T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OTONOM MOBİL ROBOT İÇİN İÇ MEKANLARDA ROTA PLANLAMA VE HARİTA ÇIKARTMA

YÜKSEK LİSANS TEZİ

İhsan ÇUBUKÇU

Elektrik-Elektronik Mühendisliği Anabilim Dalı

EYLÜL 2023

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OTONOM MOBİL ROBOT İÇİN İÇ MEKANLARDA ROTA PLANLAMA VE HARİTA ÇIKARTMA

YÜKSEK LİSANS TEZİ

İhsan ÇUBUKÇU

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof.Dr. İrfan YAZICI

EYLÜL 2023

İhsan ÇUBUKÇU tarafından hazırlanan "Otonom Mobil Robot İçin İç Mekanlarda Rota Planlama Ve Harita Çıkartma" adlı tez çalışması 05.09.2023 tarihinde aşağıdaki jüri tarafından oy birliği/oy çokluğu ile Sakarya Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı'nda Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Jüri Başkanı :	**Doç.Dr.Gökçen ÇETİNEL** Sakarya Üniversitesi	**……..…….**
Jüri Üyesi :	**Prof.Dr.İrfan YAZICI** (Danışman) Sakarya Üniversitesi	**……..……**
Jüri Üyesi :	**Dr.öğr.üyesi Ali Furkan KAMANLI** Sakarya Uygulamalı Bilimler Üniversitesi	**……**

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Sakarya Üniversitesi Fen Bilimleri Enstitüsü Lisansüstü Eğitim-Öğretim Yönetmeliğine ve Yükseköğretim Kurumları Bilimsel Araştırma ve Yayın Etiği Yönergesine uygun olarak hazırlamış olduğum Otonom Mobil Robot İçin İç Mekanlarda Rota Planlama Ve Harita Çıkartma başlıklı tezin bana ait, özgün bir çalışma olduğunu; çalışmamın tüm aşamalarında yukarıda belirtilen yönetmelik ve yönergeye uygun davrandığımı, tezin içerdiği yenilik ve sonuçları başka bir yerden almadığımı, tezde kullandığım eserleri usulüne göre kaynak olarak gösterdiğimi, bu tezi başka bir bilim kuruluna akademik amaç ve unvan almak amacıyla vermediğimi ve 20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince Sakarya Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Enstitü tarafından belirlenmiş ölçütlere uygun rapor alındığını, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun ortaya çıkması halinde doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi beyan ederim.

(26/10/2023)

İhsan ÇUBUKÇU

TEŞEKKÜR

Yüksek lisans ders ve tez dönemi süresince fikir, tecrübe ve önerileri ile desteğini esirgemeyen danışmanım Prof. Dr. İrfan YAZICI'ya teşekkür ederim, sayın danışmanımın değerli görüşleri, araştırma çalışmamın şekillenmesine katkıda bulunarak beni bu noktaya taşımıştır. Kendisine minnettarlığımı ifade etmek isterim. Tez çalışması boyunca beni teşvik eden Leo Drive ailesine ve son olarak sadece çalışma süresince değil yaşamımın her anında, her koşulda desteklerini hissettiğim annem Rafif ÇUBUKÇU'ya, babam Macittin ÇUBUKÇU'ya bana olan inançlarından dolayı sonsuz teşekkürlerimi sunarım.

İhsan ÇUBUKÇU

İÇİNDEKİLER

Sayfa

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ ..... **v**
TEŞEKKÜR ..... **vii**
İÇİNDEKİLER ..... **ix**
KISALTMALAR ..... **xi**
SİMGELER ..... **xiii**
TABLO LİSTESİ ..... **xv**
ŞEKİL LİSTESİ ..... **xvii**
ÖZET ..... **xix**
SUMMARY ..... **xxiii**
1. GİRİŞ **1**
1.1. Tezin Kapsamı 3
1.2. Tezin Amacı 4
1.2.1. Tezin ikincil amacı 4
2. LİTERATÜR ÖZETİ **5**
3. MATERYAL VE YÖNTEM **9**
3.1. TurtleBot 9
3.2. TurlteBot'un Donanımı 10
3.3. Robot İşletim Sistemi 11
3.4. Gazebo Simülatör 13
3.5. Rviz 14
3.6. SLAM 15
3.7. LiDAR SLAM 16
3.8. SLAM'da Kullanılan Algoritmamalar 17
3.9. Dijkstra Algoritması 18
3.10. A-Star Algoritması 19
3.11. A* Algoritmasının Kullanımı 19
3.12. A* Algoritması İle Dijkstra Algoritması Arasındaki Fark 21
3.13. Navigasyon 22
3.13.1. Algılama (Perception) 23
3.13.2. Konum Belirleme (Localization) 23
3.13.3. Yol Planlama (Path Planning) 23
3.13.3.1. Move Base 24
3.13.3.2. Global Planlayıcı (Global Planner) 24
3.13.3.3. Lokal Planlayıcı (Local Planner) 24
3.13.4. Hareket Kontrolü (Motion Control) 25
4. Turtlbebot3 ile Otonom Sürüş **27**
4.1. TurtleBot3 İle Bağlanmak 27
4.2. Turtlbebot3'ün Başlatılması 28
4.3. Turtlbebot3 ile SLAM 29
4.4. Haritayı Kaydetmek 30
4.5. Turtlbebot3 ile Navigasyon 31

4.6. İlk Konum Tahmini 32
4.7. Hedef Belirleme..... 33
5. A* Algoritması ile Yol PLANLAMA- Simülasyon..... 35
6. SONUÇ VE ÖNERİLER..... 37
KAYNAKLAR..... 43
ÖZGEÇMİŞ..... 45

KISALTMALAR

AMCL : Adaptif Monte Carlo Lokalizasyonu (Adaptive Monte Carlo Localization)

A* : A-Star algoritması (A-Star Algorithm)

DWA : Dinamik Pencere Yaklaşımı (Dynamic Window Approach)

Host-PC : Ana bilgisayar

IMU : Inertial Measurement Unit - Hareket Sensörü Birimi

LİDAR : lazer tarayıcı (Light Detection and Ranging)

Remote-PC : Ana bilgisayar

ROS : Robot İşletim Sistemi (Robot Operating System)

SSH : Ağ Protokolü (Secure Shell)

SLAM : Eş Zamanlı Konum Belirleme ve Haritalama (Simultaneous Localization and Mapping)

TurtleBot3 : Bu çalışmada kullanılan robotun ismi

YAML : Biçimlendirme Dili Değildir (Yet Another Markup Language)

SİMGELER

F(n) : Toplam Maliyet

G(n) : Başlangıç Noktasından Bir N Düğümüne Giden Yolun Uzunluğu

H(n) : N düğümden hedef düğüme olan buluşsal (tahmini) mesafesidir

TABLO LİSTESİ

Sayfa

Tablo 6.1. (A) haritasında DAWA ve A*'nın performans karşılaştırması 37
Tablo 6.2. (B) haritasında DAWA ve A*'nın performans karşılaştırması 38

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1. Otonom teslimat robotları. 1
Şekil 1.2. Temizlik robotu. 2
Şekil 1.3. Depolama ve lojistik robotu. 2
Şekil 1.4. Üniversitelerde Eğitim ve Araştırma Robotları 3
Şekil 2.1. Otonom sürüş seviyeleri 7
Şekil 3.1. TurtleBot serisi 10
Şekil 3.2. Turtlebot3 Burger modeli 11
Şekil 3.3. ROS'ta iletişim. 13
Şekil 3.4. Gazebo programını kullanarak turtlebot3 burger modelini incelemek. 14
Şekil 3.5. Rviz ortamında 3D LiDAR verilerini görüntülemek 15
Şekil 3.6. LDS-02 LiDAR. 16
Şekil 3.7. 2D LiDAR SLAM. 16
Şekil 3.8. Radar- LiDAR entegreli EKF için çalışma prensibi. 18
Şekil 3.9. Bir kare ızgarada hedef belirlemesi. 19
Şekil 3.10. Başlangıç noktasından varış noktasına olası yollar. 20
Şekil 3.11. A* ile en kısa yolu buluması. 21
Şekil 3.12. Harita tabanlı navigasyon mimarisi. 22
Şekil 3.13. Mobil robot lokalizasyonu için genel şema. 23
Şekil 3.14. Move_Base Düğümünün yapısı. 24
Şekil 3.15. DWA Lokal Planlayıcı 25
Şekil 4.1. Ağ ayarları. 28
Şekil 4.2. Teleoperasyon ROS düğümüyle hareket komutları göndermek. 29
Şekil 4.3. rvizde SLAM algoritması ile alanları keşfetmek. 30
Şekil 4.4. SLAM ile oluşturulan Doluluk Izgara Haritası (Occupancy Grid Map). 31
Şekil 4.5. Navigasyon Süreci – İlk konum tahmini yapılmadı. 32
Şekil 4.6. İlk konum tahmin süreci. 33
Şekil 4.7. İlk konum tahmini yapıldı. 33
Şekil 4.8. Rviz arayüzü üzerinde hedef belirleme botunu. 34
Şekil 4.9. Yol planlaması ve hedefe ulaşma. 34
Şekil 5.1. A* ile uygun yolu bulmak. 36
Şekil 5.2. TurtleBot3'ün hedefe ulaşması. 36
Şekil 6.1. Map (B) durumunda feedback konusu – A* 38
Şekil 6.2. (A) haritasında navigasyon – sol: DWA, sağ: A*. 39
Şekil 6.3. (B) haritasında navigasyon – sol: DWA, sağ: A* 39

OTONOM MOBİL ROBOT İÇİN İÇ MEKANLARDA ROTA PLANLAMA VE HARİTA ÇIKARTMA

ÖZET

Bu tez, TurtleBot3 robotu üzerinde gerçekleştirilen SLAM (Simultaneous Localization and Mapping) ve navigasyon çalışmalarını kapsamaktadır. Tezin amacı, robotun otonom sürüş yeteneklerini geliştirmek ve karmaşık bir ortamda etkin bir şekilde hareket etmesini sağlamaktır.

TurtleBot3, kompakt ve hareketli bir robot platformudur. Bu platform, ROS (Robot Operating System) çatısı altında çalışan bir dizi sensör ve hareket kontrol birimiyle donatılmıştır. Bu sensörler arasında LIDAR (Light Detection and Ranging) sensörü, IMU (Inertial Measurement Unit) ve odometri bulunur. Bu sensörler, robotun çevresini algılamasına, konumunu takip etmesine ve hareket etmesine yardımcı olur.

Otonom sürüş, TurtleBot3 robotunun insan müdahalesi olmadan kendini yönlendirebilme yeteneğini ifade eder. SLAM ve navigasyon yetenekleri sayesinde TurtleBot3, otonom sürüşü gerçekleştirebilir. Robot, harita ve konum bilgilerini kullanarak engelleri tespit eder, çevresel koşullara uyum sağlar ve hedefe doğru güvenli bir şekilde ilerler. A* algoritması gibi yol planlama algoritmaları, TurtleBot3'ün güvenli ve hızlı bir şekilde hedefe ulaşmasını sağlayarak otonom sürüşün başarılı bir şekilde gerçekleştirilmesine yardımcı olur.

SLAM, robotun çevresini haritalandırmasını ve aynı anda konumunu tespit etmesini sağlayan bir tekniktir. Bu süreçte, LIDAR sensöründen alınan veriler kullanılarak harita oluşturulur ve robotun konumu tahmin edilir. Bu sayede robot, çevredeki engelleri algılayabilir ve yolunu planlayabilir.

Navigasyon, robotun belirlenen hedefe güvenli ve etkili bir şekilde hareket etmesini sağlayan süreçtir. Robot, harita ve konum bilgilerini kullanarak en uygun yol planını belirler ve bu yol üzerinde ilerler.

Yol planlama, bir robotun veya aracın belirli bir başlangıç noktasından hedef noktasına ulaşmak için takip etmesi gereken bir yolun belirlenmesi sürecidir. Yol planlama, robotların veya araçların engelleri aşarak güvenli ve etkili bir şekilde hedefe ulaşmasını sağlamak için kullanılan bir dizi algoritma ve stratejiden oluşur.

Yol planlama algoritmaları, verilen ortamda en kısa veya en uygun yolun bulunmasına yardımcı olur. A* gibi algoritması, genellikle yol planlamada kullanılan popüler yöntemdir.

A* algoritması, tahmini maliyet fonksiyonunu kullanarak en kısa yolun yanı sıra hedefe olan uzaklığı da dikkate alır. Bu sayede, TurtleBot3 robotu daha verimli bir şekilde hareket edebilir ve hedefe daha hızlı ulaşabilir. Yol planlama algoritmaları, TurtleBot3 robotunun verilen ortamda en uygun yolu bulmasını sağlar. Bu algoritmalar, harita ve konum bilgileriyle birlikte engelleri, dar alanları ve diğer zorlukları dikkate alarak en güvenli ve hızlı yolun belirlenmesine yardımcı olur.

Ayrıca, TurtleBot3 robotunun hareket yeteneklerini optimize etmek için dinamik engelleri de göz önünde bulunduran algoritmalara odaklanılmıştır. Bu sayede, robot karmaşık ortamlarda etkin bir şekilde hareket edebilir ve engelleri aşabilir.

Bu tezde, yol planlamada Dinamik Pencere Yaklaşımı (DWA) ve A* gibi algoritmalar kullanılacaktır. DWA otonom bir robotun çevresel koşulları ve hız kontrolünü dikkate alarak anlık olarak güvenli ve etkili bir hareket planı oluşturmasına yardımcı olan bir yol planlama ve kontrol yöntemidir. A* algoritması ise Dijkstra algoritmasının geliştirilmiş bir versiyonudur ve daha hızlı sonuçlar üretebilir.

Yol planlama algoritmaları, TurtleBot3 robotunun verilen ortamda en uygun yolu bulmasını sağlar. Bu algoritmalar, harita ve konum bilgileriyle birlikte engelleri, dar alanları ve diğer zorlukları dikkate alarak en güvenli ve hızlı yolun belirlenmesine yardımcı olur. Ayrıca, TurtleBot3 robotunun hareket yeteneklerini optimize etmek için dinamik engelleri de göz önünde bulunduran algoritmalara odaklanılmıştır. Bu sayede, robot karmaşık ortamlarda etkin bir şekilde hareket edebilir ve engelleri aşabilir.

Bu tez kapsamında, TurtleBot3 robotunun SLAM, navigasyon ve otonom sürüş yeteneklerini geliştirmek amacıyla çeşitli deneysel çalışmalar gerçekleştirilmiştir. Bu çalışmalarda, Gazebo simülasyon ortamı kullanılarak farklı senaryolar test edilmiş ve A* algoritması da bu simülasyon ortamında değerlendirilmiştir.

TurtleBot3'ün ROS ile entegrasyonu, SLAM ve navigasyon yeteneklerinin geliştirilmesi açısından önemlidir. ROS, robotun sensör verilerini almasına, harita oluşturmasına, konumunu takip etmesine ve hareketini kontrol etmesine yardımcı olur. Ayrıca, ROS üzerinde çalışan diğer paketler ve araçlar, TurtleBot3'ün farklı görevleri yerine getirmesini ve etkileşimli bir şekilde çalışmasını sağlar.

Bu tez kapsamında, TurtleBot3 robotunun SLAM, navigasyon ve otonom sürüş yeteneklerinin ROS'un sağladığı araçlar ve paketlerle geliştirilmesi hedeflenmiştir. ROS'un esnek ve genişletilebilir yapısı, TurtleBot3'ün karmaşık ortamlarda etkin bir şekilde hareket etmesini ve görevleri yerine getirmesini sağlayacak olanaklar sunar.

Gazebo, robotik simülasyonlar için etkili bir platform sağlamaktadır. Bu platform sayesinde, gerçek dünya ortamlarının sanal olarak oluşturulması mümkün olur ve robot davranışları gerçekçi bir şekilde simüle edilebilir. Bu çalışmada, Gazebo kullanılarak TurtleBot3 robotunun SLAM, navigasyon ve otonom sürüş yetenekleri farklı senaryolarda test edilmiştir.

Farklı senaryolar, çeşitli engel konfigürasyonları, değişen hedef noktaları ve zorlu koşulları içermektedir. Bu senaryolar, robotun harita oluşturma, konum tahmini, engelleri algılama ve güvenli bir şekilde hedefe yönelme yeteneklerini değerlendirmek amacıyla tasarlanmıştır. Gazebo simülasyonu, bu senaryoların gerçekçi bir şekilde oluşturulmasına ve test edilmesine olanak sağlamıştır.

Gazebo simülasyon ortamında gerçekleştirilen farklı senaryolar ve A* algoritması testleri, TurtleBot3 robotunun SLAM, navigasyon ve otonom sürüş yeteneklerinin geliştirilmesi açısından önemli bir değerlendirme sağlamıştır. Bu testler, robotun performansını değerlendirmek, algoritmaları iyileştirmek ve sistem tasarımını optimize etmek için kritik bir rol oynamıştır.

Robot, farklı ortamlarda test edilmiş ve başarılı sonuçlar elde edilmiştir. Ancak bazı zorluklar da tespit edilmiştir. Özellikle dar alanlarda geçiş sırasında LIDAR sensörünün düşük hassasiyeti nedeniyle robotun verimliliği azalmıştır. Bu sorunların

çözülmesi için daha gelişmiş sensörlerin kullanılması ve algoritmaların iyileştirilmesi gerekmektedir.

Sonuç olarak, bu tez TurtleBot3 robotu üzerinde SLAM, navigasyon ve otonom sürüşün gerçekleştirilmesi konusunda önemli bir çalışmadır. Elde edilen bulgular, robotik sistemlerin karmaşık görevleri yerine getirebilme potansiyelini göstermektedir. Gelecekteki çalışmalar, robotun performansını daha da artırmak ve daha geniş bir kullanım alanına sahip olmasını sağlamak için bu alanda daha fazla araştırma ve geliştirme yapmayı hedeflemektedir.

INDOOR ROUTE PLANNING AND MAPPING FOR AUTONOMOUS MOBILE ROBOT

SUMMARY

This thesis covers SLAM (Simultaneous Localization and Mapping) and navigation studies on TurtleBot3 robot. The aim of the thesis is to improve the autonomous driving capabilities of the robot and enable it to move effectively in a complex environment.

TurtleBot3 is a compact and mobile robot platform. This platform is equipped with a set of sensors and motion controllers operating under the umbrella of ROS (Robot Operating System). These sensors include the LIDAR (Light Detection and Ranging) sensor, IMU (Inertial Measurement Unit) and odometry. These sensors help the robot sense its surroundings, track its position, and move.

Autonomous driving refers to the TurtleBot3 robot's ability to steer itself without human intervention. Thanks to its SLAM and navigation capabilities, TurtleBot3 can realize autonomous driving. Using map and location information, the robot detects obstacles, adapts to environmental conditions and safely moves towards the target. Path planning algorithms such as the A* algorithm allow TurtleBot3 to reach its destination safely and quickly, helping to achieve successful autonomous driving.

SLAM is a technique that allows the robot to map its environment and simultaneously detect its position. In this process, a map is created using data from the LIDAR sensor and the robot's position is estimated. In this way, the robot can detect obstacles in the environment and plan its path.

Navigation is the process that enables the robot to move safely and effectively to the designated destination. The robot determines the most suitable road plan using the map and location information and proceeds on this road.

Path planning is the process of determining a path that a robot or vehicle must follow to reach its destination from a given starting point. Path planning consists of a set of algorithms and strategies used to ensure that robots or vehicles overcome obstacles and reach their destination safely and effectively.

Path planning algorithms help to find the shortest or most suitable path in a given environment. Algorithm like A* is the popular method often used in path planning.

The A* algorithm considers the distance to the destination as well as the shortest path using the estimated cost function. In this way, the TurtleBot3 robot can move more efficiently and reach the target faster. Path planning algorithms enable the TurtleBot3 robot to find the optimal path in the given environment. Together with map and location information, these algorithms help determine the safest and fastest route, taking into account obstacles, tight spaces and other challenges. In addition, the focus is on algorithms that take into account dynamic obstacles to optimize the motion capabilities of the TurtleBot3 robot. In this way, the robot can effectively move around complex environments and overcome obstacles.

In this thesis, algorithms such as DWA (Dynamic Window Approach) and A* are used in road planning. DWA is a path planning and control method that helps an autonomous robot instantly create a safe and effective action plan, taking into account environmental conditions and speed control. The A* algorithm, on the other hand, is an improved version of the Dijkstra algorithm and can produce faster results.

Path planning algorithms enable the TurtleBot3 robot to find the optimal path in the given environment. Together with map and location information, these algorithms help determine the safest and fastest route, taking into account obstacles, tight spaces and other challenges. In addition, the focus is on algorithms that take into account dynamic obstacles to optimize the motion capabilities of the TurtleBot3 robot. In this way, the robot can effectively move around complex environments and overcome obstacles.

Within the scope of this thesis, various experimental studies were carried out to improve the SLAM, navigation and autonomous driving capabilities of the TurtleBot3 robot. In these studies, different scenarios were tested using the Gazebo simulation environment and the A* algorithm was also evaluated in this simulation environment.

TurtleBot3's integration with ROS is critical to improving SLAM and navigation capabilities. ROS helps the robot acquire sensor data, create maps, track its position and control its movement. Also, other packages and tools running on ROS allow TurtleBot3 to perform different tasks and work interactively.

Within the scope of this thesis, it is aimed to develop the SLAM, navigation and autonomous driving capabilities of the TurtleBot3 robot with the tools and packages provided by ROS. The flexible and extensible nature of ROS provides the possibilities to enable TurtleBot3 to effectively navigate and perform tasks in complex environments.

Gazebo provides an effective platform for robotic simulations. Thanks to this platform, it is possible to create real world environments virtually and robot behavior can be simulated realistically. In this study, SLAM, navigation and autonomous driving capabilities of TurtleBot3 robot were tested in different scenarios using Gazebo.

Different scenarios include various obstacle configurations, varying target points and challenging conditions. These scenarios are designed to evaluate the robot's ability to map, estimate location, detect obstacles, and navigate safely. Gazebo simulation allowed these scenarios to be created and tested realistically.

Different scenarios and A* algorithm tests performed in the Gazebo simulation environment provided an important evaluation in terms of improving the SLAM, navigation and autonomous driving capabilities of the TurtleBot3 robot. These tests played a critical role in evaluating the robot's performance, improving algorithms and optimizing system design.

The robot has been tested in different environments and successful results have been obtained. However, some difficulties were also identified. The efficiency of the robot has decreased due to the low sensitivity of the LIDAR sensor, especially when passing in tight spaces. To solve these problems, it is necessary to use more advanced sensors and improve algorithms.

In conclusion, this thesis is an important study on the realization of SLAM, navigation and autonomous driving on the TurtleBot3 robot. The findings show the potential of robotic systems to perform complex tasks. Future work is aimed at further research

and development in this area to further improve the performance of the robot and enable it to have a wider range of uses.

1. GİRİŞ

Robotik alanında son yıllarda önemli gelişmeler kaydedilmektedir ve bu gelişmeler, robotların günlük yaşamımızda daha fazla yer almaya göstermeye başlamasına yol açmaktadır. Günümüzde özellikle mobil robotlar çok çeşitli alanlarda kullanılmaktadır. İç mekânlarda mobil robotlar gelişen teknolojiyle daha da yaygın hale gelmiş ve daha fazla sektörde kullanılmaya başlanmıştır. İhtiyaçlara ve uygulama şekline göre, mobil robotlar farklı şekillerde tasarlanabilir ve programlanabilir. İç mekânda veya kapalı ortamlarda mobil robotlar, birçok farklı amaçla kullanılan otomatik cihazlardır. Bu robotlar, birçok sektörde ve uygulamada çeşitli görevleri yerine getirmek için kullanılmaya başlanmıştır. Birkaç örnek verecek olursak:

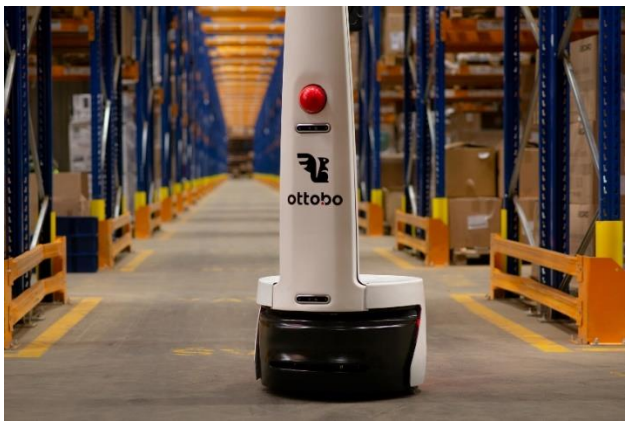
- Otonom Teslimat Robotları: Delivery Robots olarak adlandırılan " Teslimat robotları " paketleri veya ürünleri belirli bir noktadan başka bir noktaya taşıyan robotlardır. Bu robotlar, teslimat görevlerini otomatik veya özerk bir şekilde gerçekleştiren robot sistemleridir, bu robotlar, gelişmiş navigasyon ve algılama sistemleri kullanarak güvenli ve verimli teslimatlar yapabilir, bu robotlar ülkemizde yaygın şekilde kullanılmaya başlandı. Otonom teslimat robotları, perakende sektörü ve lojistik alanında hızlı ve etkili teslimat süreçleri sağlamak için potansiyel sunmaktadır. (Şekil 1.1)

Şekil 1.1. Otonom teslimat robotları.

- Temizlik robotları, iç mekanlarda zeminleri süpürmek ve silmek için kullanılan robotlardır. (Şekil 1.2)

Şekil 1.2. Temizlik robotu.

- Depolama ve Lojistik Robotları: Depolama ve lojistik süreçlerinde kullanılan robotlar, malzemelerin taşınması ve düzenlenmesi gibi görevler yapar. Raflar arasında hareket edebilir, malzemeleri toplayıp belirli bir noktaya taşıyabilir. (Şekil 1.3)



Şekil 1.3. Depolama ve lojistik robotu.

- Hastane Robotları: Hastanelerde kullanılan robotlar, sterilize etmek, ilaç taşıma veya yatakların hareket ettirilmesi gibi görevler yaparlar.
- Servis Robotları: Restoranlarda siparişlerin teslim edilmesi veya müşterilere yardım sağlamak için kullanılan robotlardır.
- Üniversitelerde Eğitim ve Araştırma Robotları: Üniversitelerin iç mekanlarında eğitim ve araştırmalar için kullanılan robotlar, aynı zamanda öğrencilerin kodlama becerilerini geliştirilmeleri ve algoritma geliştirilmesi amaçlarıyla da kullanılmaktadır. Örnek olarak TurtleBot, ROSbot, kobuki, Husky robot, PR2 robot şeklinde sıralanabilir. (Şekil 1.4)

Şekil 1.4. Üniversitelerde Eğitim ve Araştırma Robotları.

Otonom sürüşün tarihi oldukça eskiye dayanmaktadır. İnsanlar, uzun bir süredir, araçların kendi kendine hareket edebilmesi fikri üzerinde çalışmışlardır. Ancak, otonom sürüş teknolojisinin modern şekli, son yıllarda hızla gelişen bilgisayar ve sensör teknolojileri sayesinde mümkün hale gelmiştir. Günümüzdeki otonom sürüş, mobil robotların çevrelerini algılamaları, çevredeki engelleri tespit etmeleri ve bu engelleri aşarak güvenli bir şekilde hareket etmelerini sağlar.

Mobil robotlarda otonom sürüş için kullanılan teknolojilerden biri "Path planning" veya yol planlamadır. Yol planlama, mobil robotlar, otonom araçlar, insansız hava araçları ve diğer otonom sistemler için önemli bir bileşendir. Temel amaç, bir aracın hedefe ulaşırken çevredeki engelleri ve diğer araçları etkili bir şekilde önlemektir. Aynı zamanda, güvenlik, verimlilik, hız, enerji tasarrufu ve yolculuk süresi gibi faktörleri de göz önünde bulundurur. etkili bir şekilde seyahat etmesini sağlayan bir süreçtir. Yol planlama algoritmaları genellikle matematiksel ve hesaplama tabanlı yaklaşımlar kullanır. Bunlar arasında en yaygın olanları graf tabanlı arama algoritmaları ve optimizasyon algoritmalarıdır.

1.1. Tezin Kapsamı

Bu çalışmada kapsamında açık kaynak yazılıma sahip olan Turtlebot3 Burger modeli kullanılacaktır. TurtleBot3, çeşitli amaçlarla kullanılan bir açık kaynaklı mobil robot

ve yol planlaması için kullanabileceğimiz bir platformudur. Çalışmalar hem gerçek zamanlı hem simulasyon ortamında yürütülecektir.

1.2. Tezin Amacı

Bu tezde, iç mekanlarda otonom sürüş için yol planlaması üzerine odaklanmaktadır. Yol planlaması, bir mobil robotun belirlenen hedeflere güvenli, verimli ve etkili bir şekilde hareket etmesini sağlamak için temel bir bileşendir. Ancak iç mekanlarda otonom sürüşte karşılaşılan zorluklar, yol planlamasını daha karmaşık hale getirmektedir. Engeller, dar koridorlar, değişken ortam koşulları ve güncel olmayan haritalar gibi faktörler, doğru yol planlaması ve güvenli seyahat için özel çözümler gerektirmektedir. Yol planlamasında A* (A-Star) algoritması kullanılacaktır. Bu algoritmanın kullanılması, iç mekanlarda otonom sürüşte rota planlamasını daha verimli hale getirmek için bir potansiyel sağlamaktadır.

TurtleBot3 robotunun kullanılması, iç mekanlarda otonom sürüşün nasıl gerçekleştirilebileceğini, yol planlamasını incelemeyi hedeflemektedir. Bu amaç doğrultusunda, yol planlama algoritmaları ve teknikleri incelenecek ve iç mekanlarda kullanılabilirliği değerlendirilecektir.

A* algoritmasının performansı ve etkinliği, farklı iç mekân senaryolarında test edilecek ve sonuçlar analiz edilecektir. Elde edilen bulgular, otonom teslimat robotlarının güvenli ve verimli bir şekilde hareket etmelerini sağlamak için yol planlamasında A* algoritmasının önemini vurgulayacaktır.

1.2.1. Tezin ikincil amacı

Bu tez, iç mekanlarda otonom sürüş için yol planlamasında A* algoritmasını kullanarak, Turltebot3 robotun hedef noktalara en kısa ve optimize edilmiş yollarla ulaşmasını amaçlamaktadır.

Bu tezin ikincil amacı, TurtleBot3 platformu üzerinde Eş Zamanlı Konum Belirleme ve Haritalama (SLAM), navigasyon ve otonom sürüş kabiliyetlerini incelemek ve değerlendirmektir. Tez, TurtleBot3'ün SLAM, navigasyon ve otonom sürüş performansını analiz etmeyi amaçlamaktadır. Bu çalışma, mevcut yöntemlerin avantajlarını ve dezavantajlarını belirlemek ve önerilen çözümlerin uygulanabilirliğini araştırmak üzerine odaklanmaktadır.

2. LİTERATÜR ÖZETİ

Yol planlama, robotik sistemlerin veya otonom araçların belirli bir başlangıç noktasından hedefe güvenli ve etkili bir şekilde hareket etmelerini sağlamak için kullanılan bir süreçtir. Bu süreç, birçok farklı alanda kullanılmaktadır, örneğin endüstriyel otomasyon, insansız hava araçları, deniz araştırmaları, mobil robotlar ve otonom araçlar gibi. Literatürde yol planlamaya yönelik birçok farklı yaklaşım ve algoritma bulunmaktadır.

Bazı yaygın kullanılan yöntemler arasında A* (A-Star) algoritması, Dijkstra algoritması ve DWA (Dynamic Window Approach) gibi algoritmalar yer almaktadır. Her bir algoritmanın kendi avantajları, dezavantajları ve uygulama alanları bulunmaktadır.

Mobil robotlarda otonom sürüş, robotun çevresini algılayarak ve karar vererek belirli bir görevi yerine getirebilmesini ifade eder. Otonom sürüş, robotun insan müdahalesi olmadan belirlenen hedeflere doğru hareket edebilmesini sağlar.

Doğru bir yol planlama algoritması seçimi, çevresel engelleri önlemek, engelleri aşmak, en kısa veya en optimize yolu belirlemek gibi kritik faktörleri içermektedir. Ayrıca, yol planlama, enerji verimliliği, zaman tasarrufu ve etkili kaynak kullanımı gibi faktörleri de dikkate almaktadır.

Otonom sürüş, yol planlama sürecinin bir alt kategorisidir ve robotun veya otonom aracın insan müdahalesi olmadan hareket etmesini sağlar. Otonom sürüş, çeşitli algılama sensörleri ve yapay zeka tekniklerinin entegrasyonuyla gerçekleştirilir. Otonom sürüş, güvenlik, verimlilik ve kullanıcı deneyimi açısından büyük önem taşımaktadır.

Literatürde yol planlama ve otonom sürüş alanında yapılan çalışmalar, farklı yaklaşımların ve algoritmaların performansını değerlendirmek, yeni yöntemlerin geliştirilmesi ve mevcut yöntemlerin iyileştirilmesi gibi konuları içermektedir. Bu literatür özeti, yol planlama ve otonom sürüş konularında genel bir bakış sunmakta ve araştırmacılara bu alanlardaki güncel çalışmalar hakkında bilgi sağlamaktadır. Yol

planlamadaki gelişmeler, otonom sistemlerin güvenli, etkili ve akıllı bir şekilde hareket etmesini sağlayarak birçok uygulama alanında büyük potansiyel sunmaktadır. Otonom sürüş, geleceğin mobilite teknolojilerinde önemli bir rol oynamaktadır ve bu alanda yapılan çalışmaların ilerlemesi büyük bir öneme sahiptir.

Otonom sürüş için çeşitli sensörler kullanılmaktadır. Bu sensörler arasında kamera, LiDAR, radar, ultrasonik sensörler gibi çevresel bilgileri toplamak için kullanılan algılayıcılar bulunmaktadır. Bu çevresel bilgilerle robotun çevresini analiz eder, engelleri algılar, haritalama yapar ve konum bilgisini güncellemektedir. Genel olarak, otonom sürüş sistemlerinin mimarisi, algılama sistemi (perception system) ve karar verme sistemi şeklinde organize edilmiştir [9].

Algılama görevi (perception task), engel algılama, konumlandırma, haritalama ve karar verme görevini içerir; yol planlama (path planning), davranış seçimi (behavior selection) ve kontrolü dikkate almaktadır [9]. Bu görevlerin yerine getirebilmek için radar, LiDAR; (Light Detection and Ranging), kamera, ultrasonik sensörler gibi kullanılabilir.

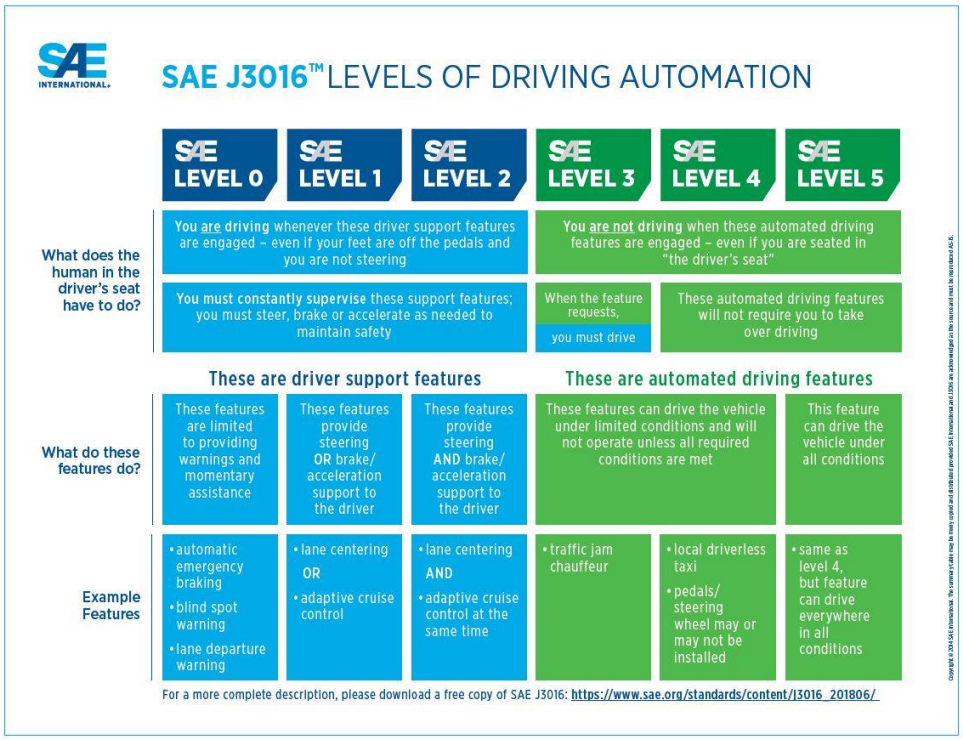
Society of Automotive Engineers (SAE), otonom sürüş arabaları bağlamında " Levels of Automated Driving" olarak bilinen SAE J3016 standardını tanımlamıştır. Bu standart, otonom sürüşün farklı seviyelerini belirlemek ve tanımlamak için bir çerçeve oluşturmayı amaçlamaktadır [10].

SAE'nin yaptığı çalışma, otonom sürüş sistemlerinin farklı seviyelerini tanımlayarak, endüstri, araştırmacılar ve düzenleyiciler arasında bir standart oluşturmayı hedefler. Bu sayede otonom sürüş teknolojilerinin gelişimi ve uygulanması daha tutarlı bir şekilde ilerleyebilir.

SAE J3016, 2014 yılında yayınlanan ve otonom sürüşün altı farklı seviyesini tanımlayan bir belgedir [10]. Otonom sürüş seviyeleri şekil 2.1.'de gösterilmektedir. Seviyeler aşağıdaki gibi sınıflandırılır:

1. Seviye 0: Otonom sürüş yetenekleri bulunmamaktadır. Tüm kontrol ve süreçler insan tarafından gerçekleştirilir.
2. Seviye 1: Sürücü destek sistemleri kullanılır. Belirli görevleri yerine getiren otomasyon özellikleri bulunabilir, fakat sürücü hala ana kontrolü elinde tutar. Örneğin hız kontrolü veya şerit takip etmek.

3. Seviye 2: Kısmi otonomi sağlanır. Sürücü hala aracı kontrol eder, ancak belirli sürüş görevlerini otomasyon sistemi gerçekleştirir. Sürücü, sistemin uygun olduğu durumlarda görevi devretme yeteneğine sahiptir.
4. Seviye 3: Koşullu otonomi sağlanır. Otonom sürüş sistemleri belirli koşullarda aracın tam kontrolünü devralabilir, ancak sürücü gerektiğinde sistemi kontrol etmek ve müdahale etmek zorundadır.
5. Seviye 4: Yüksek otonomi sağlanır. Otonom sürüş sistemleri, belirli koşullarda aracı tamamen kontrol edebilir ve sürücü müdahalesine ihtiyaç duymaz. Ancak, bazı sınırlamalar ve özel koşullar hala geçerli olabilir.
6. Seviye 5: Tam otonomi sağlanır. Aracın tüm sürüş görevlerini otonom sürüş sistemleri gerçekleştirir ve sürücü müdahalesine ihtiyaç duyulmaz. Bu seviyede sürücü olmayabilir ve araç herhangi bir yol veya hava koşulunda tamamen otonom olarak hareket edebilir.



Şekil 2.1. Otonom sürüş seviyeleri.

Mobil robotların otonom sürüş görevini etkili bir şekilde yapabilmek adına robotun çevresini algılayabilmesi, konumunu belirleyebilmesi, haritalama yapabilmesi, karar verebilmesi ve hareket edebilmesi gibi görevleri yerine getirmesi gereklidir.

3. MATERYAL VE YÖNTEM

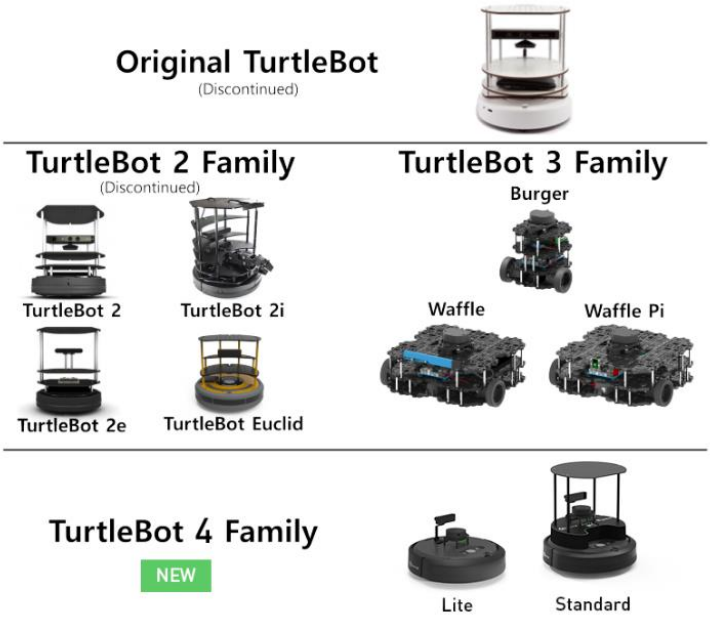
3.1. TurtleBot

TurtleBot, Open Robotics tarafından geliştirilen bir açık kaynaklı robot platformudur. Robotik yazılım geliştirme, algılama, haritalama ve navigasyon gibi alanlarda deneyim kazanmak için uygun bir platformdur. TurtleBot, taşınabilir, küçük boyutlu ve hafif bir tasarıma sahiptir.

TurtleBot3'ün entegrasyonu ROS (Robot Operating System) ile sağlanılmış. ROS, robotik uygulamaların geliştirilmesi için popüler bir platformdur. TurtleBot3, ROS üzerinde çalışır ve ROS paketlerini kullanarak robotu kontrol etmek, algılamalar yapmak ve hareket etmesini sağlamak için güçlü bir araç sunar.

TurtleBot serisinin 4 versiyonu vardır (Şekil 3.1). Her seride sunulan özellikler birbirinden farklıdır. TurtleBot serileri arasındaki bazı farklılıklar:

- TurtleBot1: TurtleBot platformunun orijinal sürümüdür. Genellikle Roomba robot süpürgelerinin temizlik üniteleri üzerine geliştirilmiş, mobil taban, kameralar ve sensörlerle donatılmıştır.
- TurtleBot2: TurtleBot2, daha gelişmiş bir sürümdür. Genellikle Kobuki mobil tabanının üzerine geliştirilmiş, daha güçlü donanımlara sahiptir ve kullanıcıların daha fazla özelleştirme ve genişleme yapmasına olanak tanır.
- TurtleBot3: TurtleBot3, birkaç farklı modelden oluşur ve her bir model farklı bir mobil tabana ve donanım setine sahiptir. Modeller arasında Burger, Waffle, Waffle Pi ve Waffle Pi Zero gibi seçenekler bulunur. TurtleBot3, güncellenmiş sensörler, iyileştirilmiş donanım entegrasyonu ve daha kompakt bir tasarıma sahiptir.
- TurtleBot4: TurtleBot serisinin en son neslidir, bu yeni versiyonda, daha güçlü bir mobil taban, daha hassas sensörler, daha gelişmiş algılama yetenekleri ve daha kullanıcı dostu bir tasarıma sahiptir. TurtleBot 4 Standard ve TurtleBot 4 Lite olarak 2 modeli bulunmaktadır.



Şekil 3.1. TurtleBot serisi.

TurtlteBot3'ün hareket kabiliyeti ve çevresel algılama yetenekleri sayesinde otonom sürüş, haritalama, nesne algılama ve takip gibi görevleri yapmak mümkündür.

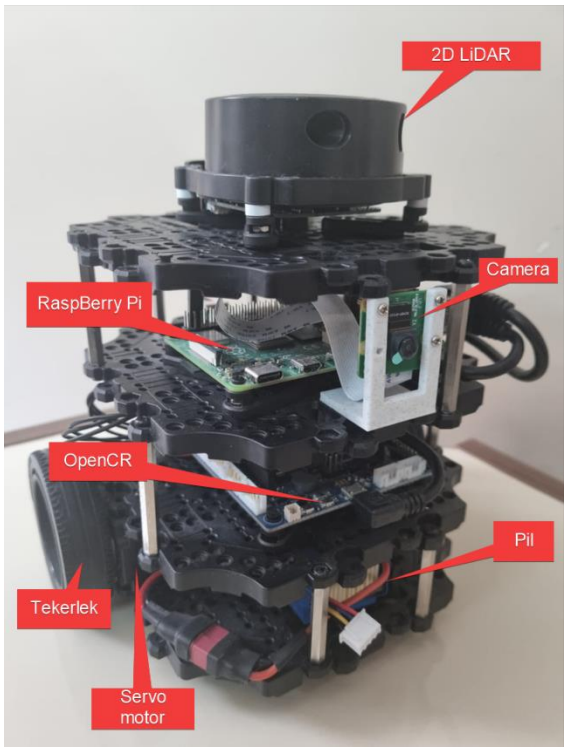
3.2. TurlteBot3'ün Donanımı

TurtleBot3, çeşitli donanım bileşenlerinden oluşur, bu bileşenler robotun hareket etme, çevreyi algılama ve karar verme yeteneklerini destekler. Bu bileşenlerin bir araya gelmesiyle TurtleBot3, otonom navigasyon, haritalama, nesne algılama gibi görevleri gerçekleştirebilir, robotun donanım bileşenlerini aşağıdaki gibi sıralayabiliriz:

- LiDAR: Robotun üst kısmına bir 360◦ lazer tarayıcı bulunur. Bu sensör, yakındaki engellerden yansıyan bir dönen lazer ışını gönderir. Yansıma süresi kullanılarak yakındaki nesnelere olan mesafeler elde edilebilir, bu da haritalama veya engel önleme için kullanılabilir.
- Raspberry Pi: Tek kartlı bir bilgisayar platformudur. Küçük boyutu, düşük maliyeti ve düşük güç tüketimine sahiptir.
- OpenCR: Raspberry Pi'nin altında yer alan OpenCR donanım kontrol kartı bulunur. Bu kart, Arduino UNO temel alınarak tasarlanmıştır. Kart, üç eksenli ivmeölçer ve jiroskop içeren Atalet Ölçüm Birimi (IMU) içerir. Temel işlevi Raspberry Pi'yi motorlara ve sensörlere bağlamak, veri okunmasını ve komut gönderilmesini sağlamaktır. Ayrıca, kart tüm bileşenler için güç bağlantıları sağlar.

- Aktüatör: Tekerlekleri çalıştıran ve robotun hareketini kontrol eden DC motorlar. TurtleBot3 Burger modelinde kullanılan aktüatör Dynamixel XL430-W250'dir. Ayrıca, tekerleğin dönüşü hakkında geri bildirim sağlayan ve doğru kontrol ve kilometre ölçümü sağlayan kodlayıcı da içerir.
- Atalet Ölçüm Birimi (IMU): İvme ve açısal hız dahil olmak üzere robotun yönünü ölçen bir sensör. 3 eksenli jiroskop ve 3 eksenli ivmeölçer kullanılmaktadır.
- Batarya: 11.1V Lithium Polymer (LiPo)

TurlteBot 3 modelin bileşenleri şekil 3.3'te gösterilmektedir.



Şekil 3.2. TurtleBot3 Burger modeli.

TurtleBot3, ROS'un bir parçası olarak geliştirilmiştir ve ROS üzerinde çalışır. TurtleBot3, ROS tarafından sağlanan çeşitli kütüphane, araç ve algoritmaları kullanır [2]. Bu sayede TurtleBot3, ROS ekosisteminden yararlanarak daha kolay geliştirilebilir, kontrol edilebilir ve genişletilebilir bir robot platformu haline gelir.

3.3. Robot İşletim Sistemi

ROS (Robot Operating System), robotların geliştirilmesi ve kontrol edilmesi için kullanılan bir açık kaynaklı yazılım platformudur. ROS, robotların algılama, hareket

kontrolü, navigasyon, iletişim ve daha birçok işlevini destekleyen bir dizi kütüphane, araç ve algoritma sunar. Farklı robot tipleri ve donanımlarıyla uyumludur.

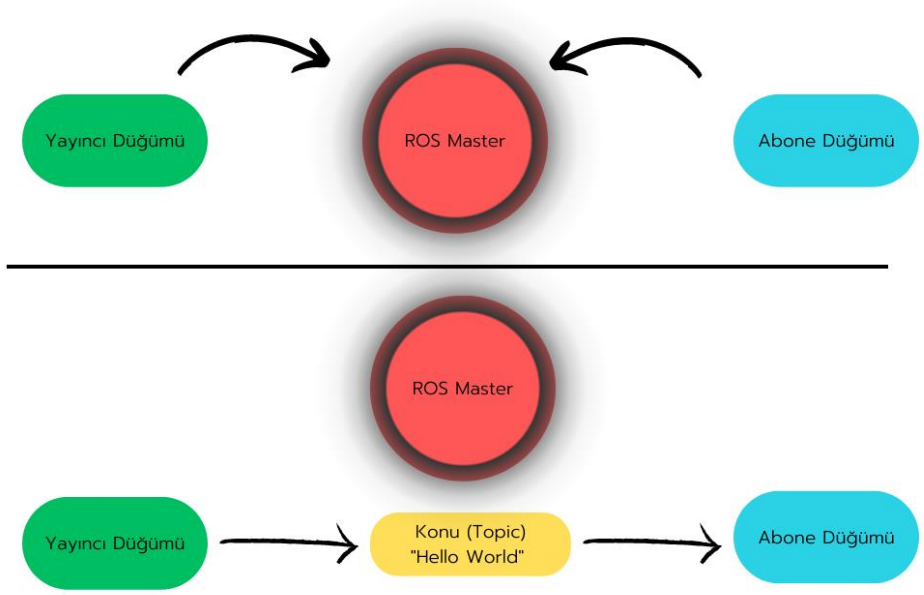
ROS, son yıllarda birçok robotik uygulama alanında yaygın olarak kullanılan bir standart haline gelmiştir. Robot geliştiricileri, araştırmacıları ve öğrencileri bir araya getiren geniş bir topluluğa sahiptir. Açık kaynaklı yapısı sayesinde, kullanıcılar ROS'un sağladığı kaynaklara erişebilir, mevcut bileşenleri kullanabilir, özelleştirebilir ve kendi robotik projelerini geliştirebilir.

ROS, düğümlerin birbirleriyle iletişim kurduğu bir mimari olan ve yayınlama-abone (publish-subscribe) mesajlaşma modelini kullanan bir sistemdir. Düğümler, işlem yapabilen birimler olarak tanımlanır ve belirli bir görevi yerine getirirler [4].

ROS'un yapılandırılmasında ve çalışma prensiplerinde temel kavramlar bulunmaktadır. Bu kavramlar, ROS'un nasıl çalıştığını, nasıl iletişim sağladığını ve robot uygulamalarının nasıl geliştirildiğini açıklar. ROS'un temeli aşağıdaki unsurlara dayanır:

- Master: ROS sistemi, host bilgisayar olarak adlandırılan bir bilgisayar üzerinde çalışır. host bilgisayar, düğümler arasında iletişimi sağlar. Düğümler, host bilgisayara kaydolur ve diğer düğümlerle iletişim kurmak için host bilgisayarı kullanır [5].
- Düğüm (Node): ROS uygulamaları, birbirleriyle iletişim kurabilen ve belirli görevleri gerçekleştiren düğümlerden oluşur. Her düğüm, bağımsız olarak çalışabilir ve mesajlaşma yoluyla diğer düğümlerle iletişim kurabilir.
- İletişim: ROS, düğümler (nodes) arasında yayınlama-abone (publish-subscribe) mesajlaşma modelini kullanarak iletişim kurar. Bir düğüm, belirli bir görevi yerine getiren bir yazılım parçasıdır. Düğümler, yayınlanan mesajları alabilir veya kendi mesajlarını yayınlayabilirler. Bu sayede düğümler arasında veri paylaşımı ve senkronizasyonu gerçekleştirilebilir [5]. (Şekil 3.3.).
- Topic: Yayınlama-abone modelini kullanarak düğümler arasında veri iletişimini sağlayan iletişim kanallarıdır [5].
- Servisler: Servisler, ROS'ta bir düğüm, başka bir düğüme bir istek gönderebilir ve isteğe bağlı olarak yanıt alabilir. Belirli bir görevi yerine getirmek için diğer düğümlerden istek göndermek için kullanılabilir [5].

- Paket (Package): ROS uygulamaları, paketler halinde organize edilir. Her paket, belirli bir işlevi yerine getiren düğümleri, mesajları, hizmetleri ve diğer dosyaları içerir. Paketler, uygulamaların modüler bir şekilde geliştirilmesini ve yönetilmesini sağlar.
- Parametreler: ROS, parametreler aracılığıyla düğümlerin yapılandırılmasını sağlar. Parametreler, düğümlerin davranışını değiştirmek veya ayarlamak için kullanılabilir. Örneğin, bir hareket kontrol düğümü için hız veya hedef konum parametreleri tanımlanabilir [5].
- Bag: ROS mesaj verilerini kaydetmek ve oynatmak için bir formattır. Bag'ler, algılayıcı verileri gibi toplanması zor olan ancak algoritmaların geliştirilmesi ve test edilmesi için gereken verilerin saklanması için önemli bir mekanizmadır [5].



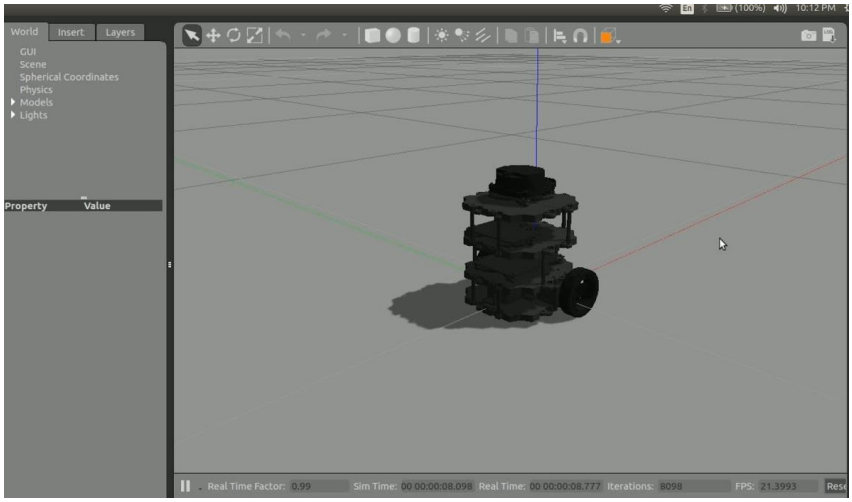
Şekil 3.3. ROS'ta iletişim.

3.4. Gazebo Simülatör

Gazebo, ROS için popüler bir 3D simülasyon ortamıdır. ROS ve Gazebo birlikte kullanıldığında, gerçek robotları fiziksel olarak inşa etmek veya test etmek yerine sanal bir ortamda simüle etmek mümkün olur [6].

Gazebo, robotik uygulamaların simülasyonu için geniş özellikler sunar. Robot modellemesi, sensör simülasyonu, fizik simülasyonu, kontrol ve planlama

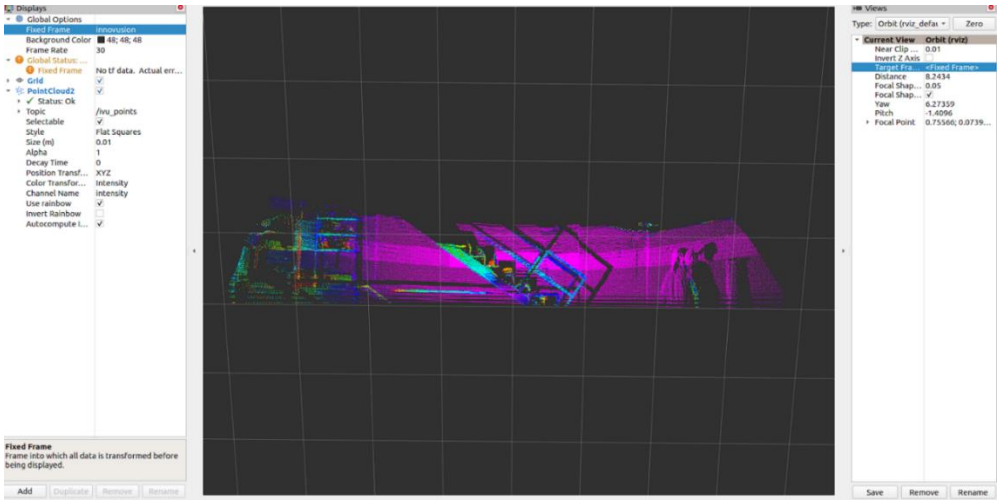
algoritmalarının test edilmesi gibi çeşitli kullanım senaryolarını destekler. Gazebo içinde bir veya birden fazla robot modeli oluşturabilir, çevre koşullarını simüle edebilir, gerçekçi fiziksel davranışları taklit edebilir ve simülasyon sürecinde çeşitli sensör verilerini elde edebilirsiniz. ROS ve Gazebo'nun birleşimi, robot sistemlerinin geliştirme, test ve hata ayıklama süreçlerini kolaylaştırır. Gazebo üzerinde yapılan simülasyonlar, gerçek koşullarda robot davranışını gözlemlemek, algoritma optimizasyonu yapmak ve çeşitli senaryoları test etmek için değerli bir araç sağlar. (Şekil 3.4).



Şekil 3.4. Gazebo programını kullanarak turtlebot3 burger modelini incelemek.

3.5. Rviz

Rviz (ROS visualization), Robotlar ve algoritmalar için bir 3D görselleştirme yazılım aracıdır. Robotik uygulamaların görsel bir arayüz üzerinde izlenmesini, analiz edilmesini ve hata ayıklanmasını sağlamaktadır. Sensör verilerini, haritaları, planlama sonuçlarını ve diğer robotik bilgileri görselleştirebilir (Şekil 3.5.). Bu sayede, robotların gerçek zamanlı durumlarını izlemek, algılama sonuçlarını görsel olarak takip etmek, planlama sonuçlarını gözlemlemek ve hataları tespit etmek için kullanılabilir.



Şekil 3.5. Rviz ortamında 3D LiDAR verilerini görüntülemek

3.6. SLAM

Eş Zamanlı Konum Belirleme ve Haritalama (Simultaneous Localization and Mapping - SLAM), bir robotun aynı anda konumunu belirlemesini ve çevresini haritalamasını sağlayan bir tekniktir. SLAM, mobil robotların bilinmeyen bir ortamda hareket ederken hem kendilerinin konumunu tespit etmelerine hem de çevredeki nesnelerin konumlarını haritalamalarına olanak sağlar [11].

S. Riisgaard, ve M. Rufus Blas göre Zamanlı Konum Belirleme ve Haritalama (SLAM), bir alanın haritasını oluştururken aynı zamanda cihazın o alan içindeki konumunu takip etme sürecidir [11]. Bu, mobil haritalamanın mümkün olmasını sağlar. Bu sayede, mobil robotlar, insansız hava araçları veya araçlar kullanılarak geniş alanların haritaları çok daha kısa sürede oluşturulabilir. SLAM sistemleri veri toplama sürecini kolaylaştırır ve açık hava veya kapalı mekân ortamlarında kullanılabilir.

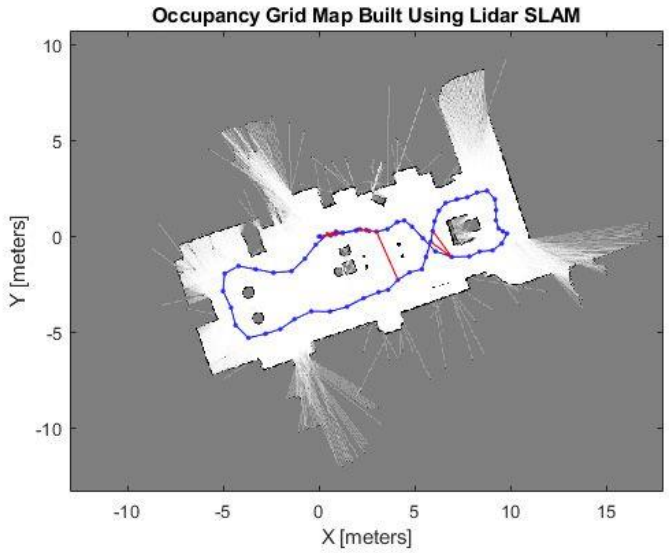
SLAM hem 2D hem de 3D ortamlarda kullanılabilen bir teknolojidir. Bu çalışmada sadece 2D ortamın ele alacağız. 2D SLAM, bir robotun düzlemde hareket ettiği ve çevresini 2D bir harita olarak haritaladığı durumlarda kullanılır. Bu durumda, robotun konumunu ve çevredeki nesnelerin konumlarını belirlemek için genellikle 2D sensörler kullanılır, örneğin Turtlebot3'ın üzerindeki takılan LDS-02 LiDAR sensörüdür (Şekil 3.6.).

Şekil 3.6. LDS-02 LiDAR.

3.7. LiDAR SLAM

Işık algılama ve mesafe belirleme (lidar), öncelikle bir lazer sensörü (veya mesafe sensörü) kullanan bir yöntemdir [12].

Lazer sensörlerin verileri genellikle 2 boyutlu (x, y) veya 3 boyutlu (x, y, z) nokta bulutu (point Cloud) verileridir. Lazer sensör nokta bulutu, yüksek hassasiyetli mesafe ölçümü sağlamaktadır [12]. Bu çalışmada x ve y eksende 2D SLAM kullanılacaktır (Şekil 3.7.).



Şekil 3.7. 2D LiDAR SLAM.

Lidar sensörü, lazer ışınlarını çevrelerine gönderir ve yansıyan ışınları algılar. Bu yansımaların zaman ve açı bilgileri kullanılarak, robotun etrafındaki nesnelerin mesafeleri ve konumları hesaplanır. Lidar-SLAM, lidar sensöründen gelen verileri kullanarak robotun anlık konumunu tahmin eder ve aynı zamanda çevredeki nesnelerin konumlarını belirler. Bu bilgileri birleştirerek, robotun konumunu doğru bir şekilde

belirleyebilir ve çevre haritasını oluşturabilir. Mobil robotların iç mekânda veya kapalı ortamlarda navigasyon yapmak için kullanılabilmektedir.

SLAM'da bazı zorluklar bulunmaktadır. Bunları aşağıdaki gibi sıralayabiliriz:

1. Lokalizasyon hataları birikerek gerçek değerlerden önemli ölçüde sapmalara neden olur: Kullandığımız sensörlerin hassasiyeti ciddi ölçüden önemlidir. Sensörlerdeki kalibrasyon eksiklikleri, ölçüm hataları, gürültü, yanlış okumalar veya doğrusal olmayan yanıtlar gibi sorunlar hatalara neden olabilir, bu hatalar zamanla birikiyor ve robotun konumunu yanlış hesaplamasına ve gerçek değerlerden önemli ölçüde sapmasına neden olur.
2. Lokalizasyon süreci başarısız oluyor ve haritadaki konum kayboluyor: Görüntü ve nokta bulutu haritalama, bir robotun hareket karakteristiklerini dikkate almaz. Bu yaklaşım bazı durumlarda kesintili konum tahminleri üretebilir. Örneğin, 1 m/s hızla ilerleyen bir robotun aniden 10 metre ileriye atladığını gösteren bir hesaplama sonucu [12].
3. Görüntü işleme, nokta bulutu işleme ve optimizasyon için yüksek hesaplama maliyeti: yüksek doğruluk ve hassasiyet sunabilmek adına görüntü işlemenin ve nokta bulutunun kalitesi yüksek olmalı, verilerin akışı yüksek frekansta gelmeli, yoğunluğu yüksek olan verilerin proses etmesi maliyetlidir. Görüntü işleme için yüksek performansa sahip CPU ve GPU kullanılmalı, bununla birlikte soğutma sistemleri ve güç dağıtım sistemleri

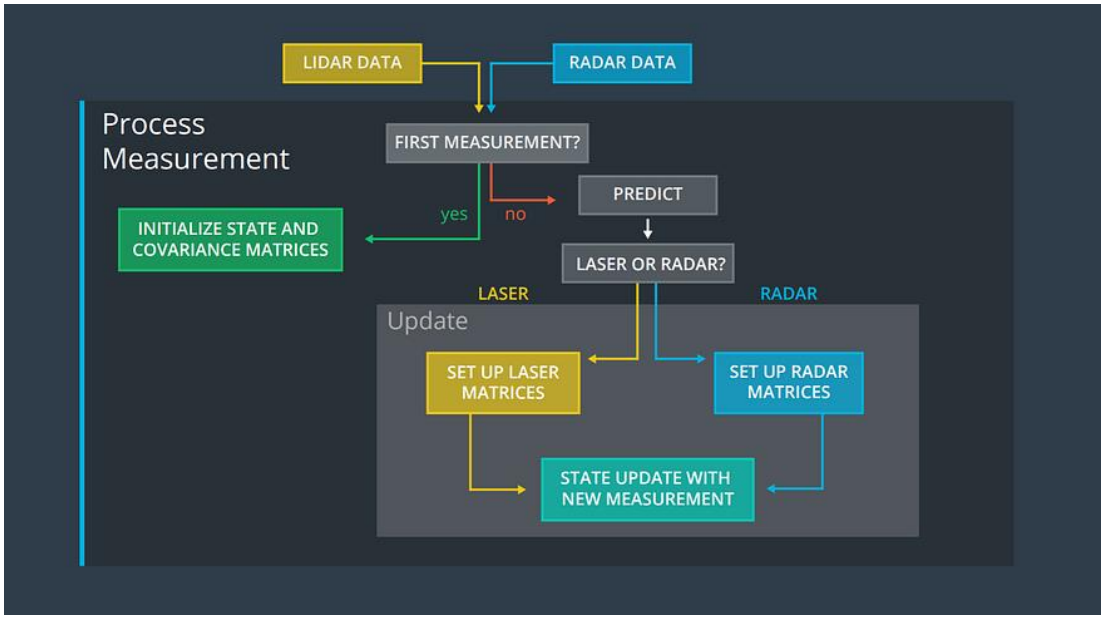
3.8. SLAM'da Kullanılan Algoritmamalar

Eş Zamanlı Konum Belirleme ve Haritalama (SLAM) ve navigasyon sistemlerinde, robotların doğru konum belirleme, harita oluşturma, hata düzeltme ve güvenli navigasyon yeteneklerini sağlamak önemlidir. Bu amaçla, Dead Reckoning ve Genişletilmiş Kalman filtresi gibi yöntemler kullanılır. Bu iki yöntemin tanımlarını şöyle özetlenebilir:

Dead Reckoning: Bir robotun mevcut konumunu önceden belirlenmiş bir konumu kullanarak hesaplama işlemidir [14]. Bu hesaplamalarda genelde tekerlek ve ivmeölçerler sensörlerinden gelen verileri kullanılmaktadır. Dead Reckoning, zamanla hata birikimi yapabilir, dış faktörlerin (sürtünme, tekerlek kayması) etkisiyle gerçek konumdan sapmalar olabilir. Bu nedenle, doğru konum belirlemek için diğer sensörler ile kullanabilir, örneğin GNSS/INS.

Extended Kalman filter (Genişletilmiş Kalman filtresi): Nonlineer sistemlerin önceki durumlarına göre, bir sonraki durumlarını tahmin etmek için kullanılan bir filtreleme yöntemidir [15]. Şekil 3.8. adar- LiDAR entegreli EKF için çalışma prensibini gösterilmektedir.

Dead Reckoning, sensör verilerini kullanarak anlık konum tahminleri yapar, ancak zamanla hatalar biriktirir ve sapmalara neden olabilir. Bu hataları düzeltmek için Genişletilmiş Kalman filtresi kullanılır. Bu filtre, sensör verilerini ve hedeflenen konumu birleştirerek daha doğru konum tahminleri yapar. SLAM ve navigasyon sistemlerinde Dead Reckoning ve Genişletilmiş Kalman filtresi kullanılarak robotların doğru konum belirleme, harita oluşturma, planlama ve güvenli hareket etme kabiliyetlerini geliştirir.



Şekil 3.8. Radar- LiDAR entegreli EKF için çalışma prensibi.

3.9. Dijkstra Algoritması

Dijkstra algoritması, graf tabanlı bir yol bulma algoritmasıdır. Bir graf içerisindeki düğümler arasında en az maliyetli yolu bulmak için etkili bir genişlik-öncelikli arama yöntemidir. Algoritma, 1959 yılında E.W. Dijkstra tarafından geliştirilmiştir [20]. Algoritma, başlangıç düğümünden itibaren tüm diğer düğümlere olan mesafeleri güncelleyerek en kısa yolu bulur. Dijkstra algoritması, her adımda en küçük mesafeye sahip düğümü seçerek ve onu işaretleyerek çalışır. İşaretlenen düğümler, başlangıç

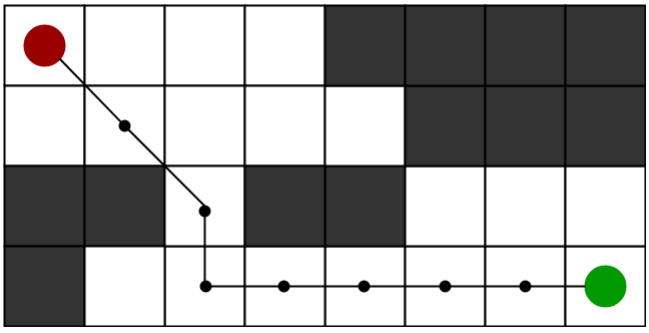
düğümünden itibaren en kısa yolun bilgisini içerir. Bu şekilde, başlangıç düğümünden hedef düğüme olan en kısa yolun bulunması sağlanır.

3.10. A-Star Algoritması

A* veya A-Star algoritması, metrik veya topolojik yapılandırma alanında uygulanabilen en tanınmış yol planlama algoritmalarından biridir [16]. En kısa yol bulmak için kullanılan algoritmalardan birisidir. A* algoritması, her bir hücrenin bir düğümü temsil ettiği bir hücre haritasına dayanır. A* algoritması, arama yapmak için her olası bir sonraki konum için bir maliyet fonksiyonu (cost function) hesaplayarak çevreyi keşfeder ve ardından arama alanına eklemek için en düşük maliyetli konumu seçer [17]. "En düşük maliyetli konum" terimi, A* algoritmasının hedefe en uygun yolun bulunmasında kullanılan bir kavramdır. A* algoritması, her hücre için bir maliyet değeri hesaplar ve bu maliyet değerlerine göre bir öncelik sırası oluşturur. En düşük maliyetli konum, bu öncelik sırasına göre seçilen ve yol planlaması için en uygun olan konumu ifade eder. Yani, A* algoritması en düşük maliyetli konuma sahip olan hücreleri tercih ederek, en etkili yolun bulunmasını sağlar.

3.11. A* Algoritmasının Kullanımı

Bu algoritmada 2D Grid ortamında kullanılır, yani ızgara tabanlı bir ortamda kullanılır. Izgara, bir haritayı hücrelere bölen bir yapıdır. Bir kare ızgarada birçok engel bulunan ve başlangıç hücresi ile hedef hücresi verilen bir senaryoda, başlangıç hücresinden hedef hücresine mümkünse en kısa sürede ulaşmak isteniyorsa, A* algoritması her adımda bir düğümü (n), ‘f’ adı verilen bir değere göre seçer. Bu değer, diğer iki parametre olan ‘g(n)’ ve ‘h(n)’ değerlerinin toplamıdır. Her adımda en düşük ‘f(n)’ değerine sahip olan düğüm/hücre seçilir ve işlenir [18]. (Örneğin Şekil 3.9.).



Şekil 3.9. Bir kare ızgarada hedef belirlemesi.

A* algoritmasının formülü:

$$f(n) = g(n) + h(n) \tag{3.1}$$

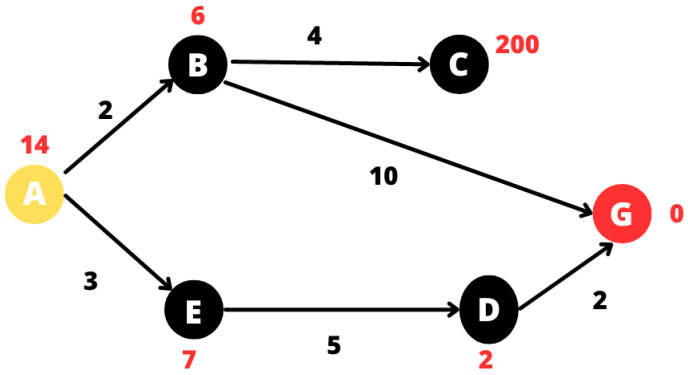
Burada:

f(n): Toplam maliyet

g(n): Gerçek maliyet (Actual Cost), başlangıç noktasından bir n düğümüne giden yolun uzunluğudur/maliyetidir.

h(n): Tahmini maliyet (Heuristic Cost), mevcut düğümden hedef düğüme varmak için tahmin edilen mesafedir/maliyetidir.

Örnek verecek olursak, Şekil 3.10.'da A* algoritmasını kullanarak A düğümden (başlangıç noktası) G düğüme en kısa yolu bulmak için aşağıdaki adımlar takip edilmelidir:



Şekil 3.10. Başlangıç noktasından varış noktasına olası yollar.

Başlangıç noktasında en yakın düğümlere bakılır, sonra her düğüm için maliyet hesaplanır (f(n) değeri bulunur):

A-B yolu için: f(n) = g(n) +h(n) = 2 + 6 = 8

A-E yolu için: 3 + 7 = 10

Burada A-B maliyeti daha düşük (yol uzunluğu), A-B yolu ile devam edilir

A-B-C yolu için: f(n) = g(n) +h(n) = (2+4) + 200 = 206

A-B-G yolu için: f(n) = g(n) +h(n) = (2+10) + 0 = 12

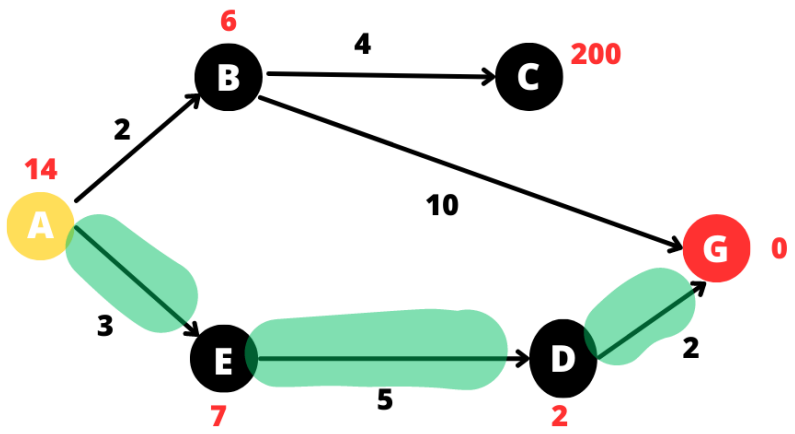
Burada A-B-G ile hedef düğüme ulaşıldı, A-B-G maliyeti12’dir.

Sonra A-E ile devam edilir.

A-E-D yolu için: f(n) = g(n) +h(n) = (3+5) + 2 = 10

A-E-D-G yolu için: f(n) = g(n) +h(n) = (3+5+2) + 0 = 10

Tüm yollar incelendikten sonra A-E-D-G yolu hedef düğüme ulaşmak için en düşük maliyete sahiptir (Şekil 3.11.).



Şekil 3.11. A* ile en kısa yolu bulması.

3.12. A* Algoritması İle Dijkstra Algoritması Arasındaki Fark

A* algoritması, Dijkstra algoritmasıyla benzerlik gösterirken, tek fark ise A* algoritmasının heuristik bir fonksiyon kullanarak daha iyi bir yol aramaya çalışmasıdır. Heuristik fonksiyon h(n), düğümler hakkında ek bilgi sağlar ve A* algoritmasının belirli düğümlere öncelik vermesini sağlar, çünkü bu düğümlerin diğerlerinden daha iyi olması beklenirken, Dijkstra tüm olası yolları keşfeder."

Dijkstra algoritması tüm olası yolları keşfederken tercih yapmazken, A* algoritması heuristik fonksiyonu kullanarak aramayı yönlendirir. Heuristik bilgiye dayanarak daha umut verici olan düğümlere odaklanır. Bu şekilde A* algoritması, arama alanını

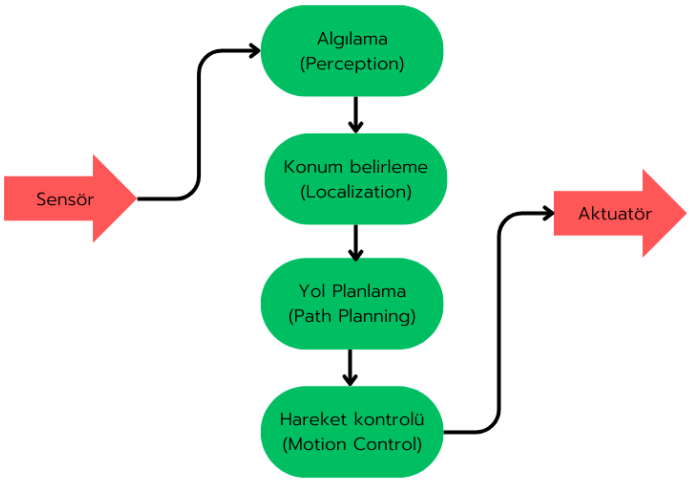
daraltır ve gereksiz yolları keşfetmekten kaçınarak genellikle en kısa yolu daha verimli bir şekilde bulabilir.

3.13. Navigasyon

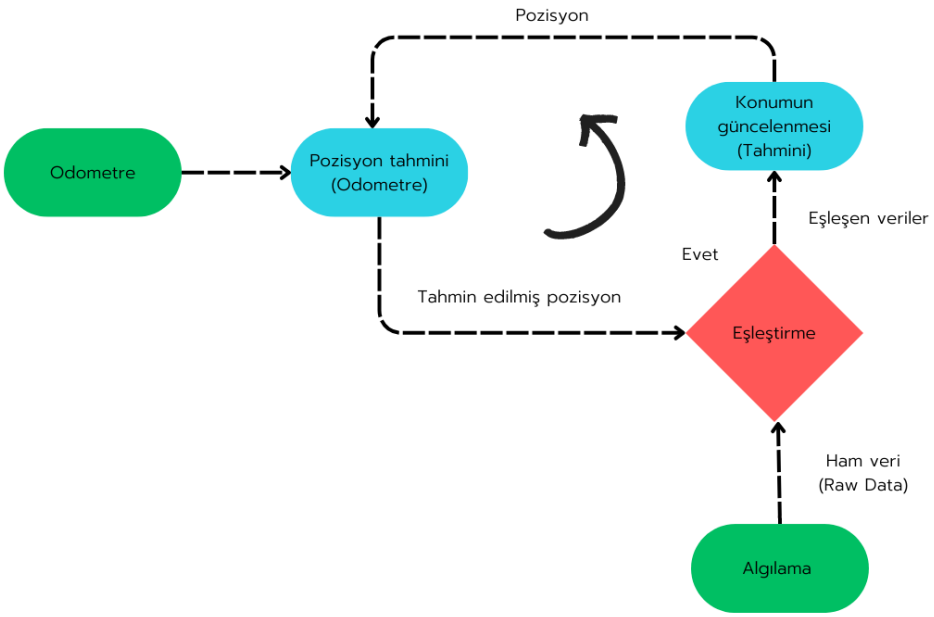
Navigasyon, bir yerden diğerine yolculuk yaparken veya bir hedefe ulaşırken kullanılan yönlendirme ve yol bulma sürecidir. Navigasyon, bir aracın güvenli ve etkili bir şekilde yolunu belirlemesine, yönünü tespit etmesine ve hedefe ulaşmasına yardımcı olur. Genellikle haritalama, konum belirleme (Lokalizasyon) ve yol planlama gibi temel bileşenleri içerir (Şekil 3.12.).

Siegwart ve Nourbakhsh'ın "Introduction to Autonomous Mobile Robots" adlı kitabıa göre başarılı bir navigasyon süreci navigasyonun dört temel bileşeninde başarılı olmayı gerektiri [8]. Bu temel bileşenler şöyle sıralanabilir; Algılama, Konum Belirleme (Lokalizasyon), Yol Planlama ve hareket Kontrolö.

1. Algılama (Perception): Robotun sensörlerinden anlamlı verileri çıkarması gereklidir
2. Konum belirleme (Localization): Robotun çevredeki konumunu belirlemesi gereklidir
3. Yol Planlama (Path Planning): Robotun hedeflerine ulaşmak için nasıl hareket edeceğine karar vermesi gereklidir
4. Hareket kontrolü (Motion Control): Robotun istenen izlemeyi elde etmek için motor çıkışlarını ayarlaması gereklidir



Şekil 3.12. Harita tabanlı navigasyon mimarisi.



Şekil 3.13. Mobil robot lokalizasyonu için genel şema.

3.13.1. Algılama (perception)

IMU (Inertial Measurement Unit) ve tekerlek sensörü (encoder) gibi sensörler kullanılır, tekerlek sensörü ve inertial sensör (IMU sensörü) kullanılarak odometri bilgisi güncellenir ve sensörün konumundan nesnelerin uzaklığı ölçülür

3.13.2. Konum belirleme (localization)

Tekerlek kodlayıcıdan tekerlek dönüş miktarı, IMU sensöründen robotun hızını, ivmesini ve dönüş hareketini temel alınarak, mevcut robotun önceden oluşturulan harita üzerinde yer belirlemesi/tahmini gerçekleştirilmektedir. Birçok yer belirleme yöntemi bulunmakla birlikte, bu bölümde parçacık filtre yer belirleme yöntemi ve Monte Carlo Yer Belirleme'nin (MCL) bir türevidi olan Adaptif Monte Carlo Yer Belirleme (AMCL) yöntemleri kullanılmaktadır.

3.13.3. Yol planlama (path planning)

Hareket planlaması, aynı zamanda yol planlaması olarak da adlandırılır, mevcut pozdan haritada belirtilen hedef poza bir yörünge (trajectory) oluşturur. Oluşturulan yol planı, tüm haritadaki global yol planlamasını (Global Path Planning) ve robotun etrafındaki daha küçük alanlar için lokal yol planlamasını (Local Path Planning) içerir. Bu çalışmada, ROS'ta 'move_base' yol planlama paketini, engel kaçınma algoritması olan Dinamik Pencere Yaklaşımı (DWA) temelli olarak kullanılması planlanmaktadır.

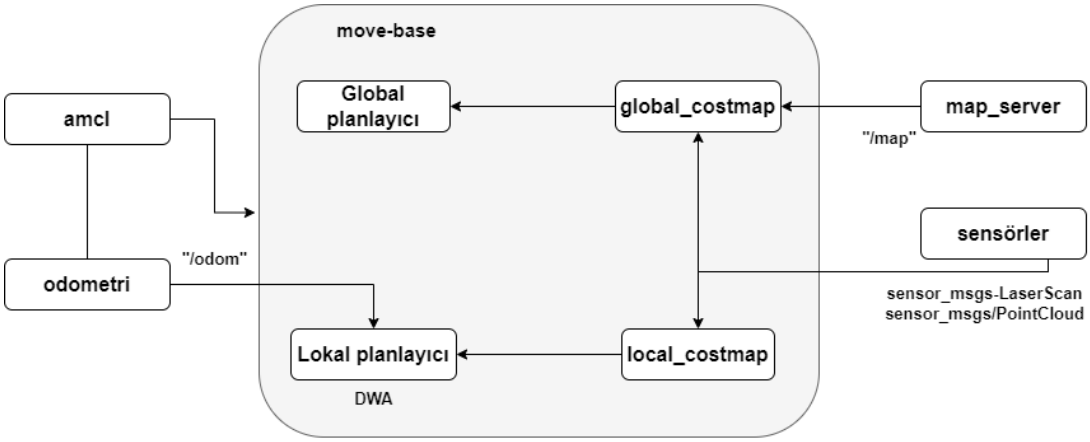
3.13.3.1. Move Base

Navigasyon parçasının temel bir bileşenidir ve robotun mevcut konumundan belirtilen hedef konuma otomatik olarak navigasyonunu gerçekleştirmede önemli bir pakettir.

Move_base, global ve lokal yol planlama, engel kaçınma ve robot kontrolünü yönetir.

Şekil 3.14'de move_base düğümünün yapısını göstermektedir.

TurtleBot3 navigasyon paketinde ayrı bir global planlayıcı bulunmasa da, "move_base" paketi içinde DWA (Dynamic Window Approach) lokal planlayıcıyı kullanarak global yol planlaması gerçekleştirir. DWA lokal planlayıcı, robotun mevcut konumundan belirtilen hedef konuma kadar olan bir yol oluşturur. Bu yol, robotun hedefe ulaşmak için takip etmesi gereken global yolun bir yaklaşımıdır. Gerçek zamanlı sensör (LIDAR, IMU ve Odometeri) geri beslemeleri ve çevre bilgileriyle gerçekleşen lokal ayarlamalar yapar. Bu sayede robot, engelleri kaçınabilir ve hareketini dinamik bir şekilde ayarlayarak düzgün ve güvenli bir şekilde ilerler. Sensörlerden gelen geri bildirimleri alarak robotun mevcut konumunu tekrar değerlendirir ve yol planını buna göre yeniden düzenler.



Şekil 3.14. Move_Base Düğümünün yapısı.

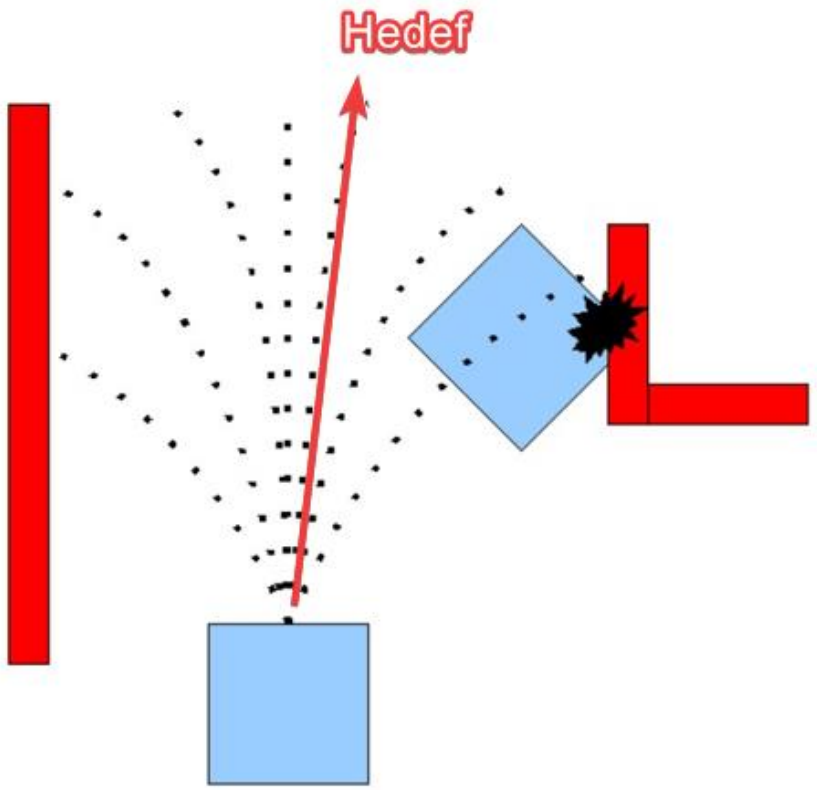
3.13.3.2. Global Planlayıcı (Global Planner)

Varsayılan TurtleBot3 navigasyon paketinde, sağlanan ayrı bir global planlayıcı yoktur. Bunun yerine, navigasyon için lokal planlayıcı olarak DWA (Dinamik Pencere Yaklaşımı) algoritması kullanılmaktadır.

3.13.3.3. Lokal Planlayıcı (Local Planner)

Move_Base düğümünde lokal planlayıcıyı olarak DWA (Dinamik Pencere Yaklaşımı) kullanılır, DWA lokal planlayıcıyı lokal yol planlaması için kullanır.

Dinamik pencere yaklaşımı, robotun mevcut durumuna ve çevresine bağlı olarak anlık olarak güvenli ve etkin bir hareket planı oluşturur. Planlayıcı robotun bir başlangıç noktasından hedef bir konuma gitmek için kinematik bir yol oluşturur. Yolda ilerlerken, planlayıcı, robotun etrafında bir değer fonksiyonu oluşturur ve bu fonksiyon bir ızgara haritası olarak temsil edilir. Bu değer fonksiyonu, ızgara hücrelerinden geçmenin maliyetlerini kodlar, Bu kıymet fonksiyonu, x, y ve açısal hızları belirlemek için kullanılır ve bunlar robotu hedefe yönlendirmek için robot'a gönderilir. (Şekil 3.15)



Şekil 3.15. DWA Lokal Planlayıcı

3.13.4. Hareket kontrolü (motion control)

Eğer bir komut, hareket planlaması tarafından oluşturulan hareket izine dayalı olarak robota verilirse, robot planlanan yola göre hedefe doğru hareket eder. Algılama (sensing), konum tahmini (pose estimate) ve yol planlaması (path planning) hareket halindeyken hala yürütüldüğünden, aniden ortaya çıkan engeller veya hareketli nesneler, belirlenen bir algoritma kullanılarak kaçınılabilecektir. Örneğin Dinamik Pencere Yaklaşımı (DWA) algoritması.

4. TURTLBEBOT3 İLE OTONOM SÜRÜŞ

Turtlebot3 burger modeli ile otonom sürüş yapabilmek adına SLAM ve Navigasyon süreçleri önemlidir. Robotun etrafındaki çevreyi algılayabilmesi için 2D LiDAR, IMU (İnertial Measurement Unit - Hareket Sensörü Birimi) ve Odometre (Tekerlek Kodlayıcıları) kullanılmaktadır. IMU, üç eksenli ivmeölçer, jiroskop ve manyetometreden oluşur ve robotun eğim, hızlanma ve dönme gibi hareketlerini algılar. Odometre ise tekerleklerin dönme hareketini ölçer ve robotun konumunu hesaplamak için kullanılmaktadır.

SLAM, robotun kendi konumunu belirlemeyi hedefler. Bu metotla robot yeni alanları keşfederek ve sensör verilerini kullanarak harita oluşturmaya başlar, oluşturduğu haritada robotun kendi konumunu tahmin etmesini sağlar. SALM süreci tamamlandıktan sonra, SLAM süreci sonuncunda oluşturulan harita ve konum bilgisi kullanılarak Navigasyon sürecine başlanabilir. Böylece haritadaki engeller ve hedefler belirlenebilir, robotun konumu takip edilebilir ve hedefe doğru ilerlenebilir. Harita üzerinden hedefe en uygun yolu belirlemek için yol planlama algoritmalarını kullanılır. Yol planlama algoritmaları ile engel tanıma, engelden kaçınma ve sürüşe müsait alanlarda sürüş yapma imkânı sağlanır.

4.1. TurtleBot3 ile Bağlanmak

Bu çalışmada hem Real-Time hem simülasyon ortamında Turtlbebot3 robotunu kullanılacaktır. TurlteBot çerçevesinde çalışır ve ROS ekosistemiyle entegre bir şekilde kullanılır.

TurtleBot3 ile iletişim kurabilmek için HOST bilgisayar veya Remote PC kullanılması lazım, bu çalışmada “Remote PC” terimi ile devam edilecektir. Remote PC, TurtleBot3 Burger modeli ile iletişim kurmak için WiFi aracı açık olmalı ve doğru IP adresi kullanılmalıdır.

Remote PC’nin ve Robotun network ayalarını girebilmek için “bashrc” dosyasına erişilmeli, dosyaya erişebilmek için bu komut kullanılabilir “$ nano ~/.bashrc”,

sonrasında netwrok ayarları o dosyaya eklenir, network ayarları aşağıdaki gibi olmalıdır:

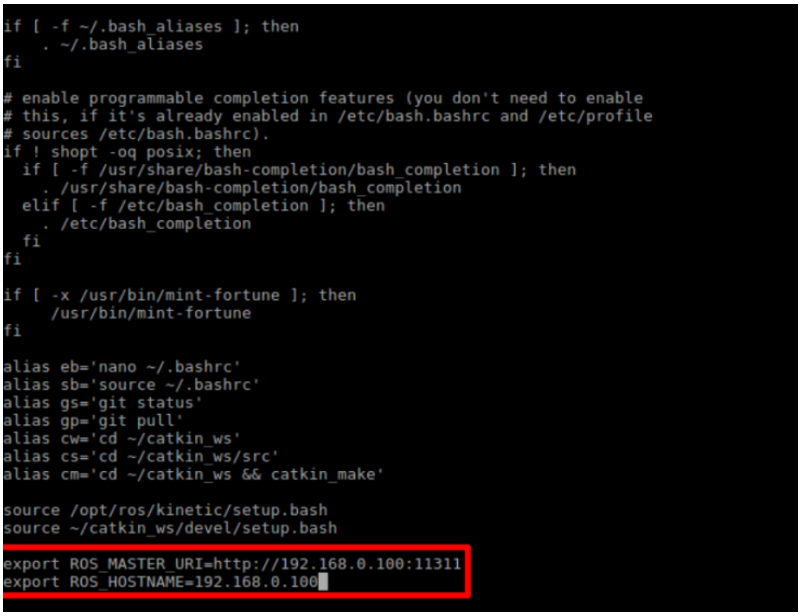
Remote PC'nin network ayarları:

- ROS_MASTER_URI = http://remot_pc_nin_ip_adresi:11311
- ROS_HOSTNAME = Remote PC'nin IP adresi

Robotun network ayarları:

- ROS_MASTER_URI = http://remot_pc_nin_ip_adresi:11311
- ROS_HOSTNAME = TurtleBot'un IP adrsesi

Şekil 4.1.'de robotun ağ ayarları gösterilmektedir.



Şekil 4.1. Ağ ayarları.

Network ayarları tamamlandıktan sonra Remote PC, TurtleBot ile bağlanmaya hazır olur, bağlanmak için "SSH" komutu kullanılır:

$ ssh ubuntu@Turtlebot'un IP adresi

4.2. Turtlbebot3'ün Başlatılması

Bringup TurtleBot terimi, TurtleBot'un çalışmaya başlaması veya hazır hale getirilmesi anlamına gelir. Robotun donanımının ve yazılımının başlatılması, bağlantıların yapılandırılması ve gerekli bileşenlerin etkinleştirilmesi işlemine atıfta bulunur. "Bringup" işlemi ROS'ta gerçekleştirilir, buna "turtlebot3_bringup" düğümü

denir. Düğümün işlevini yerine getirmek için ROS'ta bu düğümü çağıran veya çalıştıran başlatma dosaysı (launch file) çalıştırılır. Aşağıdaki komut çağrılır:

$ roslaunch turtlebot3_bringup turtlebot3_robot.launch

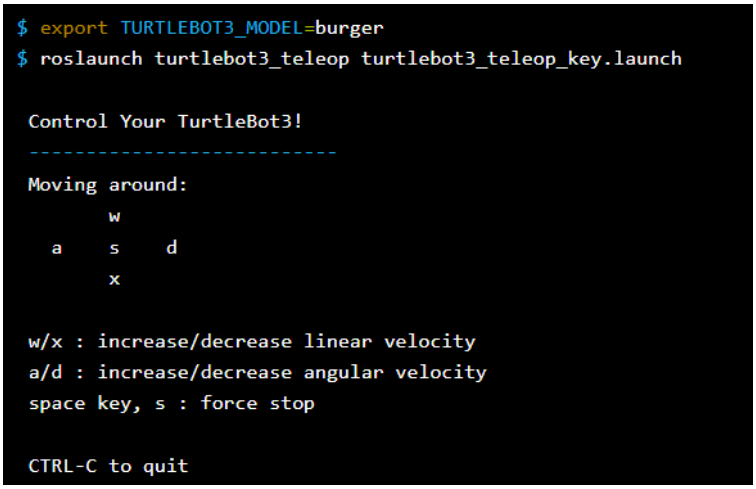
4.3. Turtlbebot3 ile SLAM

SLAM sürecini gerçekleştirmek için Robotu, Remote PC'yi kullanarak sürmek gerekir. TurlteBot SLAM düğümün/algoritması çalıştırıldıktan sonra sensörlerden gelen verileri kullanılarak robotun konumu tahmin edilir ve harita oluşturulması sağlanır. SLAM algoritmasını çalıştırmak için aşağıdaki komut kullanılır:

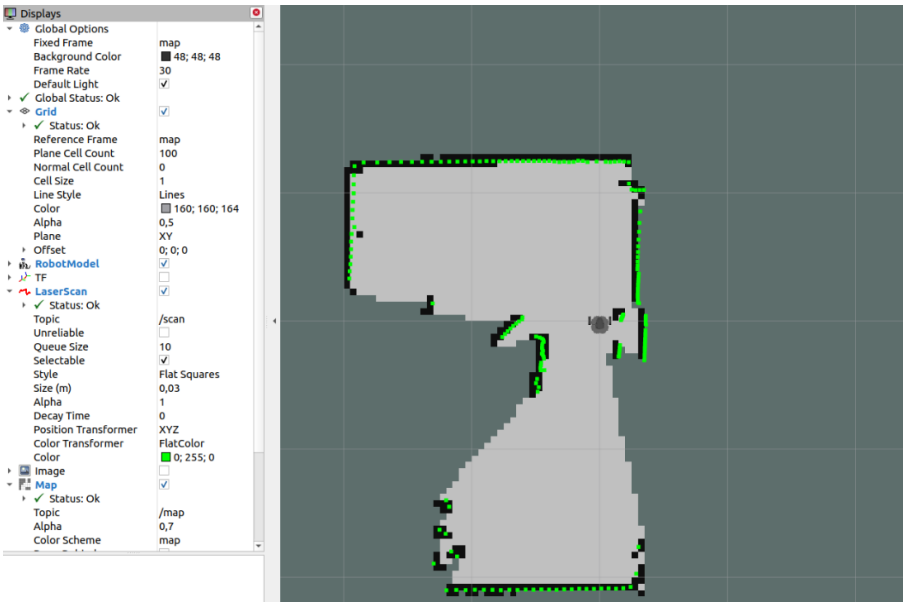
$ roslaunch turtlebot3_slam turtlebot3_slam.launch

Bu süreçte robot odometri ve lazer tarama verilerini kullanarak çevrelerini keşfetmeye başlar. SLAM algoritması, robotun konumunu güncelleyerek ve sensör tarafından elde edilen mesafe bilgisini kullanarak harita oluşturur ve robot gezdikçe, keşfettiği alanları haritaya ekler.

Robot, çevreleri keşfetmek için harekete geçmeli, keşfettiği alanları haritaya eklenir. Teleoperasyon ROS düğümüyle uzaktan kumanda veya klavye ile robotun hareketini kontrol etmek için Teleoperasyon Düğümü (Teleoperation Node) kullanılır (Şekil 4.2.). Remote PC'den hareket komutları gönderilir ve robot hareket etmeye başlar. Böylece robot çevreyi keşfeder. Şekil 4.3.'te rviz arayüzünde robotun çevreyi keşfettiğini göstermektedir



Şekil 4.2. Teleoperasyon ROS düğümüyle hareket komutları göndermek.



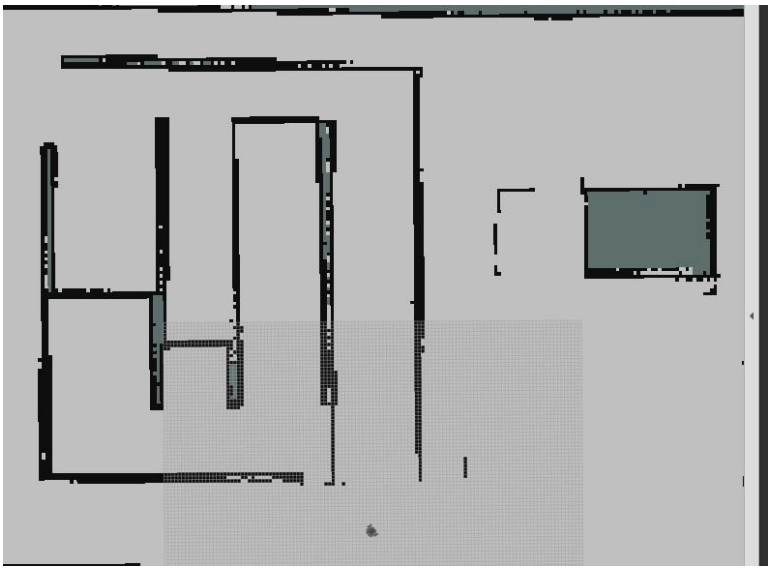
Şekil 4.3. rvizde SLAM algoritması ile alanları keşfetmek.

4.4. Haritayı Kaydetmek

SLAM süreci sonuncunda oluşturulan haritayı kaydetmek gerekir, kaydedilen harita Navigasyon sürecinde kullanılır. Haritayı kaydetmek için "map_saver" düğümü başlatılmalı, Harita "yaml" formatında kaydedilir. Aşağıdaki komut ile map_saver düğümü başlatılabilir:

```
$ rosrun map_server map_saver -f ~/map
```

Şekil 4.4'te gösterilen Doluluk Izgara Haritası (Occupancy Grid Map), bir bölgenin veya alanın ızgara tabanlı bir temsili olarak kullanılan bir harita türüdür. Her hücre, bölgedeki bir alana karşılık gelir ve o hücrenin işgal durumunu gösterir. İşgal edilmemiş alanlar genellikle boş veya açık olarak işaretlenirken, işgal edilmiş alanlar nesneler veya duvarlar tarafından doldurulmuş olarak işaretlenir.



Şekil 4.4. SLAM ile oluşturulan Doluluk Izgara Haritası (Occupancy Grid Map).

4.5. Turtlbebot3 ile Navigasyon

Navigasyon sürecinde, robotu belirli bir konumdan belirtilen hedefe hareket ettirmeyi hedefler. Bu süreçte, harita analizi, yol planlama, hareket kontrolü ve veri güncellemesi gibi işlemler yer almaktadır. Bunun için nesneler ve duvarlar gibi geometri bilgilerini içeren bir haritaya ihtiyaç vardır [12].

TurtleBot3 navigasyon paketi varsayılan olarak global planlama için önceden tanımlanmış bir algoritma içermez. Bu nedenle, navigasyon paketinizin global planlama yeteneği olmayacaktır. Ancak, paket içerisinde yerel planlama algoritması olan DWA kullanılacak ve robot, düşük seviyede çevresel değişiklere uyum sağlayarak anlık olarak güvenli ve etkin bir şekilde hareket edecektir.

Navigasyon sürecinde, TurtleBot3 robotu SLAM ile oluşturulan haritayı ve IMU sensörü ve LiDAR sensörü kullanarak mevcut konumundan harita üzerinde belirlenen hedef konuma hareket etmesi sağlar. DWA algoritması, harita analizini ve mevcut konumu dikkate alarak robotun çevresel değişikliklere uyum sağlayarak hedefe en kısa ve güvenli yoldan ulaşmasını sağlar. Bu sayede TurtleBot3, veri güncellemesi ve yol planlamayı birleştirerek güvenli ve etkili bir şekilde hedefe yönlendirilebilir.

Navigasyon görevini tamamlamak için aşağıdaki adımlar takip edilir:

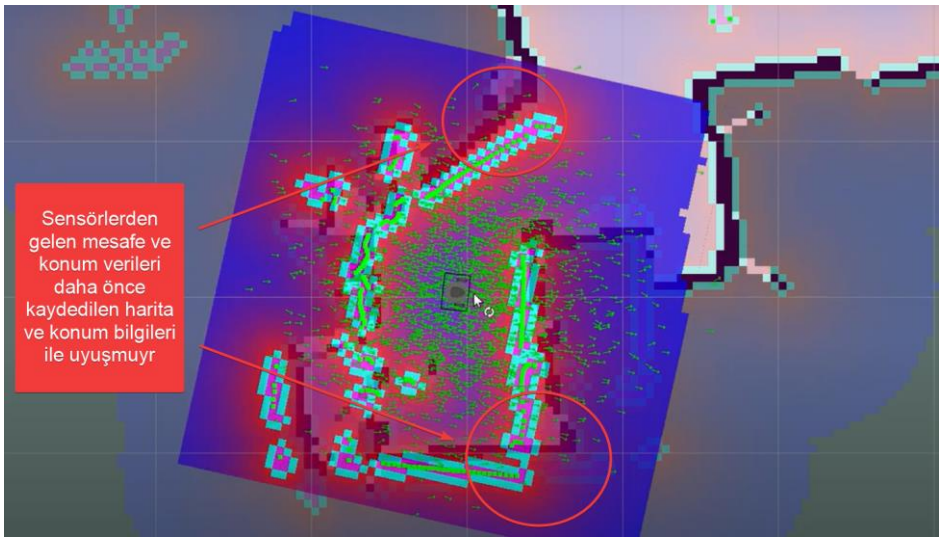
- Remote PC üzerinden “roscore” düğümü çalıştırılır.
- SSH komutu ile TurlteBot ile bağlantı kurulur.
- TurtleBot’un “turtlebot3_bringup” düğümü çalıştırılır.

- TurtleBot'un "turtlebot3_navigation" düğümü çalıştırılır. Bu adımda robot Navigasyon yapmaya başlar.
- SLAM sürecinde yaml formatında kaydedilen haritayı navigasyon düğümünü çağıran komuta eklenir.
- Navigasyonu düğümü için kullanılan komut: $ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml

Böylece, navigasyon süreci başlatılmış olur.

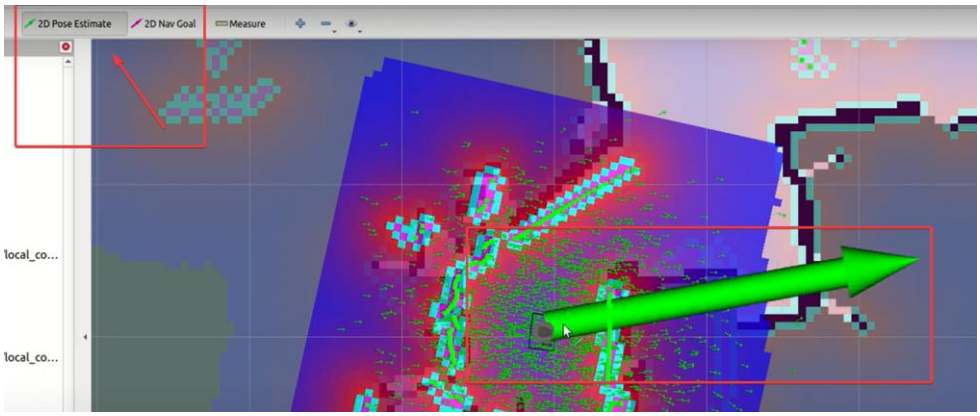
4.6. İlk Konum Tahmini

Navigasyon sürecin başlangıcında robotun başlangıç konumu (İnitial Position) doğru olmayabilir (Şekil 4.5.), belirli bir ortamda robotun başlangıç konumunu tahmin etme sürecine "İlk Konum Tahmini" denir. Bu tahmin, robotun hareketine veya algılama hatalarına bağlı olarak kaybedilen konum bilgisini geri kazanmayı amaçlar. İlk Konum Tahmini, robotun ilerleyen navigasyon, haritalama veya diğer görevleri için doğru bir başlangıç noktası sağlamak önemlidir.

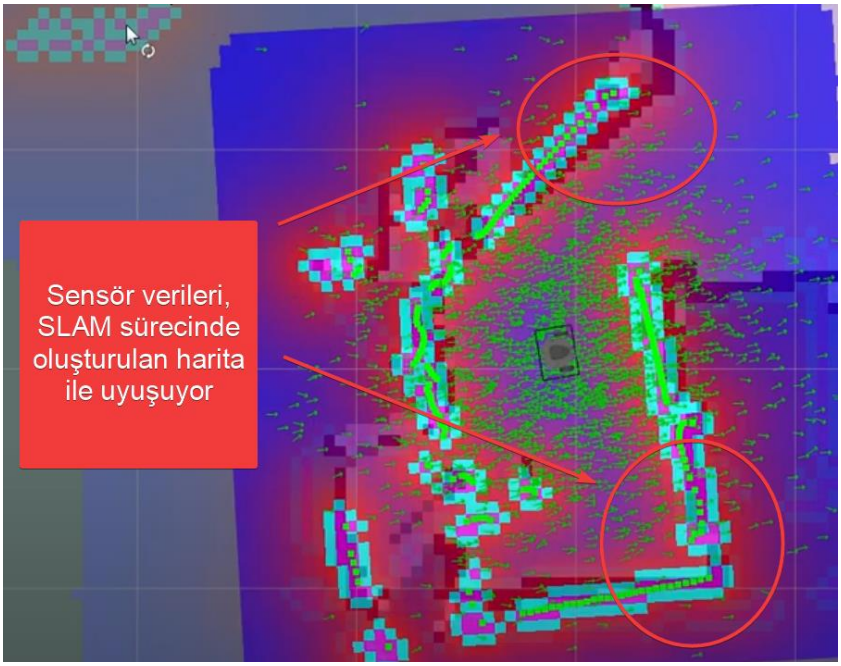


Şekil 4.5. Navigasyon Süreci – İlk konum tahmini yapılmadı.

Şekil 4.5.'te ilk konum tahmini yapılmadığından real-time olarak sensörlerden gelen konum bilgileri, daha önce SLAM sürecinde kaydedilen harita ve konum bilgileri ile uyuşmadı, bu da navigasyon sürecini olumsuz bir şekilde etkilemektedir. İlk konum tahmini yapabilmek için rviz arayüzünden "2D Pose Estimate" butona tıkalar ve sonra robotun tahmini konumunu seçeriz (Şekil 4.6.).



Şekil 4.6. İlk konum tahmin süreci.

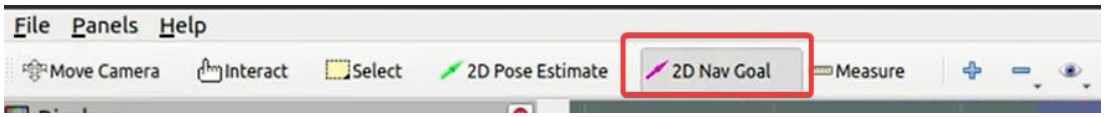


Şekil 4.7. İlk konum tahmini yapıldı.

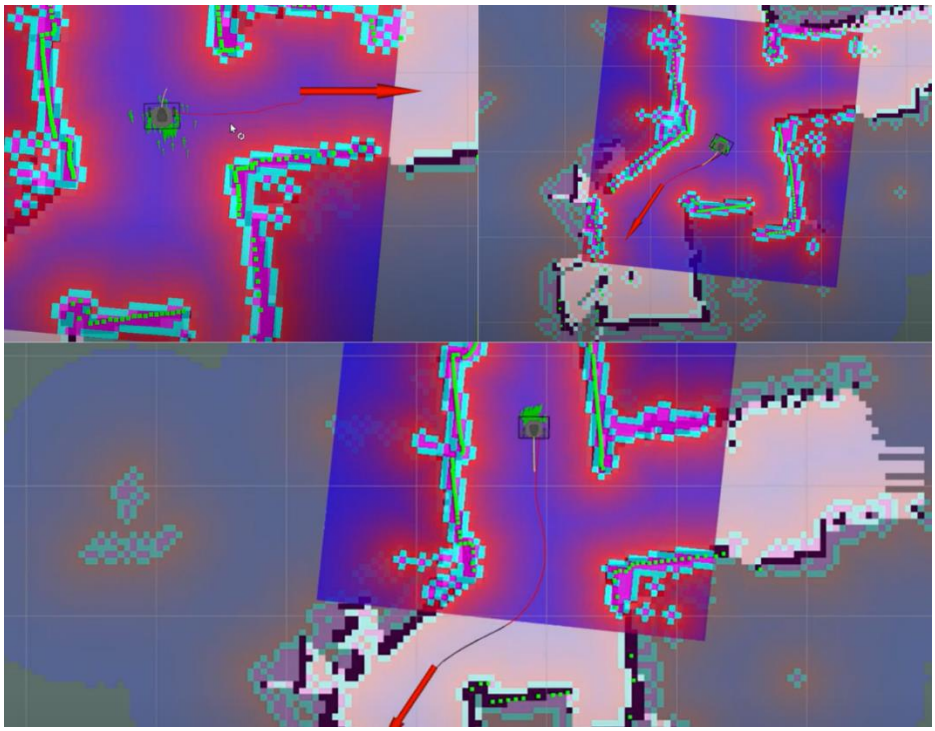
İlk konum tahmini yaptıktan sonra sensörden gelen veriler SLAM haritası ile uyuşmaya başlar (Şekil 4.7.). Böylece robot etkili bir şekilde hareket edebilir ve navigasyon görevi başlanabilir.

4.7. Hedef Belirleme

Navigasyon sürecinde robotun bulunduğu konumdan hedefe doğru hareket etmek için bir hedef noktası belirlenir. Hedef belirlemek için Rviz arayüzünden "2D Nav Goal" botuna tıklayarak ve harita üzerinde hedef noktasını seçerek hedef belirlenebilir (Şekil 4.8.). Hedef noktası belirlendikten sonra robot seçilen yol bulma algoritmasını kullanarak hedefe doğru hareket eder (Şekil 4.9.). Bu navigasyon düğümde Dynamic Window Approach (DWA) algoritması kullanılmıştır.



Şekil 4.8. Rviz arayüzü üzerinde hedef belirleme botunu.

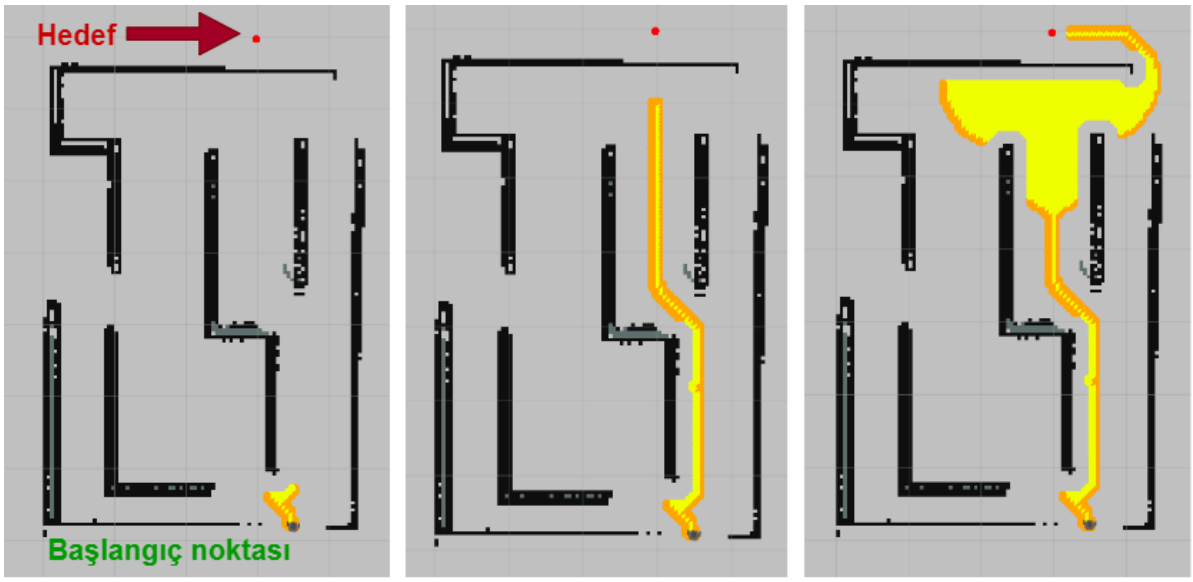


Şekil 4.9. Yol planlaması ve hedefe ulaşma.

5. A* ALGORİTMASI İLE YOL PLANLAMA- SİMÜLASYON

Navigasyon yaklaşımımızda, A* algoritmasını global planlayıcı olarak kullanarak, DWA lokal planlayıcı ile birleştiriyoruz. DWA, gerçek zamanlı engel kaçınma ve yerel yol oluşturma konularında üstün bir performans sergilerken, A* TurtleBot3'ün mevcut konumundan önceden belirlenmiş bir hedefe stratejik olarak optimal bir rota çizmektedir. A* algoritması, potansiyel yolları değerlendirerek, kat edilen mesafe ile tahmini kalan mesafenin bir kombinasyonuna dayalı olarak en verimli rotayı seçme yeteneğine sahiptir. Bu entegrasyon, A*'nın yüksek seviyeli rota planlamasını DWA'nın anlık çevre bilgilerini işleme yeteneği ile sorunsuz bir şekilde birleştirir; böylece hem global hem de lokal navigasyonda hassasiyeti sağlar. Bu uyumlu birleşim sayesinde TurtleBot3, gelişmiş yol planlama tekniklerinin bir araya gelmesiyle karmaşık ortamlarda etkili bir şekilde hareket eder. Bu navigasyon yaklaşımı, Gazebo simülasyon ortamında gerçekleştirilmektedir.

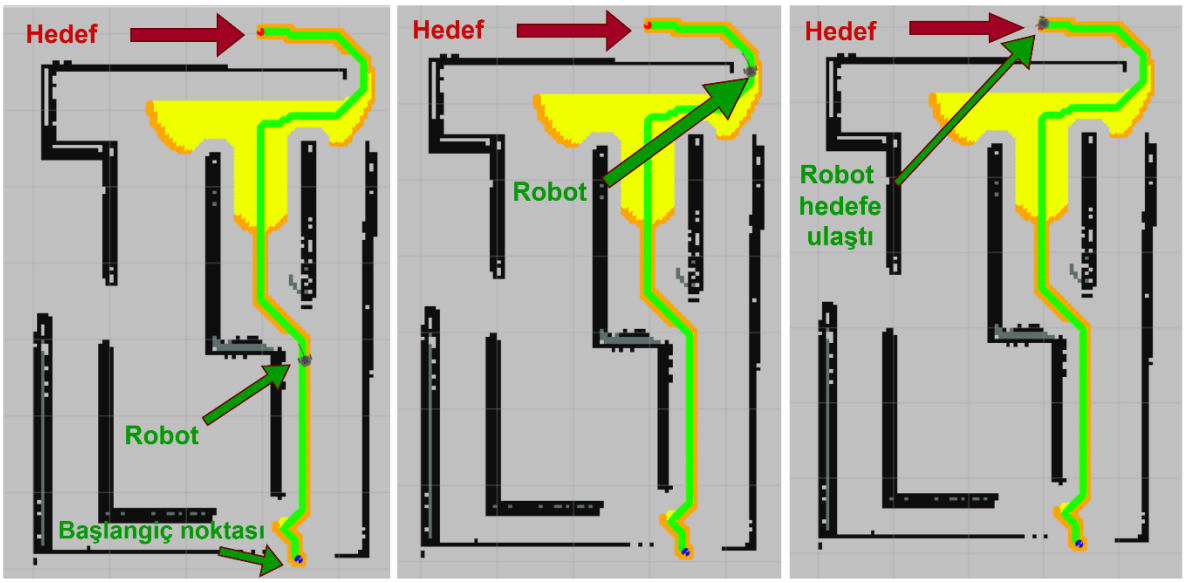
A* algoritması, ızgara haritasında her hücreyi düğüm olarak temsil eden bir graf üzerinde çalışır. Başlangıç düğümünden hedef düğüme en kısa yolu bulmak için kullanılır. Her adımda düğümlere f(n) olarak bilinen tahmini maliyet fonksiyonu atanır. Bu maliyet fonksiyonu, başlangıç düğümünden o düğüme olan gerçek maliyet ile hedef düğüme olan tahmini maliyetin toplamını gösterir. Düşük f(n) değerine sahip düğümler öncelikli olarak keşfedilir, bu da daha verimli bir yol planlaması sağlar. Algoritma, hedef düğüme ulaşana kadar düğümleri genişletir ve en kısa yol rotasını belirler. Şekil 5.1, A* algoritması kullanılarak başlangıç noktasından hedefe nasıl bir rota çizildiğini göstermektedir.



Şekil 5.1. A* ile uygun yolu bulmak.

Algoritma başlangıç noktasından hedefe en kısa yolu bulduktan sonra ızgara haritasında bu rotayı çiziyor ve robot, o rotanın üzerine hareket etmeye başlar

Robot, elde edilen rota üzerinde hareket ederek hedefe en kısa ve etkili yoldan ulaşabilir, karmaşık ortamlarda bile hızlı bir şekilde hareket edebilir. Şekil 5.2. robotun hedefe ulaşmasını göstermektedir.



Şekil 5.2. TurtleBot3'ün hedefe ulaşması.

6. SONUÇ VE ÖNERİLER

Bu çalışma kapsamında, TurtleBot3 platformundaki SLAM, navigasyon ve otonom sürüş yetenekleri incelenip değerlendirildi. Gerçek zamanlı ve simülasyon ortamında gerçekleştirilen deneyler, TurtleBot3'ün başarılı bir şekilde harita oluşturabildiğini, hedefi doğru bir şekilde konumlandırabildiğini ve yönlendirebildiğini gösterdi. Ayrıca, önerilen otonom sürüş yöntemleri sayesinde TurtleBot3, engelleri tespit edebilir, aşabilir ve hedefe etkili bir şekilde ulaşabilir hale geldi. İki önemli yol planlama algoritması olan DWA ve A* algoritmalarının performansı, Map (A) ve Map (B) olmak üzere iki farklı senaryolarda karşılaştırılacak ve analiz edilecek. Değerlendirme, başlangıç ve hedef koordinatları (X,Y), seyahat süresi ve algoritma koşturma süresi gibi temel parametreleri içerecek ve bu parametreler, algoritmaların karmaşık ortamlarda verimliliğini ve uygunluğunu kapsamlı bir şekilde gösterecek. Z koordinatı, robotun düz bir yüzeyde çalıştığı göz önüne alınmaksızın değerlendirmeye alınmadı. Tablo I'de, Map (A) üzerinde gerçekleştirilen navigasyon deneylerinin sonuçları sergilenmektedir. Navigasyon görevleri, DWA ve A* algoritmaları kullanılarak bağımsız olarak gerçekleştirildi ve bu şekilde algoritmaların Map (A) üzerindeki performanslarının karşılaştırmalı bir analizi sunuldu.

Tablo 6.1. (A) haritasında DAWA ve A*'nın performans karşılaştırması

Harita	Algoritma	Başlangıç (x,y)	Hedef (x,y)	Seyahat Süresi	Koşturma Süresi
A	DWA	0.05, 0.01	4.96, 0.99	29s	-
A	A*	0.05, 0.01	5.0, 0.97	26s	49s

Map (B), Map (A)'ya kıyasla daha yüksek bir karmaşıklık düzeyi sunmaktadır. Tablo II, Map (B) üzerinde gerçekleştirdiğimiz navigasyon deneylerinin sonuçlarını temsil etmektedir.

Tablo 6.2. (B) haritasında DAWA ve A*'nın performans karşılaştırması

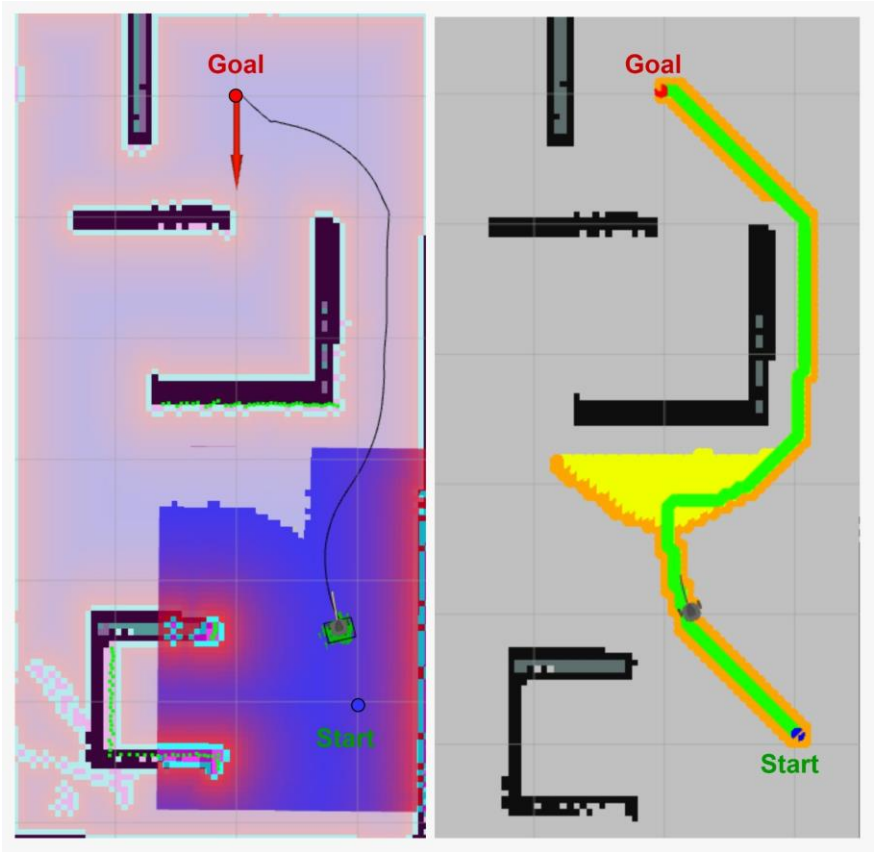
Harita	Algoritma	Başlangıç (x,y)	Hedef (x,y)	Seyahat Süresi	Koşturma Süresi
B	DWA	-1.78, 0.47	5.01, 0.95	39s	-
B	A*	-1.76, 0.55	5.07, 0.77	47.54s	102.46s
B	DWA	5.01, 0.95	-1.78, 0.47	61s	-
B	A*	4.99, 1.00	-1.80, 0.50	49.67s	50.33s

DWA için koşturma süresi tablolarda belirtilmemiştir; çünkü DWA en kısa yolu hesaplamaya çalışmıyor ve bu nedenle bir koşturma süresi gerektirmez.

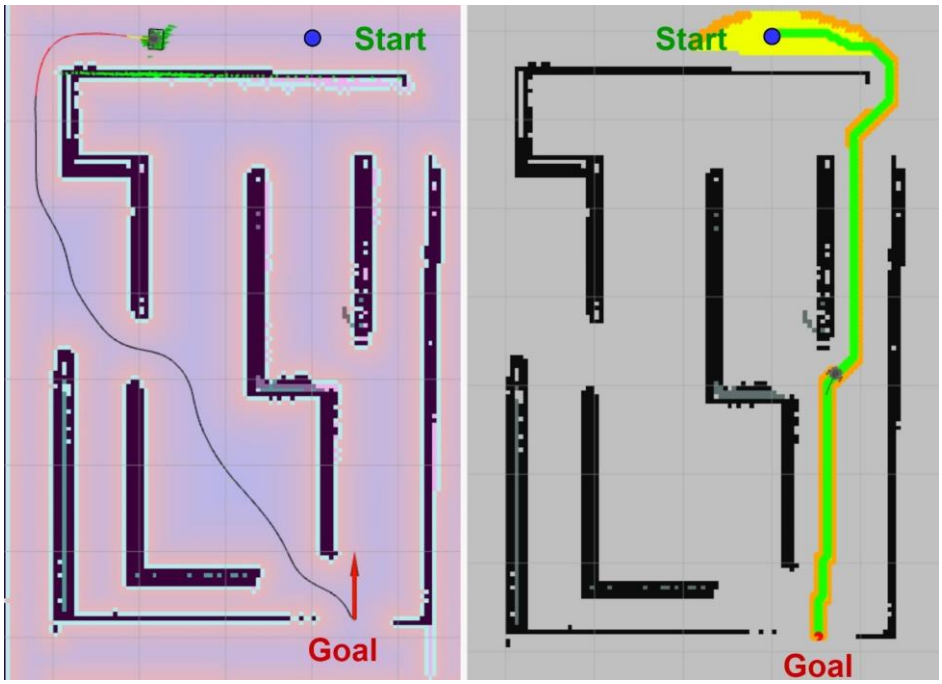
Map (B) durumunda, TurtleBot3'ün A* algoritması için başlangıç ve hedef koordinatlarını ROS üzerinde "feedback" konusu altında gözlemleyebiliriz. Şekil 6.1 feedback konusunun içeriğini göstermektedir.

Şekil 6.1. Map (B) durumunda feedback konusu – A*.

Şekil 6.2. ve Şekil 6.3.de her iki harita için yol planlama sonuçları sunulmaktadır.



Şekil 6.2. (A) haritasında navigasyon – sol: DWA, sağ: A*.



Şekil 6.3. (B) haritasında navigasyon – sol: DWA, sağ: A*.

Performans karşılaştırma tabloları ve oluşturulan yollar temel alındığında aşağıdaki gibi birkaç sonuç çıkarılabilir:

Map (A):

- Hem DWA hem de A* algoritmaları, Map (A) içindeki başlangıç noktasından hedef noktasına başarıyla navigasyon gerçekleştirdi.
- DWA algoritması, A*'a kıyasla hafifçe daha uzun seyahat süreleri sergiledi.
- DWA algoritması, A*'a kıyasla hafifçe daha uzun yollar sergiledi, çünkü A* genellikle nesnelerin kenarlarına daha yakın yollar buldu. Bu, A*'ın potansiyel olarak daha kısa ve engellere daha yakın yol üzerinde hareket etme eğilimini gösterir.
- A* algoritması yolun gerçekleştirilmesi için zaman gerektirir ve koşturma süresi yolun karmaşıklığına bağlı olarak değişir.

Map (B):

- Map (B), Map (A) ile karşılaştırıldığında daha yüksek bir karmaşıklık düzeyi sunar, her iki algoritma için artan seyahat süreleri ve değişen koşturma süreleri gözlemlendi.
- DWA algoritması, her iki denemede de A*'a kıyasla daha uzun seyahat süreleri sergiledi, bu da karmaşık ortamları işlemedeki potansiyel sınırlamaları gösterebilir.
- A* algoritması nispeten uzun koşturma süreleri sergilerken, DWA'ya kıyasla daha kısa yollar sunar ve bu nedenle haritalı tabanlı navigasyon senaryolarında önemini vurgular. Koşturma süresi yolun karmaşıklığına bağlı olarak değişir.
- A* en kısa yolu belirlemeyi hedeflerken, DWA en kısa yol olup olmadığına bakılmaksızın hedefe yönelen bir yol belirleme eğilimindedir ve genellikle mutlak en kısa rotanın yerine güvenli ve uygulanabilir yolları önceliklendirir. Bu, Şekil 6.2.'de gösterildiği gibi.

Bu çalışmanın kapsamı içinde bazı sınırlamalar da tespit edildi. Özellikle, sensör hassasiyeti ve çevresel koşulların etkisi gibi faktörlerin detaylı bir şekilde incelenmesi gerekmektedir. Ayrıca, daha karmaşık ve gerçek dünya senaryolarında performans değerlendirmesi yapmak da önemli bir gelişim alanı olarak görülmektedir. Robotun dar alanlardan geçerken düşük hassasiyetli sensörlerin sınırlamaları nedeniyle bazı zorluklarla karşılaşıldı. Örneğin, LiDAR sensörünün düşük hassasiyeti, dar alanlardaki

nesnelerin doğru tespitini engelledi. Daha yüksek hassasiyetli sensörlerin kullanılması, verimli robot hareketi için önemli bir adımdır. Özellikle dar alanlarda, duvarlar, mobilyalar ve diğer engelleri doğru bir şekilde tespit edebilen sensörler tercih edilmektedir.

Sonuç olarak, Turtlebot platformu üzerinde gerçekleştirilen bu çalışma, mobil robotların SLAM, navigasyon ve otonom sürüş yeteneklerinin geliştirilmesine yönelik önemli bir adımdır. Bu çalışmanın sınırlamaları arasında Lidar sensörünün düşük hassasiyeti nedeniyle dar alanlarda verimlilik sorunu yaşanması bulunmaktadır. Gelecekte yapılacak çalışmalarda, daha gelişmiş ve yüksek hassasiyetli sensörlerin kullanılmasıyla bu sorunun aşılması hedeflenir ve bu alandaki zorlukları ele alarak daha gelişmiş ve güvenilir otonom sistemlerin oluşturulmasına katkı sağlaması beklenmektedir.

KAYNAKLAR

[1] Riisgaard, S. Ve Blas, M. R. (2005). *SLAM for Dummies: A Tutorial Approach to Simultaneous Localization and Mapping*. Massachusetts Institute of Technology.

[2] Robotis website (2023, 01 Haziran). Turltebot Overview. https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#overview/ adresinden 01 Haziran 2023 tarihinde alınmıştır.

[3] Ros webiste (2022, 01 Haziran). ROS İntroduction. http://wiki.ros.org/ROS/Introduction/ adresinden 01 Haziran 2023 tarihinde alınmıştır.

[4] The Robotics Back-End website (2021, 05 Ocak) What is ROS? https://roboticsbackend.com/what-is-ros/ adresinden 01 Haziran 2023 tarihinde alınmıştır.

[5] Ros website (2023, 01 Haziran). ROS Concepts? http://wiki.ros.org/ROS/Concepts/ adresinden 01 Haziran 2023 tarihinde alınmıştır.

[6] Gazebo website (2020, 01 Temmuz). Gazebo Tutorials. https://classic.gazebosim.org/tutorials/ adresinden 01 Haziran 2023 tarihinde alınmıştır.

[7] Medium website (2023, 01 Haziran). Gazebo Nedir? ROS Nedir? https://cennttceylnn.medium.com/gazebo-nedir-ros-nedir-19983c017818 adresinden 01 Haziran 2023 tarihinde alınmıştır.

[8] Siegwart, R. ve Nourbakhsh, I. R. (2004). Introduction to autonomous mobile robots. MIT press.

[9] Zhang, J. (2021) AI based Algorithms of Path Planning, Navigation and Control for Mobile Ground Robots and UAVs. https://doi.org/10.48550/arXiv.2110.00910

[10] SAE website (2021, 4 Mayıs) SAE Levels of Driving Automation https://www.sae.org/blog/sae-j3016-update/ adresinden 01 Haziran 2023 tarihinde alınmıştır.

[11] Medium (2019, 01 Şubat) SLAM Algoritması Nedir? https://medium.com/@sadikkaplan/slam-algoritmas%C4%B1-nedir-a19fa11b5fbc/ 01 Haziran 2023 tarihinde alınmıştır.

[12] Math Works (2019, 05 Ocak) SLAM (Simultaneous Localization and Mapping) https://www.mathworks.com/discovery/slam.html/ 01 Haziran 2023 tarihinde alınmıştır.

[13] Merdan, M., Lepuschitz, W., Koppensteiner, G., Balogh, R., ve Obdržálek, D. (Eds.). (2020). Robotics in Education. Advances in Intelligent Systems and Computing. doi:10.1007/978-3-030-26945-6

[14] Wikipedia website (2015, 28 Eylül) Dead reckoning https://en.wikipedia.org/wiki/Dead_reckoning/ 01 Haziran 2023 tarihinde alınmıştır.

[15] Medium website (2020, 15 Niasn) Kalman filter nedir?https://medium.com/@syndrome/kalman-filter-nedir-51c38a12c423/ adresinden 01 Haziran 2023 tarihinde alınmıştır.

[16] Cui, S.-G., Wang, H., and Yang, L. (2012). A simulation study of a-star algorithm for robot path planning. In 16th international conference on mechatronics technology, PP, pages 506–510

[17] Lima, J., Costa, P., Costa, P., Eckert, L., Piardi, L., Moreira, A. Paulo. ve Nakano, A (2019). A* search algorithm optimization path planning in mobile robots scenarios. https://doi.org/10.1063/1.5114223

[18] Geekforgeeg website (2020, 15 Şubat) A* Search Algorithm https://www.geeksforgeeks.org/a-search-algorithm/ adresinden 01 Haziran 2023 tarihinde alınmıştır.

[19] Brilliant org website (2020, 15 Ocak) A* Search. https://brilliant.org/wiki/a-star-search/ adresinden 01 Haziran 2023 tarihinde alınmıştır.

[20] E. W. Dijkstra et al., “A note on two problems in connexion with graphs,” Numerische mathematik, vol. 1, no. 1, pp. 269–271, 1959

[21] Robits website (2023, 01 Haziran) Navigation. https://emanual.robotis.com/docs/en/platform/turtlebot3/navigation/#navigation/ adresinden 01 Haziran 2023 tarihinde alınmıştır.

ÖZGEÇMİŞ

Ad-Soyad : İhsan ÇUBUKÇU

ÖĞRENİM DURUMU:

- **Lisans :** 2019, Karabük Üniversitesi, Mühendislik Fakültesi, Elektrik Elektronik Mühendisliği Bölümü
- **Yükseklisans :** 2023, Sakarya Üniversitesi, Elektrik-Elektronik Mühendisliği Anabilim Dalı Anabilim Dalı, Elektrik-Elektronik Mühendisliği Programı

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2019 yılında Karabük Üniversitesi Onur Öğrenci derecesi ile mezun oldu.
- 2021 yılında Leo Drive Teknoloji şirketinde Teknik Satış Mühendisi olarak çalışmaya başladı.