



**İstanbul Teknik Üniversitesi Robotik Arama Kurtarma Ekibi**

**Görev Raporu**

**Görev Konusu:** Yol Bulma Algoritmaları ve Sensörler

**Hazırlayan: Ömer Faruk Zeybek**

**Teslim Tarihi:** 04.09.2024

**1. Yol Bulma Algoritmaları ve Sensörler**

Robotik projelerde, robotun hızlı ve güvenli çalışması için en önemli gereksinimlerden ikisi yol bulma algoritmaları ve sensörlerdir. Sensörler, robotların çevresini algılaması herhangi bir tehlikede uyarı yollaması için kritik bir rol üstlenir. Yol bulma algoritmaları ile robotun hedefine güvenli ve en kısa yoldan gitmesi sağlanır aynı zamanda sensörlerden gelen veriler ile bir tehlike durumunda daha güvenli bir rol seçiminde rol oynarlar.

**2. Yol Bulma Algoritmaları**

Robotik projelerde, robotun güvenli bir şekilde hedefine ulaşması için kullanılan yol bulma algoritmalarından bazılarından aşağıda bahsedilmiştir.

**1. Dijkstra Algoritması**

Dijkstra algoritması en kısa mesafeyi hesaplamak için kullanılır. Algoritma düğümleri kullanır ve her düğümü ziyaret ederek başlangıç noktasından hedef noktaya ulaşmaya başlar. Her düğümdeyken komşu düğümleri ziyaret ederek en kısa mesafeyi hesaplar bunu kenarların ağırlıklarını kullanarak gerçekleştirir. Düğüm işaretlendikten sonra onun da komşularına olan mesafe hesaplanır ve hedefe ulaşılana kadar devam eder. Dijkstra algoritması, robotik haritalama ve navigasyon projelerinde sıklıkla kullanılır. Özellikle, statik ve önceden bilinen haritalarda en kısa yolun bulunması gereken senaryolarda kullanılır.

**Avantajları :**

1. **Kesin Sonuç :** Graph negatif kenarlar içermediği sürece algoritma en kısa yolu kesin olarak bulur.
2. **Genel Kullanım :** Hem Directed (Yönlendirilmiş) graphlar da hem de Undirected (Yönlendirilmemiş) graphlarda kullanılabilir.

**Dezavantajları:**

1. **Zaman :** Algoritma tüm düğümleri ziyaret ettiğinden ve her kenarı ölçtüğünden karmaşık ve büyük graphlarda yavaş çalışır.
2. **Negatif Kenar Ağırlıkları :** Düğümler arasındaki kenarların negatif değerleri olabilir. Dijkstra algoritması negatif ağırlıkları desteklemediğinden bazı projelerde kullanılması zor hale gelir.
3. **Heuristik Kullanımı :** Hedef düğüme olan mesafeyi tahmin etmek için Heuristik kullanmadığı için fazla hesaplama yapar.

**2.** **A\* Algoritması**

A\* algoritması, Dijkstra’nın genişletilmiş versiyonu olarak kabul görür. g(n) + h(n) fonksiyonunu kullanır. Heuristik kullanımı (Heuristik fonksiyonlar: Bir düğümden (node) başka bir düğüme olan en kısa yolun maliyetini hesaplayan fonksiyonlardır.) Dijkstra’nın aksine vardır. g(n), başlangıçtan mevcut düğüme kadar olan gerçek mesafeyi temsil ederken, h(n) düğümden hedefe olan tahmini mesafedir. A\* algoritması, dinamik ortamlar ve gerçek zamanlı uygulamalar için uygundur. Haritalama ve robot navigasyonu gibi alanlarda, özellikle engellerin ve belirsiz bölgelerin olduğu durumlarda etkili olur. Heuristik fonksiyonun doğru seçimi, algoritmanın performansını büyük ölçüde etkileyebilir.

**Avantajları :**

1. **Verimlilik :** Heuristik fonksiyonu sayesinde diğer algoritmalara göre daha hızlı bir çözüm sunabilir.
2. **Optimal Çözüm :** Doğru heuristik fonksiyonu kullanıldığında algoritma en kısa yolu bulur ve doğru sonuçlar sağlar.
3. **Dinamik ve Esnek :** Çeşitli Heuristik fonksiyonlar kullanılarak farklı problemlerde kullanılabilir aynı zamanda değişikliklere hızlı adapte olur bu da dinamik ortamlar ve gerçek zamanlı uygulamalar için uygundur özellikle engellerin ve belirsiz bölgelerin olduğu durumlarda etkili olmasına yarar.

**Dezavantajları :**

1. **Heuristik Kullanımı :** Heuristik kullanımı algoritmanın etkinliğini artırmasının yanı sıra yanlış bir heuristik kullanımı performansın düşmesine sebep olabilir.
2. **Hafıza Kullanımı :** Büyük ve karmaşık graphlarda daha fazla düğümü hafızada tutması gerektiğinden dolayı hafıza kullanımı artabilir.
3. **Karmaşıklık :** Dijkstra gibi daha basit algoritmalara göre karmaşık olduğu için kullanımı daha zordur.

**3. Rapidly-exploring Random Tree (RRT) Algoritması**

Rapidly-exploring Random Tree kısaca RRT algoritması, daha karmaşık ve çok boyutlu uzaylar yol bulma için kullanılır. Algoritma, başlangıç noktasından rastgele bir düğüme doğru genişleyen bir ağaç oluşturur. Ağaç, düğümleri birbirine bağlayarak hedefe ulaşmaya çalışır ve engellerden kaçınır. RRT algoritması, özellikle robot kolu planlaması, drone navigasyonu ve otonom araçlarda kullanılır. Rastgelelik, algoritmanın büyük ve karmaşık ortamlarda bile etkili bir şekilde yol bulmasını sağlar.

**Avantajları :**

1. **Yüksek Boyutlu Problemler :** RRT, yüksek boyutlu ve karmaşık uzaylarda etkili bir şekilde çalışabilir.
2. **Karmaşık Engelleri Halletme :** RRT, karmaşık engellerin bulunduğu ortamlarda iyi sonuçlar verir. Rastgele genişlemesi sayesinde engelleri aşma yeteneğine sahiptir.
3. **Hız :** RRT, hedefe ulaşmak için genellikle hızlı bir şekilde genişler.

**Dezavantajları :**

1. **Optimal Sonuç :** Her zaman en kısa yolu bulamaz çünkü rastgele olarak genişler ve elde edilen sonuç en kısa yol olmayabilir ve bu sonuç optimal olmayabilir.
2. **Hedefe Ulaşma :** Genel olarak hızlı bir algoritma olmasına rağmen çok boyutlu ve karmaşık ortamlarda hedefe ulaşması zaman alabilir.
3. **Dinamik Ortamlar :** Dinamik ortamlarda algoritma güncellemeye ihtiyaç duyar ve bu durum performansı düşürür.

**3. Sensörler**

**3.1 Sensörlerin Kullanım Amaçları ve Çalışma Prensipleri**

Robotik projelerde büyük rol oynayan sensörler, çeşitli fiziksel parametreleri algılayarak veriyi toplamak ve bu veriyi analiz etmek için kullanılır. Robotlarda, çevresel izleme, hareket ve yön, mesafe ve konum, görüntüleme gibi çeşitli sensörler kullanılır. Sensör türüne göre, optik sensörler ışık veya lazer ışınlarını kullanarak çevreyi tarar ve mesafe ölçer, ivmeölçer ve jiroskoplar hareket ve dönüşleri algılar, GPS alıcıları ise uydu sinyalleri kullanarak konumu belirler. Bu sensörler, robotların çevrelerini doğru bir şekilde anlamalarını ve etkileşimde bulunmalarını sağlar, böylece görevlerini etkin bir şekilde yerine getirebilirler.

**3.2 Sensörler ve Kullanım Alanları**

**1. LIDAR (Light Detection and Ranging)**

LIDAR’ın kullanım amacı çevredik nesnelerin mesafelerini ölçmek ve 3D haritalar oluşturmak için kullanılır. Genel olarak otonom çalışan nesnelerde kullanılır (Robot,araba vb.).

**Çalışma Şekli :** LIDAR, lazer ışınları gönderir ve bu ışınların çevredeki nesnelerden yansımasını ölçer. Yansıyan ışınların geri dönüş süresi, nesnelerin mesafelerini ve konumlarını belirlemek için kullanılır.

**Popüler Modelleri ve Fiyatları**

1. **RPLidar A1:**

**Özellikler :** Uygun fiyatlıdır ve temel özellikler sunar. Basit, temel seviye projeler için idealdir.

**Fiyat :** Yaklaşık 4.000 - 4.500 lira arasında değişen bir fiyata sahip (En düşük modelini baz alıyorum)

1. **Hokuyo UST-10LX**

**Özellikler :** Yüksek hassasiyet ve geniş görüş açısı sunar. Profesyonel ve endüstriyel uygulamalarda tercih edilir. Fiyatı RPLidar A1’e göre neredeyse 10-12 katıdır.

**Fiyat :** Yaklaşık olarak 40-60 bin arası değişiyor. Türkiyede az bulunan bir sensör.

**Kendi İçinde Karşılaştırma :** RPLidar A1, Hokuyo’ya göre çok çok daha ucuzdur ve basit projlerler uğraşanlar için idealdir fakat daha profesyonel veya endüstriyel çalışanlar için Hokuyo daha iyi performans gösterir, RPLidar A1 yetersiz kalabilir.

**2. IMU (Inertial Measurement Unit)**

IMU sensörleri jiroskoplar, ivmeölçerler, manyetometreler ve basınç sensörü kombinasyonları içerir.

Genel olarak robotlarda, dronelarda ve telefonlar gibi akıllı cihazlarda kullanılır.

**Çalışma Şekli :** İvmeölçer, lineer hız değişimlerini ölçerken, jiroskop, dönüş hareketlerini algılar. Manyetometre, manyetik alan yoğunluğunu ölçerken basınç sensörleri ise uygulanan kuvveti ölçerler.

**Popüler Modelleri ve Fiyatları :**

1. **MPU-6050**

**Özellikler :** Hobi olarak yapılan robotlarda, dronelarda kullanılabilecek bir sensör modelidir. Hem 3 eksenli ivmeölçer hem de 3 eksenli jiroskop içerir. Uygun maliyetli ve temel hareket algılama işlevleri sunar.

**Fiyat :** Yaklaşık olarak 100 - 150 liradır.

1. **LSM9DS1**

**Özellikler :** Yüksek hassasiyetli hareket izleme, yön belirleme ve gelişmiş robot projeleri için uygundur. Otonom araçlar ve sensör fusion uygulamaları için de tercih edilir.

**Fiyat :** Yaklaşık olarak fiyatı 1000 liradır.

1. **ADIS16488**

**Özellikler :**  Endüstriyel uygulamalar, yüksek performanslı araçlar ve hassas yön algılama gerektiren projeler için uygundur. Yüksek hassasiyetli ve düşük gürültü sunar.

**Fiyat :** 3.820 dolar olarak gözüküyor yaklaşık olarak 100 bin lira yapıyor.

**Kendi İçinde Karşılaştırma :** Hobi olarak bir proje geliştiriliyorsa MPU-6050 kullanımı en uygundur, biraz daha kompleks projelerde LSM9DS1 kullanımı daha uygundur. ADIS 16488 ise endüstriyel daha üst düzey projelerde yer alabilecek bir sensördür.

**3. GPS (Global Positioning System) Alıcıları**

GPS alıcıları, dünya üzerindeki kesin konumu belirlemek ve rota planlaması yapmak için kullanılır. Otonom araçlar, drone’lar ve navigasyon sistemlerinde yer belirleme işlevi görür.

**Çalışma Şekli :** GPS alıcıları, uydu sinyallerini kullanarak konum bilgisi belirler. Uydu sinyallerinin alınma süresi hesaplanarak cihazın yeryüzündeki kesin konumu belirlenir.

**Popüler Modeller ve Fiyatları:**

1. **UBlox NEO-6M**

**Özellikler :** Genel konum belirleme işlemleri için uygun ve yeterlidir.

**Fiyat :** 150 liraya kadar bulunabilir.

1. **ZED-F9P**

**Özellikler :** Yüksek hassasiyet ve RTK (Real-Time Kinematic) desteği sağlar. Profesyonel ve hassas konumlandırma gereksinimleri için uygundur.

**Fiyat :** 128 dolar yaklaşık olarak 4.000 liraya kadar bulunabilir.

**Kendi içinde karşılaştırma :** UBLOX, daha uygun fiyatlıdır, ancak hassasiyet açısından RTK sistemlerinin gerisindedir. Genel konum belirleme ihtiyaçları için uygundur fakat daha hassas ve kesin konum isteniyorsa RTK özelliği olan ZED-F9P daha iyidir.

**4. Stereo Kameralar**

Stereo kameralar, çevrenin 3D haritasını oluşturmak ve derinlik algılamak için kullanılır. Robotlarda nesne tanıma ve çevre analizi için yaygın olarak tercih edilir.

**Çalışma Şekli :** Stereo kameralar, iki farklı açıdan görüntü alarak derinlik bilgisi sağlar. İki görüntü arasındaki farklar, çevredeki nesnelerin mesafelerini ve konumlarını belirlemek için kullanılır.

**Popüler Modeller ve Fiyatları :**

1. **Intel RealSense D435**

**Özellikler : Y**üksek performans ve geniş uygulama çeşitliliği sunar. Orta ve yüksek hassasiyet gerektiren projeler için uygundur.

**Fiyat :** 18.000 - 22.000 lira arası fiyatları var.

1. **ZED Stereo Camera**

**Özellikler :** Yüksek performans ve derinlik algılama özellikleri sunar. Profesyonel ve gelişmiş uygulamalar için tercih edilir.

**Fiyat :** 30.000 lira ve 60.000 lira olanları var.

**Kendi İçinde Karşılaştırma :** İki kamerada yüksek performans sunar. Çok ileri düzey bir proje veya uygulama değilse Intel Realsense D435 daha mantıklıdır.

**Özet**

Robotik sistemlerde, yol bulma algoritmaları ve sensörler, robotların çevresini algılayarak güvenli ve etkili bir şekilde hareket etmelerini sağlar. Dijkstra, A\* ve RRT gibi algoritmalar, robotların en uygun rotayı bulmasına yardımcı olurken; LIDAR, IMU, GPS ve stereo kameralar gibi sensörler, çevresel verileri toplar ve bu algoritmalara girdi sağlar. Bu teknolojiler, robotların karmaşık ve dinamik ortamlarda bağımsız hareket etmesine olanak tanır, böylece otonom sistemlerin performansını ve güvenilirliğini artırır.

**Kaynakça**

**Genel Kaynaklar**

**LaValle, S. M. (1998). *Rapidly-exploring random trees: A new tool for path planning*. [PDF]. University of Illinois at Urbana-Champaign.** [**https://msl.cs.uiuc.edu/~lavalle/papers/Lav98c.pdf**](https://msl.cs.uiuc.edu/~lavalle/papers/Lav98c.pdf)

**GeeksforGeeks. (n.d.). *A* Search Algorithm. Retrieved September 2, 2024, from** [**https://www.geeksforgeeks.org/a-search-algorithm/**](https://www.geeksforgeeks.org/a-search-algorithm/)

**W3Schools. (n.d.). *Dijkstra's Algorithm*. Retrieved September 2, 2024, from** [**https://www.w3schools.com/dsa/dsa\_algo\_graphs\_dijkstra.php#:~:text=Dijkstra's%20algorithm%20finds%20the%20shortest,all%20the%20unvisited%20neighboring%20vertices**](https://www.w3schools.com/dsa/dsa_algo_graphs_dijkstra.php#:~:text=Dijkstra's%20algorithm%20finds%20the%20shortest,all%20the%20unvisited%20neighboring%20vertices)**.**

**Programiz. (n.d.). *Dijkstra's Algorithm*. Retrieved September 2, 2024, from** [**https://www.programiz.com/dsa/dijkstra-algorithm**](https://www.programiz.com/dsa/dijkstra-algorithm)

**Github Repositoryleri**

[**https://github.com/TheAlgorithms/Python/blob/master/graphs/dijkstra.py**](https://github.com/TheAlgorithms/Python/blob/master/graphs/dijkstra.py)

[**https://github.com/TheAlgorithms/Python/blob/master/graphs/a\_star.py**](https://github.com/TheAlgorithms/Python/blob/master/graphs/a_star.py)

[**https://github.com/zhm-real/PathPlanning?tab=readme-ov-file**](https://github.com/zhm-real/PathPlanning?tab=readme-ov-file)