KSÜ Mühendislik Bilimleri Dergisi, 26(2), 2023
Araştırma Makalesi

KSU J Eng Sci, 26(2), 2023
Research Article

Kahramanmaras Sutcu Imam University
Journal of Engineering Sciences

Geliş Tarihi : 06.09.2022
Kabul Tarihi :28.10.2022

Received Date : 06.09.2022
Accepted Date : 28.10.2022

ARAMA ALGORİTMALARININ YOL PLANLAMASI PERFORMANSLARININ KARŞILAŞTIRMALI ANALİZİ

PERFORMANCE ANALYSIS OF SEARCH ALGORITHMS FOR PATH PLANNING

Mehmet GÖK [1*] (ORCID: 0000-0003-1656-5770)
Öznur Şifa AKÇAM [2] (ORCID: 0000-0003-1458-3342)
Mehmet TEKEREK [3] (ORCID: 0000-0001-6112-3651)

[1] Kahramanmaraş İstiklal Üniversitesi, Dijital Oyun Tasarımı Bölümü, Kahramanmaraş, Türkiye
[2,3] Kahramanmaraş Sütçü İmam Üniversitesi, Bilişim Sistemleri Anabilim Dalı, Kahramanmaraş, Türkiye

*Sorumlu Yazar / Corresponding Author: Mehmet GÖK, gokmehmet@outlook.com

ÖZET

Haritası bilinen ya da bilinmeyen herhangi bir ortamda, otonom mobil robotların başlangıç noktasından hedef noktasına en az maliyetle ve en hızlı ulaşımı yol planlaması ile gerçekleştirilir. Yol planlamasında alternatif yollar arasından optimum yolun seçimi önemlidir. Bu çalışmada, yol planlama görevini yerine getirmek amacıyla farklı arama yaklaşımlarına sahip algoritmaların Robot İşletim Sistemine (ROS) entegrasyonu ve performanslarının karşılaştırılması yapılmıştır. Bu amaçla, tasarımı görüntü işleme yazılımı kullanılarak yapılan ve RViz arayüzünde yayınlanan örnek haritalar üzerinde BFS, DFS, Dijkstra, Bellman-Ford, A* ve RRT algoritmaları kullanılarak yol planlaması gerçekleştirilmiştir. Yol planlama işleminin değerlendirilmesi için en kısa yol ve en kısa süre ölçütleri dikkate alınmıştır. Elde edilen bulgular, en kısa yolun planlanmasında A*; en kısa sürede planlama işleminin gerçekleştirilmesinde ise DFS algoritmasının ön plana çıktığını göstermektedir. Ayrıca, çalışmada literatüre katkı olarak, ROS ortamında farklı senaryolar için kullanılabilecek algoritmaların performans ölçümü için bir test ortamı da sunulmuştur.

Anahtar Kelimeler: Yol planlama, arama algoritmaları, GIMP, ROS, RViz

ABSTRACT

In any environment, whether the map is known or unknown, the fastest and least costly transportation of autonomous mobile robots from the starting point to the target point is realized by path planning. In path planning, it is important to choose the optimum route among alternative paths. In this study, the integration of algorithms with different search approaches into the Robot Operating System (ROS) and their performance were compared in order to fulfill the path planning task. For this purpose, path planning was carried out using BFS, DFS, Dijkstra, Bellman-Ford, A* and RRT algorithms on sample maps designed using image processing software and published in the RViz interface. The shortest route and shortest time criteria were taken into account for the evaluation of the path planning process. Obtained findings show that A* in planning the shortest route; It shows that the DFS algorithm comes to the forefront in realizing the planning process in the shortest time. In addition, as a contribution to the literature, a test environment for the performance measurement of algorithms that can be used for different scenarios in the ROS environment is presented.

Keywords: Path planning, search algorithm, GIMP, ROS, Rviz

ToCite: GÖK, M., AKÇAM, Ö. Ş., & TEKEREK, M., (2023). ARAMA ALGORİTMALARININ YOL PLANLAMASI PERFORMANSLARININ KARŞILAŞTIRMALI ANALİZİ. *Kahramanmaraş Sütçü İmam Üniversitesi Mühendislik Bilimleri Dergisi*, 26(2), 379-394.

GİRİŞ

Bir nesnenin, insanın ya da aracın; bulunduğu konumdan farklı bir konuma hareket etmesi için iki konum arasındaki güzergâhların belirlenmesine ihtiyaç duyulur. Güzergâh belirlemede bulunulan konum başlangıç noktasını, gidilmek istenen konum ise hedef noktayı temsil eder. Başlangıç noktasından hedef noktaya gidilebilecek birçok yol bulunabilir. Yollar arasından en kısa mesafe ve en kısa sürede hedef noktaya ulaşmayı sağlayacak olan yol tercih edilmektedir. Bu durum, navigasyon sistemlerinde, mobil robotların hareketinde ve insansız araçların hareket kontrolünün sağlanması gibi alanlarda yol planlama problemi olarak tanımlanmaktadır (Zhang vd., 2018). Yol planlama probleminde amaç; başlangıç noktasından hedef noktaya en kısa yoldan, çarpışmasız (collision free) ve en hızlı şekilde ulaşabilmektir (Koubâa vd., 2018). Bu amacı gerçekleştirebilmek için yollar arasında arama yapılır. Söz konusu yolu arama işlemi ise bir problem çözme yöntemi olduğu için uygulanabilirliğini sınırlayan faktör, verimlilik ile ilgilidir (Korf, 1999). Verimliliği belirleyen unsurlar ise ortam haritasının bilinip bilinmediği, engellerin boyutları ve konumları, iki konum arasındaki en kısa mesafe ile bu mesafeyi gitmek için gereken süre bilgileridir.

Yol planlama probleminde, başlangıç, hedef ve aradaki noktalara ait bilgilerin nasıl saklandığına bağlı olarak bir liste, dizi (array), ağaç (tree) ya da çizge (graph) üzerinde arama işlemi gerçekleştirilir. Bu işlem için arama algoritmaları kullanılır. Arama algoritmaları, bilmeden arama (uninformed search) ve bilerek arama (informed search) olmak üzere iki farklı arama stratejisi kullanmaktadır (Pathak vd., 2018). Bilmeden arama yaklaşımında kullanılan algoritmalar, problemin tanımı ve çözüm ile ilgili bilgiye sahip değildir. Bu algoritmalarda sadece mevcut durum bilinir ve olası tüm durumlara mevcut durum üzerinden ulaşılabileceği varsayılmaktadır. Bilerek arama algoritmalarında ise çözümü verimli hale getirmek için sezgisel (heuristic) bir yaklaşım kullanılmaktadır (Edelkamp & Schrodl, 2011; Pathak vd., 2018).

Arama algoritmaları; ihtiyaca göre yol planlama görevini (path planning task), önceden bilinen bir harita üzerinde ya da bilinmeyen bir ortamda gerçekleştirebilir (Dewang vd., 2018). Örneğin bir temizlik robotu, tam kapsama işlemini gerçekleştirebilmek için ortamın harita bilgisine ihtiyaç duyar. Öte yandan bir arama kurtarma robotu, çoğunlukla bilinmeyen bir ortamda çalışır. Bu ayrışma, arama algoritmalarının bilerek ya da bilmeden arama yaklaşımında bulunmasının önem kazanmasına neden olmuş ve araştırmacıları farklı algoritmalar üzerinde çalışma yapmaya yönlendirmiştir (Kaur, 2019).

Literatürde yer alan yol planlama algoritmaları olarak da kullanılan arama algoritmaları klasik, sezgisel (heuristic) ve çizge (graph) algoritmaları olarak sınıflandırılabilir (Patle vd., 2019). Hücre ayrışımı (cell decomposition), potansiyel alan (potential field), örnekleme tabanlı (sampling based) gibi yöntemler klasik yöntemler içerisinde yer alır (Mac vd., 2016; Injarapu & Gawre, 2017). Bu yöntemler, yol planlama probleminin araştırıldığı ilk yıllarda yaygın olarak kullanılmıştır (Campbell vd., 2020; Tan vd., 2021). Fakat klasik yöntemlerin küresel optimizasyon ve zaman karmaşıklığı gibi konulardaki eksikliklerinden dolayı sezgisel yöntemlerin öne çıktığı görülmektedir (Koubâa vd., 2018). Doğadan esinlenilerek geliştirilen genetik (genetic) (Tuncer & Yildirim, 2012; Samadi & Othman, 2013), parçacık sürü optimizasyonu (particle swarm optimization) (Wang vd., 2018; Zhang vd., 2021), karınca kolonisi optimizasyonu (ant colony optimization) (Dorigo vd. 2006; Akka & Khaber, 2018) gibi algoritmaların yer aldığı; yapay sinir ağı (Shamsfakhr & Bigham, 2017; Sung vd., 2021) ve bulanık mantık (Pradhan vd., 2009; Mohanty vd., 2020) gibi yaklaşımlar ile oluşturulan sezgisel yöntemler klasik yöntemlere göre daha verimli çözüm üretmek için tasarlanmıştır. Ayrıca son yıllarda A*, Dijkstra, Belman-Ford gibi çizge arama algoritmaları da geliştirilmiştir (Siegwart vd., 2011; Koubâa vd., 2018).

Yol planlama algoritmalarının karşılaştırılması ile ilgili yapılan çalışmalar incelendiğinde; He vd., (2021), A* ve RRT algoritmalarını kullanarak 2B labirent üzerinde yol planlaması gerçekleştirmiştir. Deneysel gözlemler Matlab ortamında yapılmıştır. Performans ölçütü olarak yol uzunluğu, zaman ve keşfedilen düğümler incelenmiş ve algoritmaların labirentteki yol planlama performansları karşılaştırılmıştır. Araştırmacılar çalışmanın sonunda, A*'ın labirenti verimli ve hızlı bir şekilde geçmek için, RRT'nin ise labirenti keşfetmek açısından uygun olduğunu gözlemlemişlerdir.

Yol planlama problemine çözüm önerisi sunmak için farklı özellikli algoritmaların bir arada kullanıldığı çalışmalar da vardır. Wang vd., (2018) yapmış oldukları çalışmada, RRT, Dijkstra ve A* algoritmaları ile dinamik pencere yaklaşımını (Dynamic Windows Approach-DWA) beraber kullanarak robotun navigasyon görevini gerçekleştirmişlerdir. Araştırmacılar çalışmalarını Robot İşletim Sisteminin (ROS) sağlamış olduğu Gazebo

simülasyon ortamında ve Rviz (ROS Visualization) görüntüleme aracında test etmişlerdir. Yol uzunluğu ve planlaması, ortalama hız ile çalışma zamanı ölçütlerini performans değerlendirmesi için kullanmışlardır.

Yol planlama sorunu; oyun endüstrisinde de çözülmek istenen problemlerden biridir. Barnouti vd., (2016) tarafından yapılan çalışmada, strateji oyunlarından görüntüler ve labirent (maze) görüntüleri kullanılmıştır. Araştırmacılar tarafından Visual Basic programlama dili ile bir deney ortamı tasarlanmış ve A* algoritmasını kullanarak görüntüler üzerinde iki nokta arasındaki en kısa yolu bulmaya çalışmıştır. Permana vd., (2018) de labirent tabanlı oyunlarda tercih edilebilecek uygun algoritmayı bulmak için Maze Runner oyununda A*, Dijkstra ve enine öncelikli arama (Breadth First Search) algoritmalarının yol planlama performanslarının analizini gerçekleştirmiştir. Labirent oyununda (maze game) en hızlı ve en güvenilir rotayı bulmak için Iloh, (2022), aynı boyutta farklı engelleri bulunan üç labirent ile DFS, BFS ve A* algoritmalarının performanslarını labirentleri çözme süreleri ve yol uzunluğu açısından karşılaştırmıştır.

Oyunlarda yol planlama problemine çözüm sunmak ve mobil robotların en kısa yoldan hedefe ulaşabilmelerini gözlemlemek için labirent yapısı kullanılabilmektedir. Aqel vd., (2017), enine öncelikli arama (Breadth First Search), en iyi en önce arama (Best First Search) ve A* algoritmalarını karşılaştırmış, mobil robotun labirenti çözme süresini azaltan ve en kısa yolu bulabilen bir yöntem önermişlerdir. BFS algoritması ile geliştirdikleri yöntemi ve duvar takip yöntemini gerçek labirentler üzerinde test ederek performanslarını değerlendirmişlerdir.

En kısa yol problemine çözüm bulmak için farklı algoritmaların incelendiği çalışmalar da vardır. Bu çalışmalarda, Chan vd., (2016), Dijkstra, Simetrik Dijkstra, A*, Belman-Ford, Floyd-Warshall ve Genetik algoritmalarının yol planlama performanslarını otobüs güzergahına ait örnek veriler ile zaman ve mesafe açısından karşılaştırmıştır. AbuSalim vd., (2020) ise Dijkstra ve Bellman-Ford algoritmalarını başlangıç ve hedef düğüm arasındaki en kısa yolu bulmak için geçen süre açısından karşılaştırmışlardır. AbuSalim ve ekibinin elde ettiği sonuçlara göre; düğüm sayısı az olduğunda Bellman-Ford algoritmasının en kısa yolu bulma süresi Dijkstra algoritmasına göre daha kısadır. Düğüm sayısı arttıkça Bellman-Ford algoritmasının verimliliği düşerken Dijkstra algoritmasının verimliliği artmakta ve en kısa yolu daha hızlı bulmaktadır.

Yol planlama görevinde; ortamın harita bilgisi, arama uzayının boyutu, ortamda bulunan engellerin hareketli ya da hareketsiz olması arama algoritmalarının performansını etkileyen faktörlerdir. Arama algoritmalarını aynı ortamda test edebilmek için bu faktörlerin etkisini en aza indirerek deney ortamının oluşturulması gerekmektedir. Bu çalışmada ortamın haritasının önceden bilindiği, sabit boyutlu bir harita kullanıldığı ve engellerin hareketsiz olduğu varsayılarak deney ortamını temsil eden ikili labirent (binary maze) tasarlanmıştır. Bu varsayımlar doğrultusunda ikili labirent üzerinde BFS, DFS, Dijkstra, Bellman-Ford, A* ve RRT algoritmalarının performans analizlerinin yapılması ve en verimli algoritmanın bulunması hedeflenmiştir. Bu algoritmaların tercih edilmesinin nedeni, ikili labirent veri yapısı için kolaylıkla uyarlanabilir olmalarıdır. İkili labirent veri yapısı ise, ROS ortamında haritalama için kullanılan ızgara haritası (grid map) veri yapısına kolaylıkla dönüştürülebilmektedir. Seçilen algoritmaların performans analizini gerçekleştirmek için planlanan yol uzunluğu (path length) ve planlama süresi ölçütleri karşılaştırılmıştır. Labirentte yol planlama görevi için arama algoritmalarının karşılaştırıldığı çalışmalar vardır ancak belirtilen görev için altı arama algoritmasının karşılaştırılmalı analizinin yapıldığı çalışmaya rastlanmamıştır.

Araştırma kapsamında; GIMP (GNU Image Manipulation Program) görüntü işleme yazılımı ile boyutları aynı engelleri farklı olan üç farklı ikili labirent haritası hazırlanmıştır (GIMP, 2022). Bu haritalar ve seçilen algoritmaların planladığı yollar RViz aracı kullanılarak görselleştirilmiştir. Planlanan yol uzunluğu ve algoritmaların planlama süreleri hesaplanmıştır.

YÖNTEM

Bu çalışmada; boyutları aynı, engel yapısı farklı üç harita üzerinde BFS, DFS, Dijkstra, Belman-Ford, A* ve RRT algoritmalarının yol planlama performanslarının analizi gerçekleştirilmiştir. Bu analiz için algoritmaların planladıkları yolun uzunlukları ve planlama süreleri değerlendirme ölçütü olarak ele alınmıştır. Çalışmada kullanılan algoritmalar aşağıda tanımlanmıştır.

Çalışmada Kullanılan Algoritmalar

Enine Öncelikli Arama (Breadth First Search) Algoritması

BFS algoritması ağaç veya çizge üzerinde dolaşmak için kullanılır (Pathak vd., 2018). Enine yayılım ile en yakındaki düğümler taranarak uzaktaki düğümlere gidilir (Indriyono, 2021). Dolaşılan düğümlere ait bilgiler, ilk giren ilk çıkar (FIFO-First in first out) kuyruk (queue) veri yapısında tutulmaktadır (Paulino vd., 2021).

Derinine Öncelikli Arama (Depth First Search) Algoritması

DFS algoritması ağaç veya çizge üzerinde dolaşmak için kullanılan özyinelemeli (recursive) bir algoritmadır. Arama uzayında sürekli derine inerek arama işlemini gerçekleştirir. Dolaşılan düğümlere ait bilgiler son giren ilk çıkar (LIFO – Last in first out) yığın (stack) veri yapısında tutulmaktadır (Pathak vd., 2018; Paulino vd., 2021).

Dijkstra Algoritması

Dijkstra algoritması; çizge tabanlı ve açgözlü (greedy) (Tan vd., 2021) yaklaşımına dayanan en kısa yolu en az maliyetle hesaplamayı amaçlayan bir algoritmadır. Dijkstra algoritmasında başlangıçta; kaynak düğümün mesafesi 0, diğer düğümlerin mesafesi sonsuz (∞) olarak kabul edilir. Kaynak düğümden erişilebilen komşu düğümlere olan mesafeler hesaplanır. Komşu düğümlerden en kısa mesafeye sahip düğüm seçilir ve diğer düğümlerin mesafesi bu düğüme göre güncellenir. Algoritma, hedef düğümü bulana kadar özyinelemeli olarak çalışır (Chan vd., 2016; Mukhlif & Saif, 2020).

Bellman-Ford Algoritması

Bellman-Ford algoritması; çizge tabanlı ve tek kaynaklı yol planlama problemini çözmek için kullanılan bir algoritmadır (AbuSalim vd., 2020). Çalışma yapısı olarak Dijkstra algoritmasına benzemektedir. Kaynak düğümden başlayarak diğer düğümlere gidilebilecek en kısa yol maliyetini hesaplamayı hedefler. Algoritmanın başlangıcında; kaynak düğümün maliyeti (mesafe) 0, diğer düğümlerin maliyeti sonsuz (∞) olarak kabul edilir. Komşu düğümlere erişim oldukça diğer düğümlere olan uzaklık değerleri güncellenir (Chan vd., 2016).

A Algoritması*

A* algoritması; sezgisel (heuristic) arama yaklaşımına dayanır (Debnath vd.,2019). İki konum arasındaki en kısa yolu bulmayı hedefler. A* algoritması iki konum arasındaki maliyeti hesaplarken; başlangıç düğümünün komşu düğümlere olan gerçek uzaklık değerini (g(n)) ve komşu düğümlerin hedef noktaya olan sezgisel yani tahmini uzaklık değerini (h(n)) dikkate alır (Mukhlif & Saif, 2020). Eşitlik 1'de görüldüğü gibi gerçek uzaklık değeri ile sezgisel uzaklık değerinin toplamı (f(n)); başlangıç noktasından hedef noktaya bulunan yolun maliyetinin en iyi tahminini verir.

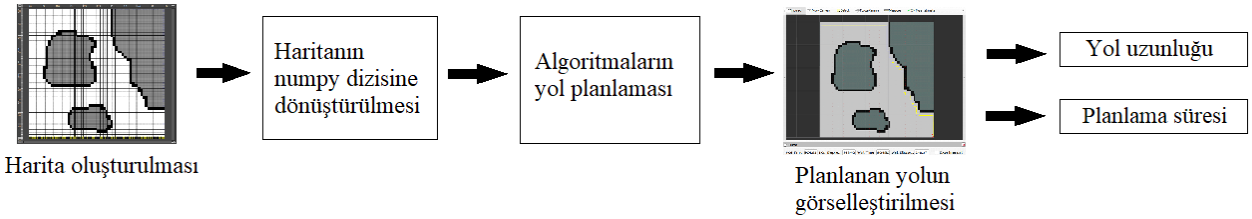
$$f(n) = g(n) + h(n) \quad (1)$$

Hızlı Rastgele Keşfeden Ağaçlar (Random Rapidly Exploring Trees) Algoritması

RRT algoritması; örnekleme tabanlı (sample based) bir algoritmadır (Tan vd., 2021). Rastgele (stokastik) değerler üreterek olasılıksal bir yaklaşımla arama işlemini gerçekleştirir (Niu vd., 2021). RRT algoritmasında başlangıç noktası kök düğüm olarak belirlenir. Arama uzayında rastgele bir alan belirlenir. Rastgele değerlerin üretilmesi ile kök düğüme en yakın yaprak düğümler ağaca eklenir ve rastgele genişletilmiş bir ağaç oluşturulur. Hedef noktaya ulaşılıncaya kadar ağaç genişletilmeye devam edilir. Bu şekilde başlangıç düğümünden hedef düğüme doğru bir yol oluşur (Elbanhawi & Simic, 2014). RRT algoritması arama kurtarma çalışmaları gibi arama uzayının bilinmediği durumlarda etkin çözümler sunabilir. Ancak rastgele değerler üreterek keşfedilmemiş bölgeleri aramaya yöneldiği için hem en az maliyetli ve en kısa yolu bulması hem de her defasında başlangıç ve hedef nokta arasında aynı yolu bulması mümkün olmamaktadır (He vd., 2021).

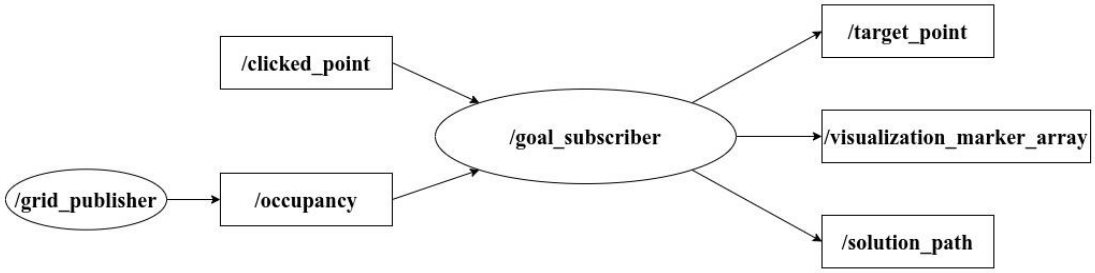
Deney Ortamının Kurulması

Algoritmaların test edileceği deney ortamı, algoritmaların uygulanacağı haritanın oluşturulması, haritanın iki boyutlu *numpy* dizisine dönüştürülmesi, arama algoritmalarının yol planlama görevini gerçekleştirmesi ve RViz görüntüleme aracında planlanan yolun görselleştirilmesi süreçlerini içermektedir (Şekil 1).



Şekil 1. Deney Ortamının Oluşturulması

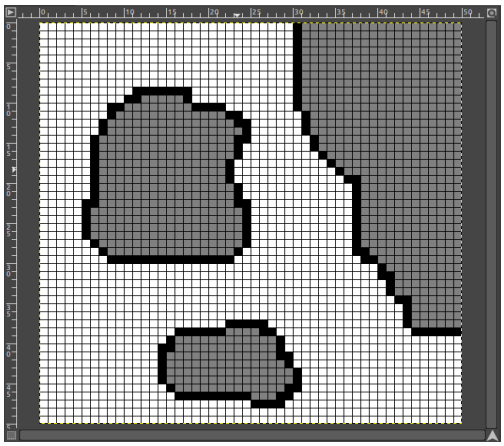
ROS düğümlerinde (ROS Nodes) seçilen algoritmalar kullanılarak hem planlanan yolun harita üzerindeki görüntüsünün elde edilmesinde hem de yol uzunluğu ve planlama sürelerinin hesaplanmasında Python programlama dili ve *rospy* kütüphanesi kullanılmıştır. ROS düğümlerinin hesaplama grafiğini, düğümleri, abone olunan ve yayınlanan ROS başlıkları (ROS topics) ile veri akışını gösteren (ROS computation graph) diyagram Şekil 2'de verilmiştir.



Şekil 2. ROS Düğümleri Hesaplama Grafiği

Şekil 2'de verilen diyagramda, ROS düğümleri oval, başlıklar ise dörtgen şekillerde verilmiştir. `clicked_point` başlığı ile tıklanan noktaya yol planlaması, RViz arayüzünde harita üstünde tıklanan noktanın koordinatlarının yayınlanması sağlanır. `target_point` başlığı ise tıklanan hedef noktada X işaretinin RViz arayüzünde görüntülenmesi için kullanılır. `solution_path` başlığı ile planlanan yolun noktalar halinde görüntülenmesi sağlanır. `visualization_marker_array` başlığı ile noktalar arasında çizgisel işaretleme yapılır.

Haritanın oluşturulması için GIMP görüntü işleme yazılımı kullanılmıştır (Şekil 3). Görüntü dosyası PGM (Portable Gray Map) formatında olup ROS'un gmapping haritalama paketi tarafından kullanılan formattır. Görüntü dosyası bir Python script ile arama algoritmalarının üzerinde hareket edebileceği iki boyutlu bir *numpy* dizi (array) biçimine dönüştürülmektedir. Algoritmalar aracılığıyla dizi üzerindeki noktalarda engel olup olmadığını kontrol ederek başlangıç noktasından hedef noktaya nasıl gidileceğine dair yol planlanır.



Şekil 3. GIMP Ortamında *Occupancy Grid* Oluşturulması

ROS ortamındaki haritalar, robotun bulunduğu ortamın fiziksel olarak dolu olması durumunu ifade eden occupancy grid adı verilen iki boyutlu bir dizi biçiminde saklanmaktadır. Occupancy grid için üç bölge tanımlanmıştır: serbest alan, bilinmeyen alan ve engeli teşkil eden alan. Bu alanların görüntü dosyasındaki piksel renk değerleri Tablo 1'de

verilmiştir. Görüntü işleme yazılımı ile oluşturulan harita 50 piksel x 50 piksel boyutunda olup; harita çözünürlük oranı 0,1 olarak seçilmiştir. Buna göre 10 piksel, 1 metrelik uzunluğa karşılık gelmektedir.

Tablo 1. Harita Alan Değerleri

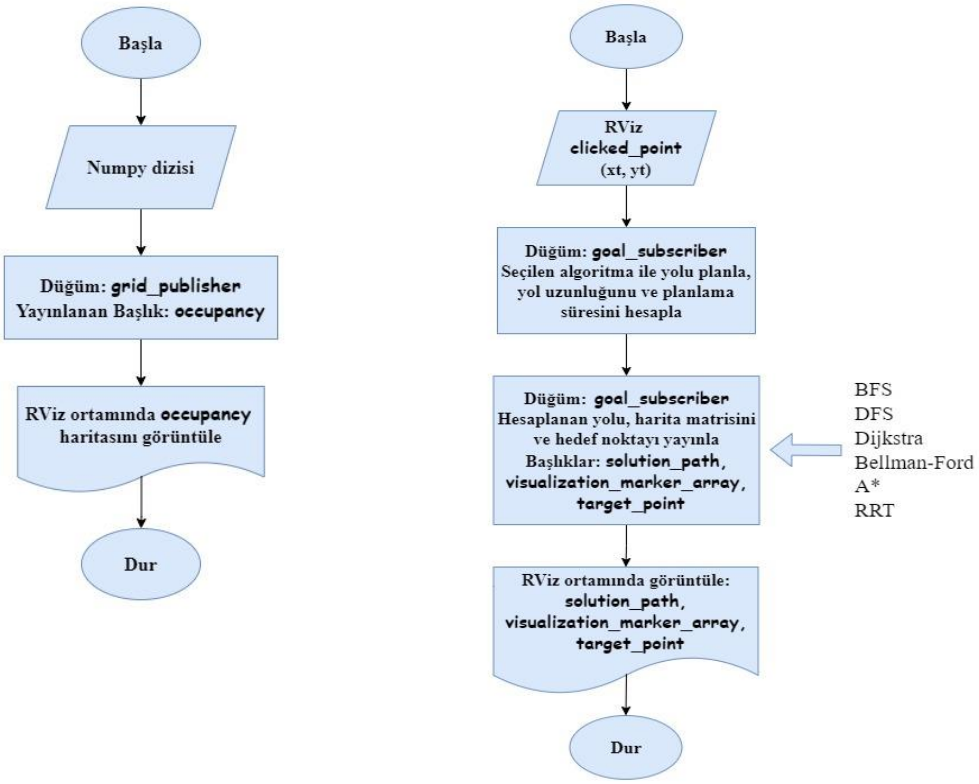
	Occupancy Değeri	Piksel Renk Değeri
Bilinmeyen Alan (Unknown)	-1	128 (Gri)
Serbest Alan (Free)	0	255 (Beyaz)
Engelli Alan (Occupied space)	100	0 (Siyah)

Görüntüden *numpy* dizisine dönüştürülen harita, `grid_publisher` adlı bir düğüm tarafından occupancy başlığı altında yayınlanmakta ve RViz aracı ile görüntülenmektedir (Şekil 4).

Şekil 4. GIMP Ortamında Oluşturulan Haritanın RViz Aracında Görüntülenmesi

`grid_publisher` düğümü, harita dizisini işlerken, piksel renk değerlerini occupancy değerlerine dönüştürmektedir. Yayınlanan harita bilgisi, aboneler tarafından ikili labirent (binary maze) olarak işlenebilmektedir. Şekil 4'te görüldüğü gibi, harita üzerindeki serbest alanlar açık gri ile, engeller ise siyah ile gösterilmektedir. Koyu gri ile gösterilen alan ise bilinmeyen ya da keşfedilmemiş bölgeyi (unexplored area) ifade etmektedir.

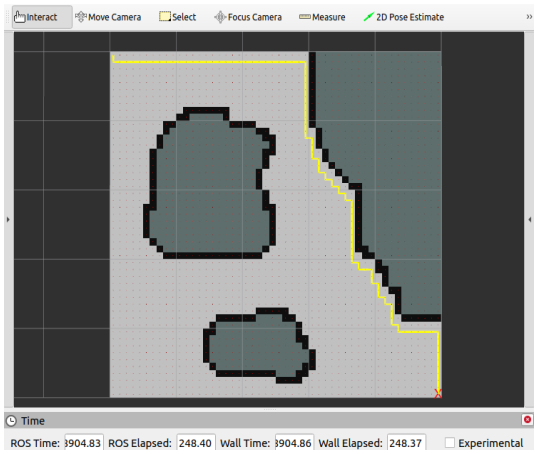
RViz aracında görüntülenen harita üzerinde tıklanan herhangi bir noktaya *(0, 0)* noktasından yol planlaması yapılmaktadır. Yol planlaması yapan `goal_subscriber` adlı düğüm, labirent matrisini, seçilen arama algoritması ile çözüp hesaplanan yolu (planned path) ve hedef noktayı yayınlayıp RViz üzerinde görüntülenmesini sağlar. Şekil 5 (a) ve Şekil 5 (b)' de sırasıyla RViz ortamında yayınlanan haritaya ve planlama algoritmalarının çalışmasına ilişkin akış diyagramlarına yer verilmektedir.



(a) (b)

Şekil 5. a. Harita Yayını Akış Diyagramı, **b.** Yol Planlama Algoritmaları Çalıştırılması Akış Diyagramı

`goal_subsciber` düğümü RViz tarafından yayınlanan `clicked_point` adlı başlığa abone olarak, tıklanan hedef nokta koordinatlarını alır ve seçilen algoritmaya göre yol planlamasını yapar. Şekil 6'da örnek olarak planlanmış yol ve tıklanmış hedef koordinat kırmızı renkli X işareti ile gösterilmektedir.



Şekil 6. Planlanan Yolun RViz Ortamında Görüntülenmesi

Bu çalışmada kullanılan arama algoritmaları, normalde düğüm ve kenarlar üzerinde çalışan sürümlerinden binary maze formatındaki iki boyutlu dizi üzerinde çalışan sürümlerine dönüştürülmüştür. Planlama yapan `goal_subscriber` düğümü, seçilen algoritmayı bu dizi üzerinde çalıştırmaktadır. Planlama işlemi tamamlandıktan sonra hesaplanan yol için uzunluk ve hesaplama süresi de terminal penceresi üzerinde yazılarak, kullanıcıya bilgi verilmesi sağlanmıştır.

Deney

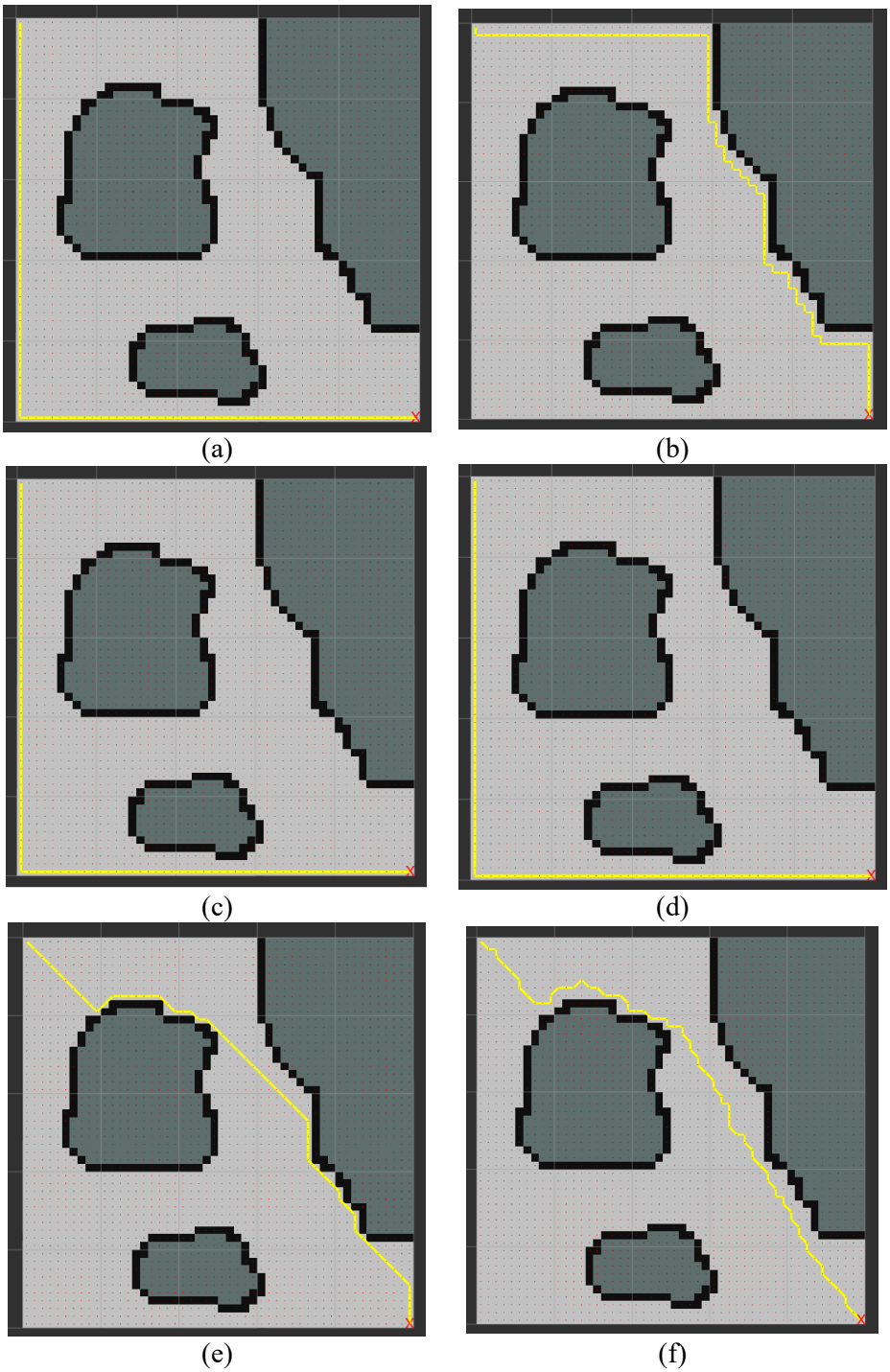
Araştırma sürecinde planlanan testler, Intel Core i7 4510U işlemci, 8 GB DDR3 RAM, Nvidia GT840M ekran kartı özelliklerine sahip, Ubuntu 20.04 işletim sistemi kurulu bilgisayarda üç farklı harita üzerinde gerçeklenmiştir. Her

bir harita için ayrı ayrı seçilen 6 farklı algoritmanın yol planlamasını yapması ve planlama zamanı karşılaştırılarak hangi durumda hangi algoritmanın başarılı olduğu yönünde bir karşılaştırma matrisi oluşturulmuştur.

BULGULAR ve TARTIŞMA

Birinci Harita Yol Planlaması ve Süreleri

Görüntü işleme yazılımı ile oluşturulan harita 50 piksel x 50 piksel boyutunda olduğu ve harita çözünürlük değeri 0,1 olarak seçildiği için tasarlanan harita 5 metre x 5 metre boyutundadır. Birinci harita için; *(0, 0)* başlangıç noktasından *(49, 49)* hedef noktasına BFS, DFS, Dijkstra, Bellman-Ford, A* ve RRT algoritmaları kullanılarak yapılan yol planlaması görülmektedir (Şekil 7).



Şekil 7. Birinci Harita için (0, 0) Başlangıç Noktasından (49, 49) Hedef Noktasına **a.** BFS, **b.** DFS, **c.** Dijkstra, **d.** Bellman-Ford, **e.** A* ve **f.** RRT Algoritmaları ile Yol Planlamasının Gerçekleştirilmesi

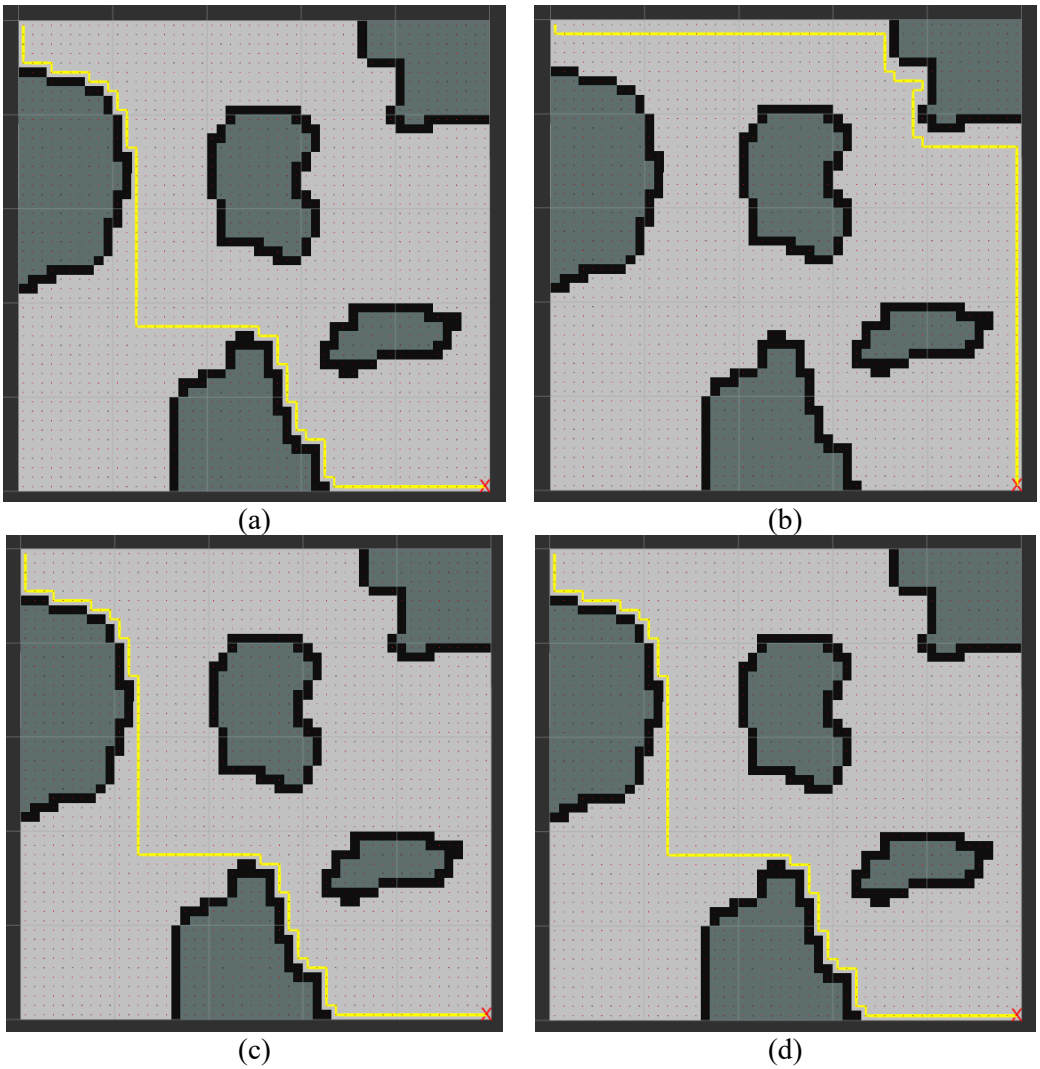
Şekil 7'de gösterilen hesaplanan yol uzunluklarının 0,1 ile çarpılması sonucunda metre cinsinden mesafeler bulunur. Birinci harita için test edilen algoritmaların yol uzunluklarına göre yol planlama görevini gerçekleştirme süreleri Tablo 2'de verilmiştir.

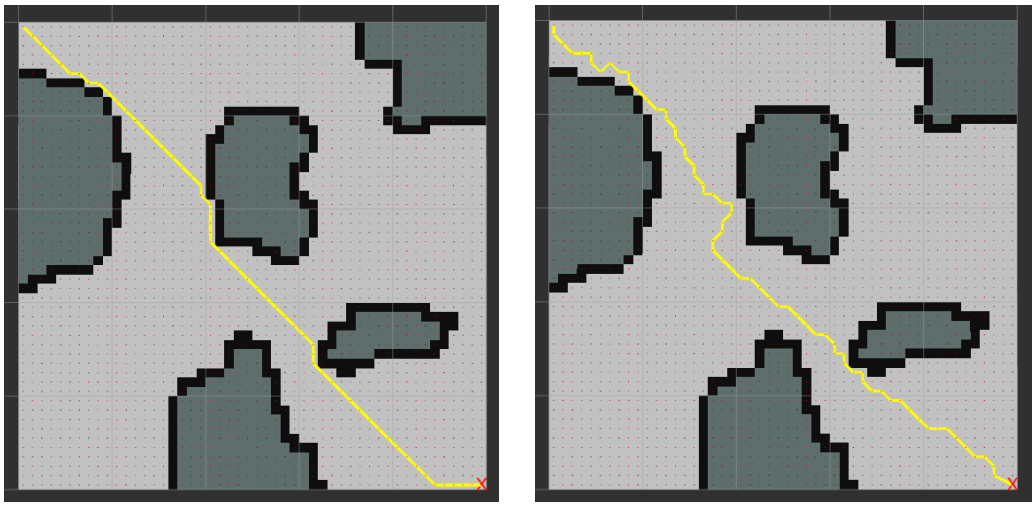
Tablo 2. Birinci Harita için Yol Planlama Görevini Gerçekleştiren Algoritmalara Ait Yol Uzunluğu ve Planlama Süreleri

Algoritma	Yol uzunluğu (m)	Planlama süresi (sn.)
BFS	9,80	0,004491
DFS	9,80	0,000802
Dijkstra	9,80	0,059042
Bellman-Ford	9,80	0,062946
A*	7,856	0,028852
RRT	8,525	0,067119

İkinci Harita Yol Planlaması ve Süreleri

İkinci harita için; *(0, 0)* başlangıç noktasından *(49, 49)* hedef noktasına BFS, DFS, Dijkstra, Bellman-Ford, A* ve RRT algoritmaları kullanılarak yapılan yol planlaması görülmektedir (Şekil 8).





(e) (f)

Şekil 8. İkinci Harita için (0, 0) Başlangıç Noktasından (49, 49) Hedef Noktasına **a.** BFS, **b.** DFS, **c.** Dijkstra, **d.** Bellman-Ford, **e.** A* ve **f.** RRT Algoritmaları ile Yol Planlamasının Gerçekleştirilmesi

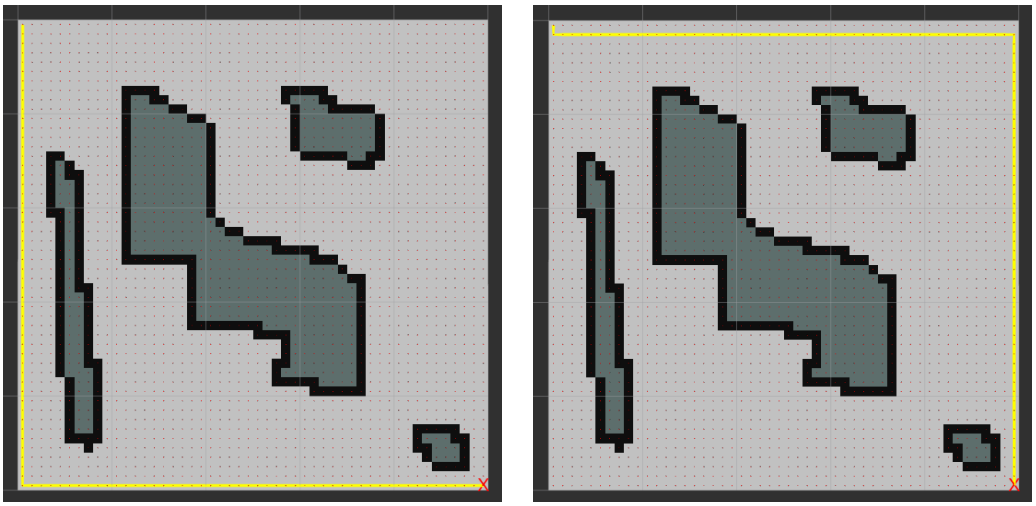
İkinci harita için test edilen algoritmaların yol uzunluklarına göre yol planlama görevini gerçekleştirme süreleri Tablo 3'te verilmiştir.

Tablo 3. İkinci Harita için Yol Planlama Görevini Gerçekleştiren Algoritmalara Ait Yol Uzunluğu ve Planlama Süreleri

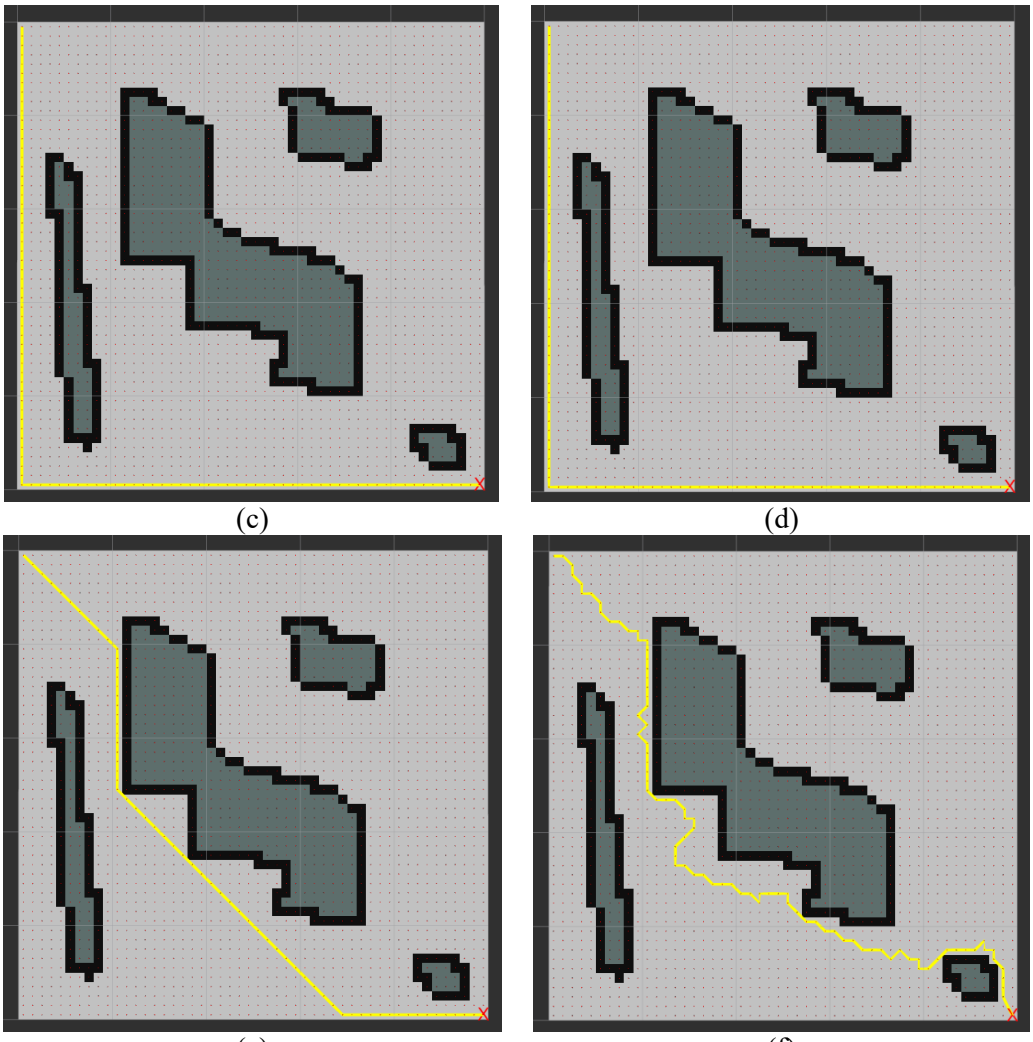
Algoritma	Yol uzunluğu (m)	Planlama süresi (sn.)
BFS	9,80	0,010481
DFS	10,00	0,001173
Dijkstra	9,80	0,054577
Bellman-Ford	9,80	0,061564
A*	7,339	0,024165
RRT	8,273	0,066652

Üçüncü Harita Yol Planlaması ve Süreleri

Üçüncü harita için; *(0, 0)* başlangıç noktasından *(49, 49)* hedef noktasına BFS, DFS, Dijkstra, Bellman-Ford, A* ve RRT algoritmaları kullanılarak yapılan yol planlaması görülmektedir (Şekil 9).



(a) (b)



(c) (d)

(e) (f)

Şekil 9. Üçüncü Harita için (0, 0) Başlangıç Noktasından (49, 49) Hedef Noktasına **a.** BFS, **b.** DFS, **c.** Dijkstra, **d.** Bellman-Ford, **e.** A* ve **f.** RRT Algoritmaları ile Yol Planlamasının Gerçekleştirilmesi

Üçüncü harita için test edilen algoritmaların yol uzunluklarına göre yol planlama görevini gerçekleştirme süreleri Tablo 4'te verilmiştir.

Tablo 4. Üçüncü Harita için Yol Planlama Görevini Gerçekleştiren Algoritmalara Ait Yol Uzunluğu ve Planlama Süreleri

Algoritma	Yol uzunluğu (m)	Planlama süresi (sn.)
BFS	9,80	0,009923
DFS	9,80	0,001157
Dijkstra	9,80	0,091006
Bellman-Ford	9,80	0,114769
A*	7,808	0,032760
RRT	9,824	0,643496

Seçilen algoritmaların yol planlama süreleri ve planlanan yol uzunluklarına ait veriler (Tablo 2, Tablo 3 ve Tablo 4) incelenmiştir. Tablolardan elde edilen verilere göre en kısa planlama süresini 0,000802 sn ile 1. haritadaki DFS algoritmasının, en kısa yol uzunluğunun ise 7,339 m ile 2. haritadaki A* algoritmasın hesapladığı görülmektedir.

Karşılaştırma için minimum planlama süresi ve minimum yol uzunluğu değerleri kullanılmıştır. Tablo 2, Tablo 3 ve Tablo 4'teki planlama süreleri, minimum planlama süresi (0,000802 sn) değerine bölünmesiyle Tablo 5 elde edilmiştir. Tablo 5'de üç harita da minimum yol planlama süresi açısından karşılaştırmalı olarak sunulmuştur.

Tablo 5. Haritaların Minimum Yol Planlama Süresi Açısından Değerlendirilmesi

Harita	**Birinci Harita**		**İkinci Harita**		**Üçüncü Harita**	
Algoritma	**Yol uzunluğu (m)**	**Planlama süresi (sn.)**	**Yol uzunluğu (m)**	**Planlama süresi (sn.)**	**Yol uzunluğu (m)**	**Planlama süresi (sn.)**
BFS	9,8	5,5998	9,8	13,0686	9,8	12,3728
DFS	9,8	**1,0000**	10	**1,4626**	9,8	**1,4426**
Dijkstra	9,8	73,6185	9,8	68,0511	9,8	113,4738
Bellman-Ford	9,8	78,4863	9,8	76,7631	9,8	143,1035
A*	7,856	35,9751	7,339	30,1309	7,808	40,8479
RRT	8,525	83,6895	8,273	83,1072	9,824	802,3641

Tablo 5 incelendiğinde en kısa sürede yol planlamasının haritaya göre değişkenlik göstermekle birlikte DFS algoritması ile gerçekleştiği ve RRT algoritmasının ise yine haritaya göre değişkenlik göstererek yol planlama süresinin en fazla olduğu görülmüştür. Ayrıca RRT algoritmasında hesaplanan yol uzunluğunun artmasının yol planlaması için harcanan hesaplama süresini artırdığı söylenebilir. Yine BFS, Dijsktra, Bellman-Ford ve A* algoritmalarının yol planlama sürelerinin ise haritaya göre değişkenlik gösterdiği görülmektedir.

Tablo 2, Tablo 3 ve Tablo 4'teki yol uzunluklarının minimum yol uzunluğu (7,339 m) değerine bölünmesiyle Tablo 6 elde edilmiştir. Tablo 6'da üç harita da minimum yol uzunluğu açısından karşılaştırmalı olarak sunulmuştur.

Tablo 6. Haritaların Minimum Yol Uzunluğu Açısından Değerlendirilmesi

Harita	**Birinci Harita**		**İkinci Harita**		**Üçüncü Harita**	
Algoritma	**Yol uzunluğu (m)**	**Planlama süresi (sn.)**	**Yol uzunluğu (m)**	**Planlama süresi (sn.)**	**Yol uzunluğu (m)**	**Planlama süresi (sn.)**
BFS	1,3353	0,0045	1,3353	0,0105	1,3353	0,0099
DFS	1,3353	0,0008	1,3626	0,0012	1,3353	0,0012
Dijkstra	1,3353	0,0590	1,3353	0,0546	1,3353	0,0910
Bellman-Ford	1,3353	0,0629	1,3353	0,0616	1,3353	0,1148
A*	**1,0704**	0,0289	**1,0000**	0,0242	**1,0639**	0,0328
RRT	1,1616	0,0671	1,1273	0,0667	1,3386	0,6435

Tablo 6 incelendiğinde en kısa yol planlamasının haritaya göre değişkenlik göstermekle birlikte A* algoritması ile gerçekleştiği görülmüştür. Haritaya göre değişkenlik göstermekle birlikte DFS algoritması tarafından planlanan yol uzunluğu 2. haritada en fazla olurken 3. haritada RRT algoritması ile planlanan yol uzunluğunun en fazla olduğu görülmüştür. BFS, Dijsktra ve Bellman-Ford algoritmalarının yol uzunlukları değerleri ise üç haritada da aynıdır.

Tablo 5 ve Tablo 6'da minimum yol ve minimum süre açısından arama algoritmalarının yol planlama performansları karşılaştırıldığında planlanan yollar arasındaki farkın fazla olmadığı ancak hesaplama süreleri arasındaki farkların daha fazla olduğu gözlemlenmiştir. Hesaplama süresi açısından en iyi sonucu DFS algoritması, yol uzunluğu açısından en iyi sonucu A* algoritması vermektedir.

Tablo 7'de bu çalışmada önerilen yöntemin literatürdeki benzer çalışmalarla karşılaştırılmasına yer verilmiştir. Çalışmalar incelendiğinde her çalışmada elde edilen sonuçların değişkenlik gösterdiği ve kullanılan algoritmaların ele alınan parametreler doğrultusunda farklı sonuçlar verdiği görülmektedir.

Tablo 7. Önerilen Yöntemin Benzer Çalışmalarla Karşılaştırılması

Yazar ve Yıl	Kullanılan Algoritmalar	Deney Ortamı	Elde Edilen Sonuçlar
He vd. (2021)	A* RRT	2 Boyutlu labirent (Matlab)	Yol uzunluğu, Çalışma süresi, Keşfedilen düğüm sayısı
Wang vd. (2018)	RRT-DWA Dijkstra-DWA A*-DWA	Gazebo ve RViz (ROS)	Yol uzunluğu, Planlama süresi, Ortalama hız, Çalışma süresi
Barnouti vd. (2016)	A*	Strateji oyunlarından elde edilen görüntüler ve labirent görüntüleri (Visual Basic)	Planlanan en kısa yol
Permana vd. (2018)	A* Dijkstra BFS	Maze Runner oyunu	Yol uzunluğu, Çalışma süresi, Blok sayısı
Iloh (2022)	BFS DFS A*	2 Boyutlu labirent (Tkinter-Python)	Yol uzunluğu, Çalışma süresi
Chan vd. (2016)	Dijkstra Simetrik Dijkstra A* Bellman-Ford Floyd-Warshall Genetik	Otobüs güzergâhına ait veriler çizge olarak temsil edilmiş, farklı bir ortam tasarlanmamıştır.	Çalışma süresi, Toplam mesafe
AbuSalim vd. (2020)	Dijkstra Bellman-Ford	Farklı düğüm sayısına sahip çizgeler üzerinde test edilmiş, ayrıca bir ortam tasarlanmamıştır.	Düğüm sayısı, Çalışma süresi
Önerilen yöntem	BFS DFS Dijkstra Bellman-Ford A* RRT	İkili labirent (GIMP-Python)	Yol uzunluğu, Planlama süresi

Tablo 7'de verilen çalışmalardan Chan vd. (2016) ile AbuSalim vd. (2020) kullandıkları algoritmaları çizge yapısı üzerinde test etmişler ancak farklı bir deney ortamı tasarlayarak gözlemlememişlerdir. Diğer çalışmalarda ise farklı platformlar kullanılarak farklı harita veya labirentler üzerinde seçilen algoritmalar test edilerek sonuçlar gözlemlenmiştir. Önerilen yöntemde tasarlanan ikili labirent haritaları literatürde yer alan farklı bir çalışmada daha önce kullanılmamış olup benzer çalışmalardaki haritalar örnek alınarak bu çalışma için tasarlanmıştır. Çalışmanın ayrılan yönleri şu şekilde sıralanabilir;

- Algoritmalar düğüm yapısında değil ikili labirent yapısında çalıştırılmaktadır,
- Algoritmaların testine yönelik deney ortamına ait haritalar çalışmaya özgü olarak tasarlanmıştır,
- Seçilen algoritmalar dışında yol planlama problemine çözüm sunabilen diğer algoritmalar da tasarlanan deney ortamında test edilebilir.

SONUÇ

Bu çalışmada boyutları aynı engelleri farklı üç adet ikili labirent haritası ile deney ortamı oluşturulmuştur. Haritalar 5m x 5m boyutlarındadır. Haritaların oluşturulması için GIMP görüntü işleme yazılımı kullanılmıştır. BFS, DFS, Dijkstra, Bellman-Ford, A* ve RRT arama algoritmaları ile ikili labirent haritaları üzerinde yol planlama görevi gerçekleştirilmiştir. Planlanan yol uzunlukları ve planlama süreleri hesaplanmıştır. Planlanan yollar RViz aracı ile görüntülenmiştir. Seçilen algoritmaların yol planlama performansları planlanan yol uzunluğu ve planlama süresi ölçütleri açısından analiz edilmiştir.

Elde edilen sonuçlara göre; A* algoritması optimum yolu bulmakta, DFS algoritması ise en kısa sürede hedefe ulaşmaktadır. Bulgular ışığında sürenin önemli olduğu durumlarda DFS algoritması, en kısa yol istendiğinde A* algoritması, keşif amaçlı çalışmalarda ise RRT algoritmasının tercih edilebileceği söylenebilir. Çalışmada sunulan

yöntem hem planlanan yolun mesafesinin hem de planlama sürelerinin kolaylıkla hesaplanabilmesi yönüyle bu tür araştırmalar için uygulanabilir olduğunu göstermektedir.

Harita içinde etkin bir yolun planlanması kadar; yolu oluşturan noktaların yol formasyonunu bozmayacak şekilde takip edilmesi de büyük önem arz etmektedir. İleriki çalışmalarda, takip algoritmalarının robotun sürüşünde gereken hız değerinin hesaplanması üzerine yoğunlaşılabilir.

KAYNAKLAR

AbuSalim, S. W., Ibrahim, R., Saringat, M. Z., Jamel, S., & Wahab, J. A. (2020, September). Comparative analysis between dijkstra and bellman-ford algorithms in shortest path optimization. In IOP Conference Series: Materials Science and Engineering, 917(1), 012077. doi: 10.1088/1757-899X/917/1/012077

Akka, K., & Khaber, F. (2018). Mobile robot path planning using an improved ant colony optimization. International Journal of Advanced Robotic Systems, 15(3), 1729881418774673. doi: 10.1177/1729881418774673

Aqel, M. O., Issa, A., Khdair, M., ElHabbash, M., AbuBaker, M., & Massoud, M. (2017, October). Intelligent maze solving robot based on image processing and graph theory algorithms. In 2017 International Conference on Promising Electronic Technologies (ICPET) (pp. 48-53). IEEE. doi: 10.1109/ICPET.2017.15

Barnouti, N. H., Al-Dabbagh, S. S. M., & Naser, M. A. S. (2016). Pathfinding in strategy games and maze solving using A* search algorithm. Journal of Computer and Communications, 4(11), 15. doi: 10.4236/jcc.2016.411002

Campbell, S., O'Mahony, N., Carvalho, A., Krpalkova, L., Riordan, D., & Walsh, J. (2020, February). Path planning techniques for mobile robots a review. In 2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE) (pp. 12-16). IEEE. doi: 10.1109/ICMRE49073.2020.9065187.

Chan, S. Y. M., Adnan, N. A., Sukri, S. S., & Wan Zainon, W. M. N. (2016). An experiment on the performance of shortest path algorithm. Knowledge Management International Conference (KMICe) (pp. 7-12). Chiang Mai, Thailand. Erişim adresi: https://repo.uum.edu.my/id/eprint/20010/

Debnath, S. K., Omar, R., Latip, N. B. A., Shelyna, S., Nadira, E., Melor, C. K. N. C. K., & Natarajan, E. (2019). A review on graph search algorithms for optimal energy efficient path planning for an unmanned air vehicle. Indonesian Journal of Electrical Engineering and Computer Science, 15(2), 743-749. doi: 10.11591/ijeecs.v15.i2.pp743-749

Dewang, H. S., Mohanty, P. K., & Kundu, S. (2018). A robust path planning for mobile robot using smart particle swarm optimization. Procedia computer science, 133, 290-297. doi: 10.1016/j.procs.2018.07.036

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE computational intelligence magazine, 1(4), 28-39. doi: 10.1109/MCI.2006.329691.

Edelkamp, S., & Schrodl, S. (2011). Basic Search Algorithms. In R. Roumeliotis, & D. Bevans (Eds.), Heuristic search: theory and applications, (pp. 47-86). Waltham, USA: Elsevier.

Elbanhawi, M., & Simic, M. (2014). Sampling-based robot motion planning: a review. IEEE Access, 2, 56-77. doi: 10.1109/ACCESS.2014.2302442

GIMP, www.gimp.org, Erişim tarihi: 25.06.2022.

He, Y., Wang, P., & Zhang, J. (2021, September). A Comparison Between A* & RRT in maze solving problem. In 2021 3rd International Symposium on Robotics & Intelligent Manufacturing Technology (ISRIMT) (pp. 333-338). IEEE. doi: 10.1109/ISRIMT53730.2021.9596830

Iloh, P. C. (2022). A Comprehensive and comparative study of DFS, BFS, and A* search algorithms in a solving the maze transversal problem. International Journal of Social Sciences and Scientific Studies, 2(2), 482-490. Erişim adresi: https://www.ijssass.com/index.php/ijssass/article/view/54

Indriyono, B. V. (2021). Optimization of breadth-first search algorithm for path solutions in mazyin games. International Journal of Artificial Intelligence & Robotics (IJAIR), 3(2), 58-66. doi: 10.25139/ijair.v3i2.4256

Injarapu, A. S. H. H. V., & Gawre, S. K. (2017, October). A survey of autonomous mobile robot path planning approaches. In 2017 International conference on recent innovations in signal processing and embedded systems (RISE) (pp. 624-628). IEEE. doi: 10.1109/RISE.2017.8378228.

Kaur, N. K. S. (2019). A review of various maze solving algorithms based on graph theory. International Journal for Scientific Research & Development (IJSRD), 6(12), 431-434. Erişim adresi: https://www.researchgate.net/publication/331481380

Korf, R. E. (1996). Artificial intelligence search algorithms. In M. J. Atallah, & M. Blanton (Eds.), Algorithms and theory of computation handbook: special topics and techniques (pp. 582-604). Boca Raton, Florida: Chapman & Hall/CRC.

Koubâa, A., Bennaceur, H., Chaari, I., Trigui, S., Ammar, A., Sriti, M. F., & Javed, Y. (2018). Robot path planning and cooperation (1st ed.). Cham, Swiss: Springer International Publishing, (Chapter 1-2).

Mac, T. T., Copot, C., Tran, D. T., & De Keyser, R. (2016). Heuristic approaches in robot path planning: a survey. Robotics and Autonomous Systems, 86, 13-28. doi: 10.1016/j.robot.2016.08.001

Mohanty, P. K., Kundu, S., Srivastava, S., & Dash, R. N. (2020, December). A new TS model based fuzzy logic approach for mobile robots path planning. In 2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE) (pp. 476-480). IEEE. doi: 10.1109/WIECON-ECE52138.2020.9397986

Mukhlif, F., & Saif, A. (2020, February). Comparative study on Bellman-Ford and Dijkstra algorithms, International Conference on Communication, Electrical and Computer Networks (ICCECN). Kuala Lumpur, Malaysia. Erişim adresi: https://www.researchgate.net/publication/340790429

Niu, C., Li, A., Huang, X., & Xu, C. (2021, October). Research on intelligent vehicle path planning method based on improved RRT algorithm. In 2021 Global Reliability and Prognostics and Health Management (PHM-Nanjing) (pp. 1-7). IEEE. doi: 10.1109/PHM-Nanjing52125.2021.9613000.

Pathak, M. J., Patel, R. L., & Rami, S. P. (2018). Comparative analysis of search algorithms. International Journal of Computer Applications, 179(50), 40-43. doi: 10.5120/IJCA2018917358

Patle, B. K., Pandey, A., Parhi, D. R. K., & Jagadeesh, A. (2019). A review: On path planning strategies for navigation of mobile robot. Defence Technology, 15(4), 582-606. doi: 10.1016/j.dt.2019.04.011

Paulino, L., Hannum, C., Varde, A. S., & Conti, C. J. (2021, September). Search methods in motion planning for mobile robots. In K. Arai (Eds.), Intelligent Systems and Applications. Proceedings of the 2021 Intelligent Systems Conference, (pp. 802-822). Cham: Springer. doi: 10.1007/978-3-030-82199-9_54

Permana, S. H., Bintoro, K. Y., Arifitama, B., & Syahputra, A. (2018). Comparative analysis of pathfinding algorithms A*, Dijkstra, and BFS on maze runner game. International Journal Of Information System & Technology, 1(2), 1-8. doi: 10.30645/IJISTECH.V1I2.7

Pradhan, S. K., Parhi, D. R., & Panda, A. K. (2009). Fuzzy logic techniques for navigation of several mobile robots. Applied soft computing, 9(1), 290-304. doi: 10.1016/j.asoc.2008.04.008

Samadi, M., & Othman, M. F. (2013, December). Global path planning for autonomous mobile robot using genetic algorithm. In 2013 International Conference on Signal-Image Technology & Internet-Based Systems (pp. 726-730). IEEE. doi: 10.1109/SITIS.2013.118.

Shamsfakhr, F., & Bigham, B. S. (2017). A neural network approach to navigation of a mobile robot and obstacle avoidance in dynamic and unknown environments. Turkish Journal of Electrical Engineering and Computer Sciences, 25(3), 1629-1642. doi: 1629-1642. 10.3906/elk-1603-75

Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). Planning and Navigation. In R. C. Arkin (Eds.), Introduction to autonomous mobile robots, (pp. 369-423). London, England: The MIT Press.

Sung, I., Choi, B., & Nielsen, P. (2021). On the training of a neural network for online path planning with offline path planning algorithms. International Journal of Information Management, 57, 102142. doi: 10.1016/j.ijinfomgt.2020.102142

Tan, C. S., Mohd-Mokhtar, R., & Arshad, M. R. (2021). A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. IEEE Access, 9, 119310-119342. doi: 10.1109/ACCESS.2021.3108177

Tuncer, A., & Yildirim, M. (2012). Dynamic path planning of mobile robots with improved genetic algorithm. Computers & Electrical Engineering, 38(6), 1564-1572. doi: 10.1016/j.compeleceng.2012.06.016

Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. Soft computing, 22(2), 387-408. doi:10.1007/s00500-016-2474-6

Wang, J., Wu, S., Li, H., & Zou, J. (2018, May). Path planning combining improved rapidly-exploring random trees with dynamic window approach in ROS. In 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA) (pp. 1296-1301). IEEE. doi: 10.1109/ICIEA.2018.8397909

Zhang, H. Y., Lin, W. M., & Chen, A. X. (2018). Path planning for the mobile robot: a review. Symmetry, 10 (10), 450. doi:10.3390/sym10100450

Zhang, L., Zhang, Y., & Li, Y. (2020). Mobile robot path planning based on improved localized particle swarm optimization. IEEE Sensors Journal, 21(5), 6962-6972. doi: 10.1109/JSEN.2020.3039275