Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı, 34, S. 254-258, Mart 2022

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue 34, pp. 254-258, March 2022

Research Article

TurtleBot 3 ile Ros Tabanlı Yol Planlama Uygulaması

Hazim İşcan[1], Ezgisu Tuncel[2*]

[1] Konya Teknik Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Bilgisayar Mühendisliği Bölümü, Konya, Türkiye, (ORCID: 0000-0002-3698-3745), hiscan@ktun.edu.tr

[2*] Konya Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Bilgisayar Mühendisliği Bölümü, Konya, Türkiye (ORCID: 0000-0001-9068-794X), tuncelezgisu@gmail.com

(2nd International Conference on Applied Engineering and Natural Sciences ICAENS 2022, March 10-13, 2022)

(**DOI:** 10.31590/ejosat.1081097)

ATIF/REFERENCE: İşcan, H. & Tuncel, E. (2022). TurtleBot 3 ile Ros Tabanlı Yol Planlama Uygulaması. *Avrupa Bilim ve Teknoloji Dergisi*, (34), 254-258.

Öz

Robot İşletim Sistemi (ROS) robotların kontrolünü sağlayan bir işletim sistemidir. Haritalama ve yer tespiti için, ROS içerisindeki Gmapping paketindeki, Eş Zamanlı Konum Belirleme ve Haritalama (SLAM) kullanılır. Gmapping, halihazırda oluşturulmuş olan harita parçaları ve sensör verileri temellidir. Her bir parçacık, robotun geçmiş pozisyon örneği ve harita üzerinde verilen önceki pozisyon örneğinin geçmişinin toplamıdır. Gmapping olasılıksal dağılım modeli robotun son gözlemlerini de hesaba katarak bir yayılım oluşturur. Harita oluşturulduktan sonra robotun hedefe gidebilmesi için yol planlamasının yapılması gerekir. Dijkstra Algoritması, hangi yolların keşfedileceğine öncelik vermemizi sağlar. Tüm olası yolları eşit olarak araştırmak yerine, daha düşük maliyetli yolları tercih eder. Yollarda ilerlemeyi teşvik etmek için daha düşük maliyetler, engellerden kaçınmak için daha yüksek maliyetler ve daha fazlası ayarlayanabilir. Dijkstra Algoritması tüm konumlara giden yolları bulabilir. A* Algoritması, Dijkstra Algoritmasının tek bir hedef için optimize edilmiş bir versiyonudur. A*, bir konuma veya birkaç konumun en yakınına giden yolları bulur. Bir hedefe daha yakın görünen yollara öncelik verir. Bu çalışmada Robot İşletim Sistemi (ROS) tabanlı yol planlama algoritmalarından A* ve Dijkstra'nın, Turtlebot 3 ile uygulaması ve analizi yapılmıştır. Uygulamada hedefe başarılı bir şekilde ulaşılmıştır.

Anahtar Kelimeler: A*, Dijkstra, Gmapping, Yol Planlama, Robotics, Ros, TurtleBot.

Ros Based Path Planning Application with TurtleBot 3

Abstract

Robot Operating System (ROS) is an operating system that provides control of robots. Simultaneous Positioning and Mapping (SLAM) in the Gmapping package in ROS is used for mapping and locating. Gmapping is based on already created map segments and sensor data. Each particle is the sum of the past position example of the robot and the history of the previous position example given on the map. The Gmapping probabilistic distribution model creates a spread, taking into account the robot's last observations. After the map is created, path planning must be done so that the robot can go to the destination. Dijkstra's Algorithm allows us to prioritize which paths to explore. Rather than exploring all possible routes equally, it prefers lower-cost routes. It can set lower costs to encourage progress on roads, higher costs to avoid obstacles, and more. Dijkstra's Algorithm can find paths to all locations. The A* Algorithm is a single-target optimized version of the Dijkstra Algorithm. A* finds paths to a location or the closest of several locations. Prioritizes paths that seem closer to a destination. In this study, the implementation and analysis of Robot Operating System (ROS) based path planning algorithms A* and Dijkstra with Turtlebot 3. In practice, the target has been successfully achieved.

Keywords: A*, Dijkstra, Gmapping, Path Planning, Robotics, Ros, TurtleBot.

[*] Corresponding Author: tuncelezgisu@gmail.com

1. Giriş

Mobil robotlarla ilgili çalışmalar 1960'larda başlamıştır. 1980'lerde dünya çapında yeni bir araştırma zirvesinin meydana gelmesiyle, General Electric, Honda ve Sony gibi dünyaca ünlü firmalar robot platformları geliştirmeye başladılar. Bu platformlar başlangıçta çoğunlukla üniversitelerde kullanılmaya başlandı. Laboratuvarlar ve araştırma enstitüleri tarafından mobil robotlar üzerine yapılan araştırmalar ve 1990'lı yıllardan itibaren bilgisayar biliminin, sensör teknolojisi ve yapay zekanın gelişmesiyle robotlar ile ilgili çalışmalar hızlanmıştır. Mobil robotların uygulamaları daha da ilerlemiştir. Geçtiğimiz birkaç on yılda tekerlekli mobil robotlar daha geniş çapta incelenmiştir. Mobil robotların endüstriyel uygulamalarda kullanımının gün geçtikçe artması, birçok araştırmacının daha fazla ilgisini çekmiş ve mobil robotlarda verimlilik ön plana çıkmıştır. Otomasyon ve robot teknolojisinin hızlı gelişimi ile birlikte, mobil robotlar için daha yüksek zorluklar ortaya çıkmaya başlamıştır. Karmaşık bir ortamda, navigasyon ve kontrol doğruluğu kritik hale gelmiştir. Bu da otonom navigasyona ve yörüngeye odaklanmayı gerektirmektedir. Otonom navigasyon, robotik araştırmaların önemli bir konusudur. Otonom navigasyon, bir robotun kaybolmadan veya diğer nesnelerle çarpışmadan bir konumdan diğerine güvenli bir şekilde hareket etmesi için doğru şekilde yapılması gereken bir görevdir. İç ortamda bulunan otonom bir robotun bulunduğu konumdan gitmesi gereken hedef noktaya ulaşabilmesi için robotun sensörler aracılığıyla çevresinden bilgi toplayabilmesi, sensör verilerini analiz edebilmesi, çalışma alanının bir modelini oluşturması ve ardından da çarpışmasız bir yol planlaması yapması gerekir.

Hu ve Yang (2004) tarafından bir mobil robot yol planlaması için bilgi tabanlı genetik algoritma (GA) kullanılmıştır. Çalışmalarını bir simülasyon ortamında test etmişlerdir.

Sarrif ve Bünyamin (2006) otonom robotlar için yol planlama algoritmalarına genel bir bakış sunmuştur.

Mohanty ve Parhi (2014) Cuckoo Search algoritmasını kullanmıştır. Yarı bilinen statik engellerin bulunduğu bir ortamda bir mobil robot navigasyonu planlamıştır.

Liang ve Lee (2015) çoklu yol planlama problemini çözmek için mobil robot navigasyonu için verimli bir yapay arı kolonisi (EABC) algoritması önermiştir.

Pandey ve Parhi (2017), mobil robot yol planlama problemini çözmk için bir tektonik tip-1 bulanık mantık sistemi (T1-SFLS) denetleyicisi ve Fuzzy-WDOmobil içeren hibrit algoritma geliştirmiştir. Bu algoritma ile statik ve dinamik bilinmeyen bir durumda çarpışmadan kaçınma sorununa odaklanmıştır.

Victerpaul ve ark. (2017), mobil robot yol planlamasına ilişkin kapsamlı bir çalışma sunmuştur. Yol içindeki çevredeki engellere odaklanarak gideceği yolu optimize eden algoritmalar üzerinde çalışmışlardır.

Bu çalışmada ROS tabanlı yol planlama algoritmaları incelenerek, TurtleBot 3'ün bulunduğu konumdan belirtilen hedefe gitmesi simülasyon ortamında gerçekleştirilmiştir.

2. Materyal ve Metot

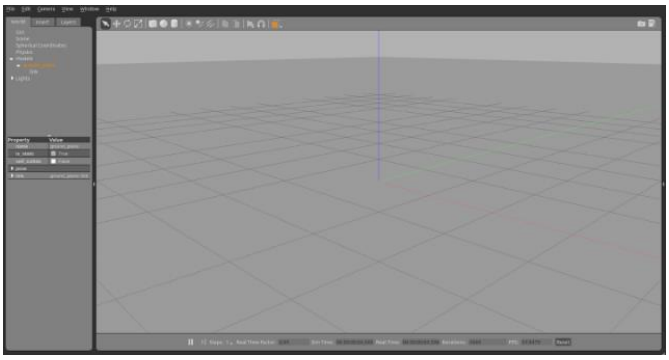
2.1. Robot İşetim Sistemi (ROS)

Robot İşletim Sistemi, araştırmacılar ve geliştiricilerin robotik uygulamalarda kullandığı, robot ve robot bileşenlerinin kontrol edilmesini sağlayan, programlama dili bağımsız, açık kaynak kodlu bir meta işletim sistemidir. Robotun sensörler yardımıyla dış dünyadan aldığı ses, görüntü gibi verileri tekrar robota komut olarak göndermeyi sağlayan yazılımdır. ROS, birbirleri ile iletişim kurabilen, işlem yapan birimlerden oluşur. Bire-çok abone modeli ve TCP / IP protokolü kullanarak söz konusu konular üzerinde iletişim kuran bağımsız bu modüllere düğüm (node) adı verilir. ROS, yayınlama/abone olma mantığında çalışır. Bilgisayar ve robot arasındaki yayınlama veya abone olma sırasındaki iletişim, topic'ler, service'ler ve action'lar ile sağlanır. Ana başlıkların detaylandırılması için 2. seviye ve 3. seviye başlıklar kullanılabilir (ROS, 2022).

2.2. Gazebo

Gazebo, açık ve kapalı mekanlar için geliştirilen bir 3 boyutlu dinamik gerçek dünya simülasyon yazılımıdır. 2002 yılında Linux ortamında açık kaynak olarak geliştirilmeye başlanmış ve 2009 yılında Ros ve PR2 Gazeboya entegre edilmiştir.

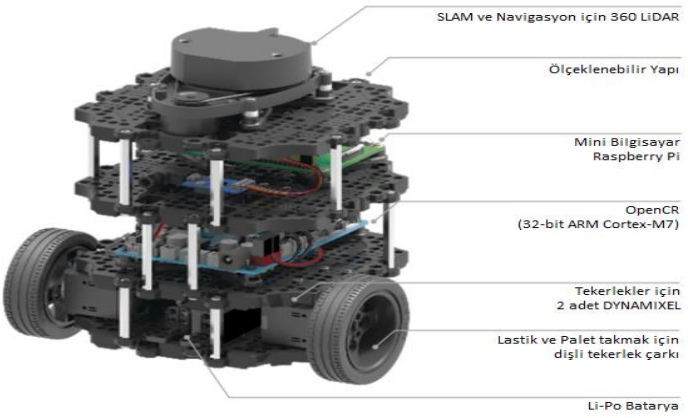
Fizik, sensörler, sahneleme gibi işlevler gelişmiş ve nettir. Gazebo temelde istemci sunucu ilişkisiyle çalışmaktadır. Gazebo server fiziksel işlemleri yaparken, gazebo client kullanıcının etkileşimini ve simülasyonun görselleştirilmesini sağlar. Gazebo ortamında robotlar gerçek dünyadaymış gibi hızlı bir şekilde test edilebilir ve çevresi ile sanal olarak etkileşime girmesi sağlanabilir (Şekil 1) (YZBS, 2022).



Şekil 1. Gazebo Simülasyonu

2.2. TurtleBot

TurtleBot, açık kaynaklı yazılıma sahip düşük maliyetli, kişisel bir robot kitidir. TurtleBot modelinin 3 versiyonu bulunmaktadır. TurtleBot1, Willow Garage'dan Tully (Open Robotics'te Platform Yöneticisi) ve Melonee (Fetch Robotics CEO'su) tarafından iRobot'un Roomba tabanlı araştırma robotu Create'in üzerinde ROS dağıtımı için 2010 yılında geliştirildi ve 2011'den beri satışta. 2012 yılında TurtleBot2, araştırma robotu iClebo Kobuki'ye dayalı olarak Yujin Robot tarafından geliştirildi. 2017 yılında TurtleBot3, önceki sürümlerin eksik olan işlevlerini ve kullanıcıların taleplerini tamamlayacak özelliklerle geliştirildi (Şekil 2) (TurtleBot, 2022; Robotis, 2022).



Şekil 2. TurtleBot 3

TurtleBot3, eğitim, araştırma, hobi ve ürün prototiplemede kullanılan ROS tabanlı bir mobil robottur. SLAM ve Navigasyon uygulamalarında çoğunlukla tercih edilmektedir. Gazebo üzerinde hazır modeli bulunduğundan robotun hareketleri simülasyon üzerinden gerçekleştirilebilir.

2.2. SLAM (-Simultaneous Localization and Mapping)

SLAM, eş zamanlı konum belirleme ve haritalamadır. Temelde iki işlemli bir süreçtir. Bilinmeyen bir yere yerleştirilen bir mobil robot için, haritalama olan ortamın haritasını oluşturur, çıktısını alır ve bunu yaparken robot, kendi konumunu lokalize eder. Yani SLAM, bilinmeyen ortamın haritasını oluşturmak ve aynı anda robotun yerini belirlemektir. Bu işlemi gerçekleştirirken robota takılan sensörler ortamın görselleştirilmesini ve haritanın oluşturulmasını sağlar. SLAM problemindeki amaç, başlangıçta robotun çevresini tamamen tanımadığı bir ortamda eş zamanlı olarak konum belirleme ve haritalama yapabilmesidir (Smith ve Cheesman, 1987).

SLAM çözümünde kullanılan 3 paradigmadan ilki ve en etkili olanı Genişletilmiş Kalman Filtresidir (Kalman, 1960). Konumlandırma yapılırken robotun konumunu yön bilgisi ile birlikte bulabilmektedir. Kalman filtresinin aksine genişletilmiş kalman filtresi, doğrusal olmayan sistemler ile de çalışabilir. Robotun hareket ve algı fonksiyonları doğrusal olmadığından bu filtre tercih edilir.

Hareket güncellemesi, robotun konumunun odometriden gelen verilerle güncellenmesidir. Algı güncellenmesi ise algılayıcı veriler ile yapılan güncellemedir.

Genişletilmiş Kalman Filtresinin gerçek zamanlı olarak robotun konumlarından ve işaretçi nesnelerin yerinden kendi konumunu n koşullu olasılığını yüksek derecede tahmin edebilmesidir. Bunu Gaussian Metotları yardımı ile yapar.

Son yıllarda popüler olmaya başlayan grafiksel sunum tabanlı paradigma, parçacık filtresidir. İki adımda gerçekleşen özyinelemeli bir algoritmadır. Bunlar tahmin ve güncelleme adımlarıdır. Parçacık Filtresi algoritması Genişletilmiş Kalman Filtresi ile benzer şekilde inanç değerlerini yaklaşık olarak hesaplar. Bu hesaplamalardaki zamansal gecikmeler kaynaklı zamanı yakınsama hataları oluşur. Temeli bir değişkenin tüm olasılık dağılım fonksiyonunun örnekleme tabanlı bir temsilini yapmasıdır. Bu örnekleme süreçlere bağlıdır, değişkenin değerleri süreçlerden çıkan sonuçlara göre değişkenin durumu değişir. Bu değişkenin birden fazla parçacık kopyası bulunur. Ve bu kopyaların her birinin kendisiyle ilişkilendirilmiş bir ağırlığı bulunur. Bu kopyalarının her birinin ağırlığı alınarak ana değişkenin değeri hakkında bir tahmin yapılır.

SLAM problemini doğrusal olmayan seyrek optimizasyon ile çözer. Graf tabanlı SLAM çözümünde temel sezgiler akıcıdır. Yer işaretleri ve robotun pozisyonu grafın üzerindeki birer düğümdür. Her yer işareti çifti birbirine yayla bağlıdır. Bunlar odometrinin okunması ve iletilmesiyle temsil edilirler. Robotun hissetmesiyle (sensörlerden) diğer yer işaretleri ve konum bilgileri de zamanla oluşur. Graf tabanlı SLAM'in temel mantığı; grafiği oluşturup, kısıtlamalar tarafından getirilen hatayı en aza indirecek bir düğüm yapılandırması bulmak üzerinedir (Bailey ve Durrant-Whyte, 2006).

2.2. Gmapping

Gmapping, halihazırda oluşturulmuş olan harita parçaları ve sensör verileri temellidir. Belirli bir SLAM algoritmasının bir uygulamasıdır. Gmapping, ROS'ta tanımlı bir algoritmadır. Bu nedenle ROS üzerinde tekrar programlanmasına gerek yoktur, sadece doğru düğüm çalıştırılmalı veya dosya terminalde başlatılmalıdır.

Gmapping paketi, slam_gmapping ROS düğümünü içeren mobil robotun konumu üzerinde lazer sensör ölçümlerine ilişkin verileri kullanarak, 2 boyutlu bir harita oluşturur. slam_gmapping düğümü, haritayı oluşturmak için gerekli verileri elde etmek için /scan ve /tf konularına abonedir. Bu düğüm lazerden gelen verileri ve dönüşümleri okur ve ardından bu verilere dayalı olarak OGM (Occupancy Grid Map - Doluluk Izgara Haritası) oluşturur. Oluşturulmakta olan harita, haritalama süreci boyunca konu/harita üzerinde yayınlanmaktadır. Konu/harita, nav_msgs/ OccupancyGrid türünde bir mesaj kullanır. OGM'de doluluk, 0 ile 100 arasında bir tamsayı değeri ile temsil edilir. 0 değeri tamamen boş ve 100 tamamen dolu (ör. duvar) anlamına gelir ve özel bir değer -1'dir, bu da bilinmeyen bir alan anlamına gelir.

Robotun konum ve yönünün takibi için Reo – Blackwellized parçacık filtresi kullanılır (Grisetti vd., 2007). Gmapping, robotun son gözlemlerini de hesaba katarak bir yayılım oluşturur ve bu sayede etkin bir şekilde haritalama yapılarak belirsizlikler de aynı zamanda ortadan kalkar. Odometri verileri olmadan gmapping ile haritalama yapılamaz. Dört adımdan oluşur:

Örnekleme: Genellikle, olasılık odometri hareket modeli teklif dağılımı olarak kullanılır.

Önem Ağırlığı: Her parçaya, önem örnekleme ilkesine göre bireysel bir önem ağırlığı atanır.

Yeniden Örnekleme: Parçacıklar, önem ağırlığına oranla orantılı olarak çizilir. Sürekli bir dağılıma yaklaşmak için yalnızca sonlu sayıda parçacık kullanıldığından bu adım gereklidir. Yeniden örneklemeden sonra tüm parçacıkların ağırlığı aynıdır.

Harita Tahmini: Her bir parçacık için o parçacığın yörüngesi ve gözlemleri temelli karşılık gelen harita tahmini hesaplanır.

2.2. Yol Planlama

Yol planlama, bulunulan nokta ve hedef nokta arasındaki ardışık noktalardan oluşan geometrik bir şekil olarak tanımlanan optimal yoldur. Yol planlama, çevrenin bir haritasını ve robotun haritaya göre konumunun farkında olmasını gerektirir.

En yaygın harita, doluluk ızgara haritasıdır. Bir ızgara haritasında çevre, üzerinde engellerin işaretlendiği, örneğin 1cm x 1cm gibi keyfi çözünürlükte karelere ayrılır. Olasılıklı bir doluluk ızgarasında, ızgara hücreleri bir engel içerme olasılığıyla

da işaretlenebilir. Bu, özellikle bir engeli algılayan robotun konumu belirsiz olduğunda önemlidir. Izgara haritalarının dezavantajları, büyük bellek gereksinimlerinin yanı sıra çok sayıda tepe noktasına sahip veri yapılarını geçmek için hesaplama süresidir. İkinci soruna bir çözüm, tüm odaları köşeler olarak kodlayan ve aralarında gezilebilir bağlantıları belirtmek için kenarları kullanan topolojik haritalardır.

Yol planlamasının görevi, mobil robotun bulunduğu konumdan hedef noktaya karşılaştığı engellerle çarpışmadan ilerlemesi için optimal yolun oluşturulmasıdır. İki farklı yol planlaması bulunmaktadır. Bunlar, Küresel Yol Planlayıcı ve Yerel Yol Planlayıcı. Küresel planlayıcı, haritadan tam yolu hesaplamak için kullanılır ve bir haritada iki konum arasında bir yol bulur. Yerel planlayıcı ise küresel bir plan verildiğinde mobil robotun tekerleklerine hangi hız komutunun gönderileceğini hesaplar.

2.2. A Star Algoritması (A*)

A* algoritması, bir sonraki adımda hangi yolun izleneceğine karar vermeye yardımcı olmak için kullanılır. Sezgisel yaklaşım sergileyen A* algoritması grafik geçişlerinde çok kullanışlıdır. Algoritmada, her adımda hesaplama yapan sezgisel fonksiyon f(n), değeri en düşük düğümü alır ve bu düğümü kuyruktan çıkarır. Gidilen bu düğüme göre komşu olan bütün düğümlerin değerleri güncellenir. Bu iki adım hedefe ulaşılıncaya kadar tekrarlanır. Denklem 1'de A* algoritmasının fonksiyonu verilmiştir.

$$f(n) = g(n) + h(n) \quad (1)$$

f(n), hesaplama yapan sezgisel fonksiyon.

g(n, Başlangıç düğümünden mevcut düğüme olan maliyet

h(n), Mevcut düğümden hedef düğüme olan tahmin edilen mesafe.

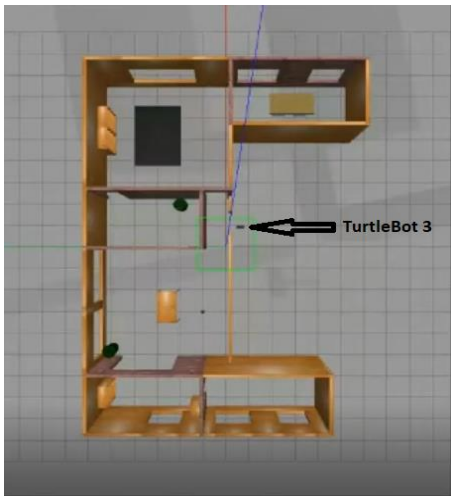
A* algoritması yalnızca hedef noktaya yaklaştıran düğümleri keşfetmeye odaklanır ve böylece yol problemini hızlı bir şekilde çözer (Hart vd., 1968).

2.2. Dijkstra Algoritması

Dikstra Algoritmasında başlangıç düğümünden tüm düğümlere olan uzaklık kesin bir sonuçla elde edilir. Bu elde edilen en kısa yol, Dijkstra algoritmasının sezgisel fonksiyon değeri sıfır olduğu için, kesin olarak en kısa yoldur. Belirlenen başlangıç noktasından diğer noktalara olan maliyet belirlenir ve düşük maliyetli nokta işaretlenir. Bu işaretlenen noktadan gidilebilen noktalar arasında da adımlar uygulanmaya devam eder. Dijkstra, keşfetmeye başlangıç düğümünden başlayarak, bu düğümü bitişe bağlayan düğümleri bir listeye ekleyerek tek bir yol tanımlar. Bu ise Dijkstra'nın başlangıç düğümüne olan uzaklıklarına göre sırayla düğümleri taradığı anlamına gelir. Başlangıç noktasından da dışarı doğru genişledikçe, hedeften daha uzak düğümleri de listeye ekleyeceğinden maliyetli bir hesaplama yöntemi olur (Dijkstra 1959).

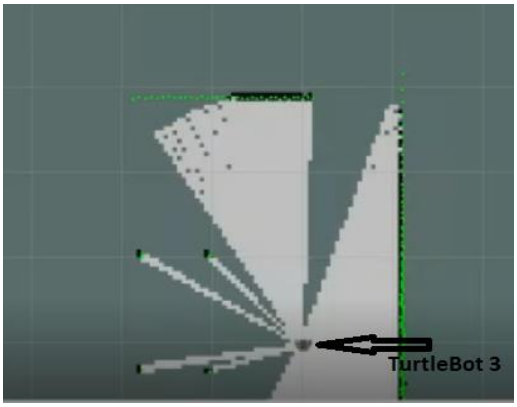
2.2. Uygulama

Öncelikle Gazebo ortamı çalıştırılır. Gazebo ortamında Şekil 3'de görüldüğü gibi örnek bir alan oluşturulur.



Şekil 3. Gazebo Simülasyonu

Daha sonra haritalama yapabilmek için SLAM paketinden Gmapping çalıştırılır. Bu modül sayesinde, robot gezerek tüm örnek alanın haritasını oluşturur. Şekil 4'de harita oluşturma aşaması, Şekil 5'te ise oluşturulan haritanın son hali gösterilmektedir. Beyaz alanlar, taranan yani haritası oluşturulan kısımları, siyah alanlar engelleri, yeşil noktalar ise lazer taramasını göstermektedir.



Şekil 4. Harita oluşturma

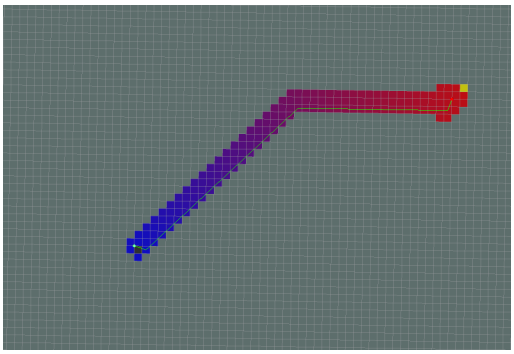
Şekil 5. Haritanın Son Hali

Robotun haritalamaya başladığı konum, koordinat anlamında 0,0 olarak kabul edilir. Haritalama işlemi bittikten sonra robot belirlenen bir hedefe, hedefin koordinat bilgisi verilerek gönderilebilir. Belirlenen hedefe ulaşma, yol planlaması

bölümüne girer ve bu çalışmada yol planlaması için A* ve Dijkstra algoritmaları ayrı ayrı kullanılmıştır.

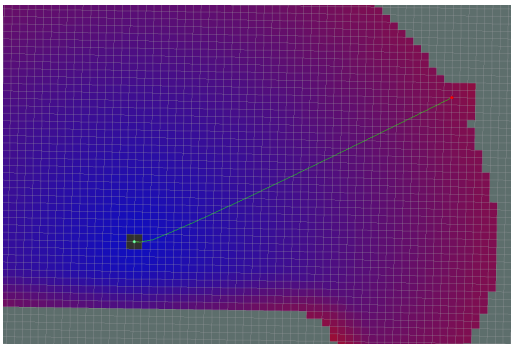
3. Araştırma Sonuçları ve Tartışma

Şekil 6'da harita üzerinde A* algoritmasının hedef noktaya ulaşmak için belirlediği düğümler ve yol haritası gösterilmektedir.



Şekil 6. A algoritmasın örnek yol planı*

Şekil 7'de Dijkstra algoritmasının belirlediği düğümler ve yol haritası gösterilmektedir.



Şekil 7. Dijkstra algoritmasın örnek yol planı

Mobil robotlarda yol planlama problemi navigasyon alanında önemli bir problemdir. Mobil robot başlangıç durumundan hedef duruma, engellerden kaçınarak, optimal veya optimale yakın bir yol bulabilir. Buradaki performans paramatreleri; düşük çalısma maliyeti, en kısa rota ve en kısa süre gibi değerlerdir. Bu durumda yol planlamasında kullanılacak algoritmanın seçimi önemlidir.

A* algoritması maliyeti düşük olan yola öncelik verirken Dijkstra algoritması tüm olası yolları keşfetmek ister. Birden fazla hedef nokta var ve bu noktalardan hangisinin en yakın olduğu bilinmiyorsa Dijkstra algoritmasını kullanmak avantajlıdır. A* kullanılamamasının sebebi her hedef nokta için yeniden algoritmanın çalıştırılması gerektiğidir.

Problem hakkında iyi bir sezgisel tahmine sahip olunduğu göz önüne alınırsa A* algoritmasını kullanmak daha verimli ve hızlı bir sonuç elde edilmesini sağlar.

4. Sonuç

Bu çalışmada, Gazebo simülasyon programında otonom mobil robot TurtleBot 3'ün, bulunduğu konumdan belirlenen hedefe gitmesi amaçlanmıştır. Bunu gerçekleştirmek için yol planlama algoritmalarından A* ve Dijkstra algoritmaları ayrı ayrı kullanılmıştır.

Dijkstra algoritması bulunduğu noktadan gidebileceği tüm noktaları tarayarak en az maliyetli olanı seçerek hedefe olan minimum mesafeyi bulmaya çalışır. Bu nedenle çalışma süresi A*'a göre daha uzundur.

A* algoritması ise maliyeti düşük olan noktaya öncelik vererek minimum yolu hesaplamaya çalışır. Fakat A* her zaman en iyi yolu vermez. En az maliyetli noktayı tercih ederken gidilebilecek minimum yol daha da uzar.

TurtleBot 3, Gmapping SLAM algoritmasıyla, haritalama ve konum belirlemesini gerçekleştirmiştir. Bulunduğu ortamın haritasını çıkarıp, yol planlaması için kullanılan her iki algoritma ile belirlenen hedefe başarıyla ulaşmıştır.

Kaynakça

Bailey,T. ve Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part i The essential algorithms. IEEE Robot Autom. Mag., 13(2): 99-108.

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. Numer. Math. 1, 269–271.

Grisetti, G., Stachniss, C. ve Burgard, W. (2007). Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE Trans. Robot., 23(1): 3446.

Hart,P. E., Nilsson, N. J. ve Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on Systems Science and Cybernetics. 4 (2): 100–107.

Hu, Y., ve Yang, S. X. (2004). A knowledge based genetic algorithm for path planning of a mobile robot. In IEEE International Conference on Robotics and Automation,. Proceedings. ICRA'04. 2004 (Vol. 5, pp. 4350-4355). IEEE.

Kalman, R.E. (1960). A New approach to linear filtering and prediction problems. J. Basic Eng., 82: 35–45.

Liang, J. H. ve Lee, C. H. (2015). Efficient collision-free path-planning of multiple mobile robots system using efficient artificial bee colony algorithm. Advances in Engineering Software, 79, 47-56.

Mohanty, P. K. ve Parhi, D. R. (2016). Optimal path planning for a mobile robot using cuckoo search algorithm, Journal of Experimental & Theoretical Artificial Intelligence, 28:1-2, 35-52.

Pandey, A. ve Parhi, D. R. (2017). Optimum path planning of mobile robot in unknown static and dynamic environments using Fuzzy Wind Driven Optimization algorithm, Defence Technology.

Robotis website. (2022 Şubat 25). https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

ROS website. (2022 Şubat 25). https://www.ros.org

Sariff, N. ve Bunyamin, N. (2006). "An Overview of Autonomous Mobile Robot Path Planning Algorithms", 4th Student Conference on Research and Development, Selangor, Malaysia.

Smith, R. ve Cheesman, P. (1987). On the representation of spatial uncertainty. Int. J. Rob. Res., 5(4): 56-68S.

TurtleBot website. (2022 Şubat 25). https://www.turtlebot.com/

Victerpaul, P., Saravanan, D., Janakiraman, S. ve Pradeep, J. (2017). Path planning of autonomous mobile robots: A survey and comparison, Journal of Advanced Research in Dynamical and Control Systems.

YZBS, Yapay Zeka ve Benzetim Sistemleri Arge Laboratuvar websitesi. (2022 Şubat 25). https://www.yapbenzetkocaeli.edu.tr/gazebo-giris/