ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

EĞİTİM AMAÇLI ROBOTİK UYGULAMA GELİŞTİRME ORTAMI

Mustafa KARAKAYA

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ANKARA
2020

Her hakkı saklıdır

ÖZET

Yüksek Lisans Tezi

EĞİTİM AMAÇLI ROBOTİK UYGULAMA GELİŞTİRME ORTAMI

Mustafa KARAKAYA

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Mehmet Serdar GÜZEL

Bu tez çalışması kapsamında amaçlanan robotik alanında çalışmak isteyen öğrenci ve araştırmacıların kullanabileceği yeni ve özgün bir benzetim ortamı (simülatör) geliştirmektir. Bu kapsamda özellikle gezgin (mobil) robotlar için bir simülatör geliştirilmiştir. Bu çalışma vasıtasıyla, farklı gezinim (navigasyon) algoritmaların farklı senaryolar için nasıl çalıştığı karşılaştırmalı olarak gözlemlenebilmektedir. Ayrıca önerilen simülatör, araştırmacıların, kendi algoritmalarını gerçekleştirip, faklı senaryolar ile çalıştırıp, sonuçlarını görsel ekranda değerlendirmelerine olanak sağlamaktadır. Kullanıcılar kendi algoritmalarını yükleyip kaydedebilir ve algoritmalarının performanslarını kıyaslayabilirler. Kullanıcılara, makro kodlar veya kısmen sürükle bırak yeteneği sağlayan uygulama, farklı senaryoların hızlıca tasarlanmasına olanak sağlamıştır. Ayrıca ilgili benzetim ortamı ile literatürde tanımlı önde gelen yol planlama algoritmalarının performansları karşılaştırılarak ilgili algoritmaların farklı senaryolar üzerinden performans değerlendirilmesi bu tez kapsamında ayrıca yapılmıştır.

Eylül 2020, 91 Sayfa

Anahtar Kelimeler: Benzetim Ortamı, Makro Dil Editörü, Gezgin Robotlar, Yol Planlama Algoritmaları

ABSTRACT

Master Thesis

EDUCATION BASED ROBOTIC APPLICATION DEVELOPMENT FRAMEWORK

Mustafa KARAKAYA

Ankara University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Mehmet Serdar GÜZEL

The aim of this thesis is to develop a new and unique simulation environment that can be used by students and researchers who want to work in the field of robotics. In this context, a simulator has been developed especially for mobile robots. Through this study, how the different navigation algorithms work for different scenarios can be observed comparatively. In addition, the proposed simulator allows researchers to perform their own algorithms, run them with different scenarios and evaluate their results on the visual screen. Users can load and save their own algorithms and compare their algorithms' performance. The application, which provides users with macro codes or partially drag-and-drop capability, allows the rapid design of different scenarios. In addition, by comparing the performances of the related simulation environment and the leading road planning algorithms defined in the literature, the performance evaluation of the related algorithms over different scenarios was made within the scope of this thesis.

September 2020, 91 Pages

Key Words: Simulator**,** Macro Language Editor ,Mobile Robot, Path Planning Algorithms

TEŞEKKÜR

Yüksek lisans eğitimim süresince bu tez çalışmasının hazırlanması ve şekillenmesi birçok kişinin yardım ve destekleriyle gerçekleşmiştir. Öncelikle tez çalışmasının her aşamasında bana destek olup yol gösteren, mesleki bilgilerini ve deneyimlerini öğrencilerinden esirgemeyip önerileri ile katkıda bulunan değerli tez danışmanım Sayın Doç. Dr. Mehmet Serdar GÜZEL (Ankara Üniversitesi Bilgisayar Mühendisliği Anabilim Dalı) hocama sonsuz teşekkürlerimi sunarım.

Mustafa KARAKAYA

Ankara, Eylül 2020

İÇİNDEKİLER

TEZ ONAY SAYFASI

ETİK ..... **i**
ÖZET ..... **ii**
ABSTRACT ..... **iii**
KISALTMALAR DİZİNİ ..... **vii**
ŞEKİLLER DİZİNİ ..... **viii**
ÇİZELGELER DİZİNİ ..... **xii**
1. GİRİŞ **1**
2. LİTERATÜR **3**
2.1 Gezgin Robot Simülatörleri **6**
3. METODOLOJİ **8**
3.1 A Star **8**
3.2 Lee Algoritması **10**
3.3 Rapidly Random Tree Algoritması **14**
3.4 Rapidly Random Tree Star Algoritması **17**
3.5 Probabilistic Road Map Algoritması **20**
3.6 Potential Field Algoritması **23**
3.7 Dynamic Windows Approach Algoritması **28**
4. DENEYLER **34**
4.1 Sabit Engellerde Yön Bulma Algoritma Karşılaştırmaları **34**
4.1.1 A Star ve Lee Algoritma Kıyaslanması **35**
4.1.2 RRT ve RRT Star Algoritma Kıyaslanması **44**
4.2 Gerçek Zamanda Yön Bulma Algoritmalarının Karşılaştırılması **56**
4.2.1 Engelsiz senaryo **56**
4.2.2 Sabit tek engel senaryosu **59**
4.2.3 Tek hareketli engel senaryosu **62**
4.2.4 Bir hareketli bir sabit engel senaryosu **65**
4.2.5 Üç hareketli iki sabit engel senaryosu **68**
4.2.6 Beş hareketli engel senaryosu **71**
4.3 Algoritmaların Kat Edilen Yol Bakımından Karşılatırılması **76**

4.3.1 Engelsiz senaryoda algoritmaların kat edilen yol karşılaştırılması..... 76
4.3.2 Tek engelli senaryoda algoritmaların kat edilen yol karşılaştırılması..... 78
4.3.3 Üç engelli senaryoda algoritmaların kat edilen yol karşılaştırılması..... 79
4.4 Benzetim Ortamı Geliştirme Makro Yorumlayıcısı 81
5. SONUÇ..... 85
KAYNAKLAR 87
ÖZGEÇMİŞ..... 91

KISALTMALAR DİZİNİ

RRT	Rapidly Exploring Random Tree
PRM	Probabilistic Road Map
RRTStar	Rapidly Exploring Random Tree Star
DWA	Dynamic Window Approach
PTA	Potential Field Algorithm
BFS	Breath First Search

ŞEKİLLER DİZİNİ

Şekil 3.1 Lee Algoritması Yönleri 12
Şekil 3.2 Lee Algoritması ilk Durum 12
Şekil 3.3 Lee Algoritması ikinci iterasyon 13
Şekil 3.4 Hedef Noktasına Ulaşan Lee Algoritması 13
Şekil 3.5 Lee Algoritması En Kısa Yol Bulması 14
Şekil 3.6 400 Noktalı PRM Algoritması Gösterimi 22
Şekil 3.7 800 Noktalı PRM Algoritması Gösterimi 22
Şekil 3.8 1600 Noktalı PRM Algoritması Gösterimi 23
Şekil 4.1 A Star Tek Engel Başlama Noktası(30,30)–Bitiş Noktası (1620,845) 35
Şekil 4.2 Lee Tek Engel Başlama Noktası(30,30)–Bitiş Noktası (1620,845) 36
Şekil 4.3 Tek Engelde AStar ve Lee Algoritmaları Açısal Hız Karşılaştırmaları 36
Şekil 4.4 Tek Engelde AStar ve Lee Algoritmaları Kat Edilen Yol Karşılaştırmaları ... 37
Şekil 4.5 Tek Engelde AStar ve Lee Algoritmaları Harcanan Zaman Karşılaştırmaları 37
Şekil 4.6 A Star İki Sabit Engel Güzergah Başlama (32,121)-Bitiş Noktası (1296,235) 38
Şekil 4.7 Lee İki Sabit Engel Güzergahı Başlama (32,121)-Bitiş Noktası(1296,235) ... 38
Şekil 4.8 İki Sabit Engel Senaryosunda A Star ve Lee Algoritması Karşılaştırması..... 39
Şekil 4.9 İki Sabit Engel A Star ve Lee Kat Edilen Yol Karşılaştırması 39
Şekil 4.10 İki Sabit Engel A Star ve Lee Harcanan Zaman Karşılaştırması..... 40
Şekil 4.11 A Star Üç Sabit Engel Güzergah Başlama(244,146)-Bitiş Noktası (1485,275) 41
Şekil 4.12 Lee Üç Sabit Engelli Güzergah Başlama (244,146)-Bitiş Noktası (1485,275) 41
Şekil 4.13 Üç Sabit Engelli Senaryoda A Star ve Lee Açısal Hız Karşılaştırması 42
Şekil 4.14 Üç Sabit Engelli Senaryoda AStar ve Lee Kat Edilen Yol Karşılaştırması.. 42
Şekil 4.15 Üç Sabit Engelli Senaryoda A Star ve Lee Harcanan Zaman Karşılaştırması..... 43
Şekil 4.16 RRT Engelsiz Senaryo Başlama (135,220) –Bitiş Noktası(1245,690)..... 45
Şekil 4.17 RRT Star Engelsiz Senaryo Başlama (135,220)–Bitiş Noktası(1245,690) ... 46

Şekil 4.18 Engelsiz Senaryoda RRT ve RRT Star Açısal Hız Karşılaştırması 47
Şekil 4.19 Engelsiz Senaryoda RRT ve RRT Star Harcana Zaman Karşılaştırması 47
Şekil 4.20 Engelsiz Senaryo RRT ve RRT Star Kat Edilen Yol Karşılaştırması 48
Şekil 4.21 RRT Tek Engel Başlama Noktası(321,214) –Bitiş Noktası(1346,207)..... 49
Şekil 4.22 RRT Star Tek Engel Güzergahı Başlama(321,214) –Bitiş Noktası (1346,207) 50
Şekil 4.23 Tek Engelli Senaryoda RRT ve RRT Star Açısal Hız Karşılaştırması..... 51
Şekil 4.24 Tek Engelli Senaryoda RRT ve RRT Star Kat Edilen Yol Karşılaştırması..... 51
Şekil 4.25 Tek Engelli Senaryoda RRT ve RRT Star harcanan Zaman Karşılaştırması..... 52
Şekil 4.26 RRT Üç Engelli Güzergah Başlama (69,224) –Bitiş Noktası(1369,207)..... 53
Şekil 4.27 RRT Star Üç Engelli Güzergah Başlama (69,224) –Bitiş Noktası (1369,207) 53
Şekil 4.28 Üç Engelli Senaryoda RRT ve RRT Star Açısal Hız Karşılaştırması 54
Şekil 4.29 Üç Engeli Senaryoda RRT ve RRT Star Harcanan Zaman Karşılaştırması .. 54
Şekil 4.30 Üç Engelli Senaryoda RRT ve RRT Star Kat Edilen Yol Karşılaştırması 55
Şekil 4.31 Engelsiz Senaryoda PTA Güzergahı Başlama (26,816)-Bitiş Noktası(1696,11)..... 56
Şekil 4.32 Engelsiz Senaryo DWA Güzergahı Başlama(26,816)-Bitiş Noktası (1696,11) 57
Şekil 4.33 Engelsiz Senaryoda PTA ve DWA Açısal Hız Karşılaştırması..... 57
Şekil 4.34 Engelsiz Senaryoda PTA ve DWA Kat Edilen Hız Karşılaştırması..... 58
Şekil 4.35 Engelsiz Senaryoda PTA ve DWA Harcanan Zaman Karşılaştırması 58
Şekil 4.36 DWA Tek Sabit Engel Güzergahı –Başlama (26,746)-Bitiş Noktası(1687,124)..... 59
Şekil 4.37 PTA Tek Sabit Engel Güzergahı–Başlama (26,746)-Bitiş Noktası (1687,124) 60
Şekil 4.38 Tek Sabit Engel Senaryosunda PTA ve DWA karşılaştırması 60
Şekil 4.39 Tek Sabit Engel Senaryosu PTA ve DWA Harcanan Zaman Karşılaştırması..... 61

Şekil 4.40 Tek Sabit Engel Senaryosu PTA ve DWA Kat Edilen Yol Karşılaştırması 61
Şekil 4.41 DWA Tek Hareketli Engel Senaryosu Başlama(21,4)-Bitiş Noktası(1567,245) 62
Şekil 4.42 PTA Tek Hareketli Engel Senaryosu Başlama (21,4)-Bitiş Noktası (1567,245) 63
Şekil 4.43 Tek Hareketli Engel Senaryosu PTA ve DWA Açısal Hız Karşılaştırması .. 63
Şekil 4.44 Tek Hareketli Engel PTA ve DWA Harcanan Zaman Karşılaştırması 64
Şekil 4.45 Tek Hareketli Engel Senaryosu PTA ve DWA Kat Edilen Yol Karşılaştırması 64
Şekil 4.46 DWA Bir Sabit bir Hareketli Engel–Başlama(14,197)-Bitiş (1595,204) 65
Şekil 4.47 PTA Bir Sabit bir Hareketli Engel–Başlama(14,197)-Bitiş(1595,204) 66
Şekil 4.48 Bir Sabit ve Bir Hareketli Engel PTA ve DWA Açısal Hız Karşılaştırması 66
Şekil 4.49 Bir Sabit ve Bir Hareketli Engel PTA ve DWA Kat Edilen Yol Karşılaştırma 67
Şekil 4.50 Bir Sabit ve Bir Hareketli Engel PTA ve DWA Harcanan Zaman Karşılaştırma 67
Şekil 4.51 DWA Üç Hareketli Engel ve İki Sabit Engel -Başlama(32,56)-Bitiş(1613,346) 68
Şekil 4.52 PTA Üç Hareketli ve İki Sabit Engel Başlama(32,56)-Bitiş(1613,346) 69
Şekil 4.53 Üç Hareketli İki Sabit Engel PTA ve DWA Açısal hız Karşılaştırması 69
Şekil 4.54 Üç Hareketli ve İki Sabit Engel PTA ve DWA Harcanan Zaman Karşılaştırma 70
Şekil 4.55 Üç Hareketli ve İki Sabit Engel PTA ve DWA Kat Edilen Yol Karşılaştırması 70
Şekil 4.56 Beş Hareketli Engel DWA Başlama(2,149)-Bitiş Noktası(1613,239) 71
Şekil 4.57 Beş hareketli Engel PTA Başlama Noktası(2,149)- Bitiş Noktası (1613,239) 71
Şekil 4.58 Beş hareketli Engel senaryosu PTA ve DWA Açısal Hız Karşılaştırması 72
Şekil 4.59 Beş Hareketli Engel Senaryosu PTA ve DWA Kat Edilen Yol Karşılaştırması 72

Şekil 4.60 Beş Hareketli Engel PTA ve DWA Harcanan Zaman Karşılaştırma 73
Şekil 4.61 Engelsiz Senaryoda Tüm Algoritmaların Alınan Yol Karşılaştırması 77
Şekil 4.62 Tek Sabit Engelli Senaryoda Tüm Algoritmaların Alınan Yol Karşılaştırması..... 78
Şekil 4.63 Üç Sabit Engelli Senaryoda Tüm Algoritmaların Alınan Yol Karşılaştırması..... 80
Şekil 4.64 Örnek Script Ekranı 81
Şekil 4.65 Boş Alan Tanımı 82
Şekil 4.66 Başlangıç ve Btişi Noktaları 82
Şekil 4.67 Engel Tanımı..... 83
Şekil 4.68 Algoritma Çalıştırıldıktan Sonra Ekrandaki Görüntü 84
Şekil 4.69 Mobil robotun hedefe doğru çizdirilmesi 84

ÇİZELGELER DİZİNİ

Çizelge 4.1 Tek Engel AStar ve Lee Sonuçları 37
Çizelge 4.2 İki Sabit Engel A STar ve Lee Sonuçları 40
Çizelge 4.3 Üç Sabit Engel A Star ve Lee Sonuçları 43
Çizelge 4.4 Engelsiz Senaryo RRT ve RRT Star Sonuçları 48
Çizelge 4.5 Tek Engel RRT ve RRT Star Sonuçları 52
Çizelge 4.6 Üç Engel RRT ve RRT Star Sonuçları 55
Çizelge 4.7 Engelsiz Senaryo PTA ve DWA sonuçları 59
Çizelge 4.8 Tek Sabit Engel PTA ve DWA Sonuçları 62
Çizelge 4.9 Tek Haraketli Engel PTA ve DWA Sonuçları 65
Çizelge 4.10 Bir Hareketli bir sabit engelde PTA ve DWA sonuçları 68
Çizelge 4.11 Üç Hareketli iki sabit engelde DWA ve PTA Sonuçları 70
Çizelge 4.12 Beş Hareketli Engelde DWA ve PTA Sonuçları 73
Çizelge 4.13 Engelsiz Senaryo Alınan Yol Sonuçları 77
Çizelge 4.14 Tek Engel Senaryosu Alınan Yol Sonuçları 79
Çizelge 4.15 Üç Sabit Engel senaryosunda alınan yol sonuçları 80

1. GİRİŞ

Hareket kontrolü(Robot Motion Control), bir robotun, önceden tanımlanmış görevleri yapmak için nasıl hareket etmesi gerektiğini belirleyen bir yazılım bileşenidir. Bir robotla bir şeyler yapmanın birden fazla yolu varsa, seçilen yolun diğer yollara göre daha özel niteliklere sahip olması gerekir. Seçilen yol zamanı azaltabilir, gücü artırabilir, çarpışmalardan maksimum oranda kaçabilir, çalışma alanı sınırlarından kaçınabilir, güç tüketimini azaltabilir ve doğruluğu artırabilir. Hareket kontrolü, her zaman belirli kısıtlamalar içermelidir. Robotların ve bağlantılarının hız ve ivme limitleri vardır. Aktüatörlerin belirli tork ve kuvveti vardır. Dünyanın ve robotun getirdiği fiziksel gerçeklik bu kısıtlamalara dahildir (Abbee, 2011).

Özellikle çevrimiçi robot hareket kontrolü çağın ilgi çeken ve sürekli gelişme gösteren alanlarındandır. Pek çok robot, hareket başlamadan önce çevrimdışı tam yollar oluşturur. Çevrimdışı yol oluşturma, önceki durumlarla ilgili hesaplamalarda robotun gelecekteki durumlarının kullanılmasına izin verir. Bu, titreşime neden olabilecek daha yüksek hareket türevlerini (sarsıntı, hızlanma türevi gibi) sınırlandırmaya yardımcı olur. Ancak dezavantajı, hareket başlamadan önce bilginin eksik olmasıdır. Robot önceden hesaplanmış bir yolda harekete başladığında, çevresel ve kullanıcı girişi değişikliklerine cevap veremez. Gerçek zamanlı olarak robot hareketini kontrol etmek için daha fazla çalışmaya ihtiyaç duyar. Bu yarışta, robot hareket kontrol algoritmalarının hesaplama hızı çok önemlidir. Güçlü bir algoritmik yaklaşımla, birçok alternatifi, zaman ve adım açısından değerlendirip en iyisini seçmek önemlidir (Wolfram vd. 2011).

Bu araştırmada odaklanılan konu robotların hareket planlamasıdır. Modern robotik biliminin başlangıcından itibaren araştırmacılar ve mühendisler robotlarının ana hatlarının yörüngesini geliştirmenin yollarını aramaktadırlar. Bu konuda zamanla çok fazla gelişme kaydedildi ve güvenilir algoritmalar geliştirildi. Robotlar, otomatik olarak nesnelere ve cisimlere çarpmadan kendi rotalarını çizebildiler. Bu rotaları çizerken de temelde iki önemli kavrama odaklanıldı. Bu kavramlar boş alan ve güzergâhtır. Robotun çalışma alanında, robotun hareket edemediği ya da giremediği bazı alanlar mutlaka olacaktır, çünkü buralarda engeller bulunmaktadır. Algoritmanın, engelli konumların

tümünü yapılandırma alanından kaldırması gereklidir. Bu düzenleme hareket planlama için gerçekten uygun olan tüm yerleri içeren “boş alan” ile robota uygun alan haline bırakır. Yapılandırma alanı sabittir. Ancak, yeni nesneler alana girip çıktığında boş alan değişebilir (Mousas, 2013).

Son olarak, hareket planlama algoritması boş alandaki en iyi yolu yani en iyi rotayı çizmelidir. Bazı hareket planlama ve yön bulma algoritmaları (kombinatoryal planlama algoritmaları olarak da bilinir) "tamamlandı" ya da başarısız sonuç döner; bu, varsa bir sorunu çözdüğü veya başarısız olduğunu bildireceği anlamına gelir. Diğerleri (örnekleme tabanlı planlama olarak bilinir) daha verimlidir ancak bazı durumlarda cevapları bulamamaktadır (Patel, 2019). Bazen de hızlı bir cevap almak yön bulma algoritmasının tüm olası güzergahları hesaplayıp, değerlendirip en iyisi için saatlerce beklemek zorunda kalmasından daha faydalıdır. Bu nedenle, Probabilistic Roadmaps (PRM) gibi örnekleme tabanlı planlama algoritmalarının alternatiflere göre sıklıkla tercih edilmesinin nedeni budur. Bu noktada yön bulma algoritmalarını kendi kulvarlarında değerlendirip sanal robotları aynı ortamda aynı engellerle test edip kıyaslanması bu araştırmanın amaçlarından biridir.

2. LİTERATÜR

Robotik ve yapay zeka alanı son yıllarda sürekli olarak gelişme halindedir ve gün geçtikçe yeni dönüm noktalarına ulaşılmaktadır. Aslında robotların tarihi düşünülenden çok daha eskiye dayanmaktadır. Eski Mısır ve Yunan medeniyetlerinde yer edinen otomatik su saatleri, Homerus'un İlyada'sında bahsi geçen insan yapımı hizmetçiler, Leonardo Da Vinci'nin yürüyen mekanik aslanı robotların temeline örnek olarak verilebilir. Örneklerin hepsi insansı robotlara uzanan bir teknolojinin ilk köşe taşlarıdır.

Otomatların, önemli ölçüde ilerleme gösterdiği dönem 13. asır oldu. Günümüz bilgisayarları dahil tüm bilgisayarlar robot kategorisine girmektedir. Atari, video oyunları ve ses çıkarıp hareket eden oyuncak bebekler robot olarak tasnif edilir. Ancak algılarımızdaki robot tarifi, George Moore'un icat ettiği, iki bacağı üzerinde yürüyebilen makine ile birebir aynıdır.

1893'te Kanadalı profesör George Moore, çelikten yapılmış ve 0.5 beygir gücündeki buhar motoruyla çalışan insansı robot prototipi "Steam Man"i üreterek robotların tarihindeki dönüm noktalarından biri oldu.

Tesla günün birinde akıllı, bağımsız bir insansı robot inşa etmenin mümkün olduğuna inandığını yazdı. Ancak Tesla'nın fikirleri 20. yüzyıla kadar ciddiye alınmadı. Aslında, bildiğimiz robot bilimi en çok 20. yüzyılın ortalarında parladı. Araştırma ve geliştirme ekipleri ciddi bir şekilde çalışmaya başladıktan sonra robotlar sanayide üretime entegre edildi; askeri, havacılık, uzay, tıp ve eğlence sektörlerinde yavaş yavaş kendini göstermeye başladı (Yap 2002).

"Robot" kelimesi ile ilk kez 1920'de Çekoslavak Karel Capek tarafından yazılan bir tiyatro eserinde dünya literatürüne girdi. 22 yıl sonra ise Amerikalı profesör Isaac Asimov'la 'robot' kelimesinden türetilen 'robotik' kavramı ile dünya tanıştı. Mısırlılarla insanlığın hayatına giren robotlar, bu süreçte sadece eğlence amaçlı unsurlar olmaktan çıkıp insanların hayatını kolaylaştırmaya yönelik geliştirilen teknolojik konseptler halini

aldı. Gelişim aşamalarında tarihe düşülen önemli notlardan biri Isaac Asimov'un 1942'de yayınladığı robot yasası oldu.

Bu üç maddelik robot yasasına göre;

1. Robotlar, insanlara zarar veremez ya da eylemsiz kalarak onlara zarar gelmesine göz yumamaz.
2. Bir robot, birinci kanunla çelişmemesi şartıyla, insanların verdiği emirlere uymak zorundadır (Şişman vd. 2016).
3. Bir robot, birinci veya ikinci kanunla çelişmediği sürece kendi varlığını korumalıdır (Erden, 2013).

20. yüzyıla gelindiğinde robotik sistemler iş dünyasına büyük oranda girmişti. Robotların işlevselliği adım adım arttı. Tekrar programlanabilen robotlar, birden fazla iş yapabilenler derken robotlar şimdi evlerde bile kullanabilir hale geldi (Şişman vd. 2016).

Tezin asıl konusuna gelindiğinde robot hareket kontrolü ve buna bağlı olarak yön bulma algoritmalarıdır. Stanford Araştırma Enstitüsü'nden Peter Hart, Nils Nilsson ve Bertram Raphael bu alandaki algoritmayı ilk kez 1968'de yayınladı. Dijkstra'nın 1959 algoritmasının bir uzantısı olarak görülmektedir. A * algoritmasının taramasına rehberlik etmek için sezgisel yaklaşım kullanılmış, bu da daha iyi performans elde etmiştir.

Lee algoritması, BFS algoritmasına dayanan yön bulma için olası bir çözümdür. Varsa, her zaman en uygun çözümü vermektedir, ancak yavaştır ve önemli ölçüde bellek gerektirir(IRE Trans. Electronic Computer, EC-10, 1961).

PRM robotlar için yön belirleme algoritması olarak kullanılabilir. Bu algoritma robotun bir başlama noktasından bitiş noktası engellerden kaçınarak gitme işlemini gerçekleştirir. Lydia E. Kavraki PRM yönteminin mucitidir. PRM yönteminde, bazıları

oldukça karmaşık olan, daha hızlı performans elde etmek için örnekleme stratejisini ve bağlantı stratejisini değiştiren birçok varyant vardır.

RRT ağaç şeklinde rastgele büyüyen büyük alanları taramak için oluşturulmuş bir algoritmadır. Oluşturulan ağaç dışbükey değildir. Ağaç arama alanından rastgele çizilen örneklerden aşamalı olarak oluşturulur ve doğası gereği problemin araştırılmamış geniş alanlarına doğru büyür.

Potential Field algoritması da ilk olarak 1992 de Yong K Hwan ve Narenda Ahıja'nın "A Potential Field Approach to Path Planning" adlı bildirisinde yer almıştır. PTA etkilidir ancak local minimaya bazen takılabilir. Örnekleme tabanlı algoritmalar, local minima sorununu önler ve birçok sorunu oldukça hızlı bir şekilde çözer. Hiçbir yol olmadığını belirleyemezler, ancak daha fazla zaman geçtikçe sıfıra inen bir başarısızlık olasılığı vardır.

Robotik hareket planlamasında Dynamic Window Approach algoritması gerçek zamanda hareketli engellerden kaçınma algoritmasıdır., Dieter Fox, Wolfram Burgard ve Sebastian Thrun 1997 de mobil robotların çalışma anında çarpışmasını önlemek için geliştirmişlerdir. Dynamic Windows approach yaklaşımı robotun özelliklerinden türetilip oluşturulur ve özellikle robotun sınırlı sürat, ivme, açısal hız gibi kıstaslarıyla başa çıkmak için tasarlanmıştır.

İlk olarak geçerli bir arama alanı oluşturan ve ikincisi arama alanında en uygun çözümü seçen iki ana bileşenden oluşur. Arama alanları robotun dinamik özelliklerine bağlı olarak belirli bir zaman aralığında erişilebilen ve kazalardan kurtarılmış güvenli bölgeleri temsil eder. Bu yaklaşımdaki optimum hedef, mobil robotu hareketli engellerden maksimum kıvraklıkla kaçınarak hedefe ulaştıran robotun dinamik özelliklerini düzgün seçebilmektedir. Hız , yön ve ivme buna örnek olarak verilebilir.

Robotik alanda yapılan navigasyon simülatörleri literatür taramasında benzer alanda geliştirilen bir simülatör bulunmuştur. Eric Antonelo otonom robot navigasyon

deneyleri için bir simülatör geliştirmiştir. Grafik kullanıcı arayüzünde menüler, komut çubukları ve ortam ekranı bulunur. Başka bir uygulamasında ise ortamında bulunan bir robotun beynini simüle eden bir program geliştirmiştir. İlgili robotun sensör verilerini (mesafe, renk ve temas) aldıktan sonra, aynı robota aktüatör verilerini (yön ayarı ve hız ayarı) gönderir. Bu döngü simülasyonlar bitene kadar tutulur (Antonelo, 2020).

2.1 Gezgin Robot Simülatörleri

Son yıllarda robotik alanının gelişmesiyle birlikte bu alanda önemli araçlar ve simülatörlerde geliştirildi. Bu alanda ROS(Robot Operating System) adında işletim sistemi geliştirildi. Robot Operating System işletim sistemleri işlevlerinin yanında bazı ekstrem işlevleri de gerçekleştirebilir. Robot konfigürasyon sistemi, eşzamanlı ve eşzamansız çağrılar buna örnek olarak verilebilir.

İlk zamanlar her robot araştırmacısı robotun donanım kısmını ve donanımla irtibata geçecek gömülü yazılım kısmını tasarlamak zorundaydı. Tipik olarak, bu şekilde tasarlanan programlar, günümüzde hizmet robotlarında karşılaşabileceğimiz gibi, elektroniklere benzer şekilde, en katı anlamda robotiklere kıyasla gömülü programlamaya daha yakındı.

Robotik bir işletim sisteminin ana fikri, tekerleği sürekli olarak yeniden icat etmekten kaçınmak ve PC'ler için geleneksel bir işletim sistemi gibi donanım soyutlamasını gerçekleştiren standartlaştırılmış işlevsellikler sunmaktır.

ROS işletim sistemi farklı alanlardan uzmanlıkları tek çatı altında birleştirmektedir. Robot tasarımının belirli süreçleri vardır.

- Sürücüler kodlanarak , donanımı kontrol etmek
- Bellek ve süreçlerinin kontrolü
- Senkronizasyon, eş zamanlı çalışma ve veri değiştirme işlemleri

- Soyut akıl yürütme algoritmaları sağlamak, yapay zekayı büyük ölçüde kullanmak.

Robotik alanı bu süreç ve kontrollerden ötürü bireysel bir programlamanın yanı sıra farklı uzmanlık alanları da gerektirir (Mazzari, 2016).

ROS, kendi bünyesinde çok fazla gelişmiş yapıtaşı barındırır. Bu araçlardan bazıları algoritmalar ve bazıları simülasyon alanında çok faydalıdır.

Player/Stage Simülatör: Player, temelde bir ağ sunucusudur. Robot kontrolü işlemini gerçekleştirir. Ethernet üzerinden robot sensörlerine bir api sağlar. Client program TCP/IP protokolü ile player ile haberleşir, sensörlerden bilgiler alınır, komutlar gönderilir ve birimler yapılandırılır.

Üç kısımdan oluşur:

- Kodlama bölümü: Player ile etkileşimli çalışır.
- Player: İstemci programdan komutlar robota gönderilir. Robotun sensör bilgileri de istemci programa iletilir.
- Stage Player'dan komutlar alır ve simülasyon ortamında robot hareket ettirilir. simülasyondaki robottan sensör bilgilerini alır ve bu bilgiler Player'a gönderilir (Tang, 2017).

Gazebo Simülatör: Gazebo son dönemin robot simülatörlerindendir. Üç boyutludur. Karmaşık mobil robot ortamlarını simüle eder. Gazebo, Player/ Stage projeleriyle işbirliği içinde geliştirilmiştir. Stage, iç mekan ortamlarındaki robotlar arasındaki etkileşimleri simüle etme yeteneğine sahip olsa da, dış mekan ortamlarını modelleme ve gerçekçi sensör geri bildirimi sağlama yeteneğine sahip bir simülatöre ihtiyaç duyuldu. Stage in yetersiz kaldığı dış mekanlarda gazebo daha etkilidir. Simülasyondaki tüm nesneler fiziki olaylara maruz bırakıldıklarında gerçekçi davranmalarını sağlayan hız, yön ,açı, sürtünme ve etkileşim gibi birçok özelliğe sahiptir. Bu da aslında gazebonun diğer simülatörlerden farkıdır (Koenig vd. 2004).

3. METODOLOJİ

Navigasyon algoritmaları modellemesi çizge kuramına dayanmaktadır. Dolayısıyla algoritma çatısını çizgelere dayandırmak ve tasarımını çizgeler üzerinde kurmak ilk adımlardan biridir.

Çizge, literatürde matematiksel bir terimdir ve çizgiler ve noktalar arasındaki bağıntının açıklayıcısıdır. Bir çizge, düğüm ve kenarlardan oluşmaktadır. Grafikteki her bir noktaya düğüm adı verilir. G = (v, e) formülü ile ifade edilir. Düğüm kenarların eşleşme noktalarıdır. Düğüm gerçek ortamda aslında bir konum belirtmektedir. Kenar ise iki düğüm arasındaki bağlantıya belirtir. Bir bağlantı, düğümler arası hareketleri ve etkileşimlerin göstergesidir. Genelde ok ile gösterilen bir yönü vardır (McDermott, 1992). Bir ok kullanılmazsa, bağlantının iki yönlü olduğu anlamına gelir. Ulaşım coğrafyası bir çizge ile tanımlanabilir. Karayolu, toplu taşıma ve demiryolu ağları gibi çoğu ağlar kenarlar ve düğümlerden oluşur. Telefonlar ve antenler bir konum belirtir ve düğüm olarak gösterilebilir. Fiber optik kablolar gibi aralarındaki fiziksel altyapı bağlantılar gibi davranabilir. Bu, tüm ulaşım ağlarının bir şekilde bir grafik teorisi ile temsil edilebileceğini göstermektedir (Wolf vd. 2004, Fakoor vd.2016, Bodrenko 2019).

3.1 A Star

A Star ,yol bulma ve çizge dolaşımında yaygın olarak kullanılan bir algoritmadır. A Star Algoritması ağ üzerinde birçok koordinat arasında gidilecek en kısa yolu etkili bir şekilde oluşturur. A Star algoritmasının işleyişi temelde her adımda önceden planlama yaparak daha optimal bir karar verilmesini sağlayacak şekilde düzenli bir çizge arama algoritmasına bir sezgisel çözüm sunar. A Star Djikstra algoritmasının bazı özelliklerine sahip bir uzantısıdır (Yap, 2002).

Djikstra gibi, A Star da başlangıç noktasından hedef noktasına en düşük maliyetli bir rota oluşturmaya çalışır. A Starı pek çok yön bulma algoritmasından daha iyi yapan şey

, her düğüm için her nokta için A Starın bu noktayı kullanan bir yolun toplam maliyetinin bir tahminini veren f (n) fonksiyonunu kullanmasıdır.

f (n) = n düğümü boyunca toplam tahmini alınacak yol tutarı

g (n) = n düğümüne varış maliyeti

h (n) = n'den varış noktasına tahmini maliyet. Bu, maliyet algoritmanın heuristik bölümüdür. Bu sezgisel formül (3.1) ile ifade edilmiştir.

$$|f\ (n) =\ g\ (n) +\ h\ (n)| \qquad (3\text{-}1)$$

Gridde, A Star algoritması başlangıçtan tüm bitişik hücreleri dikkate alır. Griddeki tüm bitişik alanlar tamamlandıktan itibaren erişilemeyen engelli alanlar kırpılır. Daha sonra f (n) tahmini olan en düşük maliyetli hücreyi seçer. Bu işlem, varış noktasına en kısa yol bulunana kadar iteratif bir şekilde tekrar edilir. F (n) 'nin hesaplanması, genellikle iyi sonuçlar veren bir sezgisel yöntemle yapılır. H(n) metodunun hesaplanması çeşitli yollarla sağlanabilir:

A Star algoritması Djikstra algoritmasının bir türevidir. Eğer h (n) = 0 ise, A Star, Dijkstra algoritması kendisine dönüşür ve bu durumda da sezgisel yaklaşımdan çıkar en kısa yolun bulunması garanti haline gelir. Sezgisel işlev makul düzeyde olmalıdır; bu, varış noktasına ulaşmak için ulaşma maliyetinin optimum düzeyde olmalıdır (Baijayanta, 2019).

A Starın performansını belirlemede iyi bir sezgisel yöntem kullanma çok önemlidir. H (n) değeri ideal olarak varış noktasına ulaşma maliyetini belirleyecek en önemli metoddur. Varış noktasına varmanın maliyetinden daha düşük olan h(n) metodu çoğunlukla kullanılır. Bu, h(n) nin doğru çalışmasını sağlar, eğer daha yüksek bir h değeri seçersek, daha hızlı ancak daha kötü performansa yol açar. Bu nedenle, genellikle gerçek maliyetten daha düşük bir h(n) seçilmelidir (Tesauro, 1995).

The Manhattan Distance Sezgisel Yaklaşımı: Bu h(n) sezgisel hesaplama yöntemine Manhattan yöntemi denir, çünkü mevcut bulunan noktadan hedef noktasına ulaşmak için yatay ve dikey olarak kat edilen toplam noktaların sayısı hesaplanır. Köşegen gidiş yönü ve buna yol açabilecek engelleri ihmal edilir. Bu yaklaşım (3.2) formülü ile ifade edilir. Başlangıç ve hedef noktalarının x eksenindeki farkı ve y eksenindeki farkından hesaplanır.

$$.\boxed{h = |X\{Başlangıç\} - X\{Hedef\}| + |Y\{Başlangıç\} - Y\{Hedef\}|} \quad (3\text{-}2)$$

Bu sezgisel yöntem, yol düz bir çizgiyi takip ettiğinde düzgün çalışmaktadır. Dolayısıyla, A, bu sezgisel yaklaşımla düz çizgi hareketlerinin türetiminden olan yolları bulacaktır. Bazı zamanlar doğrudan varış noktasına giden düz bir yoldan gitme eğilimini tercih edebiliriz.

The Euclidean Distance Sezgisel Yaklaşım: Bu sezgisel buluş, Manhattana göre biraz daha düzgün sonuçlar verir. Her ikisini de aynı labirentte eşzamanlı olarak çalıştırmayı denersek, Euclid yol izleyicisi düz bir çizgi boyunca bir yolu tercih eder. Bu daha doğrudur, ancak daha yavaştır çünkü yolu bulmak için daha geniş bir alanı taraması gerekir.(3.3) formülünün yaklaşımı temel olarak Pisagor teoreminden gelmektedir.

$$\boxed{H = \sqrt{(X\lfloor Başlama\rfloor - X\lfloor Hedef\rfloor)^2 + (Y\lfloor Başlama\rfloor - Y\lfloor Hedef\rfloor)^2}} \quad (3\text{-}3)$$

A Star algoritmasının en büyük dezavantajlarından biri, bellek gereksinimidir. A Star algoritmasında tüm açıkta kalan gridin kaydedilmesi gerektiğinden bu algoritma bellek bakımından verimsiz çalışmaktadır (Lan, 2002).

3.2 Lee Algoritması

Amaç, A noktasından B noktasına giden bir grid hücre zinciri bulmaktır (Sutton, 2017). Bir grid karesi boyutu uzay alanının her bir bölmesine denk gelen aralıktır. Engel olarak tanımlanan alanlar grid karesi olarak belirtilir.

Matriste bloke edilmiş engel denilen alanlar vardır. Bu bloke edilmiş engellerden geçilmez. Matriste bir kaynak nokta ve bir hedef nokta vardır. Her iterasyonda bir adım sayısı hesaplanır. Çapraz olarak hareket etme Lee Algoritmasında yoktur. Hedef noktasına çapraz yönler olmadan optimum güzergahın bulunması gerekir. Mümkün olan tek hamle Bitişik Sol, Bitişik Üst, Bitişik Sağ, Bitişik Alt, Lee algoritması da bu maze algoritmalarından biridir. Lee algoritmasının bir özelliği dalga yayılma tekniği kullanmasıdır. Varış noktasına ulaşıldığında dalga üretimi durur ve back tracing(Geriye doğru yol izleme) tekniği devreye girer.

Avantajları

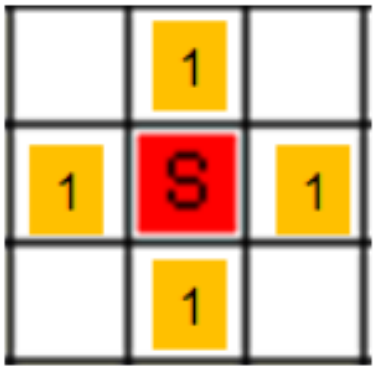
• Varsa 2 terminal arasındaki bağlantıyı bulma garantisi verir.
• Minimum yolu garanti eder.

Dezavantajları

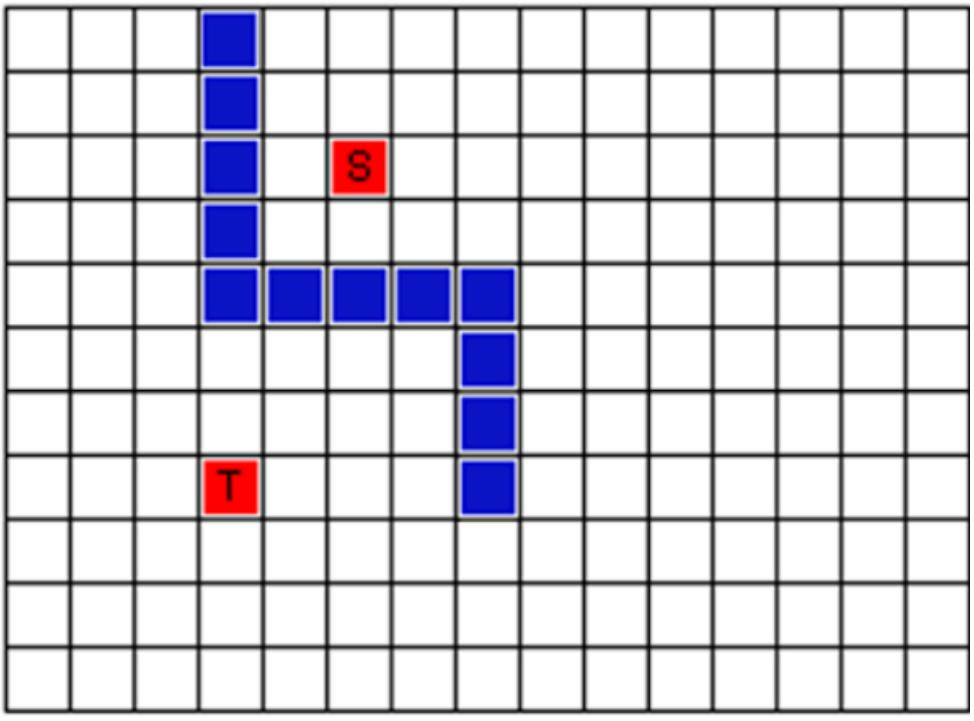
• Yoğun düzen için geniş bellek gerektirir.
• Yavaş çalışır.

Önce bir genişlik arama yaparak ve her matris noktasını kaynaktan uzaklığına göre etiketleyerek bağlantının kaynak ve hedef düğümleri arasında en kısa yol bağlantısını arar. Bir bağlantı mümkünse bu genişletme aşaması sonunda hedef düğüme ulaşacaktır. Daha sonra ikinci bir iz geri aşaması, etiketleri azalan herhangi bir yolu izleyerek bağlantıyı oluşturur. Lee Algoritması belirlenen başlangıç ve bitiş noktası arasındaki en kısa güzergahı bulması kesindir. Bulunan sonucun en kısa yol olması garantidir.. Ancak, birden çok bağlantı yapıldığında bir bağlantı diğer bağlantıları engelleyebilir. Lee algoritması, varsa 2 terminal arasındaki bağlantıyı bulmayı garanti etmesine ve minimum yolu garanti etmesine rağmen, yoğun düzen için büyük bellek gerektirir ve yavaştır. Bir M by N grid için bu algoritmanın zaman ve mekan karmaşıklığı O (MN) 'dir.

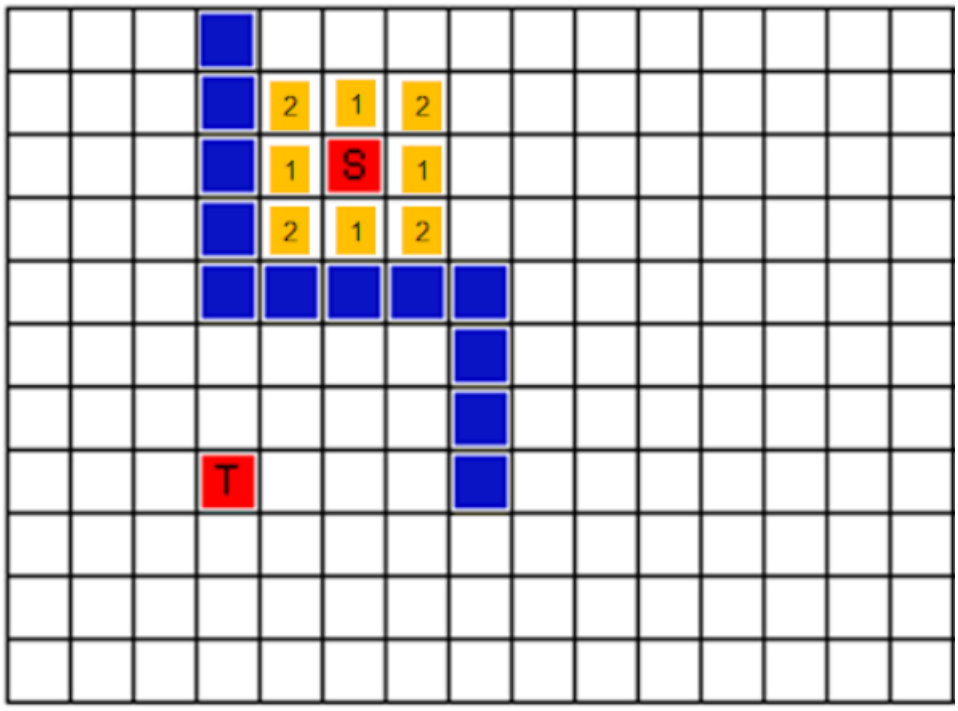
1. Yönlendirme katmanı bir grid olarak gösterilir.
2. Her grid noktası, bitişik grid noktalarına bağlantılar içerir.
3. Kaynak ve hedef arasında en kısa yol bağlantısını aranır.
4. Önce genişlik araması yapılır.
5. Her grid noktasını kaynaktan uzaklığıyla etiketler.
6. Geri izleme aşaması, etiketleri azalan yolları izleyerek bağlantıyı oluşturur (Gupta, 2018).



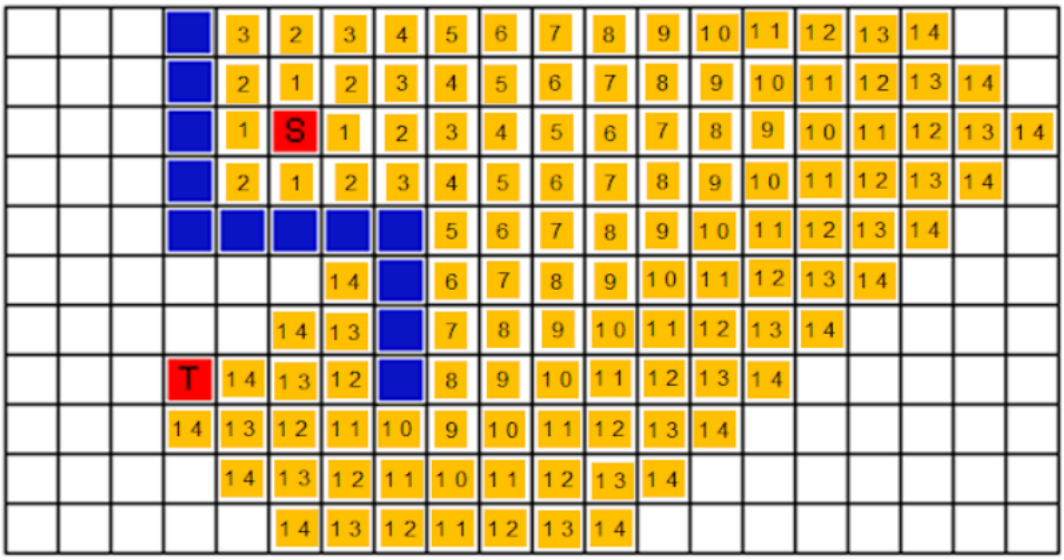
Şekil 3.1 Lee Algoritması Yönleri



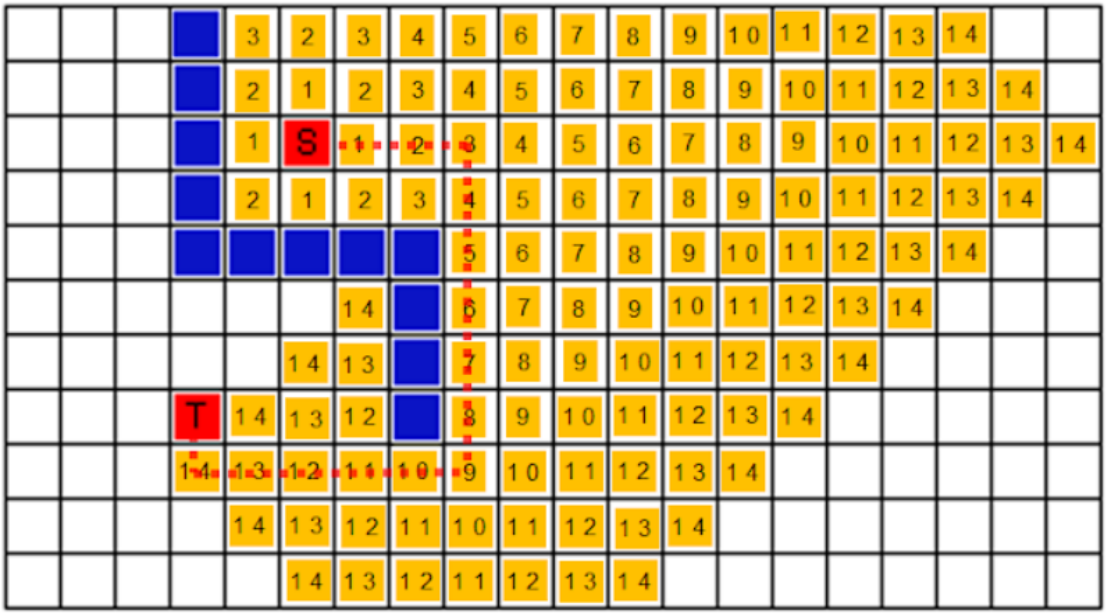
Şekil 3.2 Lee Algoritması ilk durum



Şekil 3.3 Lee Algoritması ikinci iterasyon



Şekil 3.4 Hedef Noktasına Ulaşan Lee Algoritması



Şekil 3.5 Lee algoritması en kısa yol bulması

Lee Algoritması basit ve uygulaması kolaydır; ancak, hesaplama açısından yoğun olduğu için, paralel sistemlerde uygulama için cazip bir aday algoritma gibi görünmektedir. (Gupta, 2018) Hücrelerin paralel formata eşlenmesi algoritmanın verimliliği için kritik öneme sahiptir (Router, 2020) .

3.3 Rapidly Random Tree Algoritması

RRT algoritmasının ana fikri, aramayı boş alanlara doğru kademeli olarak saptırarak ve her yinelemeyle, ağaçtaki potansiyel yeni düğümleri, önceki yinelemelerde zaten V'ye eklenmiş düğümlerden uzağa "iterek" boşluk doldurma ağacı oluşturmaktır. Başlangıçta, boşlukta tanımlanmış bir nokta (Xbaşlangıç) V'ye eklenir ve yalnızca bir düğüm keşfedildiği için E kenar kümesi boştur. Ağacı genişletmek ve önceden tanımlanmış maksimum sayıda düğüm K için dallar oluşturmak için, her yinelemede, numune fonksiyonu ile tekdüze bir dağılımdan bir Xrastgele örneği alınır (Alcazar vd. 2011).

Bu nokta, V'de X rastgele'a göre en yakın düğümü aramak için kullanılır. Sonuçta, Xrastgele ve Xenyakın arasındaki mesafeyi en aza indirerek yeni bir Xyeni düğümü bulunur. Yani, X yeni, X rastgele yönünde E'ye eşit veya daha küçük bir mesafe ile bir adım gerçekleştirerek yeni düğümdür. Global kısıtlamalar karşılanırsa, V'ye yeni düğüm eklenir ve E'ye en yakın komşudan yeni düğüme yeni bir kenar eklenir. RRT'lerin rastgele yürüteçler olmadığını, yani Xbos üzerinde rastgele bir yürüyüş yapan ve rastgele bir V'den bir düğüm seçen bir algoritmanın daha önce ziyaret edilen yerlere karşı güçlü bir önyargıya sahip olduğunu belirtmek de önemlidir. Temel fark, RRT'nin boş yerlere önyargılı olarak çalışmasıdır, çünkü V'den bir düğüm seçimi en yakın komşu işlevine dayandırılır. Aşağıda RRT'nin özelliklerinin bir listesi verilmiştir. Bu özelliklerin bazıları RRT'yi çeşitli hareket planlama sorunları için cazip kılmaktadır.

1. v ∈ V düğümü ile Xbaşlangıç düğümü arasında her zaman bir yol vardır. Çünkü Xbaşlangıç ∈ V düğümü ve her düğümün yalnızca bir ebeveyni olması gerekir.
2. RRT'ler hızla keşfedilmemiş bölgelere doğru büyür.
3. Genel varsayımlar altında RRT olasılıkla tamamlanmıştır. Sonucun bulunma olasılığı gittikçe bire yakınsar.

RRT algoritmasının Xboş'daki artımlı büyümesine göre davranışı için bir sezgisel yöntem kullanılır. Bu sezgi, bir düzlemdeki bir dizi noktanın Voronoi diyagramı dikkate alınarak açıklanmaktadır. Bu durumda RRT, düzlemdeki bölme noktaları kümesi olarak V'deki düğümler kullanılarak kavramsallaştırılır. Bu şekilde, her düğüm ilgili Voronoi hacmine sahip bir Voronoi bölgesine sahiptir. V seti rastgele numuneler alınarak ve V'den rastgele basit için en yakın komşuyu hesaplayarak oluşturulur. Daha önce de belirtildiği gibi, bu, açık alanlara doğru eğimli bir arama ağacının artımlı bir yapımına yol açar. Bu aslında RRT'lerin doğal Voronoi önyargısıdır ve arama ağacının arama alanını "hızla" keşfetmesine neden olur. Bununla birlikte, her yinelemede arama ağacının Voronoi diyagramını oluşturarak Voronoi yanlılığının tam olarak kullanıldığı bazı uygulamalar mevcuttur (Lindermann vd. 2004). Diyagramı kullanarak, arama alanının hangi bölümlerinin araştırılması gerektiğine kolayca karar verilebilir. Genel olarak, genişleme için xi ∈ V düğümünün seçilme olasılığı, Vi'nin hacmiyle orantılıdır. Arama ağacının açık alanlara doğru eğilmesinin nedeni budur (Samuel, 1959).

RRT, düzgün bir örnekleme dağılımı yoluyla ağaca bir düğüm (V köşeleri kümesi) ekler. V'nin boyutu arttıkça, seçilen en yakın komşunun içine yeni bir düğüm eklendiğinden grafiğin keşfedilmemiş alanı doldurulur. Bu şekilde, eklenen düğümler durum uzayıyla aynı olasılık yoğunluğunu izleyecektir, çünkü açık alanın hacmi, açık alan yavaş yavaş kaplandıkça sıfıra yaklaşır. RRT, bir ağacı aşamalı olarak büyüterek çalışır, örtülü Voronoi sezgisi, ağacın açık alanları hızla keşfetmesini sağlar. Algoritma örnekleme tabanlıdır ve olasılıksal tamlık özelliğine sahiptir, ancak temel algoritma bulunan çözümü geliştirmek için yol maliyetini kullanmaz. Neredeyse kesinlikle bir yetersiz yol bulacaktır. Ağaca rehberlik edebilecek veya hedefi bulma olasılığını artırabilecek bazı varyantlar mevcuttur, ancak RRT yetersiz bir yol izleyecektir.

Giriş: Başlangıç Düğüm $X_{başlangıç,}$ En büyük Düğüm K← N

V← {Xbaşlangıç }; E← *(Ø) ;*

While(sayaç<En büyük düğüm)

$X_{rastgele}$← $NoktaOlustur(i)$

$X_{enYakinDugum}$← $YakınıBul(V, Xrastgele)$

X_{yeni}← $Yonlendir(XenyakinDugum, Xyeni)$

İf(EngelYoksa($X_{enyakındugum,}X_{yeni}$) **then**

$V \leftarrow V \cup \{Xyeni\}$

$E \leftarrow E \cup \{(Xenyakındugum, Xyeni)\}$

End if

End While

Return (V,E)

RRT'nin önceliği aslında oldukça basittir. Noktalar rastgele oluşturulur ve mevcut en yakın düğüme bağlanır. Bir tepe noktası her oluşturulduğunda, tepe noktasının bir engelin dışında olup olmadığını kontrol etmek gerekir. Bitiş noktası alanının içinde yeni bir düğüm oluşturulduğunda veya limit değerlerine ulaşıldığında algoritmanın çalışması sonlandırılır. Yeni düğüm eklerken eklenecek koordinat rastgele belirlenir. Rastgele koordinat belirlemenin farklı metodları vardır. Temel uygulamalar için bu tür yaklaşımlar yeterlidir. Ama kodlama kısmında rastgele değer getiren metodların aslında tam olarak rastgele olmadığı, bir noktaya kadar algoritmik bir değer getirdiği bilinmektedir. RRT algoritmasındaki kritik metodlardan biri de tepe noktası ile en yakın kenar arasındaki en kısa mesafenin hesaplanmasıdır. Kesişim noktasında, kenar hattı üzerinde yeni bir düğüm oluşturulur ve bu rastgele oluşturulan tepe noktasına bağlanır. Bir diğer yol olarak da birbirinden ayrık düğümlerin bir kenarı zincirlenerek en yakın düğümle birleştirilmesidir. Bu diğer yöntem daha hızlı çalışır, çünkü daha az hesaplama gerektirir ve gerçeklenmesi daha kolaydır. Ancak bu yolda bellek kaybı daha fazladır. Birçok noktanın saklanmasını gerektirir. Düğümlerin en yakın komşularına bağlanması beklenir. Oluşan bu çizgenin doğası gereği optimum en kısa yolu bulma olasılığı epey düşüktür. İki nokta arasındaki hipotenüsü almak yerine , bir üçgenin iki bacağı arasında gezinir. Bu açıkça daha uzun bir mesafe, RRT tarafından oluşturulan kübik doğa ve düzensiz yollar sorunu RRT Star ile ele alınır (Kuffner vd. 2000).

Algoritmanın faydası hızı ve uygulamasıdır. RRT algoritması bu alanda diğer en kısa yol algoritmalarıyla kıyaslandığında epey hızlı olduğu görülmüştür. Bellek bakımından sıkıntılı olduğu söylenebilir. Çünkü kendi bünyesinde çok fazla nokta tutar. İterasyonlar uzadıkça en yakın komşuyu bulmak epey zorlaşacaktır (Silver vd. 2016).

3.4 Rapidly Random Tree Star Algoritması

RRT Star algoritması ,durum alanını açık alanlar için bir sapma değeri ile aşamalı olarak aradığı için RRT algoritmasına benzer şekilde çalışır. RRT *, RRT bölümünde tartışılan ilk özellikleri devralır. Bununla birlikte, RRT * algoritması optimal bir çözüme neredeyse kesin bir yakınsama özelliğine sahiptir (Karaman vd. 2011). Bunlardan biri, örnek sayısı canlılığa yaklaştıkça optimal bir çözüme neredeyse kesin

yakınsamadır. Her ne kadar sezgisel tarama kullanarak (Simmons vd. 2003) veya sadece araştırmanın Voronoi diyagramından yararlanarak bir RRT'nin büyümesini optimal bir yol bulmaya yönlendirmek için çeşitli algoritmalar önerilmiş olsa da, bulunan çözümün kalitesi, özellikle de hedefe ulaşmak için en uygun yol gereklidir. (RRT'nin yetersiz yakınsama ile ilgili bölümü dezavantajıdır). Sorun esas olarak RRT'nin yol maliyetlerini dikkate almamasından kaynaklanmaktadır. (Simmons vd. 2003) 'te bir RRT'nin büyümesi, bir düğüme en düşük yol maliyeti göz önünde bulundurularak ve her bir düğüm için tanımlanmış bir kalite ölçüsü kullanılarak yönlendirilir. V'den sadece en yakın komşuyu hesaplamak yerine, en yakın k komşuları dikkate alınır (Riedmiller, 2005). Her ne kadar yol maliyetleri göz önünde bulundurulsa da ve rehberlik daha iyi sonuçlar vermesine rağmen, bu RRT varyantı doğasında olasılıksal yetersizliği korur. RRT star RRT algoritmasının geliştirilmiş halidir. RRT algoritmasındaki yakınsama olasılığının verimini artırmak için yeniden tasarlanmıştır (Karaman vd. 2011). Yarıçapı r olan bir top tarafından tanımlanan k en yakın komşularından Xinit'ten en düşük maliyetli düğümü seçmek için bir yolun maliyeti dikkate alınır. Ayrıca, düğümler arasında minimum maliyetin korunmasını sağlamak için, bir yeniden kablolama işlem yani ağaçtan kenarları kaldırarak ve ekleyerek ağacı yeniden yapılandırır. Daha önce belirtildiği gibi, RRT * algoritması RRT algoritmasına iki önemli değişiklik getirmektedir. Her iki yol da başlangıçtan bahsi geçen düğüme giden yolun maliyetine göre karar verir. RRT *, ağacını RRT'ye benzer şekilde üretir; her yinelemede, Xrand, Xnearest ve Xnew'e RRT için tartışıldığı gibi değerler atanır. Bununla birlikte, RRT * algoritmasının optimal davranışını sağlayan üç yeni prosedürü vardır. RRT *, RRT'nin yaptığı gibi, bir ağacı kademeli olarak büyütür. (Sallans vd. 2004) Yol maliyetine bakan iki ek prosedür eklenmiştir. Üst yordam, üst öğeyi yeni bir düğüm için bir top yarıçapından seçer ve yeniden kablolama yordamı, yeni düğümü dikkate alarak ağacı daha az maliyetli yollar bulmak için yeniden yapılandırır. Bununla birlikte, yakınsama oranı yavaştır, çünkü RRT * tüm durum alanını keşfetmeye devam eder. (Guevara vd.2018)

RRT sınıfının arkasındaki teori, bu örnekleme temelli yönteme ilişkin son gelişmelerin bir özeti ile birlikte tartışılmaktadır. RRT'nin pürüzlü yollar ürettiği ve her zaman en düşük yola dönüştüğü gösterilmiştir. RRT * kademeli olarak optimal yola yakınlaştığı

gözlemlenmiştir. Ayrıca RRT, dar geçitlerden yol bulma yeteneğine sahip değildir. RRT * dar geçitten bir yol bulabilmektedir, ancak yine de bunu yapmakta zorlanmıştır.. İki algoritma aynı zaman karmaşıklığına sahiptir, buna rağmen, hesaplama süresindeki farklılıkları vurgulamak önemlidir. Bu nedenle, farklı planlama alanlarında hesaplama süresi için bir karşılaştırma önerilmektedir, çünkü bazı planlama problemlerinde hız önemli olabilir. (Kuffner vd. 2000)

RRT sınıfının düzenli (kesin) algoritmalara göre önemli bir avantajı, her bir düğüm genişlemesinde holonomik kısıtlamaları (olmayan) dikkate alabilmesidir. Artımlı ve olasılıklı doğası, bunun kesin algoritmalardan daha hızlı olmasını sağlar. (Hauer, 2006)

Giriş: Başlangıç Düğüm $X_{başlangıç,}$ En büyük Düğüm K← N

V← {Xbaşlangıç }; E← *(Ø) ;*

While(sayaç<En büyük düğüm)

$X_{rastgele} \leftarrow NoktaOlustur(i)$

$X_{enyakinNokta} \leftarrow EnYakınNoktaBul(V, Xrastgele)$

$X_{yeni} \leftarrow Steer(XenyakınNokta, Xyeni)$

$V \leftarrow V \cup \{Xyeni\}$

if(EngelYoksa($X_{enyakinnokta,}X_{yeni}$) **then**

$X_{yakindaki} \leftarrow YakiniBul(V, Xyeni)$

$X_{enyakinnokta} \leftarrow$
$AtasiniGetir(Xyakindaki, Xenyakindaki, Xyeni)$

$E \leftarrow E \cup \{(Xenyakindakinokta, Xyeni)\}$

$E \leftarrow YenidenBagla\{Xyakindaki, E, Xyeni\}$

End if

End While

Return (V,E)

3.5 Probabilistic Road Map Algoritması

Probabilistic Road Map Algoritması, RRT'ye benzer şekilde, belirli bir alanda sınırlı sayıda rastgele nokta üretir. Yeni bir düğüm oluşturulduktan sonra, PRM algoritması düğümleri bağlı bileşenlere kümeler. Düğümün ait olduğu bağlı bileşenin saklanması gerekir. Kümeleme gerçekleştirmek için, rastgele üretilen düğümün sabit bir yarıçapı içindeki tüm düğümler toplanır. Bu komşu düğümler, mesafeyi (veya istenen metriği) artırarak sıralanır. Sıralı komşu düğümlerden döngü yaparak, rasgele oluşturulan düğümün incelenen düğümle aynı kümede olmadığı kontrol edilir. Koşul sağlanmışsa kümeye yeni düğüm eklenir. Bir düğüm birden çok kümeye ait olabilir ve olmalıdır. (Nair vd. 2010) Yarıçapın boyutu kullanıcı tarafından verilir ve değeri yol haritasının yapısını tanımlar. Oluşan kümelerin bazıları bu yarıçapa sahip dairesel lekelerden olacaktır. Seçilen yarıçap aynı zamanda üretilen yol haritasının hızını ve performansını etkileyecektir. Daha geniş bir yarıçap, daha fazla komşudan ayrıştırmayı ve küme ilişkilerini belirlemeyi içerir. Kullanıcının belirlediği bir diğer parametre düğüm sayısıdır. PRM algoritması, istenen düğüm sayısı üretildiğinde sona erer. Çok fazla düğüm oluşturmak, yol haritasını geliştirmek için gereken süreyi uzatabilir. Komşu bölgelerin incelenmesi için daha fazla zaman ayrılmalıdır. Çok az düğüme sahip olmak, kırık bir grafikle sonuçlanabilir. Özellikle yüksek yoğunluklu engel bölgelerindeki kümeler yol haritasının geri kalanından ayrılabilir. Seyrek olarak oluşturulmuş bir yol haritası üzerinde oluşturulan yollar da daha düzensiz olabilir. En kısa yol planlama algoritması kullanıldığında daha az düğüm olası yolları azaltır. (Choset, 2019)

$V \leftarrow (\emptyset)$

$E \leftarrow (\emptyset)$

While($|V| < n$)

Tekrarla

q← $Alandaki\ rastgele\ bir\ nokta()$

q ya çarpışma olmayana **kadar**

$V \leftarrow V \cup \{q\}$

End While

While $\boldsymbol{q} \in \boldsymbol{V}$

$N_q \leftarrow$ $Mesafeye\ göre\ V'den\ seçilen\ q'nun\ en\ yakın\ komşuları$

While $q' \in \boldsymbol{N}q$

İf $(q,q') \notin E\ AND\ \Delta(q,q')! = NULL$ **then**

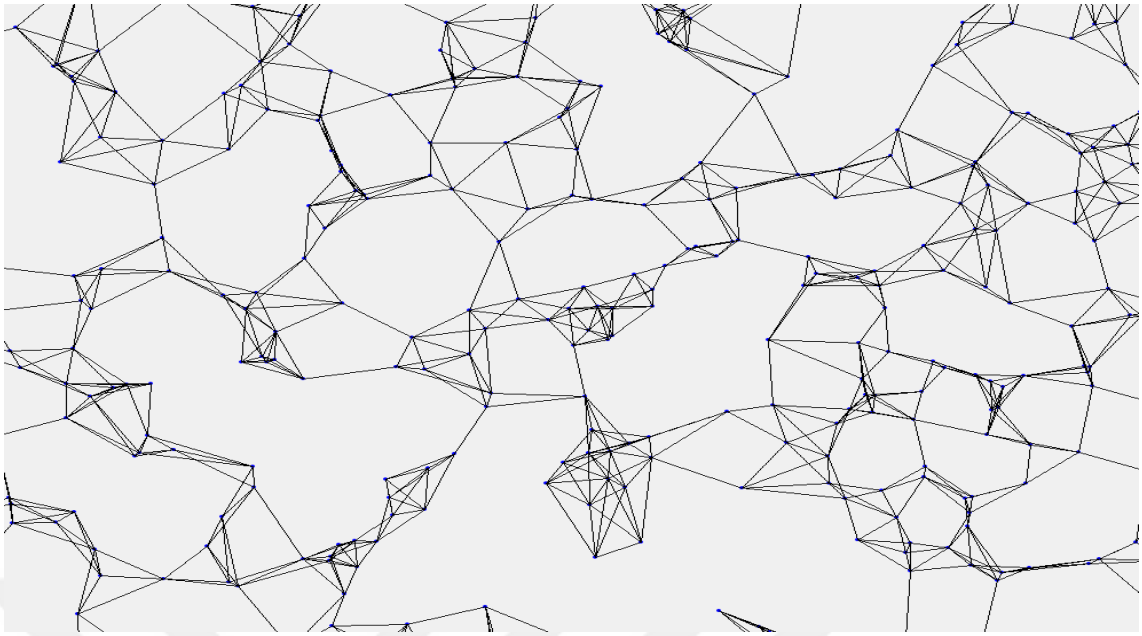
$E \leftarrow E \cup \{(q,q')\}$

End if

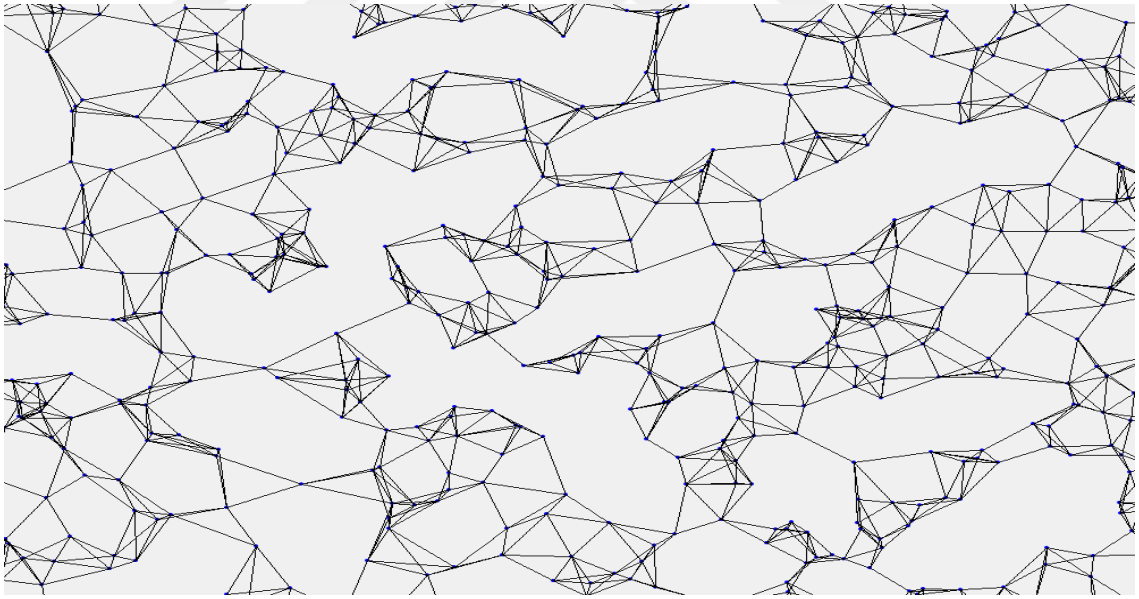
End While

End While

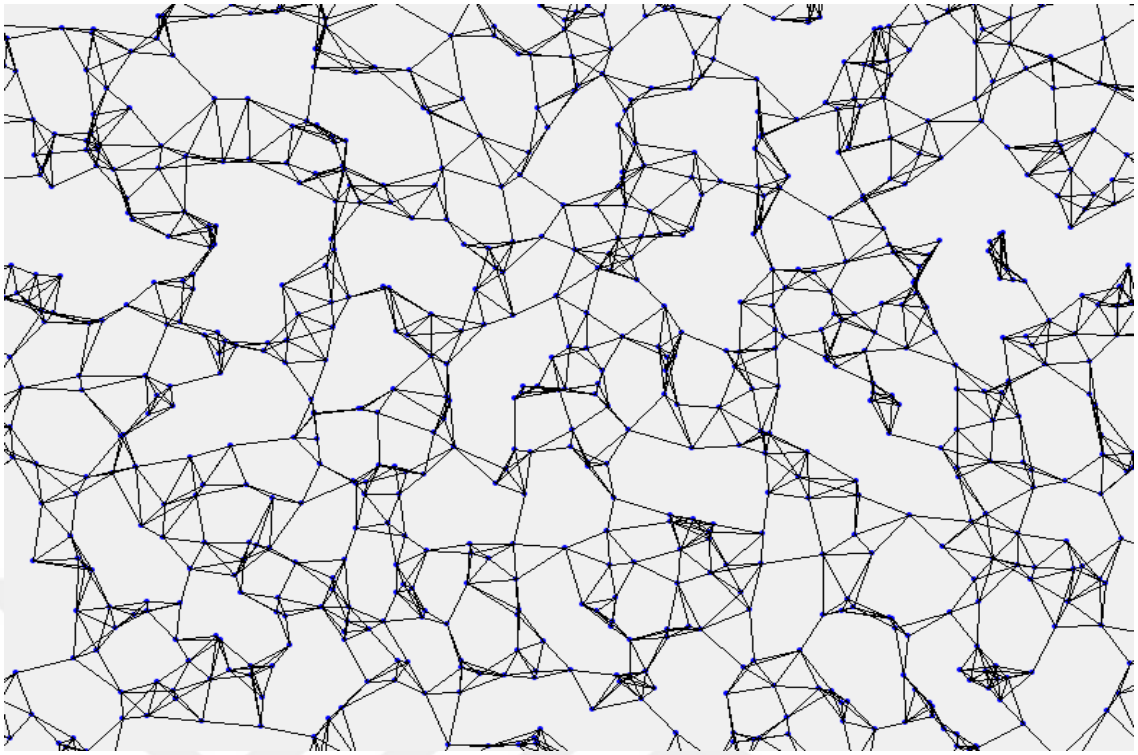
Çalışmada PRM algoritmasının görselleştirmesi de yapılmıştır. Farklı nokta yapısı ve sıklıklarında PRM algoritması çalıştırılıp görselleştirilmesi kaydedilmiştir. Son aşama olarak primler birbirine bağlanmıştır.



Şekil 3.6 400 Noktalı PRM Algoritması Gösterimi



Şekil 3.7 800 Noktalı PRM Algoritması Gösterimi



Şekil 3.8 1600 Noktalı PRM Algoritması gösterimi

3.6 Potential Field Algoritması

Bir robotun otonom navigasyonu, robotun amacına ulaşma yeteneğine dayanarak çevredeki engellerden kaçınma yeteneğine bağlıdır. Bazı durumlarda robot, çevresini tam olarak bilir ve hareketini buna göre planlar. Temel olarak mobil robotun varış noktası hakkında bilgisi vardır. Hareketli engel bilgisini robot kendi sensörlerini kullanarak edinir.

Yüksek seviyeli işlemeyi, yüksek seviyeli planların oluşturulduğu planlama seviyesi, robotun sensörler tarafından algılanan engellerden kaçınması gereken reaktif kontrol olarak düşünülebilir. Yerel yol planlama çalışma zamanında hesaplanmalı ve plan ve engellere göre önceliklendirilmelidir. Çalışma anında yol hesaplanmasının nedeni hareketli engellerden de kaçınabilmektir.

Potential Field algoritması yerel yol planlama metodlarından bir tanesidir. Basitliği sebebiyle ilgi gören bir metoddur. Buna rağmen dezavantajları da vardır. Mobil robot

bazı durumlarda yerel minimuma düşebilir. Bu problemden korunmak için bazı çözümler geliştirilmiştir.

Bu metodda temel amaç suni bir potansiyel alan yaratarak mobil robotu istenilen tarafa çekebilmektir. Yeterli potansiyel alanı tasarlanarak robotun basit davranışlar sergilemesini sağlanabilir.

Örnek olarak bulunulan uzayda hareketli ya da sabit herhangi bir engelin bulunmadığı bir durum oluşturulur. Bunu geleneksel planlamada yapmak için, robotun hedefe göreli konumunu hesaplamalı ve ardından robotu hedefe götürecek uygun kuvvetleri uygulamalıdır.

Birinci olarak potansiyel alan yönteminde, varış noktasının içine girerek çekici bir manyetik alan oluşturulur. Potansiyel alan yaklaşımında, öncelikli olarak tüm boş alanda tanımlanır ve her bir iterasyonda, robot kendi olduğu konumda kendisine etkiyen potansiyeli hesaplar ve sonra bu manyetik alan tarafından uyarılmış kuvvet hesaplanır. Robot sonraki adımda kendisine etkiyen net kuvvete göre hareket eder .

Mobil robotun engellerden kaçınabilmesi için başka çözümler de bulunmaktadır Her engelin etrafında itici bir alan oluşturması sağlanmalıdır Her iterasyonda engele yaklaşan mobil robot olursa , bu mobil robotu engelden uzaklaştırmak için itici bir güç oluşturulur. Robot birçok kuvvetin etkisinde kendi rotasını oluşturmaya çalışır. Engellerden gelen itici güç hedefin oluşturduğu çekici güçle beraber mobil robotlar engellere çarpmadan hedefe varmaya çalışır. (Latombe vd. 2019). Formül(3.4) matematiksel olarak, genel potansiyel alanın formülize edilmiş halidir. Hedef ve her bir kesme alanının toplanmasından oluşur.

$$U(q) = Ugoal(q) + \sum Kesme(q) \qquad (3\text{-}4)$$

(3.5) formülü ile indüklenen kuvvet sunulmuştur. Potansiyel alanın negatifi şeklindedir.

$$\boxed{F = -U(q) = (\frac{U}{x}, \frac{U}{y})} \quad (3\text{-}5)$$

Robot hareketi, yerel kuvvet tarafından yönlendirilen küçük adımlar atılarak elde edilebilir.

$$\boxed{Ugoal = dist(q, hedef)} \quad (3\text{-}6)$$

Literatürde birçok potansiyel alan fonksiyonu incelenmiştir. Tipik olarak Formül(3.6) parabolik bir çekicidir. Burada dist, q durumu ile hedef arasındaki Öklid mesafesidir.

$$\boxed{Uobtsacles = dist(q, engel) - 1} \quad (3\text{-}7)$$

Tek engellerde, robot engele yaklaştığında sonsuza kadar yükselen potansiyel bir engel olarak modellenişi Formül(3.7) de sunulmuştur. Burada dist, q durumunda robot ile engel üzerindeki en yakın nokta arasındaki Öklid mesafesidir.

$$Uobstacles = \beta \left(\frac{1}{dist(q,obstacle)} - \frac{1}{p}\right)^2 dist(q, goal)^2 \quad \boxed{if\ dist(q, obstacle) \leq p} \quad (3\text{-}8)$$

İtici kuvvet, en yakın engele göre hesaplanır veya ortamdaki tüm engellerin üzerinden toplanır. Bu çalışmada daha sofistike bir itici güç kullanılacak Formül(3.8) şekillenmiştir.

Potansiyel alan yaklaşımı, robotun hareketin olduğu andaki algıladığı engellere göre çalışma anında çarpışma önleme stratejisi oluşturur(Knudsen vd. 2003). Ve yerel bilgilere güvenmesinin, onu yerel bir minimumda yakalayabileceği açıktır. Bu algoritmanın çakıldığı durumlardan biri U şeklinde oluşturulan tuzaklardır. Başka bir durumda tamamen engel olan uzun bir duvar senaryosudur. İtici güçten dolayı bu uzun engeller hiçbir zaman aşılamayacağından dolayı robot asla hedef noktasına

ulaşamayacaktır (Mnih, 2015). Potansiyel alan yönteminin kronik sorunu olan yerel minimum problemi ile başa çıkmak için yeni çözümler önerilmiştir. Yerel minimum probleminden kaçınmanın başka bir yolu da potansiyel alanı yüksek rütbeli planlayıcı ile birleştirmektir. Dolayısıyla mobil robot sensörlerinden elde ettiği verileri bu alanda kullanabilir. Başka bir çözüm yolu da mobil robotun yerel minimum statüsüne geçmesine izin vermektir. Ancak burada çözüm robotun yerel minimum durumuna geçtikten sonra başlar. Bu duruma aşağıdaki şekilde çözüm önerileri getirilmiştir (Latombe vd. 2019).

- Yerel minimum probleminden kurtulmak için başka metodlar ve çözüm yolları denenmelidir.
- Rastgele hareketler yaparak yerel minimum alanından kurtulmaya çalışılmalıdır.
- Yargılayıcı bir planlayıcı kullanılabilir ya da yerel minimum bölgesinden kurtulabilmek için farklı hata algoritmalarını kullanmak.
- Daha komplike potansiyel alanlar kullanılabilir. Örnek olarak harmonik potansiyel alanlar verilir. Bu alanlarda yerel minimum tamamen serbesttir.
- Yerel minimum alanında kalan mobil robotun üzerine etkiyen potansiyel alan kuvvetleri değiştirilebilir. Böylece robot yavaş yavaş ondan kurtulabilir (Fukushima, 1988).

Tüm bu yaklaşımlar, robotun sıkışıp kaldığını keşfedebildiğine dayanıyor, bu da kötü tanımlanmış bir problemdir. Bu çözüm yollarının ortak sonucu potansiyel alan değerlerinin değiştirilmesidir.

Robot tuzağa düştüğünü algıladığında, ona sanal boş alan kuvveti (ya da sadece sanal kuvvet) adı verilen yeni bir kuvvet uygulanacaktır (Hastie, 1999). Bu yapay güç mobil robotun etrafındaki boşlukla orantılıdır. Bu güç robotu yerel minimumda sıkışıp kalmaktan kurtarabilir. Yapay güç mobil robota vurmaya başladıktan sonra robot yerel minimum bölgesinden kurtulabilecektir. Robot prosedürel bir planlayıcı ile tekrar harekete geçecektir. Ancak, yine aynı yerel minimum değere hapsolması pek olası değildir.

$$F_f = F_{cf}(\cos(\theta)\, e_x + \sin(\theta) e_{y)} \quad (3\text{-}9)$$

Daha resmi olarak, formül(3.9) da görüldüğü gibi sanal kuvvet Frobotun çevresindeki boş alanla orantılıdır.

Son olarak, robotun sıkıştığını tespit etmek için, robotun mevcut pozisyonunu (açık döngü) tahmin etmek için bir konum tahmincisi kullanılır. Eğer lokasyon belirli bir aralık boyunca hiç değişmezse , robotun yerel minimumdan kurtulabilmesi için suni bir kuvvet devreye sokulur.

Robotun süratinin ve ivmesinin engele yaklaştıkça düştüğü bilinmektedir. Bu nedenle, üst düzey planlayıcı tarafından planlanmamışsa, robot uzun süre yerel olmayan minimum bir yerde kalamaz.

Potansiyel alan yöntemlerinde yerel minimumdan kaçınmak için bir sanal kuvvet yöntemi önerilir. Potansiyel alan, basitlik, gerçek zamanlı hesaplama avantajlarına sahip olduğundan hareket planlaması için temel platform olarak kullanılır. Bu engellerin oluşturduğu yerel minimum bölgelerinden çıkmak için suni güç terimi ortaya konulmuştur. Suni kuvvet farklı yerel minimum problemlerinden kurtulmak için ortaya konan bir çözümdür (Mousas, 2013).

Bu yaklaşım birçok durumda sorunu çözmeyi başarsa da, birkaç durumda başarısızlığı saptanmıştır:

1. Kuvvet bileşeninin eklenmesi, aksine bir uygulama olmasa da, kaba bir sezgisel gibi görünmektedir. Kuvvet gücünü ve yönelimini tanımlamak bir bilimden ziyade sanattır.

2. Gücün eklenmesi, robotun engel fobisine başlama gibi garip davranışlar göstermesine de neden olabilir. Yolunu tamamen engellemeseler bile her zaman engellerden kaçan bir robot biraz gariptir.

3. Suni kuvvet yoluyla çizilen güzergah en kısa yol değildir. Orada öncelikli amaç yerel minimumdan kaçınabilmektir.

4. Sanal kuvveti eklemek, robotun yerel minimumda olduğunu tespit etme yeteneğine dayanır. Bu yöntemi pratikte uygulamak epey güçtür. Robotun yerel minimumda kaldığını anlamak ayrı bir problemdir. Bunun için eklenecek kontroller algoritmaya ayrı bir yük getirir (McCarthy vd. 1956).

5. Yerel minimumdan kaçınılması garanti edilmez, kuvveti ve yönü ortam boyutlarına uygun değilse bu meydana gelebilir.

6. Bu gücün eklenmesi, orijinal potansiyel alan tarafından çözülebilen, mümkün olmayan durumlara da neden olabilir. Örnek olarak engel arasında çok ufak bir boşluğun olduğu ve hedefe gidilmesi için bu boşluktan geçilmesi gereken durum.

Sezgisel yaklaşım aslında birçok problemi çözebilir ancak her zaman optimum yolu garanti edemez. Sezgisel çözümden ziyade sistematik yaklaşımlar da bu problemler için tercih edilebilir. Örneğin, robot bir engele çarptığında hata algoritmasını kullanırsa, yerel minimumdan kaçınabilir (Çelik vd. 2015).

3.7 Dynamic Windows Approach Algoritması

Daha önceki gerçek zamanlı engellerden kaçınma yaklaşımlarının çoğu yapay potansiyel alanlara dayanıyordu. Robot engellerin ve hedefin oluşturduğu kuvvetlerle varış noktasına ulaşmaya çalışır. Bu yaklaşımla oluşturulan yörüngeleri düzeltmek için çeşitli uzantılar önerilmiştir. Yapay potansiyel alan yaklaşımları hesaplama açısından verimli olsa da, robot hedef pozisyona ulaşmadan önce yerel minimuma takılabilir. Bunun nedeni, hareketi belirlemek için boş alanın bağlantısı hakkında hiçbir bilginin kullanılmamasıdır (LeCun vd. 1998).

Bu yaklaşımı, parametreli yol ailelerini veya daha spesifik olarak açı alanlarını genişletmek, bir hareket seçerken robotun holonomik olmayan kinematik kısıtlamalarını dikkate alır. Bu durum taranacak bölgeyi azalttığından algoritma verimini artırabilir (Nattharith vd. 2016).

Gerçek zamanda yön bulma algoritmalarından Eğrilik-hız yöntemi ve dinamik pencere yaklaşımı(DWA) bulunmaktadır. Bu algoritmaların temeli dönüş açısı yaklaşımına dayanmaktadır. Bu yaklaşımda işin içine mobil robotun teknik limitleri ve fiziksel kısıtlamalar da devreye girer. Dinamik pencere yaklaşımına kinematik limitler de devreye girer (Kobayashi vd. 2003). Bu yaklaşımlar yüksek hızlarda engellerden kaçınmak için çok iyi sonuçlar vermesine rağmen, yerel minima problemi devam etmektedir.

Dinamik pencere yaklaşımı(DWA) hareket planlayıcısı ile bütünleşik çalışmaktadır (Petrovic vd. 2007).

Engellerden kaçınmak için Bayesci bir yaklaşım, küresel yol planlaması ile bağlantılıdır. Ancak, bu yaklaşımlar bir hareket komutunun yürütülmesi için çevre hakkında önceden bilgi gerektirir. K sayıda robot için hareket planlama algoritması sunulmaktadır.

Başka bir çözüm önerisinde ise engellerin oluşturduğu potansiyel alanlardan uzaklaşma ile birlikte hareket yönü ve güzergahının oluşturulması ve uygulanabilmesi için çerçeve yöntemi ile birlikte kullanılması yaklaşımı türetilmiştir.

Bu çerçevenin kullanılmasındaki asıl amaç ise bölgede oluşabilecek çok küçük engel ya da değişiklikleri tolere edebilmesidir. Bu çerçeve bulunduğu bölge hakkında bilgi sahibi olması gerekir.

Dinamik pencere yaklaşımı(DWA), gerçek zamanlı çalışabilen robotun kinematik ve dinamik limitlerine göre rotasını belirleyen algoritmik bir yaklaşımdır. Kinematik

limitler robotun teknik özellikleri ile doğrudan ilişkilidir. Arama alanı, robot tarafından ulaşılabilen translasyon hızları v ve dönme hızları t kümeleri (v, w) 'dir.

Tüm hız ikilileri arasında, mevcut konum, mevcut hız ve robotun hızlanma yetenekleri göz önüne alındığında, robotun bir engele çarpmadan önce durmasına izin veren seçilir. Bu hızlara kabul edilebilir hızlar denir (Brock vd. 1999).

Aramayı dinamik bir pencereyle sınırlamak, robotun dinamik sınırlamalarına göre arama alanını daha da azaltır (Güzel vd. 2017). Dinamik pencere, mevcut hızı ve hızlanma yetenekleri göz önüne alındığında, belirli bir zaman aralığında robot tarafından elde edilebilecek hızları içerir.

Çeviri ve yönlendirme için ivme yetenekleri bağımsız olduğu için dinamik pencere bir dikdörtgendir. Bir sonraki hareket komutunu belirlemek için dinamik pencerede kabul edilebilir tüm hızlar dikkate alınır (Huang vd. 2017). Bunlar arasında, bir engele ulaşılana kadar robotun hedefle hizalamasını ve yörünge uzunluğunu en üst düzeye çıkaran bir hız seçilir.

Bu yaklaşımı kullanarak, yüksek hızlarda güçlü engellerden kaçınma davranışı gösterilmiştir. Bununla birlikte, dinamik pencere yaklaşımı yalnızca hedef başlığını dikkate aldığından ve boş alan hakkında bağlantı bilgisi olmadığından, yerel minimaya karşı hala hassastır.

Arama Alanı: Dinamik Pencere Yaklaşımında taranacak alan belirlenen çerçeve içinde kalan hız kümesinden oluşur.

Sabit bir dizi yön ve skaler hız seçerek kutupsal koordinatlarda ayrık hale getirilir.

Küresel bir referans çerçevesinin kullanılması, sabit hızlanma a ve hız w, için x ekseni için (3.10) hareket denklemini veren iki translasyon ekseninin ayrılmasına izin verir:

$$X(t_i) = X(0) + V_z t_i + \int_0^{t_i} a_z t dt = x(0) + v_z t_i + 1/2 a_x {t_i}^2 \quad (3\text{-}10)$$

(3.10) denkleminde belirli bir hız komutunu elde etmek için sabit bir hızdan hızlanırken robotun, istenen hıza ulaşılana kadar ikinci dereceden bir eğri tanımladığını gösterir. İvmenin değerine göre eğrilerin yapısı değişir. Yüksek hızlanmalar daha keskin eğriliğe neden olur.

Yörüngenin süresi, hareket formüllerine göre hareketin engele çarpana kadarki simülasyonu olarak tanımlanır. Yörüngenin uzunluğu analitik olarak (3.11) formülünden çıkarılabilir.

$$l(\underset{V}{\rightarrow}, \underset{a}{\rightarrow}, t_i = \int_0^{t_i} V_{t_i}\, dt$$

$$= \int_0^{t_i} \sqrt{(V_z + a_z t)^2 + (V_y + a_y t)^2 dt} \quad (3\text{-}11)$$

(3.11) formülünde yörüngenin uzunluğu, robotun bir tik süresi boyunca hareket ettikten sonra durmasına izin veriyorsa, hareket komutunun kabul edilebilir olduğu kabul edilir. Bu oluşsan fonksiyon bahsi edilen üç fonksiyonun birleştirilmesiyle oluşur. Bu fonksiyonun değer aralığı [0,1] aralığına yakınsamıştır (Fox vd. 1997).

Hedef başlığı dinamik pencere yaklaşımının önemli kriterlerindendir. Bu değer ,robotun engele olan yanal mesafesi çok aza düşerse değiştirilmelidir.

Velocity(hız) işlevi (3.12) formülünde şu şekilde özetlenmiştir.

$$vel(\vec{V}) = \begin{cases} \frac{\|\vec{v}\|}{V_{max}} & Robot\ Varış\ Noktasına\ Uzaksa \\ 1 - \frac{\|\vec{v}\|}{V_{max}} & Robot\ Varış\ Noktasına\ Yakınsa \end{cases} \quad (3\text{-}12)$$

(3.12) formülünde robot hedeften uzaksa yüksek hızları ve yakın olduğunda düşük hızları tercih edecektir. Hareket komutundan (G, 2) kaynaklanan yörünge hedef bölgeden geçerse, ikili fonksiyon hedefinin değeri (p ', G, Z) 1'dir, aksi takdirde 0 olur. P, y ve a parametreleri robotun engellere göre çizeceği güzergahı değiştirmek için kullanılabilir.

Algoritmanın performansı dinamik ve teknik kriterlere göre çok değişkendir. Bağlı olduğu parametreler çok fazladır (Fox vd. 1997).

Engelden kaçınmaya yönelik dinamik pencere yaklaşımı holonomik robotlara genişletildi. Bu tür robotların artan manevra kabiliyetinden yararlanarak, dinamik ortamlarda yüksek hızlarda engellerden kaçınma yapılabilir. Dinamik pencere yaklaşımının da kendi içinde değişik türevleri vardır. Holonomik pencere yaklaşımı efektif bir hareket planlama metodu ile entegre edilirse ortaya küresel dinamik pencere yaklaşımı çıkar. Küresel form robot yön bulma çözümleri arasında etkili bir yöntemdir. Küresel çerçeve yaklaşımı, lokal engellerden korunma ile birleşmelidir. Küresel dinamik pencere yaklaşımı özellikle kestirilemeyen ve sürekli değişen uzaylar için çok uygundur. Robotun bulunduğu ortamın bilgisi olmadan robotun ileri süratte ve güvenli bir şekilde varış noktasına hareket etmesini sağlar.

İlk olarak, mevcut konumumuza ve hedefe dayanarak hedefe istenilen hızı hesaplayabiliriz. (örneğin uzaktaysak hızlı git, yakınsak yavaş ol. Hareket Denklemlerini kullanın, bkz. Mobil robot için dairesel hareket).

$$[V - A.t, v + A.t] \qquad (3\text{-}13)$$

Araç dinamikleri için izin verilen hızlarda (doğrusal 'v' ve açısal 'w') kullanılmalıdır. (3.13) formülünde görüldüğü üzere geçerli hızdan robotların maksimum yavaşlaması ile zaman dilimini çarpılarak çıkarıldıktan sonra geçerli hıza, robotların maksimum hızlanması ile zaman dilimi toplanarak formülize edilmiştir. Aynı şekilde açısal hız için de çıkarılır.

İzin verilen tüm hızlarda arama yapılır. Her hız için, önerilen robot hızı (yani yörünge boyunca çarpışma tespiti) için en yakın engeli belirlenir. En yakın engele olan mesafenin robotların kırılma mesafesi içinde olup olmadığını belirlenir. Robot zamanında duramazsa, önerilen bu robot hızını dikkate alınmamalıdır.

Aksi takdirde, hız 'kabul edilebilir' olduğunda, ardından objektif fonksiyon için gerekli değerleri hesaplanabilir. Önerilen hız için 'maliyeti' hesaplanmalıdır. Maliyet şu ana kadar diğerlerinden daha başarılı ise, bu en iyi seçenek olarak kaydedilmelidir ve ayarlanmalıdır. Son olarak, istenen robotlar en optimum önerilen hıza ayarlanmalıdır.

4. DENEYLER

Bu çalışmada algoritmaların karşılaştırılması için bir uygulama geliştirilmiş olup, bu algoritmaların hepsi bu uygulamada gerçeklenmiştir. Uygulama üzerinden algoritmalar çalıştırılıp birbirleri ile karşılaştırılmıştır. Karşılaştırılan bu algoritmaların sonuçları uygulamada tutulup, kullanıcı arayüzüne yansıtılmıştır. Deneyler sabit engelli deneyler ve hareketli engelli deneyler olarak ikiye ayrılmıştır. Çünkü algoritmaların yalnızca bazıları hareketli engel senaryolarında çalışmaktadır. İlk olarak sabit engel testlerinde A Star ve Lee Algoritmaları karşılaştırılmıştır(Bknz. 4.1.1). Aynı kategoride RRT ve RRT star algoritmaları da birbirleri ile karşılaştırılmıştır(Bknz. 4.1.2). Hareketli engel deneylerinde ise Potential field ve Dynamic Window Approach algoritmaları birbirleri ile karşılaştırılmıştır (Bknz. 4.2). Son olarak tüm algoritmalar kat edilen yol bakımından birbirleri ile karşılaştırılmıştır (Bknz. 4.3). Tüm deneyler aynı bilgisayar üzerinde yapılmıştır. Deney yapılan bilgisayar donanım özellikleri şu şekildedir.

- İşlemci:Intel® Core™ i7-9750H CPU @ 2.60 (12 CPUs)
- Bellek :16384 MB RAM
- BIOS:1.07.04 TFB1
- İşletim Sistemi: Windows 10 Pro 64 Bit
- Ekran Kartı: GeForce GTX 1050

Yazılımın geliştirildiği ortam şu şekildedir.

- Jdk 8u261 64 Bit

Sabit engel deneylerinde deneyler her bir algoritma için bir kez yapılmış. Sonuçlar raporlanmıştır. Hareketli engel deneylerinde ise hareketli engelleri aynı senaryoda yakalayabilmek için deneyler defalarca tekrar edilmiştir.

4.1 Sabit Engellerde Yön Bulma Algoritma Karşılaştırmaları

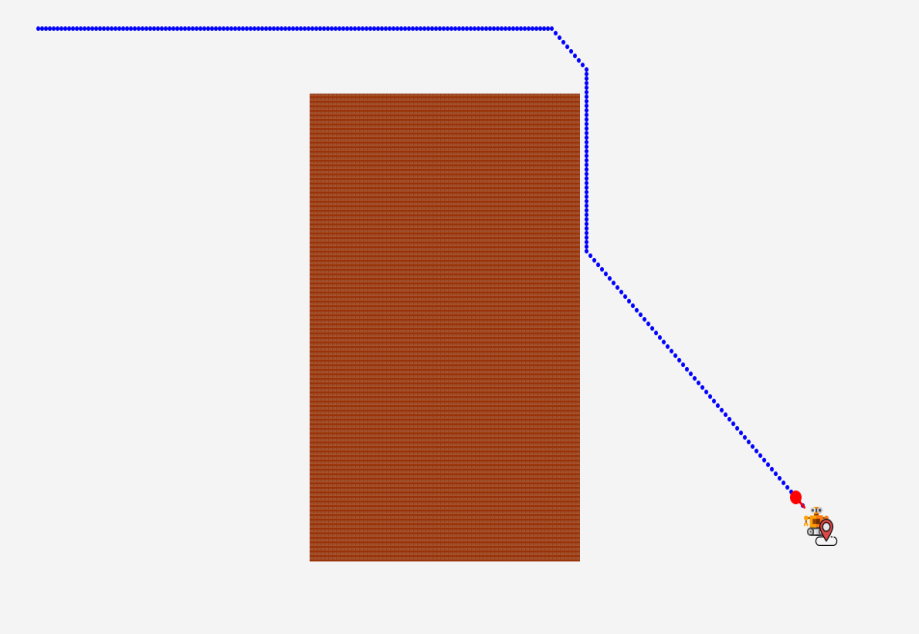
Sabit engellerden kaçınan algoritmalar arasında çalışmada incelenecekler arasında RRT, RRT Star, A Star ve Lee algoritmaları bulunur. Bu algoritmaların ortak özelliği rotayı

önceden hesaplayıp getirmeleridir. Yani dinamik zamanda güzergah bulunmadığından dolayı hareketli engellerden kaçınma olayı da yoktur. Dolayısıyla çalışma bu algoritmalar üzerinde sabit engellerin farklı senaryolarında çalıştırılıp karşılaştırılmış ve sonuçlar elde edilmiştir.

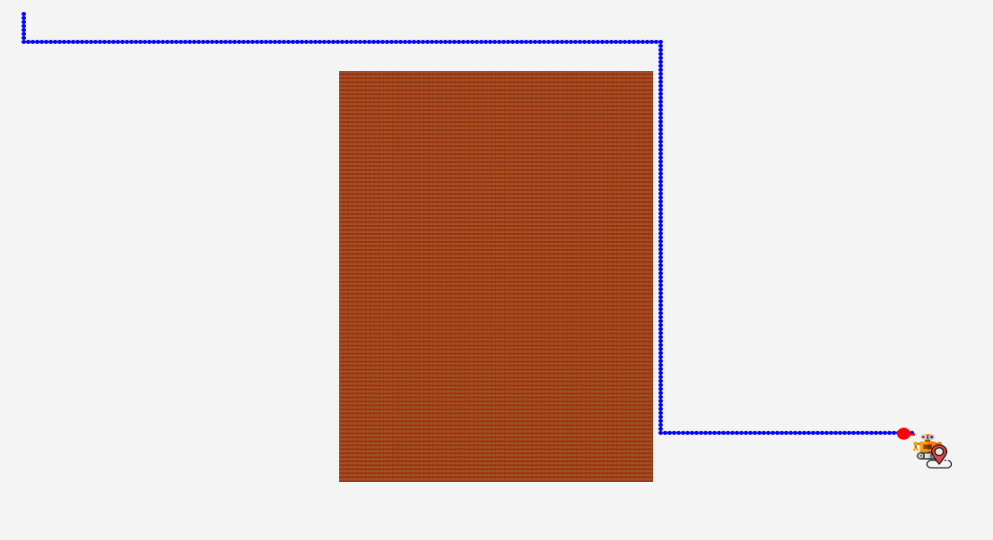
4.1.1 A Star ve Lee algoritma kıyaslanması

A Star ve Lee algoritmaları çalışmada java yazılım dilinde gerçeklenmiş ve sonuçları yazılım üzerinde görüntülenmiştir. A star algoritmasının gerçeklenmesinde robotun aşağı , yukarı, sağa,sola ,sağ üst çarpraz, sol üst çarpraz ,sağ alt çarpraz, sol alt çarpraz şeklinde gidiş güzergahları tanımlanmıştır. Ancak Lee algoritmasının gidiş güzergahında çarpraz yönler tanımlanmamıştır. Farklı senaryolarda bu iki algoritma farklı açılardan karşılaştırıldı.

Tek Engel Senaryosu: Tek sabit engelde iki algoritma açısal hız, kat edilen yol ve harcanan zaman açılarından farklı grafiklerle karşılaştırılmıştır. Senaryolar iki algoritma içinde aynı başlangıç noktasından başlamış, aynı bitiş noktası belirlenmiştir. Engellerde aynı kullanılmıştır. Kısacası tüm senaryo aynıdır.

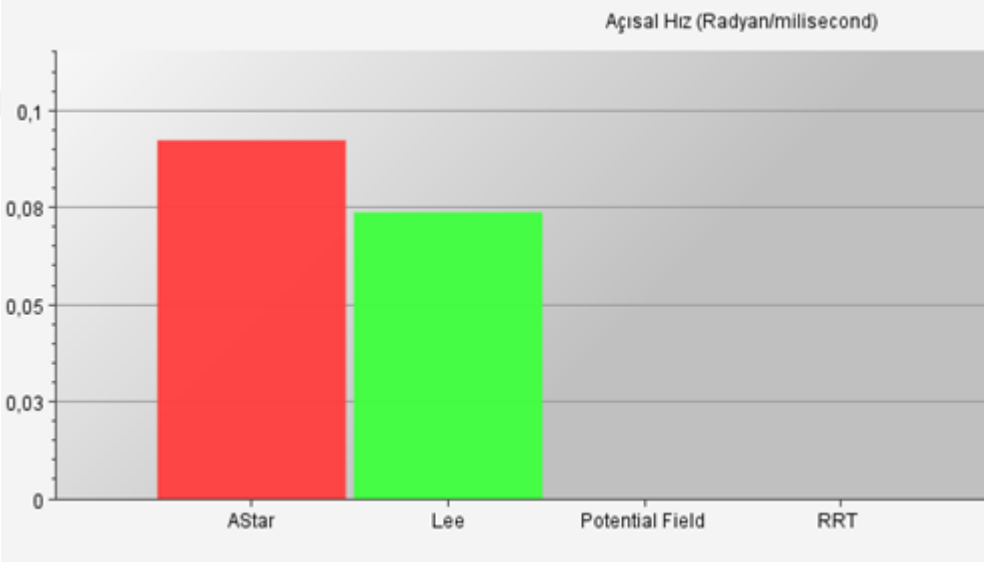


Şekil 4.1 A Star Tek Engel Başlama Noktası(30,30)–Bitiş Noktası (1620,845)



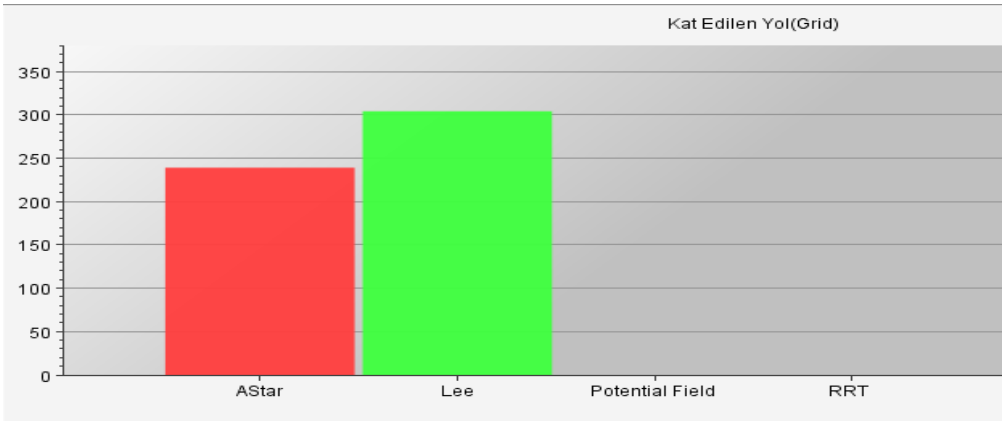
Şekil 4.2 Lee Tek Engel Başlama Noktası(30,30)–Bitiş Noktası (1620,845)

Açısal Hız: Bu senaryoda iki algoritmanın açısal hızı aşağıdaki grafikte gösterilmiştir.



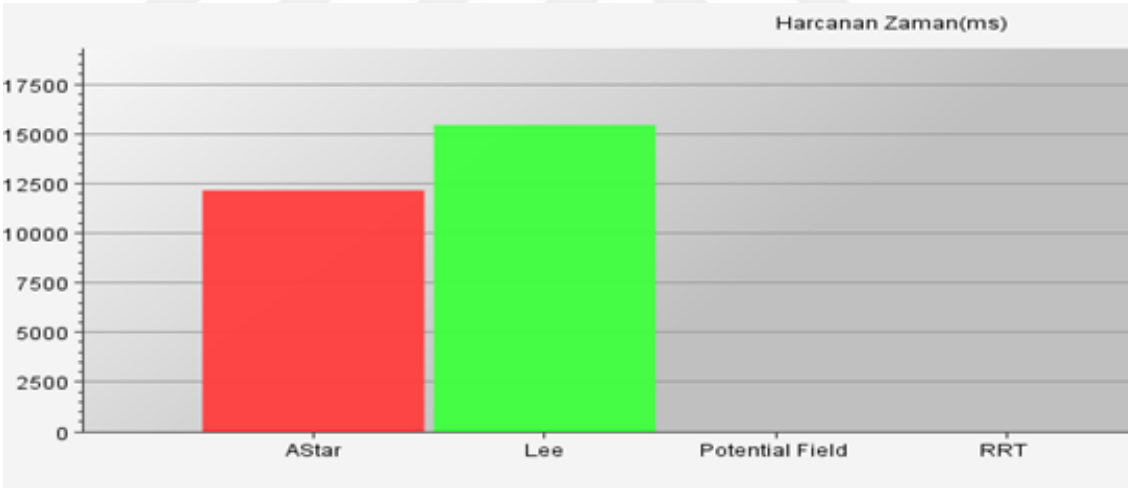
Şekil 4.3 Tek Engelde AStar ve Lee Algoritmaları Açısal Hız Karşılaştırmaları

Kat Edilen Yol: Tek engel kat edilen yol grafikleri şekildeki gibidir.



Şekil 4.4 Tek Engelde AStar ve Lee Algoritmaları Kat Edilen Yol Karşılaştırmaları

Harcanan Zaman: Tek engel senaryosunda harcanan zaman karşılaştırılması aşağıdaki şekildedir.

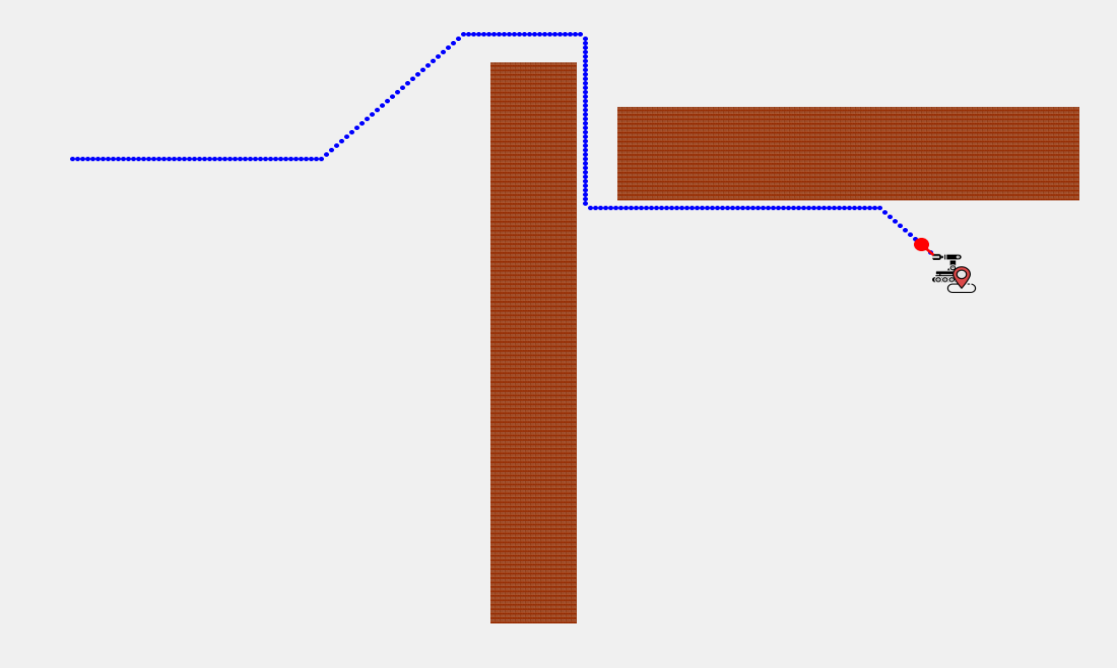


Şekil 4.5 Tek Engelde AStar ve Lee Algoritmaları Harcanan Zaman Karşılaştırmaları

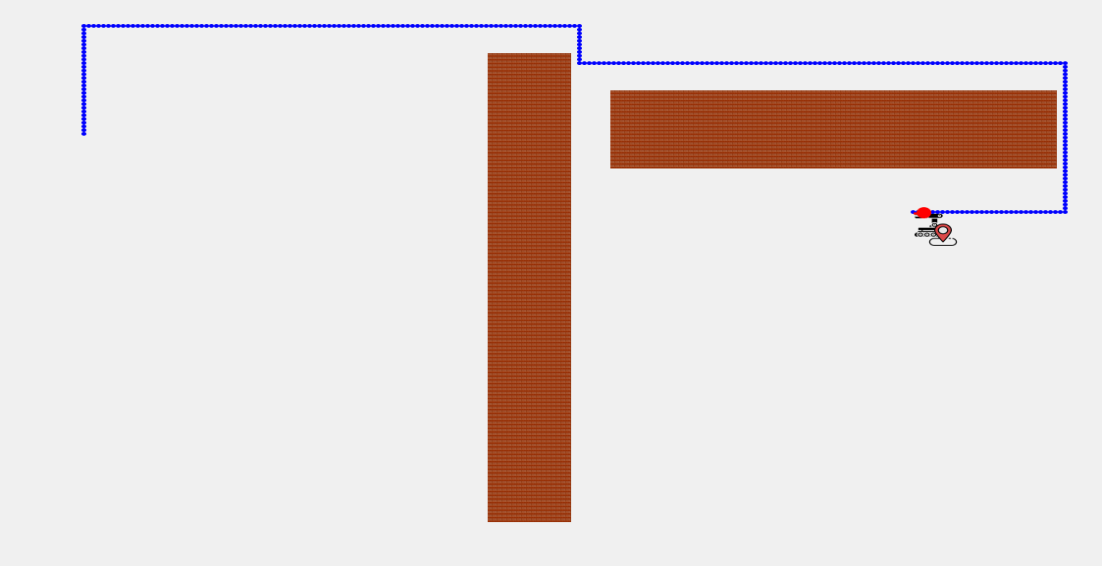
Çizelge 4.1 Tek Engel AStar ve Lee Sonuçları

	Açısal Hız(Radyan/milisecond)	Harcanan Zaman(ms)	Kat Edilen Yol(Grid)
A Star	0.095	12486	243
Lee	0.079	15321	302

İki Engel Senaryosu: İki sabit engel koyup aynı senaryoda Lee ve A star algoritmaları karşılaştırıldı. Sonuçlar tek engeldeki sonuçlara yakındı. A star iki sabit engelde de Lee algoritmasına göre üç açıdan da daha başarılıydı.

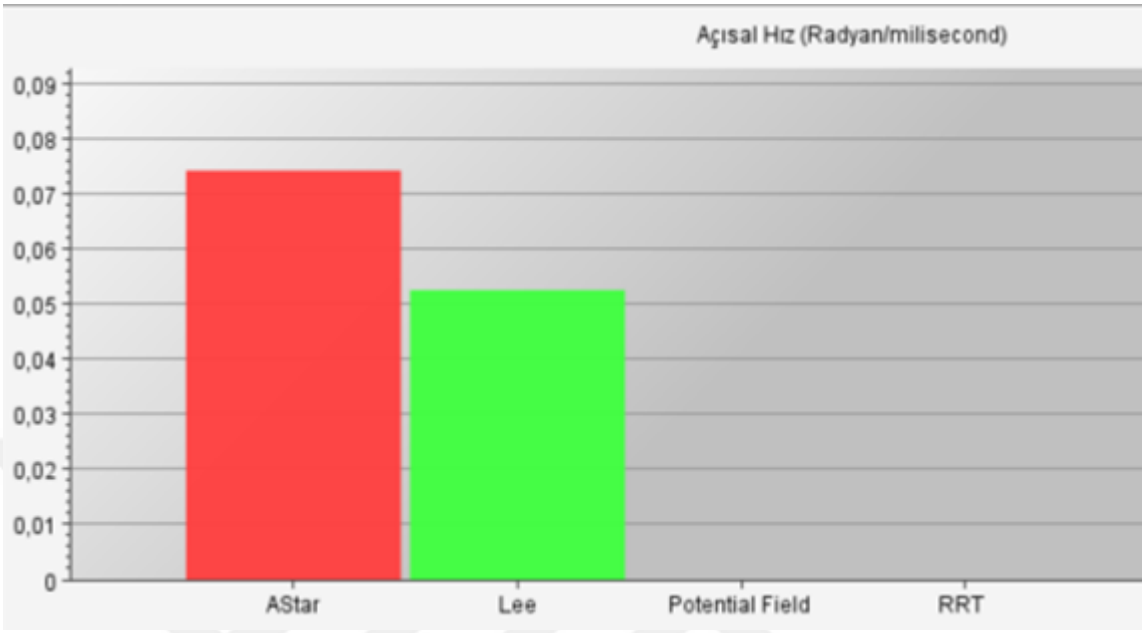


Şekil 4.6 A Star İki Sabit Engel Güzergah Başlama (32,121)-Bitiş Noktası(1296,235)



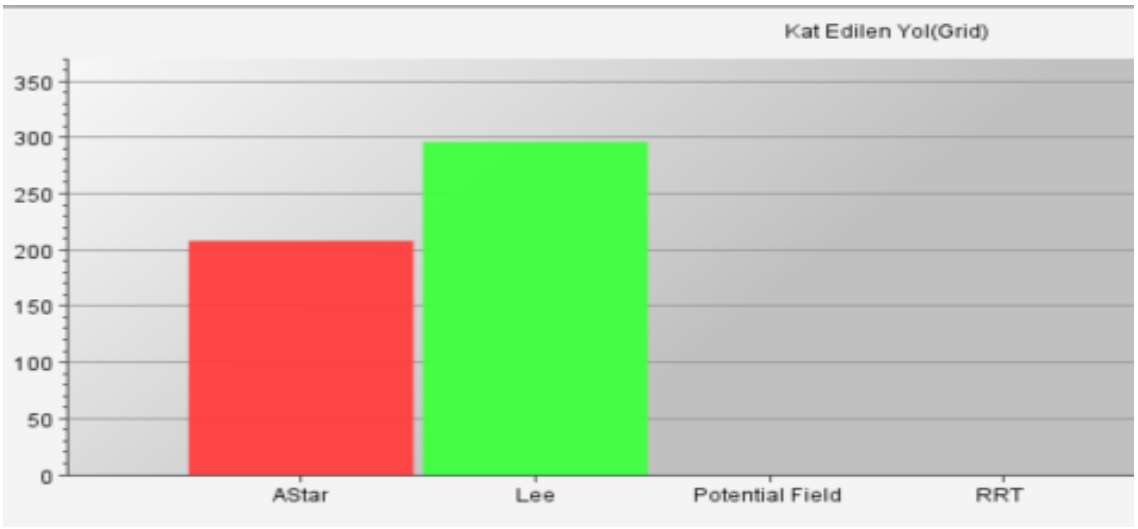
Şekil 4.7 Lee İki Sabit Engel Güzergahı Başlama (32,121)-Bitiş Noktası(1296,235)

Açısal Hız: Bu senaryoda iki algoritmanın açısal hızı aşağıdaki grafikte gösterilmiştir.



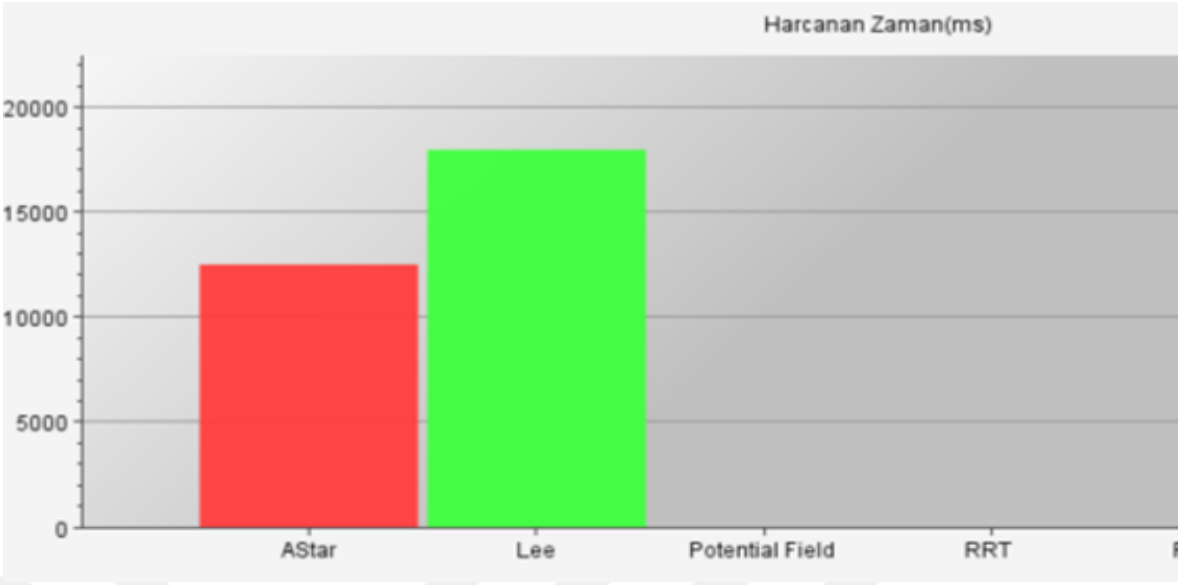
Şekil 4.8 İki Sabit Engel Senaryosunda A Star ve Lee Algoritması Karşılaştırması

Kat Edilen Yol: İki engel kat edilen yol grafikleri şekildeki gibidir.



Şekil 4.9 İki Sabit Engel A Star ve Lee Kat Edilen Yol Karşılaştırması

Harcanan Zaman: İki engel harcanan zaman karşılaştırılması aşağıdaki şekildedir.

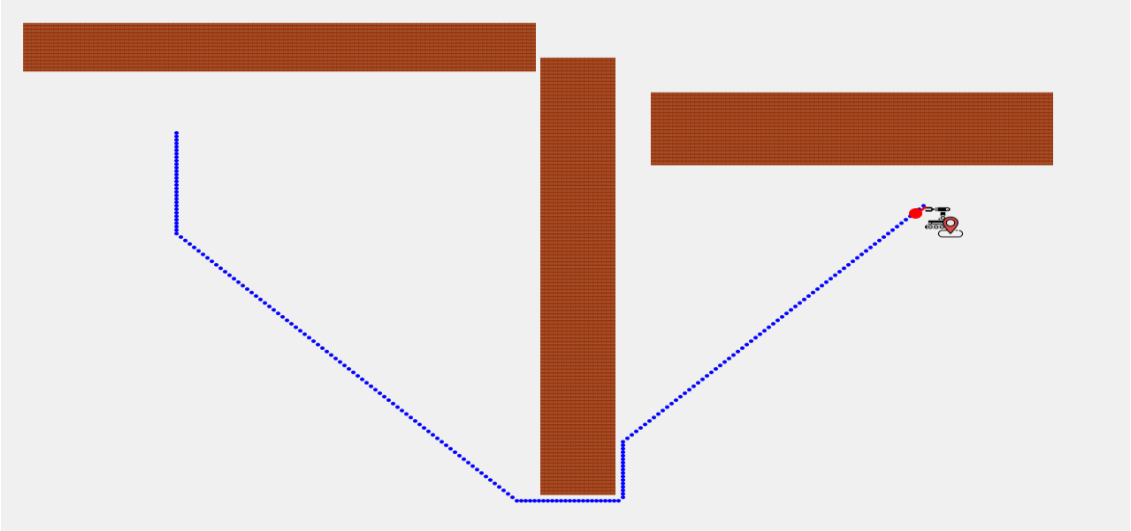


Şekil 4.10 İki Sabit Engel A Star ve Lee Harcanan Zaman Karşılaştırması

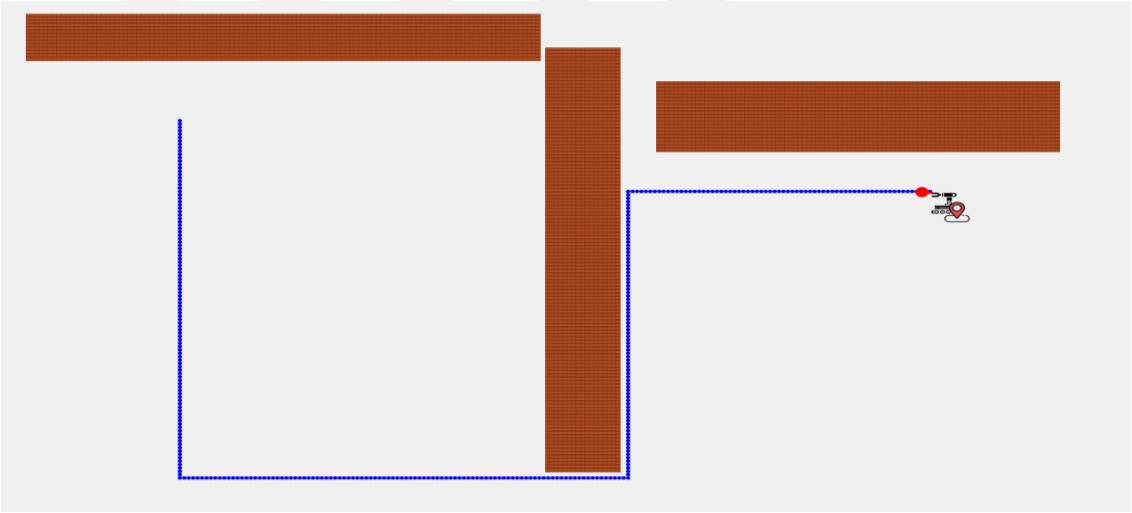
Çizelge 4.2 İki Sabit Engel A STar ve Lee Sonuçları

	Açısal Hız(Radyan/milisecond)	Harcanan Zaman(ms)	Kat Edilen Yol(Grid)
A Star Algoritması	0.073	13101	211
Lee Algoritması	0.052	17486	289

Üç Engel Senaryosu: Aynı senaryoda üç sabit engelle karşılaştırılmış. Sonuçlarda A star algoritması yine daha başarılı olmuştur.

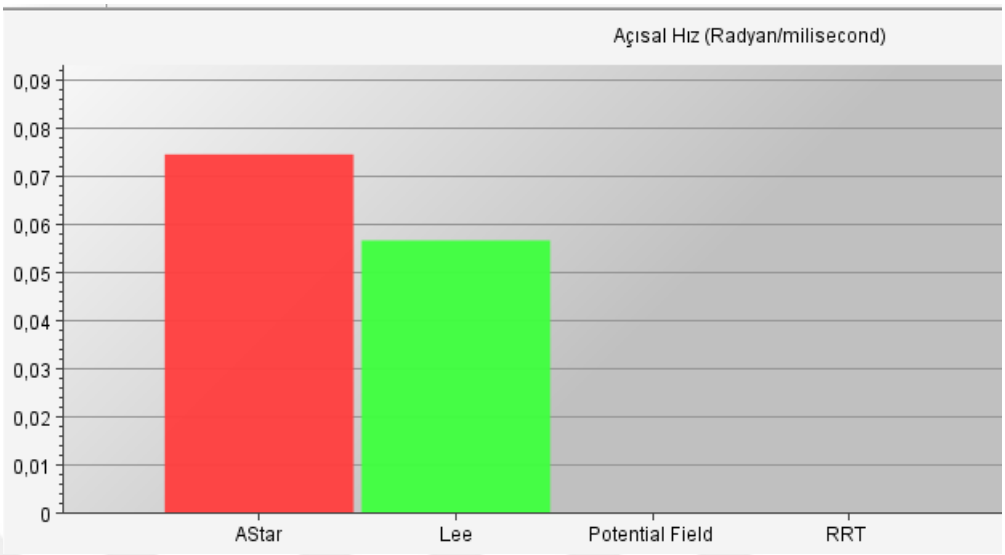


Şekil 4.11 A Star Üç Sabit Engel Güzergah Başlama(244,146)-Bitiş Noktası(1485,275)



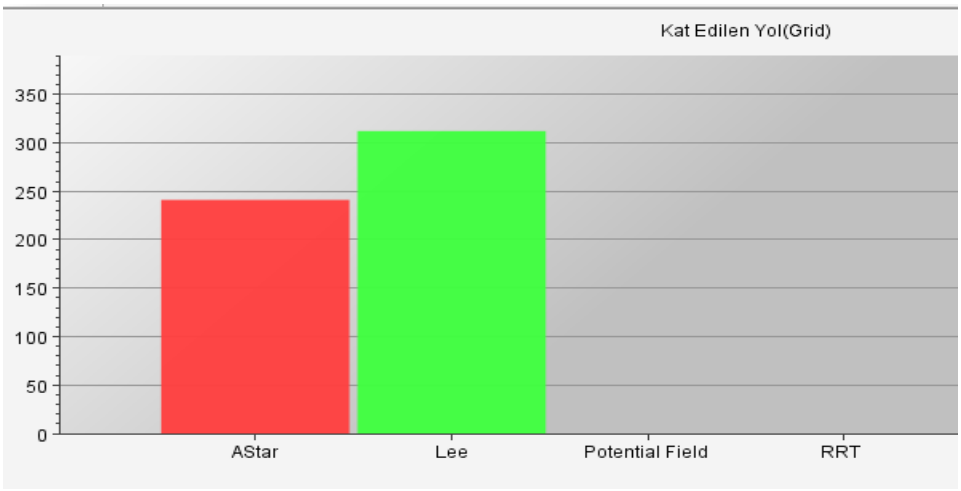
Şekil 4.12 Lee Üç Sabit Engelli Güzergah Başlama (244,146)-Bitiş Noktası(1485,275)

Açısal Hız: Bu senaryoda iki algoritmanın açısal hızı aşağıdaki grafikte gösterilmiştir.



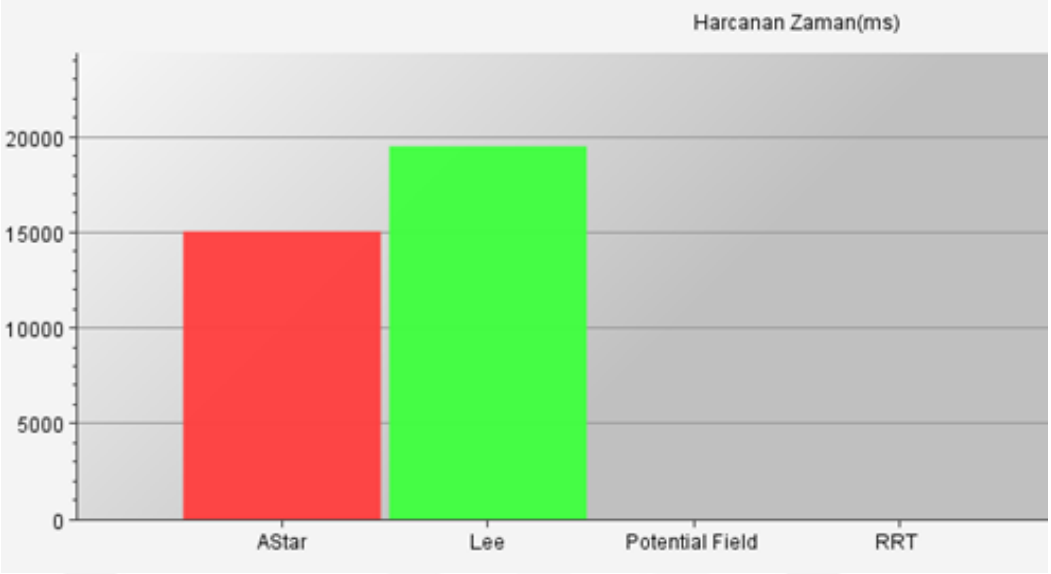
Şekil 4.13 Üç Sabit Engelli Senaryoda A Star ve Lee Açısal Hız Karşılaştırması

Kat Edilen Yol: Bu senaryoda iki algoritmanın kat edilen yolları aşağıdaki grafikte gösterilmiştir.



Şekil 4.14 Üç Sabit Engelli Senaryoda AStar ve Lee Kat Edilen Yol Karşılaştırması

Harcanan Zaman: Bu senaryoda iki algoritmanın harcanan zamanları aşağıdaki grafikte gösterilmiştir.



Şekil 4.15 Üç Sabit Engelli Senaryoda A Star ve Lee Harcanan Zaman Karşılaştırması

Çizelge 4.3 Üç Sabit Engel A Star ve Lee Sonuçları

	Açısal Hız(Radyan/milisecond)	Harcanan Zaman(ms)	Kat Edilen Yol(Grid)
A Star Algoritması	0.075	15007	244
Lee Algoritması	0.058	19876	309

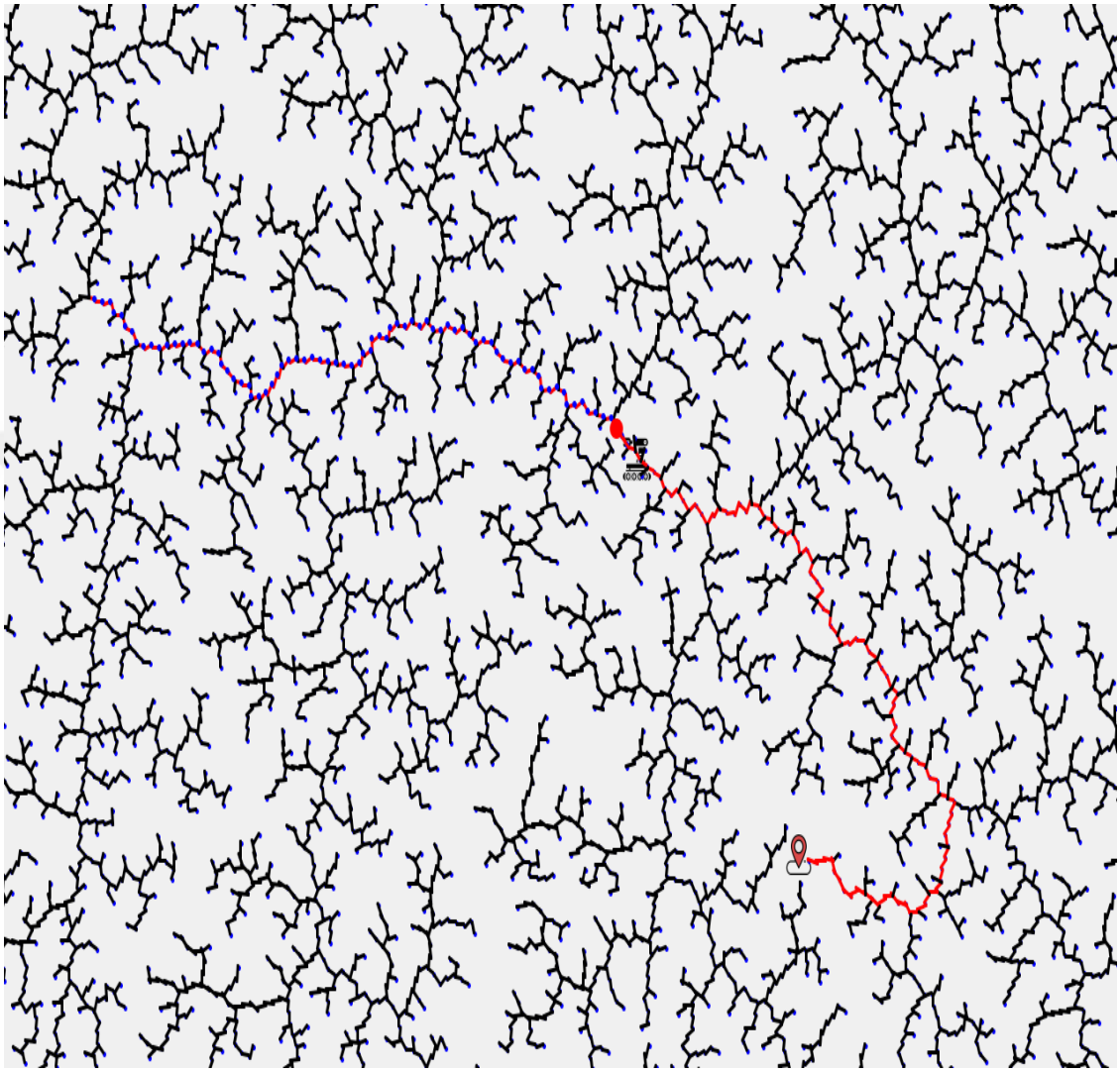
Sonuç olarak grafiklerde ve yol haritalarında görüldüğü üzere A star algoritmasının sezgisel yaklaşımı Lee Algoritmasına göre yön bulma , açısal hız, harcanan zaman bakımından her senaryoda çok daha başarılı sonuçlar vermiştir. Deneyler aynı başlangıç ve bitiş noktası için birer kez yapılmıştır.

4.1.2 RRT ve RRT star algoritma kıyaslanması

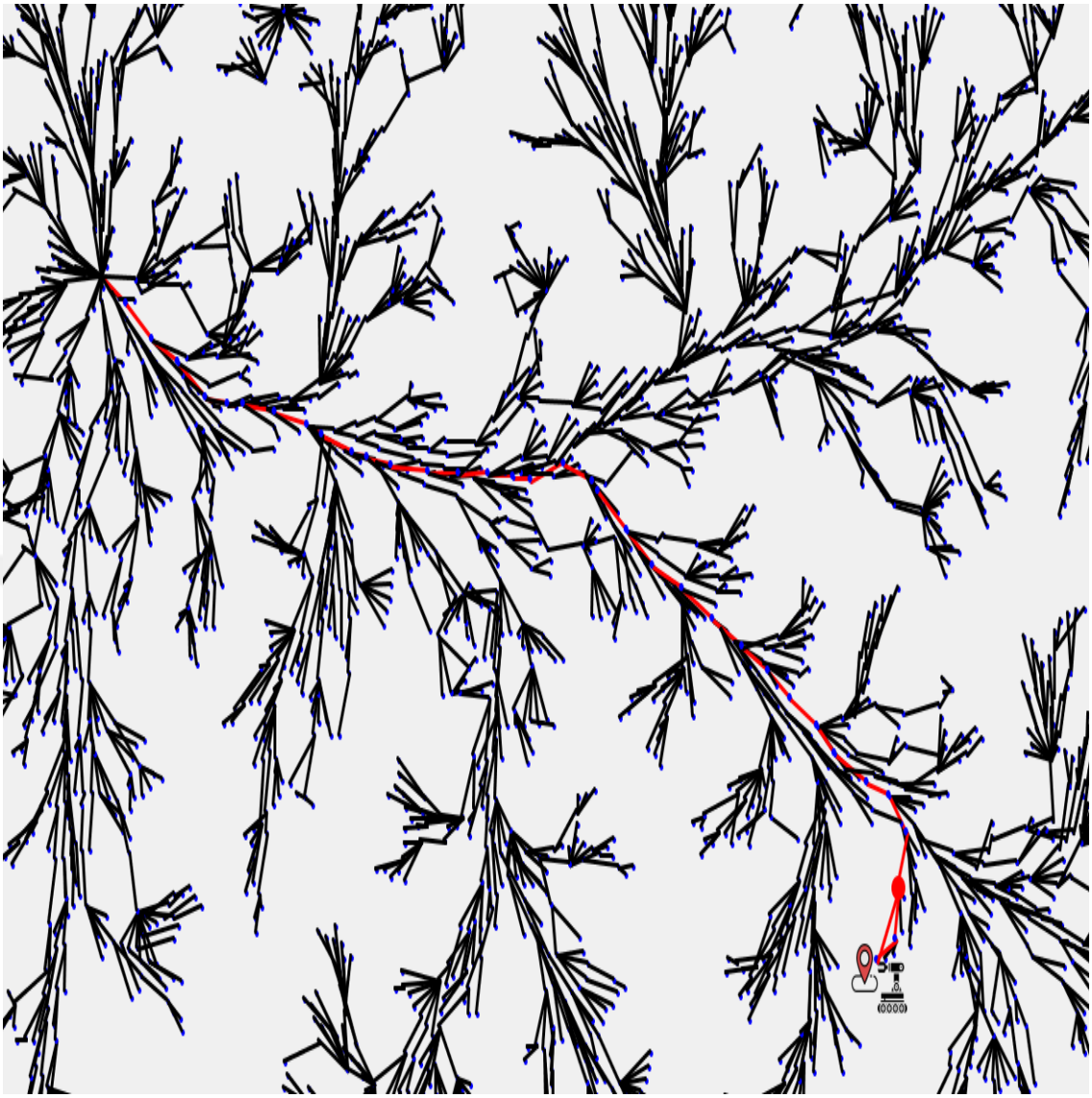
RRT Star, RRT'nin optimize edilmiş bir sürümüdür. Düğüm sayısı sonsuza yaklaştığında, RRT Star algoritması hedefe mümkün olan en kısa yolu verecektir. Gerçekçi olarak mümkün olmasa da, bu ifade algoritmanın en kısa yolu geliştirmek için çalıştığını göstermektedir. RRT Starın temel prensibi RRT ile aynıdır, ancak algoritmaya iki temel ekleme önemli ölçüde farklı sonuçlara yol açar. İlk olarak, RRT Star her tepe noktasının ana tepe noktasına göre kat ettiği mesafeyi kaydeder. Buna tepe noktasının maliyeti denir. Grafikte en yakın düğüm bulunduktan sonra, yeni düğümden sabit bir yarıçaptaki köşeleri incelenir. Proksimal düğümden daha ucuz bir düğüm bulunursa, ucuz düğüm proksimal düğümün yerini alır. Bu özelliğin etkisi, ağaç yapısına yelpaze şeklinde dalların eklenmesi ile görülebilir. RRT'nin kübik yapısı elimine edilir. RRT Starın eklediği ikinci fark, ağacın yeniden kablolanmasıdır. Bir tepe noktası en ucuz komşuya bağlandıktan sonra, komşular tekrar incelenir. Yeni eklenen tepe noktasına yeniden bağlanmanın maliyetlerini düşürüp düşürmeyeceği komşular tarafından kontrol edilir. Maliyet gerçekten azalırsa, komşu yeni eklenen tepe noktasına yeniden bağlanır. Bu özellik yolu daha düzgün hale getirir. RRT * inanılmaz derecede düz yollar oluşturur. Ek olarak, grafikleri RRT'ninkinden karakteristik olarak farklıdır. Özellikle yoğun bir engel alanında optimum bir yol bulmak için RRT Starın yapısı inanılmaz derecede faydalıdır. Grafik nesnelerin etrafına asılır ve RRT ile karşılaştırıldığında daha kısa yollar bulur. Hedef değişecek olsaydı, orijinal grafik hala bölgedeki çoğu konuma giden en hızlı yolu temsil ettiği için kullanılabilir.

Bu çalışmada bu iki algoritma üç farklı senaryoda birbiriyle üç farklı açıdan kıyaslanıp görselleştirildi ve sonuçları çıkarıldı. Her bir senaryo her algoritma için birer kez çalıştırıldı.

Engelsiz Senaryo: Hiçbir engel olmadan bu iki algoritmanın aynı başlangıç ve bitiş noktasından kıyaslanması yapılmıştır. Sonuçları da kaydedilmiştir.

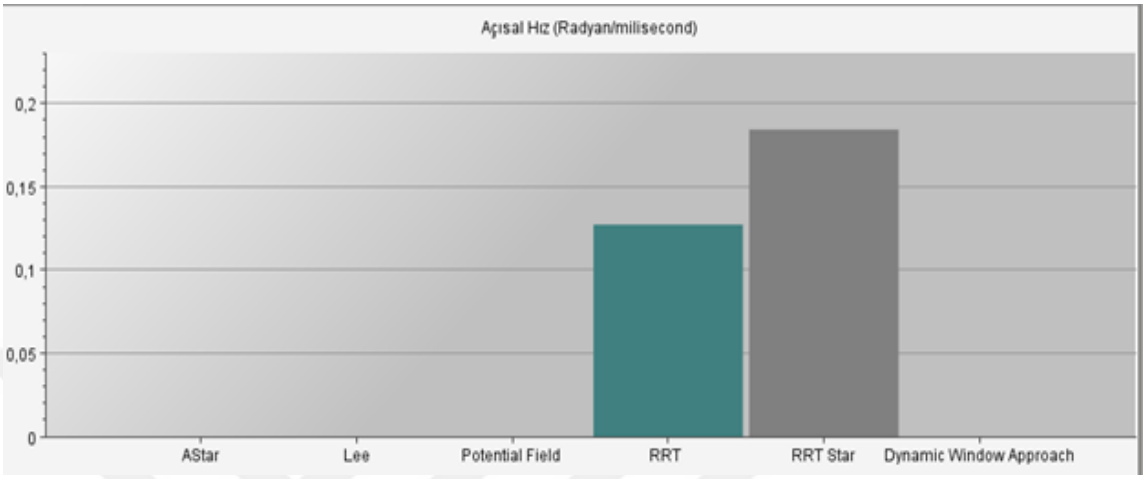


Şekil 4.16 RRT Engelsiz Senaryo Başlama (135,220) –Bitiş Noktası(1245,690)



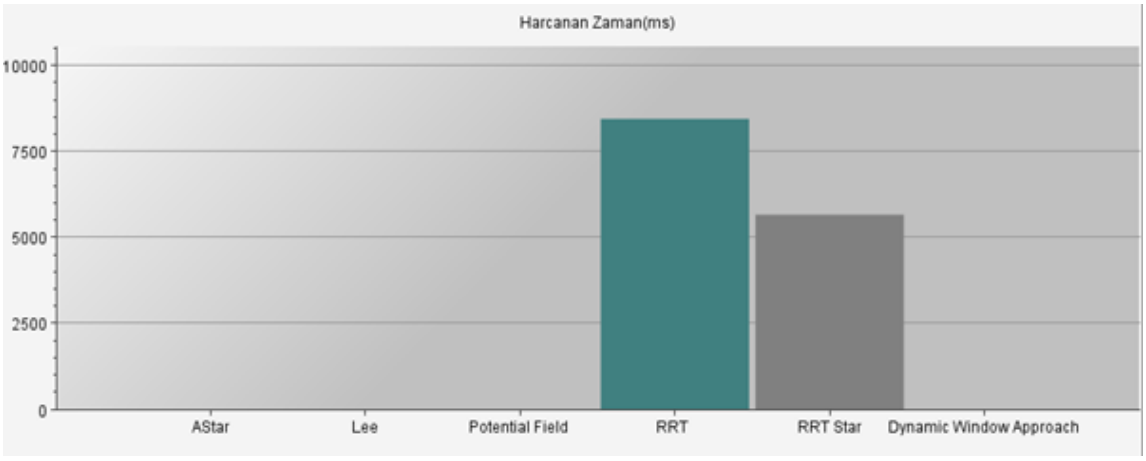
Şekil 4.17 RRT Star Engelsiz Senaryo Başlama (135,220)–Bitiş Noktası(1245,690)

Açısal Hız: Engelsiz ortamda iki algoritmanın açısal hız bakımından karşılaştırılması aşağıdaki şekildeki gibidir.



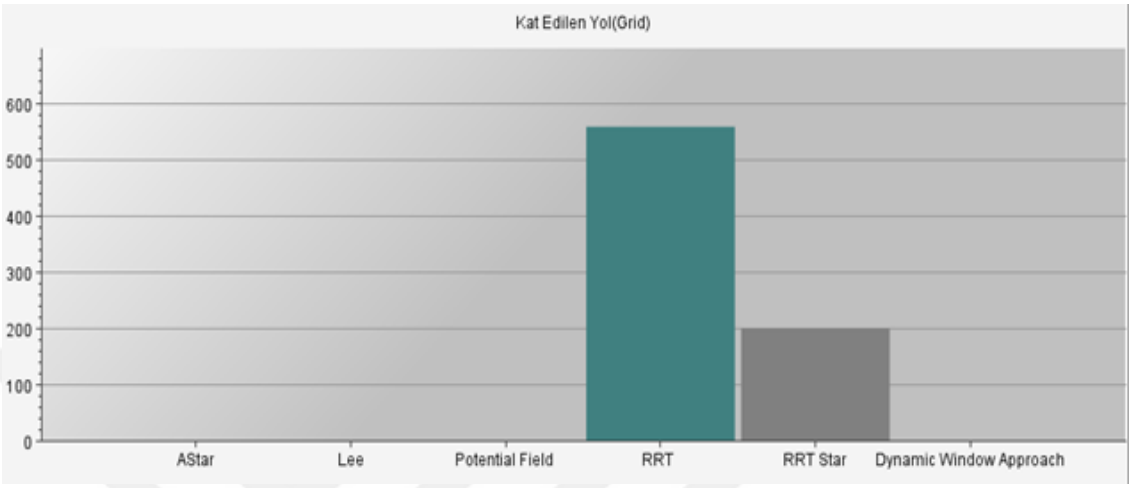
Şekil 4.18 Engelsiz Senaryoda RRT ve RRT Star Açısal Hız Karşılaştırması

Harcanan Zaman: Engelsiz ortamda iki algoritmanın harcanan zaman bakımından karşılaştırılması aşağıdaki şekildeki gibidir.



Şekil 4.19 Engelsiz Senaryoda RRT ve RRT Star Harcana Zaman Karşılaştırması

Kat Edilen Yol: Engelsiz ortamda iki algoritmanın kat edilen yol bakımından karşılaştırılması aşağıdaki şekildeki gibidir.

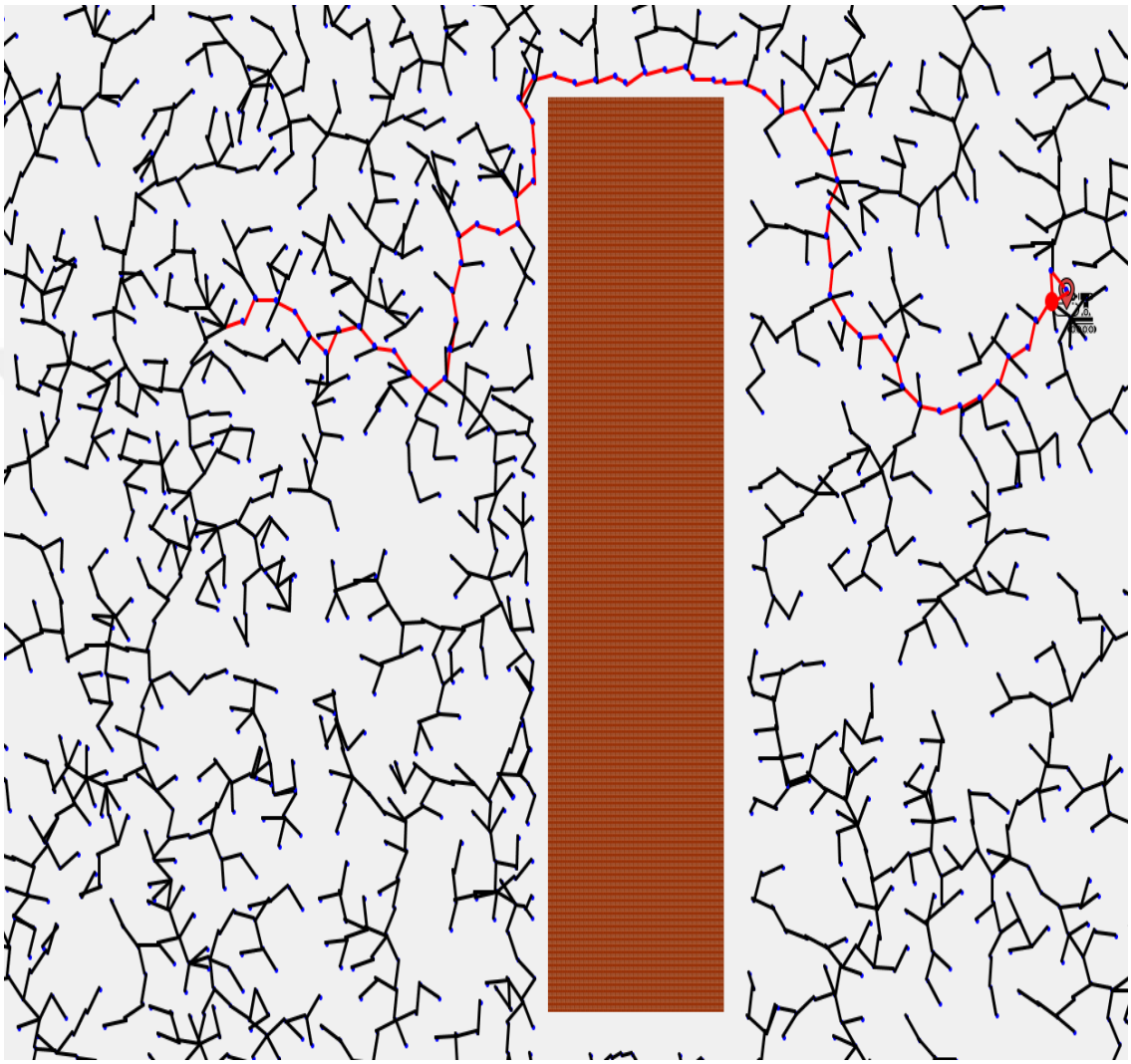


Şekil 4.20 Engelsiz Senaryo RRT ve RRT Star Kat Edilen Yol Karşılaştırması

Çizelge 4.4 Engelsiz Senaryo RRT ve RRT Star Sonuçları

	Açısal Hız(Radyan/milisecond)	Harcanan Zaman(ms)	Kat Edilen Yol(Grid)
RRTAlgoritması	0.12	8215	552
RRT Star Algoritması	0.17	5463	199

Tek Engel Senaryosu: Senaryoya tek engel eklenip RRT ve RRT Star algoritmaları bu şartlarda karşılaştırıldı. Üç farklı açıdan sonuçları da kaydedildi.

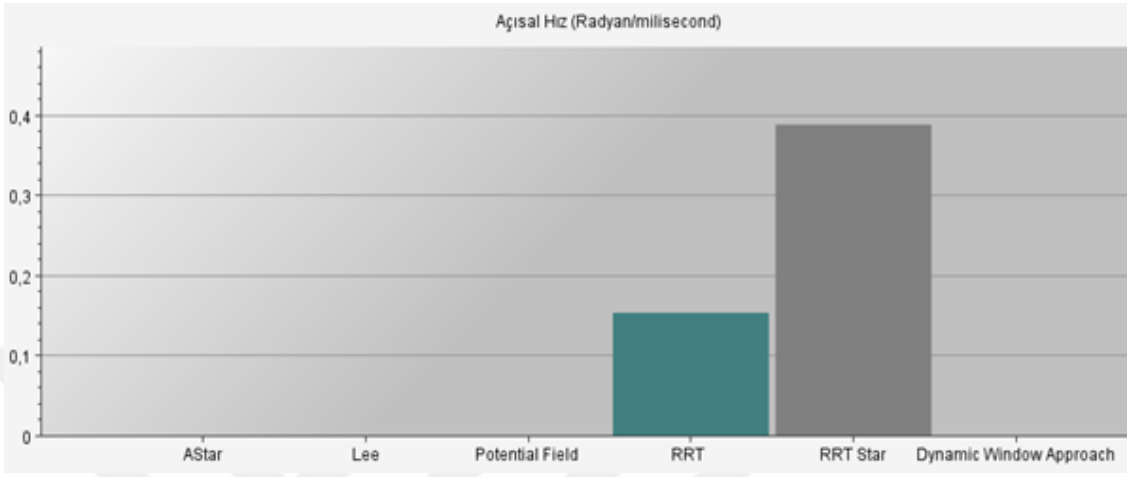


Şekil 4.21 RRT Tek Engel Başlama Noktası(321,214) –Bitiş Noktası(1346,207)



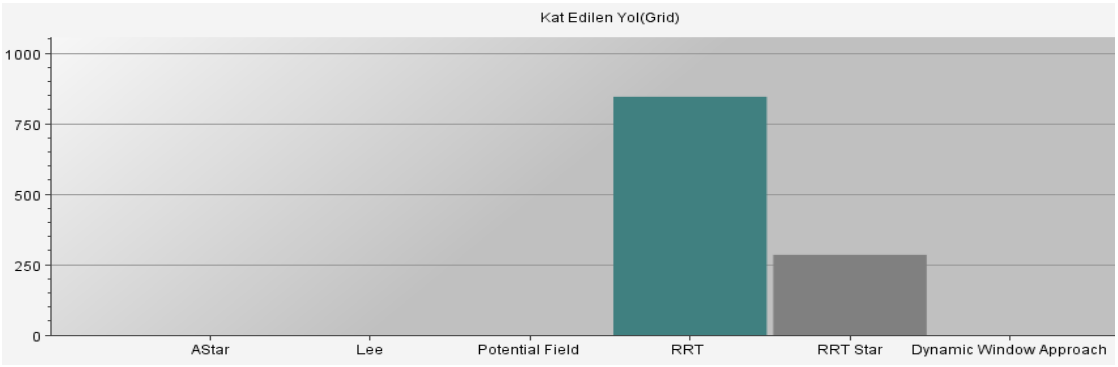
Şekil 4.22 RRT Star Tek Engel Güzergahı Başlama(321,214) –Bitiş Noktası(1346,207)

Açısal Hız: Tek engelli ortamda iki algoritmanın açısal hız bakımından karşılaştırılması aşağıdaki şekildeki gibidir.



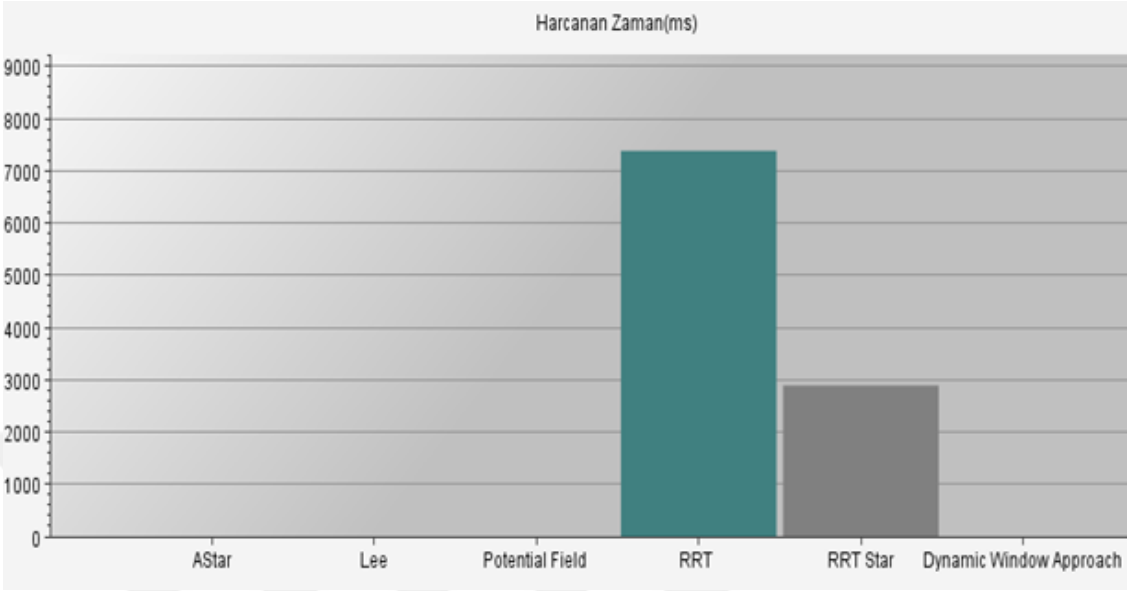
Şekil 4.23 Tek Engelli Senaryoda RRT ve RRT Star Açısal Hız Karşılaştırması

Kat Edilen Yol: Tek engelli ortamda iki algoritmanın kat edilen yol bakımından karşılaştırılması aşağıdaki şekildeki gibidir.



Şekil 4.24 Tek Engelli Senaryoda RRT ve RRT Star Kat Edilen Yol Karşılaştırması

Harcanan Zaman: Tek engelli ortamda iki algoritmanın harcanan zaman bakımından karşılaştırılması aşağıdaki şekildeki gibidir.

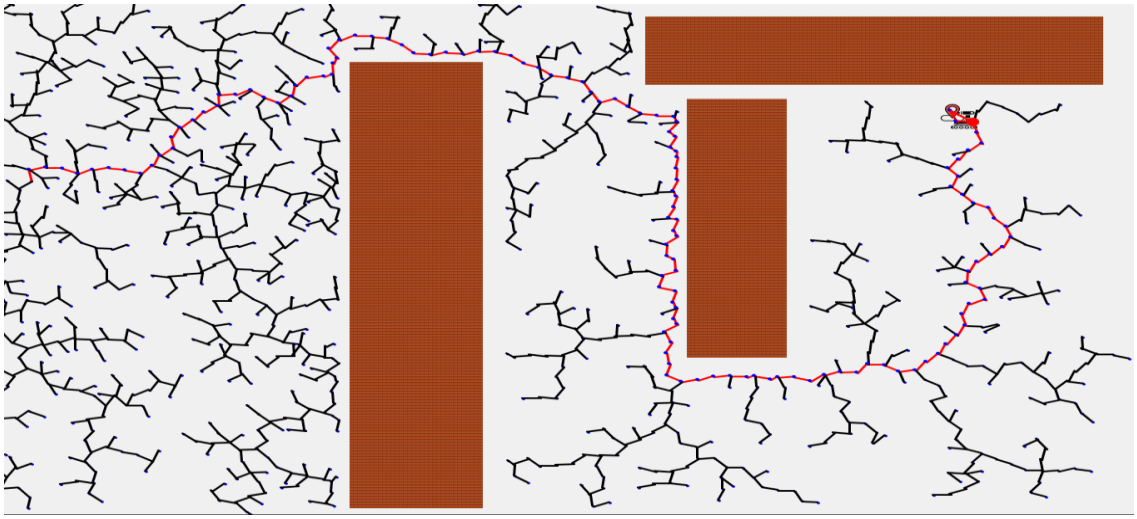


Şekil 4.25 Tek Engelli Senaryoda RRT ve RRT Star harcanan Zaman Karşılaştırması

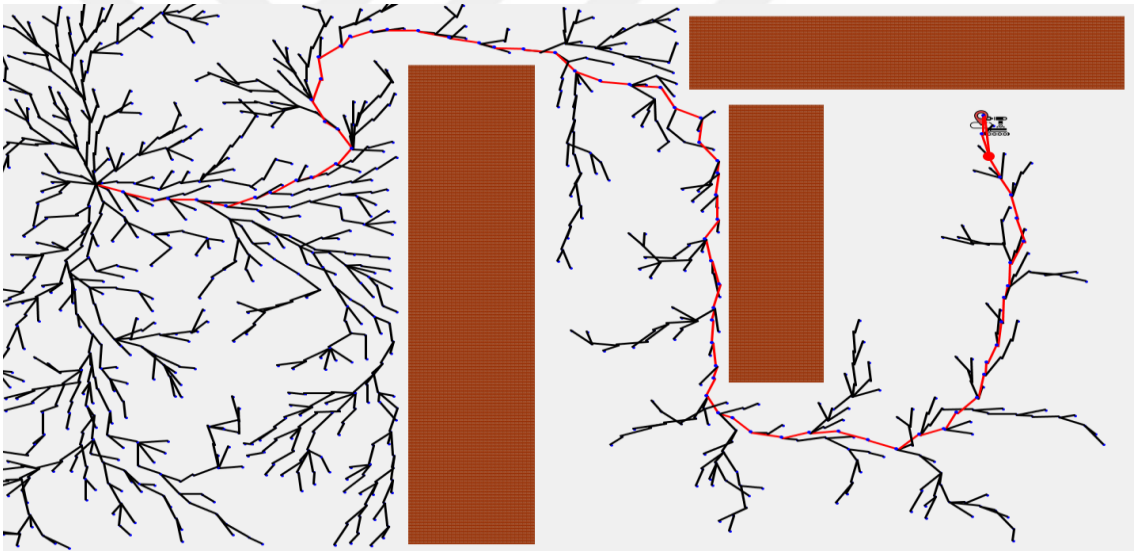
Çizelge 4.5 Tek Engel RRT ve RRT Star Sonuçları

	Açısal Hız(Radyan/milisecond)	Harcanan Zaman(ms)	Kat Edilen Yol(Grid)
RRTAlgoritması	0.12	7251	812
RRT Star Algoritması	0.38	2919	262

Üç Engel Senaryosu: Son olarak Üç engelli ortamda iki algoritma çalıştırılıp sonuçları kaydedilmiştir.

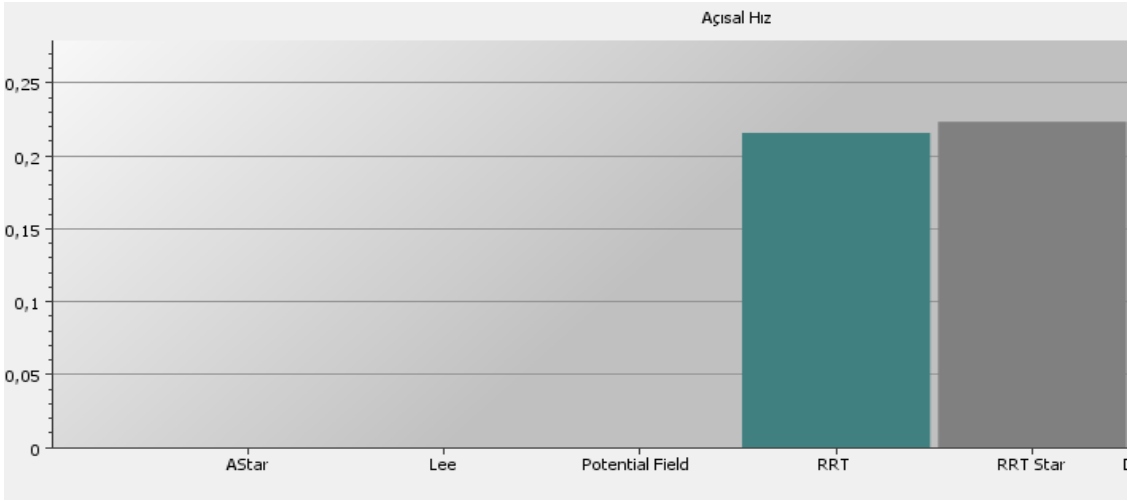


Şekil 4.26 RRT Üç Engelli Güzergah Başlama (69,224) –Bitiş Noktası(1369,207)



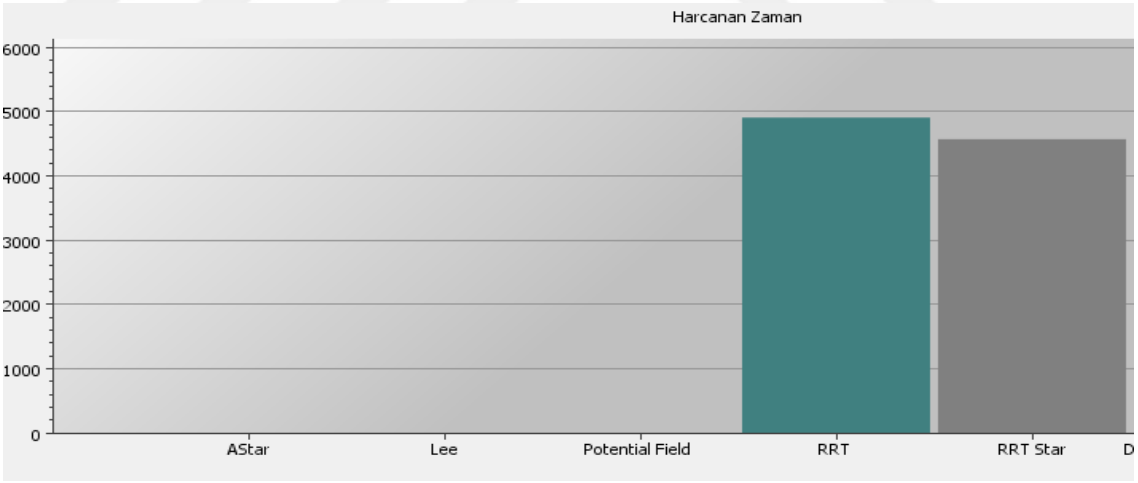
Şekil 4.27 RRT Star Üç Engelli Güzergah Başlama (69,224) –Bitiş Noktası(1369,207)

Açısal Hız: Üç engelli senaryoda iki algoritma açısal hız açısından karşılaştırıldı.



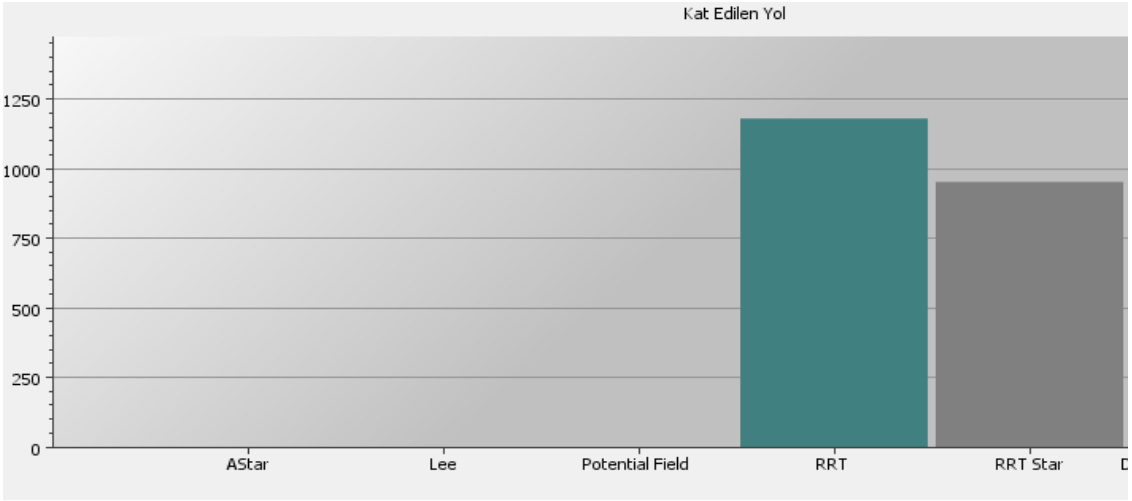
Şekil 4.28 Üç Engelli Senaryoda RRT ve RRT Star Açısal Hız Karşılaştırması

Harcanan Zaman: Üç engelli senaryoda iki algoritma harcanan zaman açısından karşılaştırıldı.



Şekil 4.29 Üç Engeli Senaryoda RRT ve RRT Star Harcanan Zaman Karşılaştırması

Kat Edilen Yol: Üç engelli senaryoda iki algoritma kat edilen yol açısından karşılaştırıldı.



Şekil 4.30 Üç Engelli Senaryoda RRT ve RRT Star Kat Edilen Yol Karşılaştırması

Çizelge 4.6 Üç Engel RRT ve RRT Star Sonuçları

	Açısal Hız(Radyan/milisecond)	Harcanan Zaman(ms)	Kat Edilen Yol(Grid)
RRTAlgoritması	0.211	4961	1173
RRT Star Algoritması	0.213	4872	964

Yukarıdaki şekillerde de gözlendiği gibi üç farklı senaryoda RRT ve RRT Star algoritması çalıştırıldı. Senaryodaki başlangıç bitiş noktaları ve engellerde aynı konumda tutulmuştur. Üç duruma da bakıldığında RRT Star üç alanda da RRT algoritmasına göre öndeydi. Sonuçlar da optimize edilmiş RRT Star ın üstünlüğü görülmüştür. Ama özellikle üç engelli senaryodaki durumda fark epey azalmış ve sonuçlar birbirine yakınsamıştır. Ama diğer iki senaryoda RRT Star'ın RRT ye önemli ölçüde farkı görülmektedir. Tabii bu sonuçlar senaryodan senaryoya değişmekle birlikte, RRT Star'ın RRT algoritmasının optimize edilmiş bir versiyonu olduğu unutulmamalıdır.

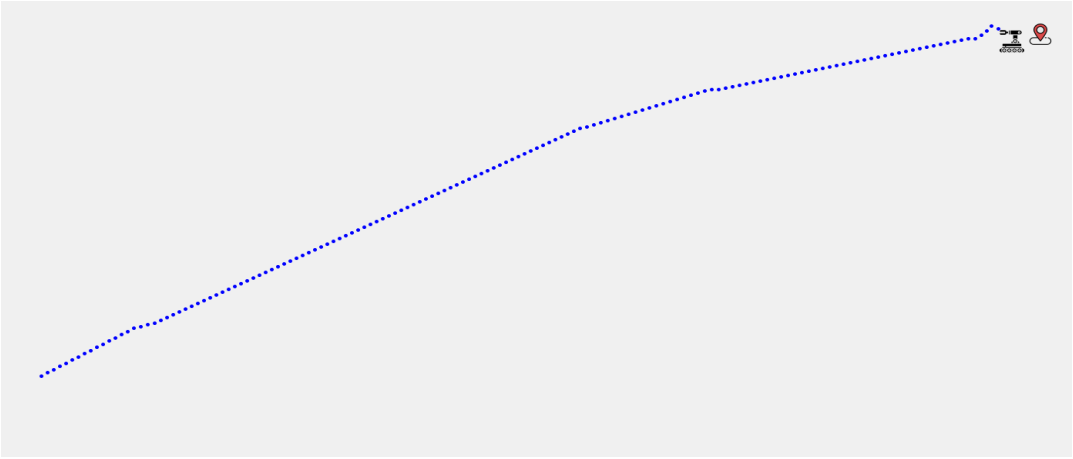
4.2 Gerçek Zamanda Yön Bulma Algoritmalarının Karşılaştırılması

Bu çalışmada gerçek zamanda yol bulabilen ve hareketli engellerden kaçabilen iki algoritma üzerine çalışılmıştır. Bunlar Potential Field algoritması ve Dynamic Windows Approach algoritmasıdır. Bu iki algoritmada çalışma zamanında yolu hesaplar ve robot buna göre hareket eder. Çalışma zamanında yol bulmasının ana nedenlerinden biri hareketli engellerdir. Eğer ortamda hareketli engellerden kaçınma problemi varsa ya da varacağı hedef noktası sürekli değişiyorsa buna çözüm olarak dinamik yol bulma eylemi gerçekleştirilir. Bu çalışmada bu noktada bu iki algoritmanın birbirlerine göre avantajlı ve dezavantajlı olduğu alanlar ortaya konmuştur.

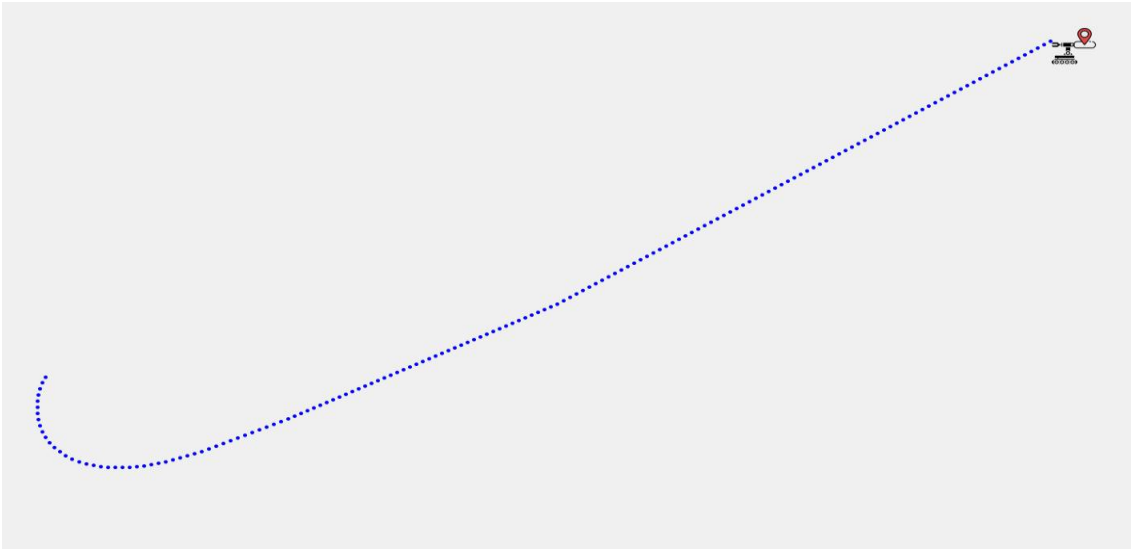
Bu iki algoritma hareketli engeller, hareketsiz engeller ve hareketli, hareketsiz engellerin olduğu farklı senaryolarda kıyaslanmıştır. Hareketli engel deneyleri aynı şartlar sağlananan kadar defalarca tekrarlanmış. İki algoritma da çalıştırıldığında hareketli engellerin konumu, büyüklüğü, doğrultusu, hızları ve güzergahı eşit olarak ayarlanmıştır.

4.2.1 Engelsiz Senaryo

PTA ve DWA ilk olarak engelsiz ortamda karşılaştırıldı. Eşit şartlarda karşılaştırabilmek için başlangıç ve bitiş noktaları aynı noktalar seçildi.

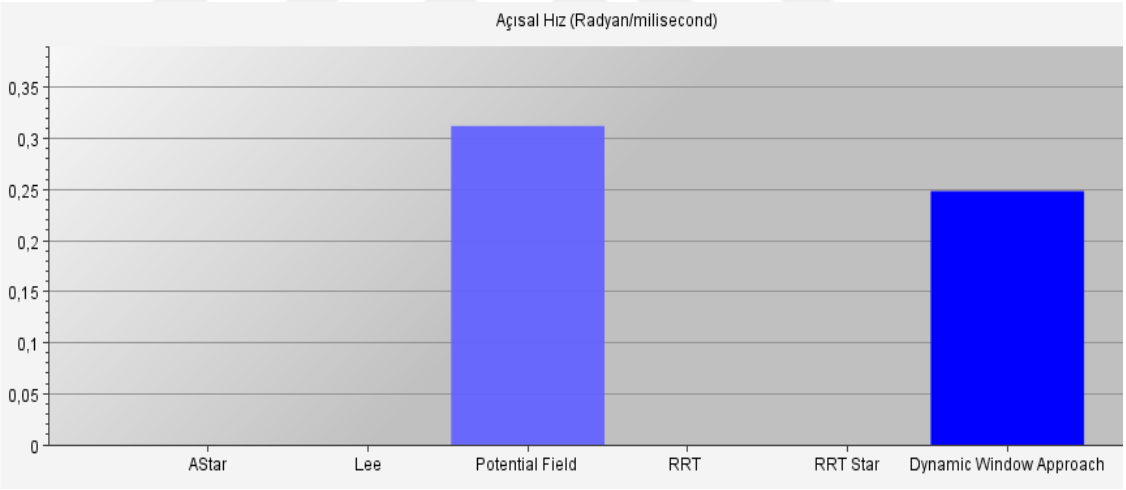


Şekil 4.31 Engelsiz Senaryoda PTA Güzergahı Başlama (26,816)-Bitiş Noktası(1696,11)



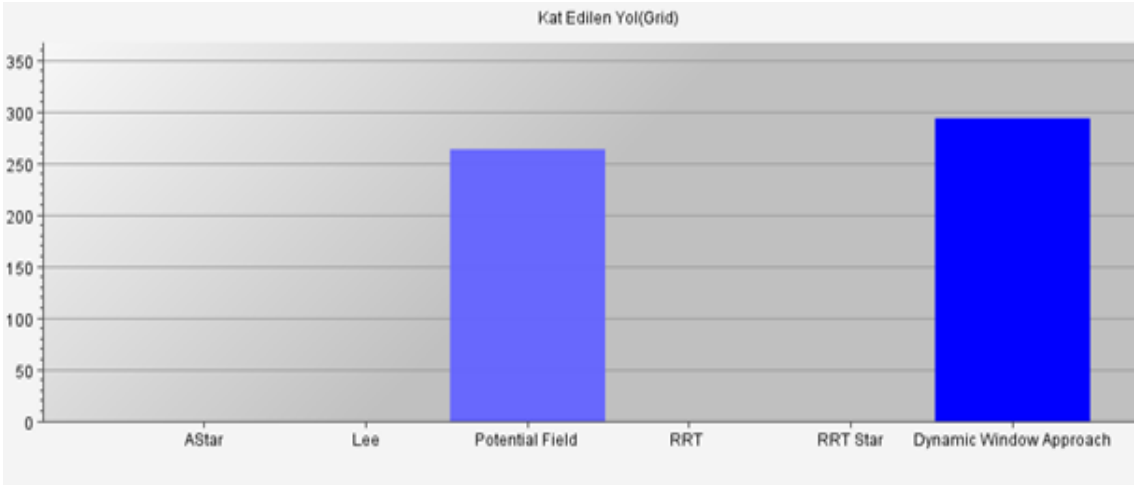
Şekil 4.32 Engelsiz Senaryo DWA Güzergahı Başlama(26,816)-Bitiş Noktası(1696,11)

Açısal Hız: Açısal Hızlar aşağıdaki grafikte karşılaştırılmıştır.



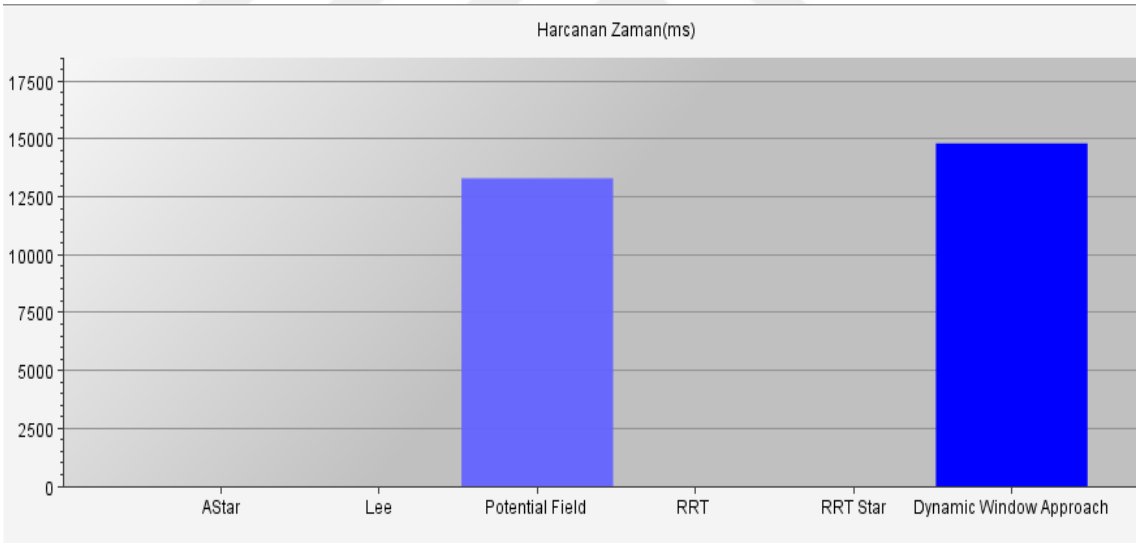
Şekil 4.33 Engelsiz Senaryoda PTA ve DWA Açısal Hız Karşılaştırması

Kat Edilen Yol: Kat Edilen yollar grafikte karşılaştırılmıştır.



Şekil 4.34 Engelsiz Senaryoda PTA ve DWA Kat Edilen Hız Karşılaştırması

Harcanan Zaman: Harcanan Zamanlar grafikte karşılaştırılmıştır.



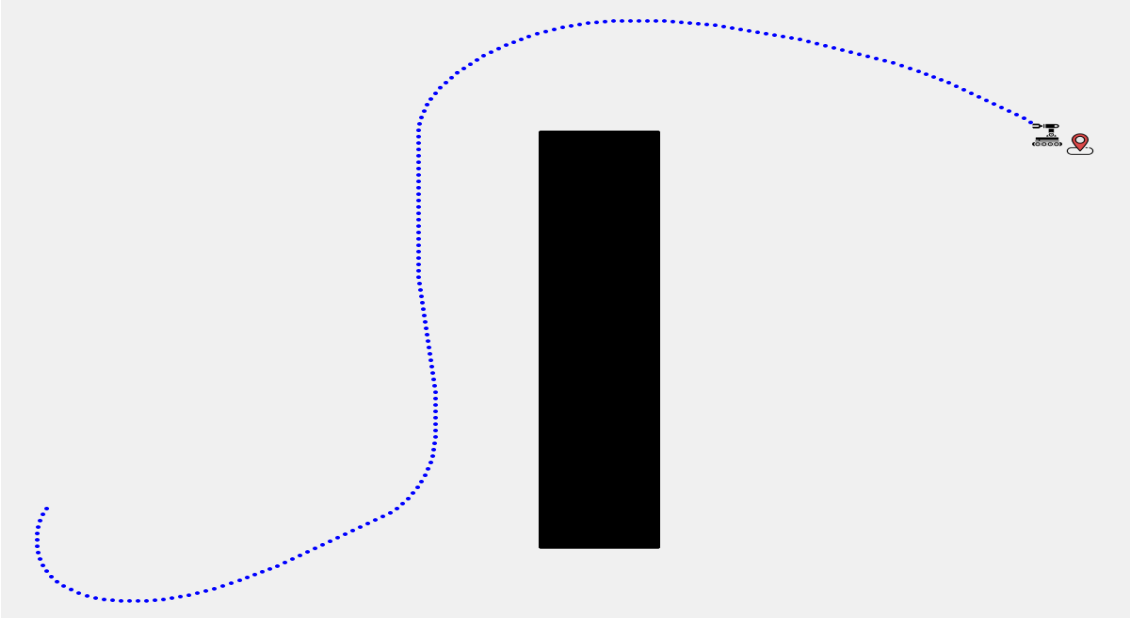
Şekil 4.35 Engelsiz Senaryoda PTA ve DWA Harcanan Zaman Karşılaştırması

Çizelge 4.7 Engelsiz Senaryo PTA ve DWA sonuçları

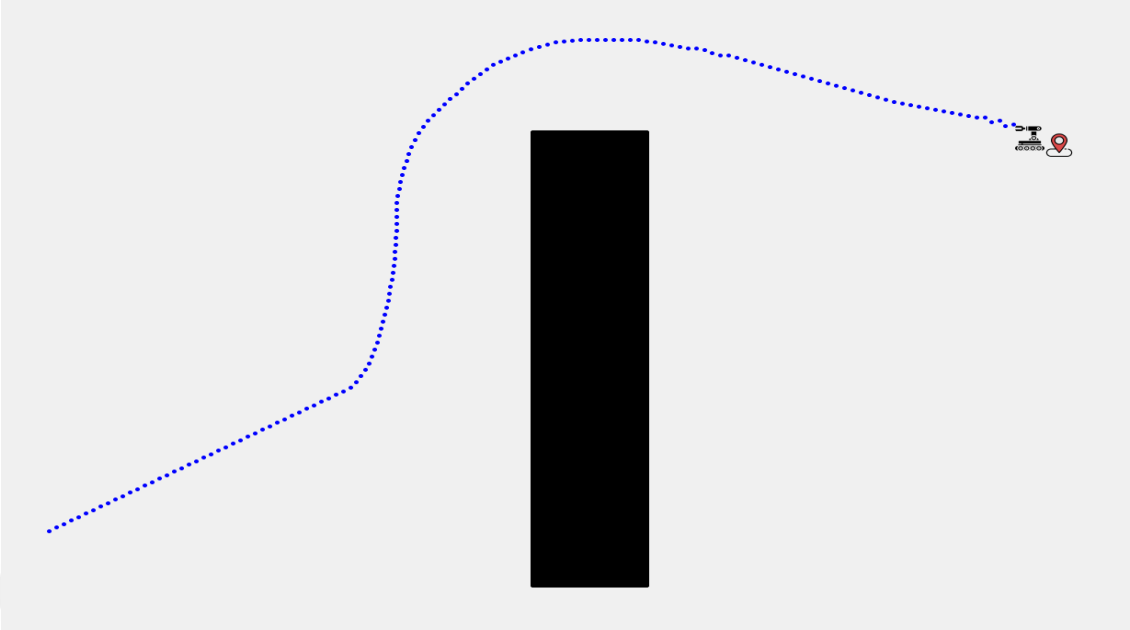
	Açısal Hız(Radyan/milisecond)	Harcanan Zaman(ms)	Kat Edilen Yol(Grid)
Potential Field Algoritması	0.31	13014	261
Dynamic Window Approach Algortiması	0.25	14889	296

4.2.2 Sabit tek engel senaryosu

Tek sabit engelde iki algoritma karşılaştırıldı.

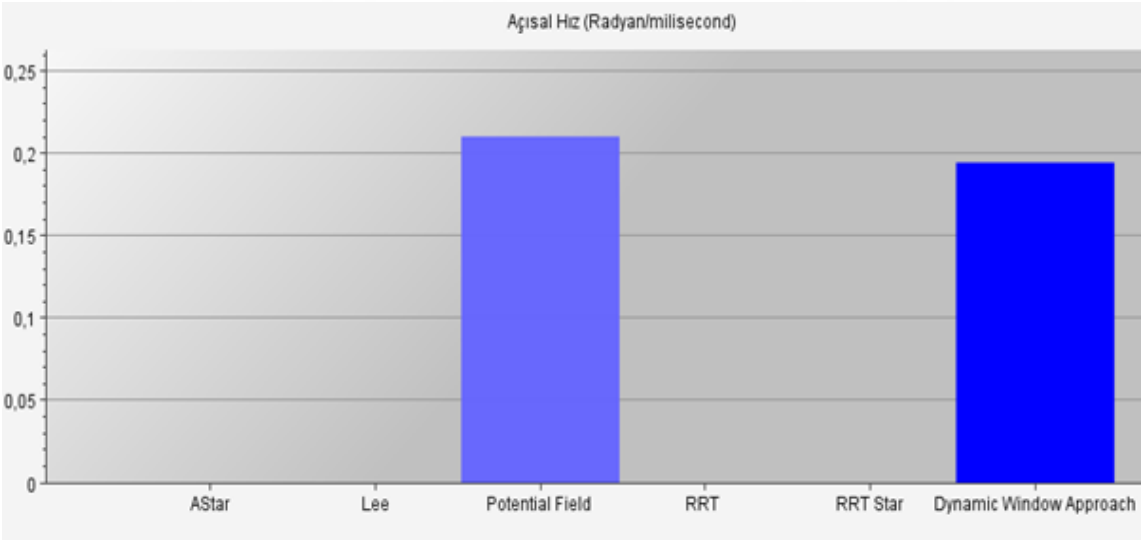


Şekil 4.36 DWA Tek Sabit Engel Güzergahı –Başlama (26,746)-Bitiş Noktası(1687,124)



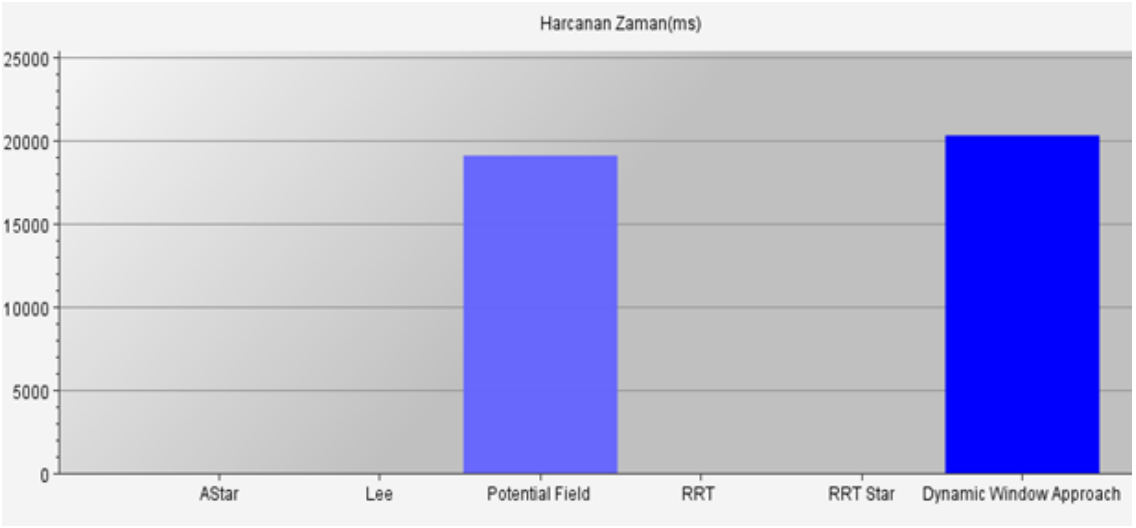
Şekil 4.37 PTA Tek Sabit Engel Güzergahı–Başlama (26,746)-Bitiş Noktası(1687,124)

Açısal Hız: Tek sabit engel senaryosunda açısal hız karşılaştırılması aşağıdaki gibidir.



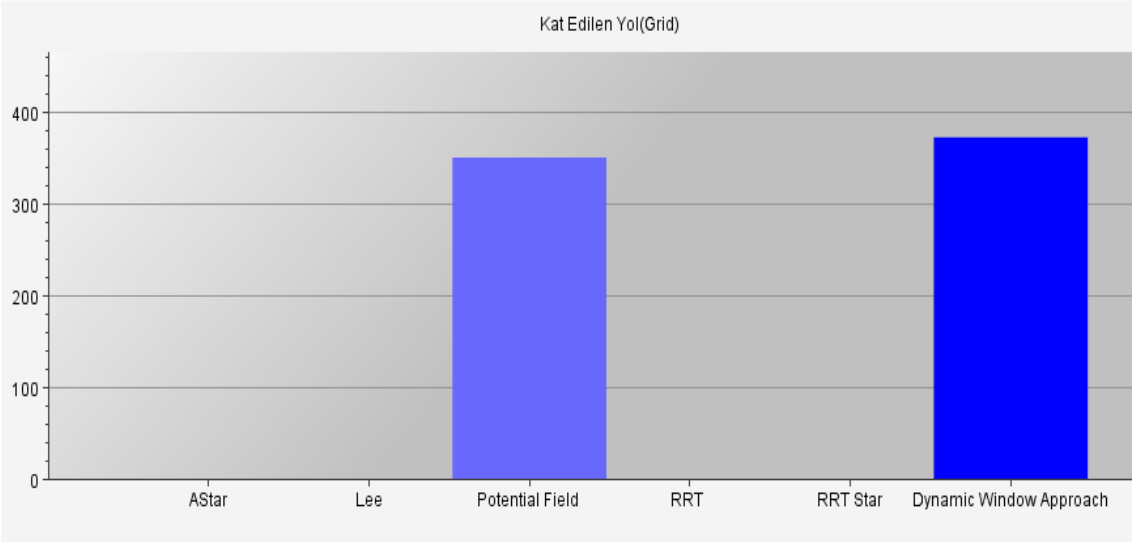
Şekil 4.38 Tek Sabit Engel Senaryosunda PTA ve DWA karşılaştırması

Harcanan Zaman: Tek sabit engel senaryosunda harcanan zaman karşılaştırılması aşağıdaki gibidir.



Şekil 4.39 Tek Sabit Engel Senaryosu PTA ve DWA Harcanan Zaman Karşılaştırması

Kat Edilen Yol: Tek sabit engel senaryosunda kat edilen yol karşılaştırılması aşağıdaki gibidir.



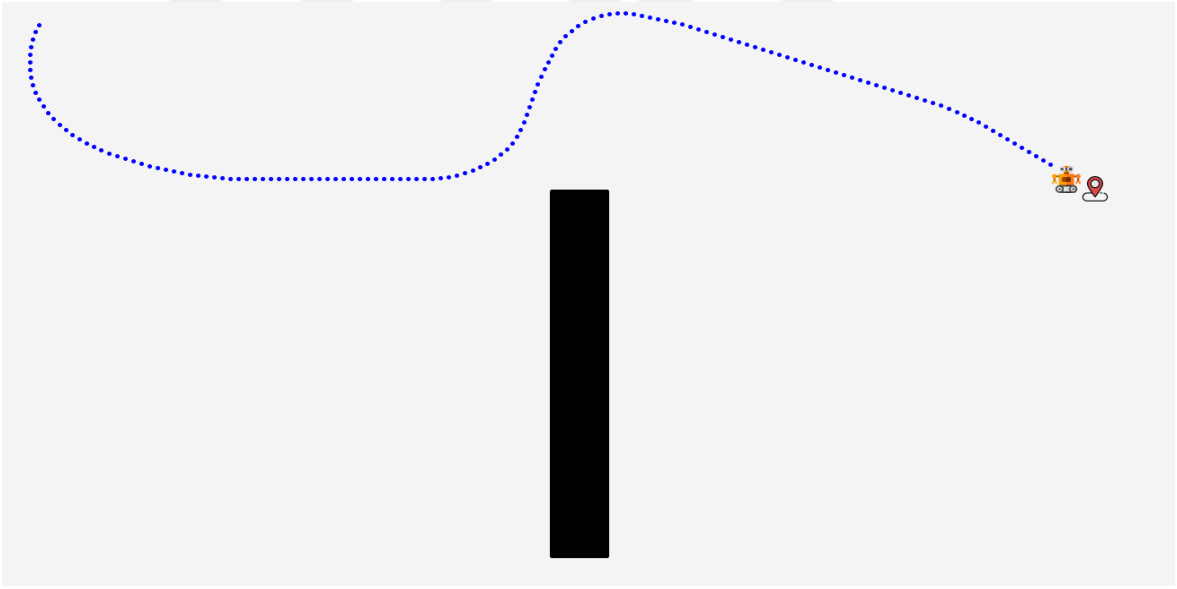
Şekil 4.40 Tek Sabit Engel Senaryosu PTA ve DWA Kat Edilen Yol Karşılaştırması

Çizelge 4.8 Tek Sabit Engel PTA ve DWA Sonuçları

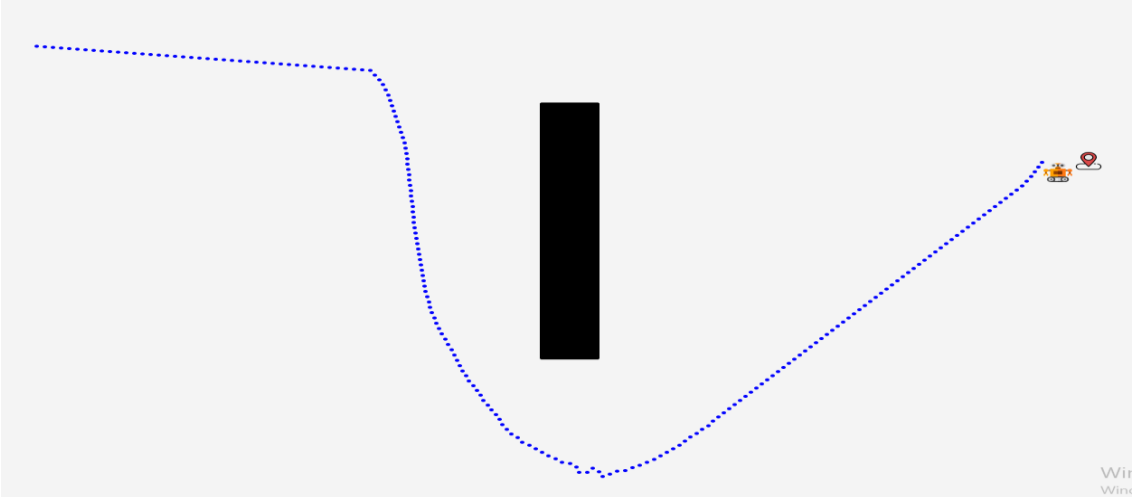
	Açısal Hız(Radyan/milisecond)	Harcanan Zaman(ms)	Kat Edilen Yol(Grid)
Potential Field	0.21	20024	362
Dynamic Window Approach	0.19	18982	383

4.2.3 Tek Hareketli Engel Senaryosu

Tek Hareketli engelde iki algoritma karşılaştırılmıştır. Sonuçları ve güzergahları aşağıdaki görselleştirilmiştir.

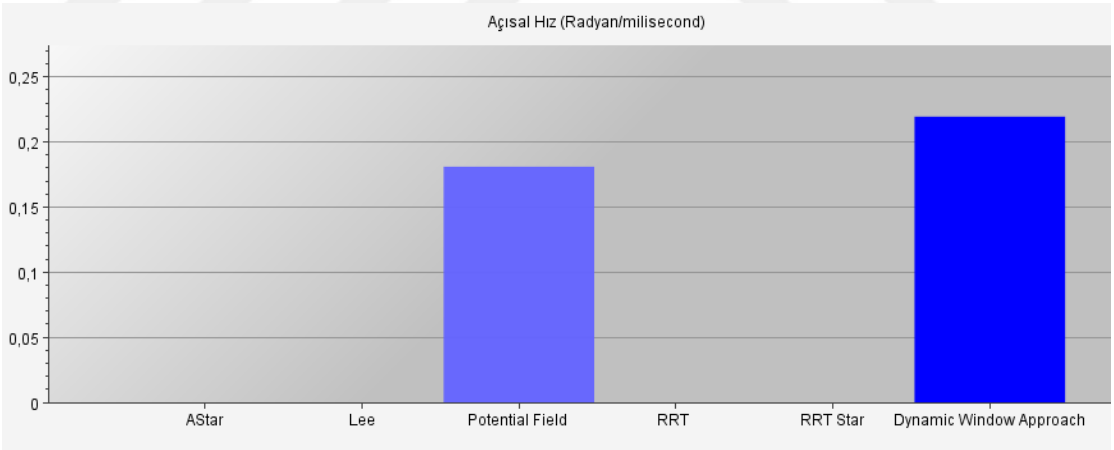


Şekil 4.41 DWA Tek Hareketli Engel Senaryosu Başlama(21,4)-Bitiş Noktası(1567,245)



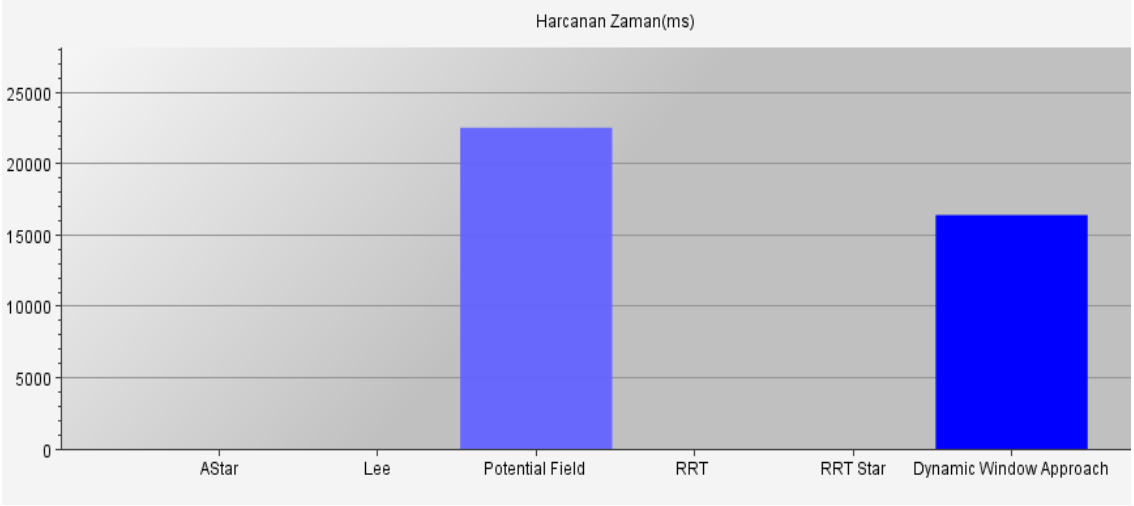
Şekil 4.42 PTA Tek Hareketli Engel Senaryosu Başlama (21,4)-Bitiş Noktası(1567,245)

Açısal Hız: Tek hareketli engel senaryosunda açısal hız karşılaştırması aşağıdaki şekildeki gibidir.



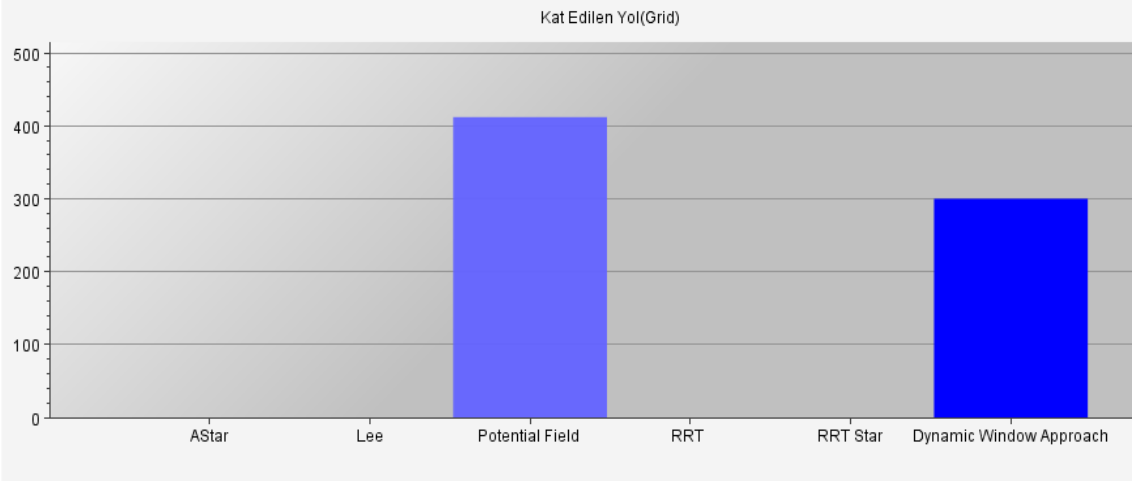
Şekil 4.43 Tek Hareketli Engel Senaryosu PTA ve DWA Açısal Hız Karşılaştırması

Harcanan Zaman: Tek hareketli engel senaryosunda harcanan zaman açısından yapılan karşılaştırmalar aşağıdaki şekildeki gibidir.



Şekil 4.44 Tek Hareketli Engel PTA ve DWA Harcanan Zaman Karşılaştırması

Kat Edilen Yol: Tek hareketli engel senaryosunda kat edilen yol açısından karşılaştırmalar aşağıdaki şekildeki gibidir.



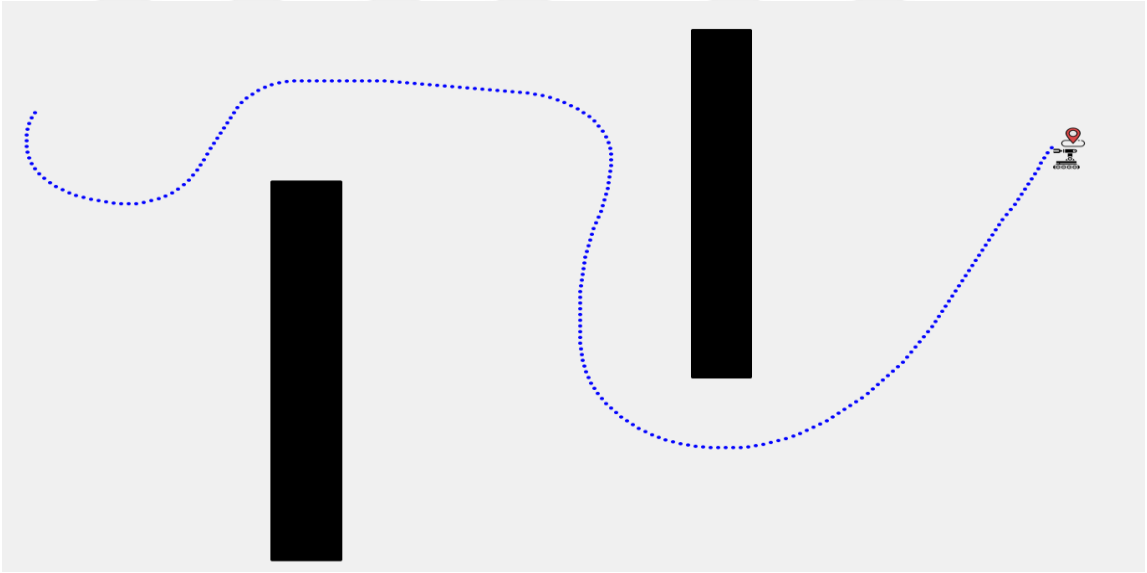
Şekil 4.45 Tek Hareketli Engel Senaryosu PTA ve DWA Kat Edilen Yol Karşılaştırması

Çizelge 4.9 Tek Haraketli Engel PTA ve DWA Sonuçları

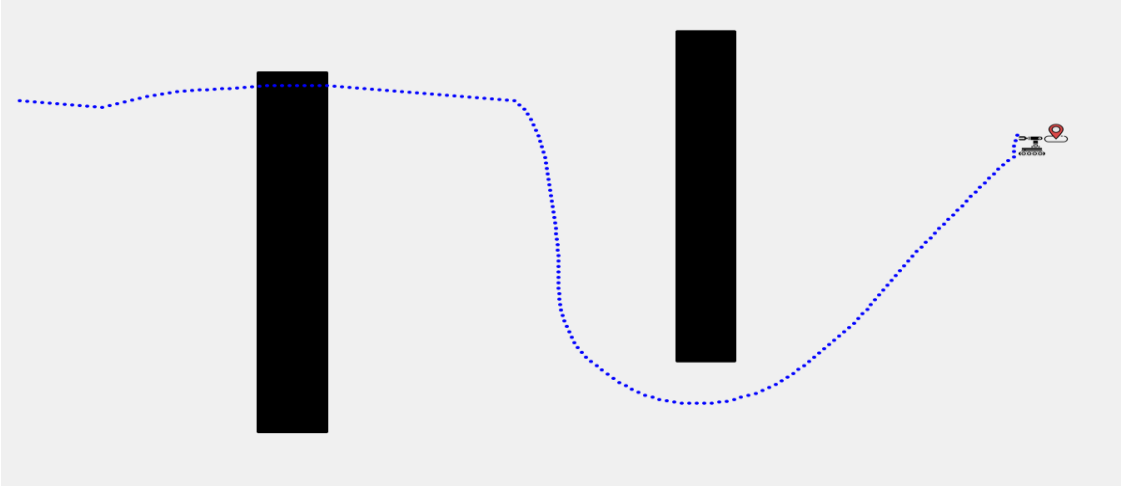
	Açısal Hız(Radyan/milisecond)	Harcanan Zaman(ms)	Kat Edilen Yol(Grid)
Potential Field	0.17	22631	414
Dynamic Window Approach	0.22	16054	300

4.2.4 Bir hareketli bir sabit engel senaryosu

Bir hareketli bir hareketsiz engel senaryosunda bu iki algoritma birbiriyle kıyaslanmış ve sonuçları aşağıdaki şekilde görselleştirilmiştir.

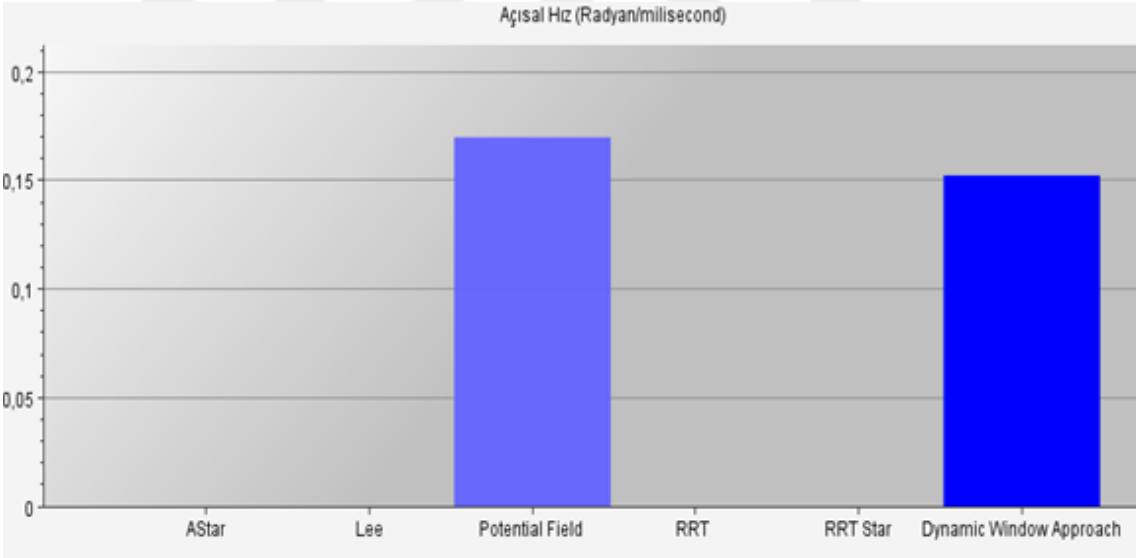


Şekil 4.46 DWA Bir Sabit bir Hareketli Engel–Başlama(14,197)-Bitiş (1595,204)



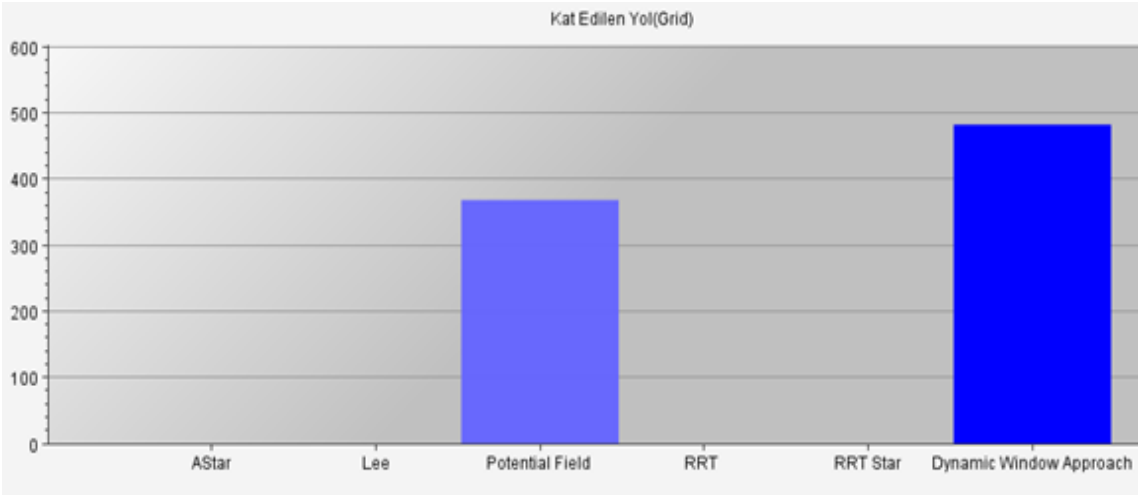
Şekil 4.47 PTA Bir Sabit bir Hareketli Engel–Başlama(14,197)-Bitiş(1595,204)

Açısal Hız: Bir hareketli, bir sabit engeldeki açısal hız karşılaştırılması grafikteki gibidir.



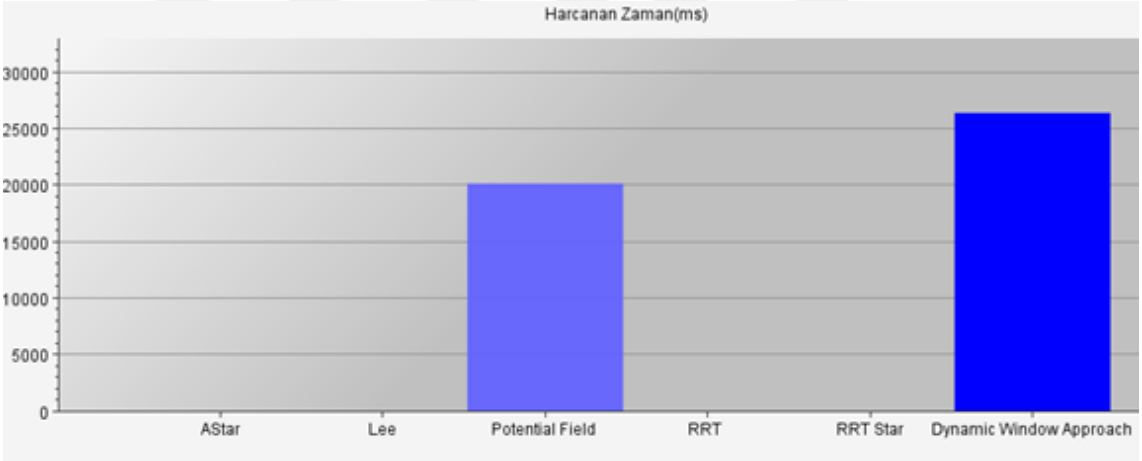
Şekil 4.48 Bir Sabit ve Bir Hareketli Engel PTA ve DWA Açısal Hız Karşılaştırması

Kat Edilen Yol: Bir hareketli , bir sabit engeldeki kat edilen yol karşılaştırılması grafikteki gibidir.



Şekil 4.49 Bir Sabit ve Bir Hareketli Engel PTA ve DWA Kat Edilen Yol Karşılaştırma

Harcanan Zaman: Bir hareketli engel, bir sabit engeldeki harcanan zaman karşılaştırması grafikteki gibidir.



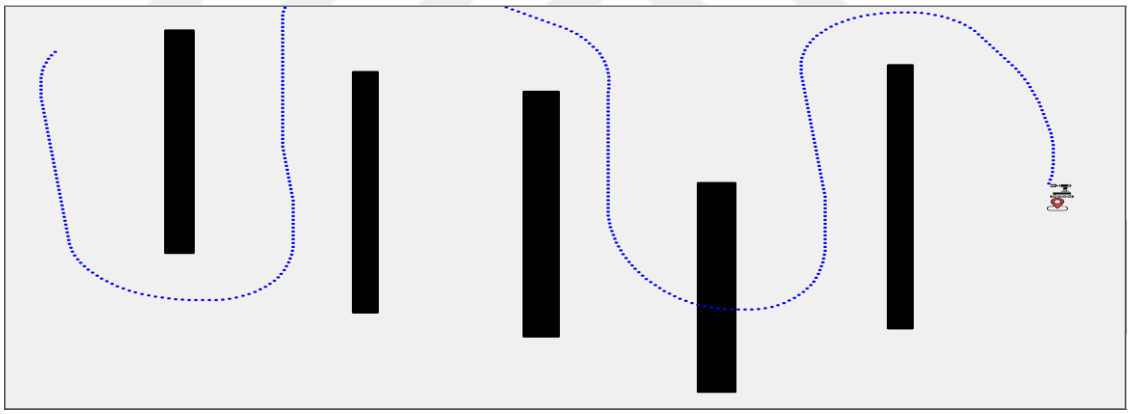
Şekil 4.50 Bir Sabit ve Bir Hareketli Engel PTA ve DWA Harcanan Zaman Karşılaştırma

Çizelge 4.10 Bir Hareketli bir sabit engelde PTA ve DWA sonuçları

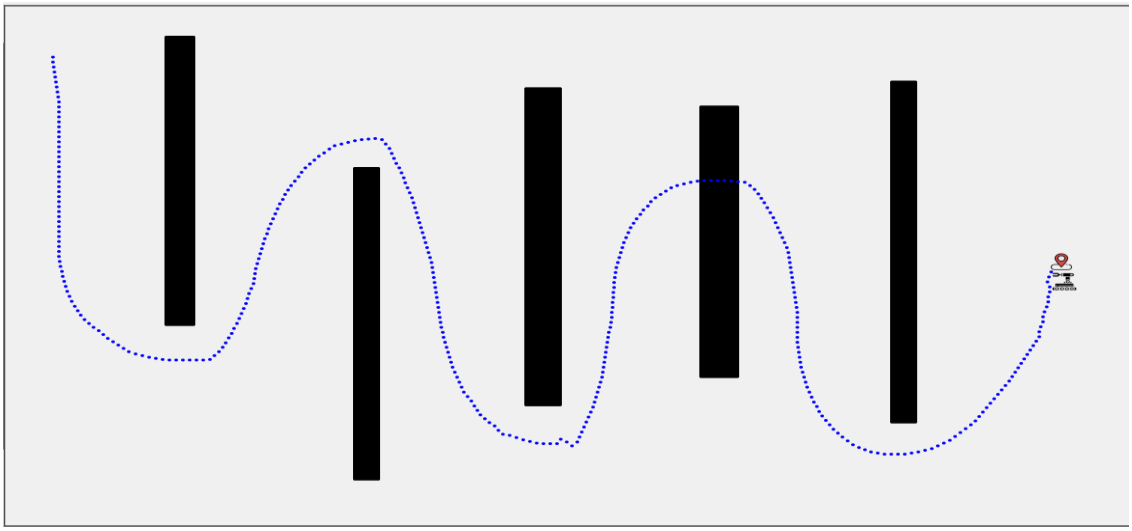
	Açısal Hız(Radyan/milisecond)	Harcanan Zaman(ms)	Kat Edilen Yol(Grid)
Potential Field	0.17	26213	367
Dynamic Window Approach	0.15	16054	492

4.2.5 Üç Hareketli İki Sabit Engel Senaryosu

Üç Hareketli engel ve iki sabit engelde iki algoritma karşılaştırılmıştır. Sonuçları ve güzergahları aşağıdaki görselleştirilmiştir.

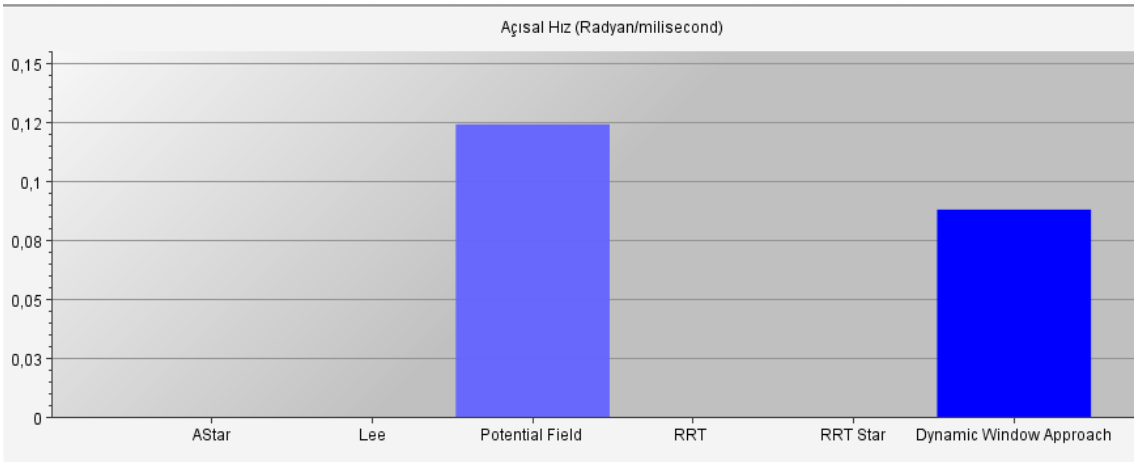


Şekil 4.51 DWA Üç Hareketli Engel ve İki Sabit Engel -Başlama(32,56)-Bitiş(1613,346)



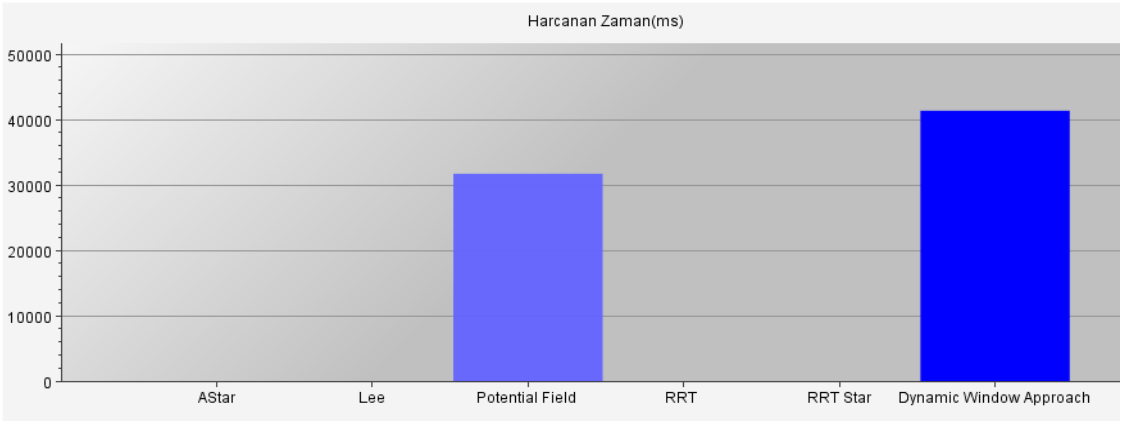
Şekil 4.52 PTA Üç Hareketli ve İki Sabit Engel Başlama(32,56)-Bitiş(1613,346)

Açısal Hız: Güzergahlardan gözlemlenen ortadaki üç engel hareketli dışarıdaki iki engel sabit engeldir. Bu senaryoda iki algoritmanın açısal hız karşılaştırması aşağıdaki gibidir.



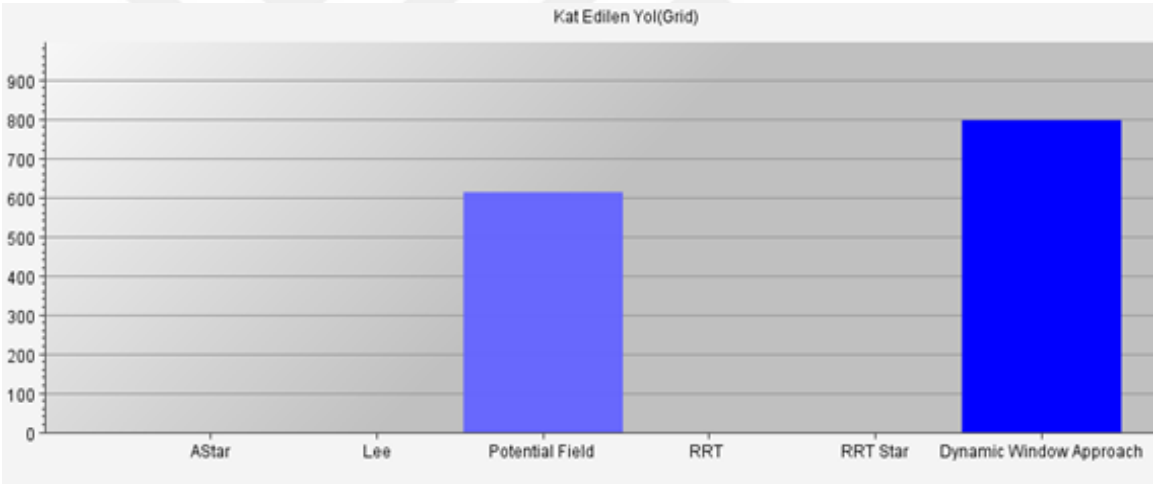
Şekil 4.53 Üç Hareketli İki Sabit Engel PTA ve DWA Açısal hız Karşılaştırması

Harcanan Zaman: Bu senaryoda bu iki algoritmanın harcanan zaman karşılaştırılması aşağıdaki gibidir.



Şekil 4.54 Üç Hareketli ve İki Sabit Engel PTA ve DWA Harcanan Zaman Karşılaştırma

Kat Edilen Yol: Bu senaryoda bu iki algoritmanın kat edilen yol karşılaştırılması aşağıdaki gibidir.



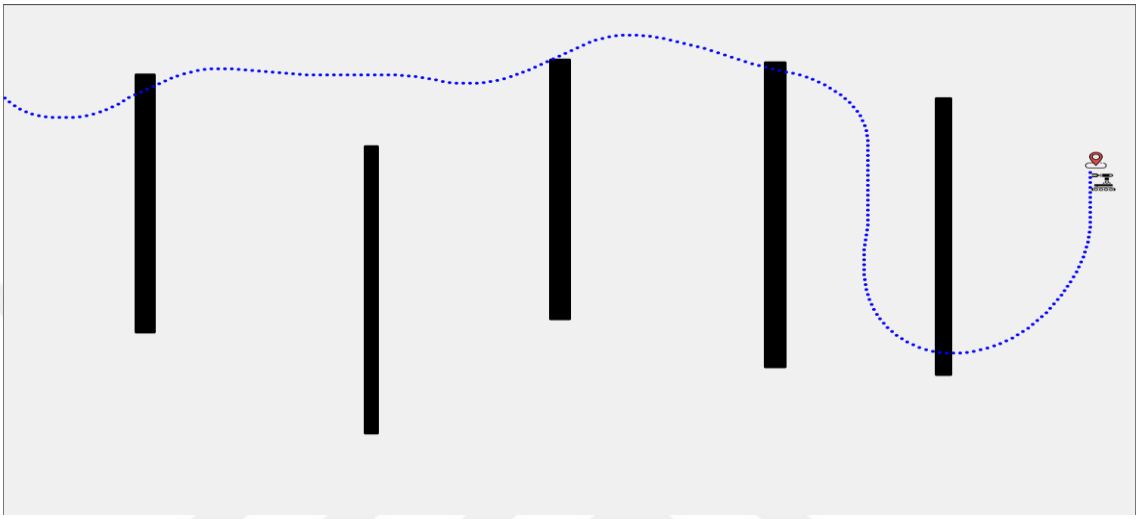
Şekil 4.55 Üç Hareketli ve İki Sabit Engel PTA ve DWA Kat Edilen Yol Karşılaştırması

Çizelge 4.11 Üç Hareketli iki sabit engelde DWA ve PTA Sonuçları

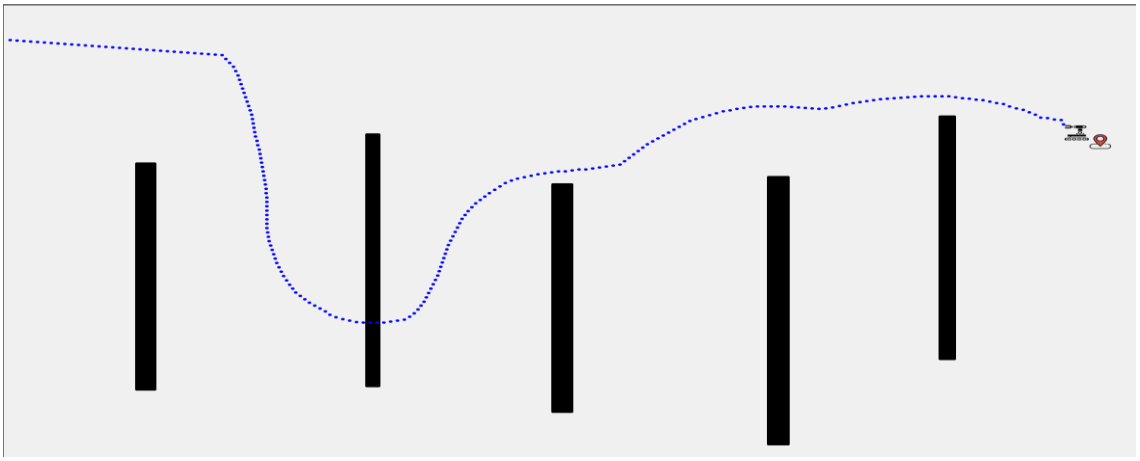
	Açısal Hız(Radyan/milisecond)	Harcanan Zaman(ms)	Kat Edilen Yol(Grid)
Potential Field	0.12	31237	611
Dynamic Window Approach	0.09	41453	802

4.2.6 Beş hareketli engel senaryosu

Beş Hareketli engel iki algoritma karşılaştırılmıştır. Sonuçları ve güzergahları aşağıdaki resimlerde görselleştirilmiştir.

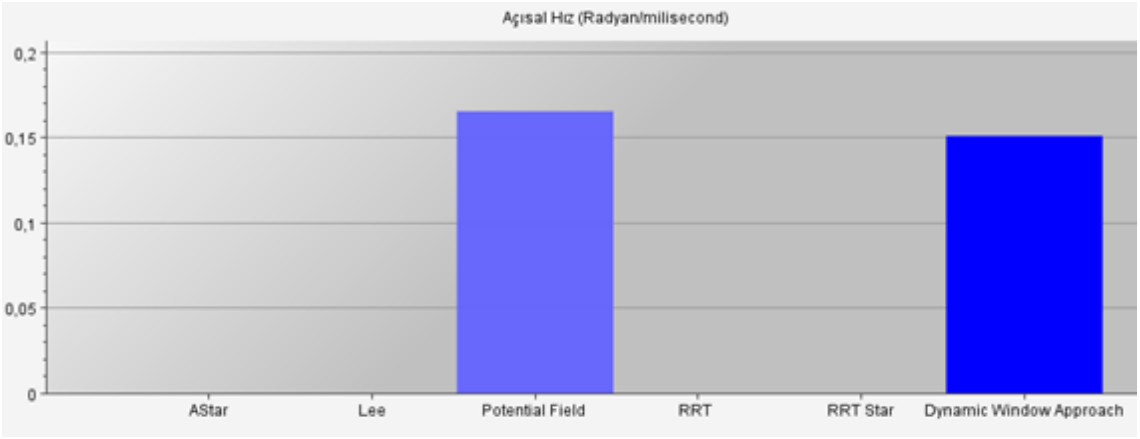


Şekil 4.56 Beş Hareketli Engel DWA Başlama(2,149)-Bitiş Noktası(1613,239)



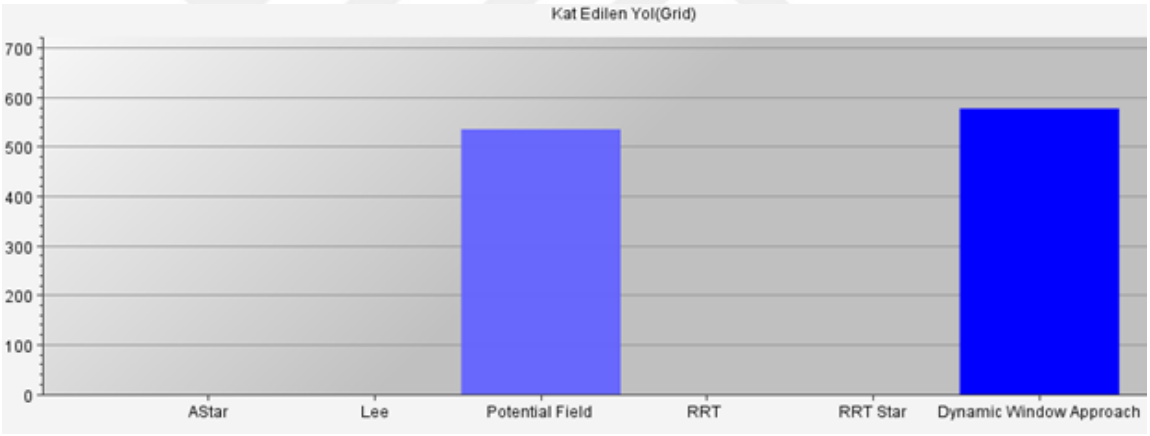
Şekil 4.57 Beş hareketli Engel PTA Başlama Noktası(2,149)- Bitiş Noktası(1613,239)

Açısal Hız: Beş hareketli engel senaryosunda bu iki algoritma açısal hız bakımından karşılaştırılmıştır.



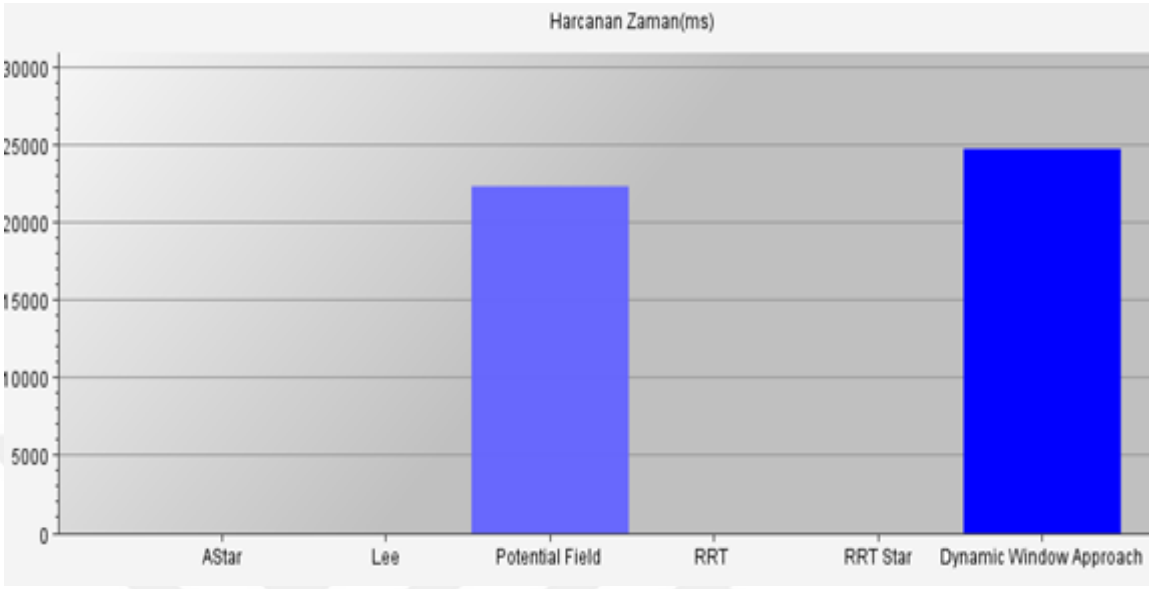
Şekil 4.58 Beş hareketli Engel senaryosu PTA ve DWA Açısal Hız Karşılaştırması

Kat Edilen Yol: Beş hareketli engel senaryosunda bu iki algoritma kat edilen yol bakımından karşılaştırılmıştır.



Şekil 4.59 Beş Hareketli Engel Senaryosu PTA ve DWA Kat Edilen Yol Karşılaştırması

Harcanan Zaman: Beş hareketli engel senaryosunda bu iki algoritma harcanan zaman bakımından karşılaştırılmıştır.



Şekil 4.60 Beş Hareketli Engel PTA ve DWA Harcanan Zaman Karşılaştırma

Çizelge 4.12 Beş Hareketli Engelde DWA ve PTA Sonuçları

	Açısal Hız(Radyan/milisecond)	Harcanan Zaman(ms)	Kat Edilen Yol(Grid)
Potential Field Algoritması	0.17	22382	532
Dynamic Window Approach Algortiması	0.15	24993	583

Öncelikle bu iki algoritma karşılaştırılırken robot yarıçapı, hedef yarıçapı, engelin etki mesafesi gibi iki algoritma için ortak olan bazı parametrelere ortak değerler verilmiştir. Diğer parametreler için hangi değerlerde optimum sonuç alınmışsa parametre değerleri ona göre ayarlanmıştır. Hareketli engellerin aynı şekilde denk gelmesi için testler defalarca tekrarlanıştır.

PTA ve DWA beş farklı senaryoda birbirleri ile kıyaslanmıştır. İlk olarak engelsiz senaryoda PTA değerleri DWA ya göre gayet başarılıdır. Hem hedefe ulaşma süresi hem alınan yol PTA da çok daha başarılı sonuçlar çıkmıştır. Doğal olarak açısal hız bakımından da PTA DWA ya göre gayet başarılı gözükmektedir.

İkinci senaryo tek sabit engel senaryosudur. Bu senaryo sonuçları senaryo bire çok yakındır. Burada PTA nın DWA ya göre üç alanda da bariz üstünlüğünden rahatça belirtilebilir.

Üçüncü senaryo hareketli engel senaryosudur. Bu senaryodaki sonuçlara göre DWA algoritması PTA ya göre daha başarılı sonuçlar çıkarmıştır. Bunun birçok sebebi olabilir. En önemlisi engel sonuçta hareketli olduğu için ilgili anda robotun konumu ve engelin konumu önemlidir. Güzergahlar karşılaştırıldığında PTA nın güzergahının çok daha farklı olup alttan dolaştığı gözlemlenmiştir.

Dördüncü senaryo bir hareketli engel bir sabit engel senaryosudur. Güzergahlara dikkat edildiğinde birbirlerine yakın bir güzergah izlemişlerdir. Ancak DWA nın güzergahları daha kavisli bir yapıya sahiptir. Sonuçlara bakıldığında PTA nın DWA dan daha başarılı sonuçlar çıkardığı gözlenmiştir. Ancak senaryo bir ve senaryo ikideki kadar bariz farklar gözlenmemiştir.

Beşinci senaryo daha karmaşık bir yapıya sahip olup üç hareketli ve iki sabit engelden oluşmaktadır. Bu iki algoritma bu senaryoda karşılaştırıldığında PTA nın güzergahının DWA ya göre net olarak daha başarılı sonuçlar verdiği gözlemlenmiştir. Daha az yol kat ederek daha kısa sürede varış noktasına ulaşan PTA algoritması olmuştur.

Son olarak altıncı senaryo tamamen hareketli engellerden oluşmaktadır. Beş adet hareketli engelden oluşan bu senaryoda yorum yapmak oldukça zorlaşmıştır. Güzergahlar hareketli engel senaryolarına göre birbirine benzer olmasa da alınan sonuçlar birbirne çok yakınsamıştır. Kat edilen yollar karşılaştırıldığında PTA 532 DWA 583 birimde varış noktasına ulaşmıştır. Ayrıca harcanan zaman ve açısal hızlar da

doğal olarak birbirine çok yakın çıkmıştır. Bu senaryoda PTA nın üstünlüğünden söz edilemez.

Ortak parametreler:

- Robot Yarıçapı
- Hedef Yarıçapı
- Robot Sensör Menzil
- Robot Sensör Yoğunluk
- Robot Başlangıç Bakış açısı
- Robot Minimum Hızı
- Robot Maksimum Hızı
- Robot Ortalama İvmesi

Daha verimli bir kıyaslama için bu ortak parametrelerin değerleri de ortak alınmış ve kıyaslama böyle yapılmıştır. Ancak hem DWA nın hem de PTA nın kendi ait farklı parametre değerleri ve sabitleri bulunmaktadır. Onlarda da en optimum değerler belirlenip karşılaştırma yapılmıştır. Sonuç olarak altı senaryo karşılaştırılmış. İlk olarak PTA nın DWA ya göre sabit engellerde daha başarılı bir güzergah çizdiği rahatlıkla söylenmektedir. Bu sabit engel senaryosundaki bariz farka dayandırıp söylenmiştir. Hareketli engel senaryolarına gelindiğinde ise DWA nın PTA ya göre hareketli engellerden kaçınma aşamasında hareketli engelden daha fazla uzaklaşıp düzgün bir kaçış yaptığı görülmektedir. Ancak hareketli engel senaryolarında bu iki algoritmadan hangisini daha başarılı olduğu net olarak söylenememektedir. Sonuçlar birbirine çok yakın çıkmıştır. Ancak güzergahlar birbirine benzememektedir. Burada hareketli engellerin o andaki konumları da çok önemlidir. Karşılaştırmanın doğruluğu için hareketli engelin hızları yine aynı verilmiştir. DWA algoritmasının kendine ait bazı sabit değerleri bulunmaktadır.

- Boş Alan Ağırlığı
- Hız Ağırlığı
- Kafa Ağırlığı

- Tahmin Edilen Zaman
- Sampling Omega
- Örneklenen Hız
- Makismum ACC Omega
- Makimum Hızlanma

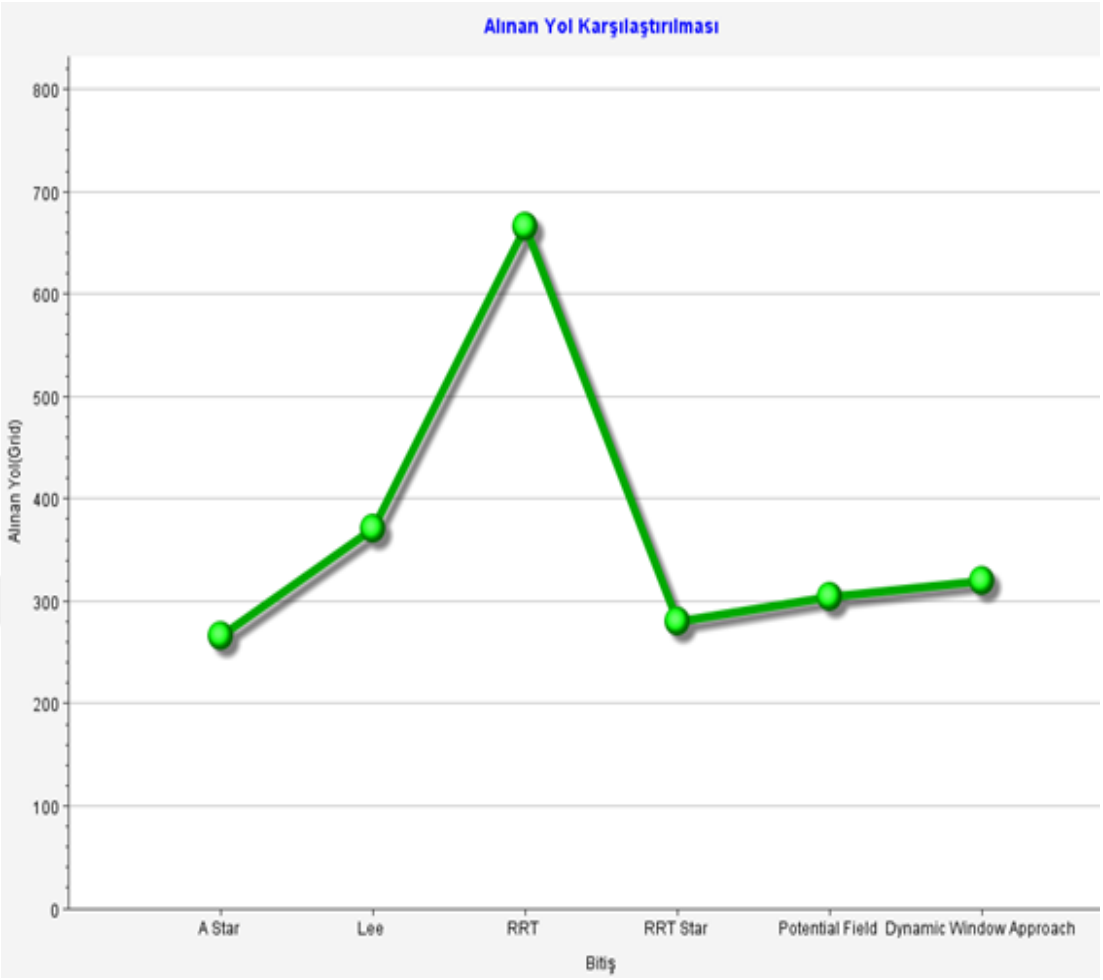
Bu değerler DWA algoritmasının sabit değerleridir. DWA için optimum sabit değerler belirlenmiştir ve buna göre algoritma çalıştırılmıştır.

4.3 Algoritmaların Kat Edilen Yol Bakımından Karşılatırılması

Birbiriyle karşılaştırdığımız A Star algoritması, Lee algoritması, PTA, RRT, RRT Star, DWA algoritmalarının çalışma şekilleri birbirinden farklıdır. Bazıları gerçek zamanda yol bulmaktadır. Bazıları ise yolu önceden hesaplamaktadır. Bu açıdan harcanan zaman ve açısal hız bakımından karşılaştırmak yerine bu algoritmalar kat edilen yol bakımından birbirleriyle karşılaştırıldı. Sonuçları tek grafikte toplanıp gösterildi.

4.3.1 Engelsiz senaryoda algoritmaların kat edilen yol karşılaştırılması

Bu altı algoritma için aynı robot aynı başlama noktasından başlayıp aynı bitiş noktasına ulaştırılmıştır. Başlama noktasından bitiş noktalarına kadar güzergahların mesafesi karşılaştırılmıştır.



Şekil 4.61 Engelsiz Senaryoda Tüm Algoritmaların Alınan Yol Karşılaştırması

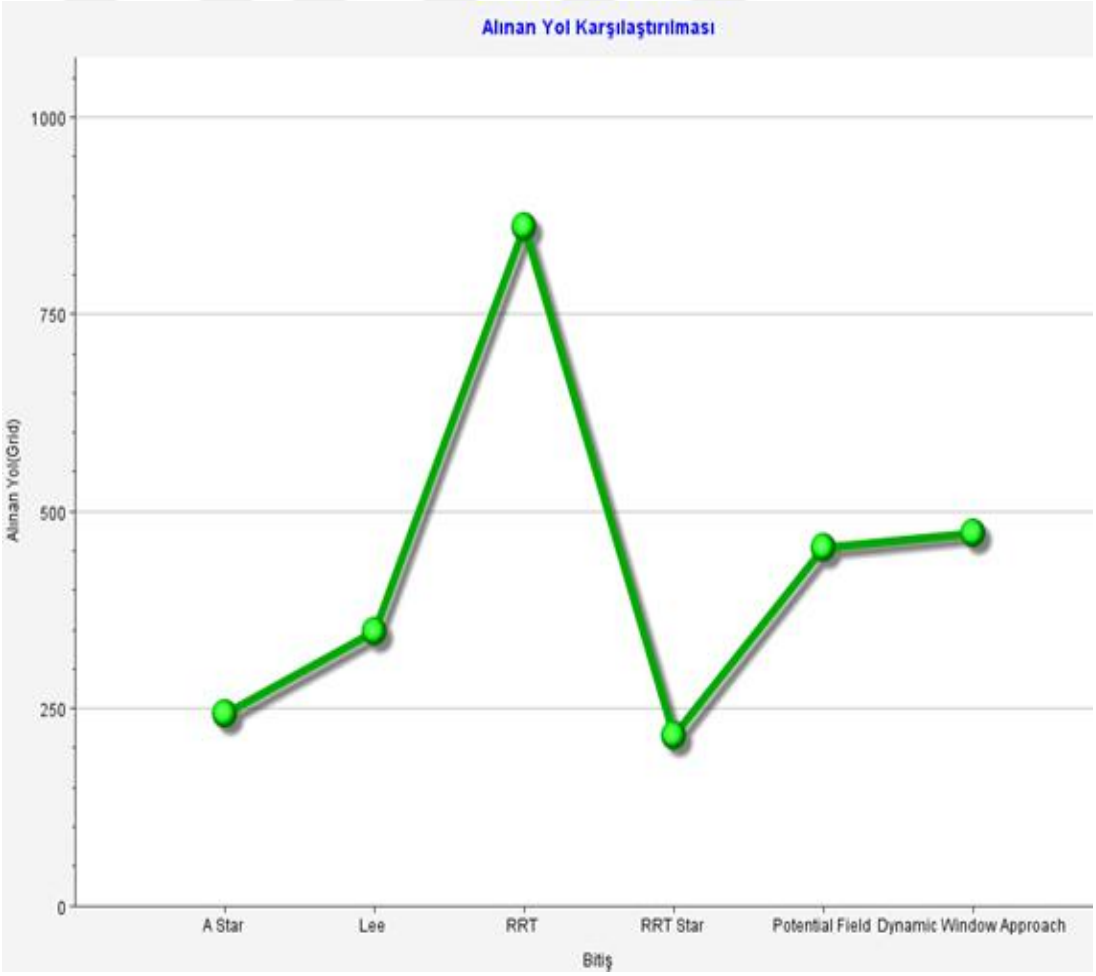
Çizelge 4.13 Engelsiz Senaryo Alınan Yol Sonuçları

Engelsiz Senaryo Sonuçları	Kat Edilen Yol(Grid)
A Star Algoritması	261
Lee Algoritması	363
RRT Algoritması	672
RRT Star Algoritması	274
Potential Field Algoritması	309
Dynamic Window Approach Algoritması	316

Engelsiz senaryo için karşılaştırılan bu durumda çıkarabilecek en düzgün yorum RRT algoritmasının güzergah bakımından pek verimli olmadığıdır. Diğer Beş algoritma birbirlerine yakın bölgelerde kümelenirken RRT algoritması farklı bir güzergah belirlemiş bu güzergah da yolu epey uzatmış görünmektedir. Dolayısıyla engelsiz senaryonun kat edilen yol bakımından en başarısızı RRT algoritmasıdır.

4.3.2 Tek Engelli Senaryoda Algoritmaların Kat Edilen Yol Karşılaştırılması

Bu altı algoritma için aynı robot aynı başlama noktasından aynı bitiş noktasına aynı sabit engel ile çalıştırılmış. Başlama noktasından bitiş noktalarına kadar güzergahların mesafesi karşılaştırılmıştır.



Şekil 4.62 Tek Sabit Engelli Senaryoda Tüm Algoritmaların Alınan Yol Karşılaştırması

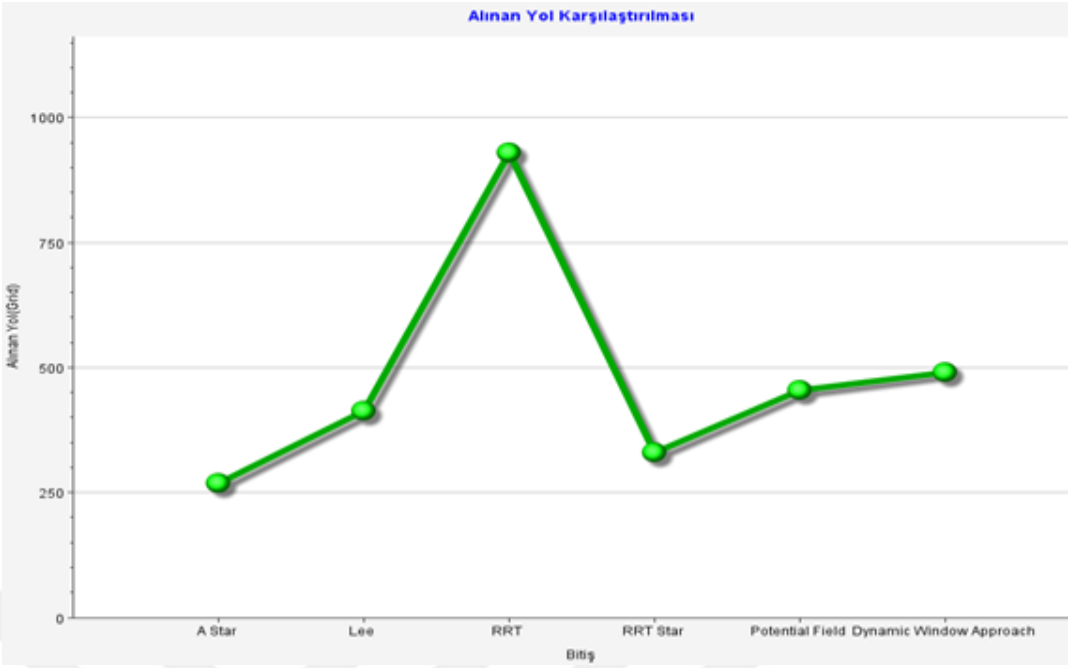
RRT algoritması kat edilen yol bakımından yine en fazla mesafe kaydeden ,en verimsiz algoritma olmuştur. A Star algoritması ise en verimli yolu seçmiş, en kısa güzergahı bulmuştur. Altı algoritmadan en başarılısı Astar olmuştur.

Çizelge 4.14 Tek Engel Senaryosu Alınan Yol Sonuçları

Tek Engel Senaryo Sonuçları	Kat Edilen Yol(Grid)
A Star Algoritması	247
Lee Algoritması	351
RRT Algoritması	856
RRT Star Algoritması	238
Potential Field Algoritması	461
Dynamic Window Approac Algohrtiması	483

4.3.3 Üç engelli senaryoda algoritmaların kat edilen yol karşılaştırılması

Bu altı algoritma için aynı robot aynı başlama noktasından aynı bitiş noktasına aynı üç sabit engelle ile çalıştırılmış. Başlama noktasından bitiş noktalarına kadar güzergahların mesafesi karşılaştırılmıştır.



Şekil 4.63 Üç Sabit Engelli Senaryoda Tüm Algoritmaların Alınan Yol Karşılaştırması

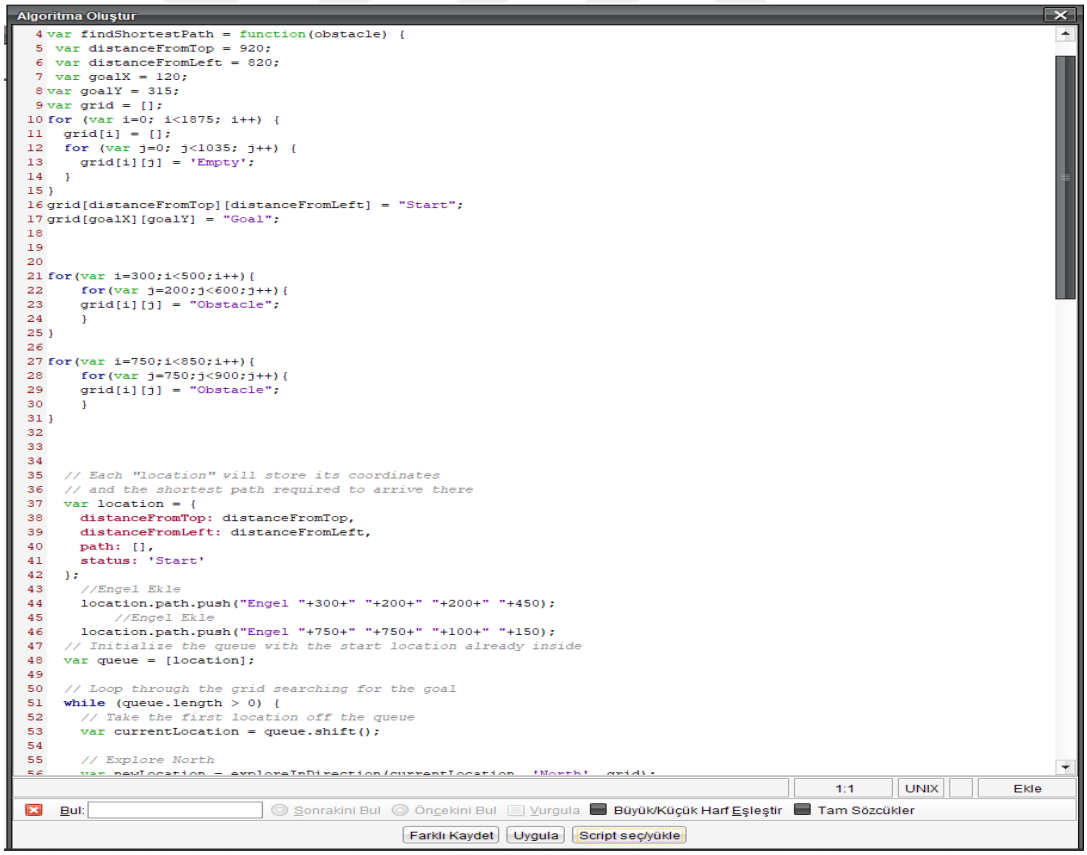
Çizelge 4.15 Üç Sabit Engel senaryosunda alınan yol sonuçları

	Kat Edilen Yol(Grid)
A Star Algoritması	261
Lee Algoritması	413
RRT Algoritması	952
RRT Star Algoritması	363
Potential Field Algoritması	472
Dynamic Window Approach Algortiması	498

Bu üç engelli senaryoda RRT algoritması verimsiz algoritmalar arasında yer almıştır. Bu senaryoda güzergah bakımından Astar ve RRT Star algoritmaları en etkili algoritmalar arasında yerini almıştır.

4.4 Benzetim Ortamı Geliştirme Makro Yorumlayıcısı

Bu karşılaştırılan beş algoritmanın yanı sıra yazılımın ayrı bir yeteneği olarak kullanıcı kendi algoritmasını da kodlayabilecektir. Kullanıcı kendi algoritmasını JavaScript dilinde geliştirir. Kodlanan bu scriptler nashorn script engine ile işlenerek yazılıma uygun hale getirilir. Kullanıcı kendi algoritmasında kendi engellerini ve kendi başlama ve bitiş noktasını belirtir. Scriptten bu başlama bitiş noktası ve engellere göre bir yol hesaplanır. Mobil robot da bu hesaplanan yola göre çizdirilir. Kullanıcı için javascript söz dizimine renklendirmesi işlemi jide framework ile yapılmıştır. Hareketli engellerde anlık olarak engelin yeri değiştiğinden hareketli engellerde bu algoritma çalışmaz. Bu yetenek sabit engel senaryolarında çalışabilir. Kulanıcının algoritması ile döndürdüğü nesne ayrıştırılarak engeller ve izlenecek yol bulunur. Script işlenmesinden sonra gezgin robot çizdirme işlemi yapılır. Örnek script ekranı aşağıdaki şekilde gösterilmiştir.



Şekil 4.64 Örnek Script Ekranı

Örnek script ekranı şekilde görünmektedir. Burada kodlamada söz dizimine göre renklendirme yapılmaktadır. Yazılan scriptler kaydedilebilir ve geri yüklenebilir. Uygula düğmesi ile scriptler de yazılımda işlenmeye başlar. İşlenme sonucu olarak robot gidiş yolu ve engel dikdörtgenleri elde edilir. Elde edilen değerlere göre yazılımda çizim işlemi yapılır. Önce engeller ardından başlangıç noktasından hedef noktasına mobil robotlar gösterilir. Scriptte algoritma yazılırken dikkat edilmesi gereken bazı hususlar vardır. Bunlar aşağıda adım adım belirtilmiştir.

1. İlk olarak scriptte bir method tanımlanmalıdır. Bu methodun adı findShortestPath() olmalıdır. Algoritma işletilirken bu method aranacaktır.
2. Scriptde method adı findShortestPath() yapıldıktan sonra script içeriğinde ilk olarak ekranın maksimum genişlik ve yükseklik bilgisi tanımlanır.(Genişlik 1875, Yükseklik 1035)

```
for (var i=0; i<1875; i++) {
  grid[i] = [];
  for (var j=0; j<1035; j++) {
    grid[i][j] = 'Empty';
  }
}
```

Şekil 4.65 Boş Alan Tanımı

3. Başlama Noktası ve bitiş noktası belirlenir.

```
var distanceFromTop = 920;
var distanceFromLeft = 820;
var goalX = 120;
var goalY = 315;
```

Şekil 4.66 Başlangıç ve Btişi Noktaları

4. Labirentte sabit engeller tanımlanır.

```
grid[distanceFromTop][distanceFromLeft] = "Start";
grid[goalX][goalY] = "Goal";



for(var i=300;i<500;i++){
    for(var j=200;j<600;j++){
    grid[i][j] = "Obstacle";
    }
}
```

Şekil 4.67 Engel Tanımı

5. Bu labirentten başlama ve bitiş noktası göre en kısa yol çıkarılır.
6. En kısa yol güzergahı ve engeller belli bir formatta yazılıma döndürülür. Format ayrıntısı şu şekildedir. Önce dikdörtgen şeklindeki engeller belirtilir. "Engel X1,Y1,X2,Y2" belirtilir . Başında engel yazısına göre engeller ayrıştırılır ve çizilir. Ardından en kısa yol koordinatları .(X,Y) formatında yazılır İlk koordinat başlama noktası son koordinat bitiş noktası olarak kabul edilir. Aradaki tüm noktalar en kısa yolu verir. Örnek algoritma çıktısı aşağıdaki şekildeki gibi olmalıdır.

```
Engel 200 200 300 450
Engel 600 500 750 620
90 240
91 240
92 240
93 240
94 240
95 240
96 240
97 240
98 240
99 240
100 240
101 240
102 240
103 240
104 240
105 240
106 240
107 240
```

Örneğin bu şekilde dönen bir sonuçta engeller öncelikli çizilir. Ardından engelsiz ilk koordinat başlama noktası(90,140) son koordinat (107,240) olarak kabul edilir ve robot

çizilir. Aradaki her koordinat her iterasyondaki bir adımdır. Bu adımlara göre yazılım tarafından mobil robot hareket ettirilmektedir.



Şekil 4.68 Algoritma Çalıştırıldıktan Sonra Ekrandaki Görüntü



Şekil 4.69 Mobil robotun hedefe doğru çizdirilmesi

Engelllere göre çizim bu şekilde oluşturulmuştur. Çizim sonuçları kaydedilmektedir. Açısal hız, alınan yol verileri tutulmaktadır. Bu verilere göre farklı kullanıcıların algoritmaları kıyaslanabilir. Daha optimum sonuçlara sahip algoritmalar seçilir.

5. SONUÇ

Günümüz dünyasında popüleritesi gittikçe artmakta olan robotik alanında arz talep ile de birlikte dikkate değer ilerlemeler olmuştur. Bu alanda gerek akademik gerek sektörel anlamda önceki yıllara göre üstel bir artış gözlemlenmektedir.

Tez kapsamında robotik alanının alt konularından olan robot hareket kontrolü konusu dahilinde çalışmalar yapılmıştır. Robot yer bulma için altı farklı algoritma ayrı ayrı incelenmiş. Farklı açılardan birbirleriyle kıyaslanmış ve sonuçlar yorumlanmış. Bu altı algoritmanın birbirlerine karşı doğal olarak avantajları ve dezavantajları vardır. Bu yüzden kesin önermeler yerine durumlara göre çıkarımlar yapılmış ve yorumlanmıştır.

PTA ve DWA algoritmaları hareketli engellerden gerçek zamanda kaçınabilen algoritmalardır. Hareketli engelin geliş açısına ve hızına göre farklı güzergah belirleyebilen hızlanıp yavaşlayabilen yani kısacası gerçek ortamda daha düzgün çalışabilecek iki algoritmadır. Dinamik zamanda yol hesaplayıp çizen bu algoritmalar doğal olarak diğer algoritmalardan daha maliyetlidir. Her iterasyonda engellerin konumu ve hızı hesaplanıp , fizik kurallarına göre yeni güzergah oluşturulur. Diğer algoritmalar gidilecek güzergahı daha önceden hesaplar ve mobil robot ona göre hareket eder. Yani kısacası diğer algoritmaların hareketli engellerden kaçınma şansı yoktur. Sabit engellerin konumuna göre önceden bir yol oluşturup robot bu yolda götürülür. Doğal olarak dinamik zamanda çalışan algoritmalara göre daha az maliyetlidir. RRT* algoritması, RRT algoritmasının optimize edilmiş halidir. Ve doğal olarak RRT Star algoritmasındaki sonuçlar daha başarılıdır. Lee algoritmasının gerçeklenmesinde çarpraz yönler tanımlanmamıştır ve diğer algoritmalara göre daha uzun mesafe almasının nedeni budur. Normalde Lee algoritması bulunan en kısa mesafeyi her zaman bulur ancak diğer algoritmalara göre bellek açısından fazlaca maliyetli bir algoritmadır. A Star algoritması sezgisel yaklaşım dayanan bir algoritmadır ve Bu sezgisel yaklaşım dan ne kadar başarılı sonuçlar çıkarsa A Star algoritması da o kadar başarılı olur. Gerçeklenen A Star algoritmasının sezgisel yaklaşımı(The Euclidean Distance) başarılı olmuş ve kat edilen yol ve açısal hız bakımından başarılı sonuçlar gözlenmiştir. Özet olarak Lee algoritması garantici bir algoritma olup doğal olarak maliyetli bir yapısı

vardır. Astar , RRT ve RRTStar algoritmaları sezgisel bir yaklaşımı vardır. Bu yüzden de farklı senaryolara göre farklı sonuçlar gözlemlenebilir bu algoritmalarda. Kullandıkları sezgisel yaklaşım bu algoritmaların verimini belirler. Diğer DWA ve PTA algoritmalarını da kendi arasında kıyaslamak daha düzgün olacaktır. PTA algoritmasıyla DWA farklı senaryolarda karşılaştırılmış ve PTA nın genel olarak daha başarılı sonuçları gözlemlenmiştir. Ancak DWA algoritmasının hareketli engellerden kaçınmasının daha başarılı olduğu gözlemlenmiştir. PTA ve DWA nın hareketli engel senaryolarında karşılaştırılmasında bu iki algoritmanın birbirlerine hiçbir senaryoda bariz bir üstünlüğü olmamıştır. Bu iki algoritmanın gerçeklenmesinde birçok parametre vardır. Ortak parametreler bu iki algoritma için de ortaklanmıştır. Algoritmalar arasında bu şekilde karşılaştırma yapılmıştır.

Sonuç olarak bu algoritmalar seçimi kullanılacak ortam ve robotun donanımına göre karar verilmelidir. Eğer ortamda hareketli engel varsa mecburen DWA ya da PTA algoritmalarından biri kullanılmalıdır. Bu ikisinden hangisinin seçileceği ise amaca göre değişmektedir. Eğer kullanılan mobil robot çok değerli ise engellerden kaçmak ilk amaçtır. Bu alanda ise DWA algoritması daha başarılıdır. Eğer amaç hedefe daha kısa sürede gitmekse PTA algoritması tercih edilir. PTA algoritması çoğunlukla daha optimize güzergahlar bulmaktadır. Eğer ortamda hareketli engel yoksa güzergah önceden hesaplanıp yola çıkılabilir. Buna bağlı olarak Lee, Astar ve RRT ve türevleri tercih edilebilir. Eğer kullanılan mobil robotun donanıma çok üst düzeyse Lee algoritması rahatlıkla seçilebilir. Ancak kullanılan mobil robotun donanımı yeterli değilse Astar ve RRT Star arasında kullanılan sezgisel algoritmaya göre bir tercih yapılmalıdır. Projenin kaynak kodları githubda paylaşıma açılmıştır (Karakaya, 2020).

KAYNAKLAR

Abbee Pieter Motion Planning // https://people.eecs.berkeley.edu/. Berkeley Edu 2011. - http://portal.ku.edu.tr/~cbasdogan/Courses /Robotics/projects/algorithm _poten_field.pdf.

Abbee Pieter people.eecs.berkeley.edu Berkeley Education Technic University. - https://people.eecs.berkeley.edu/~pabbeel/cs287-fa11/slides/MotionPlanning-v1.pdf.

Alcazar Vidal, Veloso Manuel ve Borrajo Daniel Adapting a Rapidly exploring Random Tree for Automated Planning Proceedings, The Fourth International Symposium on Combinatorial Search(SoCS-2011). - 2011.

Antonelo Eric, Çevrimiçi] // http://ericantonelo.com/. - 21 09 2020. - 21 09 2020Tree for ericantonelo.com/portfolio/autonomous-robot-navigation-simulator-186.

Baijayanta Roy towardsdatascience.com - 29 Eylül 2019. - https://towardsdatascience .com/a-star-a-search-algorithm-eb495fb156bb.

Bellemare G., Naddaf, Y., Veness, J. ve Bowling, M. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research. - 2013. - s. 47:253–279.

Bodrenko A. I. New Method Of Using Mobile Robots For Moving Cargo In Warehouse [Kitap]. - [s.l.] : Bulletin of Science and Practice., 2019.

Brock Oliver ve Khatib Oussama High-speed Navigation Using the Global Dynamic Window Approach. - Detroit. Michigan : Robotics Laboratory, Department of Computer Science Stanford University,, 1999.

Choset H. Probabilistic Roadmap Path Planning http://www.cs.columbia.edu/. - 24 08 2019. - http://www.cs.columbia.edu/~allen/F15/NOTES/Probabilisticpath.pdf.

Çelik İbrahim Baran ve Güzel Mehmet Serdar Multi-Agent Robot Coordination Framework for Goal based Navigation Problem // Second International Conference on Advances in Information Processing and Communication Technology,. - Rome Italy : [s.n.], 2015.

Dieter Foxy, Burgardy Wolfram ve Thrun Sebastian The Dynamic Window Approach to Collision Avoidance. - [s.l.] : IEEE Robotics & Automation Magazine, 1997.

EECS Pieter Abbeel UC Berkeley Motion Planning. - [s.l.] : Lavalle.

Erden Abdülkadir Robot Teknolojileri Uygulama ve Araştırma Merkezi. - Ankara : Atılım Üniversitesi, 2013.

Fakoor Mahdi, Kosari Amirreza ve Jafarzadeh Mohsen Humanoid robot path planning with fuzzy Markov decision processes. - [s.l.] : Journal of Applied Research and Technology, 2016. - 300-310 : Cilt 5.

Fukushima K. Neocognitron: A hierarchical neural network capable of visual pattern recognition // Neural Networks. - 1988. - s. Vol 1, 119–130.

Guevara Bryan Cardenas, Vreeswijk Dr G.A.W. ve Rin Dr Benjamin An Overview of the Class of Rapidly-Exploring Random Trees - Utrecht : Faculty of Humanities Artificial Intelligence Utrecht University, 2018.

Gupta Naman http://vlsi-soc.blogspot.com/. - Maze Router(Lee'sAlgorithm), 16 Aralık 2018. - http://vlsi-soc.blogspot.com/2018/12/maze-router-lees-algorithm.html.

Güzel Mehmet Serdar, Nattharith Panus ve Duran Ahmet S. Goal Oriented Navigation System Using Vision. // UTEM. - 2017. - s. Cilt 9 2-3, s.l..

Hastie T., Tibshirani, R. ve Friedman, J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction // Springer. - 1999.

Hauer Florian Deformable Rapidly-Exploring Random Trees Florian Hauer School of Aerospace Engineering Georgia Institute of Technology, Atlanta, School of Aerospace Engineering Georgia Institute of Technology. - Atlanta : [s.n.], 2006.

Huang S. [et al.] Adversarial Attacks on Neural Network Policies.. - 2017.

İbrahim Baran Çelik ve Mehmet Serdar Güzel A Multi-Agent Robot Coordination Framework for Goal based Navigation Problem. - Rome, Italy : Second International Conference on Advances in Information Processing and Communication Technology, 18 - 19 April, 2015.

Karakaya Mustafa https://github.com [Çevrimiçi]. - 20 09 2020. - 21 09 2020. - mukk38/RobotSimulator.

Karaman Sertaç ve Frazzoli Emilio Sampling-based algorithms for optimal motion planning. - [s.l.] : The international Journal of Robotics, 2011.

Knudsen Gade M. [et al.] Appying machine learning to robocode. Technical report, Department of Computer Science, Aalborg University.. - 2003.

Kobayashi K., Uchida Y. ve Watanabe K. A study of battle strategy for the Robocode SICE Annual Conference in Fukui. - 2003.

Koenig Nathan ve Howard Andrew Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator 4 IEEE!RSJ International Conference on Intelligent Robots and Systems. - Sendai, Japan : EEE!, 2004.

Kuffner James J. ve LaValle Steven M. Rapidly-Exploring Random Trees: Progress and Prospects SM Lavalle. - 2000.

Lan Chabini Adaptations of the A* Algorithm for the Computation of Fastest Paths in Deterministic Discrete-Time Dynamic Networks. Dept. of Civil & Environ. Eng., MIT , Cambridge. - MA USA : [s.n.], 2002.

Latombe Jean Claude, Bur O. ve Chan Bayazit http://portal.ku.edu.tr/ Koç Üniversitesi. - http://portal.ku.edu.tr/~cbasdogan/Courses/Robotics/projects/algorithm_poten_field.pdf.

LeCun Y. [et al.] Gradient-based learning applied to document recognition. // IEEE. - 1998. - s. 2278–2324.

Lindermann S.R. ve Lavalle S.M. Steps toward derandomizing RRTs - [s.l.] : Proceedings of the Fourth International Workshop on Robot Motion and Control, 2004. - 271-277.

Mazzari Vanessa https://blog.generationrobots.com. - 2016. - 19 06 2020. - https://blog.generationrobots.com/en/ros-robot-operating-system-2/.

McCarthy J., Shannon C. ve Goldsine H. Automata Studies // Princeton: Princeton University Press. - 1956.

McDermott Drew Robot Planning. - [s.l.] : AI Magazine, 1992. - 2 : Cilt 13.

Mehmet Serdar Güzel Panus Nattharith, Ahmet S. Duran A Goal Oriented Navigation System Using Vision. - [s.l.] : UTEM, 2017. - 2-3 : Cilt 9.

Mnih V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K. ve Ostrovski, G. Human-level control through deep reinforcement learning. Nature, - 2015. - s. 518(7540):529–533..

Mousas C, Newbury, P. Anagnostopoulos The minimum expenditure shortest path method. A clustering technique. // Journal of Graphics Tools, vol 17. - 2013. - s. 1-44.

Nair V. ve Hinton G. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning ICML. - 2010. - s. 807–814.

Nattharith Panus ve Güzel Mehmet Serdar Machine vision and fuzzy logic-based navigation control of a goal-oriented mobile robot SAGE. - 2016. - s. Cilt 24 3, s.l..

Oliver Brock 0ussama.Khatib High-speed Navigation Using the Global Dynamic Window Approach. - Detroit. Michiga : lEEE , Mayıs 1999.

Panus Nattharith Mehmet Serdar Güzel Machine vision and fuzzy logic-based navigation control of a goal-oriented mobile robot . - [s.l.] : SAGE, 2016. - 3 : Cilt 24.

Patel Amit http://theory.stanford.edu/. - 08 09 2019. - http://theory.stanford.edu/~amitp/GameProgramming/.

Petrovi Marija Seder and Ivan Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. - Roma, Italy : 2007 IEEE International Conference on Robotics and Automation , 10-14 April 2007.

Petrovic Ivan ve Seder Marija Dynamic window based approach to mobile robot motion control in the presence of moving obstacles. - Roma , Italy : 2007 IEEE International Conference on Robotics and Automation, 2007. - ThA11.3.

Riedmiller M. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In Machine Learning: ECML. - 2005. - s. 317-328.

Router Maze http://users.eecs.northwestern.edu. - 02 02 2020. - http://users.eecs.northwestern.edu/~haizhou/357/lec6.pdf.

Sallans B ve Hinton G Reinforcement learning with factored states and actions. Journal of Machine Learning Research 5 . - 2004. - s. 1063–1088.

Samuel A. Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development. - 1959. - s. 3-3.

Silver D. [et al.] astering the game of go with deep neural networks and tree search. Nature. - 2016. - s. 529:484–503.

Simmons Reid ve Urmson Chris Approaches for heuristically biasing RRT growth . - [s.l.] : Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.

Sutton R.S., Barto, A.G. Reinforcement Learning. An Introduction. Cambridge, MA: MIT Press . - 2017.

Şişman B., Odabaşı H. F. ve Akkoyunlu B. Eğitimde Robot Kullanımı . - Ankara : TOJET, 2016. - 299-314.

Tang Daisy https://www.cpp.edu/. - 01 01 2017. - 20 06 2020. - https://www.cpp.edu/~ftang/courses/player%20stage/player-stage-intro.htm.

Tesauro G. Temporal difference learning and td-gammon. Communications of the ACM,. - 1995. - s. 38(3):58–68..

Wolf Joerg Christian, Robinson Paul ve Davies Mansel Vector Field path planning and control of an autonomous robot in a dynamic environment 2004 FIRA Robot World Congress. - Busan, South Korea : [s.n.], 2004.

Wolfram Burgard Cyrill Stachniss, Maren Bennewitz, Kai Arras Introduction to Mobile Robotics. - Freiburg : [s.n.], July 2011 .

Wolfram Burgrad [et al.] Introduction to Mobile Robotics / Robot Motion Planning. - 2011.

Yap P. Grid-Based Path-Finding. Department of Computing Science, University of Alberta, Edmonton, Canada.. - 2002.

ÖZGEÇMİŞ

Adı Soyadı : Mustafa KARAKAYA

Doğum Yeri : Ankara

Doğum Tarihi : 04/06/1992

Medeni Hali : Bekar

Yabancı Dili : İngilizce

Eğitim Durumu (Kurum ve Yıl)

Lise : Ankara Gazi Anadolu Lisesi (2010)

Lisans : Hacettepe Üniversitesi Bilgisayar Mühendisliği Bölümü (2015)

Yüksek Lisans : Ankara Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı (Ekim 2020)

Çalıştığı Kurum/Kurumlar ve Yıl

CyberSoft (2015-2016)

CsTech (2017)

Simsoft Bilgisayar Teknolojileri Ltd. Şti. (2017 - …)

Ulusal Konferans Sunum

Karakaya Mustafa, GÜZEL Mehmet Serdar, BOSTANCI Gazi Erkan. 2020/03/21,Education Based Robotic Application Development Framework, OECD Studies Conference On Engineering, Technology and Robotics , 21-37 , Ankara ISBN: 978-625-7914-91-8