ve Engel Tespit Sisteminin Tasarımı, *Sakarya Üniversitesi*.

Daum, F., 2005, Nonlinear filters: beyond the Kalman filter, *IEEE Aerospace and Electronic Systems Magazine*, 20, 57-69.

Deißler, T. ve Thielecke, J., 2013, Fusıng Odometry and Sparse UWB Radar Measurements for Indoor SLAM. 2013 Workshop on Sensor Data Fusion: Trends, Solutions, Applications (SDF). Bonn, Germany, IEEE.

Deng, L., 2014, Deep Learning: Methods and Applications, *Foundations and Trends® in Signal Processing*, 7 (3-4), 197-387.

Doucet, A., Freitas, N. d. ve Gordon, N., 2001, N. Sequential Monte Carlo Methods in Practice, *Springer*.

Durrant-Whyte, H. ve Bailey, T., 2006, Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms, *IEEE Robotics & Automation Magazine*, 13 (2).

Fox, D., Hightower, J., Liao, L., Schulz, D. ve Borriello, G., 2003, Bayesian filtering for location estimation, *IEEE pervasive computing*, 2, 24-33.

Fukushima, K., 1980, Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position, *Biological Cybernetics*, 36, 193-202.

Girshick, R., Donahue, J., Darrell, T. ve Malik, J., 2014, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. IEEE Conference on Computer Vision and Pattern Recognition**:** 580-587.

Girshick, R., 2015, Fast R-CNN, *ICCV '15 Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 1440-1448.

Grisetti, G., Stachniss, C. ve Burgard, W., 2005, Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling, *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, Spain.

Grisetti, G., Stachniss, C. ve Burgard, W., 2007, Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters, *IEEE Transactions on Robotics*, 23 (1), 34-46.

Hinton, G. E., 2007, Learning multiple layers of representation, *Trends Cogn Sci*, 11 (10), 428-434.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. ve Salakhutdinov, R. R., 2012, Improving Neural Networks by Preventing Co-adaptation of Feature Detectors, *Neural and Evolutionary Computing*.

Kamarudin, K., Mamduh, S. M., Shakaff, A. Y. M. ve Zakaria, A., 2014, Performance Analysis of the Microsoft Kinect Sensor for 2D Simultaneous Localization and Mapping (SLAM) Techniques, *Sensors 2014*, 23365-23387.

Khairuddin, A. R., Talib, M. S. ve Haron, H., 2015, Review on Simultaneous Localization and Mapping (SLAM). International Conference on Control System, Computing and Engineering. George Town, Malaysia, IEEE.

Kim, B., Choi, B., Kwang, E. K. ve Yang, W., 2012, Indoor Localization Using Laser Scanner and Vision Marker for Intelligent Robot. 12th International Conference on Control, Automation and Systems. JeJu Island, South Korea, IEEE.

Kohlbrecher, S., Stryk, O. v., Meyer, J. ve Klingauf, U., 2011, A Flexible and Scalable SLAM System with Full 3D Motion Estimation, *In Proceedings of the 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR 2011)*, Kyoto, Japan, 155-160.

Krizhevsky, A., Sutskever, I. ve Hinton, G. E., 2012, ImageNet Classification with Deep Convolutional Neural Networks *Proceedings of the 25th International Conference on Neural Information Processing Systems*, Lake Tahoe, Nevada, 1097-1105.

Kuleli, A., 2009, Mobil Robotlarda Parçacık Filtresi Kullanarak Eş Zamanlı Lokalizasyon Ve Haritalama, *İstanbul Teknik Üniversitesi*.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E. ve Jackel, L. D., 1990, Handwritten Digit Recognition with a Back-Propagation Network, In: Advances in Neural Information Processing Systems 2 (NIPS 1989), Eds, p. 396-403.

LeCun, Y., 2016, Deep Learning & Convolutional Networks. 2015 Ieee Hot Chips 27 Symposium (Hcs).

Leonard, J. J. ve Durrant-Whyte, H. F., 1991, Simultaneous Map Building and Localization for an Autonomous Mobile Robot. IEEE/RSJ International Workshop on Intelligent Robots and Systems. Osaka, Japan.

Marck, J. W., Mohamoud, A., Houwen, E. v. ve Heijster, R. v., 2013, Indoor Radar SLAM A radar application for Vision and GPS Denied Environments., *Proceedings of the 10th European Radar Conference*, Nuremberg, Germany.

Martinelli, A. ve Siegwart, R., 2007, Exploiting the Information at the Loop Closure in SLAM. Proceedings 2007 IEEE International Conference on Robotics and Automation**:** 2055-2060.

Montemerlo, M., Thrun, S., Koller, D. ve Wegbreit, B., 2002, FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem, *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Alberta, Canada, 593-598.

Montemerlo, M. ve Thrun, S., 2003, Simultaneous Localization and Mapping with Unknown Data Association Using FastSLAM, *2003 IEEE International Conference on Robotics and Automation*, Taipei, Taiwan.

Nazemzadeh, P., Fontanelli, D., Macii, D. ve Palopoli, L., 2017, Indoor Localization of Mobile Robots Through QR Code Detection and Dead Reckoning Data Fusion, *IEEE/ASME Transactions on Mechatronics*, 22 (6), 2588-2599.

Orderud, F., 2005, Comparison of kalman filter estimation approaches for state space models with nonlinear measurements. Proc. of Scandinavian Conference on Simulation and Modeling**:** 1-8.

Ren, S., He, K., Girshick, R. ve Sun, J., 2016, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Computer Vision and Pattern Recognition.

Santos, J. M., Couceiro, M. S., Portugal, D. ve Rocha, R. P., 2015, A Sensor Fusion Layer to Cope with Reduced Visibility in SLAM, *Journal of Intelligent & Robotic Systems*, 80 (3-4), 401-422.

Scaramuzza, D. ve Siegwart, R., 2008, Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles, *IEEE Transactions on Robotics*, 24 (5), 1015-1026.

Scaramuzza, D. ve Fraundorfer, F., 2011, Visual Odometry [Tutorial], *IEEE Robotics & Automation Magazine*, 18 (4), 80-92.

Smith, R., Self, M. ve Cheesemans, P., 1988, Estimating Uncertain Spatial Relationships in Robotics, In: Machine Intelligence and Pattern Recognition, Eds, p. 435-461.

Spielmann, R., Duchoň, F., Kostroš, J., Fico, T. ve Balog, R., 2013, Sensor Module for Mobile Robot, *American Journal of Mechanical Engineering*, 1, 378-383.

Şeker, A., Diri, B. ve Balık, H. H., 2017, Derin Öğrenme Yöntemleri ve Uygulamaları Hakkında Bir İnceleme, *Gazi Mühendislik Bilimleri Dergisi*, 3 (3), 47-64.

Thrun, S., Burgard, W. ve Fox, D., 1998, A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots, *Autonomous Robots*, 31 (5).

Thrun, S., Burgard, W. ve Fox, D., 2005, Probabilistic Robotics, MIT Press, p.

Thrun, S. ve Leonard, J. J., 2007, Simultaneous Localization and Mapping, In: Springer Berlin Heidelberg, Eds: Springer Berlin Heidelberg, p.

Weingarten, J. ve Siegwart, R., 2005, GKF-based 3D SLAM for Structured Environment Reconstruction. 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. Edmonton, Alta., Canada, IEEE.

Weiss, G., Wetzler, C. ve Puttkamer, E. v., 1994, Keeping Pack of Position and Orientation of Moving Indoor Systems by Correlation of Range-Finder Scans, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94)*, Munich, Germany.

Xiong, J., Liu, Y., Ye, X., Han, L., Qian, H. ve Xu, Y., 2016, A Hybrid Lidar-based Indoor Navigation System Enhanced by Ceiling Visual Codes for Mobile Robots, *Proceedings of the 2016 IEEE International Conference on Robotics and Biomimetics*, Qingdao, China.

Zhang, T., Chong, Z. J., Qin, B., Fu, J. G. M., Pendleton, S. ve Ang, M. H., 2014, Sensor Fusion for Localization, Mapping and Navigation in an Indoor Environment. 2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM). Palawan, Philippines, IEEE.

Zunino, G. ve Christensen, H. I., 2001, Simultaneous Localization and Mapping in Domestic Envoriments. Conference Documentation International Conference on Multisensor Fusion and Integration for Intelligent Systems. Baden-Baden, Germany.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Ahmet Murat ERTURAN
Uyruğu	**:**	T.C.
Doğum Yeri ve Tarihi	**:**	ERZURUM/ 31.07.1990
Telefon	**:**	0543 960 77 03
Faks	**:**	
E-Posta	**:**	a.muraterturan@gmail.com

EĞİTİM

Derece		**Adı ,İlçe,İl**	**Bitirme Yılı**
Lise	:	Erzurum Anadolu Lisesi, Yakutiye, ERZURUM	2011
Üniversite	:	Karadeniz Teknik Üniversitesi, Ortahisar, TRABZON	2016
Yüksek Lisans	:	Konya Teknik Üniversitesi, Selçuklu, KONYA	2019
Doktora	:		

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2017-2018	Erzurum Teknik Üniversitesi	Araştırma Görevlisi
2018-	Konya Teknik Üniversitesi	Araştırma Görevlisi

UZMANLIK ALANI

Savunma Teknolojileri ve Entegrasyonu, ROS, SLAM, Robotik

YABANCI DİLLER

İngilizce

YAYINLAR

1. Erturan, A.M., Durdu, A., Erturan, E.M., *The Use of LIDAR Technology in Arthitectural Offices*. European Journal of Engineering Science and Technology, 2019. 2(2): p. 40-48.