DİNAMİK ENGELLERİN BULUNDUĞU ORTAMDA GEZGİN ROBOT İÇİN HAREKET PLANLAMA

2016
YÜKSEK LİSANS TEZİ
ELEKTRİK - ELEKTRONİK MÜHENDİSLİĞİ

Mehmet LAFCI

DİNAMİK ENGELLERİN BULUNDUĞU ORTAMDA GEZGİN ROBOT İÇİN HAREKET PLANLAMA

Mehmet LAFCI

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalında
Yüksek Lisans Tezi
Olarak Hazırlanmıştır

KARABÜK
Ocak 2016

Mehmet LAFCI tarafından hazırlanan "DİNAMİK ENGELLERİN BULUNDUĞU ORTAMDA GEZGİN ROBOT İÇİN HAREKET PLANLAMA" başlıklı bu projenin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Prof. Dr. İhsan ULUER
Tez Danışmanı, Elektrik Elektronik Mühendisliği Anabilim Dalı .....

Bu çalışma, jürimiz tarafından oy birliği ile Elektrik Elektronik Mühendisliği Anabilim Dalında Yüksek Lisans Tezi olarak kabul edilmiştir. 20/01/2016

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Prof. Dr. İhsan ULUER (KBÜ)	.....
Üye : Prof. Dr. Şerafettin EREL (YBÜ)	.....
Üye : Yrd. Doç. Dr. Mustafa B. TÜRKÖZ (KBÜ)	.....

...../...../2016

KBÜ Fen Bilimleri Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Prof. Dr. Nevin AYTEMİZ
Fen Bilimleri Enstitüsü Müdürü .....

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Mehmet LAFCI

ÖZET

Yüksek Lisans Tezi

DİNAMİK ENGELLERİN BULUNDUĞU ORTAMDA GEZGİN ROBOT İÇİN HAREKET PLANLAMA

Mehmet LAFCI

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik - Elektronik Mühendisliği Anabilim Dalı

Tez Danışmanı:
Prof. Dr. İhsan ULUER
Ocak 2016, 64 sayfa

Sabit ve hareketli engellerin bulunduğu ortamda, bir robotun engellere çarpmadan istenilen hedef bölgeye ilerleyebilmesi, robotik teknolojilerin birçok alanda kullanılmaya başlandığı günümüzde yoğun çalışılan bir konudur. İnsanların giremediği bölgelerde, robotların rahatça dolaşarak herhangi bir görevi yerine getirmesi, o bölgede engellere çarpmadan istenilen yerlere gidebilmesi ile mümkün olacaktır.

Robotun hareket ettiği ortam sabit engelli ve sabit bir bölge olabileceği gibi hızlı değişen dinamik engelli dinamik bir bölge de olabilir. Gezgin robotlar engellerden kaçınarak hedefine giderken optimum olmayan yolları izleyebilir.

Hareket planlama algoritmaları; sabit veya hareketli engellerin bulunduğu belirli bir arazide, gezgin robotların engellere ve birbirlerine çarpmadan otonom hareket

etmelerini amaçlamaktadır. Bu kapsamda; algoritma çalışmalarını gerçekleştirebilmek için, pratik alanda yapılacak test maliyetlerini azaltmak üzere, bir simülasyon ortamında çalışmalar yapılmaktadır.

Bu tezde uygulanacak olan "A* Arama Algoritması"; en kısa yol bulmak için kullanılan algoritmalardan birisidir. Robot bilinen bir ortamda hedef olarak gösterilen noktaya optimum yoldan giderken karşısına çıkacak dinamik engellerden kaçınarak yoluna devam etmesi amaçlanmaktadır.

Bu çalışmada birden fazla robotun sabit ya da hareketli engellerin bulunduğu bilinen bir ortamda engellere çarpmadan otonom olarak hareketinin bir simülasyon ortamında incelenmesi sağlanacaktır. En uygun olan yolu bulma ve izleme, farklı senaryolar kullanılarak engelden kaçınma, hız kontrol algoritmaları ve bu çalışmanın simülasyon ortamında performansının gözlemlenmesi amaçlanmıştır. Simülasyon ortamında yapılan çalışmaların sağlayacağı faydaların değerlendirilmesi sonucunda Gazebo robot simülatörü kullanılması tercih edilmiştir.

Anahtar Sözcükler : Mobil robot, Gazebo simülatörü, engel aşma, lazer sensor, çoklu robot, otonom hareket, yol bulma algoritmaları, A* yol bulma algoritması.

Bilim Kodu **:** 905.1.014

ABSTRACT

M. Sc. Thesis

MOTION PLANNING FOR MOBIL ROBOT IN ENVIROMENTS WHERE DYNAMIC OBSTACLES

Mehmet LAFCI

Karabük University
Graduate School of Natural and Applied Sciences
Department of Electric - Electronic Engineering

Thesis Advisor:
Prof. Dr. İhsan ULUER
January 2016, 64 pages

Nowadays is widely studied topic which robots can progress to the desired target point without hitting the obstacles in the presence of fixed and moving obstacles, which started to be used in many fields of robotics technology. Because robots circulate freely fulfill any task in the regions where people can not enter, they will be able to the desired location without hitting the obstacles in the area that.

Environment in which the robot moves can be fixed obstacle and fixed a region as well as can be rapidly changed dynamic obstacle a region. Mobile robots can follow non-optimal paths to the destination by avoiding obstacles.

Motion planning algorithms aim at that mobile robots can autonomously move without hitting obstacles and each other a certain region which has fixed or moving obstacles. In this context, To perform the algorithm works, to reduce the costs of

testing to be done on the practice field, are being studied in a simulation environment.

To be implemented in this thesis, "A * Search Algorithm" is one of algorithms used to find the shortest path. While the robot goes to the optimal way to towards the target point, it is intended to continue the path avoiding dynamic obstacles.

This study shall be examined that the robot autonomously move without hitting obstacles in a simulation environment which has multiple fixed or moving obstacles. Finding the most appropriate way and tracking, obstacle avoidance using different scenarios and speed control algorithms, and is intended to observe the performance of the simulation environment of this study. Benefits of the studies in the simulation environment are evaluated, and is preferred to use the gazebo robot simulator.

Key Word : Mobile robot, simülatör of Gazebo, overcoming obstacles, laser sensor, multi-robot autonomous movement, path finding algorithms, A * pathfinding algorithm.

Science Code : 905.1.014

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Prof. Dr. İhsan ULUER ‘e sonsuz teşekkürlerimi sunarım.

Sevgili eşime manevi hiçbir yardımını esirgemeden yanımda olduğu için tüm kalbimle teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

KISALTMALAR

RPP : Robot Path Planning (Robot Yol Planlama)

LRTA* : Learning Real Time A star (Öğrenmeye Dayalı Gerçek Zamanlı A Yıldız)

LSS-LRTA* : Local Search Space Learning Real Time A* (Yerel Arama Uzayı Öğrenmeye Dayalı Gerçek Zamanlı A*)

RRT : Rapidly-Exploring Random Tree (Hızlıca Rastgele Keşfetme Ağacı)

MA-RRT : Metric Adaptive Rapidly-Exploring Random Tree (Metrik Uyarlamalı Hızlıca Rastgele Keşfetme Ağacı)

IAA* : Iterative Adaptive A star (Tekrarlamalı Uyarlamaya Dayalı A*)

RTAA* : Real Time Accelerated A* (Gerçek Zamanlı Hızlandırılmış A*)

ARA* : Anytime Repaired A* (Herhangi Zamanlı Onarılmış A*)

CPU : Central Processing Unit (Merkezi İşlem Birimi)

BÖLÜM 1

GİRİŞ

Karmaşık durumlarda robot kontrolünün, doğruluk ve çabukluk (mesafede ve zamanda) açısından amaca ulaşması için çalışmalar yapılmaktadır. Bu problemin zorluğu robotun daha önce çevresi hakkında hiç bir ön bilgisinin olmamasıdır. Bu ön bilgiden yoksun robotun engellere çarpmadan hedefine gidebilmesi için bazı yöntemler geliştirilmesi gerekmektedir. Diğer yandan hareketli robotun yol bulma ve izleme sisteminin karmaşıklığı sistemi yüksek maliyetli ve riskli yapabilir. Dolayısıyla bilinmeyen engellerle donatılmış ortamlarda robot navigasyon/yol planlama sistem tasarımı önem kazanır. Bu tür sistemler için en sık kullanılan kontrol metotları robotun hareketlerini sensör bilgisi ile ilintilendirir.

A* algoritması başlangıç düğümünden bitiş düğümüne olan en az maliyetli yolu bulmada kullanılır. Bu yöntemde robotun bulunduğu ortamın bir harita üzerinde bilindiği kabul edilmektedir. O anki konumundan hedef olarak gitmesi istenilen bir noktaya ilerlemesi için izleyeceği en kısa yolu bulmak için bu algoritma ile yol belirlenebilmektedir. Fakat harita üzerinde görünmeyen engellerden kaçınma noktasında bu algoritma yetersiz kalmaktadır. Bunların tespiti için de robot üzerine yerleştirilmiş bir lazer sensörden yararlanılacaktır.

Robot platformu üzerine yerleştirilen bir lazer sensör yardımı ile, çevredeki nesnelerin robota olan uzaklık bilgileri elde edilmektedir. Bu bilgi kullanılarak, robotun karşısına çıkan bir engele çarpmadan yoluna devam etmesini sağlamak mümkün olmaktadır.

Lazer sensör ile belirli bir mesafenin altındaki bir yerde engel tespit edilmesi durumunda, robotun yönünü değiştirmesi gerekmektedir. Robotun hızı, yönü ve engelin robota karşı konumuna göre robotun vermesi gereken tepkiler değişmektedir.

Robotun hızlanma süresi uygulanan ivme ile orantılı olmaktadır. Robotun ağırlığına göre uygulanacak maksimum ivme durumu değişmektedir. Engelden kaçınma algoritması geliştirirken robot üzerindeki hız kontrolü önemli bir etken olmaktadır.

1.1. HAREKET PLANLAMA ALGORİTMALARI

Hareket planlama algoritmaları; sabit veya hareketli engellerin bulunduğu belirli bir arazide, gezgin robotların engellere ve birbirlerine çarpmadan otonom hareket etmelerini amaçlamaktadır. Bu kapsamda; algoritma çalışmalarını gerçekleştirebilmek için, pratik alanda yapılacak test maliyetlerini azaltmak üzere, bir simülasyon ortamında çalışmalar yapılmaktadır.

Hareket Planlamanın Önemi:

i. Robotların belirlenen amaçlar doğrultusunda, etkin biçimde kullanılabilmeleri.
ii. İnsan gücünden tasarruf etme, oluşabilecek hatalardan korunma.
iii. Gezgin robotlar engellerden kaçınarak hedefine giderken optimum olmayan yolları izleyebilir. Birçok araştırmacı yol planlama probleminin çözümünü global ve lokal olarak tanımlamakta ve değişik kontrol metotlarıyla çözüm önermektedir.
iv. Robotun, karmaşık çevre hakkında bir ön bilgisi olmaması ve var olan klasik algoritmaların, robotun doğru ve hızlı bir şekilde kontrolü açısından yetersiz kalması.
v. Yol planlama sisteminin karmaşık, riskli ve oldukça yüksek maliyetli olma sorunlarıyla karşı karşıya olması.
vi. Bahsi geçen tüm bu olumlu ve olumsuz durumlardan dolayı, her ne kadar üzerinde çalışmaların süregeldiği bir alan olsa da; hareket planlama sistemleri ile ilgili çalışmalara halen ihtiyaç duyulmaktadır.

Robotikte Sezgisel robotik, çoklu robot sistemleri, robot yol planlama gibi değişik araştırma alanları vardır. Robotikte başlıca alanlardan birisi olan robot yol planlama (robot path planning) şimdiye kadar çoğu araştırmacı tarafından çalışılmıştır. Çünkü

bu problem otonomi, robotik cerrahi ve otomasyon gibi değişik robotik uygulamalarında uygulanmıştır. RPP bir görevi yapabilmesi için robotun davranışını planlamaktadır. Bu nedenle eğer robotun mevcut durumu başlangıç noktası olarak ve görevini bitirdikten sonraki durumu hedeflenen nokta olarak düşünülürse, RPP olası bir yol bulma ve bu yol boyunca robotun hedeflenen noktaya ulaşabileceği üzerine plan yapmalıdır.

RPP statik ortamlar, hareketli engellerin olduğu statik ortamlar, statik ve dinamik engellerin olduğu dinamik ortamlar gibi farklı ortamlarda çalışılmıştır. Bu ortamlar içerisinde diğer ortamlar ile karşılaştırıldığında en zor arama problemi dinamik engellerin olduğu yol olarak bilinmektedir.

Otonom robotlar dinamik ortamlarda, dinamik ve statik engeller arasında güvenli bir şekilde hareket etmelidir. Dinamik ortamlarda önerilen RPP için iki tane zorlu problem vardır. Bunlar yerel minimum izleme ve gerçek zaman performansını sürdürmedir [1,2].

Yerel minimum problemi hatalı sezgisel değerlere sahip durum uzayındaki bazı bölgelerle ilgilidir. Bu hatalı sezgisel değerler umut verici görünür ve algoritmalar bu değerlerin bulunduğu bölgelerden hedefe doğru aramasını sürdürür. Bu nedenle bu bölgeler içindeki izleme (trapping) arama zamanını artırır. Bunlara ek olarak, heyecan verici metotların çoğu gerçek zaman performansını garanti edemezken ortalama çözüm maliyetini artırmaktadır. Bunun anlamı robot ortam değişikliklerini doğru bir şekilde izleyemediğinden en yakın optimum yolu bulma maliyeti artmaktadır.

Dinamik ortamda robot hareket planlaması hakkında genel bir anket yapan Yebenes vd. problemi çözmek için uygulanan değişik yaklaşımları tekrar gözden geçirmişlerdir [3]. Keshmiri and Payandeh diferansiyel kısıtlamaların olduğu ve olmadığı yörünge planlama çözümlerinde uygulanmış değişik yaklaşımları karşılaştırmaktadır [4]. Tang vd. ve Goerzen vd. tarafından yapılan çalışmalar robot hareket planlaması ve yol planlama kapsamı alanında ki yaklaşımları sınıflandıran

bir başka arama incelemeleridir [5,6]. Bu yaklaşımların hiçbiri yukarıdaki zorluklara özel bir şekilde ilgilenmemekte ve ilgili yaklaşımları gözden geçirmemektedir.

1.1.1. Yol Planlama Problemi Üzerine Uygulanmış Algoritmalar

Dinamik ortamlarda yol planlama problemini çözmek için değişik yaklaşımlar kullanılmıştır. Temel olarak bu yaklaşım teknikleri ve algoritmaları 3 kategoride sınıflandırılmaktadır, bunlar; sezgisel metotlar (heuristic methods), rastgele örnekleme metotları (randomized sampling methods) ve reaktif metotlardır (reactive methods).

1.1.1.1. Sezgisel (Heuristic) Metotlar

Yol bulma için basit bir fikir standart maliyetli temel bir arama algoritması uygulamaktır. Bu arama metodu tam ve optimum olmasına rağmen çözüm yoluna gerçekten yakın olmayan bir çok düğümlere genişler. Bir sezgisel fonksiyon, arama uzayında herhangi bir düğümden hedefe doğru en ucuz maliyetli yolu tahmin edebilmek için alternatif bir metottur. Sezgisel fonksiyon arama problemi kısıtlamalarına göre tasarlanmıştır. Buna ek olarak, sezgisel fonksiyon standart maliyetli arama tarafından ziyaret edilmiş düğümlerin çoğuna genişlemeden hedefe doğru aramaya yol gösterir. Bu yüzden sezgisel fonksiyon arama zamanını azaltabilmektedir ve gerçek zamanlı bir arama algoritması olarak kullanılabilmektedir. Sezgisel metotlar 3 sınıfa ayrılmaktadır. Bunlar; sezgisel, çevrimiçi ve çevrimdışıdır. Çevrimiçi metotlar, gerçek zamanlı sezgisel arama ve herhangi bir zamanlı algoritmalar olarak sınıflandırılır. Aşağıda dinamik ortamlarda RPP çözümü için bu algoritmaların avantajlarını ve dezavantajlarını gözden geçireceğiz.

Sezgisel

Basit doğrusal mesafeli sezgisel (simple straight-line distance heuristic) ve statik iki boyutlu Dijkstra sezgisel (static 2D Dijkstra heuristic), robot yol planlamada kullanılan iki sezgisel fonksiyonlardır. Bu metotlar herhangi bir noktadan hedefe

olan mesafeyi tahmin etmelerine rağmen onlar statik engellerden dolayı görülen yerel minimumdan sıkıntı çekmektedir. Yebenes tarafından önerilen A* algoritması sezgisel arama algoritmasının en iyi bilinenidir [3]. Bu algoritmada açık liste ve kapalı liste olmak üzere iki tane liste vardır. Açık liste A* tarafından üretilen ve aranan bütün durumları düzenler ve depolar. Açık liste bir öncelikli dizi listesi (ortalama kümesi) tarafından oluşturulmaktadır. N bir düğüm olmak üzere, A* için kullanılan sezgisel fonksiyon f(n) aşağıdaki gibi tanımlanır:

$$f(n) = g(n) + h(n) \quad (1.1)$$

Burada g(n) başlangıç düğümünden n düğümüne hareket etmek için gereken maliyeti hesaplarken, h(n) n düğümünden hedefe varmak için gerekli olan maliyeti hesaplamaktadır. Her durum açık listeye eklenerek, bu liste f değerleri tarafından doldurulmuş kayıtlarını artırmalı bir şekilde tekrar sıralamaktadır. Bu yüzden listenin en yukarısı f değerlerinin en düşüğü olarak tanımlanır. A* aramayı bitirdikten sonra açık listenin en yukarısında olan durumu çıkarıp o durumu kapalı listede depolamaktadır. Dolayısıyla kapalı liste A* tarafından seçilmiş durumları içermektedir. Bu algoritma robotun gerçek zaman performansını sürdürememekte ve yerel minimum problemine takılmaktadır. A* 'ın bir versiyonu olan ağırlıklı A*, aşağıda ki gibi ağırlaştırılmış bir sezgisel fonksiyon kullanır:

$$f(n) = g(n) + w.h(n) \quad (1.2)$$

Ağırlıklı A* arama zamanını azaltmasına rağmen dinamik ortamlarda A* ile aynı problemlere sahiptir.

Artırmalı Sezgisel Arama

Artırmalı sezgisel arama algoritması mevcut hücresinden hedefe doğru varsayılan engellenmemiş yolu bulur. Bu arama robot hedefe ulaşıncaya kadar ya da engelle karşılaşıncaya kadar yinelemeli olarak yapılır [5]. Yol planlamada kullanılan artırmalı algoritmalar 3 grupta sınıflandırılır [6]. Birinci grup algoritmalar A-star' ın artırmalı versiyonlarıdır. FSA* bu sınıfta önemli algoritmalardan birisidir [7]. Bu

metodun işleyişi A-star kullanarak başlangıç düğümünden hedefe en kısa bir yol bulmaktır. Eğer bazı aşamalar statik engel olarak işaretlenmişse, bu algoritma başlangıç ve hedef arasındaki bir başka kısa yolu bulmak için, mevcut A* arama eğrisine benzer olan önce gelen A* arama eğrisini tekrar kullanır.

İkinci grup önceki aramadan düğümlerin h değerlerini öğrenir ve daha iyi bir tahmin oluşturur. Bu arama algoritması h değerlerini artırarak yerel minimumdan kurtulmaktadır. Uyarlamalı A* ikinci grupta yer alan artırmalı bir algoritmadır [8]. Üçüncü grup önceki aramaya dayalı olarak g değerlerini değiştirir. Diferansiyel A* [9], odaklanmış dinamik (D*) [10] ve D* lite [11] den oluşan bu grup diğer gruplarla kıyaslandığında daha verimlidir[6]. Bu grupta D* ve D* lite diferansiyel A*' dan daha gelişmiştir. Marsa araç göndermek için kullanılan uzaktan kontrollü cihazlarda ve taktiksel gezgin robot projelerinde gezgin robotlar için yol planlamada D* kullanılmaktadır [12,13]. Bu algoritmalar yerel minimum problemine karşı zayıf performansa sahiptir ve onlar dinamik ortamlarda gerçek zaman performansını garanti etmezler.

Çevrimiçi Metotlar

Gerçek zamanlı arama algoritmaları onların çevreleri ile sürekli bir haberleşmeye sahiptir ve çevrelerindeki değişiklikleri izleyebildiğinden dolayı robot başlangıçta çevresi hakkında kesin bilgiye sahip olmadığında arama problemini çözmek için uygun metotlardır. Yol planlama problemini çözmek için değişik sayıda gerçek zamanlı algoritmalar önerilmiştir [5, 14-20].

Gerçek zamanlı arama algoritmalarında planlama aşaması ve eylem aşaması olmak üzere iki tane aşama vardır. Planlama aşamasında robot bir ya da birkaç sayıda olası eylemleri seçer (seçme adımı) ve seçilen durumların sezgisel değerlerini günceller (öğrenme adımı) ve daha sonra eylem aşamasında onları yerine getirir. Bu süreç robot hedefe ulaşıncaya kadar devamlı olarak tekrar eder. Eylem aşamasında robot ortamı izleyebilmekte ve bu ortam hakkında bilgilerini güncelleyebilmektedir.

Gerçek Zamanlı Sezgisel Arama Algoritmaları

İlk gerçek zamanlı algoritma learning real-time A*'dır (LRTA*) [19]. LRTA* 'nın birkaç tane değiştirilmiş versiyonları tanıtılmıştır. Planlama aşamasında, LRTA* başlangıçta robotun mevcut pozisyonunun mirasçılarını üretir ve A* algoritmasını kullanarak bütün mirasçılar arasından en iyi mirasçıları seçer. Öğrenme aşamasında bu algoritma seçilen mirasçıların sezgisel değerlerini günceller. Bu metot çevrimdışı metotlardan daha hızlı optimal yola yakın bir yol bulabilir. LRTA* 'ın işleyişi dinamik ortamlarda çok zayıftır ve yüksek çözüm maliyetine sahiptir. LRTA*(k)' nın yapısı LRTA*'a benzer olmasına rağmen bu algoritma sezgisel değerleri güncellemek için LRTA*' dan daha farklı bir yaklaşım kullanır. Daha fazla öğrenme gerçekleştirerek, LRTA*(k) yerel minimumdan kurtulabilir. Bu algoritma dinamik ortamlarda çok zayıf performansa sahiptir.

Hebert vd. tarafından önerilen $LRTA_{ls}$ (k, d), LRTA*(k)' nın değişik bir versiyonudur [18]. Bu algoritmanın öğrenme işleyişi, ilerisi için planlama yapmayı artırarak ve her bir planlama aşamasına daha fazla öğrenmeyi ekleyerek iyileştirilmiştir. Bu metot yüksek hızda hareket planlayabilir fakat gerçek zaman performansını garanti etmez.

Yerel arama uzayı öğrenme gerçek zamanlı A* (local search space learning real time A* (LSS-LRTA*)) [6] en son gelişmeleri yansıtan (state-of-the-art algorithm) algoritma olarak bilinir. Seçim aşamasında, robot durumların bir sınır değerini seçer ve düğümlerin bu seti yerel arama uzayı olarak adlandırılır. Durumların seçilmesinden sonra ve onları kapalı listede depoladıktan sonra LSS-LRTA* açık listede depolanmış olan durumlardan bir yerel hedef seçer. En düşük f değerli durum yerel hedef için algoritmanın adayıdır. Öğrenme aşamasında, algoritma kapalı listenin sezgisel değerlerini değiştirmek için Dijkstra algoritmasını kullanır. Bu öğrenme mekanizmasını kullanarak LSS-LRTA* yerel minimumdan kurtulabilmektedir. Bu algoritmanın sakıncası gerçek zaman performansını garanti edemez. RTD* dinamik ortamlar için en son gelişmeleri yansıtan algoritmadır [21]. Bu metot D* Lite [11] ve LSS-LRTA* [6]' ı birleştirmektedir. Bu metotta planlama aşaması, 1) D* Lite ve 2) LSS-LRTA* olarak ikiye bölünmüştür. İlk aşamada, RTD*

hedef durumdan robotun mevcut durumuna geri gelir ve tam bir yol bulabilirse eğer optimal yola geri döner aksi takdirde planlama zamanından geriye kalan vakitte uygun bir eylem bulmak için LSS-LRTA* çalışır. Bu algoritmanın sakıncası D* Lite' ye geri dönmesidir çünkü bu algoritma yüksek hızlı eylemleri planlayamaz ve yüksek boyutlu durum uzaylarında kolay bir şekilde başarısız olmaktadır.

PLRTA*, LSS-LRTA*' ın yeni bir versiyonudur [38]. Bu metot statik ve dinamik ortamlar için maliyetleri ayırarak onları ayrı ayrı öğrenmeyi denemektedir. Bu algoritma dinamik ortamlarda diğer gerçek zamanlı sezgisel arama algoritmaları ile kıyaslandığında daha iyi performansa sahiptir. Ek olarak bu algoritma gerçek zaman performansını garanti etmektedir.

Yerel minimum izleme daha önce ifade edildiği gibi bu alanda önemli zorluklardan birisidir. Bu problem robotlar sezgisel depresyon bölgesinde tuzağa düştüğünde meydana gelir. Sezgisel depresyon bölgeleri, onların durumlarının sezgisel değerlerini birkaç kez güncellemeye ihtiyaç duyabildiğinden dolayı gerçek zamanlı sezgisel fonksiyonlar kolay bir şekilde tuzaklanmış olan ve sezgisel fonksiyonun doğru olmadığı arama uzayının sınırlı alanları olarak adlandırılır [23].

Ishida tarafından önerilmiş bir sezgisel depresyon bölgesini erken tanıma, bölgenin sınırı üzerindeki durumlardan hangisinin sezgisel değerinin bölge içindeki durumların sezgisel değerine eşit ya da büyük olduğunu gösteren sınırlı bir bölgedir [1]. Sturtevant vd. tarafından değiştirilen bu tanım gerçek zamanlı arama algoritmaları için iki yeni teknik sunmaktadır [23]. Bu teknikler İşaretle ve Korun (Mark and Avoid) ve sınıra taşımaktır (move to border). Onlar LSS-LRTA* ve RTAA* üzerine bu teknikleri uyguladı ve 4 yeni değiştirilmiş versiyon sundular. Bunlar içinde en iyisi daLSS-LRTS* olarak adlandırılmaktadır. Sezgisel depresyon bölgeleri ile karşılaştığında bu algoritmalar sezgisel depresyon bölgesi içindeki durumları işaretlemekte ve daha sonra yaklaşımlarla onlardan korunmaktadır. Onların sonuçlarına dayalı olarak sınıra taşıma daha iyi bir tekniktir ve bu teknik ya sınıra taşımak için ya da sınıra daha yakın olan durumlara taşımak için aramaya rehberlik eder. Bu yeni versiyonlar diğer gerçek zamanlı arama algoritmaları ile kıyaslandığında algoritmanın performansı yerel minimumdan korunduğu için

artmasına rağmen bu yeni versiyonlar orjinali gibi gerçek zaman performansını garanti etmez.

Her Hangi Bir Zamanlı Algoritmalar

Herhangi bir zamanlı algoritmalar zaman ilerliyormuş gibi onların çözümlerini daha iyi yapmayı denemektedirler. Algoritmaların bu türü herhangi bir zaman da herhangi bir kesmeden önce bile kısmi bir çözüm döndürebilmektedir. Bu algoritmalar tarafından üretilen cevapların kalitesi hesaplama zamanı ile doğrudan bir ilişkiye sahiptir. Şöyle ki herhangi bir zamanlı algoritma tarafından üretilen çözüm ya optimuma yakındır ya da en iyi çözümdür ve çözümün kalitesi bir hesaplama zamanı ekleyerek artmaktadır. Korf herhangi bir zamanlı algoritmayı tanıtmaktadır [2]. Bu metot zamana bağlı problemleri çözmek için kullanışlıdır. Onlar Hernandez and Meseguer tarafından önerilen herhangi bir zamanlı A* onarımını(ARA*) kullanarak büyük bir başarıya ulaştılar [21]. Herhangi zamanlı algoritmanın uygulamalarından birisi robot hareket planlamadır ve ARA* önemli yol planlama algoritmalarından bir tanesidir.

Çevrimdışı Algoritmalar

Çevrimdışı algoritmalar her zaman adımında başlangıç pozisyonundan hedefe olan bütün yolları üretir [25]. Bir başka deyişle, her zaman adımında bir eylem üreten diğer metotların aksine bu algoritma türü herhangi bir zaman adımında başlangıçtan hedefe tam bir yol üretebilme yeteneğine sahiptir.

Kushleyev and Likhachev dinamik ortamlar için yeni bir metot önermiştir. Şöyle ki bu metotta planlama aşaması gürültülü bir ortamda hareket eden nesnelerin konumunu tahmin etmek zorundadır. Aynı zamanda, planlamanın hesaplanması durum uzayının bir değişkeni olarak eklenmiş olan zamandan dolayı pahalıdır. Bu yüzden hareket eden nesnenin yörüngesini tahmin etmek için tekrar planlama bir ihtiyaçtır. Bu metotta zaman planlaması iki zaman adımına bölünmektedir. İlk zaman adımında her durum 5 değişken (x,y,Q,v,t) tarafından temsil edilmektedir. Algoritma bu adımda dinamik engellerin hareketini tahminin güvenilirliği olan T_b^{max} değerine

kadar tahmin eder. Bu makalenin yazarı, eğer onlar çarpışma olmadan kabul edilebilir bir davranış kazanmak istiyorsa onların T_b^{max} sınırını koymak zorunda olduklarını vurgulamaktadır. Zaman planlaması için hızlı bir şekilde tekrar planı takip eden bu sınırlama ortamdaki son değişikliklere hızlı bir reaksiyon verir ve daha sonra algoritma ikinci zaman adımına geçer. Bu metotta algoritma hedefe ulaşıncaya kadar planlama yapar. Optimum yolu bulmak için ağırlıklı A* arama yöntemini kullanır. Ayrıca dünya uzayı daha büyük olduğunda ve başlangıç pozisyonu ve hedef pozisyon birbirinden uzak olduğunda zaman miktarı artacaktır. Eğer ortamdaki dinamik engellerin sayısı artarsa planlama zamanı büyümektedir. Bu metot hakkındaki önemli nokta şudur ki; zaman sınırı bittiğinde ve daha sonra arama 2D Dijkstra arama yöntemine geçtiğinde bu teknik yerel minimuma yakalanmış olacaktır.

Dechter and Perl tarafından yapılan çalışmada IAA* olarak adlandırılan hızlandırılmış A*'ın yeni bir versiyonu sunulmaktadır [42]. Uyarlamalı örneklemeye (adaptive sampling) dayalı olan bu algoritma hız ve hassasiyet arasındaki dengeyi ortadan kaldırmaktadır. Hızlı bir planlama metotlu IAA* büyük ortamlar için uygun bir tekniktir. Aracın kabul edilebilir hareket eylemlerinin bir dizisi olan uyarlamalı bir parametreleştirme vardır. Her üretimde küçük durumlar büyüyünceye kadar bu uyarlamalı parametreleştirme düşünülmektedir. Şöyle ki robot engelden uzak olduğunda algoritma bir daha büyük bir adım yapabilir ve robot engele yakın olduğunda bir daha küçük adım yapabilir. Bu teknik yolu buluncaya kadar bir çok arama yaklaşımı çalıştırır. Bu yüzden bu algoritmanın planlama aşamasını zaman sınırına nasıl paylaştırdığı net değildir ve aynı zamanda algoritma zaman sınırından daha önce bir çözüm bulamazsa eğer gelecek adım için ne yapılması gerektiğini bilmemektedir.

1.1.1.2. Rastgele Örnekleme Metotları

Rastgele yaklaşım arama boyunca keşfedilmiş durumları önemli sayıda azaltmak için kullanılmıştır. Bu yaklaşım, artan durum uzayından ve gereğinden fazla keşfedilmiş durumdan dolayı aramanın imkansız olduğu durumların üstesinden kolay bir şekilde gelebilmektedir. Hızlıca rastgele keşfetme ağacı (Rapidly-Exploring Random Tree-

RRT) bir rastgele örnekleme algoritmasıdır [43]. Bu algoritmanın çalışma prensibi arama için plan yapar ve başlangıç durumundan dışa doğru rastgele bir ağaç uzatır. Ağacın uzaması rastgele olmasına rağmen keşfedilmemiş bölgeler diğer bölgelerden daha fazla göze çarpan ağırlığa sahiptir ve arama algoritması onlara karşı önyargılıdır. RRT robotik alanında en ünlü rastgele örnekleme algoritmalarından birisidir ve bu algoritma çeşitli robotik uygulamalarında kullanılır. RRT gerçek zaman performansını garanti edemez. Ek olarak, RRT ağaç uzatmada maliyeti düşünmez bu yüzden bulunan yol yüksek maliyetli alanlarda yerleşmiş olabilir. RRT' nin gerçek zamanlı versiyonu ERRT' dir [28]. ERRT her yaklaşımda bazı bilgileri kaydeder ve daha iyi bir yol bulmak için önceki yaklaşımlarda kaydedilmiş olan bilgiyi kullanır. Ölçülü uyarlamalı RRT (MA-RRT) algoritması RRT' nin değişik bir türüdür [29]. Bu algoritma RRT' nin problemleri ile ilgilenmektedir.

MA-RRT tıpkı ERRT gibi önceki yaklaşımın bilgisini kullanır ve önceki bilgiyi kullanarak MA-RRT aramayı daha iyi ve optimum yola daha yakın yolları içeren bölgelere doğru yönlendirebilmektedir. RRT* ağaç uzatmak için farklı yaklaşım kullanır [30]. RRT*' a göre ağaç bir düğüm boyunca henüz rastgele büyür. Aynı zamanda RRT* ilk olarak ağacı bir düğümle büyütür ve bu uzamayı değerlendirir. Eğer o düğümden uzama başarılı bir uzama olarak sayılırsa daha sonra RRT* eş zamanlı olarak önerilen düğüme olan mesafesi belli bir eşik değeri aşmayan diğer bütün düğümlerde ağacı inceler ve büyütür. Bu algoritma RRT' den daha iyi bir çözüm bulmasına rağmen özellikle arama zamanı artarsa o yüksek hesaplama maliyetine sahip olmaktadır.

1.1.1.3. Reaktif Metotlar

Genelde reaktif metotlar onların yolunu bulmak için herhangi bir arama yapmaz. Aramanın yerine onlar türevlenebilir fonksiyon kullanır. Varsayıma dayalı bu metodun işleyişi engelden uzak ve hedefe doğru yamaç eğimlerinin türevlenebilir bir fonksiyonuna harita olabilir. Robotun eylemleri eğime yaklaşmış olmalıdır. Potansiyel alan [54] bu kategorideki algoritmalardan birisidir [31]. Bu kategorinin temel sıkıntısı yerel minimuma yakalanmadır.

1.2. OTONOM HAREKET

Ortam özelliklerinin sıklıkla ve hızlıca değiştiği gerçek dünyada, mobil robotlar, karşılaştıkları engellere çarpmadan ilerleyerek istenilen noktaya ulaşabilmeli ve diğer robotlarla iletişim kurabilmelidir. Üstelik eşzamanlı olarak kendi kaynaklarını denetleyebilmeli ve mümkün olduğunca kendi kendine yetebilmeli, yani otonom olmalıdırlar [33]. Robotların karşılaştıkları engellere çarpmadan ilerlemesi istenilen noktaya ilerleyebilmesi için engelin tespit edilmesi ve engelden kaçınması aşamalarından oluşmaktadır.

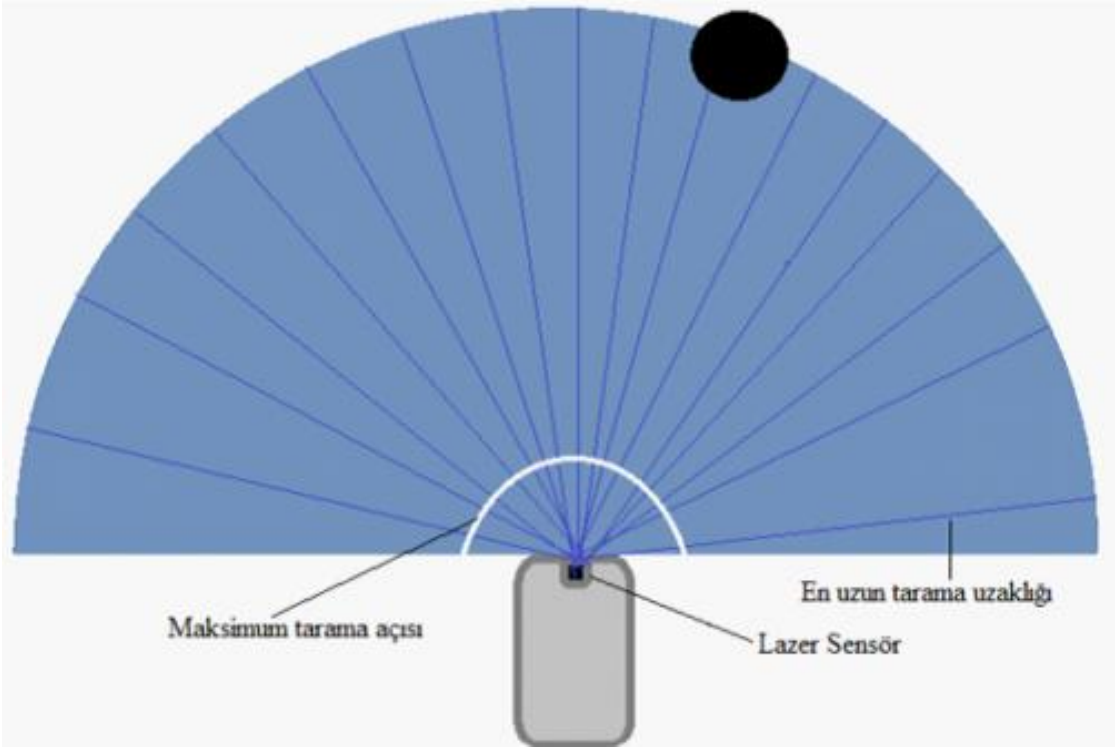
Engel tespiti için çeşitli sensörlerin kullanılması söz konusu olabilir. En çok önerilen yöntemler sonar ya da lazer sensörlerdir. Mobil robotlar için engel tespitinde sonar sistemler ile çok çalışmalar yapılmıştır. Sonar sistemlerin maliyeti de oldukça düşüktür, bunun yanında lazer sensörden gelen veriler daha güvenilirdir ve daha çok sonuçlar vermektedir. Lazer sensörde ışınlar bir alandaki her bir açıda karşısındaki uzaklık bilgisini vermektedir. Lazer tarayıcılar engel tespiti için daha iyi bir tercih olmaktadır [34].

Robotun belirlenen bir hedefe ilerleyebilmesi için o anki konumunu, hedef noktayı ve yönünü bilip ona göre hedefe yönelmesi gerekmektedir. Bunun için robotun kendi konumunu tayin edebilmesi için küresel konumlama sistemi (gps) ve yön bilgisi için magneto sensör kullanılması gerekmektedir.

1.3. LAZER SENSÖR

Temelde bir lazer ve bu lazer ışınının algıladığı alanı algılayan kamera teknolojisi çalışma prensibi ile 3 boyutlu nokta bulutu elde eden sistemlerdir. Lazer tarama sistemleri lazer ışının parça üzerine yansıması ve bu ışının geri yansıması sonucu kameranın lazer ışınının üzerinde düşürdüğü noktaların koordinatlarının belirlenmesi ile veriyi elde eder.

Kullanılacak sensorun özelliğine göre maksimum tarama açısı, maksimum tarama mesafesi değişmektedir. Maksimum tarama aralığındaki tarama aralıkları çözünürlüğü vermektedir.



Şekil 1.1. Tarayıcı uzaklık tespiti.

1.4. KÜRESEL KONUMLAMA SİSTEMİ (GPS)

Dünya üzerinde herhangi engelsiz bir görüş hattında, dört veya daha fazla uydusu ile her türlü hava koşulunda yer ve zaman bilgileri sağlayan uzay tabanlı uydu navigasyon sistemidir.

Bu sistem robotun dünya üzerindeki konum bilgisini enlem ve boylam olarak elde etmemizi sağlamaktadır.

1.5. MAGNETOMETRE

Magneto sensör sistemi dünya referansına göre uzaydaki duruş yönünü vermektedir. Robotun baş açısının hangi yönde olduğu bilgisini elde etmek için kullanılmaktadır.

1.6. ROBOTİK UYGULAMALARDA SİMULASYON ÇALIŞMALARI

Robotik çalışmalar yaparken bütün çalışmaların gerçek ortamda yapılması zaman, enerji ve para bakımından maliyetli olmaktadır. Bu bağlamda işlerin daha hızlı ve maliyetsiz olarak ilerleyebilmesi için bir simülatör önem kazanmaktadır.

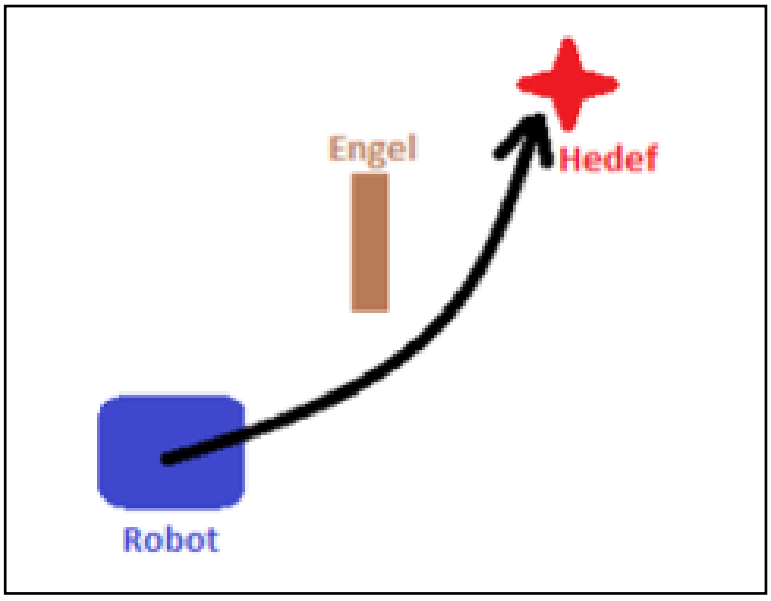
Gazebo, hem kapalı hem de açık mekanlar için geliştirilmiş açık kaynak kodlu bir çoklu robot simülatörüdür. Birden fazla robot, sensor ve cisimleri üç boyutlu ortamda simüle edebilme yeteneğine sahiptir [36].

BÖLÜM 2

ENGEL AŞMA VE ALGORİTMASI

Belirli bir hızda ve bir yöne doğru hareket eden robot karşısına bir engel çıktığında ondan kaçmak için yönünü değiştirmesi gerekmektedir. İleri yönde hızla ilerlemekte olan robota, devrilmeden yön verebilmek için hızını düşürmek ve farklı bir yöne yönelmesini sağlamak gerekmektedir. Engele maksimum ne kadar uzaklıkta iken hızını azaltmaya başlaması robotun hangi hızla gittiği ile orantılıdır.

Belirli bir ortamda hareket etmesi planlanan robot için ortamda bulunan sabit engellerin konumlarının bilgisi elimizde mevcuttur. Robot bulunduğu konumdan hedef olarak belirlenen noktaya giderken sabit engellerin konumlarını kullanıp bu bölgelere çarpmayacak şekilde bir rota planlaması yaparak hareketine devam edebilir.



Şekil 2.1. Sabit engelin bulunduğu bir ortam.

Bu noktada dikkat edilecek bir konu hedef noktaya giderken en kısa mesafeden gitmesi, en uygun yolu bulması olacaktır.

Bu amaç doğrultusunda en kısa yol bulma algoritmaları incelenerek, A* algoritmasının problemin çözümünde faydalı olabileceği öngörülmüştür.

2.1. A* ALGORİTMASI İLE ROTA PLANLAMASI

Birçok uygulamalarda, hareketi denetlenen nesnenin (robotun) engellere çarpmadan bir başlangıç konumdan bir hedef konumuna en kısa yolla gitmesinin sağlanması gerekir. Problemin sayısal çözümü için çeşitli algoritmalar önerilebilir.

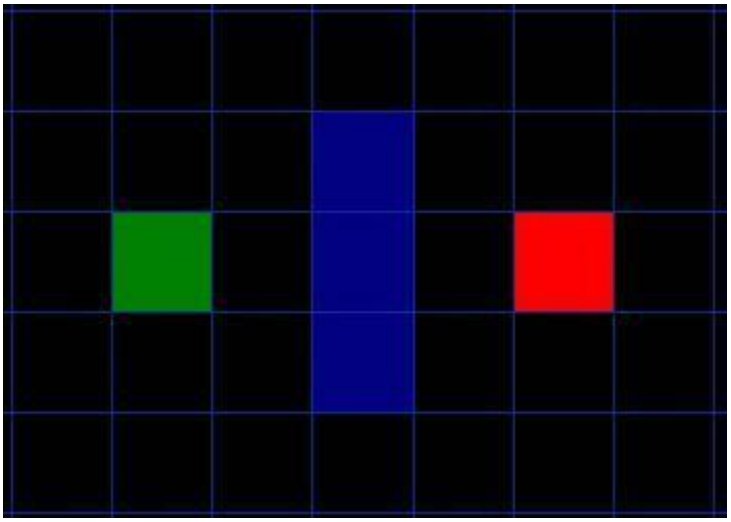
A* başlangıç düğümünden hedef düğüme giden en az maliyetli yolu bulur. Ortamın bilgisine ulaşarak, toplam maliyet veya yolun en az olduğu yolu izler. A* algoritmasında hedef düğüme doğru, doğrudan bir maliyet hesaplanır [39].

En kısa yol hesaplanırken; toplam maliyet işlevi f(x), uzaklık işlevi g(x) ile sezgisel (heuristic) işlev h(x) toplanarak elde edilir.

$$F(x) = g(x) + h(x) \tag{2.1}$$

2.1.1. Arama Alanı

A noktasından B noktasına gidilmek istendiğini varsayalım. Ayrıca iki nokta arasında bir duvar olduğunu varsayalım. Şekil 2.2’ de gösterildiği üzere yeşil nokta (A) başlangıç noktasını, kırmızı nokta hedef noktasını (B) ve mavi noktalarda aradaki duvarı temsil etmektedir .



Şekil 2.2. A* arama alanı.

Arama alanını kare ızgaralara bölerek arama alanının sadeleştirilmesi işlemi yol bulmanın ilk adımıdır. Bu özel metot arama alanımızı iki boyutlu basit bir diziye çevirmektedir. Dizideki her eleman ızgara üzerindeki bir kareyi temsil etmektedir ve elemanın durumu geçilebilir ya da geçilemez olarak kaydedilmektedir.

Yol, A' dan B' ye gitmek için geçmemiz gereken karelerin bulunmasıyla elde edilmektedir. Yol bulunduktan sonra, karakterimiz hedefine ulaşana kadar sırasıyla kareden kareye hareket etmektedir.

Karelerin merkez noktalarına düğüm (node) denilmektedir. Arama alanımızı kareden başka şekillere de bölebiliriz. Bunlar üçgen, dikdörtgen, altıgen veya herhangi bir şekil bile olabilir. Düğümler bu şekiller içerisinde herhangi bir konumda da olabilir, köşelerden birinde, kenarlardan vb. Biz karelerin merkezini kullanıyoruz çünkü bu en basit olanıdır.

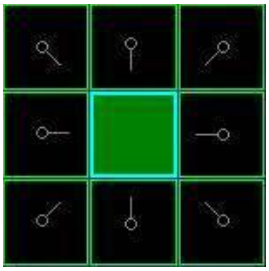
2.1.2. Aramaya Başlamak

Arama alanımızı yukarıdaki gibi grid dağılımıyla sadeleştirdikten sonra sıradaki adım en kısa yolu bulmak için arama yapmak olacaktır. Bu işi A noktasından başlayarak yapacağız. Komşu kareleri tarayarak, hedefe ulaşana kadar dışa doğru açılarak tarama işlemi yapılacaktır.

Arama işlemine şu işlemlerle başlıyoruz:

i. A noktasından başla ve onu hesaba alınacak karelerin bulunduğu açık listeye ekle.
ii. Duvar, su birikintisi vb. erişilemez kareleri ihmal ederek başlangıç karesinin çevresinde bulunan (komşu) erişilebilir karelere göz at. Bunları da açık listeye ekle ve başlangıç noktasını bu noktaların ebeveyni olarak ata.
iii. Başlangıç karesi A'yı açık listeden çıkar ve kapalı listeye ekle.

Bu noktada Şekil 2.3’ de ki gibi bir durum elde edilecektir. Şekil 2.3’ de yeşil kare başlangıç noktasını temsil etmektedir. Açık mavi kareyle çevrilmiş olması kapalı listeye eklendiğini göstermektedir. Tüm komşu kareler şu an açık listededir, açık yeşil ile işaretlenmişler ve kontrol edilmek için beklemektedirler. Her birinde ebeveyn karesini (şu durumda başlangıç karesini) gösteren gri bir işaretçi vardır.



Şekil 2.3. Arama yapılması.

2.1.3. Yol Skorlama

Yolu bulurken hangi karelerimizi kullanacağımızı belirlemede aşağıdaki denklem anahtar bir görev üstlenir:

F = G + H

G = Başlangıç noktası A 'dan grid üzerinde istenen bir kareye ulaşmak için gidilmesi gereken yolun ölçütüdür. Şuana kadar bulunmuş olan yol kullanılarak hesaplanır.

H = Verilen bir kareden hedef noktası B'ye ulaşmak için gidilmesi gereken yol miktarının tahmini ölçütüdür. Bu kavrama heuristic (tahmin) denir. Bu şekilde isimlendirilmesinin sebebi bir tahmin yürütmesidir. Gerçek yolu bulana kadar ne kadar yol gidilmesi gerektiğini bilemeyiz çünkü yol üzerinde öngöremeyeceğimiz engeller olabilmektedir.

Yolumuz, açık liste üzerinden tekrar tekrar geçerek, en düşük F skoruna sahip kareleri seçerek oluşmaktadır.

G, başlangıç noktasından istenilen bir noktaya türetilen yol ile gitmek için kat edilmesi gereken yolun miktarıdır. Bu örnekte, bir kareden diğerine yapılan yatay ve dikey hareketler 10'ar puan, köşegen üzerinden yapılan çapraz hareketler 14 (10*kök(2)) puandır. Bu değerler işi basit tutmak için seçilmiştir. Böylece karekök hesaplarına ya da ondalıklı sayılarla uğraşmamıza gerek kalmayacaktır.

G değerini istenilen kareye giden belirli bir yol vasıtasıyla hesapladığımıza göre bunu hesaplamanın yolu istenilen karenin ebeveyninin G değerine (yatay yada çapraz hareket olabilir) 10 yada 14 eklemektir. Bu işlemin gerekliliği başlangıçtan birden fazla kare uzaklaşınca daha da açık hale gelecektir.

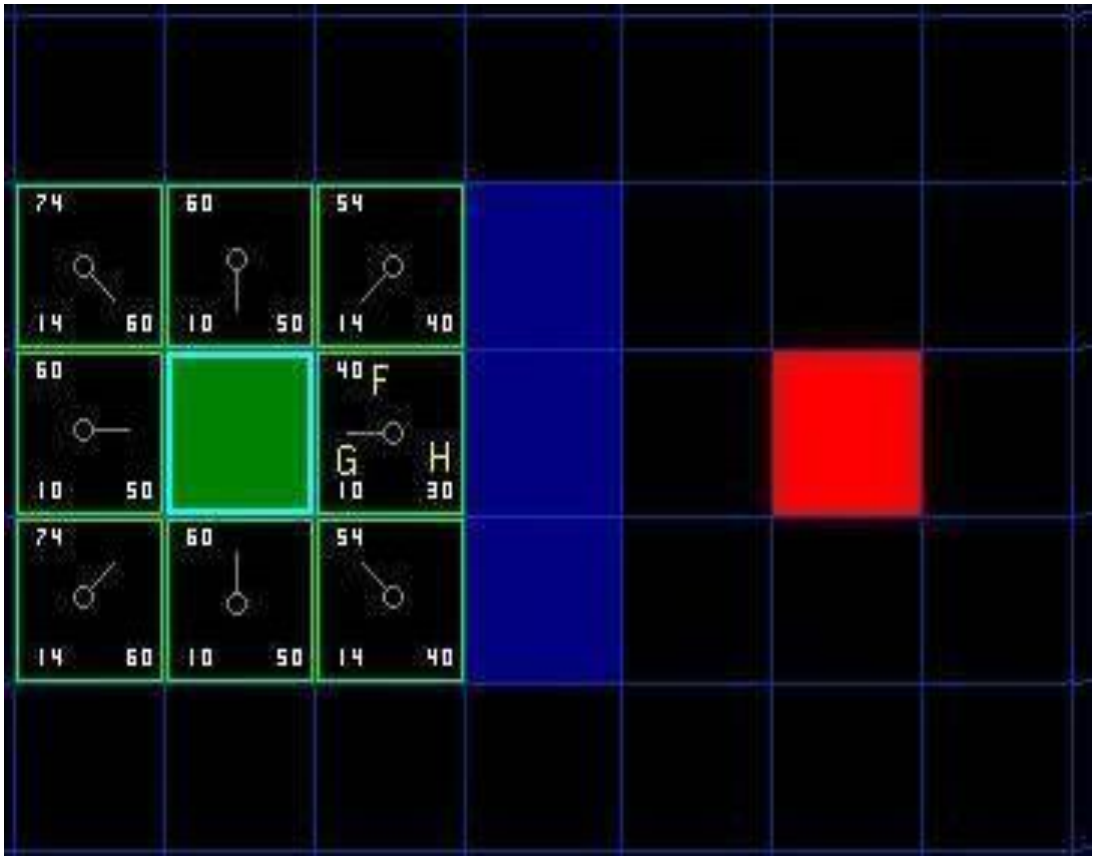
H değeri birçok yolla hesaplanabilir. Burada kullandığımız metodun adı Manhattan metodudur. Bu metot istenilen kare ile aramızda bulunan yatay ve dikey kare farkının toplamını, aradaki engelleri ve çapraz hareketleri göz ardı ederek hesaplamaktadır. Elde edilen sonuç 10 (yatay yada dikey bir kare hareket etmenin ölçütü) ile çarpılmaktadır. Bu ismin kullanılmasının muhtemel sebebi iki mekan arasındaki apartman bloklarının hesaplanmasına benzer olmasıdır. H değerleri kırmızı kareye olan manhattan mesafesi ile hesaplanırken yol üzerindeki duvar vb. engeller dikkate alınmaz.

Sezgisel (Heuristic) mevcut kare ile hedef arasındaki mesafenin tahmini bir ölçütüdür. Aslında burada yapılan tahmin kalan yolun uzunluğunun tahminidir. Yapılan tahmin kalan yolun uzunluğuna ne kadar yakınsa, algoritma o kadar hızlı

çalışmaktadır. Eğer fazla iyimser tahminler yaparsak en kısa yolu bulabilmemiz garanti olmaz. Bu tip sezgisellere kabul edilemez sezgisel denir.

Bu örnekte teknik olarak manhattan metodu kabul edilemezdir. Çünkü kalan mesafeyi tahmin ederken iyimser davranır. Yine de biz bu metodu kullanacağız çünkü hem anlaşılması kolay hem de tahmin iyimserliği bizi optimal olmaktan çok uzak yollar bulmaya sevk etmez.

F, G ve H' nin toplanması ile hesaplanır. Arama olayındaki ilk adımımızın sonucu Şekil 2.4' de gösterilmektedir. Her karenin F, G ve H skorları yazılıdır. Başlangıç karesinin hemen sağındaki karede gösterildiği üzere, F karenin tepesindeki, G sol alttaki ve H sağ alttaki sayıdır.



Şekil 2.4. Arama olayında ilk adımın sonucu.

2.1.4. Aramaya Devam Etmek

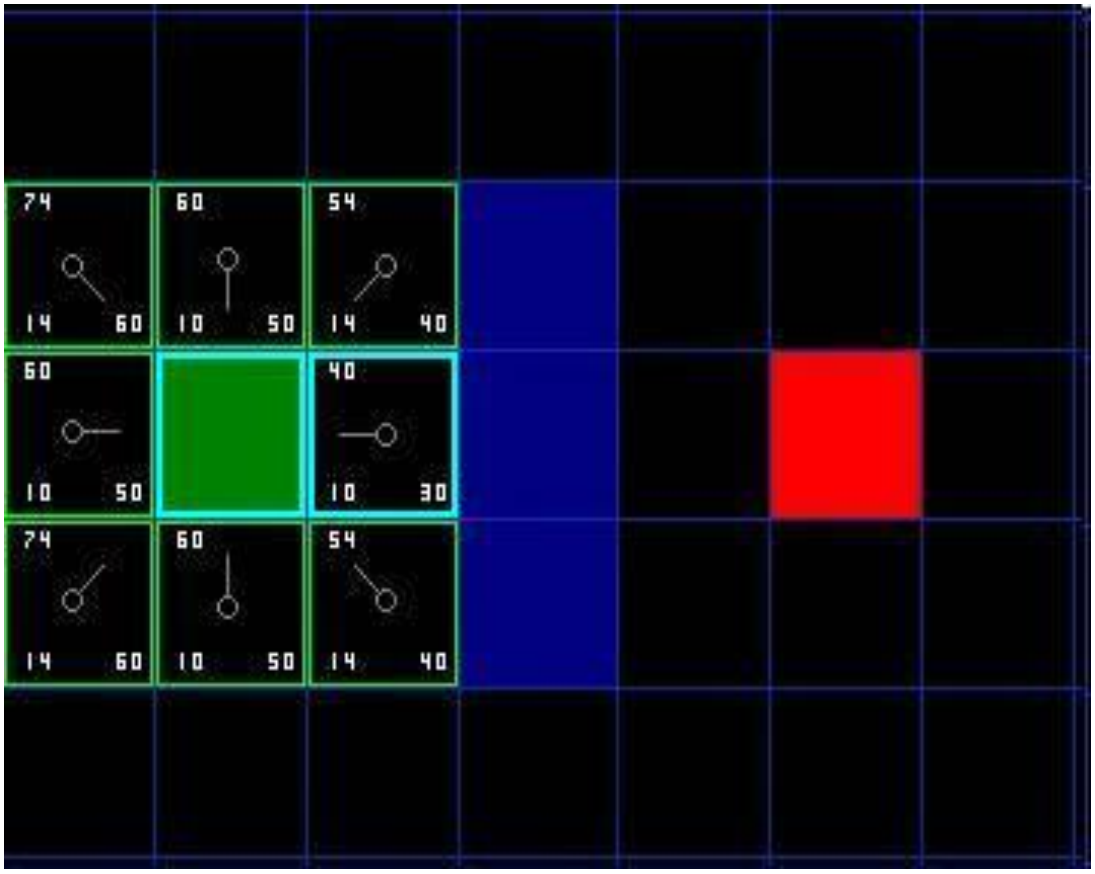
Arama işlemine devam etmek için, açık listede bulunan en düşük F skorlu kare seçilmektedir. Daha sonra seçilen kare ile şu işlemler yapılmaktadır:

i. Açık listeden çıkar ve kapalı listeye ekle. Seçili kare olarak düşün.

ii. Tüm komşu kareleri kontrol et. Erişilemez (duvar, su vb.) yerleri ve zaten kapalı listede olanları göz ardı ederek, açık listede olmayan kareleri açık listeye ekle. Bir önceki adımda seçtiğimiz kareyi eklenen elemanların ebeveyni olarak işaretle.

iii. Eğer komşu karelerden biri zaten açık listede ise, şu an elde ettiğimiz yolun önceki yola göre daha iyi olup olmadığını kontrol et. Bir başka deyişle, şu anki seçili kareyi kullanarak bu komşu kareye ulaşırsak G değeri daha mı düşük olur diye kontrol et. Daha düşük olmuyorsa hiçbir şey yapma. Diğer taraftan G değeri daha düşük oluyor ise, komşu karenin ebeveynin seçili kare olarak değiştir. Son olarak bu karenin F ve G değerlerini yeniden hesapla.

Yukarıda anlatılanları bir örnek ile açıklamak gerekirse, ilk 9 kareden başlangıç karesi kapalı listeye alınıp ve diğer 8 karede açık listede tutulur. Bu 8 komşu kareden en düşük F skoruna sahip olan hemen sağ taraftaki 40 puanlı karedir. Daha sonra bu kareyi sıradaki kare olarak seçip ve aşağıdaki şekilde mavi renk ile işaretlenmektedir.



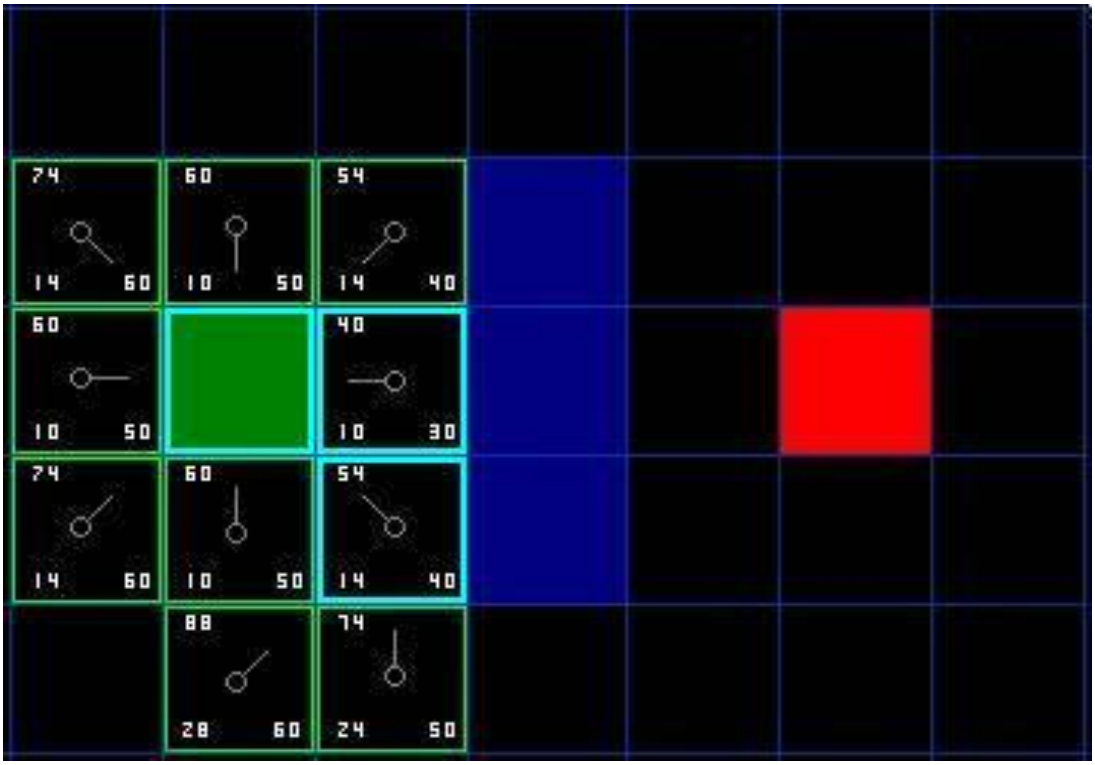
Şekil 2.5. Arama olayında en küçük F değerini seçmek.

İlk olarak seçili (mavi kare içinde) kare açık listeden çıkarılıp kapalı listeye eklenmektedir. Sonra bu karenin komşuları kontrol edilmektedir. Sağdaki kareler duvar olduğu için bunlar görmezden gelinmektedir, hemen solundaki kare ise çoktan kapalı listede olduğu için o da ihmal edilmektedir.

Kalan 4 kare zaten açık listede olduğu için seçili kareyi kullanarak o karelere gitmek daha mı uygun kontrol edilmelidir (G skorunu referans alarak). Seçili karenin hemen üstündeki kareyi ele almak gerekirse, (Şu anki G skoru 14) eğer seçili kare üzerinden oraya gidilseydi G değeri 20 olacaktır (10 seçili kareye gelmek için, +10 seçili kareden bir üstündeki kareye gitmek için). 20 değeri 14'ten büyük olduğu için bu yolu kullanmak daha iyi bir seçim değildir. Başlangıç karesinden oraya doğrudan çapraz olarak bir hamlede gitmek, bir adım sağa sonra bir adım yukarı gitmekten daha mantıklı görünmektedir.

Bu işlemi açık listede bulunan 4 komşu kare için tekrarlarsak göreceğiz ki seçili kareyi kullanmak hiç biri için yolu iyileştirmemektedir. Bu yüzden hiçbir şeyi değiştirmeden tüm komşuları kontrol ettikten sonra seçili kareyle işlem bitmektedir.

Bu adımda açık listedeki 7 kareyi tek tek kontrol edip en düşük F puanına sahip olanı seçilmektedir. Şu durumda en düşük skora (54) sahip iki kare bulunmaktadır. Hangisinin seçileceği önemli değildir fakat performans açısından açık listeye en son ekleneni seçmek daha uygun olacaktır. Böylece arama işlemi hedefe yaklaştıkça daha geç bulunmuş kareleri kullanma eğilimine itilmektedir. Aslında çokta önemi olan bir şey değildir, iki farklı A* versiyonu aynı uzunlukta farklı yollar bulabilir. Bu durumda Şekil 2.6 'da gösterildiği üzere sağ alttaki kareyi seçerek devam edilmektedir.



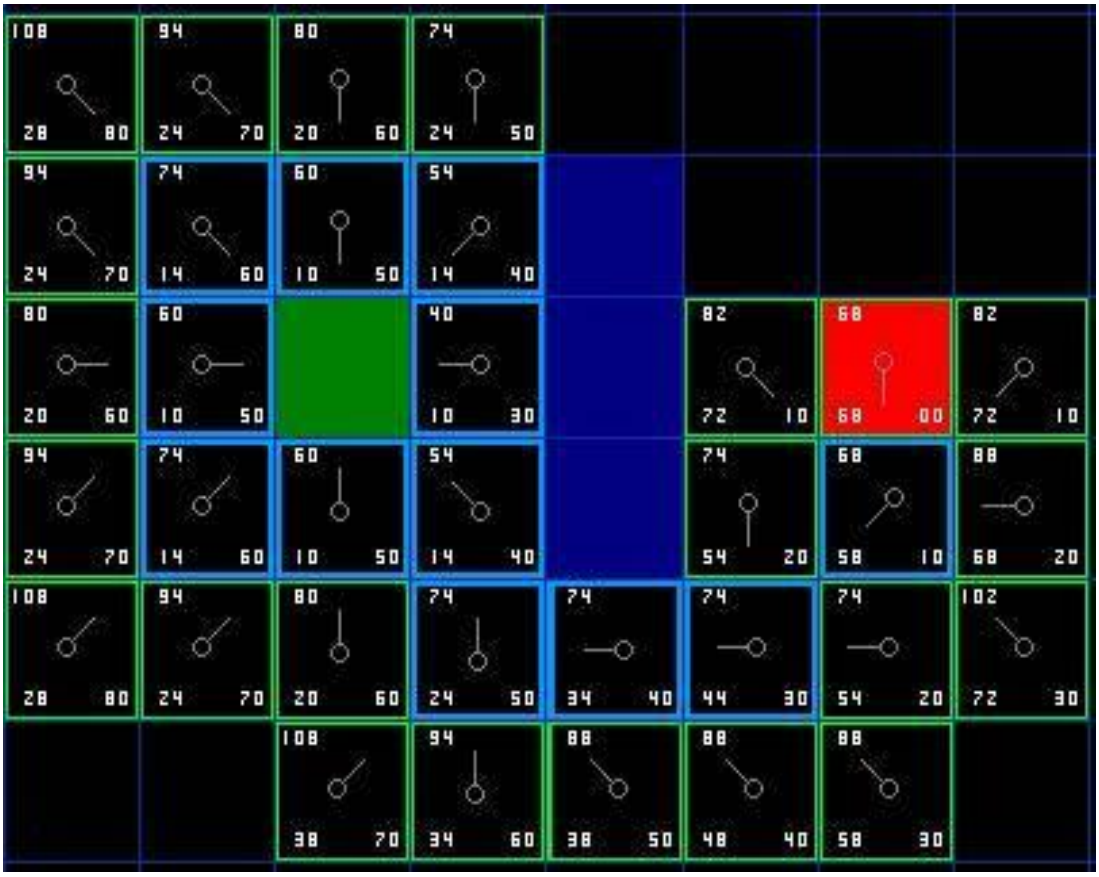
Şekil 2.6. Arama olayına devam etmek.

Bir sonraki adımda komşu kareler kontrol edilir ve hemen sağdaki kare bir duvar olduğu için pas geçilmektedir. Onun üstündeki kare içinde aynı şey geçerlidir. Duvarın hemen altındaki kare de pas geçilmektedir. Çünkü şu anki kareden o kareye

duvarı köşeden kesmeden gidilemez. İlk önce aşağı inmeli sonra köşeyi dönerek o kareye gidilmelidir.

Böylece geriye beş kare kalmaktadır. Seçili karenin altında kalan 2 kare şu an açık listede değildir, bu yüzden onlar listeye eklenip seçili kare onların ebeveyni yapılmaktadır. Kalan üç kareden ikisi zaten kapalı listede (başlangıç karesi ve hemen sağındaki) olduğu için bunlar da pas geçilmektedir. Sona kalan hemen soldaki karenin ise G skoru, seçili kareden erişilirse düşer mi diye kontrol edilmektedir. Böylece komşu kareler biter ve açık listede bulunan sıradaki kareye geçilebilir.

Bu süreci aşağıdaki şekilde de görüldüğü üzere, hedef kare kapalı listeye eklenene kadar devam etmektedir.

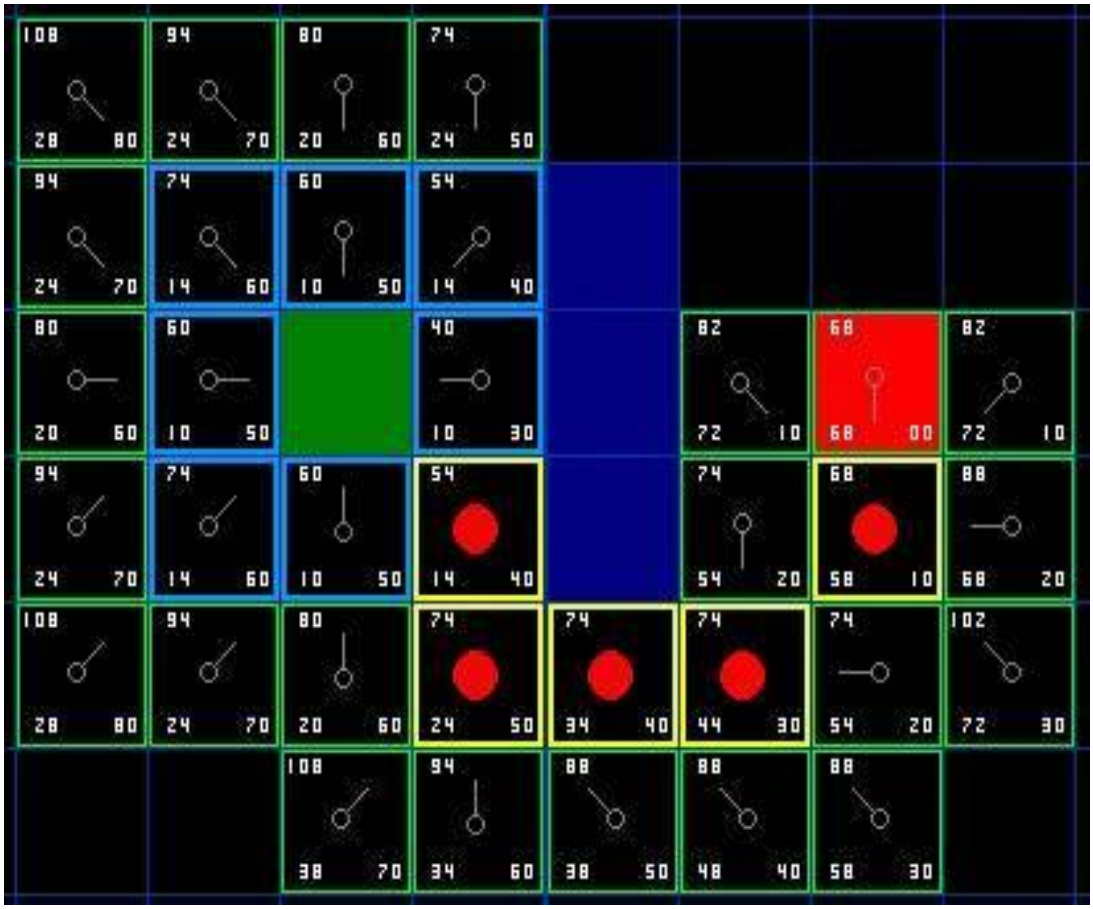


Şekil 2.7. Arama olayında hedefin kapalı listeye eklenmesi.

Dikkat etmek gerekirse, başlangıç karesinin 2 kare altında bulunan karenin ebeveyni önceki şekildekinden farklıdır. Daha önce G skoru 28 idi ve de sağ-üst kareyi

gösteriyordu. Şimdi ise skoru 20 ve hemen üstündeki kareyi göstermektedir. Bu olay, arama sürecinin bir bölümünde, G skoru kontrol edilip, yeni yol ile daha düşük olacağı bulunduğunda, o karenin ebeveyni değiştirilip, G ve F skorları yeniden hesaplanarak gerçekleşmiştir. Bu değişim bu örnekte pek önemli değil gibi gözükse de, birçok durumda en kısa yolun bulunması konusunda dramatik farklılıklar ortaya çıkaracaktır.

Yolu bulmak için kırmızı kareden başlanmaktadır ve okları takip ederek ebeveyn kareler üzerinden devam edilmektedir. Bu eninde sonunda sizi başlangıç karesine ulaştıracak ve yolu tarif etmiş olacaktır. Aşağıdaki şekilde daha net bir şekilde gözlemlenebilir. A noktasından B noktasına gitmek için basitçe her karenin merkezinden geçerek hedefe ulaşana kadar yolu takip etmek yeterli olacaktır.



Şekil 2.8. Arama olayında yolun bulunması.

2.1.5. A* Metodunun Özeti

A* algoritmasının sözde kodu (pseudo code) aşağıdaki gibidir.

1) Hedef düğümü oluştur; hedef_düğüm diye isimlendir

2) Başlangıç düğümü oluştur;

başlangıç_düğüm seklinde isimlendir

3) Bütün düğümleri AÇIK listeye koy

4) başlangıç_düğümü KAPALI listenin içine koy

5) ata_düğüm_indis listesini sıfırla

6) AÇIK liste boş değilken aşağıyı tekrarla

{

KAPALI listedeki her bir düğüm için en düşük f maliyetli ardıl düğümü bul;

Hedef düğüme ulaşılıp ulaşılmadığını kontrol et; eğer ulaşıldıysa DUR;

Bu ardıl noktayı AÇIK listeden çıkar KAPALI listeye koy

mevcut düğümü ata_düğüm_indis listesine koy}

ata_düğüm_indis listesini kullanarak en kısa yolu bul.

A* metodunu kısaca özetlemek gerekirse,

i. Başlangıç karesini açık listeye ekle.
ii. Aşağıdakileri tekrarla:
    - Açık listedeki en düşük F puanlı kareyi bul. Bunu seçili kare olarak düşün.
    - Seçili kareyi kapalı listeye al.
    - Seçili karenin 8 komşu karesi için;
        - Erişilemez (duvar vb.) ise veya zaten kapalı listede ise pas geç.
        - Açık listede değilse açık listeye ekle. Seçili kareyi bu karenin ebeveyni yap. F, G ve H değerlerini hesapla.
        - Eğer açık listede ise G değerini kullanarak şu anki yol öncekinden daha mı iyi kontrol et. Daha düşük bir G değeri daha iyi bir yol demektir. Öyleyse seçili kareyi yeni ebeveyn olarak ata ve G ve F değerlerini yeniden hesapla. Eğer açık listeyi F skoruna göre sıralı tutuyorsan, listeyi yeniden sıralaman gerekebilir.
    - Aşağıdakilerden biri gerçekleşirse dur:
        - Hedef kareyi kapalı listeye eklersen (Yol bulunmuş demektir).

- Açık liste boş kalırsa (Yol bulunamadı yani yol yok demektir).

iii. Bulunan yolu kaydet. Hedef kareden başlayarak başlangıç karesine ulaşana kadar her karenin ebeveynine git.

2.1.6. A* Metodunun Geliştirilmesi Gereken Yönleri

Bu metot ile varılacak hedefe engellere çarpmadan en kısa yoldan ulaşacak rota hesaplanmaktadır. Ancak bu yöntem ile sadece haritadaki engellerden kaçınılabilmektedir.

Ortamda birden fazla robot dolaşacağı düşünüldüğünden bu robotlar herhangi bir yerde ve herhangi bir zamanda karşı karşıya gelip çarpışabilirler. Bu noktada ek tedbirlerin alınması gerekli olmaktadır. İki veya daha fazla robotun birbirlerine çarpmadan istenilen hedef noktaya ilerleyebilmeleri için lazer sensörden gelen veriler kullanılacaktır.

2.2. LAZER SENSÖR İLE ENGELDEN KAÇINMA

Belirli bir hızda ve bir yöne doğru hareket eden robot karşısına bir engel çıktığında ondan kaçmak için yönünü değiştirmesi gerekmektedir. İleri yönde hızla ilerlemekte olan robota, devrilmeden yön verebilmek için hızını düşürmek ve farklı bir yöne yönelmesini sağlamak gerekmektedir.

Engele maksimum ne kadar uzaklıkta iken hızını azaltmaya başlaması robotun hangi hızla gittiği ile orantılıdır. Geliştirilen yöntemde işlem hızını maksimum tutmak ve hafıza gereksinimini minimumda tutmak için engelin en yakın noktası bulunarak bu noktadan kaçmanın yöntemleri araştırılmıştır.

2.2.1. Engelin Robota Göre Konumunun Belirlenmesi

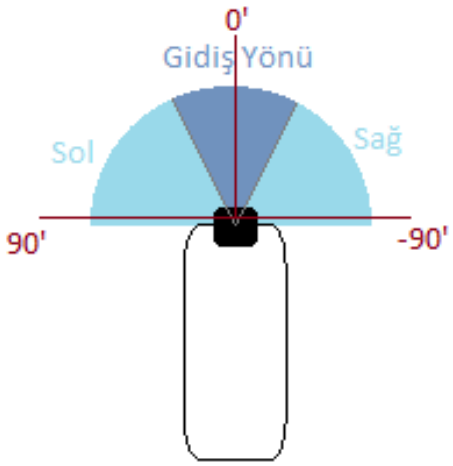
Lazer sensör taradığı alan kapsamında robotun etrafı hakkında veriler vermektedir. Bu bilgiden yola çıkarak robotun baş açısına göre kaç derece açıya karşılık gelen bölgede ne kadar uzaklıkta engel var bilgisi çıkarılabilmektedir.

Lazer tarayıcının taradığı alan robotun baş açısı 0 derece, sağ yanı negatif açı değerleri ve sol yanı pozitif açı değerleri ile temsil edilecek şekilde Şekil 2.9' da ki gibi dönüştürülmektedir. Eğer engel negatif açı karşılığına gelecek bir bölgede ise robot sol tarafa yönelmelidir ya da engel pozitif açı karşılığına gelecek bölgede ise robot sağ tarafa yönelmelidir. Sıfıra yakın açı değerlerinde engel var ise engelin robotun karşısında olduğu anlamına gelmektedir.

2.2.2. Engel Durumlarının İncelenmesi Ve Yön Kontrolü

Lazer sensör robota göre hangi açıda ne kadar uzaklıkta bir cisim olduğu bilgisini vermektedir. Kullandığımız sensör 180 derece tarama yapmaktadır (Şekil 2.9). Robotun yön değiştirmesine karar verildikten sonra ne tarafa döneceğine karar verilmesi gerekmektedir. Bu karar için öncelikli olarak robotun dönüş yapacağı yönün boş olup olmadığı veya ne tarafın daha boş olduğuna bakılması gerekir. Robotun sağ ve sol tarafındaki boş alanların hesaplaması yapılır. Sağ taraftaki engelli alan sayısı sol taraftakinden daha az ise sağ tarafa dönüş yapılması gerekir. Fakat robotun köşeye denk geldiği noktalarda sağ ve sol taraftaki engel bulunan açısal bölge sayıları eşittir, bunun için buraya bir sınır getirilmesi uygun olacaktır.

Okunan açı değeri robota göre sol tarafta ise ve uzaklık değeri belirli bir değerin altındaysa sol tarafta engel büyüklüğü artırılır ve sol taraftaki belirli mesafenin altındaki uzaklık değerleri arasında robota en yakın değer bulunarak o değerin hangi açıda yer aldığı bilgisi elde edilir. Aynı işlemler sağ taraf için ve gidiş yönü için de hesaplanır.



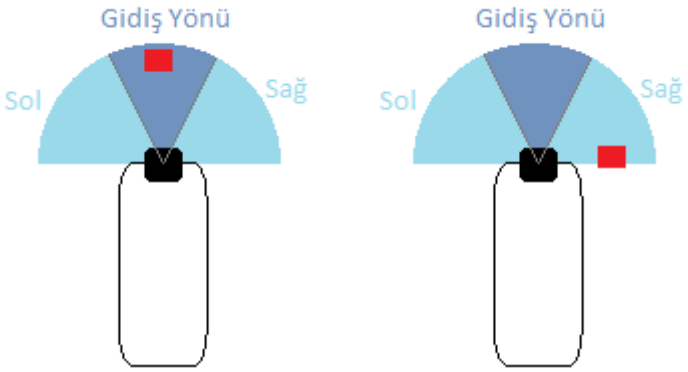
Şekil 2.9. Robota göre yönler.

Böylece elimizde her bir bölge için şu bilgiler elde edilir:

- Yakın mesafedeki engelin alanı,
- Engelin robota en çok ne kadar yakın olduğu ve bu uzaklığın hangi açıda olduğu.

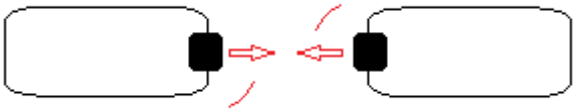
Sağ taraftaki engel alanı sol taraftaki engel alanından küçük iken sol taraftaki engel robota oldukça yakın mesafede ise robot sağa dönmeli veya tam tersi de geçerlidir. Engel alanları eşit ise hangi taraftaki engel daha uzak ise o tarafa yönelme olmalıdır. Her iki taraftaki engel robota çok yakın mesafede ise robot iki tarafa da dönemeyeceği için durmalıdır. Unutulmamalıdır ki robot hiç bir durumda geri gitmemektedir, çünkü robot arkasını görmüyor.

Engelin gidiş yönünde olması ve yan taraflarda olması durumuna göre alınacak önlem farklılık göstermesi beklenmektedir. Şöyle ki; eğer engel gidiş yönündeyse hemen tedbir alınmalıdır fakat sağ veya sol yönlerde ise robotun gidişini engellememektedir. Bu durumda engellerin robotun yan taraflarının çarpmayacağı kadar uzağında olması yeterli olmaktadır (Şekil 2.10).



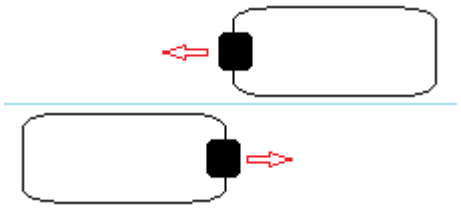
Şekil 2.10. Robota göre engel durumları.

Göz önünde bulundurulması gereken bir diğer nokta Şekil 2.11‘ de görüldüğü gibi robotun, karşısında bir engel gördüğünde sağa veya sola doğru yön değiştirip engeli geçip daha sonra hedefine tekrar yönelmesidir.



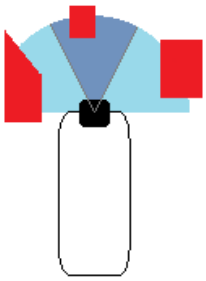
Şekil 2.11. Robotların karşı karşıya gelmesi.

Robot karşısında engel gördüğünde sağa dönsün dediğimizde her iki robotta sağa yönelecektir, engeli aştığını sanıp tekrar sola hedefine dönmek isteyecektir tekrar engel görüp sağa dönecektir ve burada robotların takıldığı gözlemlenecektir. Böyle bir duruma mahal vermemek için robotlar sağa döndüğünde, solda en uçta engel varsa engelden kurtulana kadar ileri gidip hedef noktaya daha sonra yönelmelidir (Şekil 2.12).



Şekil 2.12. Robotun karşısındaki engeli geçmesi.

Robotun etrafındaki engelli alanlar robotun geçemeyeceği kadar büyükse robot durmalı (Şekil 2.13).



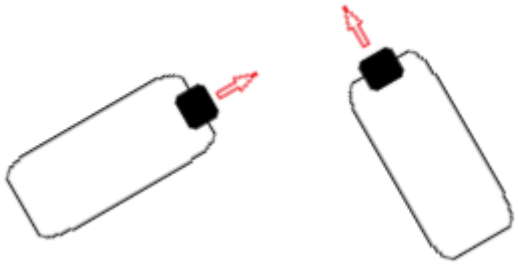
Şekil 2.13. Robotun geçemeyeceği kadar büyük engeller.

Robotlar çapraz yönlerden birbirlerine doğru hareket ettiğinde (Şekil 2.14) orta noktada çarpışma ihtimalleri vardır ve böyle bir durumda robotlar aynı hareketleri yaparsalar kilitlenip aynı yerde kalabilirler. Robotlar şekil 2.14' de ki gibi karşı karşıya geldikleri durumda robotlardan sağdaki engeli solda gördüğü için sağa doğru, soldaki engeli sağda gördüğü için sola doğru dönecek ve her ikisi paralel olarak ilerlemeye başlayacaktır. Engelden kurtulana kadar ileri git ve sonra hedefe tekrar yönel durumundan dolayı her ikisi de birbirini geçmeye çalışacak, ama geçemeyecek önlerine bir engel çıkana kadar ileri doğru yol alacaklardır.

Şekil 2.14. Robotların çapraz karşılaşması durumu.

Şekil 2.14'de ki durumu atlatabilmek için robotlardan birinin diğerine öncelik vermesi gerekmektedir. Her ikisi de diğerini aşmaya çalışırsa ikisi de ileri doğru gider ama birbirlerini geçemezler. Bu problem için trafik kurallarında iki araba bir noktada karşılaştığında nasıl öncelik sağ taraftakinde ise robotlarda da bu durum uygulanmalıdır. Yani eğer engel robotun solunda ise bunu daha önce tespit edip sağa doğru yönelmeye başlamalı ve diğer robot henüz engeli görmediği düşünülmelidir

(Şekil 2.15). Böylece simetri bozulur ve robotlar birbirlerini geçerek yollarına devam edebilirler.



Şekil 2.15. Engel robotun sol tarafında ise engeli daha önce tespit eder.

2.2.3. Hız Kontrolü

Robot ivme sayesinde ilk hareketine başlamakta ve sabit bir ivme sürekli uygulandığında robot düzgün hızlanan hareket yapmaktadır. Robotun belirli bir hızın üstüne çıktığında takla aşması, kontrolünü kaybetmesi söz konusu olduğundan maksimum bir hız sınırı getirilmesi gerekmektedir. Robotun çıkabileceği maksimum hız robotun teker boyları, ağırlığı, yer şekli, sürtünme durumları ile orantılıdır. Engellerin yakınlarından geçerken de daha düşük hızda geçmesi daha güvenli bir hareket olacaktır.

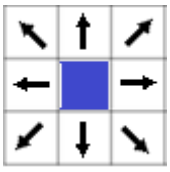
Uygulanan ivmenin değeri de robotun ağırlığı ile ilişkilidir. Ayrıca robotun sola doğru dönme hareketi yapması için sağ taraftaki 2 teker ileri yönde, soldaki iki teker geri yönde dönmelidir ya da robotun sağa doğru dönme hareketi yapması için sol taraftaki 2 teker ileri yönde, sağdaki iki teker ise geri yönde dönmelidir. Bu durumda ileri yönde giden bir robotun dönme hareketi yapması için robotun önce süratle yavaşlaması ve sonra dönmesi gerekmektedir. Robot belli bir hızın üstünde ilerlerken birden dön emri verildiğinde ileri doğru dönen tekeri aniden geriye zorlayacak ve kontrolün elden çıkmasına sebep olacaktır. Yavaşlama ve dönme yapmak için uygulanacak ivme engelin yakınlığı ile ilişkilidir. Ne kadar yakında ise o kadar süratli dönmesi lazımdır. Dönüş yaparken de belirli bir hızın üstüne çıkmaması gerekmektedir ki dönüşünü tamamladığında düzgün hareketi doğru açıdan, engelden kaçtığı noktadan tam olarak devam edebilsin.

BÖLÜM 3

YOL PLANLAMA

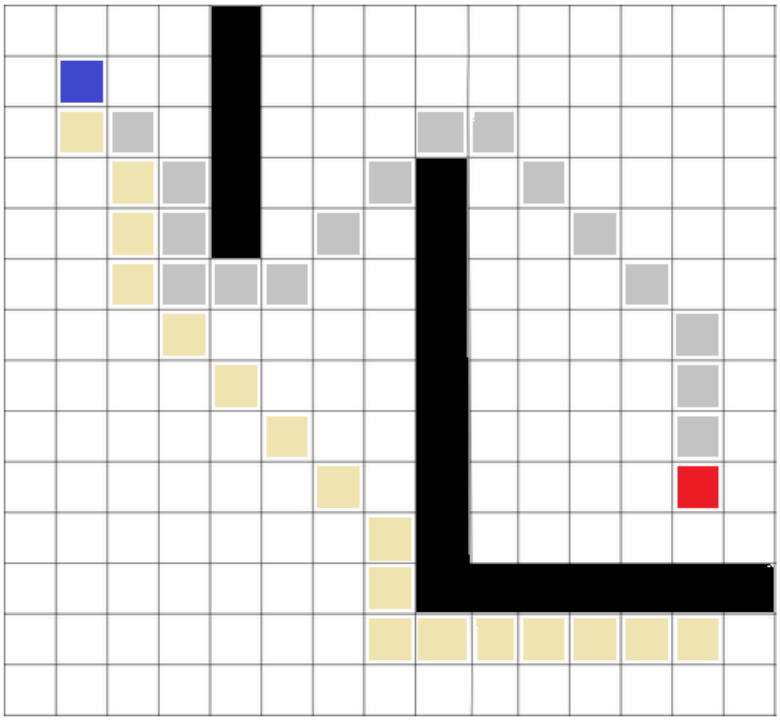
A* algoritması ile sabit engellerin belli olduğu bir harita üzerinden rota hesabı yapılmakta ve lazer sensör ile de engeller tespit edilerek ve engellere çarpma önlenerek hareket sağlanmaktadır. Bu bölümde her iki yöntemin birlikte kullanılması ile hedefe ulaşılması anlatılacaktır. A* ile hedefe en kısa yoldan ulaşmak için rota planlanmakta ve bu rota izlenirken haritada olmayan olası engellerden sakınmak için lazer sensörden yararlanılmaktadır.

A* rota hesabında sistemdeki haritayı daha küçük karelere bölerek kullanmaktadır. Örneğin Şekil 3.3'de 14mx15m lik bir harita 1mx1m'lik karelere ayrılmıştır. Her bir bölge içinde engel olup olmama durumuna göre 1 ve 0 ile temsil edilmektedir. Burada mavi bölge bir robotu kırmızı bölge ise robotun gideceği noktayı göstermektedir. A* algoritması robotun sadece sekiz yönde (Şekil 3.1) gidebileceğini düşünerek rota planlaması yapmaktadır.



Şekil 3.1. A* için robot gidiş yönleri.

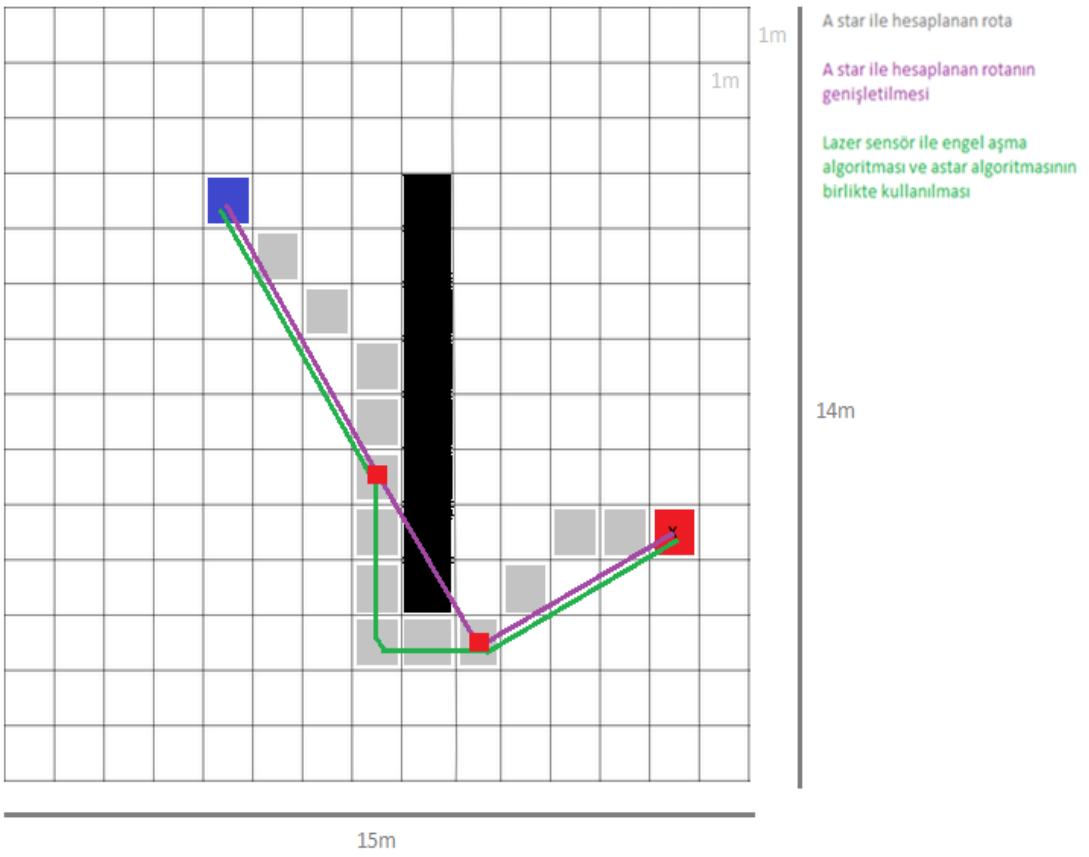
A* algoritmasında asıl fayda robotun yanlış istikametlere gitmesini önlemektir. Kapalı alanları harita üzerinden tespit edip açık yolları izleyerek hedefe daha kısa bir zamanda ulaşmak amaçlanmaktadır (Şekil 3.2).



Şekil 3.2. A* ile rota hesaplamanın avantajı.

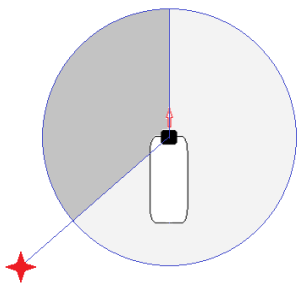
Şekil 3.3’ de gri ile gösterilen rota A* ile hesaplanmış olan bir rotadır. Bu rota robota hedefe ulaşmak için bir yol göstermektedir ancak bu yol gerçek uygulama noktasında gerçekçi değildir. A* ile hesaplanmış rotada her bir adım ana hedefe ulaşmak için bir ara hedefi göstermektedir. Bu kadar kısa hedeflerle ilerlemek robotun çalışma performansını olumsuz etkilemekte ve robotun sadece sekiz yönde gideceğini varsaymaktadır. Hâlbuki gerçek uygulamada robot 360 derecelik her yönde gidiş yapabilmektedir.

Robotun hedefe doğru daha stabil bir şekilde ilerleyebilmesi için bu rotaya bağlı kalınarak ara hedefler arasındaki mesafe arttırılmıştır. Şekil 3.3’ de gri hedef üzerinde kırmızı işaretler bulunan kareler yeni ara hedefleri ve mor çizgi yeni rotayı göstermektedir. Bu rotayı robotun takip etmesi durumunda robot engele çarpacaktır. Bu çarpmanın da önlenebilmesi için lazer sensör devreye girmekte ve engelin de aşılarak rotaya geri yönelmesi sağlanmaktadır. Şekil 3.3’ de yeşil olarak gösterilen rota yeni rotayı temsil etmektedir.



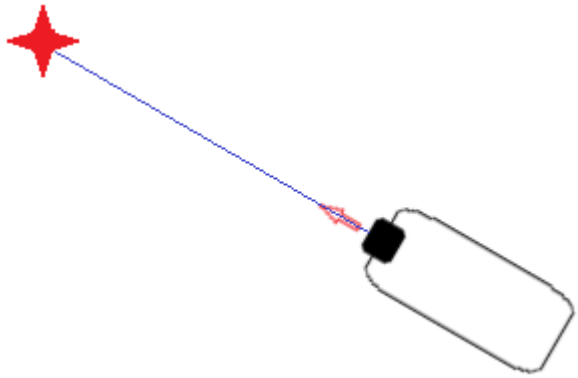
Şekil 3.3. A * için haritanın oluşturulması ve rota planlamaları.

Robotun hedef noktaya doğru ilerleyebilmesi için ilk olarak robotun baş açısını hedefe doğru yöneltmesi ve sonrasında hedefe doğru ilerlemesi gerekmektedir. Hedefe yönelmek için robotun baş açısı ile hedef nokta arasındaki açı farkı bulunarak robotun sağa veya sola doğru kendi etrafında dönerek baş açısını değiştirmesi gerekmektedir. Hangi yönden döneceğine karar verirken kısa olan yönü tercih edebilmelidir.



Şekil 3.4. Robotun hedefe dönüş yönünü belirlemesi.

Robot hedefe yönelimi sağladıktan sonra hedef ile arasındaki mesafeyi kontrol ederek hedefe doğru ilerlemeli ve hedefe yaklaştıkça hızını azaltarak hedef noktaya geldiğinde durmalıdır.



Şekil 3.5. Robotun hedefe doğru ilerlemesi.

Önce ara hedeflere doğru tek tek ulaşan robot ana hedefe kadar bu yöntemle yol alırken engel kontrolünü yapmaktadır ve engel aşma yöntemine göre hedefe ilerlemektedir.

BÖLÜM 4

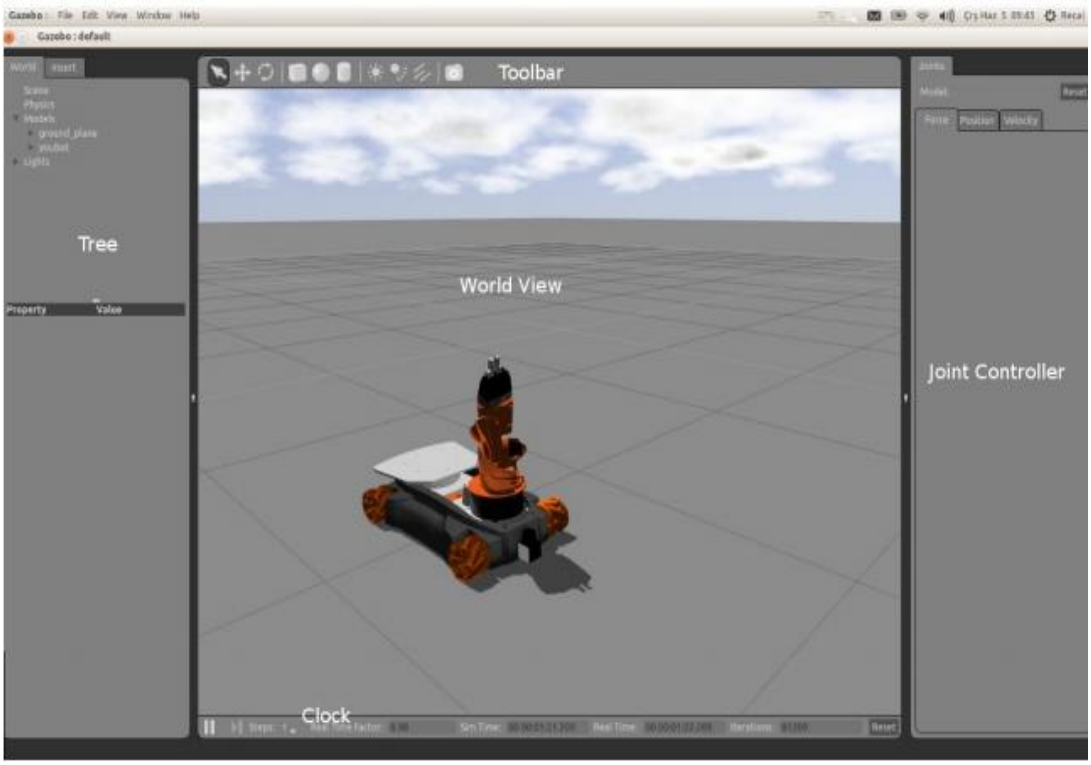
GAZEBO ÇOKLU ROBOT SİMÜLATÖRÜ

Robotik teknolojisi, mekanik, elektronik ve yazılımın bir arada bulunduğu bir teknolojidir. Bu nedenle bir robot geliştirilirken bütün çalışmaların gerçek ortamda yapılması zaman, enerji ve para bakımından maliyetli olmaktadır. Bu bağlamda işlerin daha hızlı ve maliyetsiz olarak ilerleyebilmesi için bir simülatör önem kazanmaktadır [36].

Gazebo, hem kapalı hem de açık mekanlar için geliştirilmiş açık kaynak kodlu bir çoklu robot simülatörüdür. Birden fazla robot, sensör ve cisimleri üç boyutlu ortamda simüle edebilme yeteneğine sahiptir [36].

Gazebo açık kaynak kodlu bir yazılımdır ve Ubuntu Linux üzerinde geliştirilmeye başlanmıştır. Bu çalışmanın yapıldığı tarihte Gazebonun üzerinde çalıştığı en iyi işletim sistemi Ubuntu 12.04' tür. Gazeboyu kullanırken simüle ettiğiniz ortama ve robotlara bağlı olarak işlemci (CPU) gücü ihtiyacı değişecektir. Ancak mümkün olduğunca güçlü bir işlemciye sahip olmanız önerilmektedir [36].

Gazebo sahneleme işlemi için açık kaynak kodlu bir grafik motoru olan OGRE' yi (Object-Oriented Graphics Rendering Engine) kullanmaktadır. Bu sebepten dolayı işletim sisteminiz ile iyi konfigüre edilmiş bir grafik kartına sahip olmanız gerekmektedir. En iyi performans NVIDIA GPU' lar ile elde edilmiştir. Grafik kullanıcı ara yüzü, simülasyonun gösterilmesini ve beş bileşeni aracılığıyla kullanıcı ile simülasyon arasında etkileşimi sağlamaktadır [36].



Şekil 4.1. Grafik kullanıcı ara yüzü.

World View: Oluşturulan ortamın ve modellerin görüntülendiği alandır. Bu alana model ekleyebilir, kaldırabilir veya manipüle edebilirsiniz [36].

Toolbar: Bu araç çubuğunu kullanarak sahneye yakınlaşıp uzaklaşabilir, bir modeli sahne üzerinde taşıyabilir, döndürebilir, kamera açısını ve pozisyonunu değiştirebilir, küp, silindir, küre ve ışık kaynağı ekleyebilirsiniz [36].

Tree: Bu liste sahnedeki modellerin hiyerarşik olarak görüntülenmesini sağlar. Ayrıca sahneye yeni model eklemek için de kullanılır [36].

Clock: Simülasyonu durdurmak, başlatmak için kullanılır. Ayrıca simülasyonun gerçek zamanlı veya gerçek zamana göre ne kadar yavaş çalıştığını görüntüler [36].

Joint Controller: Bu bileşen sahnedeki modellerin eklemlerine kuvvet uygulamak, hız kazandırmak veya açısını değiştirmek için kullanılır [36].

Gazebo bir simülasyon ortamı veya bir model hakkındaki bilgileri yüklemek veya saklamak için XML kullanır. Gazebonun geliştiricileri XML içeren SDF adındaki kendi formatlarını tanımlamışlardır. Bu format ile ilgili ayrıntılı bilgi http://gazebosim.org/sdf.html adresinden alınabilir. Temel olarak Gazebo için bir model, bir robotun veya çevresinin özelliklerinin tanımlanmasıdır. Bir model basit bir şekil olabileceği gibi çok kompleks bir robot da olabilir [36].

4.1. GAZEBO KURULUMU

Gazeboyu Ubuntu işletim sisteminize apt-get kullanarak kurmak için aşağıdaki adımlar takip edilmelidir [36].

1. Bilgisayarınızı packages.osrfoundation.org adresinden yazılım kabul etmesi için ayarlamalısınız.

.Ubuntu Linux 12.04 (precise)

$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main" > /etc/apt/sources.list.d/ros-latest.list'

$ sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu precise main" >/etc/apt/sources.list.d/gazebo-latest.list'

.Ubuntu Linux 12.10 (quantal)

$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu quantal main" > /etc/apt/sources.list.d/ros-latest.list'

$ sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu quantal main" >/etc/apt/sources.list.d/gazebo-latest.list'

2. ROS ve DRC (Darpa Robotics Challenge) depolarını güvenilir olarak işaretlemek için aşağıdaki komutları çalıştırmalısınız.

$ wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
$ wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add –

3. apt-get aracınızı güncelleştirerek Gazeboyu kurmak için aşağıdaki komutları çalıştırmalısınız.

$ sudo apt-get update
$ sudo apt-get install gazebo

4. Kütüphanelerin erişilebilir olması için Gazeboyu başlatmadan önce aşağıdaki komutu çalıştırmalısınız.

$. /usr/share/gazebo/setup.sh

Bu işlemi her seferinde yapmamak için terminal açıldığında otomatik olarak çalıştırılan .bashrc betiğine aşağıdaki komutu çalıştırarak ekleyebilirsiniz.

$ echo "source /usr/share/gazebo/setup.sh" >> ~/.bashrc
$ source ~/.bashrc

5. Gazeboyu başlatmak için terminal ekranından adını yazarak çağırmanız yeterli.

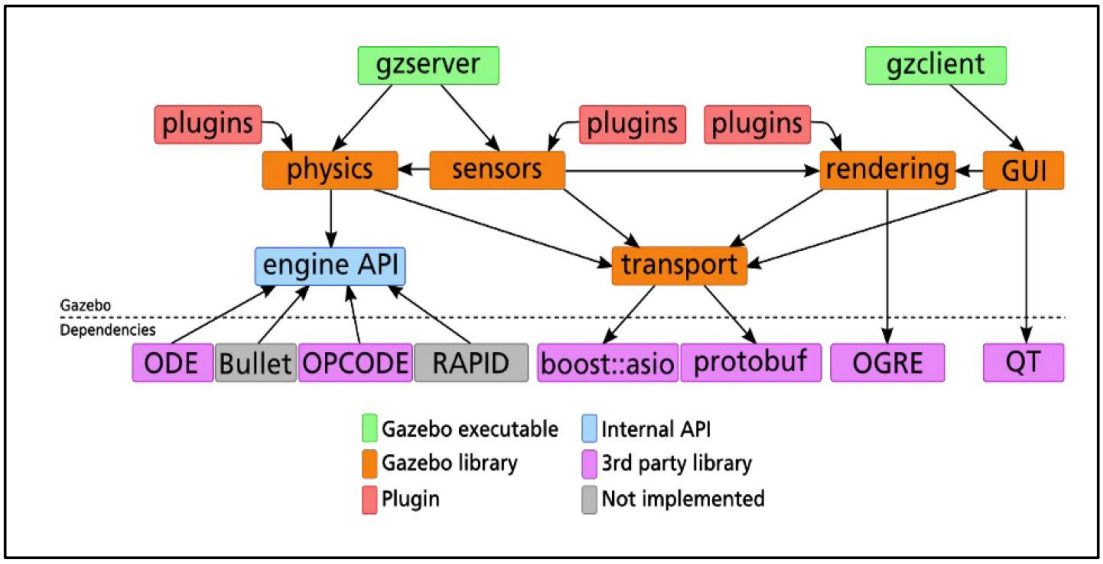
$ gazebo

4.2. GAZEBONUN MİMARİSİ

Gazebo çeşitli kütüphanelere ayrılmıştır.
· Fizik : Simülasyonun fiziksel durumunu günceller
· Sahneleme : Simülasyon durumunu görselleştirir.
· Sensör : Sensör verilerini oluşturur.
· Taşıma : İşlemler arası iletişimi yönetir.
·Grafik Kullanıcı Arayüzü (GUI) : Simülasyonun görselleştirilmesi ve manipülasyonunu sağlar [36].

Bu kütüphaneler iki ana işlem tarafından kullanılmaktadır [36].

- Sunucu (gzserver) : Fizik döngülerini çalıştırır ve sensör verilerini üretir.
- İstemci (gzclient) : Kullanıcı etkileşimi ve simülasyonun görselleştirilmesini sağlar.



Şekil 4.2. Gazebonun mimarisi [38].

4.3. EKLENTİLER

Bir eklenti, paylaşımlı kütüphane olarak derlenen ve simülasyona eklenen bir C++ kodudur [36].

Eklentiler standart C++ sınıflarını kullanarak Gazebonun tüm fonksiyonlarına erişebilirler. Gazebo simülasyonunda program aracılığıyla ortama veya bir modele müdahale etmek, sensörlerden veri okumak istendiğinde eklenti kullanılmalıdır [36].

Dört tip eklenti vardır [36]:

- Model
- World
- Sensor
- System

Model eklentisi Gazebo içerisinde herhangi bir modele, World eklentisi ortama, Sensor eklentisi bir sensöre eklenir. System eklentisi ise komut satırı için özelleşmiş bir eklentidir. Bu nedenle istenilen fonksiyonelliğe göre uygun bir eklenti seçilmelidir. Eğer ortam özellikleri değiştirilecekse bir World eklentisi, bir modelin durumu hakkında bilgi alınacaksa, bir modelin eklemleri kontrol edilecekse Model eklentisi kullanılmalıdır [36].

4.4. MODEL

Temel olarak gazebo için bir model, bir robotun veya çevresinin özelliklerinin tanımlanmasıdır. Modeller SDF kullanılarak tanımlanmaktadır. Bir model basit bir şekil olabileceği gibi çok kompleks bir robot da olabilir. Modelleri kendiniz oluşturabilir veya çevrimiçi model veri tabanından alınan modelleri kullanabilirsiniz. Modeller hiyerarşik bir yapıda da kullanılabilir. Yani bir model başka bir modeli kendi içerisine dahil edebilir. Örneğin; yeni oluşturduğunuz bir robotunuza önceden tanımlanmış bir lazer mesafe sensörünü ekleyebilirsiniz [36].

4.4.1. Bir Modelin Bileşenleri

Bir model aşağıdaki bileşenlerde oluşur [36].

Link : Bir link, bir modelin bir gövdesinin fiziksel özelliklerini içerir. Her link çok sayıda Collision ve Visual bileşeni içerebilir. Ancak bir modeldeki link sayısı azaldıkça performans artacaktır. Örneğin; bir masayı birbirlerine Joint bileşenleri ile bağlanmış 4 ayak ve 1 yüzey olmak üzere 5 linkten oluşturabilirsiniz. Ancak, bu tasarım, ayaklar ile yüzeyi birleştiren eklemler hareket etmeyeceğinden gereğinden fazla kompleks olacaktır. Bu nedenle masayı 5 Collision bileşeni içeren bir linkten meydana getirmek daha doğru olacaktır [36].

Collision : Collision bileşeni çarpışma kontrolü için kullanılan bir geometriyi kapsar. Genelde az kaynak tüketmesi açısından basit bir şekil olması tercih edilir [36].

Visual: Visual bileşeni bir linkin parçalarını görselleştirmek için kullanılır. Bir link sıfır veya daha fazla visual bileşeni içerebilir [36].

Inertial: Inertial bileşeni bir linkin kütle, eylemsizlik matrisi gibi dinamik özelliklerinin tanımlandığı bileşendir [36].

Sensor: Bu bileşen, plugin bileşeninde kullanmak üzere ortamdan veri toplar [36].

Joint: Bir joint bileşeni iki linki birbirine bağlar [36].

Plugin : Bu bileşen simülasyon içerisinde modeli kontrol etmek üzere üçüncü parti olarak geliştirilmiş kütüphaneleri eklenti olarak kullanmayı sağlar [36].

4.5. MOBİL ROBOT

Bu bölümde iki tekerlekli bir mobil robot modeli oluşturacağız. Aşağıdaki kodu çalıştırarak bir klasör oluşturulur [36].

```
$ mkdir ~/.gazebo/models/my_robot
```

Aşağıdaki kodu çalıştırarak gedit metin editörü ile az önce oluşturduğunuz my_robot klasörü altında model.config dosyasını oluşturulur.

```
$ gedit ~/.gazebo/models/my_robot/model.config
```

Metin editörü açıldıktan sonra aşağıdaki kod yapıştırıp kaydedilir.

```
<?xml version="1.0"?>
<model>
  <name>My Robot</name>
  <version>1.0</version>
  <sdf version='1.3'>model.sdf</sdf>

  <author>
   <name>Recai Sinekli</name>
   <email>recai@recaisinekli.com</email>
  </author>

  <description>
    Yeni mobil robotum.
  </description>
</model>
```

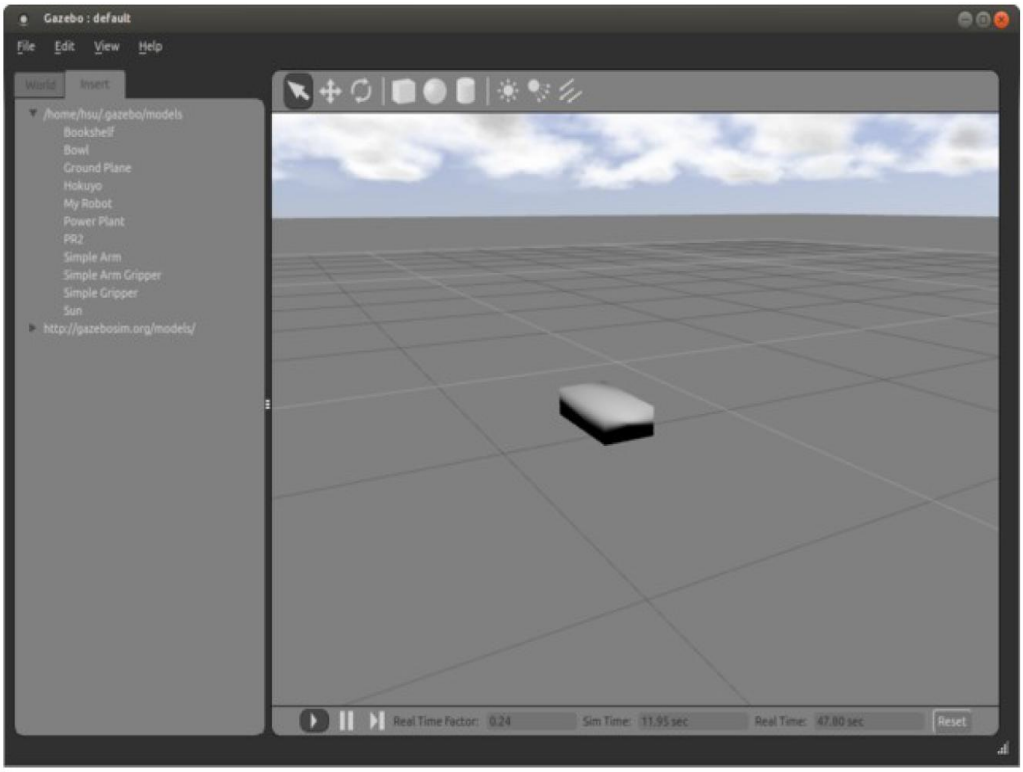
Aşağıdaki kod çalıştırarak model.sdf dosyasını oluşturulur.

$ gedit ~/.gazebo/models/my_robot/model.sdf

Metin editörü açıldıktan sonra aşağıdaki kod yapıştırıp kaydedilir.

```
<?xml version='1.0'?>
<sdf version='1.3'>
  <model name="my_robot">
    <static>true</static>
    <link name='chassis'>
        <pose>0 0 .1 0 0 0</pose>
        <collision name='collision'>
          <geometry>
            <box>
              <size>.4 .2 .1</size>
            </box>
          </geometry>
        </collision>
        <visual name='visual'>
          <geometry>
            <box>
              <size>.4 .2 .1</size>
            </box>
          </geometry>
        </visual>
      </link>
  </model>
</sdf>
```

Gazeboyu çalıştırıp 'My Robot' isimli modeli sahneye eklediğinizde aşağıdaki gibi görüntülenecektir.



Şekil 4.3. Gazeboda robot oluşturulması.

Daha sonra model.sdf dosyasının visual etiketinin </visual> şeklinde kapatıldığı satırın hemen altından devam ederek aşağıdaki kod yapıştırılır ve kaydedilir.

```
    <collision name='caster_collision'>
        <pose>-0.15 0 -0.05 0 0 0</pose>
        <geometry>
          <sphere>
           <radius>.05</radius>
          </sphere>
        </geometry>

        <surface>
         <friction>
          <ode>
            <mu>0</mu>
            <mu2>0</mu2>
            <slip1>1.0</slip1>
            <slip2>1.0</slip2>
          </ode>
         </friction>
        </surface>
    </collision>

    <visual name='caster_visual'>
      <pose>-0.15 0 -0.05 0 0 0</pose>
      <geometry>
        <sphere>
          <radius>.05</radius>
        </sphere>
      </geometry>
    </visual>
```

Böylece mobil robotu hemen ön tarafına küre şeklinde bir tekerlek eklemiş olduk. Daha sonra model.sdf dosyasının link etiketinin </link> şeklinde kapatıldığı satırın hemen altına aşağıdaki kod eklenerek kaydedilir.

```
<link name="left_wheel">
  <pose>0.1 0.13 0.1 0 1.5707 1.5707</pose>
  <collision name="collision">
    <geometry>
      <cylinder>
        <radius>.1</radius>
        <length>.05</length>
      </cylinder>
    </geometry>
  </collision>
  <visual name="visual">
    <geometry>
      <cylinder>
        <radius>.1</radius>
        <length>.05</length>
      </cylinder>
    </geometry>
  </visual>
</link>
```

Böylece mobil robotunuza sol tekerleğini eklemiş olduk. Sol tekerleğin link etiketinin </link>şeklinde kapatıldığı satırın hemen altına aşağıdaki kod eklenerek kaydedilir.

```
<link name="right_wheel">
  <pose>0.1 -0.13 0.1 0 1.5707 1.5707</pose>
  <collision name="collision">
    <geometry>
      <cylinder>
        <radius>.1</radius>
        <length>.05</length>
      </cylinder>
    </geometry>
  </collision>
  <visual name="visual">
    <geometry>
      <cylinder>
        <radius>.1</radius>
        <length>.05</length>
      </cylinder>
    </geometry>
  </visual>
</link>
```

Böylece mobil robotunuza sağ tekerleğini eklemiş olduk. Daha sonra static etiketinin değerini false yapılır ve kaydedilir.

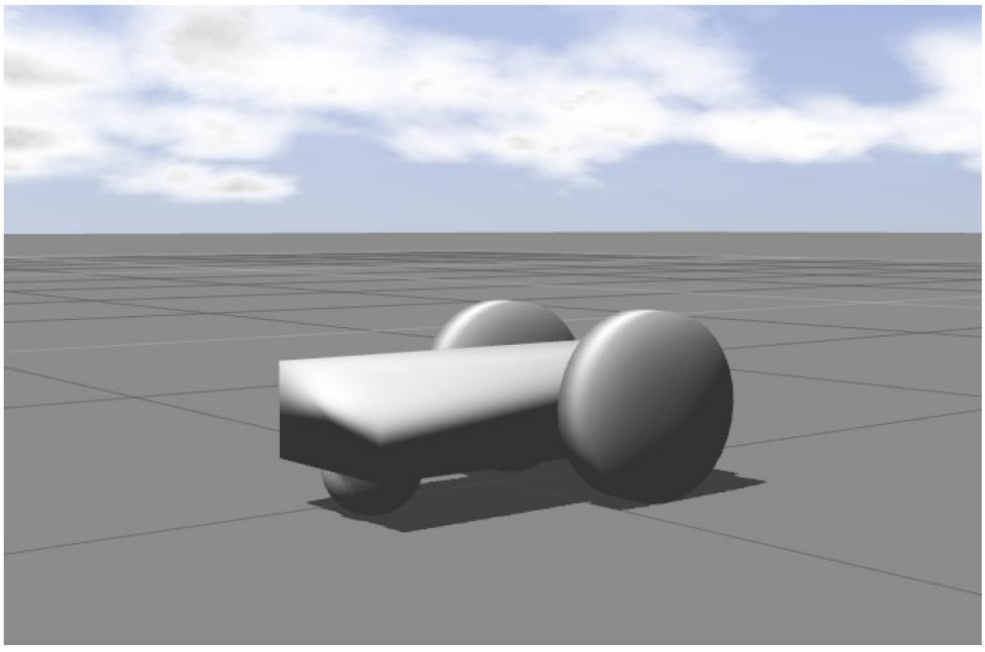
Son olarak sağ ve sol tekerleklerin gövdeye eklemler aracılığıyla bağlanması için aşağıdaki kodu az önce eklediğiniz sağ tekerleğin link etiketinin </link> şeklinde kapatıldığı satırın hemen altına yapıştırılır ve kaydedilir.

```
<joint type="revolute" name="left_wheel_hinge">
   <pose>0 0 -0.03 0 0 0</pose>
   <child>left_wheel</child>
   <parent>chassis</parent>
   <axis>
     <xyz>0 1 0</xyz>
   </axis>
 </joint>

 <joint type="revolute" name="right_wheel_hinge">
   <pose>0 0 0.03 0 0 0</pose>
   <child>right_wheel</child>
   <parent>chassis</parent>
   <axis>
     <xyz>0 1 0</xyz>
   </axis>
 </joint>
```

Böylece her iki tekerleği de y ekseninde dönebilecek şekilde gövdeye bağlamış olduk.

Not : Gazebo sağ el koordinat sistemini kullanır. +Z'in düşey, +X'in ekranın içerisine doğru ve +Y'nin ise sola doğru olduğu bir koordinat sistemidir.



Şekil 4.4. Gazebo ortamında oluşturulmuş mobil robot [38].

Mobil robotu sahneye ekledikten sonra 'Joint Controller' kullanılarak pencereden sağ ve sol tekerleklere kuvvet uygulayarak mobil robotun hareketini gözlemlenebilir [36].

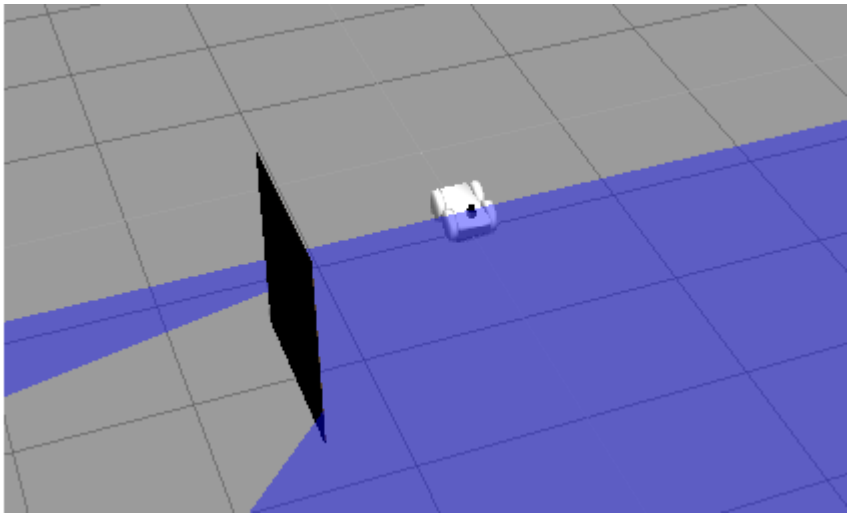
BÖLÜM 5

ROBOTA GÖRE ENGEL DURUMLARININ SİMULASYON ORTAMINDA İNCELENMESİ

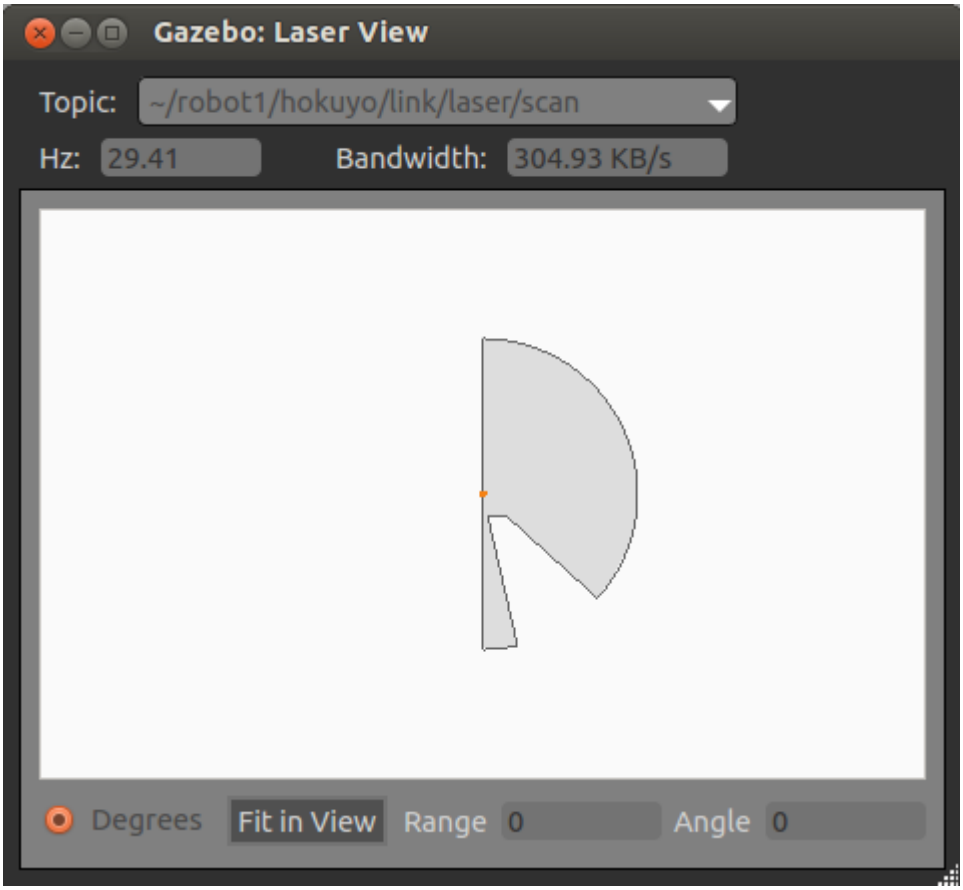
Robotun hareketi boyunca engel çok farklı açılarda karşısına çıkabilir. Bu bölümde robotun ne gibi engel durumlarıyla karşılaşabileceği ve lazer sensörün ortamdan topladığı veriler incelenmiştir.

5.1. ROBOTUN SAĞINDA ENGEL OLMASI

Engel robotun sağında iken robotun geçebileceği kadar uzağında olduğunda, hareket yönünde bir değişiklik yapılmasına gerek yoktur.



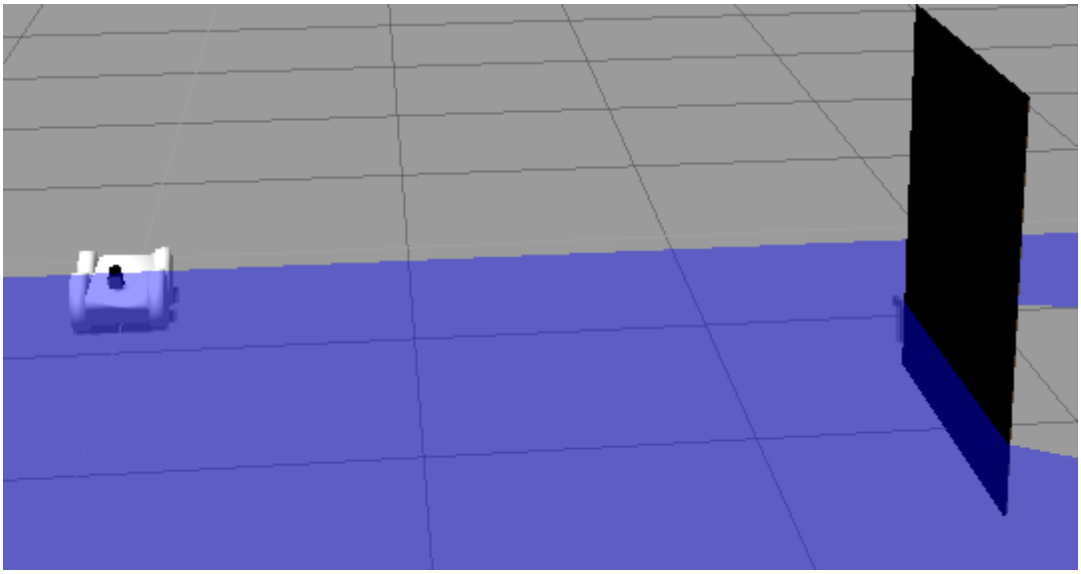
Şekil 5.1. Robotun sağında engel olması.



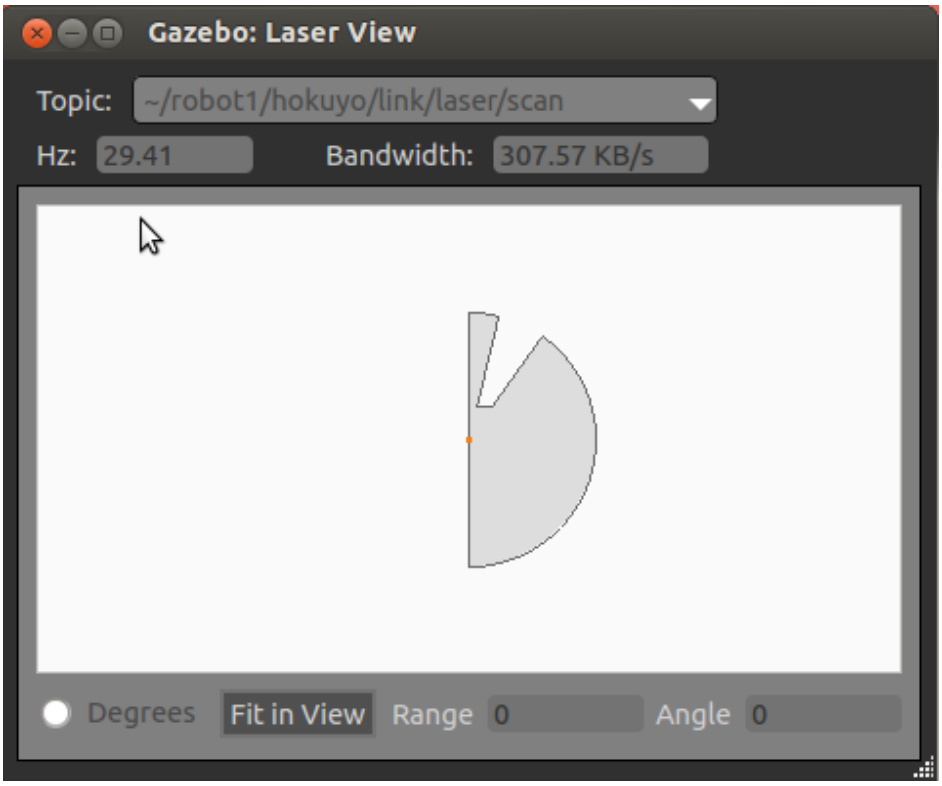
Şekil 5.2. Engel robotun sağında iken lazer sensörün ortamdan aldığı veri.

5.2. ROBOTUN SOLUNDA ENGEL OLMASI

Engel robotun solunda iken robotun geçebileceği kadar uzağında olduğunda, hareket yönünde bir değişiklik yapılmasına gerek yoktur.



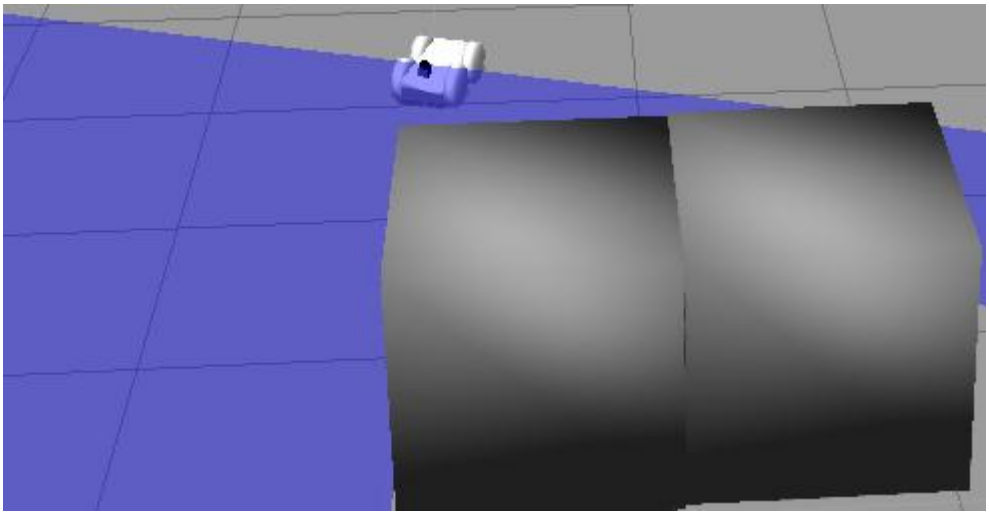
Şekil 5.3. Robotun solunda engel olması.



Şekil 5.4. Engel robotun solunda iken lazer sensörün ortamdan aldığı veri.

5.3. ROBOTUN KARŞISINDA ENGEL OLMASI

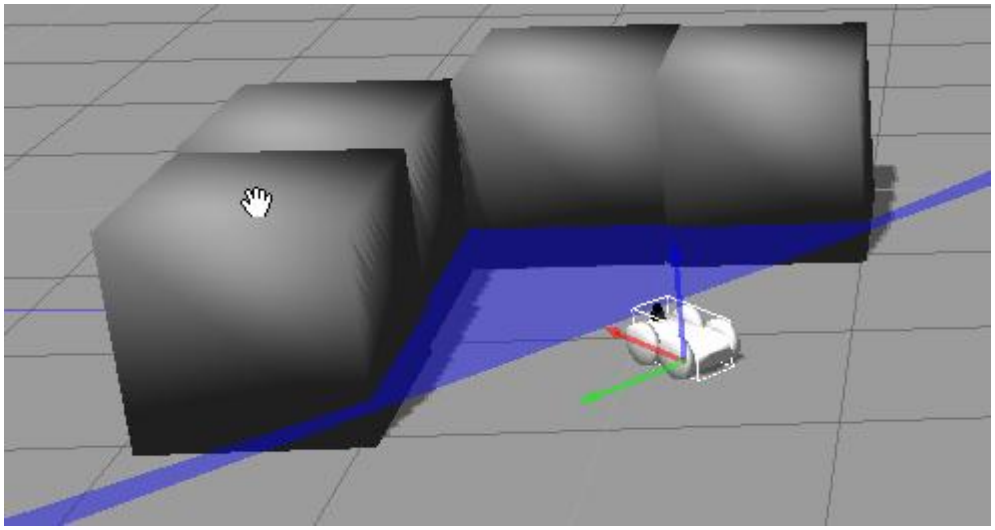
Engel robotun karşısında olduğu durumda robotun yönünü değiştirmesi gerekmektedir. Engel tam olarak karşısında iken sağa veya sola dönmesi sağlanmalıdır. Bunun tercihi bize kalmaktadır. Engel robotun karşısında fakat sağ tarafta daha geniş yer kaplıyorsa robotun sol tarafa yönelmesi ya da engel sol tarafta daha geniş yer kaplıyorsa robotun sağ tarafa yönelmesi daha doğru olacaktır.



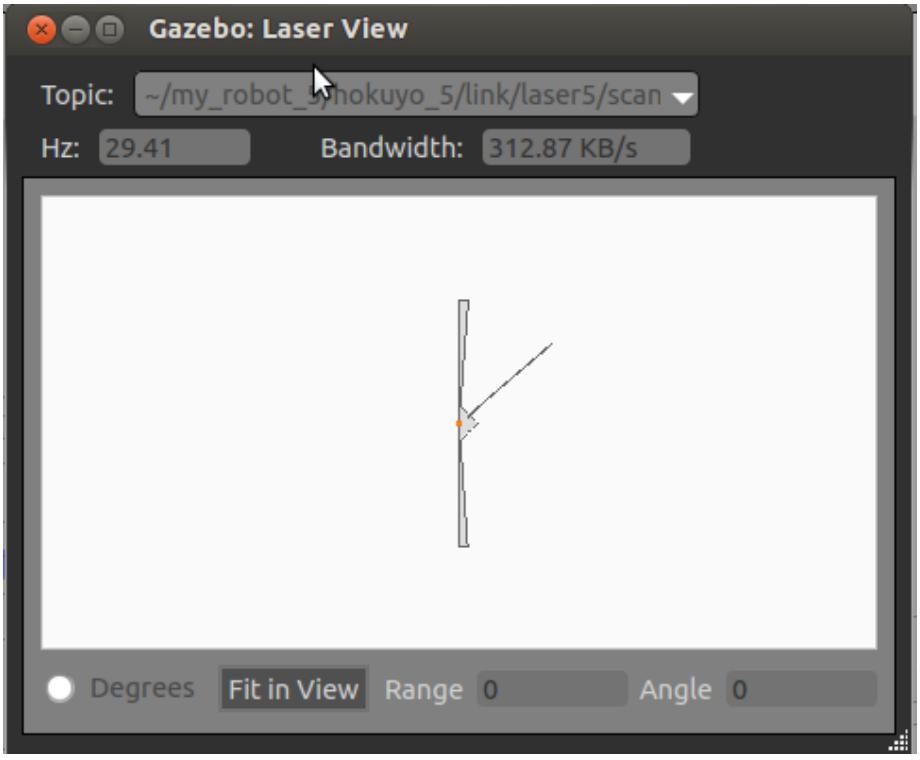
Şekil 5.5. Robotun karşısında engel olması.

5.4. ROBOTUN KÖŞEYE GELDİĞİ DURUMLAR

Robot köşeye geldiği durumlarda sağda ve solda eşit büyüklükte engel bulunduğu için ne yöne gideceği konusunda karar verememe sorunu oluşmaktadır. Bu probleme karşı robotun sağ tarafa yönelmesi tercih edilmiş ve köşeden çıkışı sağlanmıştır.



Şekil 5.6. Robotun köşeye geldiği durumlar.



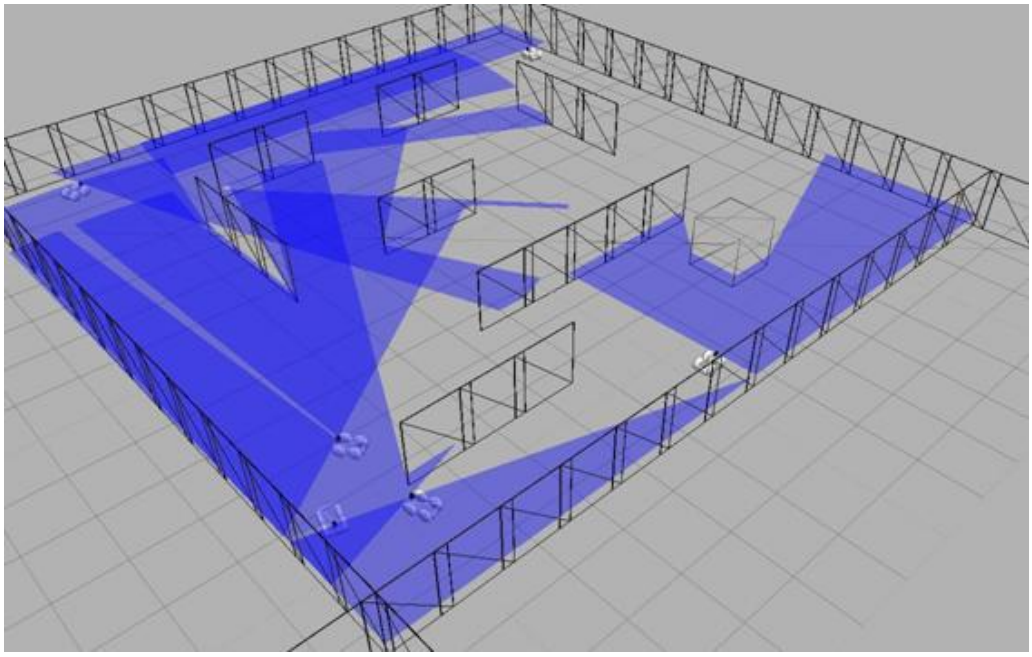
Şekil 5.7. Robotun köşeye geldiği durumda lazer sensorun ortamdan aldığı veri.

5.5. SIMULASYON ORTAMINA BİRDEN FAZLA ROBOT EKLEMEK

Sabit engellerin bulunduğu bir alanda birden fazla mobil robot oluşturulmuş ve alana bırakılmıştır. Böylece her bir robot bir diğeri için karşısında hareketli bir engel

durumundadır. Birbirlerine çarpışmadan yoluna devam edip edemeyeceği incelenecektir. Her bir robot aynı özelliklerde ve engel karşısında aynı tepkilere sahiptir.

Robotların arkadan veya yandan geldiği durumlarda çarpışma durumları gözlemlenmiştir. Karşıdan gelen hareketli engellere ve karşısına çıkan sabit engellere çarpmadan boş olan yöne doğru yönelebilmeleri geliştirilmiştir.



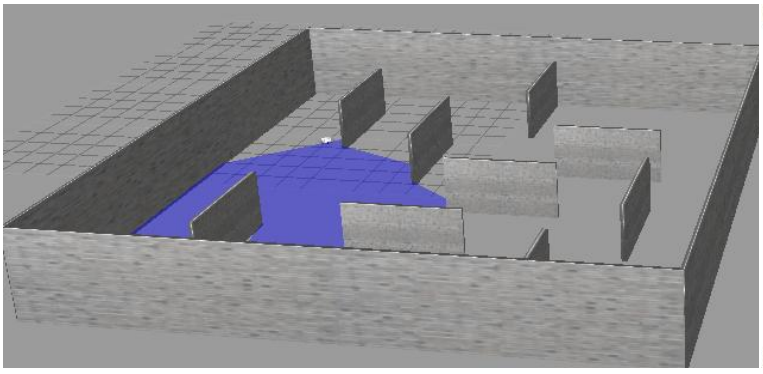
Şekil 5.8. Sabit engellerin bulunduğu bir ortamda birden fazla robotun hareket etmesi.

BÖLÜM 6

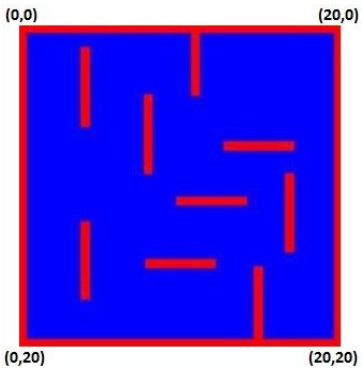
SONUÇLAR VE ÖNERİLER

A* yol planlama yöntemi yapılırken ortamın genel yapısının (ziyaret edilecek noktaların koordinatları ve bunlar arasındaki bağlantı durumlarının) önceden bilindiği varsayılmaktadır. A* algoritması, ortamın bilinen bu genel özellikleriyle gezgin robot için genel bir rota planlanmasında kullanılmıştır. Başlangıç ve hedef noktaları arasındaki bağlantıları kullanarak, robotun takip etmesi gereken yolu planlamaktadır. Bu yol planı robota verilmekte ve robot ilerlemektedir. Robotun hareketi sırasında lazer tarayıcı çevrede engel olup olmadığını algılamakta, eğer robotun yolu üzerinde bir engel bulunursa geliştirilen algoritma hesaplamaları kullanılarak robotun engelden sakınması sağlanmaktadır.

Robotun gezebileceği harita bir matriste tutulmaktadır. Bu ortam matrisinde engellerin bulunduğu yerler 1 (kırmızı) ile diğer alanlar 0 (mavi) ile temsil edilmektedir (Şekil 6.2). Önerilen yöntem gezgin robotlar için Gazebo benzetim ortamında uygulanmıştır.

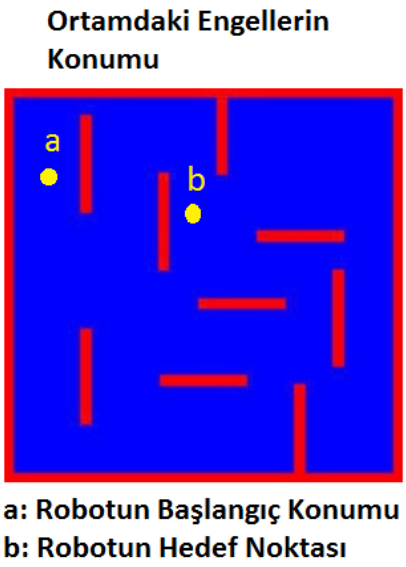


Şekil 6.1. Benzetimlerin gerçekleştiği Gazebo ortamı.

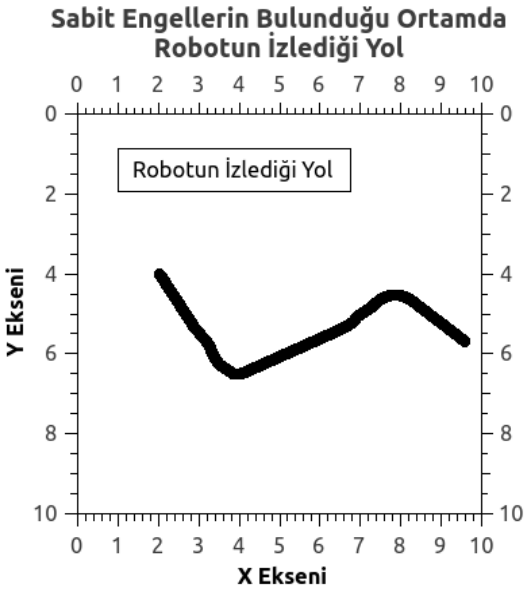


Şekil 6.2. Benzetimlerin gerçekleştiği ortamın harita görünümü.

Robotun sabit engeller arasından geçerken izlediği yol bilgisi kaydedilmiş ve bu bilgi harita üzerinde çizdirilmiştir (Şekil 6.3, Şekil 6.4). Şekilde görüldüğü gibi robot giderken engelle karşılaşmakta ve bu engeli aşmak için aşağı yönelmekte, engelden uzaklaşınca asıl rotasına dönmektedir.



Şekil 6.3. Robotun hedef ve başlangıç noktaları.

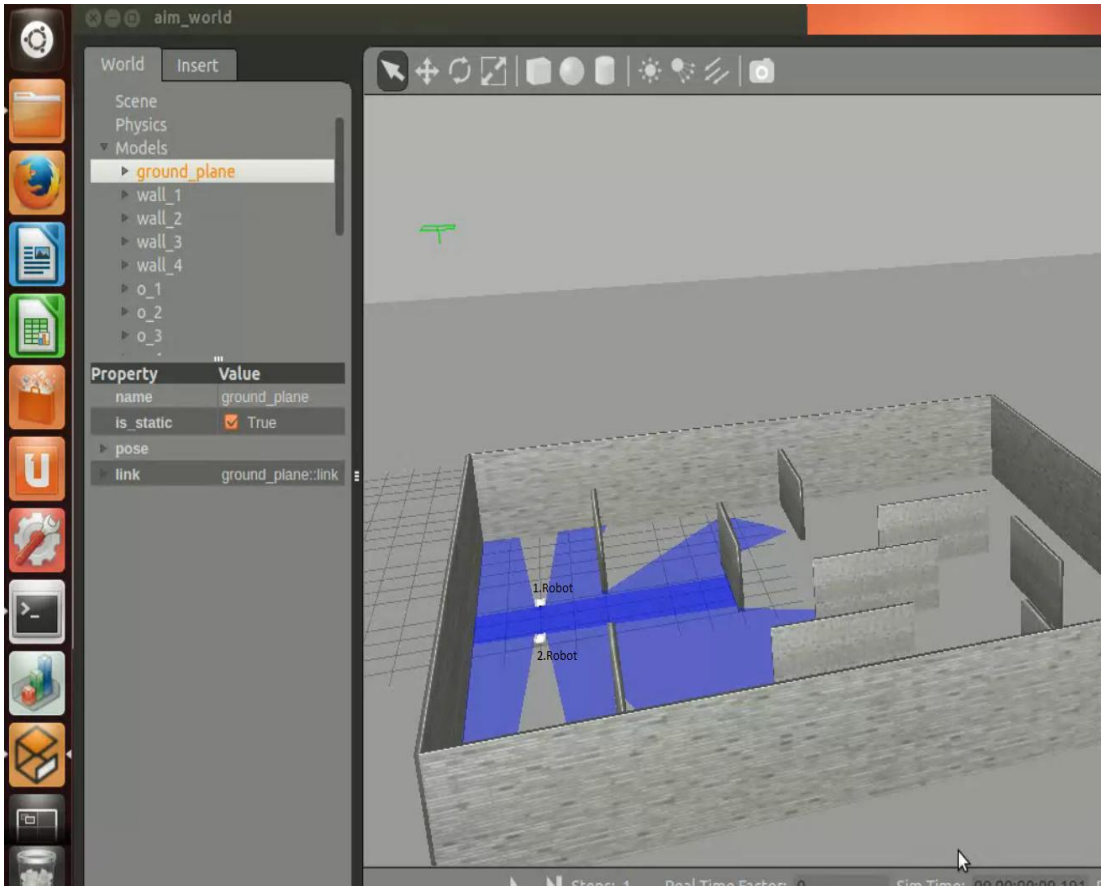


Şekil 6.4. Robotun izlediği yol.

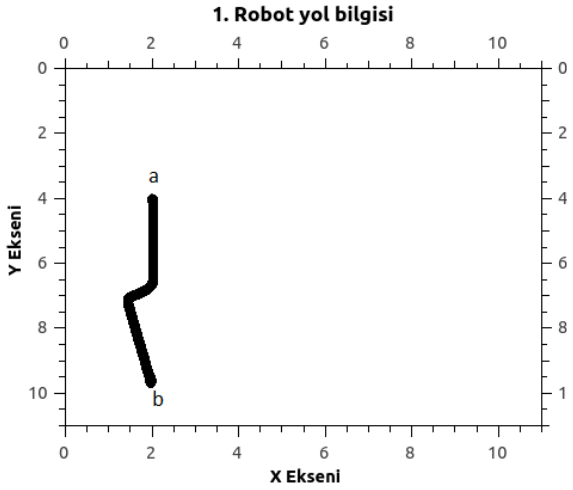
Ortama birden fazla robot eklendiğinde ise robotlar birbirlerine göre dinamik engel olmaktadır. A* yol planlama yöntemi ile rota planlamasına göre hareket eden birden fazla robot aynı ortamda hareket ederken çarpışmalar olabilmektedir. Geliştirilen yöntem ile lazer sensor yardımıyla hedef noktaya giderken karşılaşılan dinamik engellerin aşılması sağlanmıştır.

Şekil 6.5' de a ve b noktalarına 2 robot yerleştirilmiştir. a noktasındaki robotun b noktasına, b noktasındaki robotun a noktasına rota planlaması yapılmıştır. Arada hiçbir engel olmadığı için A* algoritmasına göre dümdüz istenilen hedef noktalarına gitmesi beklenmektedir. Bu durumda karşı karşıya hareket eden iki robot çarpışacaktır. Geliştirilen yöntemde ise robotlar birbirlerine çarpmadan hedef noktalarına ulaşmaktadır. 1. Robot a noktasından b noktasına doğru gitmektedir, 2. robot ise b noktasından a noktasına gitmektedir. Her iki robot da karşısındaki engeli

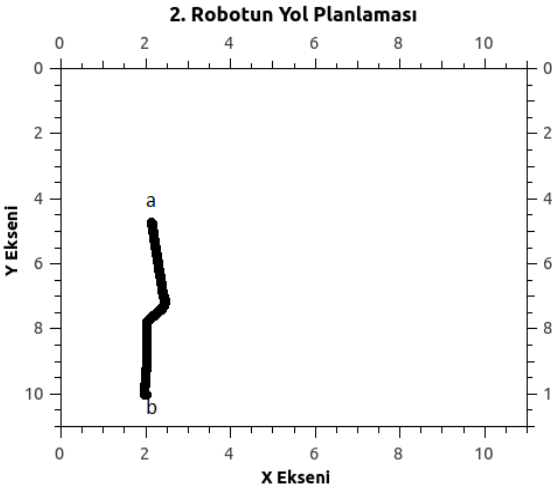
algıladığında sağa doğru engeli aşmak için yönelmektedir ve birbirlerini geçince hedef noktalarına doğru tekrar yönelmektedirler (Şekil 6.6, Şekil 6.7).



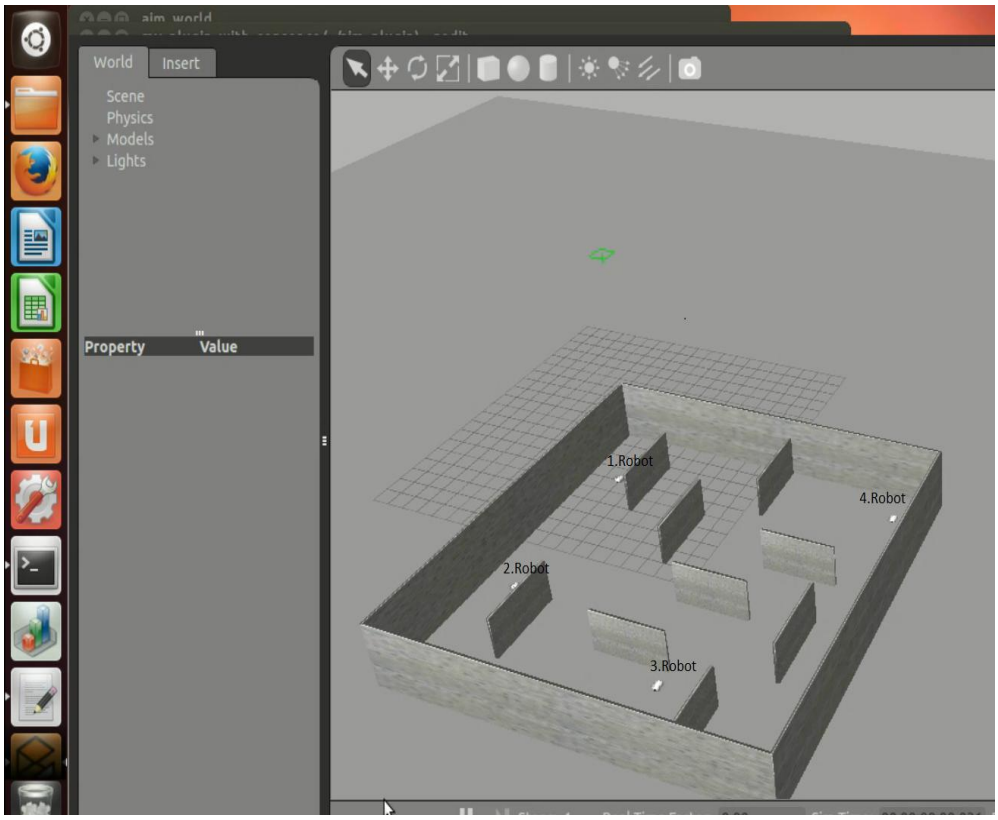
Şekil 6.5. Karşı karşıya hareket eden iki robot.



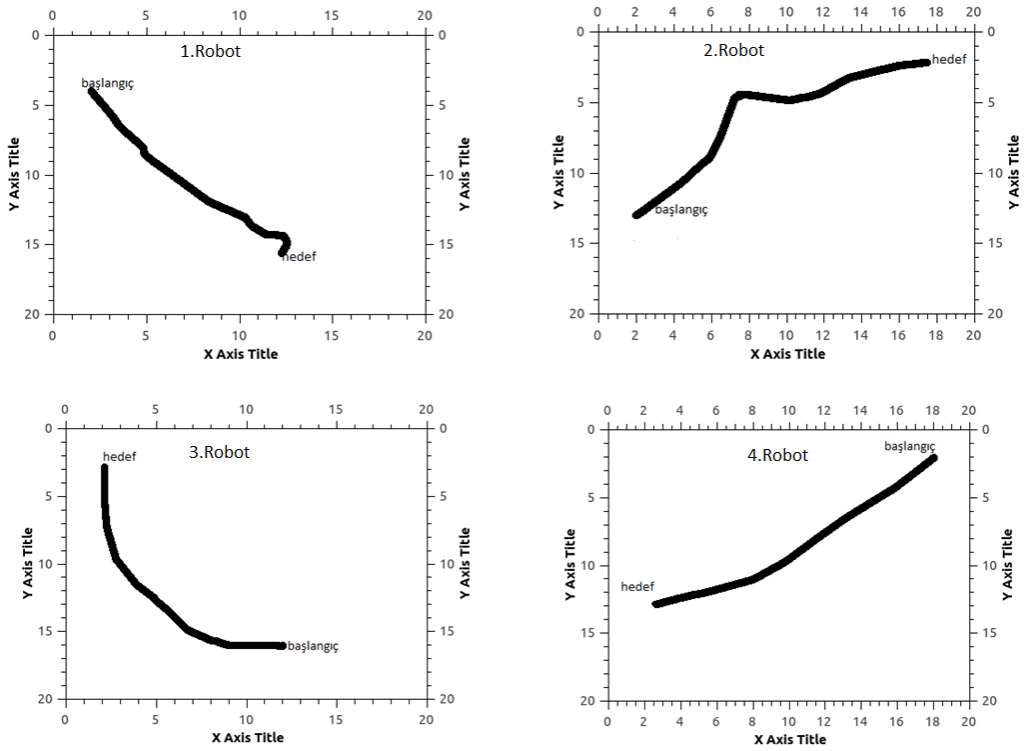
Şekil 6.6. Karşısında engel olan 1. robotun yol planlaması.



Şekil 6.7. Karşısında engel olan 2. robotun yol planlaması.



Şekil 6.8. Statik ve dinamik engellerin olduğu ortam.



Şekil 6.9. Şekil 6.8' de ki robotların izlediği rotalar.

Bu çalışmada otonom bir robotun simulasyon ortamında gezinebilmesi için bir yöntem geliştirilmiştir. Robotun ziyaret edeceği noktalar ve bu noktalar arasındaki bağlantı ve mesafeler bilindiğinde, A* yöntemi kullanılarak robotun izleyeceği yol planlanmıştır. Robotun planlanan yolu takip etmesi ve yolda çıkabilecek statik ve dinamik engelleri aşması için lazer sensör kullanılarak robotun hareketi sağlanmıştır. Robotun gideceği hedef noktada bir dinamik engelin olduğu durumda A* yeterli olmamaktadır. Ancak geliştirilen yeni yöntem bu problemi ortadan kaldırmaktadır. Bu çalışmada simulasyon ortamında A* algoritmasının dinamik engeller karşısında etkisiz olduğu durumun, lazer sensör kullanılarak aşılabildiği gösterilmiştir.

KAYNAKLAR

1. Ishida, T., “Moving target search with intelligence”, ***In Proceedings of the 10th National Conference on Artificial Intelligence***, 525-532 (1992).

2. Korf, R. E., "Real-time heuristic search", ***Artificial Intelligence***, 42(2): 189-211 (1990).

3. Galceran, Y. E. and Carreras, P. M., “A survey on coverage path planning for robotics”, ***©Robotics and Autonomous Systems***, 61 (12): 1258-1276 (2013).

4. Keshmiri, S. and Payandeh, S., “An overview of mobile robotic agents motion planning in dynamic environments”, ***In Proceedings of the 14th IASTED International Conference,*** Robotics and Applications (RA20), MA, Boston, 152-159 (2009).

5. Tang, S., Khaksar, W., Ismail, N. and Ariffin, M., “A review on robot motion planning approaches”, ***Pertanika Journal of Science and Technology***, 20(1): 15-29 (2012).

6. Goerzen, C., Kong, Z. and Mettler, B., “A survey of motion planning algorithms from the perspective of autonomous uav guidance”, ***Journal of Intelligent and Robotic Systems***, 57 (1-4): 65-100 (2010).

7. Nilsson, N. J., “Problem-solving methods in artificial ıntelligence.”, ***McGraw-Hill Publishing Co***., 100-150 (1971).

8. Likhachev, M. and Ferguson, D., “Planning long dynamically feasible maneuvers for autonomous vehicles”, ***International Journal of Robotic Research***, 28: 933–945 (2009).

9. Hart, P. E., Nilsson, N. J. and Raphael, B., “A formal basis for the heuristic determination of minimum cost paths”, ***IEEE Transactions on Systems Science and Cybernetics***, 4 (2): 100–107 (1968).

10. Pohl, I., “Heuristic search viewed as path finding in a graph”, ***Artificial Intelligence,*** 1(3-4): 193–204 (1970).

11. Koenig, S., Likhachev, M., Liu, Y. and Furcy, D., “Incremental heuristic search in artificial intelligence”, ***Artificial Intelligence Magazine***, 25 (2): 99–112 (2004).

12. Koenig, S. and Sun, X., "Comparing real-time and incremental heuristic search for real-time situated agents", ***Autonomous Agents and Multi-Agent Systems*** 18 (3): 313-341 (2009).

13. Sun, X. and Koenig, S., “The fringe-saving A*search algorithm - a feasibility study”, ***In Proceedings of the International Joint Conference on Artificial Intelligence***, 2391–2397 (2007).

14. Koenig, S. and Likhachev, M., “A new principle for incremental heuristic search: Theoretical results”, ***In Proceedings of the International Conference on Autonomous Planning and Scheduling***, 402–405 (2006).

15. Trovato, K., “Differential A*: An adaptive search method illustrated with robot path planning for moving obstacles and goals, and an uncertain environment”, ***Journal of Pattern Recognition and Artificial Intelligence***, 4 (2): 245–268 (1990).

16. Stentz, A., “The focussed d* algorithm for real-time replanning”, ***In Proceedings of the International Joint Conference on Artificial Intelligence***, 1652–1659 (1995).

17. Koenig, S. and Likhachev, M., “D* Lite”, ***In Proceedings of the National Conference on Artificial Intelligence***, 476–483 (2002).

18. Hebert, M., McLachlan, R. and Chang, P., “Experiments with driving modes forurban robots”, ***In Proceedings of the SPIE Mobile Robots***, doi: 10.1117/12.369249 (1999).

19. Thayer, S., Digney, B., Diaz, M., Stentz, A., Nabbe, B. and Hebert, M., “Distributed robotic mapping of extreme environments”, ***In Proceedings of the SPIE: Mobile Robots XV and Telemanipulator and Telepresence Technologies VII***, 84–95 (2000).

20. Furcy, D. and Koenig, S., “Speeding up the convergence of real-time search”, ***In Proceedings of the 10th National Conference on Artificial Intelligence***, 530-535 (2000).

21. Hernandez, C. and Meseguer, P., “Improving real-time heuristic search on initially unknown maps”, ***Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), Workshop on Planning in Games***, Citeseer, 8 (2007).

22. Rayner, D. C., Davison, K., Bulitko, V., Anderson, K. and Lu, J., “Real-time heuristic search with a priority queue.”, ***International Joint Conferences on Artificial Intelligence (IJCAI)***, 2372-2377 (2007).

23. Sturtevant, N. R., Bulitko, V. and Björnsson, Y., “On learning in agentcentered search.”, ***Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems***: ***volume 1-Volume 1***, ***International Foundation for Autonomous Agents and Multiagent Systems***, 333-340 (2010).

24. Hernández, C. and Meseguer, P., "LRTA*(k). Proceedings of the 19th international joint conference on Artificial intelligence", ***Morgan Kaufmann Publishers Inc.***, 1238-1243 (2005).

25. Bond, D., Widger, N. A., Ruml, W. and Sun, X., "Real-time search in dynamic worlds", ***Proceedings of the Symposium on Combinatorial Search (SoCS-10)***, (2010).

26. Cannon, J., Rose, K. and Ruml, W., "Real-time motion planning with dynamic obstacles", ***Proceedings of the Symposium on Combinatorial Search (SoCS-12)***, (2012).

27. Hernández, C. and Baier, J. A., "Avoiding and escaping depressions in real-time heuristic search", ***J. Artif.Intell. Res.(JAIR)***, 43: 523- 570 (2012).

28. Likhachev, M., Gordon, G. and Thrun, S., "ARA*: Anytime A* with provable bounds on sub-optimality", ***Advances in Neural Information Processing Systems***, (2003).

29. Chiddarwar, S. S. and Babu, N. R., "Offline decoupled path planning approach for effective coordination of multiple robots" ***Robotica***, 28(4): 477-491 (2010).

30. Kopriva, S., Sislak, D., Pavlicek, D. and Pechoucek, M., "Iterative accelerated A* path planning", ***Decision and Control (CDC), 2010 49th IEEE Conference on***, 1201-1206 (2010).

31. Lavalle, S. M., "Rapidly-exploring random trees: A new tool for path planning", ***Technical Report TR 98-11, Computer Science Departmentat Iowa State University***, 98-110 (1998).

32. Kushleyev, A. and Likhachev, M., "Time bounded lattice for efficient planning in dynamic environments", ***Robotics and Automation, ICRA'09, IEEE International Conference on***, IEEE, 1662-1668 (2009).

33. Yannier, S., Şabanoviç, A. ve Onat, A., "Mobil robotlar için engelden sakındıran ve hedef yönlendiren katmanlı denetim yönetim", ***ELECO'02 Bildiri Kitabı***, Bursa, Türkiye, 168-172 (2002).

34. Sachin, B., Modi, P. C., Murty, V. S. and Hall, E. L., "Comparison of three obstacle - avoidance methods for a mobile robot", ***Proc. SPIE 4572, Intelligent Robots and Computer Vision XX***: Algorithms, Techniques, and Active Vision, doi:10.1117/12.444194 (2001).

35. Hart, P. E., Nilsson, N. J. and Raphael, B., "Correction to *a formal basis for the heuristic determination of minumum cost paths", ***SIGART Newsletter***, 28-29 (1972).

36. İnternet: Sinekli, R., "GAZEBO multi-robot simülatörü", **www.recaisinekli.com**, 3-21 (2013).

37. Manz, A., Liscano, R. and Green, D., "A comparison of realtime obstacle avoidance methods for mobile robots", ***In Experimental Robotics II***, 299-316 (1991).

38. Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. and Sebastain, T., "Principal of robot motion:" ***Theory, Algorithms, and Implementation***, MIT Press, 373-401 (2007).

39. Kshirsagar, S. and Shukla, K., "A Study of motion planning algorithms for mobile robots", ***International Journal on Computer Science and Engineering***, 34-38 (2010).

40. Wijaya, S. H. and Djamal, M., "Simulation of mobile robot navigation system using combination method of A* algorithm with virtual force field", ***Indonesian Journal of Physics***, 18 (1): 10-15 (2007).

41. Dechter, R. and Pearl, J., "Generalized best-first search strategies and the optimality of A*", ***Journal of the ACM***, 505 – 536 (1985).

42. Nguyen, H. V., Vien, N. A., SeungGwan, L. and TaeChoong, C., "Obstacle avoidance path planning for mobile robot based on multi colony ant algorithm advances in Computer-Human interaction", ***Automation and Systems,*** 285-289 (2008).

43. Xuan-Thu, L., Eun-Zu, H., Han-Sung, K., Young-Rok, C., Se-Han, L., Sung, H. H. and Yong-Guen, A., "Real-time obstacle avoidance of mobile robots Control", ***Automation and Systems***, 2294-2298 (2007).

ÖZGEÇMİŞ

Mehmet LAFCI 1986 yılında Kırşehir'de doğdu; ilk ve orta öğrenimini aynı şehirde tamamladı. Kırşehir Lisesinden mezun oldu. 2006 yılında Kocaeli Üniversitesi Mühendislik Fakültesi Elektronik ve Haberleşme Bölümü'nde öğrenime başlayıp 2011 yılında iyi derece ile mezun oldu. 2012-2013 yılları arasında Karabük Üniversitesi Elektrik Elektronik Mühendisliği Bölümü'nde araştırma görevlisi olarak çalıştı. 2013 yılında Kamu İhale Kurumunda göreve başladı ve halen aynı yerde Kamu İhale Uzman yardımcısı olarak çalışmaya devam etmektedir.

ADRES BİLGİLERİ

Adres : Hacettepe Üniversitesi Beytepe Kampüsü,
Kamu İhale Kurumu binası 1596. cadde, No:8
06800 Beytepe/Çankaya / ANKARA
E-posta : mehmetlfc@gmail.com