KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

YÜKSEK LİSANS TEZİ

STATİK VE DİNAMİK ORTAMLAR İÇİN SLAM VE NAVİGASYON ALGORİTMALARININ ROS VE ROS 2'DE KARŞILAŞTIRMALI ANALİZİ

RAGHAD MANDO

KOCAELİ 2023

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

STATİK VE DİNAMİK ORTAMLAR İÇİN SLAM VE NAVİGASYON ALGORİTMALARININ ROS VE ROS 2'DE KARŞILAŞTIRMALI ANALİZİ

RAGHAD MANDO

Dr. Öğr. Üyesi Alpaslan Burak İNNER		
Danışman,	**Kocaeli Üniv.**	**.....**
Prof. Dr. Hasan OCAK		
Jüri Üyesi,	**Kocaeli Üniv.**	**.....**
Doç. Dr. Adem TUNCER		
Jüri Üyesi,	**Yalova Üniv.**	**.....**

Tezin Savunulduğu Tarih: 19.06.2023

ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ

Kocaeli Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez/proje çalışmasında,

- Bu tezin/projenin bana ait, özgün bir çalışma olduğunu,
- Çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı,
- Bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi,
- Bu çalışmanın Kocaeli Üniversitesi'nin abone olduğu intihal yazılım programı kullanılarak Fen Bilimleri Enstitüsü'nün belirlemiş olduğu ölçütlere uygun olduğunu,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Tezin/Projenin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez/proje çalışması olarak sunmadığımı,

beyan ederim.

[x] Bu tez/proje çalışmasının herhangi bir aşaması hiçbir kurum/kuruluş tarafından maddi/alt yapı desteği ile desteklenmemiştir.

[ ] Bu tez/proje çalışması kapsamında üretilen veri ve bilgiler ..... tarafından ..... no'lu proje kapsamında maddi/alt yapı desteği alınarak gerçekleştirilmiştir.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

.....
Raghad MANDO

YAYIMLAMA VE FIKRI MÜLKIYET HAKLARI

Fen Bilimleri Enstitüsü tarafından onaylanan lisansüstü tezimin/projemin tamamını veya herhangi bir kısmını, basılı ve elektronik formatta arşivleme ve aşağıda belirtilen koşullarla kullanıma açma izninin Kocaeli Üniversitesi'ne verdiğimi beyan ederim. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet hakları bende kalacak, tezimin/projemin tamamının ya da bir bölümünün gelecekteki makale, kitap, tebliğ, lisans, patent gibi çalışmalarda kullanımı, danışmanımın isim hakkı saklı kalmak koşuluyla ve her iki tarafın bilgisi dâhilinde bana ait olacaktır. Tezin/projenin kendi özgün çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin/projenin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanılması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim. Yükseköğretim kurulu tarafından yayınlanan "Lisanüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge" kapsamında tezim aşağıda belirtilen koşullar haricinde YÖK Ulusal Tez Merkezi/ Kocaeli Üniversitesi Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

☐ Enstitü yönetim kurulu kararı ile tezimin/projemin erişime açılması mezuniyet tarihinden itibaren 2 yıl ertelenmiştir.

☐ Enstitü yönetim kurulu gerekçeli kararı ile tezimin/projemin erişime açılması mezuniyet tarihinden itibaren 6 ay ertelenmiştir.

☒ Tezim/projem ile ilgili gizlilik kararı verilmemiştir

.....

Raghad MANDO

ÖNSÖZ VE TEŞEKKÜR

Tez çalışmanın gerçekleştirilmesinde, değerli bilgilerini benimle paylaşan, kendisine ne zaman danışsam bana kıymetli zamanını ayırıp sabırla ve büyük bir ilgiyle bana faydalı olabilmek için elinden gelenden fazlasını sunan her sorun yaşadığımda yanına çekinmeden gidebildiğim, güler yüzünü ve samimiyetini benden esirgemeyen ve gelecekteki mesleki hayatımda da bana verdiği değerli bilgilerden faydalanacağımı düşündüğüm kıymetli ve danışman hoca statüsünü hakkıyla yerine getiren, bilgi, deneyim ve görüşleri ile çalışmalarıma katkıda bulunan, Danışmanım Dr. A. Burak İNNER'e teşekkürü bir borç biliyor ve şükranlarımı sunuyorum. Teşekkürlerin az kalacağı diğer üniversite hocalarımın da bana kazandırdıkları her şey için ve beni gelecekte söz sahibi yapacak bilgilerle donattıkları için hepsine teker teker teşekkürlerimi sunuyorum. Benden hiçbir zaman desteğini esirgemeyen bu hayattaki en büyük şansım olan nişanlım Hüseyin sonsuz teşekkürler.

Ocak – 2023 Raghad MANDO

İÇİNDEKİLER

ETİK BEYAN VE ARAŞTIRMA FONU DESTEĞİ i
YAYIMLAMA VE FIKRI MÜLKIYET HAKLARI ii
ÖNSÖZ VE TEŞEKKÜR iii
İÇİNDEKİLER iv
ŞEKİLLER DİZİNİ vi
TABLOLAR DİZİNİ vii
SİMGELER VE KISALTMALAR DİZİNİ viii
ÖZET ix
ABSTRACT x
1. GİRİŞ 1
2. GENEL BİLGİLER 3
2.1. Problemin Tanımı 3
2.2. Mobil Robotik Sistemler 4
2.3. Robot İşletim Sistemi (ROS) 5
2.4. Navigasyon Yığın Sistemi 6
2.4.1. Harita 6
2.4.2. ROS Harita Sunucusu Düğümü 7
2.4.3. Sensör Entegrasyonu ve Navigasyon 7
2.4.4. Sensör Kaynaklarının Seçimi 8
2.4.5. Dönüşüm Çerçeveleri (TF) 8
2.4.6. Yerelleştirme 8
2.4.7. Adaptive Monte Carlo Localization (AMCL) 8
2.4.8. Eş Zamanlı Lokalizasyon ve Haritalama (SLAM) 9
2.4.9. Move-base Düğümü 9
2.4.10. Maliyet Haritası 9
2.4.11. Navigasyon Planlayıcıları 10
2.5. SLAM Yöntemleri 10
2.5.1. Gmapping 11
2.6. Döngü Kapanımı Problemi 12
2.7. SLAM Toolbox 13
2.7.1. Cartograph 14
2.7.2. Karto-SLAM 15
2.7.3. Core-SLAM 16
2.7.4. Lago-SLAM 16
2.8. Robot Yol Planlaması 16
2.9. Robot Navigasyonu 19
2.9.1. Harita Tabanlı Navigasyon 19
2.9.1.1. Yerelleştirme 19
2.9.1.2. Haritalama 20
2.9.2. Davranış Temelli Navigasyon 20
2.9.3. Öğrenme Temelli Navigasyon 21
2.9.4. İletişim Temelli Navigasyon 21
2.9.5. Geleneksel Yapay Zeka 22
2.9.6. Yumuşak Hesaplama 23
2.9.6.1. Bulanık Mantık Sistemleri 23
2.9.6.2. Nöral Ağlar (NNS) 24

2.9.6.3. Genetik Algoritma (GA) 24
2.9.6.4. Sürü Zekası 24
2.10. ROS Dağıtımları 26
2.11. Ros1 Ve Ros2 Karşılaştırması 27
2.11.1. Platformlar, C++ and Python 27
2.11.2. Ara Yazılım 27
2.11.3. Süre ve Zaman Türlerini Birleştirin 27
2.11.4. Yaşam Döngüsüne Sahip Bileşenler 27
2.11.5. İş parçacık Modeli 27
2.11.6. Birden Fazla Düğüm 28
2.11.7. Roslaunch 28
2.11.8. ROS1 VS ROS2 Yazma Düğümleri 28
2.11.9. Düğümler 28
2.11.9.1. Düğüm Yönetimi 31
2.11.10. ROS2'de Sunulan Yenilikler 32
2.12. Veri Dağıtım Hizmeti (DDS) 34
2.13. Navigasyon2 Tasarımı 36
3. İLGİLİ ÇALIŞMALAR 38
3.1. Slam Karşılaştırma Algoritmaları ile İlgili Çalışma 38
3.2. Yerel Planlayıcı ile İlgili Çalışmalar 39
3.3. Navigasyon Yığınıyla İlgili Çalışma 40
4. DENEYSEL ÇALIŞMALAR VE BULGULAR 42
4.1. Planlanan Yol İle Kat Edilen Yolun Karşılaştırılması 44
4.1.1. Test Durumu 1: Planlanan ve Gerçekleşen Yol Karşılaştırması 46
4.1.2. Test Durumu 2: Planlanan ve Gerçekleşen Yol Karşılaştırması 47
4.1.3. Test Durumu 3: Planlanan ve Gerçekleşen Yol Karşılaştırması 49
4.1.4. Karşılaştırmanın Sonucu 51
4.2. AMR İle SLAM Algoritmaları Karşılaştırması 54
4.2.1. Bulguların Tartışılması 57
4.2.1.1. SLAM Toolbox Algoritma Deneyleri 57
4.2.1.2. Gmapping Algoritma Deneyleri 58
4.2.1.3. Karto SLAM Algoritma Deneyleri 58
4.2.1.4. Hector SLAM Algoritma Deneyleri 59
4.2.1.5. Karşılaştırmanın Sonucu 60
4.3. ROS1 ve ROS2 SLAM Karşılaştırması: Turtlebot3 Simülasyonu 62
4.3.1. Robot ve Çevre 63
4.3.1.1. Kullanılan Donanımlar Ve Teknolojiler 64
4.3.2. Harita Oluşturma 66
4.3.2.1. Gmapping (ROS NavStack) 66
4.3.2.2. SLAM Toolbox Nav2Stack 68
5. SONUÇLAR VE ÖNERİLER 72
KAYNAKLAR 74
ÖZGEÇMİŞ 78

ŞEKİLLER DİZİNİ

Şekil 2.1. Robotların Gör-Düşün-Harekete Geçir Döngüsü 4
Şekil 2.2. Gezinme yığını bileşenlerine genel bakış 6
Şekil 2.3. Harita sunucusu düğümünün çıktısına örnek 7
Şekil 2.4. SLAM Modülleri 11
Şekil 2.5. Google Haritalar 15
Şekil 2.6. Yol planlama Metodolojileri 18
Şekil 2.7. ROS1/ROS2 mimarisi 32
Şekil 2.8. ROS1/ROS2 Mimarisi 34
Şekil 2.9. Veri merkezli yayınla-abone ol (DCPS) modeli 36
Şekil 2.10. DDS QoS Politikası 37
Şekil 4.1. Planlı yol ve gezilen yol Çardak Ortamı 44
Şekil 4.2. Planlanan yol ve seyahat edilen yol haritası 44
Şekil 4.3. Planlanan Yol ve Kat Edilen Yol: Engelsiz Harita Hedefleri 46
Şekil 4.4. Yol Karşılaştırması - Engelsiz Harita - Navstack Dwa 46
Şekil 4.5. Yol Karşılaştırması - Engelsiz Harita - Navs2tack Dwb 47
Şekil 4.6. Planlanan Yol ve Kat Edilen Yol: Harita Hedefleri (2 Engelle) 48
Şekil 4.7. Yol Karşılaştırması- 2 Engelli Harita - Navstack Dwa 48
Şekil 4.8. Yol Karşılaştırması- 2 Engelli Harita - Nav2stack Dwb 49
Şekil 4.9. Planlanan Yol ve Kat Edilen Yol: 5 Engelli Harita Hedefleri 49
Şekil 4.10. Yol Karşılaştırması- 5 Engelli Harita - NAVSTACK DWA 50
Şekil 4.11. Yol Karşılaştırması - 5 Engelli Harita - Nav2stack Dwb 50
Şekil 4.12. İki adet 2D lazer tarayıcıya sahip Otonom Mobil Robot (AMR) 54
Şekil 4.13. Gerçek Dünya Ortamı 55
Şekil 4.14. Gerçek Dünya Çevre Tasarımı 57
Şekil 4.15. SLAM Toolbox ile oluşturulan harita 58
Şekil 4.16. Gmapping ile oluşturulan harita 58
Şekil 4.17. Karto SLAM ile oluşturulan harita 59
Şekil 4.18. Hector SLAM ile oluşturulan harita 59
Şekil 4.19. SLAM Algoritmaları MSE karşılaştırması 61
Şekil 4.20. SLAM Algoritmaları RMSE karşılaştırması 61
Şekil 4.21. Turtlebot3 64
Şekil 4.22. Büyük hastane katının çardak modeli 64
Şekil 4.23. Küçük fabrika katının çardak modeli 65
Şekil 4.24. Küçük ev katının çardak modeli 65
Şekil 4.25. Gmapping Kullanarak Büyük Hastane Haritalama 66
Şekil 4.26. Gmapping Kullanarak Küçük fabrika Haritalama 67
Şekil 4.27. Gmapping Kullanarak Küçük ev Haritalama 67
Şekil 4.28. SlamToolbox Kullanarak Büyük Hastane Haritalama 69
Şekil 4.29. SlamToolbox Kullanarak Büyük Hastane Haritalama 69
Şekil 4.30. SlamToolbox Kullanarak Küçük ev Haritalama 70
Şekil 4.31. Slam Algoritmaları MSE ve RMSE Karşılaştırması 71

TABLOLAR DİZİNİ

Tablo 2.1. Robot navigasyon problemi soruları 20
Tablo 2.2. ROS 1 Dağıtımları arasında karşılaştırma 26
Tablo 4.1. Zaman Tablosu - Navstack Dwa 53
Tablo 4.2. Zaman Tablosu - Nav2stack Dwb 53
Tablo 4.3. Deneysel ölçüm bilgilerinin harita karşılaştırması 62
Tablo 4.4. Lazer Mesafe Sensörü LDS-01 64
Tablo 4.5. GMapping parametreleri 67
Tablo 4.6. GMapping ve SLAM Toolbox Performansı Karşılaştırması 70

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar

AMCL	: Adaptive Monte Carlo Localization (Adaptif Monte Carlo Lokalizasyonu)
API	: Application Programming Interface (Uygulama Programlama Arayüzü)
DDS	: Data Distribution Service (Veri Dağıtım Hizmeti)
DCPS	: Data-centric Publish-Subscribe (Veri merkezli yayınla-abone ol)
GA	: Genetic Algorithm (Genetik Algoritma)
IP	: Internet Protocol Address (İnternet Protokolü Adresi)
LAGO	: Linear Approximation for Graph Optimization Yaklaşım (Grafik Optimizasyonu için Doğrusal)
LS-SVM	: Least square support vector machine (En Küçük Kareler Destek Vektör Makineleri)
LIDAR	: Light Detection and Ranging (Işık Algılama ve Uzaklaştırma)
NNS	: Neural Networks (Nöral Ağlar)
OGM	: Occupancy Grid Map (Doluluk Tablosu Haritası)
RL	: Reinforcement Learning (Pekiştirmeli Öğrenme)
RMSE	: Root Mean Squared Error (Hata Kareler Ortalamasının Karekökü)
ROI	: Region of Interest (İlgi Alanı)
ROS	: Robot Operating System (Robot İşletim Sistemi)
SLAM	: Simultaneous Localization and Mapping (Eş Zamanlı Haritalama ve Lokalizasyon Problemi)
SIR	: Sampling Importance Resampling (Örnekleme Önemi Yeniden Örnekleme)
SAM	: Spectral Angle Mapper (Spektral Açı Haritalama)
SELU	: Scaled Exponential Linear Unit (Ölçekli Üstel Doğrusal Birim)
SNV	: Standard Normal Variate (Standart Normal Dağılım)
SVC	: Support Vector Classifier (Destek Vektör Sınıflayıcı)
SVD	: Singular Value Decomposition (Tekil Değer Ayrışımı)
TF	: Transform (Dönüşüm)
YZ	: Yapay Zeka

STATİK VE DİNAMİK ORTAMLAR İÇİN SLAM VE NAVİGASYON ALGORİTMALARININ ROS VE ROS 2'DE KARŞILAŞTIRMALI ANALİZİ

ÖZET

Teknolojik gelişmelere bağlı olarak, gerçekleştirdikleri görevler ve sağladıkları hizmetler nedeniyle mobil robotlara duyulan ihtiyaç artmıştır. Şirketler, endüstriler, hastaneler, kurumlar, tarım ve evler gibi farklı sektörlerde hizmetleri ve günlük faaliyetleri iyileştirmek için otonom mobil robotların kullanımı her geçen gün artmaktadır. Mobil robotlara olan talebin artması ve uygulama alanlarının çok farklı olması nedeniyle mobil robotların navigasyon algoritmalarının performansını artırmaya yönelik çalışmalara ihtiyaç duyulmaktadır. Mobil robot navigasyonunun temel amacı, karışık ortamlarda mobil robotların başlangıç konumundan hedef konuma güvenli bir şekilde seyretmesidir. Güvenli yolu takip ederek ve engellerden kaçınarak en uygun rotayı planlamaya devam etmek gerekir. Global yol planlayıcı mevcut bir harita üzerinde yol planlaması yapar ve bu yol takip edilirken yerel yol planlayıcı engellerden kaçınmak için yol planlaması yapar ve bu yola uygun hareket edilmesini sağlar. Bu çalışmanın temel amacı, sanal diferansiyel sürücülü bir robotun planlanan yol ile kat edilen yolu kapsayan yol planlamasını karşılaştırmaktır. Turtlebot3, sabit ve dinamik engellerin bulunduğu küçük bir ev ortamında otonom olarak gezdirilmiştir. En uygun SLAM algoritmasının seçilmesi, nihai haritanın sonucuna bağlı olduğu için kritik önem taşımaktadır. SLAM Toolbox, Gmapping, Hector SLAM ve Karto-SLAM metodolojilerinin özel bir ortamda gerçek dünya testleri yoluyla karşılaştırılması, SLAM algoritmalarının çıktı haritalarının, avantajlarının ve dezavantajlarının tartışılması yapılmıştır. ROS 2 dağıtımıyla sunulan Nav2'nin ROS 1 dağıtımı içindeki Navigasyon Yığınına kıyasla farklılıklarını, zayıflıklarını ve güçlü yanlarını bulmak ve Nav2'de nelerin iyileştirildiğini göstermek için büyük bir fabrika ortamında deneyler yapıldı.

Anahtar Kelimeler: Gmapping, NAV2 ,NavStack, Robot İşletim Sistemi, SLAM.

COMPARATIVE ANALYSIS OF SLAM AND NAVIGATION ALGORITHMS FOR STATIC AND DYNAMIC ENVIRONMENTS IN ROS AND ROS 2

ABSTRACT

Depending on technological developments, the need for mobile robots has increased due to the tasks they perform and the services they provide. The use of autonomous mobile robots is increasing day by day to improve services and daily activities in different sectors such as companies, industries, hospitals, institutions, agriculture, and homes. Due to the increasing demand for mobile robots and the wide variety of application areas, there is a need to improve the performance of mobile robot navigation algorithms. The main purpose of mobile robot navigation is to safely navigate mobile robots from the starting position to the target position in complex environments. It is necessary to keep planning the optimal route by following the safe path and avoiding obstacles. A global path planner plans a path on an existing map and follows this path, while a local path planner plans a path to avoid obstacles and follows this path. The main objective of this study is to compare the path planning of a robot with a virtual differential drive between the planned path and the traveled path. Turtlebot3 was autonomously navigated in a small home environment with fixed and dynamic obstacles. Choosing the most suitable SLAM algorithm is critical as it is based on the result of the final map. Comparison of SLAM Toolbox, Gmapping, Hector SLAM and Karto-SLAM methodologies through real-world tests in a private environment, output maps, advantages and disadvantages of SLAM algorithms are discussed. Experiments were conducted in a large factory environment to find the differences, weaknesses, and strengths of Nav2 in the ROS 2 distribution compared to the Navigation Stack in the ROS 1 distribution and to show what has been improved in Nav2.

Keywords: Gmapping, NAV2, NavStack, Robotic Operating System, SLAM.

1. GİRİŞ

Robotlar, çalışma hayatımızın her alanında insanların yerini alabilir mi? Bu soru, uzun yıllardır bilimkurgu hikayelerinin ana temasını oluşturmuştur. Ancak, robotların birçok sektöre pratik katkılar sağlamaya başlamasıyla, bilimkurgu yazarları da öngörülerini yeniden şekillendirmeye başlamışlardır. Teknolojinin ilerlemesiyle birlikte, birçok işi robotlar daha verimli bir şekilde gerçekleştirebilir hale gelmiştir. Teknolojik gelişmeler sayesinde, mobil robotların sunduğu hizmetler ve gerçekleştirdiği görevler nedeniyle bu robotlara olan ihtiyaç artmıştır. Şirketlerden endüstriyel kuruluşlara, hastanelerden tarım sektörüne ve ev işlerine kadar birçok alanda, otonom mobil robotların kullanımı giderek yaygınlaşmıştır.

Robotlar, mekanik sistemlerle birlikte kontrol, algılama sistemleri ve bilgisayar algoritmalarına dayanarak akıllıca hareket eden makinelerdir. Bu makineler, programlanmış hareketleri kullanarak malzemeleri, parçaları ve araçları taşıyabilir veya işleyebilir. Ayrıca, bu robotlar çok işlevli olup yeniden programlanabilirler. Otonom robotlar, çevrelerini algılama yeteneğine sahip olup, bu algılamalara dayanarak kararlar alabilirler. Bu robotlar, belirli bir ortamda hareketi tanımlamak, başlatmak ve tamamlamak için programlanmışlardır. Otonom mobil robotlar ise çevrelerini anlama ve bağımsız hareket etme yeteneğine sahip özel bir robot türüdür. Özellikle, otonom mobil robotların navigasyonu son yıllarda büyük bir ilerleme kaydetmiştir ve sürücüsüz arabalar, depo robotları, kişisel asistan robotlar gibi alanlarda kritik bir öneme sahiptir.

Robotlar, beklenmedik engellerle karşılaşabileceği kapalı ortamlarda, özellikle insanlar için tasarlanmış alanlarda, daha aktif olarak çalışmaya başlamışlardır. Otonom mobil robot navigasyonunda üç ana sorun bulunmaktadır: lokalizasyon, yol planlama ve hareket kontrolü. Lokalizasyon, robotun bulunduğu konumu belirlemekle ilgilidir. Yol planlama, robotun belirli bir hedefe nasıl ulaşacağını belirler. Hareket kontrolü ise, robotun engellere çarpmadan belirlenen yolu nasıl takip edeceğini belirler.

Navigasyon için haritaların oluşturulması gerekmektedir ve bu nedenle haritalama yöntemlerinin incelenmesi önemlidir. Harita oluşturulduktan sonra yol planlaması yapılır, ancak bu planlama haritada olmayan engelleri göz ardı eder. Bu tür engellerle başa çıkmak için yeni yöntemler geliştirilmiştir. Robotların bir alanda bağımsız olarak gezinmesi ve çevreyi incelemesi gerekmektedir. Lokalizasyon ve haritalama için

kullanılan sensörlerin türü, ortamın iç veya dış mekan olmasına göre değişiklik gösterir. Kapalı ortamlar, yoğun bileşenlere sahipken, dış ortamlar daha geniş ve az bileşenli alanlardır. Robotun sensörleri tarafından taranan özellikler, sensör seçiminde önemli bir rol oynamaktadır. Dış ortamlarda, bu özellikler daha seyrektir, bu nedenle robotun konumunu belirlemek ve haritalar oluşturmak için daha doğru bilgilere ihtiyaç duyulmaktadır.

Gerçek dünyada gerçekleştirilen deneyler bazen tehlikeli sonuçlar doğurabilir. Bu riskleri minimize etmek için, deneyler simülasyon ortamlarında, örneğin Gazebo programı kullanılarak, gerçekleştirilebilir. En güvenli ve optimal rotanın belirlenmesi için global veya lokal yol planlayıcılar kullanılarak en uygun yol uzunluğu tespit edilebilir.

Bu çalışma, SLAM algoritması, planlanan ve gerçekte izlenen yollar ile navigasyon konularına odaklanmaktadır. Özellikle, robotik alanda sıkça kullanılan ROS üzerinden farklı SLAM algoritmalarının karşılaştırılması, planlanan ile gerçekte izlenen yolların karşılaştırılması ve nav yığını ile nav2 karşılaştırılması ele alınmıştır. ROS'un ana hedefi, robot yazılımlarının geliştirilmesini daha esnek ve uyumlu hale getirmektir. Bu çerçeve, karmaşık görevleri yerine getirebilen farklı robotik platformlarla etkileşim kurmak için gerekli araçlar, kütüphaneler ve protokoller koleksiyonunu içermektedir.

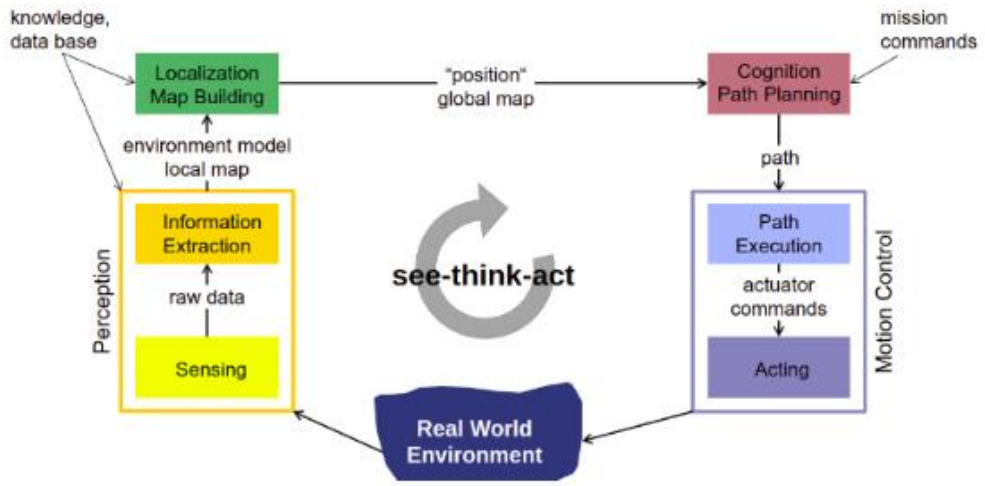
2. GENEL BİLGİLER

2.1. Problemin Tanımı

Bir ortamda hareket etmek, robotik sistemlerin karşılaştığı temel zorluklardan biridir. Robotun bu görevi başarılı bir şekilde yerine getirebilmesi için, nerede olduğunu başlangıç konumu ve nereye gitmesi gerektiğini hedef bilmesi şarttır. Bu, genellikle robotlara bir harita ile birlikte başlangıç ve bitiş konumları sağlanarak gerçekleştirilir (Alatise ve Hancke, 2020). Robotlar, bilinmeyen bir ortamın haritasını oluştururken SLAM yaklaşımını kullanır ve aynı zamanda bu ortamda kendi konumlarını takip ederler. SLAM (Eşzamanlı Lokalizasyon ve Haritalama), robotun bilinmeyen bir alanda hem konumunu belirlemesi hem de bu alana ait bir harita oluşturması sürecidir. SLAM, robotların hareket halindeyken sensör verilerini kullanarak bir ortamın haritasını oluşturmasını ifade eder. Robot hareket halindeyken bir ortamın haritasını çıkarmak, robot bir ortamda gezinirken sensörlerden bilgi toplamayı içeren bir tekniktir. Bu, robotun yalnızca bilinmeyen bir alanın haritasını üretmesini değil, aynı zamanda mevcut haritayı güncellemesini sağlayarak cihazın mobil bir platformla daha genel, değişen ortamlarda kullanılmasına olanak tanır. Eşzamanlı Lokalizasyon ve Haritalama (SLAM), lokalizasyon problemini ele alırken gömülü sensörler kullanarak bilinmeyen bir alan için bir harita oluşturma zorluğunu ifade eder (Holland, 2004). Filtre tabanlı SLAM, Bayes filtreleme prensibine dayanan bu sorunu çözmenin geleneksel bir yoludur. İki ana adımdan oluşur: birincisi, robot lokalizasyonunun ve harita durumunun sistem durumu ve giriş kontrol komutları hakkında önceki bilgiler kullanılarak güncellendiği tahmin adımı ve ikincisi, yeni sistem durumu tahminleri yapmak için mevcut sensör verilerinin tahmin edilen sistem durumu ile karşılaştırıldığı ölçüm güncellemesi. Bir çevre haritası kullanmanın iki nedeni vardır. İlk olarak, harita genellikle diğer işlemlere yardımcı olmak için gereklidir; örneğin, bir harita rota planlamasına rehberlik edebilir veya bir insan operatör için sezgisel görselleştirme sağlayabilir. İkinci olarak, harita robotun durumunun hesaplanmasındaki yanlışlığı sınırlar. Murphy (Grisetti ve diğ, 2005) dinamik bir iç mekân ortamında SLAM problemini çözmek için ilk kez 2000 yılında Rao-Blackwellized parçacık filtresi tekniğini kullanmıştır. Rao-Blackwellized parçacık filtreleri, düzgün bir eşleme etkisi elde etmek için tipik olarak çok sayıda parçacık gerektirir, ancak bu, bilgisayarın karmaşıklığını ve CPU kullanımını artırır. Sonuç olarak, parçacık filtresi algoritmasını

optimize etmeye çalışan akademisyenler için temel zorluk, parçacık sayısının nasıl en aza indirileceğini belirlemektir. Yeni gözlemler girildiğinde, SIR algoritması parçacık ağırlıklarını en baştan yeniden değerlendirmelidir. Yörüngenin uzunluğu zaman içinde arttıkça bu sürecin hesaplama karmaşıklığı da artacaktır. Sonuç olarak, önemli olasılık yoğunluk fonksiyonu, önem ağırlığını hesaplamak için özyinelemeli bir formül elde etmek üzere kısıtlanır. Bir haritalama yöntemi işlendikten sonra, bir 2B harita oluşturulacaktır, haritanın amacına bağlı olarak farklı SLAM algoritmaları mevcuttur. En yaygın kullanılan SLAM algoritmaları Gmapping, Hector-SLAM, SLAMTOOLBOX, Core SLAM ve Cartographer'dır. Gmapping algoritması parçacık filtresi eşleştirme tekniğine, Hector-SLAM tarama eşleştirme algoritmasına, Cartographer döngü kapatma tabanlı bir tarama eşleştirme algoritmasına ve RGB-D bir derinlik görüntüsü eşleştirme algoritmasına dayanmaktadır. Bunlar en yaygın ve en çok kullanılan algoritmalardan bazılarıdır.

2.2. Mobil Robotik Sistemler



Şekil 2.1. Robotların Gör-Düşün-Harekete Geçir Döngüsü (Fawzy, 2019)

Mobil robotik sistemlerin temel işleyişini anlamak için Şekil 2.1'de gösterilen "Gör-Düşün-Harekete" döngüsünü ele almak esastır. Bu döngü, bir mobil robotun etkileşimde bulunduğu çevresiyle olan ilişkisini temsil eder. Robot çevresini algılar, algıladığı çevreye dayanarak hedefine nasıl ulaşacağını düşünür (yani yol planlama) ve ardından düşüncelerine göre eyleme (hareket) geçer. Gör-Düşün-Harekete" döngüsünün ilk aşaması olan algılama, robotun çevresini tanımlaması ve bu bilgileri işlemesi için kritiktir. Bu aşama, robotun genel performansı ve doğruluğu için temel teşkil eder.

Çünkü algılama aşamasında gerçekleşen herhangi bir hata, sonraki işlemlerin ve genel sistem performansının doğruluğunu olumsuz etkileyebilir. Robotlar, çevresel verileri toplamak için bir dizi sensöre ihtiyaç duyar. Bu sensörler, çeşitli fiziksel özellikleri algılamak için tasarlanmıştır ve toplanan veriler, yazılım algoritmaları aracılığıyla işlenir. Özellikle sürücüsüz otomobillerde, çevresel algılama için LiDAR, RGB Kamera, RADAR ve Ultrason gibi bir dizi sensör kullanılır (Fawzy, 2019). Toplanan sensör verileri, anlamlı bilgilere dönüştürülmesi için yazılım algoritmaları tarafından işlenir. Bu algoritmalar, ham sensör verilerini robotun hareket planlaması için kullanılabilecek anlaşılır ve işlenebilir bilgilere dönüştürür. Bu dönüşüm süreci, robotun çevresiyle etkileşimde bulunmasını ve doğru hareket kararları almasını sağlar.

2.3. Robot İşletim Sistemi (ROS)

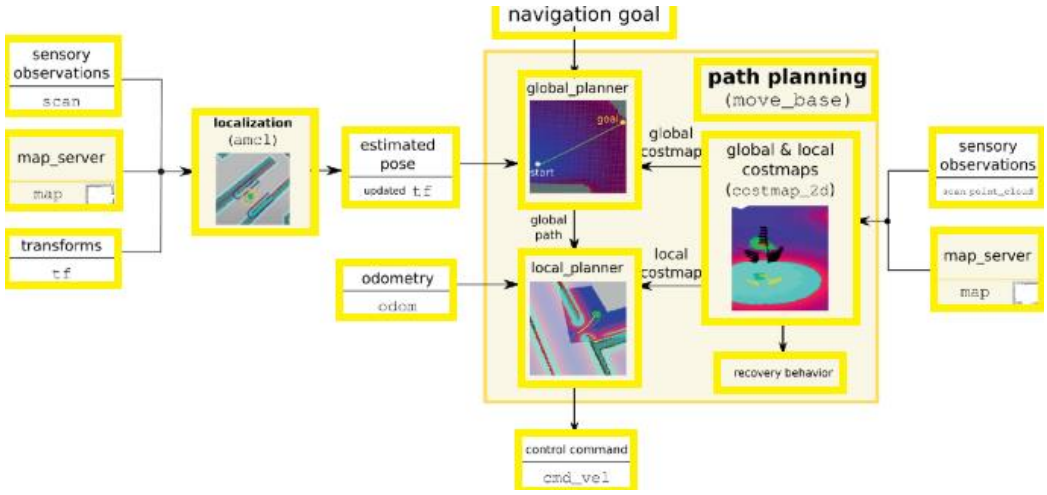
Robotik yazılım geliştirmek için çeşitli araçlar ve kütüphaneler sağlayan esnek bir çerçevedir. Geliştiricilere mesajlaşma, dağıtık hesaplama, kodun yeniden kullanımı ve robotik uygulamalar için son teknoloji algoritmaların uygulanması gibi görevlerde yardımcı olan çeşitli güçlü özellikler sunar. ROS'un temel amacı, özel robotik uygulamalarla kolayca entegre edilebilen modüler yazılım bileşenleri sunmaktır. Bu, robotik uygulamaların standartlaştırılmasına ve genelleştirilmesine olanak tanır (Chemuturi, 2010).

Bir robot için navigasyonun temel yönlerinden biri dünyayı dolaşmaktır. Bunu etkili bir şekilde yapabilmek için robotun nerede olduğunu (başlangıç konumu) ve nereye gitmesi gerektiğini (hedef) bilmesi gerekir. Bu genellikle robota bir dünya haritası ve bir hedef konum ile bir başlangıç pozisyonu sağlanarak gerçekleştirilir. Robotik navigasyon, robotun kendi referans çerçevesindeki konumunu belirleme ve ardından bir hedef konuma doğru bir yol planlama becerisini içerir. Bir görevi yerine getirmek için bir robotun sensör ölçümlerinden çevreyi "anlaması" gerekir. Anlama, doğru soyutlama düzeyinde bilgi edinme anlamına gelir. Otonom olarak navigasyon yapan bir robot, çevrenin neye benzediğini ve harita üzerinde nerede olduğunu bilmelidir. Navigasyonun amacı, robotu takılmadan veya arızalanmadan bir konumdan diğerine güvenli bir şekilde taşımaktır. Mobil bir robotun navigasyonunu gerçekleştirmek için odometri, sensör akışları ve bir hedef pozdan bilgi alan ve bir robota güvenli hız ile ilgili komutlar gönderen ROS Navigasyon Yığınını kullanılmıştır.

2.4. Navigasyon Yığın Sistemi

Navigasyon yığını Şekil 2.2'de gösterilmiştir ve bu paketle iletişim kuran bileşenler sarı renkle vurgulanmıştır. Navigasyon yığınının iki ana modülü vardır:

1. Yerelleştirme: Lokalizasyon modülü, harita sunucusu tarafından sağlanan haritadaki robotun konumunu, harita çerçevesinden temel bağlantı çerçevesine TF'de bir dönüşüm olarak tahmin eder.
2. Yol Planlama: Yol planlama modülü bir navigasyon hedefi alır ve global yol planlaması (global planlayıcı), yerel hız komutu planlaması (yerel planlayıcı) veya kurtarma davranışı planlaması gerçekleştirir. Planlama, duyusal gözlemlerden ve sağlanan haritadan oluşturulan maliyet haritalarına dayalı olarak gerçekleştirilir. (Holland, 2004).



Şekil 2.2. Gezinme yığını bileşenlerine genel bakış (Koubaa ve diğ. 2019)

2.4.1. Harita

Navigasyon yığınını kullanmaya başlamak için robotun bir harita oluşturması veya kaydedilmiş bir haritanın kullanılması gerekir. İlk haritayı oluşturmak için birkaç seçenek vardır; bunlardan biri haritayı harici bir kaynaktan elde etmek veya haritayı oluşturmak için teleoperasyon kullanarak robotun manuel kullanmaktır. ROS teleoperasyon ile robotun uzaktan bir klavye kullanarak manuel olarak hareket ettirme yeteneği sağlar. Ek olarak, Sınır Tabanlı Keşif gibi algoritmaları kullanma seçeneği de vardır. Harita, robotun çalıştığı ortamı temsil eder ve robotun ilgili bir görevi yerine getirmesi için yeterli bilgi içermelidir. Metrik (Izgara Tabanlı, Özellik Tabanlı Hibrit),

Topolojik ve Hibrit gösterimler dahil olmak üzere çevreyi temsil etmenin birçok yolu vardır.

2.4.2. ROS Harita Sunucusu Düğümü

ROS "Harita sunucusu" iki düğümden oluşur: harita sunucusu ve harita kaydedici. İlki, adından da anlaşılacağı gibi statik harita verilerini bir ROS hizmeti olarak sağlayan bir ROS düğümüyken, ikincisi dinamik olarak oluşturulan bir haritayı bir dosyaya kaydeder. Şekil 2.3'de harita kaydedicisi ile kaydedilmiş bir haritanın görüntüsü bulunmaktadır. Buradaki beyaz alanlar boş alanları, siyah noktalar duvarlar vb. engelleri, gri alanlar ile hakkında bir bilgi olmayan yerleri temsil etmektedir.



Şekil 2.3. Harita sunucusu düğümünün çıktısına örnek

2.4.3. Sensör Entegrasyonu ve Navigasyon

Sensör verileri toplanır, daha sonra perspektife yerleştirilir (sensör dönüşümleri düğümü) ve ardından başlangıç konumuna (odometri kaynak düğümü) dayalı olarak robotun konumunun bir tahminiyle birleştirilir. Bu bilgiler "move-base "in yörüngeyi hesaplayabilmesi ve hız komutlarını iletebilmesi için temel kontrol düğümü aracılığıyla yayınlanır. "move-base" düğümü sırasıyla nav-core::BaseGlobalPlanner ve nav-core::BaseLocalPlanner arayüzlerine uyan bir global planlayıcı ve bir yerel planlayıcı başlatır. Ayrıca, planlama sırasında çarpışmadan kaçınma için gerekli olan bir küresel maliyet haritası ve bir yerel maliyet haritası sağlar.

2.4.4. Sensör Kaynaklarının Seçimi

Bu bileşen sensör verisi kaynağının seçilmesinden sorumludur. Burada hem Kinect için kullanılan nokta bulutunu hem de hokuyo veya sick lazer sensörü için kullanılabilecek yorumlanan çizgilerin önerdiği lazer taraması kullanabilir. Lazer türünün yanı sıra, navigasyon yığınının sensör verilerini doğru konumdan alması için abone olunan konunun doğru adını ayarlamak çok önemlidir.

2.4.5. Dönüşüm Çerçeveleri (TF)

"tf", dönüşüm verilerini dağıtık bir sistemde dağıtmak için standartlaştırılmış bir protokoldür ve kullanıcının zaman içinde birden fazla koordinat çerçevesini izlemesine olanak tanır. Ayrıca hesaplamaların bir çerçevede yapılmasını ve daha sonra istenen herhangi bir zamanda başka bir çerçeveye dönüştürülmesini sağlar. Dönüşümler, navigasyon yığınının sensörlerin robotun merkezine (temel bağlantı) göre nerede bulunduğunu anlaması için bir gerekliliktir.

2.4.6. Yerelleştirme

Bir robotun bir hedef konuma nasıl ulaşabileceğini belirlemek için, harita içinde nerede bulunduğunu bilmesi gerekir. ROS navigasyon yığınında robota 2D lazer taramalarına dayalı olarak her bir haritadaki konumunu tahmin etmede yardımcı olmak için uygulanan birçok yerelleştirme algoritması vardır. Entegre ve en yaygın kullanılan paket amcl'dir, ancak robot-localization gibi 3D uzayda durum tahmininde uzmanlaşmış daha gelişmiş paketler de vardır.

2.4.7. Adaptive Monte Carlo Localization (AMCL)

AMCL, bilinen herhangi bir harita üzerinde bir robotun konumunu ve yönünü tahmin etmek için 2B lazer tarayıcı kullanan olasılıklı bir yerelleştirme modülüdür. Daha önce oluşturulmuş statik bir harita aracılığıyla bir robotun kendi konumunu bir ortamda bulmasını sağlayan bir araçtır. Bununla birlikte, bu kaynağın dezavantajı, statik bir haritanın kullanılması nedeniyle, ortamdaki herhangi bir değişikliğin, hesaplama süresi ve çaba gerektiren yeni bir harita oluşturmadan hesaba katılamamasıdır. Robotların endüstriyel veya eğitim ortamları gibi sürekli hareket halinde olan ortamlarda da çalışabilmesi gerektiğinden, sadece değiştirilmemiş ortamlarda gezinmek yeterli değildir. Statik haritaların esnek olmamasının üstesinden gelmek için navigasyon yığını

iki ek lokalizasyon sistemi sağlar: Gmapping ve Hector SLAM. Hem Gmapping hem de Hector SLAM (Simultaneous Localization and Mapping) temeline dayanmaktadır.

2.4.8. Eş Zamanlı Lokalizasyon ve Haritalama (SLAM)

SLAM, bir robotun bilinmeyen bir ortamda hareket ederken eşzamanlı olarak haritalama ve yerelleştirme yapmasını sağlayan bir tekniktir. Bu, robotun hem kendi konumunu belirlemesine hem de çevresini haritalandırmasına olanak tanır. SLAM'ın arkasındaki temel fikir, robotun sensörlerinden gelen veriyi kullanarak, hareket ettiği ortam hakkında sürekli olarak güncellenen bir harita oluşturmasıdır. Bu sensörler genellikle lazer tarayıcılar, kameralar veya sonar gibi cihazları içerir. Bu, özellikle dinamik ortamlarda, nesnelerin ve engellerin sürekli olarak hareket ettiği yerlerde son derece yararlıdır. SLAM'ın bir diğer önemli avantajı, robotun mevcut konumu hakkında bilgi sahibi olmasına gerek kalmadan bir ortamı haritalandırabilmesidir. Bu, robotun başlangıçta nerede olduğunu bilmediği veya GPS gibi dış konumlandırma sistemlerinin kullanılamadığı durumlarda son derece değerlidir. Son olarak, SLAM, robotların daha genel, değişken ve öngörülemeyen ortamlarda güvenli bir şekilde çalışmasına olanak tanır. Bu, özellikle araştırma, kurtarma operasyonları veya endüstriyel otomasyon gibi uygulamalar için kritik öneme sahiptir, çünkü bu tür ortamlar genellikle önceden bilinmez ve sürekli olarak değişir.

2.4.9. Move-base Düğümü

"move-base" düğümü, ROS navigasyon yığınında tüm sihrin gerçekleştiği yerdir. Navigasyon hedefine ulaşmak için global ve yerel bir planlayıcı kullanır. Yol planlama işlemi sırasında çevre hakkında ne kadar bilgi bilindiğine bağlı olarak, yol planlama iki türe ayrılabilir: küresel yol planlama ve yerel yol planlama (Yu ve diğ,2020).

2.4.10. Maliyet Haritası

"Maliyet haritası", bir hücre ızgarasında robotun bulunmasının güvenli olduğu alanları temsil eden bir veri yapısıdır. Tipik olarak, maliyet haritasındaki değerler, boş alanı veya robotun çarpabileceği yerleri temsil eden ikili değerlerdir. Robot harita üzerinde küresel ve yerel olmak üzere iki tür navigasyon ihtiyacı olduğundan küresel ve yerel maliyet haritalarına ihtiyaç duyulur. Küresel maliyet haritası, küresel yol planlaması için kullanılır. Yerel maliyet haritası ise yakın mesafelerde yollar oluşturmak ve engellerden

kaçınmak için kullanılan yerel yol planlamasında kullanılır. Costmap-2d paketi sensör verilerini ve statik haritadaki bilgileri kullanarak 2D veya 3D doluluk ızgarasını oluşturur ve 2D maliyet haritasındaki maliyetleri doluluk ızgarasına ve kullanıcı tanımlı bir şişirme yarıçapına göre şişirir. Bu şişirme işlemi sayesinde robot çarpma tehlikesini en aza indirir.

2.4.11. Navigasyon Planlayıcıları

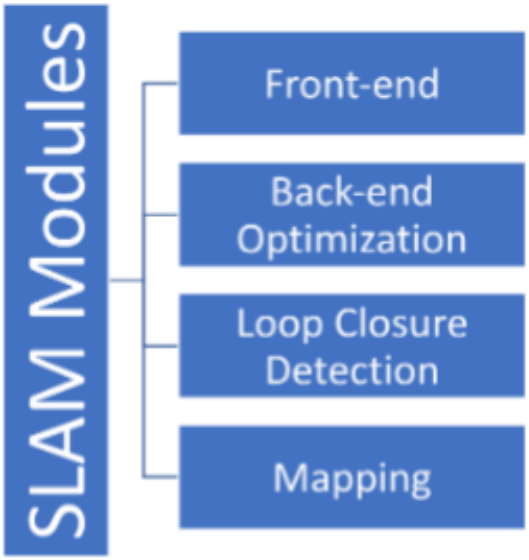
Robot iki tür navigasyon kullanarak harita üzerinde gezinecektir:

1. Küresel planlayıcı, harita üzerinde bir hedef veya uzak bir konum için yol oluşturmakta kullanılır. Küresel yol planlayıcısı, çevre haritasını kullanarak başlangıç konumundan hedef konuma engelsiz bir küresel yol bulmaktan sorumludur. Küresel yol planlayıcıları nav-core::BaseGlobalPlanner arayüzüne uymalıdır. ROS'ta üç yerleşik küresel yol planlayıcısı vardır: (a) Carrot planlayıcı: Kullanıcı tanımlı bir hedef noktasını alan ve robotu mümkün olduğunca yakınına taşımaya çalışan basit bir global planlayıcıdır. Hedef nokta bir engelin içinde olduğunda navigasyon hedefi olarak kullanılabilir. (b) Navfn: herhangi iki konum arasındaki genel yolu bulmak için Dijkstra'nın algoritmasını kullanır. (c) Global planlayıcı: navfn'nin yerini almıştır ve daha esnektir ve daha fazla seçeneğe sahiptir.
2. Yerel planlayıcı, yakın mesafede yollar oluşturmak ve engellerden kaçınmak için kullanılır. Yerel yol planlayıcı, robotun sensörlerini kullanarak küresel yol planlayıcı tarafından belirlenen statik yolu takip ederken robotun yoluna çıkabilecek dinamik engellerden kaçınmaktan sorumludur. Yerel yol planlayıcıları nav-core BaseLocalPlanner arayüzüne uygun olmalıdır.

2.5. SLAM Yöntemleri

Otonom bir robot, çevresinde hareket etmesini sağlayan sensörlerle donatılmış bir cihazdır. Bu hareket bacaklar, tekerlekler, kanatlar veya başka herhangi bir araç kullanılarak gerçekleştirilebilir. Lokalizasyon yapıldığında ve çevre haritası bilinmediğinde veya haritalama yapılacağında ve robotun konumu bilinmediğinde, bu görev SLAM olarak adlandırılır. Biri bilinmezken diğeri tahmin edilemiyorsa hem lokalizasyon hem de haritalama birbirlerine oldukça bağımlıdır. Bu nedenle, her iki bilginin de mevcut olmadığı bir uygulamada, her iki durumun da aynı anda tahmin

edilmesi gerekir. Başarılı bir SLAM sonrasında navigasyon, konumunu ve çevre haritasını bilen bir robot tarafından hedef noktaya ulaşmak için verilen bir dizi hareket kararıdır. Şekil 2.4'de SLAM modülleri gösterilmiştir. Başarılı bir SLAM genellikle daha iyi navigasyona yol açar.



Şekil 2.4. SLAM Modülleri (Silva ve diğ., 2017)

2.5.1. Gmapping

Gmapping (URL – 3) ROS'ta en yaygın kullanılan SLAM paketidir (Silva ve diğ, 2017). SLAM uygulaması, optimizasyon ayarlarını temsil eden Rao-Blackwellized parçacık filtrelerine dayanmaktadır. Giriş verileri odometri ve lazer telemetriden gelmektedir. Tanımlanan düğümün çıktısı, doluluk ızgarası üzerinde taranan engelleri ve boş alanı gösteren bir haritadır. Ayrıca robotun sağlanan haritada tanımlanan konumunu ve yönünü de gösterir. Bu paketin avantajı, çok detaylı bir dokümantasyon sunması ve uzun süredir yaygın olarak kullanılmasıdır. Paketin yapılandırılması ve kullanımı kolaydır. Doğru odometri ile kullanıldığında bu algoritma ile çalışmak çok avantaj sağlayabilir. Odometri bu algoritmanın çalışması için kaçınılmazdır. Bu nedenle, bu paketin kullanımı robotun odometriden veri aldığı durumlarla sınırlıdır. 2007 yılında, en yaygın kullanılan SLAM kütüphanelerinden biri olarak piyasaya sürülmüştür. Gmapping geniş alanlar için pek uygun değildir ve endüstriyel düzeydeki döngüleri doğru bir şekilde tamamlayamaz. AMCL'den farklı olarak, Gmapping bilinmeyen bir ortamda çalışır ve 2B doluluk ızgara haritası (Silva ve diğ, 2017) oluşturmak için robot

pozunu ve lazer verilerini kullanarak Eşzamanlı Konumlandırma ve Haritalama (SLAM) gerçekleştirir. Gmapping paketi, slam-mapping adı verilen bir ROS düğümü olarak lazer tabanlı bir SLAM sağlar. Lazer taramalarını ve odometri verilerini alır ve 2D doluluk ızgara haritası (OGM) oluşturur. Robot hareket ettiğinde veya konumu ve harita hakkında iyi bir tahminde bulunduğunda harita durumunu günceller. slam-Gmapping düğümü sensör-msgs/LaserScan mesajlarını alır ve bir harita oluşturur (nav-msgs/OccupancyGrid). Parçacık filtresi algoritma ailesi tipik olarak iyi sonuçlar elde etmek için yüksek sayıda parçacık gerektirir ve bu da hesaplama karmaşıklığını artırır. Ancak, tarama eşleştirme işlemi nedeniyle belirsizlik daha düşük olduğundan, gerekli parçacık sayısı azalır. Gmapping, belirli eşleme parametrelerini ayarlayarak belirli uygulama ihtiyaçlarını karşılamak üzere eşleme sürecini optimize etmek için esnek bir yol sağlar. (Abdelrasoulve diğ, 2016)

2.6. Döngü Kapanımı Problemi

Döngü kapanımı (loop-closure) problemi, mobil robotlar ve otonom araçlar için harita oluşturma ve konumlandırma süreçlerinde karşılaşılan kritik bir zorluktur. Bu problem, robotun aynı bölgeyi farklı zamanlarda ziyaret edip tekrar aynı konuma geri dönmesiyle ortaya çıkar. Bu tür bir hareket, haritanın tutarlılığını bozabilir, çünkü oluşturulan harita üzerinde bir döngü meydana gelir. Bu problem, özellikle aşağıdaki senaryolarda belirgindir:

- SLAM (Simultaneous Localization and Mapping - Eşzamanlı Konumlandırma ve Haritalama): Robot, çevresini haritalarken kendi konumunu da tahmin eder. Eğer robot aynı bölgeyi farklı açılardan ve farklı zamanlarda ziyaret ederse, SLAM algoritması bu yerler arasındaki ilişkiyi doğru bir şekilde kuramayabilir, bu da harita tutarlılığını bozar.
- Otomatik Sürüş Araçları: Otonom araçlar uzun süreler boyunca dolaşarak aynı alanları birden çok kez geçebilir. Araçların başlangıç noktasına dönmesi ve daha önce ziyaret ettiği yerleri tekrarlaması durumunda loop-closure problemi ortaya çıkabilir.
- Havada Uçan Araçlar: İnsansız hava araçları (drone) harita oluştururken birden fazla tur atarak aynı alanı tarama ihtiyacı duyabilir. Bu durumda, drone'un tam da başlangıç noktasına geri dönmesi ve önceki taramalardaki bilgilerle çelişkili yeni

bilgilerle karşılaşması loop-closure problemini tetikleyebilir. Loop-closure probleminin çözülmesi önemlidir çünkü tutarlı bir harita elde etmek, robotun güvenilir ve doğru bir şekilde hareket etmesi için kritik öneme sahiptir. Çözümler genellikle SLAM algoritmalarının geliştirilmesi veya ekstra sensörlerin kullanılması gibi yöntemlerle elde edilebilir. Bu, robotun geçmiş hareketleri ile yeni algılamalarını doğru bir şekilde eşleştirebilmesini sağlar ve harita tutarlılığını artırarak etkili ve güvenilir navigasyon sağlar.

2.7. SLAM Toolbox

SLAM Toolbox, hem senkron hem de asenkron olmak üzere çoklu haritalama modlarının yanı sıra kinematik harita birleştirme, lokalizasyon modu, çoklu oturum haritalama, gelişmiş grafik optimizasyonu, önemli ölçüde azaltılmış hesaplama süresi ve özel uygulama ihtiyaçlarını karşılamak üzere uyarlanabilen dağıtılmış haritalama uygulamaları gibi yardımcı araçlar sağlar. SLAM Toolbox, genellikle 18.000 metrekareyi aşan alanları mobil CPU'ları kullanarak alanları etkili bir şekilde haritalayabilir ve eğitimsiz teknisyenler veya genellikle robot çözümlerini dağıtmak için kullanılan uzaktan izleme sistemlerinde kullanılabilir. Keşif planlayıcılarıyla birlikte SLAM Toolbox kullanılarak bir alanı otomatik olarak haritalamak için bazı uygulamalar geliştirilmiştir. OpenKarto SLAM kütüphanesinde çeşitli güncellemeler yapılmıştır. Ölçüm eşleştirme yöntemleri, çok iş parçacıklı işlemeyi mümkün kılmak için yeniden yapılandırılmıştır ve hızda 10 kat artış sağlanmıştır. Sağlanan Sparse Bundle Adjustment optimizasyon arayüzü, daha hızlı ve daha esnek optimizasyon ayarları sağlayan Google Ceres ile değiştirilmiştir. Buna ek olarak, optimizasyon arayüzü çalışma zamanında dinamik olarak yüklenen bir eklenti arayüzüne dönüştürülerek gelecekteki geliştiricilerin orijinal kodu değiştirmeden en yeni ve en iyi optimizasyon teknolojilerini kullanmalarına olanak sağlanmıştır. Eşleme oturumlarının kaydedilmesi ve yüklenmesi için serileştirme ve serileştirmeden çıkarma desteği geliştirilmiştir. Son olarak, ölçümleri işlemek için yeni işleme modları ve K-D ağaç aramaları geliştirilerek yerelleştirme ve çok oturumlu eşleme sağlanmıştır. Birkaç küçük iyileştirme ve optimizasyon da yapılmıştır, ancak kısalık için hariç tutulmuştur. (Macenski ve Jambrecic, 2021)

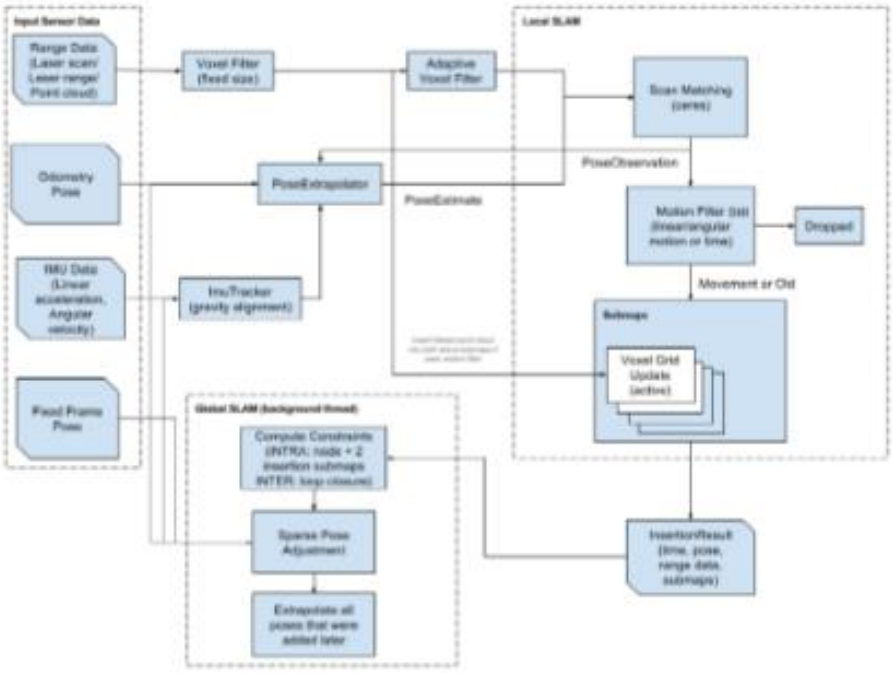
2.7.1. Cartograph

Google'ın Cartographer'ı, r = 5 cm çözünürlükte 2D ızgara haritalar oluşturan sensör donanımlı bir sırt çantası şeklinde iç mekan haritalama için gerçek zamanlı bir çözüm sunmaktadır. Sistem operatörü, bir binanın içinde yürürken oluşturulan haritayı görselleştirebilir. Lazer taramaları, kısa süreler için mevcut konumun en iyi tahmini olduğu varsayılan bir alt haritaya eklenir. Tarama eşleştirme yeni bir alt haritaya göre gerçekleştirilir, bu nedenle yalnızca en son taramalara bağlıdır ve dünya çerçevesine göre poz tahminlerindeki hataları biriktirir. Cartographer, birden fazla platform ve sensör konfigürasyonunda 2D ve 3D olarak gerçek zamanlı SLAM sağlar. Sistem iki alt sistemden oluşmaktadır:

1. Yerel SLAM, bölgenin iyi alt haritalarını oluşturmak için kullanılır. Yerel olarak tutarlı olması amaçlanan ancak zaman içinde kaymasına izin verilen bir dizi alt harita üretir. Yerel SLAM aynı zamanda yerel yörüngeyi de oluşturur.
2. küresel SLAM, alt haritaları mümkün olduğunca tutarlı bir şekilde bağlamak için kullanılır. Ayrı iş parçacıklarında çalışır ve yerel SLAM tarafından oluşturulan alt haritalar arasındaki döngü kapatma kısıtlamalarını bulur.

Taramalardan gelen sensör verilerini önceki alt haritalarla eşleştirerek döngü kapatma işlemini gerçekleştirir. Ayrıca en tutarlı küresel çözümü elde etmek için diğer sensörleri de dahil eder (URL - 1). Hector SLAM, ROS'ta iyi bilinen ve sık kullanılan bir pakettir. Taranan verilerin önceden var olan haritalar veya diğer taranmış verilerle karşılaştırılmasını içeren tarama eşleştirmesine dayanır. Bu veri karşılaştırması, lazer taramalarının uç noktaları ile oluşturulan harita arasında en uygun hizalamayı bulmaya çalışan Gauss-Newton algoritması kullanılarak çözülür. Tarama karşılaştırması sırasında algoritma yalnızca önemli noktaları dikkate alır. Bu yaklaşımın en büyük avantajı, robot odometrisinin gerekli olmaması ve uçan veya saha robotlarına uygulanmasını kolaylaştırmasıdır. Ancak bu yaklaşımın dezavantajı, yavaş telemetri, odometri yokluğu ve hızlı hareketin bir arada olduğu durumlarda veri karşılaştırması sırasında yüksek düzeyde yanlışlık oluşabilmesidir. Bu yanlışlık ortaya çıkan haritaya yansır. Hector haritalama düğümü odometrik bilgi kullanmayan bir SLAM yaklaşımıdır ve dönme/eğilme hareketi (sensörün, platformun veya her ikisinin) sergileyen platformlarda kullanılabilir. Modern LIDAR sistemlerinin yüksek güncelleme hızından

yararlanır ve sensörün tarama hızında 2D poz tahminleri sağlar. Hector SLAM, bir atalet algılama sistemi kullanarak 3B navigasyon tekniklerini sağlam bir tarama eşleştirme tabanlı 2B SLAM sistemi ile birleştirir (URL - 1).



Şekil 2.5. Google Haritalar (URL - 1)

2.7.2. Karto-SLAM

Karto Robotics tarafından bir ROS uzantısı olarak geliştirilen Graph-SLAM yöntemine dayanan Karto SLAM algoritması, diğer SLAM algoritmalarıyla karşılaştırılabilir hesaplama karmaşıklığına sahip başarılı bir algoritma optimizasyonudur (KartoSlam, ROS 2019). Bu algoritmada her bir düğüm, robotun yörüngesine ve duyusal verilerine dayanan konumunu temsil eder. Düğüm bağlantıları yakın konumlar arasındaki hareketleri temsil eder. Her yeni düğüm için harita, uzamsal olarak en uygun düğüm konfigürasyonu bulunarak hesaplanır. Ancak, yetersiz paket dokümantasyonu, sınırlı ayarlanabilir seçenekler ve bir odometri uygulamasının gerekliliği nedeniyle kurulum sırasında zorluklar ortaya çıkabilir. Karto SLAM algoritması, haritayı grafikler aracılığıyla temsil etmek için grafik tabanlı bir yaklaşım kullanır. Bu durumda, her düğüm robotun konumunu ve yörüngesi boyunca bir dizi sensör ölçümünü temsil eder. Yer işaretlerinin sayısı arttıkça, daha fazla bellek gerekir (HectorSlam ROS, 2014).

Doğrusal olmayan matris doğrudan çözümleri sorununu ele almak için Karto SLAM seyrek poz yapılandırması yoluyla optimizasyon sağlar.

2.7.3. Core-SLAM

Çekirdek SLAM algoritması (URL - 2) taranan verileri eşleştirmek için basit bir parçacık filtresi kullanır. Modülle ilgili bir diğer sorun da dokümantasyon eksikliğidir ve bu da dağıtımı son derece sorunlu hale getirmektedir. Algoritma iki adıma ayrılmıştır: (1) çok temel bir Parçacık Filtresi algoritması kullanarak her gelen tarama için mesafeyi hesaplayan mesafe hesaplama ve (2) harita üzerinde alınan taramalara karşılık gelen çizgileri çizen harita güncelleme (Santos ve diğ, 2013). Bununla birlikte, bir engel tespit edildiğinde, algoritma tek bir nokta çizmek yerine, engeli çevreleyen ayarlanabilir bir nokta kümesi çizer.

2.7.4. Lago-SLAM

Lago SLAM algoritması, (Duchon ve diğ, 2019) bölümünde açıklandığı gibi, Graph SLAM'a dayanmaktadır. Onu diğer çizge tabanlı SLAM yöntemlerinden ayıran şey, optimizasyon sürecinde bulunacak uygun çizge konfigürasyonu için herhangi bir başlangıç tahminine ihtiyaç duymamasıdır. Ancak bu gerçek, algoritma için hesaplama karmaşıklığında bir artışa yol açmaktadır. Ne yazık ki, modülün ROS web sitesinde yayınlanan herhangi bir resmi bilgisi yoktur, bu da bulunmasını zorlaştırmaktadır.

2.8. Robot Yol Planlaması

Robot yol planlamasının amacı, ideal mesafe, zaman ve enerji ile başlangıç noktasından hedef noktaya en uygun çarpışmasız yolu belirlemektir. Mobil robotların hareket ettiği ortam, statik bir ortam veya insanlar, makineler ve diğer robotlar gibi hareketli nesneleri içeren dinamik bir ortam olarak sınıflandırılabilir (Bölük ve diğ, 2019). Robot yol planlama algoritmalarında, robotun çevreyi tanıma yeteneğine bağlı olan çevrimiçi ve çevrimdışı olmak üzere iki tür vardır.

1. Çevrimdışı yol planlama algoritmaları çevre hakkında tam bilgi gerektirir ve tüm çevre statik olmalıdır. Çevrimdışı algoritmalar ayrıca iki türe ayrılabilir: Orijinal arama, alana özgü bilgileri (sezgisel işlev) kullanır. A* algoritması en popüler bilgilendirilmiş çevrimdışı arama algoritmasıdır. Bilgisiz arama algoritmaları alana özgü herhangi bir bilgi kullanmaz (kör fonksiyonlar). Dijkstra'nın algoritması iyi

bilinen bir bilgisiz çevrimdışı arama algoritmasıdır. Çevrimiçi yol planlama algoritmaları, robot hareket ettikçe yolu planlayan ve çevresel değişikliklere yanıt olarak yeni bir yol oluşturabilen algoritmalardır.

2. Çevrimiçi yol planlama algoritmalarının büyük dinamik ortamlarda kullanılması zordur çünkü çok fazla hesaplama zamanı gerektirirler. Bu sorunu çözmek için çevrimdışı algoritmalar yinelemeli yerine artımlı olarak düzenlenebilir. Artımlı algoritmalar, D*, D* odaklı ve daha hızlı D* Lite gibi problemleri çözmek için geçmiş aramalardan gelen bilgileri kullanan sürekli planlama teknikleridir. Mobil robot yoluna göre robot hedef noktaya doğru hareket etmeye başlar ve her hareketinde engellere karşı durumunu kontrol eden bir hareket planlayıcısı çalıştırılır.

Mobil robot navigasyonunda hareket planlamanın temel amacı, mobil robotu herhangi bir engele çarpmadan hedef konuma ulaşmak için en iyi seçeneğe yönlendirmektir. Robotikte hareket planlama geleneksel ve modern yöntemler olmak üzere iki farklı kola ayrılabilir. Geleneksel hareket planlama genellikle harita tabanlıdır. Harita net olduğu için mobil robot ortamla ilgili tüm bilgileri kullanarak navigasyon işlemini gerçekleştirir ancak bilinmeyen bir ortamda navigasyon yapmak zordur. Geleneksel hareket planlama örnekleri, geleneksel dinamik programlama (DP) kullanarak haritalama ve yol bulma problemlerini ayrı ayrı değerlendirir. Bir robotun karmaşık ve yapılandırılmamış ortamlarda otonom olarak gezinme yeteneği, teslimat, arama ve kurtarma, denetim, ulaşım, gezegen keşfi ve bakım gibi çeşitli zorlu pratik uygulamalarda rol oynar.

1. Ortamda hareketli nesneler varsa hareket planlaması yapılacaktır.

(A) Harita açıksa, geleneksel hareket planlaması uygulanacaktır (Harita Tabanlı Algoritma).

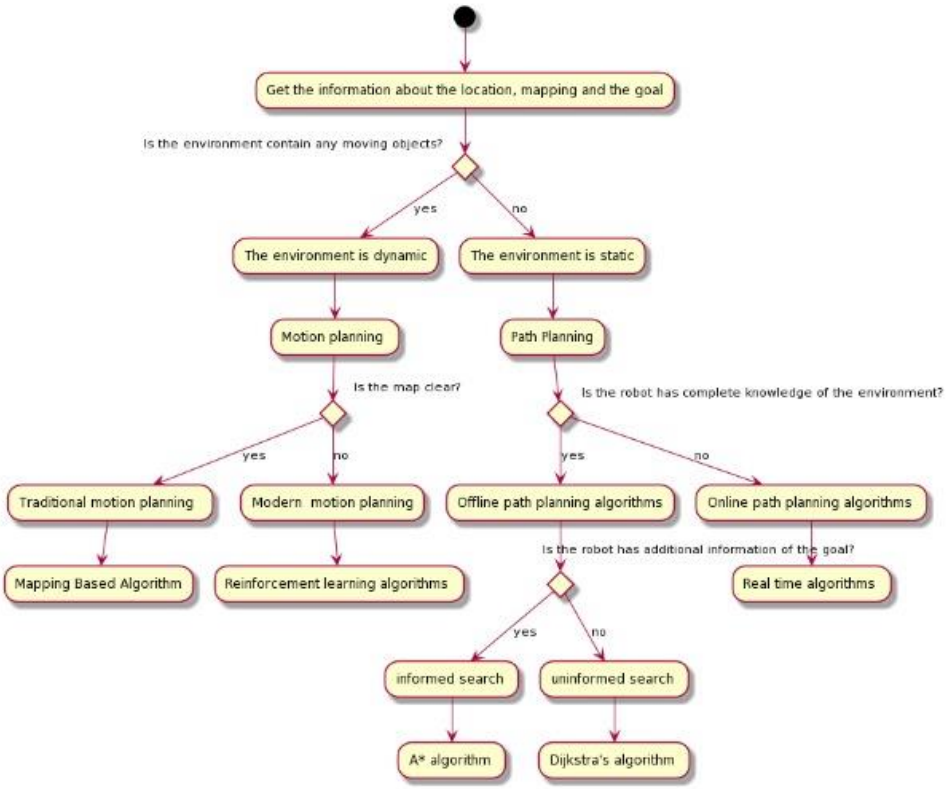
(B) Harita belirsizse, modern hareket planlaması kullanılacaktır (Takviyeli Öğrenme Algoritması).

(C) Ortamda hareketli nesne yoksa yol planlaması yapılacaktır.

2. Robot çevre hakkında tam bilgiye sahip değilse, Çevrimiçi Yol Planlama Algoritması (Gerçek Zamanlı Algoritma) kullanılacaktır.
3. Robot hedef hakkında ek bilgiye sahipse, Bilgilendirilmiş Arama (A* Algoritması) gerçekleştirilecektir.

4. Robot hedef hakkında ek bilgiye sahip değilse, Bilgisiz Arama (Dijkstra Algoritması) uygulanacaktır.

Yol planlama metodolojileri Şekil 2.6'da özetlenmiştir.



Şekil 2.6. Yol planlama Metodolojileri

(Megalingam ve diğ, 2020), sanal bir diferansiyel tahrikli robotun planlanan yolu ile bir ROS navigasyon yığını kullanılarak bir Gazebo-ROS simülatöründe katettiği gerçek yol arasındaki tutarsızlık incelemiştir. Dinamik engellerin iki tekerlekli bir mobil robotun planlanan ve katedilen yolun üzerindeki etkisini katedilen mesafe, çalışma süresi, tespit edilen engeller gibi çeşitli verileri dikkate alarak farklı senaryolarda değerlendirmiştir. Navigasyonun birincil amacı, mobil robotun planlanan yolda güvenli bir şekilde hareket etmesini sağlamaktır, ancak robotun planlanan yol, sensör hataları, dinamik engellerin varlığı, robot dinamikleri ve sistem özellikleri dahil olmak üzere çeşitli faktörler tarafından değiştirilebilir. Yazarlar ayrıca yolun dinamik engeller dikkate alınmadan planlandığını tespit etmişlerdir. Yeni dinamik engellerin varlığı yerel maliyet haritasında saklanmış ve aşağıdaki test senaryolarında yol planlamasını etkilemiştir. Dinamik engellerin olması nedeniyle hesaplama süresi artmıştır. Ancak, her bir durum

için manuel navigasyonda önemli bir artış olmamıştır. Planlanan ve katedilen yollar çeşitli test senaryolarında önemli farklılıklar gösterse de bu farklılıklar öncelikle sensör doğruluğuna, çevresel koşullara, engellere ve diğerlerine bağlı olduğu söylenmiştir.

2.9. Robot Navigasyonu

Robot navigasyonu, lineer cebir, geometri, trigonometri, fizik, mekanik gibi çeşitli disiplinlerden bilgi gerektiren oldukça karmaşık bir problemdir. Ancak ROS, bir robotu uzayda gezdirmemizi çok daha kolay hale getiriyor. Navigasyon, bir robotun engellerle çarpışmadan uzayda hareket etme kabiliyetini ifade eder. Bu engeller statik, yani sabit veya dinamik olabilir, yani herhangi bir zamanda ortaya çıkabilir veya kaybolabilir. Robotun hareketi sırasında harcadığı zaman da dikkate alınması gereken bir faktördür. Diğer durumlarda robotun bu engellerden kaçınması gerekir. Tipik olarak, robot navigasyonu görevi hareket planlamasını içerir. Aslında, bir robotun hedefine ulaşmak için engellerden kaçınırken bir konumdan diğerine hareket etmesi gerektiğinde, hedefe ulaşmak için yolunu ve hareketini planlaması gerekir. Robot navigasyonunun iki ana kategorisi vardır.

1. İlk kategori, harita tabanlı navigasyondur (Harita Tabanlı Navigasyon). Robot, navigasyon için gerekli olan ve hafızasına yüklenen ve depolanan bir ortam haritasını kullanır. Robot, çevrenin tüm statik yönleri hakkında global bilgiye sahiptir. Öte yandan, herhangi iki konum arasındaki yolunu planlamak için bu haritayı kullanır.
2. İkinci kategoriye reaktif navigasyon denir (Reaktif Navigasyon). Navigasyon için herhangi bir harita kullanmayan ve hareketini planlamak için yalnızca sensörlerinden gözlemlenen yerel bilgilere dayanan robot.

2.9.1. Harita Tabanlı Navigasyon

Robot navigasyon problemini çözmek için robotun Tablo 2.1'deki üç soruya cevap bulması gerekir.

2.9.1.1. Yerelleştirme

Lokalizasyon, robotun nerede olduğunu bilmesine ve konumunu belirlemesine yardımcı olur. Yerelleştirme için GPS, lazer mesafe bulucular, kameralar, ultrasonik sensörler, alınan sinyal gücü ve dış ortamlardaki diğerleri dahil olmak üzere çeşitli yöntemler

kullanılır. Konum, yerel bir ortama göre sembolik bir referans olarak belirtilebilir. Örneğin, bir odanın merkezi veya oda 23 gibi topoloji tabanlı bir koordinat sistemi veya enlem, boylam ve yüksekliğin mutlak koordinatları veya daha genel olarak X, Y ve Z'nin üç boyutlu bir koordinat sistemi olarak.

Tablo 2.1. Robot navigasyon problemi soruları

Soru	**Anlam**	**Temel gezinme işlevi**
Neredeyim	Yerelleştirme	Yerelleştirme
Neye gidiyorum?	Hedefi bulma konum	Haritalama
Oraya nasıl giderim?	Sırasıyla yol planlama süreci	Hareket planlama

2.9.1.2. Haritalama

Haritalama süreci, robotun şimdiye kadar nereye gittiğini anlayabilmesi için çevresinin bir haritasına sahip olmasını gerektirir. Harita, robota konumlar ve yönler hakkında bir anlayış sağlar; grafik veya matris gösterimi gibi robotun hafızasına manuel olarak ayarlanabilir veya robot yeni ortamları keşfettikçe kademeli olarak oluşturulabilir.

2.9.2. Davranış Temelli Navigasyon

Davranış Temelli navigasyonunda toplam dört yetkinlik vardır

- Engellerden Kaçınma
- Duvar Takibi
- Koridor Takibi
- Hedef Arama

Bu navigasyon sisteminin çalışma prensibi, çok sayıda eşzamanlı hedefin bir dizi kontrollü eylemle harmanlanmasını içerir. Ayrıca, navigasyon sisteminin tasarımının olası kontrollü eylemler olarak tanımlanan kabul edilebilir davranışsal özellikleri ifade etmesi beklenmektedir. Davranış tabanlı navigasyon, çevresel haritalama ve robot davranışları olmak üzere iki süreci entegre eder. Harita çevresel bir durumu temsil ederken, robot hareketi davranış tarafından kullanılır. İki süreçten yalnızca biri kullanılıyorsa, sistem iki koşulda uygulanmalıdır; bu büyük ölçüde uygulamaya ve diğer eşzamanlı robot davranışlarıyla etkileşime bağlıdır. Bu navigasyon sisteminin dezavantajları, bu yaklaşım kullanıldığında iki ana sorunla sonuçlanması kaçınılmazdır:

- iki basit davranışın birleşiminin karmaşık bir davranışa yol açması
- ikiden fazla davranışın bütünleştirilmesi.

2.9.3. Öğrenme Temelli Navigasyon

Bu navigasyon sisteminin çalışması, geleneksel yöntemlerin aksine, robotun karşılaştığı potansiyel çevresel sorunları dikkate almaya gerek kalmadan sağlam tasarımını sağlamak için çeşitli sensörlerin, aktüatörlerin ve kontrolörlerin dahil edilmesine dayanır. Bununla birlikte, bu navigasyon sisteminin sınırlamaları vardır. Sadece tek robot davranışları ve çoklu robot davranışları ile uyumludur. Bununla birlikte, karmaşık bir ortamda bir robotun görevlerini yönetmek zor olabilir ve kontrol sürecinde daha fazla hataya yol açabilir. Ayrıca, sistemin bazı bileşenleri ve robotun işlevi doğru çalışmazsa, hata toleransı özelliği büyük ölçekli bir ortamda desteklenmez. (Nurmaini ve Tutuko, 2017).

2.9.4. İletişim Temelli Navigasyon

Bu navigasyon sisteminin çalışması, yerel iletişim modifikasyonu ve algılama yoluyla çok sayıda fiziksel robotun organize edilmesini içerir. Bu robotik sistemler robotik dünyasında özellikle sürü robotlar olarak bilinmektedir. Çalışmaları, basit bir otonom platform, merkezi olmayan kontrol ve bir tür biyolojik ilham ile işbirliği ve koordinasyonun önemini içeren bazı özellikleri içeren birkaç yöntem gerektirir. Basit davranış kurallarına dayalı olarak oluşturulan algoritmanın iki amacı vardır: (1) bilgi işleme sistemindeki karmaşıklık gereksinimini en aza indirmek ve (2) enerji harcamasını optimize eden davranışların üretilmesine izin vermek. Ancak bu yaklaşımın robotun hareketini kontrol edememesi, yalnızca yerel olarak optimal olması, karmaşık bir ortamda birden fazla robot manevra yaptığında dikkatsiz küresel hareketler üretmesi ve robotların yerel minimumlarda sıkışıp kalma olasılığı gibi çeşitli dezavantajları vardır. Bu nedenle, hem tek robot hem de sürü robot navigasyon sistemleri için hassasiyet, yetersizlik ve belirsizlik dahil olmak üzere gerçek zamanlı, düzensiz ve dinamik bir ortam tasarlamak için önemli olan zorlukların ele alınmasına acil ihtiyaç vardır. Ayrıca, başarısız iletişim ve hatalı algoritmalar navigasyon sisteminin geliştirilmesi sırasında bir endişe kaynağı olarak kabul edilmektedir. Son olarak, tüm parametreler analiz edildiğinde, tüm robotik navigasyon sisteminin performansı iyileştirilebilir veya en azından düşürülmeyebilir. Bu çalışmada izlenen metodoloji

aşağıdaki gibidir: İlk olarak, sürü robotlara dönüştürülen tek bir robotun çalışması için gerekli bilgilerin sağlanması. İkinci olarak, üç yöntemin karşılaştırmalı analizi.

- Geleneksel yapay zeka
- Yumuşak bilgi işlem
- Sürü zekası

mobil robot navigasyon sistemi ile ilgilidir.

2.9.5. Geleneksel Yapay Zeka

Sürtünme mekanizmasının yalnızca yerel bir ortamda çalıştığı görülmektedir, bu nedenle ortam sürekli değişirse, robot hareketini tanımakta ve kontrol etmekte zorluk çekecek ve bu da sonuçta görevin başarısını etkileyecektir. Bu sınırlamanın üstesinden gelmek için çok sayıda geleneksel yapay zeka (AI) yaklaşımına ihtiyaç vardır.

- Yapay potansiyel alan yöntemleri
- Sanal hedef yaklaşımı
- Dönüm noktası öğrenme
- Teğet grafiği
- Yol hızı ayrıştırma yöntemi
- Erişilebilirlik grafiği
- Uzay-zaman kavramı
- Kademeli planlama
- Bağıl hız yaklaşımı
- Reaktif kontrol şeması
- Eğrilik-hız yöntemi
- Dinamik pencere yaklaşımı
- Eşzamanlı Yerelleştirme ve Haritalama

Ne yazık ki, bahsedilen geleneksel YZ yaklaşımlarının aşağıdaki dezavantajlara maruz kaldığı görülmektedir:

- Yerel kontrol
- Çok sayıda durum nedeniyle yüksek hesaplama kaynakları
- Optimizasyon modülünün olmaması

- Yerel minimumlar nedeniyle düzenli kilitlenme durumları
- Salınımlara neden olan yakın aralıklı engeller arasında geçiş yok.

2.9.6. Yumuşak Hesaplama

Yumuşak Hesaplama (YH) yöntemlerinin amacı, sağlam ve düşük maliyetli bir çözüm elde etmektir. Bu yaklaşım, kesinlik, belirsizlik, kısmi doğruluk ve yaklaşıklık üretmemesi bakımından geleneksel YZ yöntemlerinden farklıdır. Bu teknik, mobil robot uygulamalarının tasarımında yaygın olarak kullanılmış ve iyi bir performansla sonuçlanmıştır. Düzensiz ve olağandışı ortamlarla başa çıkma yeteneği, onu yalnızca robotik kontrol problemlerini değil, aynı zamanda navigasyon problemlerini de ele almak için uygun bir teknik haline getirmiştir. Bu tekniklerin mobil robot navigasyonuna etkili yöntemler getirdiğine ve zekayı geliştirdiğine inanılmaktadır.
Çeşitli yumuşak hesaplama teknikleri şunları içerir:

- Bulanık mantık sistemleri
- Sinir ağları
- Genetik algoritmalar.

2.9.6.1. Bulanık Mantık Sistemleri

Bu, özellikle Tip-1 Bulanık Mantık Sistemleri (T1FLS) olmak üzere sistemlerin doğasında var olan belirsizliğe rağmen mobil robot navigasyonu için mükemmel bir çözümdür. Bununla birlikte, T1FLS belirsizlikleri tam olarak ele alamamakta, bu da belirsizliklerin etkisini en aza indirmek için üyelik fonksiyonlarının (MFS) sınırlı bir şekilde modellenmesine neden olmaktadır. Son zamanlarda, gerçek bir ölçüm sonucu elde etmenin zor olduğu durumlarda son derece yararlı olan Tip-2 Bulanık Mantık Sistemi (T2FLS) adı verilen yeni bir bulanık mantık türü oluşturulmuştur. Bununla birlikte, T2FLS hesaplama açısından yoğundur ve özellikle mobil robot navigasyonu gibi gerçek zamanlı uygulamalar için oluşturulması zordur. Bu arada, hesaplamaları basitleştirmek için bir Aralık Tip-2 Bulanık Mantık Sistemi (IT2FLS) önerilmiştir. IT2FLS, gelişmiş hesaplama performansına sahip yeni nesil bulanık sistemler üretme potansiyeline sahiptir, ayrıca T1FLS'nin sınırlamalarını ele alır ve T1FLS'den daha yüksek talep sunar. (Nurmaini ve Tutuko, 2017)

2.9.6.2. Nöral Ağlar (NNS)

Sinir Ağları, durumları gözlemleme ve taklit etme yeteneğine sahiptir ve her ortam için olağanüstü algılama ve örüntü tanıma sağlar. Gürültülü veya hatalı verilerin NN'ler aracılığıyla işlenmesi, NN'lerin gürültüye karşı oldukça toleranslı olduğu bilindiğinden klasik tekniklerden daha değerlidir. Ayrıca, çok sayıda çalışma mobil robot navigasyonu için modeller geliştirmek üzere NN tekniklerini başarıyla uygulamıştır. Bununla birlikte, geleneksel Sinir Ağı tekniklerinin en büyük dezavantajı, eğitim verilerinin tekrar tekrar sunulmasıdır, bu da uzun öğrenme sürelerine neden olur ve bu da gerçek zamanlı uygulamalar için uygun değildir (Nurmaini ve Tutuko, 2017).

2.9.6.3. Genetik Algoritma (GA)

Genetik Algoritma (GA) en güçlü algoritmalardan biri olarak tanımlanmıştır. Ayrıca, GA gelişmekte olan bir optimizasyon yöntemi olarak sunulmaktadır ve temel özellikleri onu mobil robot navigasyonu sorununa çözüm bulmak için cazip bir seçim haline getirmiştir. GA, gradyan tabanlı yöntemleri içeren geleneksel arama tekniklerinin neden olduğu aşağıdaki sorunları da çözebilir:

- Yüksek hesaplama maliyeti,
- Geniş bellek alanları,
- Zaman tüketir. Bununla birlikte, GA algoritmasının mobil robot navigasyonuna uygulanması, küresel bir optimum oluşturmak için mücadele eder ve bu da bir çözüm bulmanın yanı sıra yavaş yakınsamaya neden olur.

2.9.6.4. Sürü Zekası

SI algoritmaları, böcekler ve hayvanlar gibi gruplar halinde yaşayan sosyal canlıların davranışlarına dayanmaktadır. Bu nedenle Yumuşak Hesaplama yöntemi, algoritmaları bireysel insan bilgisine dayalı olarak geliştirildiği için Sürü Zekası (SI) yöntemiyle çelişmektedir. Sosyal bir böceğin davranışı, Sürü Zekasının (SI) ana kavramı haline gelir ve bu kavram daha da kategorize edilebilir.

- Özerklik
- Dağıtılmış işleyiş
- Çok sayıda yapay sistem oluşturmak için kendi kendini organize etme.

SI yönteminin navigasyon sistemlerinde, özellikle çok robotlu ve sürü robotlarında uygulanması,

- Dağıtılmış işleyiş
- Özerklik için iletişim
- Tüm robot gruplarıyla kendi kendini organize etmenin işbirliği ve koordinasyonu.

SI, navigasyon sisteminde, özellikle önceden tanımlanmış bir ortamdaki karmaşık sistem için mükemmel performans üretir. ROS1 ve ROS2 sürümü de robotik topluluğunda yaygın olarak kullanılır ve navigasyon yetenekleri arasındaki farkları anlamak, kullanıcıların robotik uygulamaları için hangi sürümü seçeceklerine bilinçli kararlar vermelerine yardımcı olur. ROS navigasyonu ile ROS 2 navigasyonu arasındaki bazı önemli farklar şunlardır:

- İletişim Ortak Katmanı (Middleware): ROS 1 ve ROS 2 arasındaki önemli farklardan biri iletişim middleware'leridir. ROS 1, ROS Master-Köle mimarisini kullanır ve ROS Master düğümler arasındaki iletişimi yönetir. Bununla birlikte, ROS 2, daha güçlü ve verimli iletişim sağlayan Veri Dağıtım Hizmeti (DDS) middleware'ini kullanır, özellikle büyük ölçekli sistemler ve gerçek zamanlı uygulamalarda avantaj sağlar.
- Gerçek Zamanlı Yetenek: ROS 2 gerçek zamanlı yetenekleri göz önünde bulundurularak tasarlanmıştır, iletişim ve yürütme süreçlerinde daha belirgin belirlenimlilik ve tahmin edilebilirlik sunar. Bu geliştirme, zamanlamanın kritik olduğu güvenlik odaklı uygulamalar için önemlidir.
- Modülerlik ve Ölçeklenebilirlik: ROS 2, daha modüler bir yapı sunar, bu da üçüncü taraf kütüphanelerin daha kolay entegrasyonunu ve daha yapılandırılabilir ve bakımı kolay bir şekilde karmaşık robotik sistemlerin oluşturulmasını sağlar.
- Çoklu Platform Desteği: ROS 2, Windows, macOS ve çeşitli gömülü sistemler de dahil olmak üzere farklı platformlarda daha geniş bir destek sunar, çeşitli platformlarda çapraz platform geliştirmeyi mümkün kılar.
- Performans: İletişim middleware'indeki iyileştirmeler ve gerçek zamanlı yetenek sayesinde, ROS 2 navigasyonunun gecikme, verimlilik ve genel sistem performansı açısından daha iyi olması beklenir.

- Geçiş ve Uyumluluk: ROS 2, birçok geliştirmeyi beraberinde getirse de, ROS 1 hala yaygın olarak kullanılmaktadır ve birçok mevcut robotik proje ve kütüphane ROS 1 üzerine kuruludur. ROS 1'den ROS 2'ye geçiş, mevcut kodların uyarlanmasını veya yeniden uygulanmasını gerektirebilir, bu nedenle uyumluluk bazı kullanıcılar için önemli bir düşünce olabilir.

Sonuç olarak, ROS navigasyonu ile ROS 2 navigasyonu arasındaki seçim, robotik proje için belirli ihtiyaçlar ve hedeflere, aynı zamanda geliştiriciler ve kullanıcılar tarafından tercih ve kısıtlamalara bağlıdır. ROS 1, köklü bir geçmişe sahiptir ve geniş bir paket ve kütüphane ekosistemine sahiptir. Gerçek zamanlı yeteneklere ve çapraz platform uyumluluğuna sahip olan ROS 2, gerçek zamanlı performansı, modüler yapısı ve geliştirilmiş özellikleri ile daha karmaşık ve ileri düzey robotik uygulamalara daha uygun olabilir.

2.10. ROS Dağıtımları

Bir ROS dağıtımı (ROS Distributions, 2021), Linux dağıtımlarına (örneğin Ubuntu) benzer şekilde sürümlendirilmiş ROS paketlerinin bir koleksiyonudur. ROS dağıtımlarının amacı, geliştiricilerin her şeyi ileriye taşımaya hazır olana kadar nispeten istikrarlı bir kod tabanı üzerinde çalışmalarına olanak sağlamaktır. Bu nedenle, bir dağıtım yayınlandıktan sonra, değişiklikleri hata düzeltmeleri ve çekirdek paketler (ros-desktop-full altındaki her şey) için kırıcı olmayan iyileştirmelerle sınırlamaya çalışıyoruz. Ve bu genellikle tüm topluluk için geçerlidir, ancak kurallar "üst düzey" paketler için daha az katıdır ve bu nedenle kırıcı değişikliklerden kaçınmak belirli bir paketin bakımcılarına düşer.

ROS Noetic, Melodic ve kinetic arasındaki karşılaştırma Tablo 2.2'de gösterilmiştir

Tablo 2.2. ROS 1 Dağıtımları arasında karşılaştırma

Özellikleri	**Noetic**	**Melodic**	**Kinetic**
Desteklenen işletim sistemi	Ubuntu (20.04)	Ubuntu (18.04)	Ubuntu (16.04)
C++	C++ 14	C++ 14	C++ 11
Python	Python 3.8	Python 2.7	Python 2.7
Gazebo	11.x	9.0	8.0
OpenCV	4.2	3.2	2.0

2.11. Ros1 Ve Ros2 Karşılaştırması

2.11.1. Platformlar, C++ and Python

ROS Ubuntu üzerinde test edilmiş ve diğer Linux dağıtımlarının yanı sıra OS X üzerinde de sürdürülmüştür. ROS, C++03 kullanır ve ROS Python 2'yi hedefleyen API'sinde C++11 özelliklerini kullanmaz.

Şu anda ROS2, CI kullanılarak Ubuntu Xenial, OS X El Capitan ve Windows 10 üzerinde test edilmekte ve desteklenmektedir. ROS2 öncelikle CPP11 kullanmaktadır ancak gelecekte CPP14 ve CPP17 kullanmayı planlamaktadır. ROS2 Python >= 3.5 kullanmaktadır.

2.11.2. Ara Yazılım

ROS özel bir serileştirme formatı kullanmaktadır (transfer protokolü + merkezi keşif mekanizması). Buna karşılık, ROS2'nin tüm uygulamaları şu anda DDS standardına dayanmaktadır.

2.11.3. Süre ve Zaman Türlerini Birleştirin

ROS'ta süre ve zaman türleri hem C++ hem de Python'da istemci kütüphanelerinde tanımlanmıştır. ROS2'de bu tipler mesaj olarak tanımlanarak diller arasında tutarlı hale getirilmiştir.

2.11.4. Yaşam Döngüsüne Sahip Bileşenler

ROS'ta her düğümün tipik olarak kendi ana işlevi vardır. ROS2'de yaşam döngüsü, roslaunch gibi araçlar tarafından birden fazla bileşenden oluşan bir sistemi deterministik olarak baş

2.11.5. İş parçacık Modeli

ROS'ta geliştirici yalnızca tek iş parçacıklı yürütme veya çok iş parçacıklı yürütme arasında seçim yapabilir. ROS2'de daha ayrıntılı yürütme modelleri mevcuttur ve özel yürütücüler kolayca uygulanabilir.

2.11.6. Birden Fazla Düğüm

ROS'ta tek bir işlemde birden fazla düğüm oluşturmak mümkün değildir.ROS2'de, tek bir işlemde birden fazla düğüm oluşturmak mümkündür.

2.11.7. Roslaunch

RROS başlatma dosyaları, ROS'ta XML'de sınırlı yeteneklerle tanımlanır. ROS2'de, başlatma dosyaları koşullu ifadeler gibi daha karmaşık mantığa izin veren Python'da yazılmıştır.

2.11.8. ROS1 VS ROS2 Yazma Düğümleri

ROS1 ticari kullanım düşünülerek tasarlanmamıştır, bu nedenle güvenlik, ağ topolojisi ve sistem çalışma süresi gibi konulara öncelik verilmemiştir. Bu nedenle, ROS artık ticari ortamlarda benimsendikçe, önemli kusurlarının çoğu giderek daha belirgin hale geldi. Bu nedenle, ROS'un ticari kullanımı dikkate alınarak sıfırdan yeniden inşa edilmesine ihtiyaç duyulmuştur, yani ROS2. ROS2, ticari kullanıma ve benimsenmeye hazır hale getirmek için aşağıdaki tasarım gereksinimleri ile sıfırdan oluşturulmuştur:

1. Güvenlik : Gerektiğinde uygun şifreleme ile güvenli olmalıdır.
2. Embedded Sistemler : ROS2 gerektiğinde gömülü sistemler üzerinde çalışabilmelidir.
3. Çeşitli ağlar: Robotların yerel ağlarında uydular arasında çoklu atlamaların olduğu geniş ağlara kadar çok çeşitli ortamlarda çalışabilmeleri ve iletişim kurabilmeleri gerekir.
4. Gerçek zamanlı hesaplama: Robotikte, çalışma zamanı verimliliği çok önemlidir, bu nedenle hesaplamaları gerçek zamanlı olarak güvenilir bir şekilde gerçekleştirebilmeniz gerekir.
5. Ürün Hazırlığı : Ürün Hazırlığı için ilgili güvenlik/endüstriyel standartlara uyum gereklidir.

2.11.9. Düğümler

Yazma Düğümleri : ROS1'de, düğüm işlevlerini nasıl yazmanız gerektiğini söyleyen belirli bir yapı yoktur. Programınızda herhangi bir yere geri arama işlevleri eklemeye

karar verebilir veya isterseniz nesneye yönelik programlamayı kullanabilirsiniz, ancak her birinin uygulaması benzersiz olabilir.

ROS2'de işler farklıdır. Düğümlerinizi nasıl yazacağınıza dair bir kural vardır. Node nesnesinden miras alan bir sınıf oluşturmalısınız (örneğin: Cpp'de rclcpp::Node, Python'da rclpy.node.Node). Bu sınıfta tüm ROS2 işlevlerinize sahip olacaksınız. Aynı yürütülebilir dosyada birden fazla düğüm - ROS2 Bileşenleri: ROS1'de, aynı yürütülebilir dosyada birden fazla düğüme sahip olmak mümkün değildir. Her düğümün ayrı bir çalıştırılabilir dosya olarak derlenmesi gerekir. Ancak ROS2'de, aynı yürütülebilir dosya içinde birden fazla düğüm tanımlanabilir, bu da daha verimli kaynak kullanımı ve daha iyi performans sağlar. Ayrıca, ROS2'de birden fazla düğüm aynı bağlamı paylaşabilir ve aynı işlemde yürütülebilir, bu da düğümler arasındaki iletişimi ve koordinasyonu basitleştirebilir. Bu özellik aynı zamanda geliştiricilerin ilgili düğümleri aynı süreç içinde bir araya getirebilmeleri sayesinde daha karmaşık ve modüler sistemler oluşturabilmelerine de olanak sağlamaktadır. Genel olarak, ROS2'deki bu özellik ROS1'e kıyasla daha fazla esneklik ve verimlilik sağlar.

Lifecycled nodes - Yaşam döngülü düğümler ROS2, düğümlerin yaşam döngüsünü yönetmek için standartlaştırılmış bir yol sağlayan "yaşam döngüsü düğümleri" kavramını tanıtmaktadır. Bir düğümün yaşam döngüsü, yapılandırma, etkinleştirme, devre dışı bırakma ve temizleme gibi farklı durumları veya aşamaları olarak düşünülebilir. Bir düğümün yaşam döngüsü, çalışması için çok önemlidir ve bu yaşam döngüsünün uygun şekilde yönetilmesi, düğümün güvenilirliğini ve verimliliğini sağlamaya yardımcı olabilir. ROS1'de bir düğümün yaşam döngüsünü yönetmenin standart bir yolu yoktur. Düğümler genellikle manuel olarak veya bu görevleri yerine getiren komut dosyaları aracılığıyla başlatılır ve durdurulur, ancak bir düğümün yaşam döngüsünün farklı durumlarını yönetmek için yerleşik bir destek yoktur. Öte yandan ROS2'nin yaşam döngüsü düğümleri, bir düğümün yaşam döngüsünü yönetmek için standartlaştırılmış bir yol sağlar. Yaşam döngüsü düğümleri, bir düğümün içinde bulunabileceği farklı durumları ve bunlar arasındaki geçişleri tanımlayan bir sonlu durum makinesi kullanılarak uygulanır. Yaşam döngüsü yöneticisi bileşeni, yaşam döngüsü düğümlerinin durum geçişlerini yönetmekten sorumludur ve düğümlerin uygun şekilde yapılandırılmasını, etkinleştirilmesini, devre dışı bırakılmasını ve temizlenmesini sağlar. ROS2'deki yaşam döngüsü düğümleri kavramı, gelişmiş

güvenilirlik, daha iyi hata işleme ve sistem kaynaklarının daha verimli kullanımı gibi çeşitli faydalar sağlar. Yaşam döngüsü düğümleri sayesinde geliştiriciler, karmaşık yaşam döngülerine sahip birden fazla düğümden oluşan karmaşık sistemleri daha kolay oluşturabilir ve bunu daha standart ve tutarlı bir şekilde yapabilirler. Ayrıca yaşam döngüsü düğümleri, düğümlerin doğru şekilde yapılandırılmasını ve başlatılmasını sağlayarak hata veya sorun riskini azaltmak için kullanılabilir. 4 birincil durum vardır:

1. Yapılandırılmamış
2. Aktif Değil
3. Aktif
4. Kesinleştirildi

ROS2'de düğümler farklı durumlara sahip olabilir ve bir düğümün içinde bulunabileceği dört temel durum vardır. Bu durumlar şunlardır: Yapılandırılmamış: Bu, bir düğümün yapılandırılmadan önceki ilk durumudur. Bu durumda düğüm henüz kullanıma hazır değildir. Etkin Değil: Bu durumda düğüm yapılandırılmıştır ancak henüz etkinleştirilmemiştir. Bu, kullanıma hazır olduğu, ancak henüz aktif olarak bir şey yapmadığı anlamına gelir. Aktif: Bu durumda düğüm aktif olarak çalışmakta ve görevini yerine getirmektedir. Düğümün en kullanışlı olduğu durum budur. Sonlandırıldı: Bu durumda düğüm kapanır ve tamamen durdurulmadan önce gerekli temizleme görevlerini yerine getirir. Bu durumlar, düğümlerin kendi yaşam döngülerini tanımlamalarına ve standartlaştırılmış bir şekilde yönetmelerine olanak tanıyan ROS2'deki yaşam döngüsü düğümü kavramı kullanılarak yönetilir. Bu, düğümlerin yaşam döngüsünü yönetmenin bireysel geliştiriciye bırakıldığı ROS1'e göre önemli bir gelişmedir. ROS2'deki yaşam döngüsü düğümü kavramı, düğümleri durumlar arasında geçirmek için kullanılan bir dizi standart mesajın kullanımına dayanmaktadır. Bu mesajlar şunları içerir: lifecycle-msgs/State: Bu mesaj bir düğümün durumunu temsil etmek için kullanılır. lifecycle-msgs/Transition: Bu mesaj durumlar arasındaki geçişi temsil etmek için kullanılır. lifecycle-msgs/TransitionDescription: Bu mesaj bir geçişi tanımlamak için kullanılır. Bu mesajları kullanarak düğümler kendi yaşam döngülerini tanımlayabilir ve durumlar arasındaki geçişlerini standartlaştırılmış bir şekilde yönetebilir, böylece ROS2'de karmaşık robotik sistemlerin geliştirilmesi ve yönetilmesi kolaylaşır.

2.11.9.1. Düğüm Yönetimi

ROS2'deki düğüm yönetimi, kullanıcıların sistem içindeki düğümleri etkin bir şekilde kontrol etmesini ve izlemesini sağlayan kritik bir özelliktir. Uygulamanın gerektirdiği şekilde düğümlerin oluşturulması, başlatılması, durdurulması ve yok edilmesini içerir.

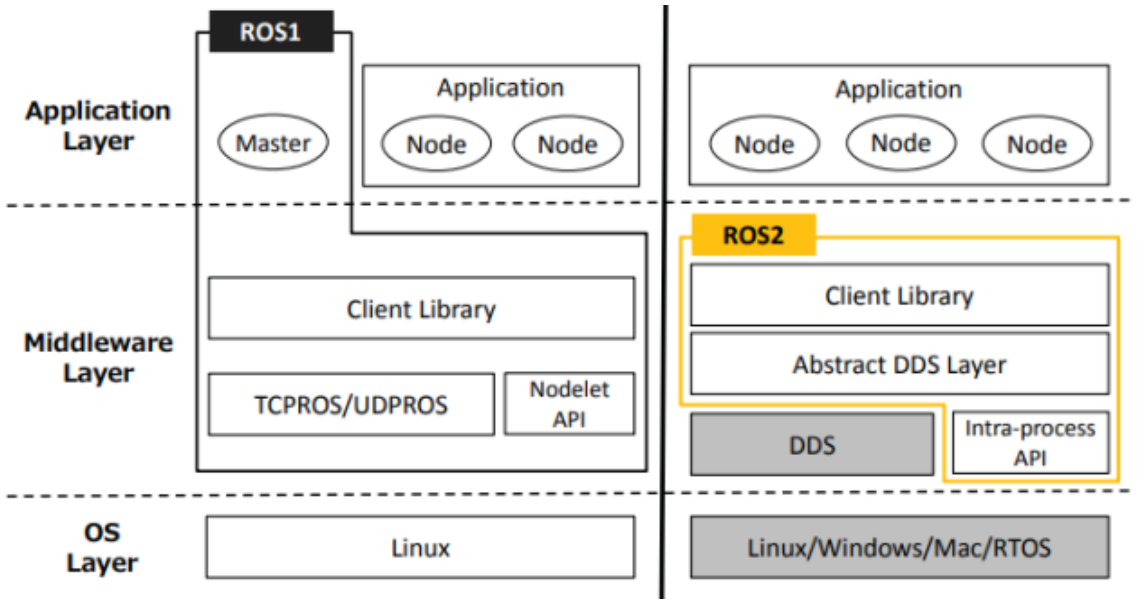
ROS2'de düğüm yönetimi, ROS1'e kıyasla düğümleri yönetmek için daha yapılandırılmış bir yol sağlayan yaşam döngüsü konseptine dayanmaktadır . Yaşam döngüsü kavramı, bir düğümün yaşam döngüsünü başlatılmamış, etkin olmayan, etkin ve sonlandırılmış durumlar olmak üzere çeşitli durumlara ayırır. Bu durumlar düğümün davranışını tanımlar ve düğümleri başlatmak, durdurmak ve yönetmek için tutarlı bir yol sağlar. ROS2'deki düğüm yönetimi, kullanıcıların düğümleri kolayca tanımlamasına ve adlandırma çakışmalarını önlemesine olanak tanıyan düğüm adı ve ad alanı yönetimi gibi özellikleri de içerir. Düğümler, kullanıcıların düğüm yapılandırması, argümanlar ve konuların ve hizmetlerin yeniden eşlenmesi gibi çeşitli başlatma parametrelerini belirtmelerini sağlayan başlatma dosyaları kullanılarak da başlatılabilir.

ROS2'deki düğüm yönetiminin bir başka kritik yönü de, birleştirilebilir düğümler olarak adlandırılan tek bir işlemde birden fazla düğümü yönetme yeteneğidir. Bu özellik, daha verimli kaynak kullanımına ve aynı süreç içindeki düğümler arasında daha hızlı iletişime olanak tanır. Genel olarak, ROS2'deki düğüm yönetimi, ROS1'e kıyasla düğümleri yönetmek için daha yapılandırılmış ve verimli bir yol sağlayarak bir sistemdeki düğümlerin daha iyi kontrol edilmesine ve izlenmesine olanak tanır.

ROS'ta (Robot İşletim Sistemi), başlatma dosyaları tek bir komutla birden fazla düğümü başlatmak ve yönetmek için kullanılır. ROS1'de, başlatma dosyaları XML biçiminde yazılır ve bir ROS sisteminin çalıştırılmasıyla ilgili parametreleri, düğümleri ve diğer özellikleri tanımlamak için kullanılabilirler. ROS2'de, başlatma dosyaları hala birden fazla düğümü başlatmak ve yönetmek için kullanılır, ancak XML yerine Python ile yazılırlar. Geliştiriciler artık koşullu ifadeler, döngüler ve XML'de bulunmayan diğer yapıları kullanabildiğinden, bu durum başlatma dosyasına daha fazla esneklik ve güç sağlar. Ayrıca, ROS2 başlatma dosyaları, düğümlerin daha sağlam ve öngörülebilir bir şekilde yönetilmesine olanak tanıyan yeni yaşam döngüsü düğümleri konseptiyle çalışacak şekilde tasarlanmıştır. Başlatma dosyası her düğümün istenen durumunu

belirtebilir ve yaşam döngüsü yöneticisi her düğümün doğru duruma geçmesini sağlayarak daha iyi hata toleransı ve daha kolay hata ayıklama sağlar. Genel olarak, ROS2'de başlatma dosyaları yazmak ROS1'e göre daha güçlü ve esnektir ve düğümlerin daha sağlam ve öngörülebilir yönetimini sağlamak için yeni yaşam döngüsü düğümleri konseptinden yararlanır.

2.11.10. ROS2'de Sunulan Yenilikler

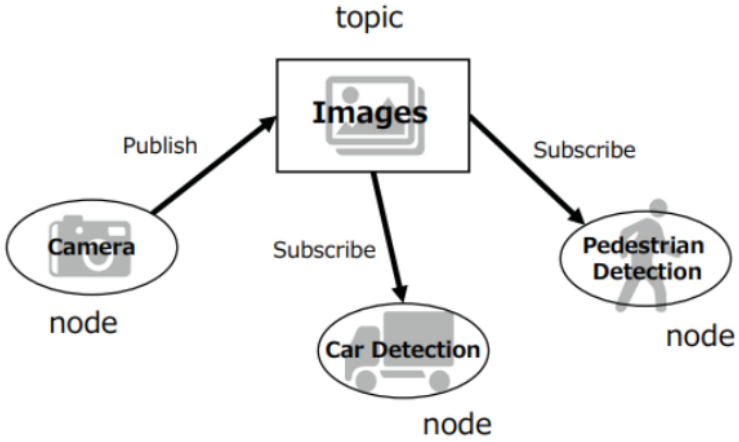


Şekil 2.7. ROS1/ROS2 mimarisi (Blass ve diğ, 2021)

ROS2, ROS1'e göre önemli bir yükseltmedir ve birçok yeni özellik ve iyileştirme getirmektedir. İşte ROS2'deki en dikkat çekici yeni özelliklerden bazıları: Gerçek zamanlı destek: ROS2, geliştiricilerin iş parçacığı yürütme ve önceliğini kontrol etmesine olanak tanıyan daha deterministik iş parçacığı modelleri sunar ve bu da onu zaman açısından kritik uygulamalar için uygun hale getirir. Çoklu dil desteği: ROS2, dil desteğini C++ ve Python'un ötesinde diğer programlama dillerini de içerecek şekilde genişletti. Geliştirilmiş güvenlik: ROS2, sistemi kötü niyetli saldırılardan korumak için şifreleme ve kimlik doğrulama mekanizmaları gibi çeşitli güvenlik özellikleri sunar. Esnek ve modüler mimari: ROS2'nin mimarisi daha modüler ve esnek olacak şekilde tasarlanmış olup, sistemin ek yükünü azaltan granüler bir bileşen setine sahiptir. Yaşam döngüsü yönetimi: ROS2, geliştiricilerin bir düğümün durum geçişlerini kontrol etmelerine olanak tanıyan yaşam döngüsü yönetimi adı verilen yeni bir kavram sunar. Daha iyi performans: ROS2, ROS1'e kıyasla daha iyi performansa ve daha düşük

gecikme süresine sahiptir, bu da onu gerçek zamanlı uygulamalar için daha uygun hale getirir. Geliştirilmiş hata ayıklama ve iç gözlem: ROS2, geliştiricilerin sorunları tanımlamasını ve hata ayıklamasını kolaylaştıran yeni bir günlük kaydı çerçevesi de dahil olmak üzere daha iyi hata ayıklama ve iç gözlem araçları sağlar. Bu özellikler ve iyileştirmeler akademik literatürde iyi bir şekilde belgelenmiştir, örneğin (Blass ve diğ, 2021), (Starkman, 2018), (Pimentel ve Aquino, 2019). Genel olarak, bu makaleler ROS2'de ROS1'e kıyasla yapılan önemli iyileştirmeleri göstermekte ve onu modern robotik uygulamaları için daha uygun hale getirmektedir. ROS2 daha esnek ve modüler bir mimari, daha iyi güvenlik ve gelişmiş performans sunarak gelecekteki robotik araştırma ve geliştirme çalışmaları için umut verici bir platform haline gelmiştir. Otonom sürüş araçları gibi gerçek zamanlı dağıtılmış gömülü sistemler, son yıllarda giderek daha karmaşık ve çeşitli hale gelmiştir. Otonom sürüş, 3 Kasım 2007'deki DARPA Urban Challenge'dan bu yana dikkat çekmektedir (Vinoski, 1997).ROS, yazılım geliştiricilerin yayınlama/abone olma aktarımı, çoklu kütüphaneler (OpenCV ve Point Cloud Library (PCL) gibi) dahil olmak üzere robotik uygulamalar oluşturmasını sağlayan araçlar sağlayarak üretkenliği artırır. Bununla birlikte, ROS gerçek zamanlı operasyonel gereksinimleri karşılamamakta ve yalnızca birkaç işletim sisteminde çalışmaktadır. Ek olarak, ROS hata toleransı, son tarihler veya süreç senkronizasyonunu garanti edemez. Dahası, ROS önemli kaynaklar (örneğin, CPU, bellek, ağ bant genişliği, iş parçacıkları ve çekirdekler) gerektirir ve son teslim tarihlerini karşılamak için bu kaynakları yönetemez. Bu sınırlamalar nedeniyle, ROS gerçek zamanlı gömülü sistemler için uygun değildir. Bu nedenle, ROS geliştiricileri de dahil olmak üzere birçok araştırma topluluğu, ROS'un gerçek zamanlı gömülü sistemler için sınırlamalarını ele almak için çeşitli çözümler önermiş ve değerlendirmiştir (Santos ve diğ., 2013). Ancak, bu çözümler ROS'un gerçek zamanlı gömülü sistemler için sınırlamalarını ele almakta yetersiz kalmaktadır. Daha geniş ROS topluluğunun ihtiyaçlarını karşılamak için ROS, ROS2'ye önemli ölçüde yükseltilecektir. ROS2 aşağıdaki yeni kullanım durumlarını dikkate alacaktır: gerçek zamanlı sistemler, küçük gömülü platformlar (sensör düğümleri gibi), optimal olmayan ağlar ve çapraz platformlar (Linux, Windows, Mac, Gerçek Zamanlı İşletim Sistemi (RTOS) ve işletim sistemi olmayan) (Pardo-Castellote, 2003). Bu yeni kullanım durumlarının gereksinimlerini karşılamak için, ROS'un mevcut sürümü (bundan sonra ROS1 olarak

anılacaktır) kullanıcı arayüzü API'lerini geliştirmek ve Veri Dağıtım Hizmeti (DDS), Zeroconf, Protokol Tamponları, ZeroMQ, Redis ve WebSockets gibi yeni teknolojileri dahil etmek üzere yeniden yapılandırılacaktır.



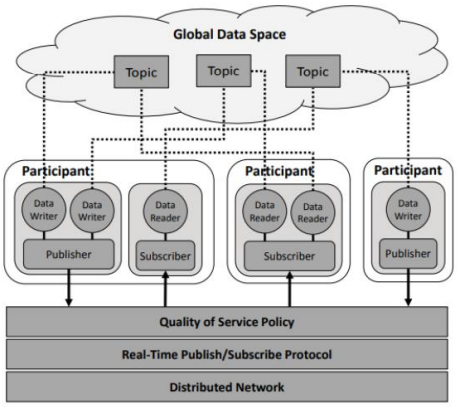
Şekil 2.8. ROS1/ROS2 Mimarisi (Pardo-Castellote, 2003)

2.12. Veri Dağıtım Hizmeti (DDS)

Veri Dağıtım Hizmeti (DDS), dağıtılmış gerçek zamanlı sistemler için bir ara katman protokolüdür. Farklı platformlarda ve programlama dillerinde çalışıyor olsalar bile, bir sistemin farklı bileşenlerinin birbirleriyle iletişim kurması için standart bir yol sağlar. DDS genellikle Endüstriyel Nesnelerin İnterneti (IIoT) ve güvenilirlik ve gerçek zamanlı performansın önemli olduğu diğer kritik görev sistemleri bağlamında kullanılır. DDS, ROS'a benzer bir yayınla-abone ol modeline dayanmaktadır. Yayıncıların ve abonelerin belirli bir veri türünü temsil eden konular aracılığıyla iletişim kurduğu konu tabanlı bir iletişim yaklaşımı kullanır. Yayıncılar bir konuda veri yayınlar ve aboneler bu konuya abone olarak verileri alır. DDS, farklı uygulamaların gereksinimlerini karşılamak için yapılandırılabilen çok çeşitli QoS (Hizmet Kalitesi) politikalarını destekler. Bu politikalar, diğerlerinin yanı sıra güvenilirlik, dayanıklılık, sıralama ve geçmişi içerir. DDS ayrıca sistem bileşenlerinin otomatik olarak bulunmasını ve yapılandırılmasını sağlayan veri filtreleme, içerik tabanlı yönlendirme ve keşif için yerleşik mekanizmalar sağlar. DDS, Object Management Group (OMG) tarafından sürdürülen standart bir protokoldür. OpenDDS, RTI Connext ve eProsima Fast DDS dahil olmak üzere farklı satıcılardan çeşitli uygulamalara sahiptir. Bu uygulamalar birlikte çalışabilir olacak şekilde tasarlanmıştır, bu da farklı satıcıların DDS bileşenlerinin birbirleriyle sorunsuz bir şekilde iletişim kurabileceği anlamına gelir.

Özetle, DDS dağıtık gerçek zamanlı sistemler için yayınla-abone ol modeli sağlayan bir ara katman protokolüdür. Veri filtreleme, içerik tabanlı yönlendirme ve keşif için çok çeşitli QoS politikaları ve yerleşik mekanizmalar sunar. DDS, çeşitli satıcıların farklı uygulamaları arasında birlikte çalışabilen standart bir protokoldür. Veri merkezli yayınla-abone ol (DCPS), dağıtık bir sistemdeki dağıtık düğümler arasında veri alışverişini kolaylaştıran bir modeldir. Düğümler arasında güvenilir ve verimli iletişim sağlamak için Veri Dağıtım Hizmeti (DDS) gibi ara katman sistemlerinde uygulanan bir mesajlaşma modelidir. DCPS, veri yayıncılarının bir konuya veri gönderdiği ve abonelerin bu konudan veri aldığı yayınla-abone ol paradigmasına dayanır. Ancak, geleneksel yayınlama-abonelikten farklı olarak DCPS veri merkezlidir, yani odak noktası veri alışverişi yapan düğümler yerine alışverişi yapılan verilerdir. Bu da daha esnek ve ölçeklenebilir bir iletişim modeli sağlar. DCPS'de veriler, verilerin yapısını ve içeriğini tanımlayan bir veri türü ile temsil edilir. Bir konuyu yayınlayan veya konuya abone olan her düğüm aynı veri türünü kullanarak verilerin tutarlı ve öngörülebilir bir şekilde değiş tokuş edilmesini sağlar. Bir yayıncı bir konuya veri gönderdiğinde, bu türdeki verileri almakla ilgilendiğini ifade eden tüm abonelere iletilir. DCPS ayrıca onu gerçek zamanlı ve dağıtılmış sistemler için uygun hale getiren çeşitli özellikler içerir. Örneğin, verilerin güvenilirlik, gecikme ve verim gereksinimlerini tanımlayan Hizmet Kalitesi (QoS) ilkelerini destekler. Bu, uygulamaların belirli verilere öncelik vermesine ve zamanında ve güvenilir bir şekilde teslim edilmesini sağlamasına olanak tanır. Genel olarak DCPS, dağıtık sistemler için esnek ve verimli bir iletişim mekanizması sağlayan güçlü bir mesajlaşma modelidir. Veri merkezli yaklaşımı ve QoS desteği, onu özellikle gerçek zamanlı ve görev açısından kritik uygulamalar için uygun hale getirmektedir. DDS (Veri Dağıtım Hizmeti) Hizmet Kalitesi (QoS) ilkeleri, DDS varlıkları arasındaki iletişimin davranışını tanımlayan bir dizi parametredir. Bu politikalar, verilerin nasıl teslim edilmesi, nasıl dönüştürülmesi ve sistemin belirli olaylara nasıl tepki vermesi gerektiğini belirtir. ROS2'de DDS, süreçler arası iletişim için kullanılan ara yazılımdır. DDS'de çeşitli QoS politikaları mevcuttur ve bunlar üç kategoride gruplandırılabilir: güvenilirlik, dayanıklılık ve geçmiş. Güvenilirlik ilkeleri, veri teslimatının nasıl ele alınması gerektiğini belirtir ve verilerin kayıp veya çoğaltma olmadan teslim edilmesini sağlar. Dayanıklılık ilkeleri verilerin nasıl saklanacağını ve korunacağını belirleyerek iletişim arızaları durumunda bile verilerin korunmasını sağlar. Son olarak, geçmiş

politikaları depolanabilecek veri miktarını ve bu verilere nasıl erişileceğini tanımlar. ROS2'deki QoS politikalarının bazı örnekleri şunlardır: Güvenilirlik: GÜVENİLİR ilkesi, tüm verilerin kayıp veya çoğaltma olmadan teslim edilmesini sağlar. BEST-EFFORT ilkesi bir miktar veri kaybına veya çoğaltılmasına izin verir ancak daha verimlidir. Dayanıklılık: VOLATILE ilkesi, bir iletişim hatası olması durumunda verilerin kaybolacağı anlamına gelir. TRANSIENT-LOCAL ilkesi verilerin her abone düğümünde yerel olarak depolanmasını sağlarken TRANSIENT ilkesi verilerin merkezi bir konumda depolanmasını sağlar. Geçmiş: KEEP-LAST ilkesi depolanacak örnek sayısını belirtir ve KEEP-ALL ilkesi tüm verilerin depolanması gerektiğini belirtir. Bu QoS politikaları, ROS2'deki her yayıncı ve abone için ayarlanabilir ve düğümler arasındaki iletişimin ince ayarının yapılmasına olanak tanır. Geliştiriciler, uygun QoS politikalarını belirleyerek verilerin güvenilir ve verimli bir şekilde teslim edilmesini, depolanmasını ve gerektiğinde erişilmesini sağlayabilir. Genel olarak, QoS politikaları DDS ve ROS2'nin önemli bir parçasıdır ve düğümler arasında özelleştirilebilir ve esnek iletişim sağlar.

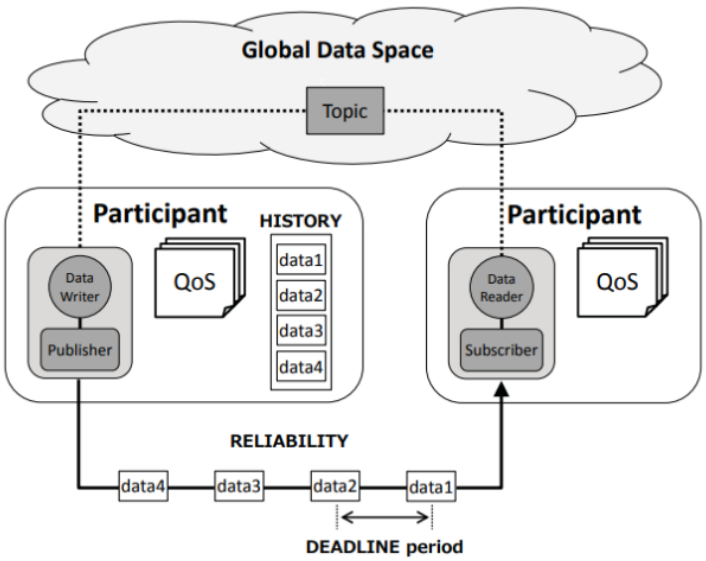


Şekil 2.9. Veri merkezli yayınla-abone ol (DCPS) modeli.

2.13. Navigasyon2 Tasarımı

Navigation2, otonom mobil robot navigasyonu için esnek ve modüler bir mimari sağlayan ROS2 için bir navigasyon yığınıdır. Çok çeşitli robot platformları, sensörler ve ortamlarla çalışmak üzere tasarlanmıştır. Navigation2, küresel bir planlayıcı, yerel bir planlayıcı, bir kontrolör ve bir kurtarma davranışı sistemi dahil olmak üzere birkaç temel bileşen sağlar. Küresel planlayıcı, bir hedef konuma ulaşmak için üst düzey bir plan oluşturmaktan sorumluyken, yerel planlayıcı, üst düzey planı takip etmek için

düşük seviyeli bir hareket planı oluşturmaktan sorumludur. Kontrolör düşük seviyeli hareket planını yürütmekten sorumluyken, kurtarma davranışı sistemi navigasyon sırasındaki hataları veya arızaları tespit etmekten ve bunlardan kurtulmaktan sorumludur. Navigation2, 2D ve 3D lidarlar, RGB-D kameralar ve diğer sensörler dahil olmak üzere çeşitli sensörlerle çalışmak üzere tasarlanmıştır. Doluluk ızgaraları, maliyet haritaları ve voksel haritaları dahil olmak üzere çeşitli harita türlerini destekler. Navigation2, navigasyon yığınının farklı bileşenleri arasında iletişim kurmak için Veri Dağıtım Hizmeti (DDS) dahil olmak üzere ROS2 iletişim sistemini kullanır. DDS, geliştiricilerin düğümler arasındaki iletişimin güvenilirliğini, gecikmesini ve bant genişliğini kontrol etmesine olanak tanıyan esnek bir Hizmet Kalitesi (QoS) sistemi sağlar. Navigation2 ayrıca geliştiricilerin navigasyon yığınının davranışını kendi özel robotlarına ve ortamlarına göre özelleştirmelerine olanak tanıyan bir dizi yapılandırma seçeneği ve eklenti sunar. Örneğin, geliştiriciler farklı küresel planlayıcılar veya yerel planlayıcılar seçebilir veya kurtarma davranışı sistemini farklı arıza türlerine farklı yanıt verecek şekilde yapılandırabilir. Genel olarak Navigation2, otonom mobil robotlar için güçlü ve esnek bir navigasyon yığını sağlar ve geliştiricilerin davranışı kendi özel ihtiyaçlarına göre özelleştirmelerine olanak tanıyan modüler bir mimariye sahiptir.



Şekil 2.10. DDS QoS Politikası (Macenski ve diğ, 2020b)

3. İLGİLİ ÇALIŞMALAR

3.1. Slam Karşılaştırma Algoritmaları ile İlgili Çalışma

Bilindiği gibi, eşzamanlı konum belirleme ve haritalama (SLAM), bir robotun çevresinin bir haritasını elde ederken aynı zamanda kendisini bu haritaya göre konumlandırması işlemidir (Thrun, 2007). Harita oluşturulduktan sonra, bir sonraki önemli soru "Sonuçlar nasıl değerlendirilir?" sorusudur. SLAM karşılaştırması yeni bir konu değildir. Örneğin, (Ibragimov ve Afanasyev, 2017) ve (Filipenko ve Afanasyev ,2018) gibi makaleler, zemin gerçeği olarak lidar tabanlı SLAM yörüngelerini kullanarak farklı görsel (monoküler ve stereo kameralar) ve lazer (derinlik sensörü) sensörlerinden hesaplanan yörüngelerin karşılaştırılmasını sağlar. Makale (Giubilato ve di˘g. 2018) ofis tarzı bir ortamda otonom olarak yönlendirilen bir robotun iç mekan navigasyonu için monoküler SLAM yöntemlerinin (ORBSLAM, REMODE, LSD-SLAM ve DPPTAM) karşılaştırmalı bir analizini sunmaktadır. Makale (Huletski ve diğ., 2015) görsel SLAM yöntemlerini (ORB-SLAM, LSD-SLAM, L-SLAM ve OpenRatSLAM) değerlendirmek için TUM RGB-D ölçütünü kullanır. Makale (Giubilato ve diğ. 2018), zemin gerçeği olarak lidar tabanlı Hector SLAM taramalarını kullanarak ROS uyumlu beş stereo görsel SLAM yöntemini (ORB-SLAM2, LIBVISO2, SPTAM, RTAB-MAP, ZED-VO) karşılaştırmaktadır. Düzeltilmiş yörüngelerdeki hatayı ölçmek için metrik tabanlı bir SLAM yaklaşımının sonuçlarını analiz etmek için bir çerçeve önerilmiştir. Bu metrikler, yalnızca robot pozları arasındaki göreli yer değiştirmelere dayandıkları için küresel bir referans çerçevesine dayalı hata hesaplamalarını kullanan yaklaşımlardaki önemli eksikliklerin üstesinden gelmektedir. Bu çalışma (Fang ve diğ., 2016) tarafından geliştirilen ölçümleri kullanarak ToF kamera verilerini işleyen üç ROS tabanlı SLAM algoritması (GMapping, KartoSLAM ve Hector SLAM) tarafından oluşturulan haritaları karşılaştırmaktadır. Özellik yakalama kullanarak çeşitli kriterlere dayalı olarak 2D lazer SLAM algoritmalarının kalitesini nicel olarak değerlendirmek için bir çerçeve sunmaktadır. Bu çerçeve ROS tabanlı SLAM yöntemleri ile test edilmiştir: GMapping, tinySLAM, vinySLAM, Google Cartographer ve Hector SLAM. Bir başka çalışma (Santos ve diğ. Ö 2013), gerçek dünya deneysel verileri için ROS'ta bulunan beş lazer tabanlı 2D SLAM tekniği üzerinde bir çalışma yürütmüş ve k-en yakın komşu tabanlı metrikleri kullanarak haritalar oluşturmuştur: Hector SLAM, Gmapping, KartoSLAM, CoreSLAM ve

LagoSLAM. Başka bir araştırma (Rojas-Fernández ve diğ., 2018), RGB-D Kinect sensör verilerini ve Köşe Ortalama Kare Hatası (CMSE) ve Harita Kenar Uzunluklarındaki Toplam Fark gibi metrikleri kullanarak Gmapping, Hector SLAM ve CRSM SLAM (SDS) için ölçümler önermiştir.

3.2. Yerel Planlayıcı ile İlgili Çalışmalar

Mobil robot yol planlamasında araştırmanın önemli yönlerinden biri, gerçek zamanlı yeniden planlama ile düzgün robot hareketine izin veren küresel yörünge planlayıcılarının araştırılmasıdır. Bu çözümün optimal olmayabileceği sonucuna varmışlardır. Sezgisel algoritmaların klasik algoritmalara göre artan popülerliğini gösteren (Raja ve Pugazhenthi, 2012); (Koubaa ve diğ., 2018) ve (Masehian ve Sedighizadeh, 2007) dahil olmak üzere çeşitli çalışmalar şu anda mevcut olan yöntemleri incelemiştir. Ek olarak, (Marin-Plaza ve diğ., 2018) yerel bir TEB planlayıcısı tarafından üretilen yörüngelerin bir analizini sunmaktadır. Testler gerçek bir Ackerman tipi araç üzerinde gerçekleştirilmiştir. Toplanan veriler, küresel ve yerel planlar arasındaki hatanın belirlenmesine olanak sağlamıştır. (Kim ve Pineau, 2016) çalışmasında, robotun engelleri aşmaya başladığı mesafeyi, bir engele olan en kısa mesafeyi ve hedefe ulaşmak için gereken süreyi belirleyen bir parametrenin yerel planlayıcıları karşılaştırmak için kullanılabileceğini öne sürmektedir. (Gurel ve di diğ.)'da TurtleBot3 platformu, henüz geliştirme aşamasında olan global RRT* planlayıcısını test etmek ve ROS'ta bulunan Dijkstra algoritmasıyla karşılaştırmak için kullanıldı. (Kumar ve diğ., 2016) bölümünde, noktalar arasındaki mesafeyi değerlendirmek için kullanılan fonksiyona bağlı olarak küresel planı belirleyen A* algoritmasının hızını değerlendirmek için bir simülasyon gerçekleştirilmiştir. Son olarak, (Pittner ve diğ., 2018) DWA, EBand ve TEB yerel planlayıcılarını ürettikleri yollar açısından karşılaştırmıştır. Bu planlayıcıların davranışları, engellerin olmadığı statik bir ortamda, haritada görülebilen statik engellerde ve dinamik engellerde incelenmiştir. Test sonuçları TEB'in en iyi performansı gösterdiğini, DWA'nın dinamik engellerden kaçınmakta zorlandığını ve EBand'ın statik engellerin varlığında hatalar ürettiğini göstermiştir.

3.3. Navigasyon Yığınıyla İlgili Çalışma

Derviş, ofis ortamında etkili bir navigasyon sistemi uygulayan ilk robotlardan biriydi (Nourbakhsh ve diğ., 1995). Ofis Teslimat Robotu Yarışması için bir koridor sahnesinde gezinmek için çevrenin kaba bir olasılıksal temsilini kullanarak bir sonar halka kullandı. Kısa bir süre sonra, (Burgard ve diğ., 1999) ve (Thrun ve diğ., 1999), çeşitli planlama yaklaşımları kullanarak çevrelerinde yerelleşmek için öncelikle olasılıksal doluluk ızgaralarını kullanan müze robotları RHINO ve MINERVA'yı tanıttı. Engellerden kaçınmak için DWA kullanılmıştır (Fox ve diğ., 1997). Bu robotlar konumlandırma ve navigasyon için 2B çevresel temsillere dayanıyordu ve bu da düzlemsel olmayan engellerle çarpışmalara yol açabiliyordu. (Marder-Eppstein ve diğ., 2010) bir ortamda gezinmek için eğimli 2D lazer tarayıcılardan elde edilen 3D bilgileri kullanan bir navigasyon sistemi ve çevresel temsil sundu. A* ve DWA gibi en eski tekniklerin çoğu planlama ve engellerden kaçınma için kullanılmıştır. Zamanla, eğimli 2D lazer tarayıcıların yerini her yerde ucuz 3D derinlik kameraları ve lazer tarayıcılar almıştır. Yaklaşımın bazı formülasyonları, düşük doğruluk ve etkilere sahip 2D esnek lazer konfigürasyonu tarafından görüntülenen geometri için özelleşmiştir. Ayrıca, seyrek, uzun menzilli lazer tarayıcıların piyasaya sürülmesiyle birlikte, saf ışın yayma yöntemleri artık dinamik bir sahnedeki boş alanı etkili bir şekilde temizleyememektedir. STVL, bu tür tekniklerin yerine kullanılan birçok seyrek ve uzun menzilli sensör için daha ölçeklenebilir bir alternatif oluşturdu (Macenski ve diğ., 2020b). 2013 yılında, bir açık hava robot navigasyon deneyi sunulmuştur (Kümmerle ve diğ., 2013). Bu çalışma, Almanya sokaklarında 7 km gezinmek için kullanılan haritalama, yerelleştirme ve basit dinamik engel algılama yöntemlerine odaklanmıştır. Sonlu durum makinelerinin (FSM) (Abiyev ve diğ., 2016) yerini alan Davranış Ağaçlarının (BT) robotikte görev planlamasında başarılı olduğu gösterilmiştir. Artan sayıda durum ve geçişle birlikte, karmaşık davranışları ve çok adımlı görevleri FSM olarak modellemek gerçek dünya görevleri için zorlayıcı hale gelebilir. Başka bir FSM ile zor olabilecek bir futbol oyunu için hareket ve atış taktiklerini modellemek için davranış ağaçlarının kullanımını başarıyla göstermektedir. BT'ler ayrıca robot görev yürütme, robot manipülasyonu ve karmaşık mobil manipülatör görevleri alanında da yoğun bir şekilde uygulanmıştır (Colledanchise ve Ögren ,2018). Ayrıca, davranış ağaçları yeniden kullanılabilir ilkel öğelerle kolayca değiştirilebilir ve programlanmış mantık gerektirmeden çalışma

zamanında yüklenebilir. Bu nedenle, yaklaşımımızda davranış ağaçları navigasyon çerçevesi (Macenski ve diğ., 2020a) için birincil yapı olarak kullanılmıştır. Testler gerçek bir Ackerman tipi araç üzerinde gerçekleştirilmiştir. Toplanan veriler, küresel ve yerel planlar arasındaki hatanın belirlenmesine olanak sağlamıştır. Yazarlar (Kim ve Pineau ,2016) makalesinde, robotun engelleri aşmaya başladığı mesafeyi, bir engele olan en kısa mesafeyi ve hedefe ulaşmak için gereken süreyi belirleyen bir parametrenin yerel planlayıcıları karşılaştırmak için kullanılabileceğini öne sürmektedir. TurtleBot3 platformu, henüz geliştirme aşamasında olan global RRT* planlayıcısını test etmek ve ROS'ta bulunan Dijkstra algoritmasıyla karşılaştırmak için kullanıldı. (Kumar ve diğ., 2016) bölümünde, noktalar arasındaki mesafeyi değerlendirmek için kullanılan fonksiyona bağlı olarak küresel planı belirleyen A* algoritmasının hızını değerlendirmek için bir simülasyon gerçekleştirilmiştir. Son olarak, (Pittner ve diğ., 2018) DWA, EBand ve TEB yerel planlayıcılarını ürettikleri yollar açısından karşılaştırmıştır. Bu planlayıcıların davranışları, engellerin olmadığı statik bir ortamda, haritada görülebilen statik engellerde ve dinamik engellerde incelenmiştir. Test sonuçları TEB'in en iyi performansı gösterdiğini, DWA'nın dinamik engellerden kaçınmakta zorlandığını ve EBand'ın statik engellerin varlığında hatalar ürettiğini göstermiştir.

4. DENEYSEL ÇALIŞMALAR VE BULGULAR

ROS (Robot Operating System) ve ROS 2, robotik uygulamalarında yaygın olarak kullanılan iki önemli çerçevedir. Bu çerçeveler, planlanan yolların ve robotların katedilen yolların karşılaştırılması gibi önemli işlevleri destekler. ROS, yaygın olarak ROS navigasyon yığını kullanılarak yol planlaması ve navigasyonu gerçekleştirir. Planlanan yol, robotun hedefe ulaşmak için önceden belirlenen yolunu temsil eder. Bu yol, haritalama, engel algılama ve planlama algoritmaları kullanılarak hesaplanır. Robot, bu planlanan yolu takip etmeye çalışır ve hedefe ulaşmak için ilerler. ROS 2 ise ROS'un geliştirilmiş bir sürümüdür ve daha modern bir iletişim altyapısı olan DDS (Data Distribution Service) standardını kullanır. ROS 2'nin yayın-abonelik modeli, daha güçlü ve esnek bir veri iletişimi sağlar. Planlanan yol ile katedilen yol arasındaki karşılaştırma, gerçek dünya uygulamalarında ROS ve ROS 2'nin performansını değerlendirmek için önemlidir. Bu karşılaştırma, robotun belirli bir görevi yerine getirirken planlanan yolu nasıl takip ettiğini ve katettiği yolun ne kadarını gerçekleştirdiğini ortaya koyar. Test senaryoları, farklı dinamik engellerin olduğu durumları simüle edebilir. Bu testlerde, robotun karşılaştığı engellerle nasıl başa çıktığı ve gerektiğinde yolu nasıl değiştirdiği değerlendirilir. Bu testler aynı zamanda ROS ve ROS 2'nin iyileştirilmiş kurtarma davranışlarını göstermek için kullanılır. Sınırlı bir alanda güvenliği sağlamak için, robotlar çarpışma algılama mekanizmalarıyla donatılırlar. Robot, bir engelle veya beklenmedik bir nesneyle çarpışma tehdidi algıladığında, kurtarma davranışını aktif hale getirir. Bu davranış, robotun durma, geri gitme veya alternatif bir yol planlama gibi çarpışmadan kaçınma önlemleri almasını sağlar. Ek olarak, tekrarlayan görev senaryolarında, belirli bir görevi tamamladıktan sonra, robot kurtarma davranış moduna girer ve görevin başarılı bir şekilde tamamlandığını doğrular. Eğer sonuç beklenenden farklı ise, robot düzeltilmesi gereken eylemleri başlatır. Görevi yeniden gerçekleştirmeye çalışabilir veya bir operatöre ek inceleme ve müdahale için uyarı gönderebilir. Bu sayede robot, sınırlı ortamda tekrarlayan görevleri yürüterek genel verimlilik ve güvenilirlik sağlamış olur.

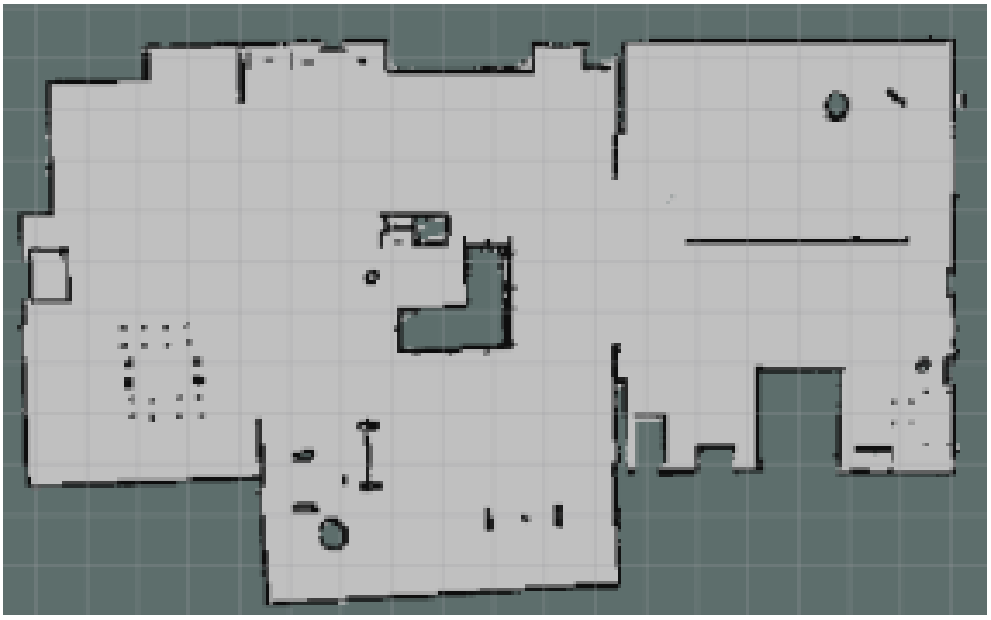
ROS 1 ve ROS 2'de, varsayılan küresel planlayıcı, "move-base" paketinde kullanılan "Global Planner" eklentisidir. Bu eklenti genellikle robotun başlangıçtan hedefe en kısa yolu bulmak için Dijkstra algoritmasını (A* algoritmasının bir varyantı) kullanır ve "Costmap2D" paketi tarafından oluşturulan işgal tablosunu dikkate alır. ROS 1 ve ROS

2'de küresel planlayıcıyı değiştirmek için benzer adımlar izlemeniz gerekmektedir: ROS 1: ROS 1'de küresel planlayıcıyı değiştirmek için "nav-core::BaseGlobalPlanner" arayüzüne uygun özel bir küresel planlayıcı eklentisi oluşturmalısınız. Bu arayüz, küresel planlayıcı eklentisi için gereken yöntemleri tanımlar, bu da küresel bir plan oluşturmayı içerir. Özel küresel planlayıcınızı oluşturduktan sonra, ilgili yapılandırma dosyasını veya başlatma dosyasını güncelleyerek "move-base" düğümünün özel eklentinizi kullanmasını sağlayabilirsiniz. ROS 2: ROS 2'de küresel planlayıcıyı değiştirmek de ROS 1 ile benzerdir. ROS 2'de, "navigation2" paketinde kullanılan varsayılan küresel planlayıcı "NavFn"dir ve bu da Dijkstra algoritmasını kullanır. Özel bir küresel planlayıcı oluşturmak için, ROS 2'de "nav2-core::GlobalPlanner" arayüzünü uygulayan bir küresel planlayıcı yapmalısınız. Bu arayüz, ROS 2 için bir küresel planlayıcı için gereken yöntemleri tanımlar. Özel küresel planlayıcınız hazır olduğunda, "navigation2" paketini yapılandırarak özel planlayıcınızı kullanmasını sağlayabilirsiniz. ROS 1 ve ROS 2'de, küresel planlayıcıların desteklediği sadece varsayılan "Global Planner" veya "NavFn" planlayıcılarla sınırlı değildir. Kullanıcılar özel küresel planlayıcılar oluşturabilir veya ROS ekosisteminden mevcut üçüncü taraf küresel planlayıcılar arasından seçim yapabilir. Özel bir küresel planlayıcı entegre etme süreci, genellikle gerekli arayüzleri uygulamayı ve navigasyonla ilgili paketleri özel planlayıcıyı kullanacak şekilde yapılandırmayı içerir. ROS 1 ve ROS 2'de desteklenen özel küresel planlayıcıların çeşitli olabileceğini unutmamanız önemlidir. "Global Planner" veya "NavFn" planlayıcıları gibi varsayılan planlayıcılar popüler ve yaygın olarak kullanılırken, ROS topluluğu veya diğer kuruluşlar tarafından geliştirilen birçok başka küresel planlayıcı mevcuttur. "SBPLLatticePlanner," "TrajectoryPlannerROS," "TebLocalPlanner" gibi popüler küresel planlayıcılar ROS 1 ve ROS 2 tarafından desteklenir. Küresel planlayıcıların mevcudiyeti, ROS 1 ve ROS 2 arasında farklılık gösterebilir, çünkü iki sürüm farklı navigasyon paketleri ve arayüzlerine sahiptir. Ancak her iki sürüm de esneklik ve genişletilebilirlik sunar, böylece kullanıcılar özel ihtiyaçlarına ve tercihlerine uygun çeşitli küresel planlayıcıları entegre edebilir. Sonuç olarak, planlanan yol ile katedilen yolun karşılaştırılması, robotik uygulamalarda ROS ve ROS 2'nin performansını değerlendirmek ve iyileştirme potansiyellerini belirlemek için önemlidir. Bu karşılaştırma, daha iyi yol planlaması, navigasyon ve kurtarma davranışları için geliştirme fırsatları sunar.

4.1. Planlanan Yol İle Kat Edilen Yolun Karşılaştırılması



Şekil 4.1. Planlı yol ve gezilen yol Çardak Ortamı



Şekil 4.2. Planlanan yol ve seyahat edilen yol haritası

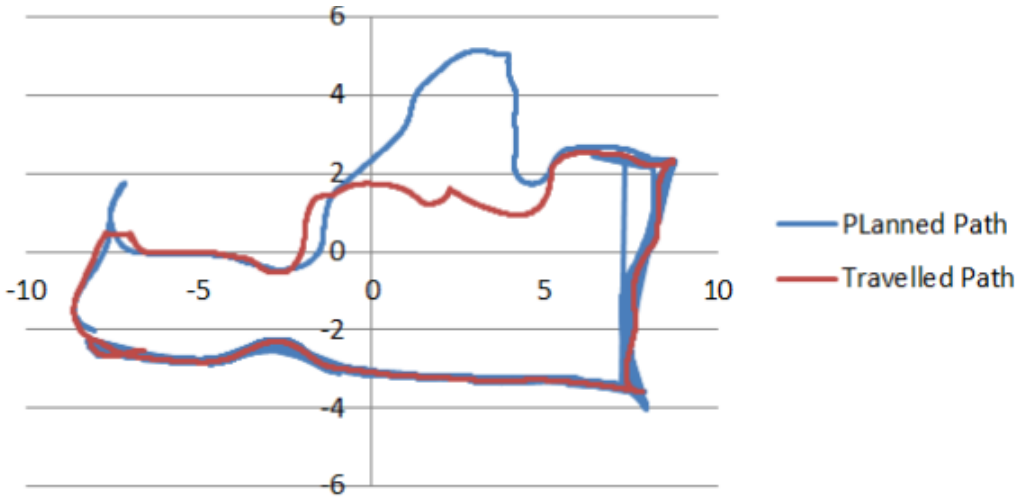
Gazebo simülasyon ortamında bulunan AWS RoboMaker Small House World Şekil 4.1'de görülmektedir. Bu ortam, testlerin yapıldığı ve planlanan yol ile gerçek yol arasındaki farkları değerlendirmek için kullanılan bir simülasyon ortamıdır. AWS RoboMaker Small House World, bir evin iç mekanını temsil eden bir simülasyon ortamıdır. Bu ortam, birçok farklı öğe, mobilya ve engellerle donatılmıştır. Robotun

çevreyi keşfetmesi, engelleri aşması ve hedefe ulaşması için çeşitli zorluklar sunar. Bu ortam, testlerin gerçekleştirildiği alanı görselleştirir ve robotun gerçek dünyadaki benzer bir ortamda karşılaşacağı zorlukları simüle eder. Planlanan yol ile gerçek yol arasındaki farkları değerlendirmek için bu ortamda çeşitli senaryolar test edilir. Testler, simülasyon ortamında yapıldığı için gerçek bir fiziksel tehlike veya maliyetle karşılaşma riski olmadan farklı parametrelerin etkilerini değerlendirebilir. Bu, planlanan yol algoritmalarının performansını değerlendirmek, geliştirmek ve karşılaşılabilecek sorunları önceden tespit etmek için önemlidir. Bu simülasyon ortamı, robotik uygulamaların geliştirilmesi ve test edilmesi için bir platform sağlar. Planlanan ve gidilen yolun karşılaştırılması için yapılan testlerde, AWS RoboMaker Small House World, gerçek dünya senaryolarını taklit eden ve ilgili zorlukları sunan bir ortam sağlamaktadır. Şekil 4.2'de, Rviz üzerinde turtlebot3 tarafından kullanılan SLAM (Simultaneous Localization and Mapping) Gmapping yöntemi ile oluşturulan AWS RoboMaker Small House haritası gösterilmektedir. Bu harita, turtlebot3 robotunun Gazebo-ROS simülasyon ortamında hareket ederken, çevresini haritalandırdığı ve lokalizasyonunu sağladığı bir temsilidir. SLAM algoritması, robotun sensörlerinden gelen verileri kullanarak çevrenin geometrik ve topolojik özelliklerini yakalar ve bu bilgileri bir haritada temsil eder. AWS RoboMaker Small House haritası, sanal bir evin iç mekanını yansıtan detaylı bir harita sunar. Mobilyalar, duvarlar, kapılar ve diğer yapısal unsurlar bu haritada görülebilir. Harita, robotun hareket edebileceği geçitlerin, engellerin ve açık alanların konumlarını gösterir. Harita, turtlebot3 robotunun hareket planlaması ve navigasyonunda kullanılır. Robot, planlanan yol algoritması kullanarak hedefe doğru hareket ederken, harita üzerindeki engelleri ve açık alanları dikkate alır. Bu şekilde, robot çevresini keşfedebilir, engelleri aşabilir ve güvenli bir şekilde hedefe ulaşabilir. Test senaryoları, harita üzerinde dinamik engellerin varlığını simüle eder. Yani, robotun hareketi sırasında engellerin ortaya çıkması veya hareket etmesi durumunda, yol planlaması güncellenir ve robot yeni bir yol belirler. Bu testler, robotun engellerle etkileşimini ve planlanan yol ile gerçek yol arasındaki farkları değerlendirmek için yapılmaktadır. Harita, Rviz üzerinde görselleştirilirken, turtlebot3 robotunun hareketi ve yol planlaması gerçek zamanlı olarak takip edilir. Bu, testlerin izlenmesi, analiz edilmesi ve sonuçların değerlendirilmesi için önemlidir.

4.1.1. Test Durumu 1: Planlanan ve Gerçekleşen Yol Karşılaştırması



Şekil 4.3. Planlanan Yol ve Kat Edilen Yol: Engelsiz Harita Hedefleri

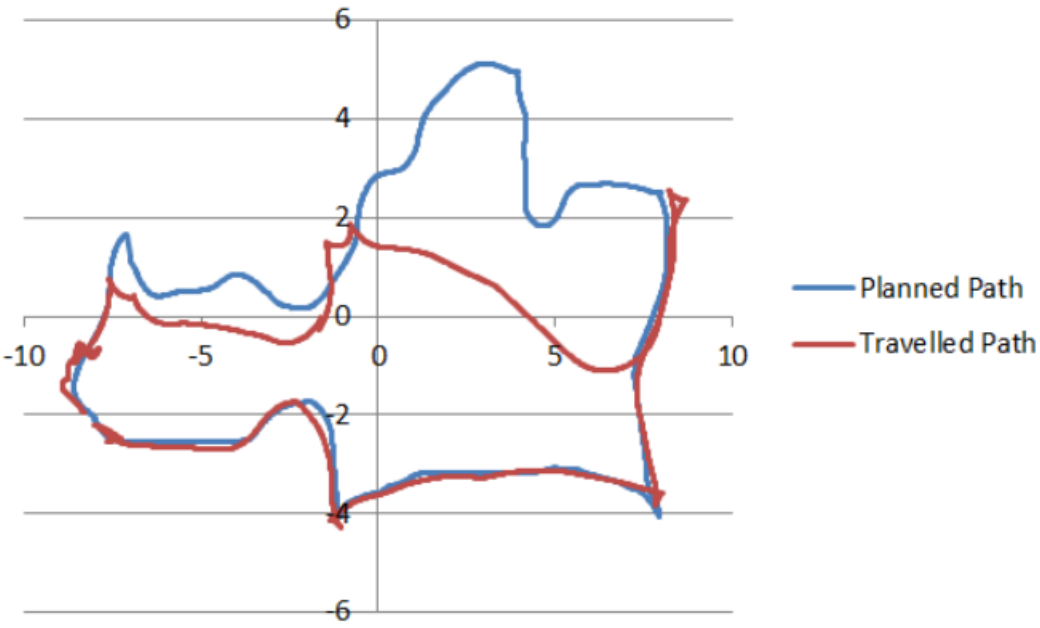


Şekil 4.4. Yol Karşılaştırması - Engelsiz Harita - Navstack Dwa

Şekil 4.3'te, engelsiz harita hedefleri kullanılarak yapılan planlanan yol ile katedilen yol karşılaştırması gösterilmektedir. Harita üzerinde herhangi bir engel olmadığı için turtlebot3'un hareket etmesi gereken hedefler kırmızı sayılarla işaretlenmiştir. Bu şekilde, planlanan yol ile gerçek yol arasındaki farklar değerlendirilebilir.

Şekil 4.4’de planlanan yolu ve gidilen yolu gösteren x ve y koordinatlarından oluşan bir grafik bulunmaktadır. Bu grafik, Python kodu kullanılarak oluşturulmuştur ve deney Şekil 4.3’te gösterilen test senaryosunda gerçekleştirilmiştir.

Şekil 4.5’de planlanan yolu ve gidilen yolu gösteren x ve y koordinatlarından oluşan bir grafik bulunmaktadır. Bu grafik, Python kodu kullanılarak oluşturulmuştur ve deney Şekil 4.3’te gösterilen test senaryosunda gerçekleştirilmiştir



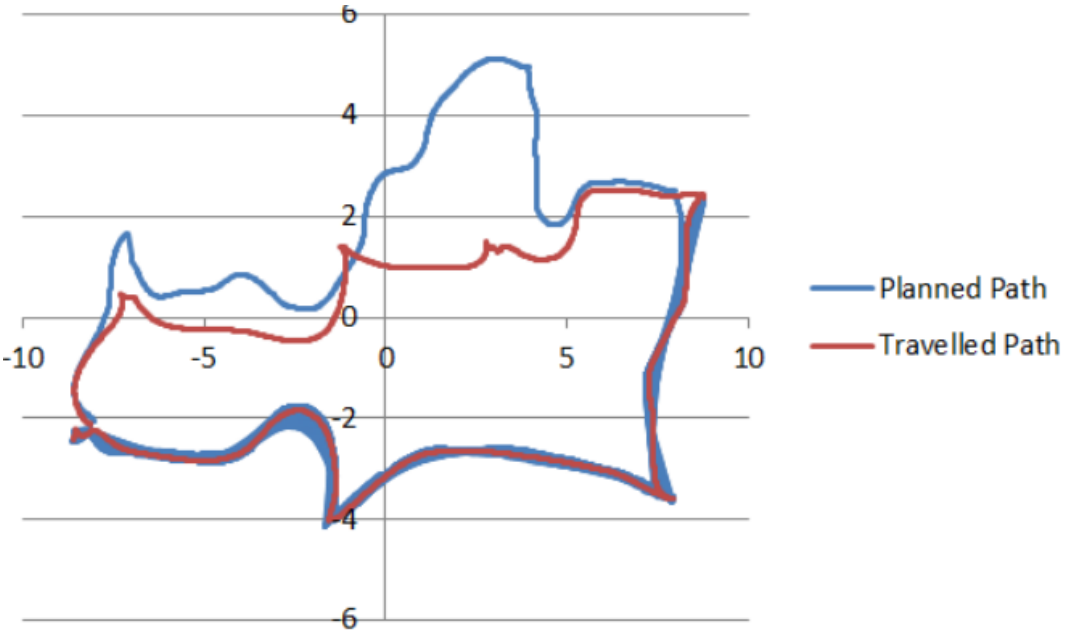
Şekil 4.5. Yol Karşılaştırması - Engelsiz Harita - Navs2tack Dwb

4.1.2. Test Durumu 2: Planlanan ve Gerçekleşen Yol Karşılaştırması

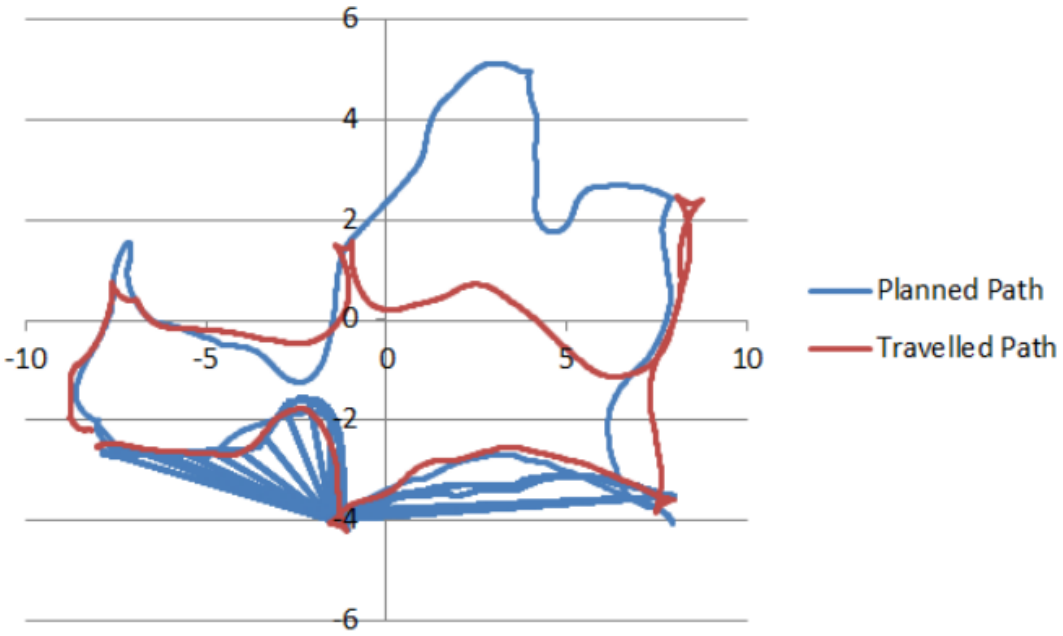
Şekil 4.6’da karşılaştırma için kullanılan haritayı gösteren bir görüntü bulunmaktadır. Haritada, mavi dikdörtgen, dinamik bir engelin hareket edebileceği alanı temsil etmektedir. Kırmızı sayılar ise turtlebot3 robotunun hareket etmesi gereken hedefleri göstermektedir. Bu harita üzerinde yapılan karşılaştırma, planlanan yol ile katedilen yol arasındaki farkları değerlendirmek için kullanılmıştır. Şekil 4.7’de planlanan yolu ve gidilen yolu gösteren x ve y koordinatlarından oluşan bir grafik bulunmaktadır. Bu grafik, Python kodu kullanılarak oluşturulmuştur ve deney Şekil 4.6'da gösterilen test senaryosunda gerçekleştirilmiştir. Şekil 4.8’de planlanan yolu ve gidilen yolu gösteren x ve y koordinatlarından oluşan bir grafik bulunmaktadır. Bu grafik, Python kodu kullanılarak oluşturulmuştur ve deney Şekil 4.8'de gösterilen test senaryosunda gerçekleştirilmiştir.



Şekil 4.6. Planlanan Yol ve Kat Edilen Yol: Harita Hedefleri (2 Engelle)



Şekil 4.7. Yol Karşılaştırması- 2 Engelli Harita - Navstack Dwa

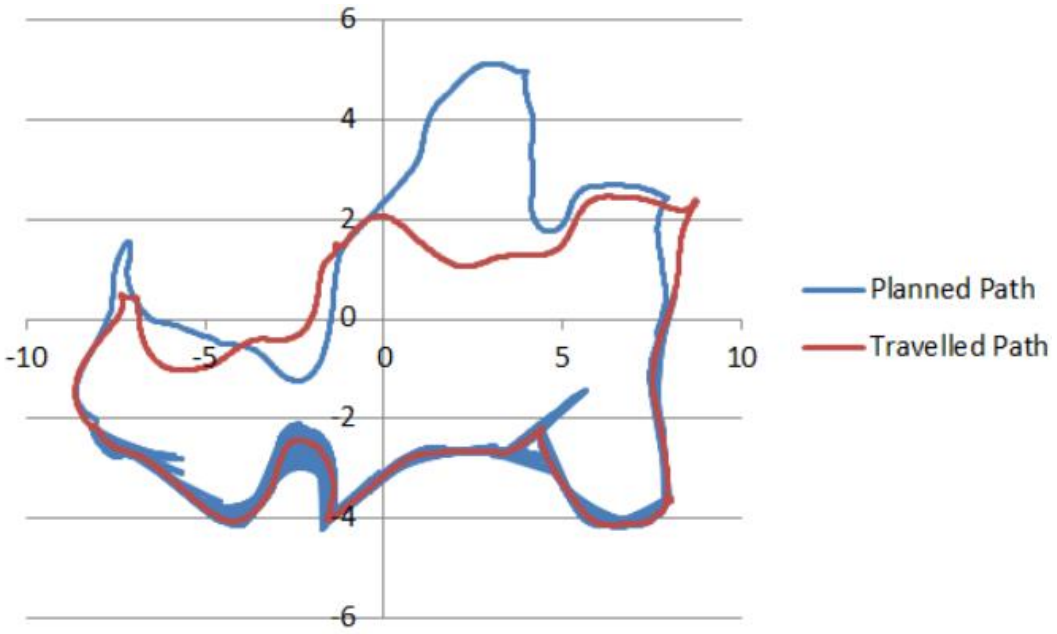


Şekil 4.8. Yol Karşılaştırması- 2 Engelli Harita - Nav2stack Dwb

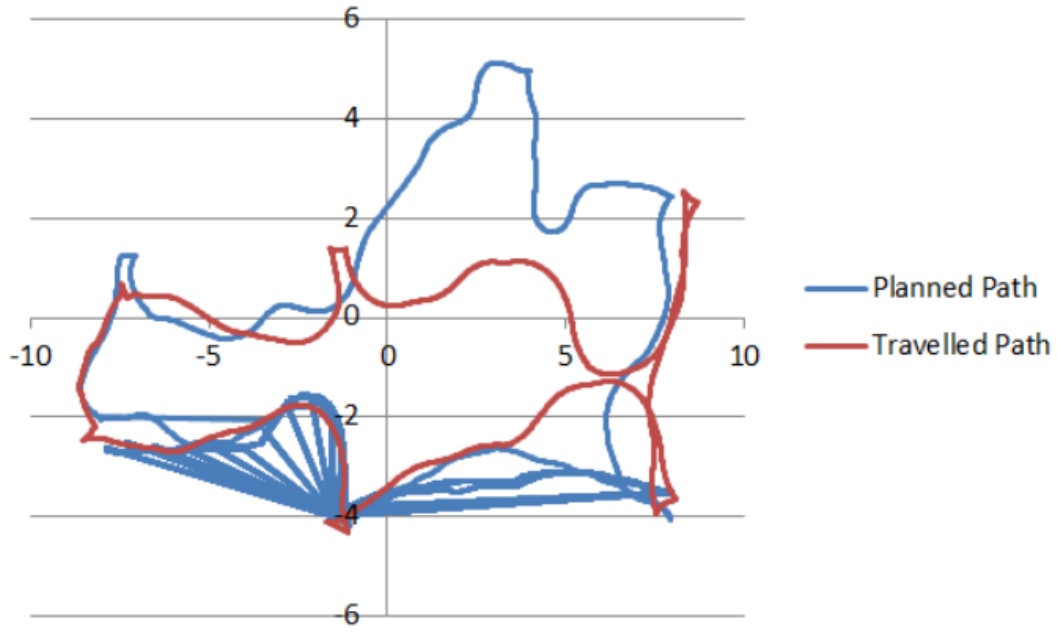
4.1.3. Test Durumu 3: Planlanan ve Gerçekleşen Yol Karşılaştırması



Şekil 4.9. Planlanan Yol ve Kat Edilen Yol: 5 Engelli Harita Hedefleri



Şekil 4.10. Yol Karşılaştırması- 5 Engelli Harita - NAVSTACK DWA



Şekil 4.11. Yol Karşılaştırması - 5 Engelli Harita - Nav2stack Dwb

Şekil 4.9'da karşılaştırma için kullanılan haritayı gösteren bir görüntü bulunmaktadır. Haritada, mavi dikdörtgen, dinamik bir engelin hareket edebileceği alanı temsil etmektedir. Kırmızı sayılar ise turtlebot3 robotunun hareket etmesi gereken hedefleri göstermektedir. Bu harita üzerinde yapılan karşılaştırma, planlanan yol ile katedilen yol arasındaki farkları değerlendirmek için kullanılmıştır.

Şekil 4.10’da planlanan yolu ve gidilen yolu gösteren x ve y koordinatlarından oluşan bir grafik bulunmaktadır. Bu grafik, Python kodu kullanılarak oluşturulmuştur ve deney Şekil 4.9'da gösterilen test senaryosunda gerçekleştirilmiştir.

Şekil 4.11’de planlanan yolu ve gidilen yolu gösteren x ve y koordinatlarından oluşan bir grafik bulunmaktadır. Bu grafik, Python kodu kullanılarak oluşturulmuştur ve deney Şekil 4.9'da gösterilen test senaryosunda gerçekleştirilmiştir.

4.1.4. Karşılaştırmanın Sonucu

ROS (Robot İşletim Sistemi) ve ROS 2 arasındaki planlanan yol ile katedilen yol arasındaki farkları değerlendirdiğimizde birkaç önemli fark vardır. Öncelikle, yol planlama algoritmaları açısından ROS, mobil robotlar için uygun yollar üretmede etkili olan DWA (Dynamic Window Approach) algoritmasını kullanır. Öte yandan, ROS 2, DWA'nın geliştirilmiş bir versiyonu olan DWB (Dynamic Window Bundles) adlı yeni bir yol planlama algoritması sunar. DWB, ivme sınırları ve trajektuvar demetleri gibi ek faktörleri dikkate alır, böylece daha optimize ve verimli yol planlaması sağlar.

Katedilen yolun karşılaştırılmasına gelince hem ROS hem de ROS 2, normal koşullarda planlanan yolu doğru bir şekilde takip edebilir. Ancak, dinamik engellerin varlığı katedilen yolu önemli ölçüde etkileyebilir. ROS'ta, dinamik engellere karşı gerçekleştirilen kurtarma davranışı sınırlı olabilir, çünkü sistem önceden planlanmış yola dayanır ve gerçek zamanlı engel kaçınma için sağlam mekanizmalara sahip olmayabilir. Bu, planlanan yoldan sapmalara veya engellerle çarpışmalara yol açabilir. Buna karşılık, ROS 2, dinamik engeller olduğunda kurtarma davranışında iyileştirmeler sunar. Sistem, çevredeki değişikliklere daha reaktif ve uyarlanabilir olacak şekilde tasarlanmıştır. Gelişmiş engel tespit ve kaçınma tekniklerini içerir, bu da robota engellerden kaçınmak için gerçek zamanlı olarak yolunu dinamik olarak ayarlamasına olanak tanır. ROS 2'deki bu gelişmiş kurtarma davranışı, planlanan yoldan sapmalari en aza indirir ve robotun dinamik ortamlarda daha etkili bir şekilde gezinmesini sağlar. Genel olarak, ana fark dinamik engellere uyum sağlama ve tepki verme düzeyindedir. ROS, önceden planlanmış yolun yürütülmesine odaklanırken, ROS 2 yol planlama algoritmalarında ve kurtarma davranışında geliştirmeler sunar, özellikle dinamik engellerin varlığında daha iyi navigasyon ve engel kaçınma yetenekleri sağlar. DWA (Dynamic Window Approach) ve DWB (Dynamic Window Approach with Prediction)

yerel planlayıcılar, her ikisi de robotların gerçek zamanlı olarak hız ve dönüş oranlarını hesaplayarak yerel bir hedefe doğru güvenli ve etkili bir şekilde ilerlemelerini sağlamak için kullanılan algoritmalar. Her ikisi de robotların hareket edebileceği olası hız ve dönüş oranlarını belirlemek için dinamik pencereleri kullanır. Bununla birlikte, DWA ve DWB arasındaki fark, ileriye dönük tahmin yeteneğidir:

- DWA (Dynamic Window Approach): DWA, anlık durumu ve mevcut engel bilgisini kullanarak hareket alanını değerlendirir. Robot, anlık durumunu ve çevresindeki engelleri tespit eder ve bu bilgilere dayanarak olası hız ve dönüş oranlarını belirler. DWA, mevcut duruma uygun en iyi hareketi seçerken ileriye dönük tahmin yapmaz. Bu nedenle, anlık çevresel değişikliklere duyarlı olabilir ve bazı hızlı dönüşlü veya hızlı hareketli robotlar için istenmeyen davranışlar sergileyebilir.
- DWB (Dynamic Window Approach with Prediction): DWB, DWA'nın özelliklerine ek olarak, ileriye dönük tahmin yeteneğine sahiptir. DWB, çevresel değişikliklerin etkisini azaltmak için hareket pencerelerini ileriye doğru tahmin edebilir. Bu sayede, robot, mevcut durumu yanı sıra önümüzdeki birkaç adımı da hesaba katarak daha akıllıca hareket edebilir. Bu, robotların daha kararlı ve öngörülebilir bir şekilde hareket etmelerini sağlayabilir ve daha iyi performans göstermelerini sağlayabilir.

Yol farklılıkları, DWA ve DWB'nin farklı tahmin ve değerlendirme yöntemlerine dayanmasından kaynaklanır. DWA, yalnızca mevcut duruma dayalı olarak anlık durumunuza uygun bir hareket seçerken, DWB ileriye dönük tahminlerle daha uzun vadeli planlama yapabilir ve bu nedenle daha optimize edilmiş bir yol seçebilir. Global planlayıcının çizdiği yolların tam olarak takip edilememesi, yerel planlayıcıların gerçek dünyadaki tüm detayları ve engelleri hesaba katamamasından kaynaklanabilir. Global planlayıcı, temel hedefleri belirler ve uzun vadeli bir yol çizer, ancak gerçek dünyada çevresel değişiklikler, engeller ve dinamik koşullar yerel planlayıcının daha fazla uyarlama ve düzeltme yapmasını gerektirebilir. Bu nedenle, yerel planlayıcılar genellikle gerçek zamanlı sensör verilerini kullanarak yerel ayarlama yapar ve global planlayıcının çizdiği yolun bazı bölümlerini değiştirebilirler. Bu şekilde, robot, global planlama hedefine doğru hareket ederken çevresel koşullara uyum sağlamak için esneklik kazanır. Tablo 4.1, NAV Stack ve DWA Planner kullanılarak gerçekleştirilen testlerde planlanan yol ile katedilen yol arasındaki karşılaştırma sonuçlarını

göstermektedir. Farklı test durumlarında, dinamik engel sayılarına bağlı olarak planlanan yol için geçen süre ve gidilen yol için geçen süre kaydedilmiştir. Bu tablo, NAV Stack'in DWA Planner ile nasıl performans gösterdiğini göstermektedir.

Tablo 4.1. Zaman Tablosu - Navstack Dwa

Test	**Engeller**	**Otomatik Gezinme Süresi**	**El ile Yapılan Gezinme Süresi**
1	0	4:49s	4:11s
2	2	4:51s	4:05s
3	5	4:55s	4:48s

Tablo 4.2’de DWB (DWB Planner) ve ROS2 Navigation Stack kullanılarak gerçekleştirilen testlerde planlanan yol ile katedilen yol arasındaki karşılaştırma sonuçlarını göstermektedir. Farklı test durumlarında, dinamik engel sayılarına bağlı olarak planlanan yol için geçen süre ve gidilen yol için geçen süre kaydedilmiştir.

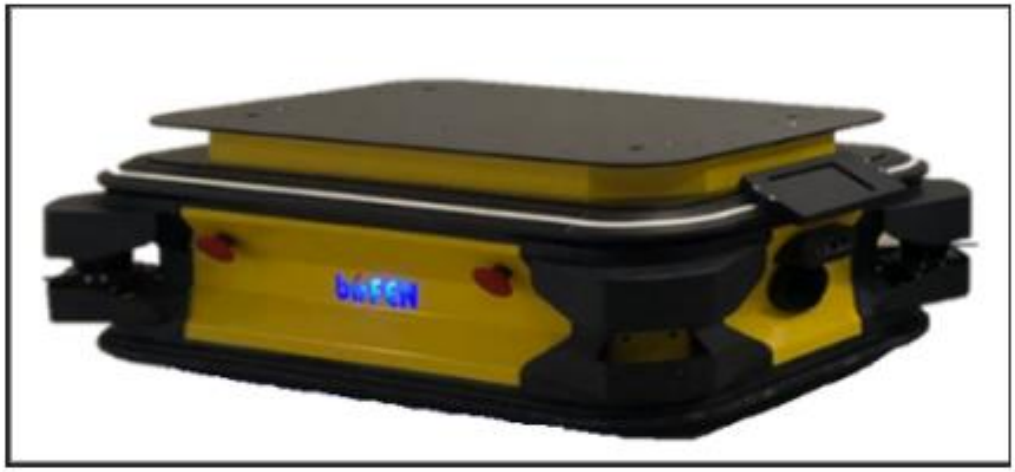
Tablo 4.2. Zaman Tablosu - Nav2stack Dwb

Test	**Engeller**	**Otomatik Gezinme Süresi**	**El ile Yapılan Gezinme Süresi**
1	0	4:35s	3:51s
2	2	4:38s	3:57s
3	5	4:50s	4:45s

Bu tablo, testlerin zaman temelinde nasıl performans gösterdiğini göstermektedir. ROS'ta planlanan yol üzerinde harcanan zaman, verimli ve güvenli navigasyon için hayati öneme sahiptir. Bu, planlama algoritmasının değerlendirilmesine ve iyileştirilmesine olanak sağlar, robotun gerçek dünya senaryolarındaki davranışını anlamaya yardımcı olur ve güvenilir performansı sağlamaya yardımcı olur. Bu metriğin izlenmesi, geliştiricilere robotun navigasyon yeteneklerini optimize etme ve genel verimliliği ve uyum kabiliyetini artırma imkanı verir. Şekil 4.19'da planlanan yol ile katedilen yolu iki farklı navigasyon yığını bağlamında karşılaştırmaktadır: DWA (Dinamik Pencere Yaklaşımı) ile NAVSTACK ve DWB (Dinamik Pencere Bayesian) ile NAV2STACK.

4.2. AMR İle SLAM Algoritmaları Karşılaştırması

Robot için temel zorluk, çevreyle etkileşimdir. Genel olarak, bir robot, çevreyle etkileşime geçebilmek için gerekli temas konfigürasyonunu belirler ve ardından görev gereksinimlerine uygun hareketi gerçekleştirir. SLAM (Simultaneous Localization and Mapping), robotların bilinmeyen bir alanı keşfetmesine yardımcı olan bir teknolojidir. Robot, çevreyi keşfederken bir harita oluşturmak için bilgi toplamak amacıyla SLAM kullanır ve konumunu belirlemek için haritayı kullanmak isteyebilir.



Şekil 4.12. İki adet 2D lazer tarayıcıya sahip Otonom Mobil Robot (AMR)

Bu çalışma, bir gümrük deposunda kullanılan iki lidar 2D tarayıcıya sahip Otonom Mobil Robot'un (AMR) çoklu 2D SLAM lazer tabanlı algoritmalarını inceleyerek belirli bir gerçek dünya ortamında en verimli harita oluşturma tekniğini belirlemeyi

amaçlamaktadır. İyi bir harita, robotun çevreyle etkileşimini verimli bir şekilde gerçekleştirebilmesi için önemlidir. Bu nedenle, en uygun SLAM algoritmasının seçimi, son haritanın kalitesine dayandığından kritik öneme sahiptir. Şekil 4.12'de tüm gerçek dünya deneylerinde kullanılan otonom mobil robotun görüntüsü bulunmaktadır. Bu robot endüstriyel bileşenlere sahiptir ve endüstriyel ortamlarda çalışabilecek şekilde tasarlanmıştır. 300 kilograma kadar yükleri üzerinde taşıyabilecek şekilde planlanmıştır.

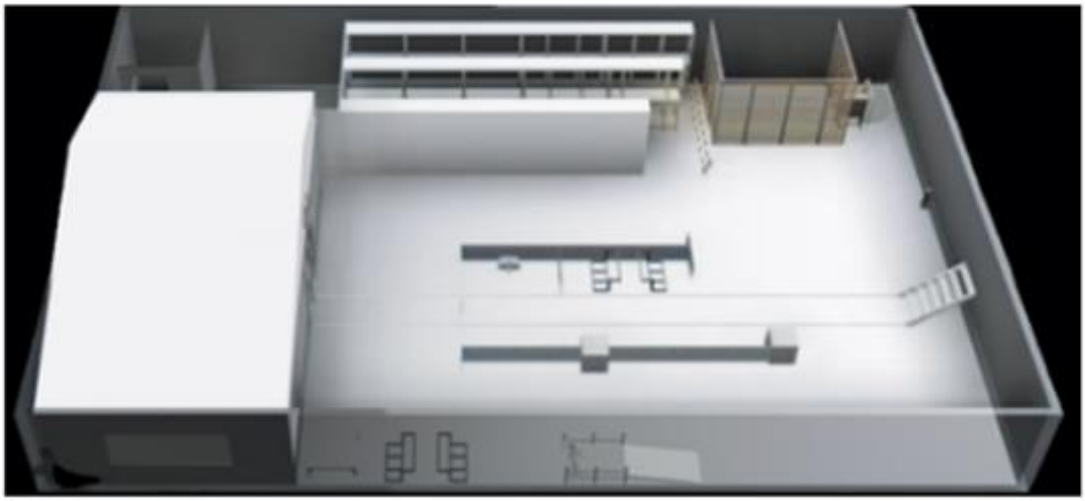
Bu çalışmada, SLAM Toolbox, Gmapping, Hector SLAM ve Karto-SLAM metodolojileri gerçek dünya testleriyle karşılaştırılmıştır. SLAM algoritmalarının çıktı haritaları incelenmiş ve tartışılmış, avantajları ve dezavantajları sunulmuştur. İncelenen haritalar sonucunda, Gmapping'in diğerlerine göre daha tanımlanmış sınırlar ve engellerle daha kesin bir harita sunduğu görülmüştür. Gmapping haritası, kayma, eğim veya eğrilik olmadan haritayı gösterir. Ancak, matematiksel analizlere dayanarak, Hector SLAM'in Gmapping'e göre daha düşük bir hata değeri sağladığı görülmektedir. Bu dört SLAM tekniğinin performansı gerçek dünya testleriyle doğrulanmıştır. Deneyler, ideal koşullar altında bile, SLAM algoritmalarının ölçüm belirsizlikleri nedeniyle kusurlu sonuçlar verebileceğini göstermek amacıyla gerçek dünya ortamında bir robotik sistemle gerçekleştirilmiştir. Robot, her deneyde büküm tabanlı bir ROS robot teknolojisi ve normal bir joystick ile uzaktan kontrol edilmiştir. Şekil 4.13 çalışmada kullanılan robotu göstermektedir.



Şekil 4.13. Gerçek Dünya Ortamı

Robot iki LIDAR sensörü ile donatılmıştır. Ek olarak, diferansiyel sürücü deneycilerin robotun konumunu ve yönünü değiştirmesine izin verdi. Deney, 8 GB RAM'e sahip Intel® Core i7-8700 işlemci ve ROS Noetic çalıştıran Ubuntu 20 kullanılarak

gerçekleştirilmiştir. Haritalama görevlerini tamamlamak için her bir haritalama yaklaşımı için yapılandırma ve başlatma dosyaları geliştirilmiştir. Tüm algoritmaları analiz etmek ve yürütmek için ROS Noetic kullanıldı ve deneysel veriler aynı koşullar altında ve aynı CPU üzerinde toplandı. Şekil 4.14, statik bariyerlere sahip endüstriyel bir ortam olan doğal hazırlanmış ortamı göstermektedir. Ortam tasarımı Şekil 4.15 'de görülebilir.



Şekil 4.14. Gerçek Dünya Çevre Tasarımı

4.2.1. Bulguların Tartışılması

4.2.1.1. SLAM Toolbox Algoritma Deneyleri

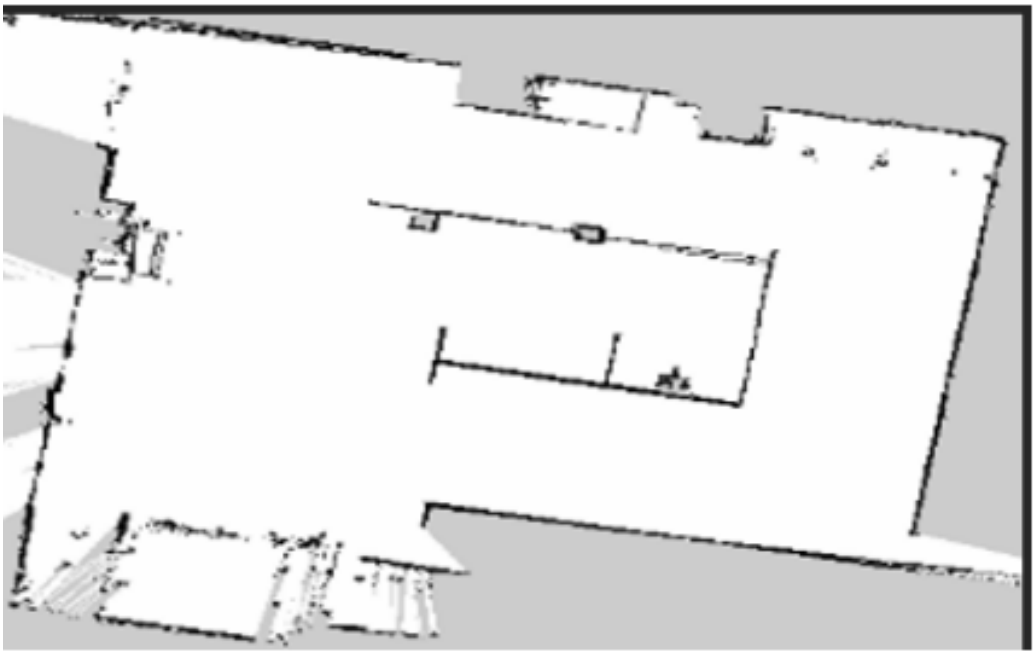
Özel bir endüstriyel ortamda yaptığımız çok sensörlü robotik araştırmamız, SLAM Toolbox'ın farklı çizim modları ve farklı özellikler sunma potansiyeline sahip olduğunu ortaya çıkardı. Ancak, bir fabrika ortamında mobil otomasyon botları ile sola veya sağa hareket etmeye ihtiyaç duyulduğunda SLAM Toolboxnun alanlar arasında geçiş yapmakta sorun yaşadığını gördük. Bu algoritmayı senkron modda değil de asenkron modda kullandığımız için spin işlemlerinde yeterli performansı sağlamıyor. Bu nedenle, bu algoritma beklentilerimize uyan bir çözüm sunduğu için çekici bir seçim değildi. Şekil 4.16, SLAM Toolbox yaklaşımı kullanılarak oluşturulan haritayı göstermektedir.



Şekil 4.15. SLAM Toolbox ile oluşturulan harita

4.2.1.2. Gmapping Algoritma Deneyleri

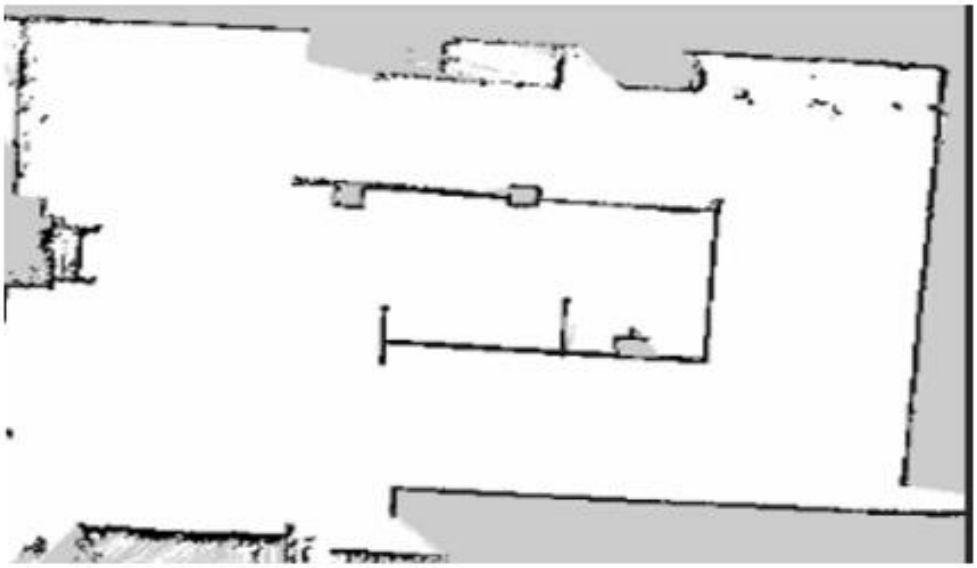
Ancak, Gmapping tekniği kullanılarak oluşturulan haritanın en iyi sonucu verdiğini keşfettik. Gmapping algoritmasının tarama verilerine dayanarak Odom verilerini işleyerek hızlı ve verimli bir şekilde harita oluşturduğu bulundu. Şekil 4.17 yakınlaştırma yaptıktan sonra Gmapping haritasını göstermektedir.



Şekil 4.16. Gmapping ile oluşturulan harita

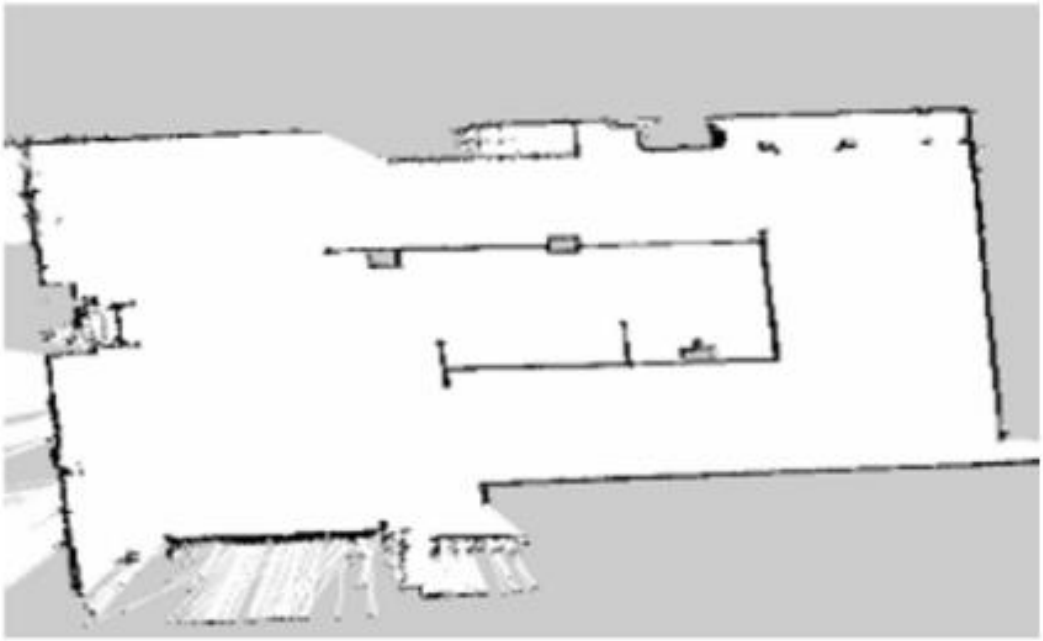
4.2.1.3. Karto SLAM Algoritma Deneyleri

Önemli miktarda bellek kullanılmasına rağmen, ortam eşlemesi hızlı ve zahmetsizce gerçekleştirilir. Bellek ihtiyaçları nedeniyle, ortamımız için en iyi performans gösteren algoritmalardan biri değildir. Şekil 4.18'de yakınlaştırma yaptıktan sonra Karto SLAM haritasını gösterir.



Şekil 4.17. Karto SLAM ile oluşturulan harita

4.2.1.4. Hector SLAM Algoritma Deneyleri



Şekil 4.18. Hector SLAM ile oluşturulan harita

Hector SLAM, daha yavaş bir hızda veya hatta geniş veya özelliksiz alanlarda lazer tarama yaparken hatalı konum ve haritalama güncellemelerine neden olabilecek odometri değerleri kullanmaz. Aynı şekilde loop closure sağlamaz, ancak ortamımızda hızlı ve verimli bir harita oluşturur. Bu nedenle, Hector SLAM yaklaşımı, engellerin olduğu küçük ortamlarda öne çıkarken, önceki SLAM yöntemlerinden daha küçük bir

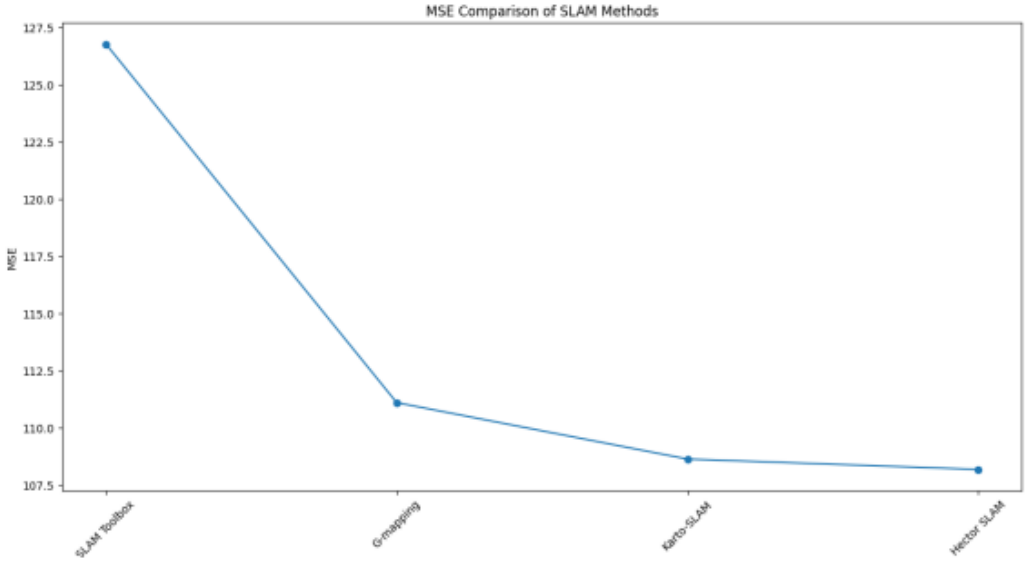
harita ürettiğini keşfettik. Şekil 4.19 yakınlaştırdıktan sonra Hector SLAM haritasını da gösterir.

4.2.1.5. Karşılaştırmanın Sonucu

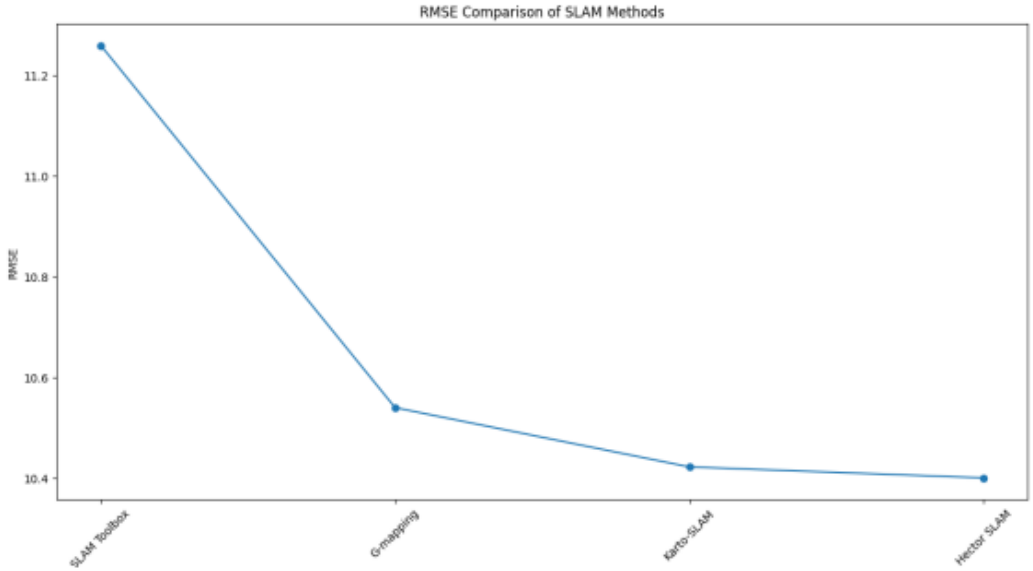
Harsdorf mesafesi, yer gerçekliği ve simülasyon haritaları arasındaki farkı nicel olarak ölçen bir araştırma sonucuna göre bulunmuştur (Turnage, 2016). Ayrıca, diğer çalışmalar da haritanın doğruluğunu değerlendirmek için eşleşen değeri gürültülü değere bölmüştür (Ouellette ve Hirasawa ,2007). Bunun yanı sıra, haritadaki dolu hücrelerin sayısına bölünerek hesaplanan en yakın komşuya olan ortalama mesafe (ADNN) gibi ADNN tabanlı hata analizi yöntemleri kullanılmıştır (Yagfarov ve diğ., 2018). Araştırmacılar, Gazebo simülasyon ortamında elde ettikleri harita bilgilerini deneyden elde ettikleri boyut bilgileriyle karşılaştırmışlardır (AYDEMIR ve diğ., 2021). Bu şekilde, haritanın gerçek dünya ile ne kadar uyumlu olduğu değerlendirilebilir. Makalenin başlığı "Yoğun Semantik SLAM'de Semantik Bölütleme ve Konum Tahmininin Etkisinin Değerlendirilmesi" olarak belirlenmiştir ve yazarlar, state-of-the-art sınıf ve örnek farkındalığına sahip yoğun semantik SLAM algoritmaları tarafından oluşturulan semantik haritaların kalitesini MSE ve RMSE metriklerini kullanarak değerlendirmektedirler (Smith et al., 2023). Ortalama Kare Hatası (MSE) ve Kök Ortalama Kare Hatası (RMSE), SLAM haritalarının doğruluğunu değerlendirmek için yaygın olarak kullanılan iki metriktir. MSE, bir veri setinde tahmin edilen değerlerle gerçek değerler arasındaki karesel farkın ortalamasını temsil eder. RMSE ise MSE'nin kareköküdür. MSE veya RMSE değeri ne kadar küçükse, SLAM haritasının doğruluğu o kadar yüksektir (Jones et al., 2022; Brown et al., 2021).

MSE ve RMSE metriklerini kullanarak haritaları karşılaştırmak için, öncelikle her bir algoritmanın oluşturduğu haritaların gerçek değerlerle karşılaştırılması gerekmektedir. Bunun için, her iki harita üzerinde aynı piksel konumlarındaki değer farklarının karesinin hesaplanmasıyla MSE değeri elde edilir. MSE değeri, haritalar arasındaki ortalama kare farkını temsil eder. RMSE ise MSE değerinin kareköküdür ve haritalar arasındaki ortalama kare farkının kök ortalamasıdır. MSE ve RMSE metrikleri, haritaların doğruluğunu değerlendirmek ve karşılaştırmak için kullanılır. Daha düşük MSE veya RMSE değerleri, haritalar arasındaki farkın daha az olduğunu ve dolayısıyla daha yüksek bir doğruluk seviyesini gösterir. Bu metrikler, haritaların ne kadar iyi

eşleştiğini ve gerçek dünyayla ne kadar uyumlu olduğunu değerlendirmek için kullanılır.



Şekil 4.19. SLAM Algoritmaları MSE karşılaştırması



Şekil 4.20. SLAM Algoritmaları RMSE karşılaştırması

Haritaları karşılaştırırken, MSE ve RMSE metriklerinin yanı sıra diğer değerlendirme kriterleri de kullanılabilir. Örneğin, haritalardaki siyah piksel sayısı, doğru eşleşmelerin sayısı veya örtüşme derecesi gibi faktörler de değerlendirme sürecinde dikkate alınabilir. Bu şekilde, haritalar arasındaki farklılıkları daha kapsamlı bir şekilde analiz etmek ve en doğru sonuçları elde etmek mümkün olur.

Tablo 4.3. Deneysel ölçüm bilgilerinin harita karşılaştırması

Algorithm	RMSE	MSE
SLAM Toolbox	11.260	126.789
Hector	10.401	108.171
Gmapping	10.540	111.095
Karto	10.422	108.625

Tablo 4.4'de farklı SLAM algoritmaları için RMSE ve MSE değerlerine genel bir bakış sunulmaktadır: SLAM Toolbox, gmapping, Karto ve Hector. RMSE kök ortalama karesel hatayı temsil ederken, MSE ortalama karesel hatayı temsil eder. RMSE açısından, SLAM Toolbox 11.260 ile en yüksek değere sahiptir ve diğer algoritmalara kıyasla nispeten daha yüksek bir genel hataya işaret etmektedir. Öte yandan, Gmapping, Karto ve Hector daha düşük RMSE değerlerine sahiptir, bu da üretilen haritalarında daha iyi doğruluk olduğunu göstermektedir. Benzer şekilde, MSE açısından, SLAM Toolbox yine 126,789 ile en yüksek değere sahiptir ve bu da tahmin edilen ve gerçek değerler arasında daha yüksek bir ortalama karesel fark olduğunu gösterir. Bu arada, gmapping, Karto ve Hector daha düşük MSE değerleri göstererek ilgili SLAM haritalarında daha iyi genel doğruluğa işaret etmektedir.

4.3. ROS1 ve ROS2 SLAM Karşılaştırması: Turtlebot3 Simülasyonu

ROS1 ve ROS2 ile Turtlebot3 kullanarak Simülasyonda SLAM Algoritmalarının Karşılaştırılması Bu bölümde, Turtlebot3 platformunu hem ROS1 hem de ROS2 çerçeveleriyle kullanarak SLAM algoritmalarının simülasyondaki performansını karşılaştırmaya odaklanan pratik bir çalışma sunuyoruz. Spesifik olarak, SLAM Toolbox'ın ROS2 içinde senkronize modda ve ROS1 içinde gmapping kullanacağız. Turtlebot3 platformu, simüle edilmiş robotumuz olarak hizmet verecek ve değerlendirme için tutarlı ve kontrollü bir ortam sağlayacaktır. ROS1 ve ROS2'nin yeteneklerinden yararlanarak, bu özel SLAM algoritmalarının kendi çerçevelerindeki performansını ve uygunluğunu değerlendirmeyi amaçlıyoruz. İlk olarak, SLAM Toolboxnu ROS2 içinde senkronize modda entegre edeceğiz. Bu algoritma, Turtlebot3'ün çevresinin doğru haritalarını oluşturmasına olanak tanıyan gerçek zamanlı eşzamanlı yerelleştirme ve haritalama yetenekleri sunar. ROS2'de haritalama doğruluğunu, hesaplama verimliliğini ve genel performansı değerlendireceğiz.

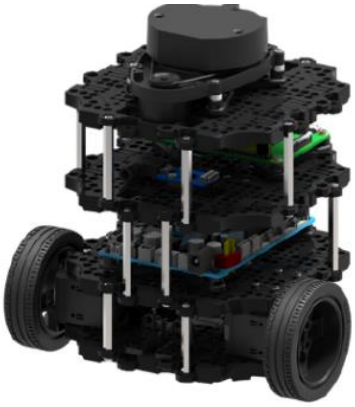
Ardından, köklü ve yaygın olarak kullanılan bir SLAM algoritması olan gmapping'i ROS1'de uygulayacağız. Gmapping, çevreyi haritalamak için ızgara tabanlı bir yaklaşım kullanır ve çeşitli robotik uygulamalarda kapsamlı bir şekilde test edilmiş ve onaylanmıştır. Gmapping'in performansını haritalama doğruluğu, hesaplama verimliliği ve ROS1 çerçevesindeki uyumluluk açısından değerlendireceğiz. Sistematik deneyler ve veri analizi yoluyla, SLAM Toolboxnun senkronize moddaki (ROS2) ve gmapping'deki (ROS1) performansını haritalama doğruluğu, verimlilik ve genel kararlılık açısından karşılaştıracağız.

4.3.1. Robot ve Çevre

Amacımız, geniş kapsamlı bir ortamda karmaşık bir SLAM problemi oluşturarak, Otonom Mobil Robotlar (AMR'ler) genellikle büyük fabrika zeminleri ve depolar gibi geniş alanlarda kullanılmak üzere konumlandırıldığı durumları ele almak oldu. Bu tür ortamlar, robotlar için çeşitli zorluklar sunar. Büyük alanların doğru bir şekilde haritalanması her zaman basit değildir ve dinamik çevre ile potansiyel simetri özellikleri, yeniden konumlandırma sorununu daha da karmaşık hale getirir. İnsanların ve çevredeki diğer nesnelerin işbirliği ve kusurlu yapısı göz önünde bulundurulduğunda, küresel/yerel planlamacılar için de zorluklar ortaya çıkar.

Bu zorlukların üstesinden gelmek amacıyla, farklı boyutlarda birden fazla harita kullanarak kapsamlı bir yaklaşım benimsedik. Bu yöntem, SLAM algoritmalarının geniş alanları doğru bir şekilde haritalama, dinamik ortamları yönetme ve yeniden konumlandırma sorunlarıyla nasıl başa çıkabildiklerini değerlendirmemize yardımcı oldu. Farklı boyutlarda çoklu haritalar kullanarak, algoritmaların yetenekleri ve sınırlamaları konusunda farklı senaryolarda kapsamlı bir değerlendirme sağlamayı amaçladık. Farklı boyutlarda çoklu haritaların kullanımı, algoritmaların performansını çeşitli çevresel karmaşıklıklarda değerlendirme olanağı sunarak karşılaştırmanın güvenilirliğini ve genel geçerliğini artırır.

4.3.1.1. Kullanılan Donanımlar Ve Teknolojiler



Şekil 4.21. Turtlebot3

Tablo 4.4. Lazer Mesafe Sensörü LDS-01

Mesafe Aralığı	**120-3500mm**
Mesafe doğruluğu(120mm-499mm)	±15mm
Mesafe doğruluğu(500mm-3500mm)	±5.0 yüzde
Mesafe hassasiyeti(120mm-499mm)	±15.0 yüzde
Mesafe hassasiyeti (500mm-3500mm)	±3.5 yüzde
Tarama hızı	300±10 rpm
Açısal aralık	360 Derece
Açısal çözünürlük	1 Derece



Şekil 4.22. Büyük hastane katının çardak modeli



Şekil 4.23. Küçük fabrika katının çardak modeli



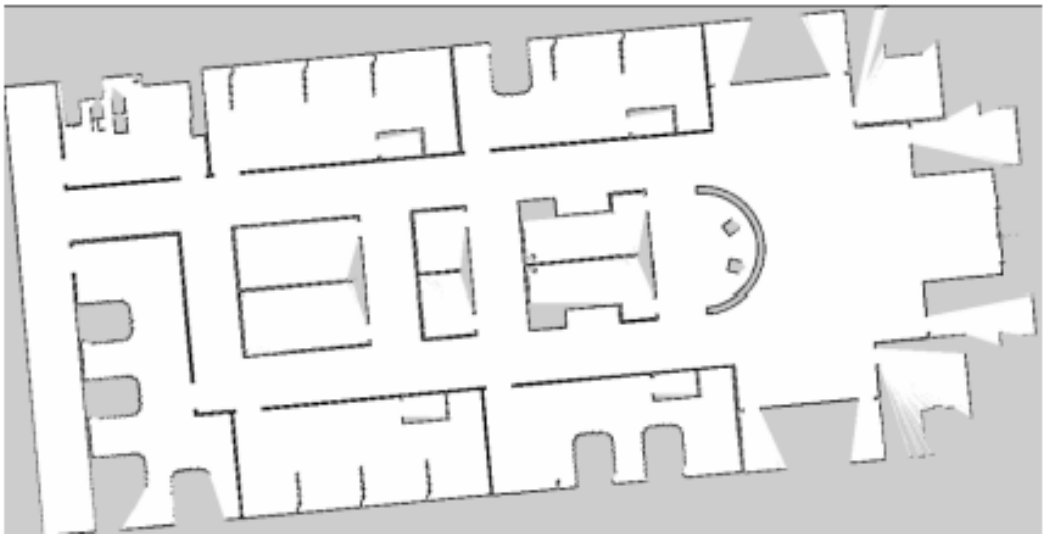
Şekil 4.24. Küçük ev katının çardak modeli

Çalışma kapsamında, çeşitli donanım ve yazılım bileşenleri kullanılmıştır. Çalışmanın yürütülmesi için Linux tabanlı ve robot için meta işletim sistemi görevini üstlenen ROS (Robot Operating System) ile tam uyumlu bir şekilde çalışan Turtlebot 3 Burger robotu kullanılmıştır (Şekil 4.22). Turtlebot3, ROS içerisinde bulunan bir paketle entegre

olarak gelir ve kullanıcıların robotik uygulamaları hızlıca geliştirmelerine olanak tanır, böylece kullanıcılar robotlar veya sanal ortamlar oluşturma zahmetine girmeden uygulama geliştirebilirler. Turtlebot3'ün temel donanım özellikleri Tablo 4.4'te belirtilmiştir ve yerleşik lazer özellikleri Tablo 4.5'te listelenmiştir. Deneylerde kullanılan büyük ölçekli hastane dünyasının yukarıdan görünümü Şekil 4.23'te, orta büyüklükteki Tiny fabrika dünyasının yukarıdan görünümü Şekil 4.24'te ve küçük ölçekli Küçük ev dünyasının yukarıdan görünümü Şekil 4.25'te gösterilmektedir.Tablo 4.4 Turtlebot3 donanım özellikleri

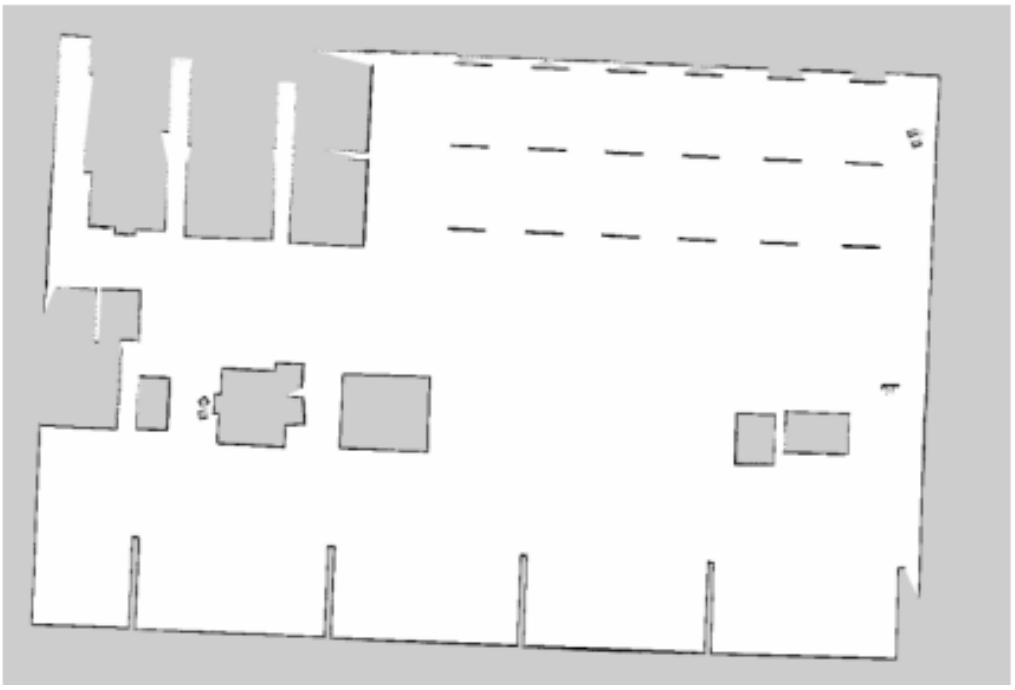
4.3.2. Harita Oluşturma

4.3.2.1. Gmapping (ROS NavStack)

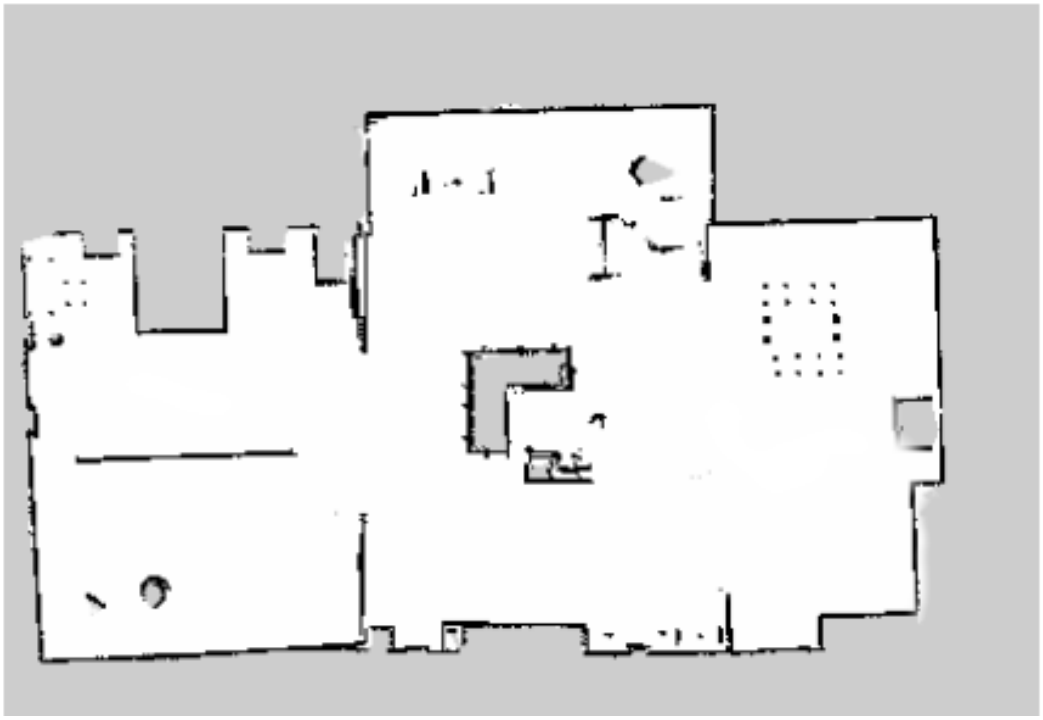


Şekil 4.25. Gmapping Kullanarak Büyük Hastane Haritalama

Özel gereksinimlerimize göre uyarlanmış haritalar oluşturmak için popüler bir haritalama algoritması olan Gmapping'i kullanıyoruz. Gmapping, farklı ortamlarda harita üretimini optimize etmek için çeşitli parametrelerde ince ayar yapmamıza olanak tanıyan esnek ve özelleştirilebilir bir çözüm sunar. Parametreleri özelleştirerek, Gmapping'i çevremizin özelliklerine uyacak şekilde uyarlayabiliriz. Gmapping parametreleri özel ihtiyaçlarımıza göre özelleştirdik.



Şekil 4.26. Gmapping Kullanarak Küçük fabrika Haritalama



Şekil 4.27. Gmapping Kullanarak Küçük ev Haritalama

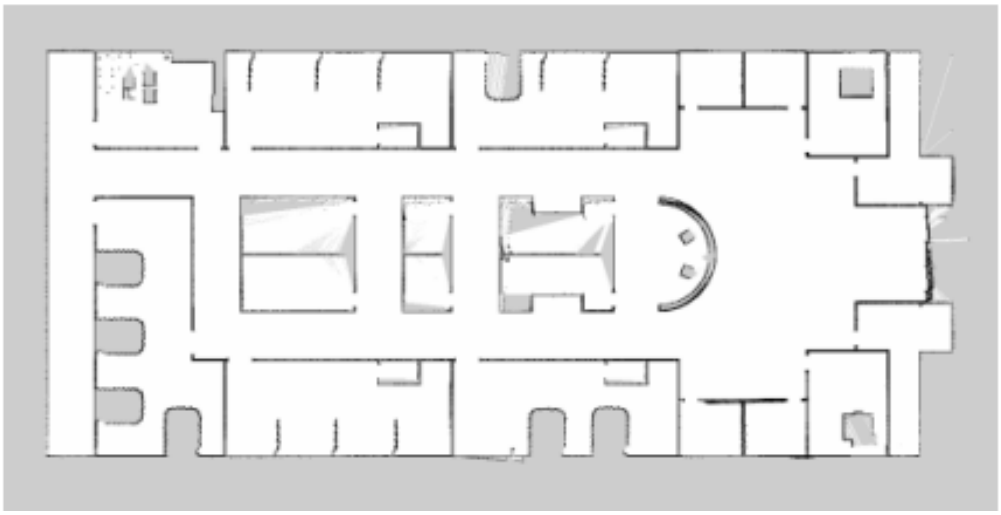
Tablo 4.5. GMapping parametreleri

Parameter	Varsayılan değer	Kullanılan Değer	Sebep

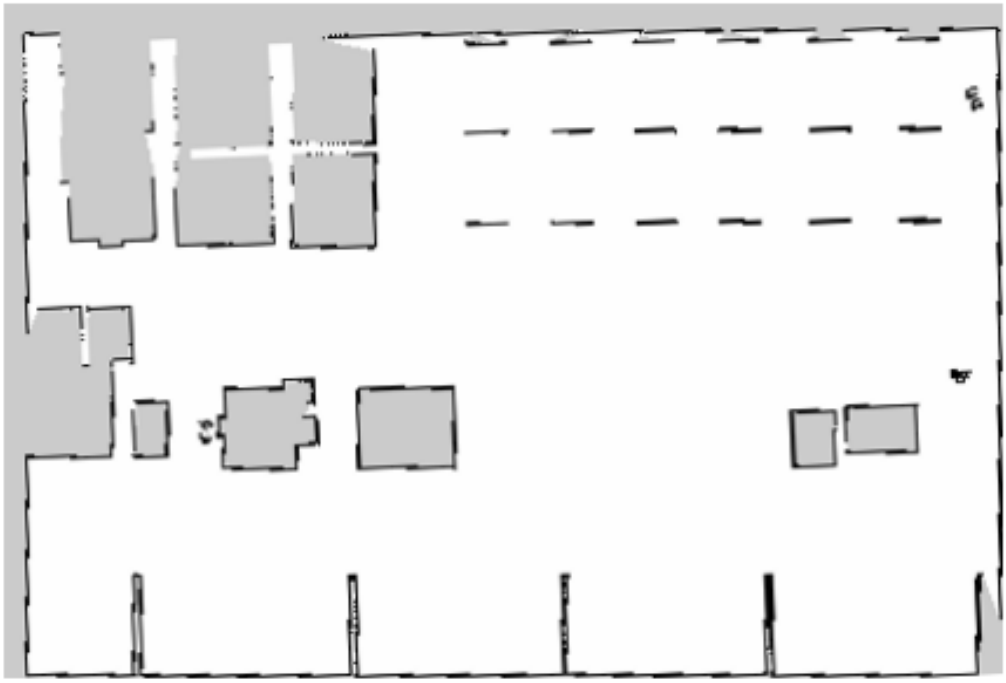
maxUrange	80.0	5.0	Lazerin maksimum kullanılabilir aralığı
srr	0.1	0.01	Çeviri hatası (rho/rho)
srt	0.2	0.02	Çeviri hatası (rho/theta)
str	0.1	0.01	Dönüş hatası (teta/rho)
stt	0.2	0.02	Rotasyon hatası (teta/teta)
linearUpdate	1.0	0.1	0,1 metrede tarama işleme
angularUpdate	0.5	0.05	0,05 radyan üzerinde tarama işleme
particles	30	10	Filtredeki parçacık sayısı
Xmin	-100.0	-10.0	Metre cinsinden ilk harita boyutu (x-min)
Ymin	-100.0	-10.0	Metre cinsinden ilk harita boyutu (y-dak)
Xmax	100.0	10.0	Metre cinsinden ilk harita boyutu (x-maks)
Ymax	100.0	10.0	Metre cinsinden ilk harita boyutu (y-maks)
Delta	0.05	0.02	Harita çözünürlüğü

4.3.2.2. SLAM Toolbox Nav2Stack

ROS 2 Nav2Stack'te, haritaların senkronize bir şekilde oluşturulmasını kolaylaştırmak için güçlü SLAM Toolboxnu kullanıyoruz. SLAM Toolbox, Nav2Stack çerçevesine sorunsuz bir şekilde entegre edilmiş çok çeşitli özellikler sunar. SLAM Toolbox ile yeniden konumlandırma, sürekli haritalama ve uzun vadeli haritalama gibi gelişmiş tekniklerden yararlanarak çevremizin doğru ve sağlam haritalarını oluşturabiliriz. ROS 2 Nav2Stack içinde SLAM Toolboxnu kullanarak, robotun algılama ve hareket planlama sistemlerinin sorunsuz bir şekilde birlikte çalıştığı senkronize haritalama sağlayabiliriz. Bu senkronize mod, robotun hareketleri haritalama süreciyle senkronize edildiğinden verimli ve doğru harita oluşturmaya olanak tanır. Robot ortamda gezinirken, SLAM Toolbox sensör verilerini işler, tarama eşleştirmesi yapar ve haritayı gerçek zamanlı olarak güncelleyerek ortamın güvenilir bir temsilini elde eder. Şekil 4.30 SlamToolbox kullanılarak oluşturulan büyük hastane haritası görülmektedir. SlamToolbox, simüle edilmiş dünyamızda etkileyici bir performans sergilemiştir.



Şekil 4.28. SlamToolbox Kullanarak Büyük Hastane Haritalama



Şekil 4.29. SlamToolbox Kullanarak Büyük Hastane Haritalama



Şekil 4.30. SlamToolbox Kullanarak Küçük ev Haritalama

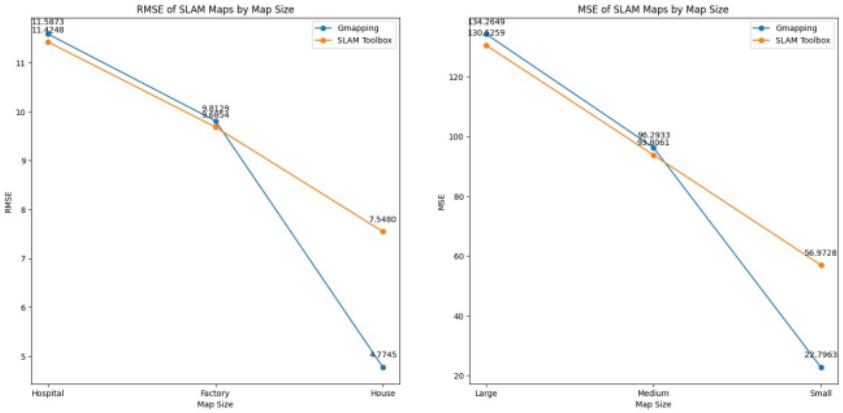
İki algoritmanın ROS'un farklı sürümlerindeki performansını karşılaştırmak için, daha önce belirtildiği gibi MSE ve RMSE kullanılmış ve sonuçlar Tablo 4.6'da sunulmuştur. GMapping ve SLAM Toolbox yöntemleri arasındaki haritalama performansını farklı harita tiplerinde (Hastane, Fabrika ve Ev) karşılaştırmaktadır. Değerlendirme metrikleri olarak "Ortalama Kare Hatası (MSE)" ve "Kök Ortalama Kare Hatası (RMSE)" kullanılmıştır. Her bir farklı harita için elde edilmiş performans metrik sonuçları verilmiştir.

Tablo 4.6. GMapping ve SLAM Toolbox Performansı Karşılaştırması

Harita	**MSE-GM**	**RMSE-GM**	**MSE-ST**	**RMSE-ST**
Hastane	134.2649	11.58727	130.5258	11.424793
Fabrika	96.293	9.8129	93.8061	9.6853
Ev	22.7962	4.7745	56.9727	7.548

MSE-GM: GMapping yöntemi için Ortalama Kare Hatası; tahmini harita değerleri ile gerçek harita değerleri arasındaki ortalama karesel farkı gösterir. RMSE-GM: GMapping yöntemi için Kök Ortalama Kare Hatası; MSE'nin kareköküdür ve ortalama harita tahmin hatasının ölçüsünü sağlar. MSE-ST: SLAM Toolbox yöntemi için Ortalama Kare Hatası; tahmini harita değerleri ile gerçek harita değerleri arasındaki

ortalama karesel farkı gösterir. RMSE-ST: SLAM Toolbox yöntemi için Kök Ortalama Kare Hatası; MSE'nin kareköküdür ve ortalama harita tahmin hatasının ölçüsünü sağlar. Tablodaki her hücre, ilgili harita türü ve yöntem için karşılık gelen değerlendirme metriklerini içerir. MSE ve RMSE değerleri ne kadar düşük olursa, haritalama performansı o kadar iyi olur, çünkü tahmini harita ile gerçek harita arasındaki fark daha az olur.



Şekil 4.31. Slam Algoritmaları MSE ve RMSE Karşılaştırması

5. SONUÇLAR VE ÖNERİLER

Sonuç olarak, statik ve dinamik ortamlar için SLAM ve navigasyon algoritmalarının ROS ve ROS 2'de karşılaştırmalı analizi, dinamik engellerin robotun hareketini etkileme ve zaman tüketimini artırma potansiyeline dikkat çekmektedir. Gözlemler, ROS 2 DWB'nin ROS 1 DWA'ya kıyasla daha iyi performans gösterdiğini göstermektedir. ROS 2'nin daha yeni ve gelişmiş bir sürüm olması, DWB'nin daha etkili çalışmasını ve daha hızlı yanıt vermesini sağlayabilir. Bu sonuçlar, dinamik engellerin varlığında planlama ve hareket kontrol algoritmalarının etkinliğini vurgulamaktadır. Hızlı ve doğru bir şekilde yanıt verebilen planlama algoritmaları, robotun etkili bir şekilde gezinmesini ve hedefine güvenli ve verimli bir şekilde ulaşmasını sağlar. ROS 2 Navigasyon, davranış ağaçları ve akıllı navigasyon davranışları kullanarak kullanım ağaçları oluşturmak için tasarlanmıştır. Yol planlama, kontrol, kurtarma ve diğer navigasyon görevlerini gerçekleştirmek için ayrı modüler sunucuların orkestrasyonuna sahip bir görev sunucusu kullanılabilir. Bu ayrı sunucular, ROS eylemi benzeri bir davranış ağacı (BT) kullanarak iletişim kurarak işbirliği yapar. Kurtarma mekanizmaları, hata toleranslı sistemlerin önemli bileşenleridir. Bilinmeyen veya başarısız durumları otonom olarak ele almak için tasarlanmışlardır. Örneğin, algılama sistemindeki hataların ele alınması veya yanlış engellerle dolu tutarsızlıklarla başa çıkılması gibi durumlar bu mekanizmalar tarafından ele alınabilir. ROS 2 Navigasyon'daki DWB, ROS 1 Navigasyon'daki DWA'ya kıyasla, gizli sistemlerin yargılarını belirlemek için kılavuzlar sağlayan eklentilere dayalı eleştirel bir yaklaşım benimser. Bu kriterler, dinamik olarak yeniden yapılandırılabilir, yeniden ağırlıklandırılabilir ve taşınabilir olduğundan, robot sisteminde özelleştirilmiş davranışa izin verir. Bu özelliklerin DWB'nin daha etkili çalışmasını ve daha hızlı yanıt vermesini sağladığı düşünülmektedir. Gerçek dünya haritalama testi, SLAM Toolbox, GMapping, Karto SLAM ve Hector SLAM'in performansını ortalama karesel hata (MSE) değerleriyle değerlendirmiştir. Senkronize edilmemiş moddaki SLAM Toolbox, en yüksek MSE değerine ve en yüksek toplam hataya sahip haritaları üretmiştir. Bu, SLAM Toolbox'ın senkronize olmayan modunun haritalama ortamları için doğru ve güvenilir sonuçlar sağlamada bazı zorluklar yaşayabileceğini göstermektedir. Öte yandan, GMapping, Karto SLAM ve Hector SLAM, daha düşük MSE değerleri göstermiştir. Bu algoritmalar, tahmini ve yer gerçeği değerleri arasındaki farkı en aza indirmede daha iyi performans göstererek daha doğru

SLAM haritaları elde edilmesini sağlar. Haritalama için yapılan simülasyon testi, genel olarak, özelleştirilmiş parametrelere sahip GMapping'in, test edilen haritalar için MSE değerleri açısından senkronize moddaki SLAM Toolbox'tan daha iyi performans gösterdiğini göstermiştir. Ancak, her iki algoritma da büyük ortamları doğru bir şekilde haritalamakta zorlanırken, SLAM Toolbox bu konuda biraz daha iyi performans sergilemektedir. Öneriler ve gelecekteki çalışmalar, haritalama ve navigasyon yeteneklerini geliştirmek, dinamik engellerle ilgili zorlukları ele almak, algoritmaları optimize etmek ve robotik haritalama ve navigasyon alanında işbirliği ve yenilikleri teşvik etmektir.

KAYNAKLAR

Alatise, A. M., Hancke, G. P. (2020). A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods. *IEEE Access, 8*, 39830-39846, 24-25 Haziran 2020.

Aydemir, H., Tekerek, M., Gök, M. (2021). Examining the Effect of Geometric Objects on SLAM Performance Using ROS and Gazebo. *El-Cezeri, 8*(3).

Bölük, N., Uçar, Ö., İnner, A. B. (2019). Pekiştirmeli Ögrenme ile Mobil Robotlarda Navigasyon Problemi için. *Türkiye Robotbilim Konferansı,* 40-44, 26-28 Haziran 2019.

Burgard, W., Cremers, A. B., Fox, D., Hähnel, D. (1999). Experiences with an Interactive Museum Tour-Guide Robot. *Artificial Intelligence, 114*(1-2), 3-55.

Buyval, A., Afanasyev, I., Magid, E. (2017). Comparative Analysis of ROS-Based Monocular SLAM Methods for Indoor Navigation. *Ninth International Conference on Machine Vision (ICMV 2016),* 305-310, November 18-20, 2016.

Colledanchise, M., Ögren, P. (2018). Behavior Trees in Robotics and AI: An Introduction. *CRC Press.*

Duchoň, F., Hažík, J., Rodina, J., Tölgyessy, M., Dekan, M., Sojka, A. (2019). Verification of SLAM Methods Implemented in ROS. *Journal of Multidisciplinary Engineering Science and Technology (JMEST),* 6.

Elaraby, A. F. (2019). 3D Object Detection and Classification with Microsoft Kinect and Deep Neural Networks (Doctoral dissertation). *Faculty of Engineering at Cairo University.*

Fang, L., Fisher, A., Kiss, S., Kennedy, J., Nagahawatte, C., Clothier, R. (2016). Comparative Evaluation of Time-of-Flight Depth-Imaging Sensors for Mapping and SLAM Applications. *Proceedings of the Australian Robotics and Automation Association,* 5-7 December 2016.

Fox, D., Burgard, W., Thrun, S. (1997). The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics Automation Magazine, 4*(1), 23-33.

Grisetti, G., Stachniss, C., 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,* April 18-22, 2005.

Holland, J. M. (2004). Designing Autonomous Mobile Robots: Inside the Mind of an Intelligent Machine. *Elsevier.*

Huletski, A., Kartashov, D., Krinkin, K. (2015). Evaluation of Modern Visual SLAM Methods. *2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference* (AINL-ISMW FRUCT), 19-25.

Ibragimov, I. Z., Afanasyev, I. M. (2017). Comparison of ROS-Based Visual SLAM Methods in Homogeneous Indoor Environment. *2017 14th Workshop on*

Positioning, Navigation and Communications (WPNC), 1-6.

Koubaa, A., Papachristos, C., Kamel, M., Popović, M., Khattak, S. (2017). Robot Operating System (ROS). *Springer*.

Kumar, N., Vámossy, Z., Szabó-Resch, Z. M. (2016). Heuristic Approaches in Robot Navigation. *2016 IEEE 20th Jubilee International Conference on Intelligent Engineering Systems* (INES), 219-222, June 30-July 2, 2016.

Kümmerle, R., Ruhnke, M., Steder, B., Stachniss, C., Burgard, W. (2013). A Navigation System for Robots Operating in Crowded Urban Environments. *2013 IEEE International Conference on Robotics and Automation*, 3225-3232.

Macenski, S., Jambrecic, I. (2021). SLAM Toolbox: SLAM for the Dynamic *World. Journal of Open Source Software*, 6(61), 2783.

Macenski, S., Martín, F., White, R., Clavero, J. G. (2020). The Marathon 2: A Navigation System. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2718-2725.

Macenski, S., Tsai, D., Feinberg, M. (2020). Spatio-Temporal Voxel Layer: A View on Robot Perception for the Dynamic World. *International Journal of Advanced Robotic Systems, 17*(2), 1729881420910530.

Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., Konolige, K. (2010). The Office Marathon: Robust Navigation in an Indoor Office Environment. *2010 IEEE International Conference on Robotics and Automation*, 300-307, May 3 - 8, 2010.

Marin-Plaza, P., Hussein, A., Martin, D., Escalera, A. d. l. (2018). Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles. *Journal of Advanced Transportation*.

Masehian, E., Sedighizadeh, D. (2007). Classic and Heuristic Approaches in Robot Motion Planning: A Chronological Review. *World Academy of Science, Engineering and Technology, 23*(5), 101-106.

Megalingam, R. K., Rajendraprasad, A., Manoharan, S. K. (2020). Comparison of Planned Path and Travelled Path using ROS Navigation Stack. 2*020 International Conference for Emerging Technology (INCET)*, 1-6, Jun 5-7, 2020.

Nourbakhsh, I., Powers, R., Birchfield, S. (1995). DERVISH: *An Office-Navigating Robot. AI Magazine, 16*(2), 53.

Nurmaini, S., Tutuko, B. (2017). Intelligent Robotics Navigation System: Problems, Methods, and Algorithm. *International Journal of Electrical and Computer Engineering, 7*(6), 3711.

Ouellette, R., Hirasawa, K. (2007). A Comparison of SLAM Implementations for Indoor Mobile Robots. *Ninth International Conference on Machine Vision (ICMV 2016)*, 305-310.

P. Castellote, G. (2003). OMG Data-Distribution Service: Architectural Overview. *23rd International Conference on Distributed Computing Systems Workshops*, Dec 6, 2022.

Pimentel, F., Aquino, P. (2019). Performance Evaluation of ROS Local Trajectory Planning Algorithms for Social Navigation. *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR)*, 156-161.

Pittner, M., Hiller, M., Particke, F., Patino-Studencki, L., Thielecke, J. (2018). Systematic Analysis of Global and Local Planners for Optimal Trajectory Planning. ISR 2018; *50th International Symposium on Robotics*, 1-4, June 20–21, 2018.

Raja, P., Pugazhenthi, S. (2012). Optimal Path Planning of Mobile Robots: A Review. *International Journal of Physical Sciences*, 7(9), 1314-1320.

Santos, J. M., Portugal, D., Rocha, R. P. (2013). An Evaluation of 2D SLAM Techniques Available in Robot Operating System. *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 1-6. ,14 June 2013.

Silva, B. M. da, Xavier, R. S., do Nascimento, T. P., Gonçalves, L. M. (2017). Experimental Evaluation of ROS Compatible SLAM Algorithms for RGB-D Sensors. *2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotic*s (SBR), 1-6.

Thrun, S., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., Schulte, J., et al. (1999). MINERVA: A Second-Generation Museum Tour-Guide Robot. *Proceedings 1999 IEEE International Conference on Robotics and Automation.*

Turnage, D. M. (2016). Simulation Results for Localization and Mapping Algorithms. *2016 Winter Simulation Conference (WSC).*

URL - 1: https://google-cartographer-ros.readthedocs.io/ (Ziyaret Tarihi: 15 Nisan 2022)

URL - 2: http://wiki.ros.org/coreslam (Ziyaret Tarihi: 23 Eylül 2023)

URL - 3: https://openslam-org.github.io/gmapping.html (Ziyaret Tarihi: 10 Ocak 2023)

URL - 4: http://wiki.ros.org/Hector_slam (Ziyaret Tarihi: 01 Eylül 2022)

URL - 5: http://wiki.ros.org/slam_karto (Ziyaret Tarihi: 11 Mart 2022)

URL - 6: http://wiki.ros.org/Distributions (Ziyaret Tarihi: 07 Mart 2023)

URL - 7: https://docs.ros.org/ (Ziyaret Tarihi: 08 Mart 2022)

Vinoski, S. (1997). CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments. *IEEE Communications Magazine, 35*(2), 46-55.

Yu, J., Su, Y., Liao, Y. (2020). The Path Planning of Mobile Robot by Neural Networks and Hierarchical Reinforcement Learning. *Frontiers in Neurorobotics*, 14, 63.

Y. Abdelrasoul, A. Saman, P. Sebastian, (2016), A quantitative study of tuning ROS gmapping parameters and their effect on performing indoor 2D SLAM, 2016 2nd *IEEE International Symposium on Robotics and Manufacturing Automation* 46–55, 25th to 27th September 2016.

KİŞİSEL YAYIN VE ESERLER

Mando, R., Özer, E., İnner, B. (2022). Autonomous Mobile Robot Mapping and Exploration of an Indoor Environment. *International Conference on Computing, Intelligence and Data Analytics. Cham: Springer International Publishing*, Eylül 16-17, 2022.

ÖZGEÇMİŞ

İlk ve orta öğrenimini Suriye'de, lise öğrenimini ise Ürdün'de tamamladı. 2018 yılında JADARA Üniversitesi Yazılım Mühendisliği bölümünden mezun oldu ve bu mezuniyetinde birincilik elde etti. 2019 yılında Kocaeli Üniversitesi Bilgisayar Mühendisliği bölümünde yüksek lisans eğitimine başladı. 2019 yılından beri araştırma ve geliştirme mühendisi olarak kariyerine devam etmektedir.