KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRONİK VE BİLGİSAYAR EĞİTİMİ
ANABİLİM DALI

DOKTORA TEZİ

SONAR ALGILAYICILAR VE SEZGİSEL YÖNTEMLER KULLANARAK OTONOM ROBOTLARIN ENGELDEN SAKINIMI

RASİM TOPUZ

KOCAELİ 2016

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRONİK VE BİLGİSAYAR EĞİTİMİ

ANABİLİM DALI

DOKTORA TEZİ

SONAR ALGILAYICILAR VE SEZGİSEL YÖNTEMLER KULLANARAK OTONOM ROBOTLARIN ENGELDEN SAKINIMI

RASİM TOPUZ

Prof.Dr. Mehmet YILDIRIM
Danışman, Kocaeli Üniv. .....

Doç.Dr. Emine DOĞRU BOLAT
Jüri Üyesi, Kocaeli Üniv. .....

Yrd.Doç.Dr. Umut ALTINIŞIK
Jüri Üyesi, Kocaeli Üniv. .....

Yrd.Doç.Dr. Ali ÇALHAN
Jüri Üyesi, Düzce Üniv. .....

Yrd.Doç.Dr. Adem TUNCER
Jüri Üyesi, Yalova Üniv. .....

Tezin Savunulduğu Tarih: 20.06.2016

ÖNSÖZ ve TEŞEKKÜR

Otonom robotlar günümüzde birçok alanda önemli görevler icra etmektedirler. Robot, kendisine verilen bir görevi yerine getirirken, hedefe ilerlemek için yol bulma problemini çözmeli, ayrıca kendi bekasını korumak için, algılayıcıları yardımıyla çevreden elde ettiği verileri kullanarak varsa olası bir çarpışmayı önlemek için statik veya dinamik olarak engellerden sakınma yapabilmelidir.

Bu tez çalışmasında, sonar algılayıcılar ve sezgisel yöntemler kullanarak, otonom robotların engelden sakınma probleminin çok yönlü optimizasyonunun yapılması sağlanmıştır. Bu kapsamda robotların engelden sakınma ve yol bulma probleminin çözümüne ilişkin bir hibrit otonom robot seyir sistemi geliştirilerek, yeni algoritmaların bu sistem üzerinde gerçeklenmesi sağlanmıştır.

Gerek hazırlamış olduğum tezin ve gerekse tez kapsamında oluşturulan seyir sisteminin bundan sonraki otonom robotlar üzerindeki çalışmalarda, başta eğitim ve gerçekleme aşamaları olmak üzere, tüm aşamalarda katkı sağlamasını umuyorum.

Tez çalışması süresince her türlü desteği bana sunan, yaptığım çalışmaların tüm aşamalarında bana yardımlarını esirgemeyen danışmanım Prof.Dr. Mehmet YILDIRIM'a, değerli hocalarım Doç.Dr. Emine DOĞRU BOLAT ve Yrd.Doç.Dr. Umut ALTINIŞIK'a ayrıca Pioneer 3-DX robot ile ilgili değerli katkıları nedeniyle Yrd.Doç.Dr. Adem TUNCER'e teşekkür ederim.

Bu süreçte bana büyük destek veren eşime, çalışmalarım süresince kendilerine yeterli zaman ayıramadığım canlarım; kızım ve oğlum ile tüm hayatım boyunca her türlü çalışmamda arkamda olan annem, babam ve kardeşlerime verdikleri destek için çok teşekkür ederim.

Haziran - 2016 Rasim TOPUZ

İÇİNDEKİLER

ÖNSÖZ ve TEŞEKKÜR i
İÇİNDEKİLER ii
ŞEKİLLER DİZİNİ iv
TABLOLAR DİZİNİ vii
SİMGELER VE KISALTMALAR DİZİNİ viii
ÖZET ix
ABSTRACT x
GİRİŞ 1
1. OTONOM ROBOTLAR 6
1.1. Robotun Tanımı ve Tarihçesi 6
1.2. Robotların Çeşitleri 7
1.2.1. Endüstiyel robotlar/manipulatör/kollar 7
1.2.2. Mobil robotlar 8
1.2.2.1. Ayaklı robotlar 9
1.2.2.2. Tekerlekli robotlar 10
1.2.2.3. Otonom denizaltı araçları 11
1.2.2.4. İnsansız hava araçları 11
1.3. Robotların Sahip Olması Gereken Üç Yetenek 12
1.3.1. Algılayıcılar 12
1.3.1.1. Ultrasonik mesafe bulucular 13
1.3.1.2. Lazer mesafe bulucular 15
1.3.1.3. Kızılötesi (Infrared) yakınlık algılayıcıları 15
1.3.1.4. Takometreler 15
1.3.1.5. Optik kodlayıcılar (Encoders) 15
1.3.1.6. Cayrolar ve pusulalar 16
1.3.1.7. Görüntü tabanlı algılayıcılar 16
1.3.1.8. Global konumlandırma sistemi (GPS) 17
1.3.2. Kontrol bilgisayarı 17
1.3.3. Eyleyiciler 17
2. YOL BULMA 18
2.1. Lokalizasyon 18
2.1.1. Izgara tabanlı haritalama 18
2.1.2. Öz nitelik tabanlı haritalama 20
2.2. Yol Bulma Algoritmaları 20
2.2.1. Dijkstra algoritması 21
2.2.2. A* (A yıldız) algoritması 23
2.2.3. Karınca kolonisi algoritması 24
2.2.4. Genetik algoritma 25
3. ENGELDEN SAKINMA ALGORİTMALARI 28
3.1. Böcek Algoritmaları 28
3.1.1. Böcek-1 algoritması 29
3.1.2. Böcek-2 algoritması 29
3.1.3. Dist-bug algoritması 30

3.2. Potansiyel Alan Yöntemi 31
3.3. Vektör Alan Histogramı (Vector Field Histogram-VFH) 32
3.4. Kabarcık Bandı Tekniği (The Bubble Band Technique) 34
3.5. Dinamik Pencere Yaklaşımı (Dynamic Window Approach) 34
4. PIONEER 3-DX ROBOT 36
4.1. Robotun Genel Özellikleri 36
4.2. Robotun Yönetilmesi 38
4.2.1. Robotun ARCOS komutları ile direkt yönetilmesi 38
4.2.2. Robotun ARIA kütüphanesi ile yönetilmesi 45
4.3. Robot Benzetim Yazılımı 48
4.4. Robot Üstündeki Sonarların Kullanımı 50
5. ROBOT SEYİR SİSTEMİ VE TEST EDİLEN ALGORİTMALAR 57
5.1. Laboratuarda Oluşturulan Test Ortamı 59
5.2. Görüntü İşleme Teknikleriyle Ortam Haritasının Tespiti 61
5.3. Genetik Algoritmanın Yol Bulma Maksatlı Kullanımı 66
5.3.1. Başlangıç nüfusunun tespiti 66
5.3.2. Amaç fonksiyon değerinin tespit edilmesi 67
5.3.3. Optimizasyon kriterinin kontrolü 68
5.3.4. Yeni bireylerin seçilmesi 68
5.3.5. Çaprazlama 70
5.3.6. Mutasyon 71
5.3.7. Elitizm 71
5.3.8. Genetik algoritma parametrelerinin değişimi 72
5.4. Robotun Düğüm Noktaları Üzerinden Hedefe İlerlemesi 72
5.4.1. Sadece rota ve mesafe bilgisiyle tek komutla ilerleme 73
5.4.2. Dinamik olarak rota ve süratin ayarlanması 74
5.4.3. Robot hareketinin robot seyir sisteminde gerçeklenmesi 75
5.4.4. Engelden sakınma yeteneği olmadan hedefe ilerleme 77
5.5. Engelden Sakınma 80
5.5.1. Böcek algoritmaları için geliştirilen duvar takibi 80
5.5.2. Böcek-1 algoritmasının gerçeklenmesi 83
5.5.3. Böcek-2 algoritmasının gerçeklenmesi 86
5.5.4. Dist-bug algoritmasının gerçeklenmesi 89
5.5.5. Böcek1, böcek2 ve dist-bug algoritmalarının karşılaştırması 91
5.5.6. İyileştirilmiş Dist-bug algoritması 92
5.5.7. Dist-bug ve iyileştirilmiş Dist-bug karşılaştırması 96
5.6. Engelden Sakınma Yerine Sadece Yol Bulma İle Seyir Emniyeti 97
6. SONUÇLAR VE ÖNERİLER 99
6.1. Sonuçlar 99
6.2. Öneriler 101
KAYNAKLAR 102
KİŞİSEL YAYIN VE ESERLER 107
ÖZGEÇMİŞ 108

ŞEKİLLER DİZİNİ

Şekil 1.1. Endüstriyel robot örneği 8
Şekil 1.2. İnsansı robot Asimo 9
Şekil 1.3. Katır robot, Bigdog 9
Şekil 1.4. Katır robot, Bigdog bileşenleri 10
Şekil 1.5. Curiosity 10
Şekil 1.6. Hugin 11
Şekil 1.7. Predator 12
Şekil 1.8. Bir nesnenin, sonara göre konumu 14
Şekil 1.9. Sonar ses dalgalarının ses basıncına göre dağılımı 14
Şekil 1.10. Optik kodlayıcının çalışma prensibi 16
Şekil 2.1. Ortamın ızgara tabanlı haritaya aktarılması 19
Şekil 2.2. Izgara tabanlı haritada boş ve engel olan hücreler 19
Şekil 2.3. Dijkstra algoritmasının ve işlem adımlarının gösterimi 22
Şekil 2.4. A* algoritmasının ve işlem adımlarının gösterimi 23
Şekil 2.5. A* algoritmasının ızgara tabanlı bir haritada gösterimi 24
Şekil 2.6. Karıncaların engel ile karşılaşma durumları 25
Şekil 2.7. Genetik algoritmada kullanılan kromozom örneği 26
Şekil 2.8. Genetik algoritma akış şeması 27
Şekil 3.1. Böcek algoritmalarında robotun hareketi 28
Şekil 3.2. Böcek-1 algoritması 29
Şekil 3.3. Böcek 2 algoritması 30
Şekil 3.4. Dist-bug algoritması 31
Şekil 3.5. Robot ve engel arasındaki itici ve çekici vektörler 32
Şekil 3.6. Polar Histogram 33
Şekil 3.7. Robot ve önündeki engeller 33
Şekil 3.8. Robotun önündeki engellerin kutupsal histogram görüntüsü 33
Şekil 3.9. Kabarcık dizisi tekniği 34
Şekil 3.10. Dinamik pencere örneği 35
Şekil 3.11. Bir forkliftin dinamik pencere içinde engelden sakınması 35
Şekil 4.1. Pioneer 3-DX gezgin robot 36
Şekil 4.2. Pioneer 3-DX gezgin robot genel özellikleri 37
Şekil 4.3. Pioneer 3-DX gezgin robot teknik özellikleri 38
Şekil 4.4. Bir argümana ait H ve L değerlerinin tespiti 39
Şekil 4.5. Seri port üzerinden robota komut paketinin iletilmesi 40
Şekil 4.6. Seri port üzerinden haberleşme komutlar 40
Şekil 4.7. Senkronizasyon paketleri 40
Şekil 4.8. Direkt komutlar ile geliştirilen yazılım arayüzü 45
Şekil 4.9. Aria kütüphanesi yardımı ile geliştirilen yazılım arayüzü 47

Şekil 4.10. Algoritmaların denenmesi amacıyla geliştirilen yazılım arayüzü 48
Şekil 4.11. MobileSim benzetim ortamında robotun hareket izleri 49
Şekil 4.12. MapperBasic ile harita ve engellerin oluşturulması 50
Şekil 4.13. Pioneer 3-DX gezgin robot sonar dizisi 51
Şekil 4.14. Pioneer 3-Dx üzerindeki sonarlar ve numaraları 52
Şekil 4.15. Benzetim ortamında uygulamanın elde ettiği sonar verileri. 52
Şekil 4.16. MobileSim benzetim programı ekran görüntüsü 53
Şekil 4.17. MATLAB'da timer oluşturma 53
Şekil 4.18. Yazılım üzerinde sonar bilgilerinin gösterimi 54
Şekil 4.19. Sol taraftaki 0 numaralı sonarın gerçek engelleri algılama testi 55
Şekil 4.20. Öndeki 3 ve 4 numaralı sonarların gerçek engelleri algılama testi 55
Şekil 4.21. Gerçek engel olarak kutuların kullanılması 56
Şekil 5.1. Tasarlanan robot seyir sistemi yapısı 58
Şekil 5.2. Geliştirilmiş arayüze ait ekran görüntüsü 59
Şekil 5.3. Laboratuardaki test ortamı 60
Şekil 5.4. Laboratuarda engeller 61
Şekil 5.5. Tavan kamerası ile alınan zemin fotoğrafı 62
Şekil 5.6. Görüntü işleme ile ilgili kullanılan MATLAB komutları 62
Şekil 5.7. Temizlenmiş zemin fotoğrafı 63
Şekil 5.8. Fotoğraftaki diğer bilgilerin süzülmüş hali 64
Şekil 5.9. Çalışmada kullanılan kırmızı ve sarı renklerin RGB değerleri 64
Şekil 5.10. Fotoğraftaki engel ve robotun piksellerinin tespit edilmesi 65
Şekil 5.11. Elde edilen 10x8 lik matrisin uygulama üzerinde gösterimi 65
Şekil 5.12. Çalışma alanı 66
Şekil 5.13. Tam sayı kromozom örneği 67
Şekil 5.14. Başlangıç nüfusunun doldurulması 67
Şekil 5.15. Seçilme aralıklarının rulet tekerleği şekline gösterimi 70
Şekil 5.16. Çaprazlama ile ebeveynlerden yeni çocuk bireylerin üretilmesi 71
Şekil 5.17. Genetik algoritma parametre giriş ekranı 72
Şekil 5.18. Sadece rota ve mesafe bilgisi ile ilerleme algoritması 73
Şekil 5.19. Dinamik rota ve sürat ayarlaması ile ilerleme algoritması 74
Şekil 5.20. Robot seyir sistemi ile oluşturulan yol güzergahı 75
Şekil 5.21. MobileSim ve robot seyir sistemi üzerinde düğüm noktaları 75
Şekil 5.22. MobileSim üzerinde robotun ilerleyişi 76
Şekil 5.23. Robot seyir sisteminde robotun ilerleyişi 77
Şekil 5.24. Senaryo kapsamında hazırlanan ilk ortam 78
Şekil 5.25. Senaryo kapsamında hazırlanan ikinci ortam 78
Şekil 5.26. Sonradan eklenen engel ile robotun durumu 79
Şekil 5.27. Engel ile karşılaşan robotun durumu 81
Şekil 5.28. Eğim hesaplı duvar takip algoritması yok iken robot 82
Şekil 5.29. Eğim hesaplı duvar takip algoritması ile robot 82
Şekil 5.30. Böcek 1 algoritması 83
Şekil 5.31. Böcek 1 ayrılış noktasına kadar duvar takip yapma algoritması 84
Şekil 5.32. MobileSim'de duvar takibi yapan Böcek-1 algoritması 85
Şekil 5.33. Robot seyir sisteminde duvar takibi yapan Böcek-1 algoritması 85
Şekil 5.34. Robot seyir sisteminde duvar takibi yapan Böcek-2 algoritması 86
Şekil 5.35. Böcek 2 ayrılış noktasına kadar duvar takip yapma algoritması 87
Şekil 5.36. MobileSim'de duvar takibi yapan Böcek-2 algoritması 88
Şekil 5.37. Robot seyir sisteminde duvar takibi yapan Böcek-2 algoritması 88

Şekil 5.38. Dist-bug ayrılış noktasına kadar duvar takip yapma algoritması 89
Şekil 5.39. MobileSim'de duvar takibi yapan Dist-bug algoritması 90
Şekil 5.40. Robot seyir sisteminde duvar takibi yapan Dist-bug algoritması 91
Şekil 5.41. Düğüm atlama özellikli Dist-bug algoritması 93
Şekil 5.42. Ayrılış noktasına kadar duvar takip yapma algoritması 94
Şekil 5.43. Ulaşılabilir sonraki düğümü hesapla fonksiyonu 95
Şekil 5.44. MobileSim'de iyileştirilmiş Dist-bug algoritması..... 95
Şekil 5.45. Robot seyir sisteminde iyileştirilmiş Dist-bug algoritması 96
Şekil 5.46. Periyodik yol bulma maksatlı benzetim yönetim ekranı 97

TABLOLAR DİZİNİ

Tablo 4.1. Komut paketi bileşenleri 39
Tablo 4.2. Sunucu bilgi paketi bileşenleri 41
Tablo 4.3. Sunucu bilgi paketi çözümlemesi örnek 1 42
Tablo 4.4. Sunucu bilgi paketi çözümlemesi örnek 2 43
Tablo 4.5. Sunucu bilgi paketi çözümlemesi örnek 3 44
Tablo 4.6. Direkt komut yöntemi ve Aria komutlarının karşılaştırması 46
Tablo 5.1. Rulet tekerleği seçilme olasılıkları 69
Tablo 5.2. Böcek algoritmalarında elde edilen ölçüm değerleri 91
Tablo 5.3. Dist-bug ve iyileştirilmiş Dist-bug algoritmalarının karşılaştırması 96
Tablo 5.4. Yol bulma işlemine ait süreler 98

SİMGELER VE KISALTMALAR DİZİNİ

mm. : Milimetre
ms. : Milisaniye
V. : Volt

Kısaltmalar

ARCOS : Advanced Robot Control and Operations Software (Gelişmiş Robot Kontrol ve İşlem Yazılımı)
ARIA : Advanced Robotics Interface for Applications (Uygulamalar İçin Gelişmiş Robotik Arayüzü)
CCD : Charged Couple Device (Yükten Bağlaşımlı Aygıt)
CMOS : Complementary Metal Oxide Semiconductor (Tamamlayıcı MetalOksit Yarıiletken)
GA : Genetik Algoritma
GPL : General Public Licence (Genel Kamu Lisansı)
MEX : Matlab Executable (Matlab Çalıştırabilir)
SIP : Server Information Packet (Sunucu Bilgi Paketi)
VFH : Vector Field Histogram (Vektör Alan Histogramı)

SONAR ALGILAYICILAR VE SEZGİSEL YÖNTEMLER KULLANARAK OTONOM ROBOTLARIN ENGELDEN SAKINIMI

ÖZET

Otonom robotlar günümüzde birçok alanda önemli görevler icra etmektedirler. Robot, kendisine verilen bir görevi yerine getirirken, hedefe ilerlemek için yol bulma problemini çözmeli ayrıca kendi bekasını korumak için, algılayıcıları yardımıyla çevreden elde ettiği verileri kullanarak, varsa olası bir çarpışmayı önlemek için statik veya dinamik olarak engellerden sakınma yapabilmelidir.

Bu tez çalışmasında, sonar algılayıcılar ve sezgisel yöntemler kullanarak, otonom robotların engelden sakınma probleminin çok yönlü optimizasyonunun yapılması sağlanmıştır. Bu kapsamda robotların engelden sakınma ve yol bulma probleminin çözümüne ilişkin bir otonom robot seyir sistemi geliştirilerek, yeni engelden sakınma algoritmalarının bu sistem üzerinde gerçeklenmesi sağlanmıştır.

Bu kapsamda, laboratuar ortamında bir tavan kamerasından alınan resim bilgisi, görüntü işleme teknikleri kullanarak engelleri de içerecek şekilde bir harita haline getirilmiştir. Hibrit bir algoritma olarak tasarlanan seyir sistemi içinde, genetik algoritma yardımıyla global yol bulma problemi çözülmüştür. Dinamik engellerden sakınmak amacıyla da çeşitli engelden sakınma yöntemleri uygulanmıştır. Çalışmalar, öncelikle MobileSim benzetim ortamında denenmiş ve ardından Pioneer 3-DX robot üzerinde ayrıca gerçeklenmiştir.

Ayrıca, engelle karşılaşıldığında engelin etrafından dolanan böcek algoritmaları için, duvar eğimini hesaplayarak daha performanslı bir duvar takip yöntemi kullanılmıştır. Engelden sakınma için yaygın olarak kullanılan Dist-bug algoritması iyileştirilerek, düğüm atlatma özellikli iyileştirilmiş Dist-bug algoritması geliştirilmiştir. Geliştirilen algoritmalar test edilerek yol ve zaman karşılaştırmaları yapılmıştır.

Anahtar Kelimeler: Engelden Sakınma, Genetik Algoritma, Otonom Robot, Sonar, Yol Bulma.

OBSTACLE AVOIDANCE OF AUTONOMOUS ROBOTS BY USING SONAR SENSORS AND HEURISTIC METHODS

ABSTRACT

Today autonomous robots are performing important tasks in various fields. While performing a task, the robot should solve the path planning problem to reach the target point, additionally to protect the survival of itself, it should practice dynamic or static obstacle avoidance activity for avoiding a possible collision by using the data acquired by its sensors from the environment.

In this study, by using sonar sensors and heuristic methods, the obstacle avoidance problem of autonomous robots were addressed and optimized in multiple areas. As a solution to obstacle avoidance and path finding problem of robots, an autonomous robot navigation system was created, and the improved obstacle avoidance algorithms were implemented on this system.

In this context, the picture of the environment taken from a camera on the ceiling of the laboratory was transformed into a map of the environment including the obstacles by using image processing techniques. In the navigation system, designed as a hybrid system including both path finding and the obstacle avoidance modules, the path finding problem was solved with genetic algorithm. For the avoidance from the dynamic obstacles, multiple obstacle avoidance methods were implemented. The algorithms first simulated on MobileSim simulation environment and later implemented on Pioneer3-DX mobile robot.

In the study, for the wall following behavior of the bug algorithms, by calculating the slope of the obstacle edge, a wall follow algorithm with a better performance was used. The dist-bug algorithm, a widely used algorithm for obstacle avoidance, was also improved and node skipping improved dist-bug was developed. By testing the developed algorithms, path-time comparisons were made.

Keywords: Obstacle Avoidance, Genetic Algorithm, Autonomous Robots, Sonar, Path Finding.

GİRİŞ

Otonom robotlar, birçok alanda önemli görevler icra etmektedirler. Görev alanları askeri alandan bilimsel alanlara, endüstriden günlük yaşama çok geniş bir yelpazeyi kapsamaktadır. Otonom olma özelliği, bir robotun karşısına çıkabilecek önceden bilinen ya da beklenmedik durumlar karşısında, insan yardımı olmadan kendi kendine karar verebilme yeteneğine sahip olmasıdır. Mars'ta araştırmalar yapmakta olan Curiosity [1] ve kalabalık fuarlarda bağımsız olarak rehberlik yapan RoboX [2] ilk akla gelebilecek kullanımdaki otonom robot örnekleridir.

Robot, kendisine verilen bir görevi yerine getirirken, hedefe ilerlemek için yol bulma problemini çözmeli ayrıca kendi bekasını korumak ve çevreye zarar vermemek için, algılayıcıları yardımıyla çevreden elde ettiği verileri kullanarak, varsa olası bir çarpışmayı önlemek için statik veya dinamik olarak engellerden sakınma yapabilmelidir.

Robotlarda yol bulma problemi, iki nokta arasında en kısa ve en az maliyetle gidilebilen yolun belirlenmesi olarak açıklanabilir. Bu konuda yapılan çalışmalar sonucunda, Dijkstra [3], A* (A-yıldız) [4], D* (D-yıldız) [5] ve potansiyel alan metodu [6] gibi çeşitli yol bulma amaçlı algoritma ve yöntemler geliştirilmiş, ayrıca optimizasyon amaçlı geliştirilmiş genetik algoritma [7] gibi algoritmalar da yine bu amaçla kullanılmışlardır. Bu algoritmalar karşılaştırıldıklarında, her bir metodun kendine özgü avantajları olduğu gibi, dezavantajlarının da bulunduğu görülebilmektedir. Bu nedenle, probleme yönelik en uygun algoritmanın seçilmesi oldukça önemlidir.

Engelden sakınma veya çarpışma önleme, otonom robotun güzergahı üzerindeki öngörülebilen veya görülemeyen engellerden sakınma amacıyla yapılan bir faaliyetler dizisidir. Bu amaçla, çeşitli engelden sakınma algoritmaları geliştirilmiştir.

Böcek algoritmaları [8,9], potansiyel alanlar metodu [10], vektör alan histogramı [11], kabarcık dizisi yöntemi [12], eğrilme hızı yaklaşımı [13], dinamik pencere yaklaşımı [14] engelden sakınma amaçlı kullanılan algoritmaların başlıcalarını oluşturmaktadır.

Otonom bir robot çevresindeki bilgileri algılayıcıları yardımı ile almaktadır. Robotlarda çok çeşitli algılayıcı tipleri bulunmakla birlikte, sonar algılayıcılar ekonomik olmaları ve pratik kullanımları sayesinde otonom robotlarda oldukça yaygın kullanım yerine sahiptirler.

Literatürde engelden sakınma ve yol bulma problemi ile ilgili bir çok çalışma bulunmaktadır. Bu çalışmaların büyük bir bölümü tez çalışması kapsamında incelenmiş, bunların bir kısmı bu bölümde özetlenmiştir.

Philippsen ve Siegwart 2003 yılında, bir robot tur rehberi tasarlamışlar ve kalabalık bir fuarda fiilen denemişlerdir [15]. Bu çalışmada yol bulma problemi için potansiyel alan metodunun bir çeşidi kullanılırken, engelden sakınma için elastik band ve dinamik pencere yaklaşımı kullanılmıştır. Geliştirilen robotun, insanların yoğun bir şekilde bulundukları ortamlarda başarı ile görevini yerine getirdiği görülmüştür

Savage ve arkadaşları tarafından 2013 yılında genetik algoritma, yapay sinir ağları ve potansiyel alan metotlarının bir bileşkesi kullanılarak, engelden sakınma problemi çözülmeye çalışılmıştır [16]. Genetik algoritma yardımı ile ağırlıkları hesaplanan yapay sinir ağları ile potansiyel alan metodu engelden sakınma amacıyla kullanılmıştır.

Stone tarafından 2007 yılında yapılan çalışmada oyuncu konumundaki robot köpekler, sonar algılayıcılarının yanı sıra kamera ve lazer mesafe bulucu algılayıcılarının da yardımı ile futbol maçında kullanılmış, robotların hedefe ilerlemesi esnasında, engellerden sakınmak için potansiyel alan yöntemi kullanılmıştır [17].

Li tarafından 2007 de yapılan bir çalışmada, mobil robotlarda engelden sakınma üzerine çalışılmış, makine öğrenmesi ve bu kapsamda çeşitli öğrenebilen sınıflandırma sistemleri incelenmiştir [18].

Zhu ve arkadaşları tarafından 2012 yılında, hibrid bir navigasyon algoritması önerilmiştir. Çalışmada, A* algoritması ile yol bulma problemi çözülmüş, ayrıca engelden sakınma için böcek ve potansiyel alan metotları denenmiştir [19].

Li ve Choi tarafından 2013 yılında bulanık mantık tabanlı bir engelden sakınma sistemi önerilmiştir. Çalışmada, sonar algılayıcılardan elde edilen bilgilerin bulanık mantık kullanılarak, tekerlerin ivmesi ve robotun gidiş açısının değiştirilmesinde kullanımı denenmiştir [20].

Bulanık mantığın farklı bir kullanımı olarak, Faisal ve arkadaşları iki bulanık mantık sistemi kullanmayı önermiştir. İlki, yol takip eden bulanık mantık kontrolörü (tracking fuzzy logic controller) olarak hedefe ilerlemeyi sağlarken; ikincisi, engelden sakınma bulanık mantık kontrolörü (obstacles avoiding fuzzy logic controller) olarak engelden sakınma yapması planlanmıştır [21].

Engelden sakınma sadece otonom robotun kendi algılayıcıları ile de yapılmayabilinir. 2002'de yapılan bir çalışmada, robotların bir laboratuar ortamı veya ev gibi çeşitli algılayıcılarla donatılmış bir ortam içinde hareketlerinin izlenmesi hedeflenmiştir. Akıllı ortam, kendi içinde gezinen varlıkların durumunu algılayıcıları aracılığı ile takip edebilmekte, engeller konusunda onları bilgilendirmekte ve gerektiğinde yönlendirmektedir [22].

Bununla birlikte, engelden sakınma ve sonarlar, sadece otonom robotlar için değil, farklı alanlarda da önemli çalışmalara konu olmuşlardır. Bell ve arkadaşları tekerlekli engelli sandalyesi için sonar ile engelden sakınma üzerine çalışmışlardır. Bu çalışmalarında vektör alan histogramı yöntemini kullanmışlar, çalışma sonucunda tekerlekli sandalyeyi kullanan görme engelli bir kişinin emniyetli bir şekilde yol alması sağlanmıştır [23].

Benzer şekilde, Shraga ve arkadaşları sonarın robotlardaki engelden sakınma işlevini, giyilebilir teknoloji olarak görme engellilerde denemişlerdir. Vektör alan histogramı yönteminin kullanıldığı otonom robotlar için uygulanan çalışma, görme engelliler için hazırlanan bir sonar kemeri yardımı ile görme engellilerin engellerden sakınarak yollarını bulmaları sağlanmıştır [24].

Engelden sakınma problemi, aynı zamanda, su altında da önemli bir konu olarak karşımıza çıkmaktadır. Özellikle sonarların, su altındaki bu çok yaygın kullanım alanı ile ilgili olarak literatürde birçok çalışma mevcuttur. Bu çalışmaların bir kısmına bakacak olursak; Conte ve Zanoli, bir sualtı otonom robotunun sonarlar yardımı ile engelden sakınmasında tek engel ve çok engel konularını ele almışlardır. Özellikle engellerin hareketli olması durumunda, Kalman filtresinden faydalanmışlar, çok engel durumunda, robotun önce en yakındaki engel ile ilgilenmesini hedeflemişlerdir [25].

Lee ve arkadaşları [26] bulanık mantıkla, sualtı otonom araçlarında engelden sakınma üzerinde çalışmışlar, benzer şekilde, Schultz [27] genetik algoritma ile sualtı otonom açlarının engelden sakınma ve yol bulma problemini çözmüştür.

Bu tez çalışmasında; sonar algılayıcılar ve genetik algoritmalar kullanarak, otonom robotların engelden sakınma probleminin çok yönlü optimizasyonunun yapılması amaçlanmıştır. Bu kapsamda yapılan çalışmalar üç ana başlık altında toplanabilir. İlk olarak, robotun bir hedefe ilerleyebilmesi için ortam haritasının üretimi sağlanmış ve bu harita üzerinden genetik algoritmalar yardımı ile yol bulma probleminin çözümü sağlanmıştır. Bunun için yol bulma algoritmaları incelenmiş ve harita üretme sıklıkları belirlenmiştir. İkinci olarak, engelden sakınma için kullanılan algoritmalar incelenmiş, engelden sakınma konusunda daha etkin bir algoritma geliştirilmiştir. İlk iki aşamada kullanılan algoritmalar benzetim ortamında test edilmiş, üçüncü ve son aşamada; geliştirilen algoritmalar laboratuar ortamında Pioneer 3-Dx robot üzerinde gerçeklenmiştir.

Tez çalışması beş ana bölümden oluşmaktadır:

Birinci bölümde; otonom robotlar hakkında bilgi verilmekte, ayrıca robotlarda kullanılan algılayıcılardan bahsedilmektedir.

İkinci bölümde; engelden sakınma problemi hakkında bilgi verilmekte, bu kapsamda kullanılan algoritmalar ele alınmakta ve tez çalışması kapsamında geliştirilen algoritma açıklanmaktadır.

Üçüncü bölümde; yol bulma algoritmaları incelenerek, genetik algoritmalar ile yol bulma probleminin çözümü hakkında bilgi verilmektedir.

Dördüncü bölümde; tez çalışmasında kullanılan Pioneer 3-DX gezgin robot hakkında bilgi verilmektedir.

Beşinci bölümde; geliştirilen algoritmaların test edilmesi amacıyla hazırlanan robot seyir sistemi hakkında bilgi verilmekte, bu kapsamda geliştirilen yazılım, laboratuarda hazırlanan test ortamı ve testlerden elde edilen sonuçlar anlatılmaktadır.

Altıncı bölümde; tez çalışmasından elde edilen sonuçlar ve öneriler sunulmaktadır.

1. OTONOM ROBOTLAR

1.1. Robotun Tanımı ve Tarihçesi

Günümüzde, tüm dünyada yaygın olarak kullanılan Robot kelimesi, ilk olarak 1921 yılında Çek oyun yazarı Karel Capek'in "Rossum's Universal Robots " adlı oyununda kullanılmıştır [28]. Yazar zorunlu iş anlamındaki "robota" kelimesinden "robot" kelimesini türetmiş, kendi kendilerine çalışabilen işçileri tanımlamıştır [29].

Robotun, günümüzdeki tanımına bakacak olursak, Webster sözlüğünde "bir bilgisayar yardımı ile kontrol edilebilen veya otomatik olarak çalışan, insanların gerçekleştirdikleri işleri yerine getiren araçlar" olarak tanımlandığını, Oxford sözlüğü tarafından ise, "karmaşık bir dizi işlevi otomatik olarak icra edebilen makine" olarak tanımlandığını görmekteyiz.

Kendi kendine çalışabilen makine kavramı elbette çok daha eskilere dayanmaktadır. Bu makinelerin benzerleri eski yunan yazıtlarında bahsedilmiş, 13 üncü yüzyılda El-Cezeri'nin "Maharetli Mekanik Cihazların Bilgisini İçeren Kitap" eserinde tarif edilmiş [30], ve 15 nci yüzyılda Leonardo da Vinci tarafından tasarlanmışlardır [31].

İlk programlanabilen makine ile ilgili patent 1954 yılında George Devol tarafından alınmıştır. Programlanabilir özelliği nedeni ile ilk endüstriyel robotun temellerini oluşturan bu fikir, kısa süre sonra yine Devol'un ortağı olduğu Unimation şirketi tarafından ilk robotun (Unimate) 1960 yılında kullanıma sunulmasını sağlamıştır [31]. Bu robotun asli görevi, General Motors'un araba kalıp-döküm fabrikasında, sıcak araba parçalarının, soğutma sıvısına sokulmasıydı. Böylece, fabrika çalışanlarının sıcak metalle teması minimuma indirilmiş oluyordu.

Unimation tarafından kullanıma sunulan bu robotta olduğu gibi, robotlar insanların yapmayı istemedikleri alanlarda görev almaktadırlar. Bu görevleri; insanlar için riskli olan görevler (uzay, nükleer ve askeri gibi), insan gücünün kullanımının verimsiz olabileceği yerler (endüstri, tarım) ve insan hayatı için tehlikeli görevler (mayınların temizlenmesi gibi) olarak sınıflandırabiliriz. Bu tür görevler ayrıca, İngilizce'de 3-Ds

(Dull, Dirty and Dangerous) yani sıkıcı, kirli ve tehlikeli görevler olarak da adlandırılmaktadırlar [32]. Bununla birlikte, alan sınırlaması olmaksızın, robotlar ihtiyaç duyulabilecek her türlü alanda bizlere yardımcı olmaktadırlar.

1.2. Robotların Çeşitleri

Robotların kullanım yerlerine ve amaçlarına göre çok çeşitli sınıflandırmaları bulunmaktadır. Honda tarafından geliştirilmiş dans eden insansı robot Asimo ile araba fabrikasında boya yapan robot kolunun arasındaki ilişkiyi bu aşamada anlamak faydalı olacaktır.

1.2.1. Endüstiyel robotlar/manipulatör/kollar

Uluslararası Standardizasyon Örgütü'nün ISO 8373 standardındaki robot tanımı, bize endüstriyel robotları tasvir etmektedir. Bu tanıma göre, bir robot, endüstri uygulamalarında kullanılan, otomatik kontrollü, tekrar programlanabilen, çok amaçlı, üç veya dört eksene sahip, sabit veya hareketli manipülatörlerdir.

İlk programlanabilen robot Unimate de olduğu gibi, bu robotlar günümüzde endüstride çok yaygın olarak kullanılmaktadır. Uluslararası Robot Federasyonu'nun 2015 yılında yaptığı bir çalışmada, kullanımdaki çok amaçlı endüstriyel robotların sayısının önümüzdeki üç yılda, 2 milyonu geçmesinin tahmin edildiği, 2014 satışları baz alındığında, sadece endüstri robotları satışının 10 milyar dolarlık bir pazarı oluşturduğu belirtilmektedir [33]. Şekil 1.1'de yaygın olarak kullanılan, çok amaçlı endüstriyel bir robot görülmektedir.

Otomotiv sektörü, üretimde endüstriyel robot kullanımında açık ara lider durumda bulunmaktadır.



Şekil 1.1. Endüstriyel robot örneği [34]

Endüstriyel robotların kolları onların en işlevsel bölümleridir. Bu kol sayesinde; boyama, delme, kesme, şekil verme benzeri her türlü işlevi yapabilmektedirler. Onları ağır sanayideki büyük kalıp makinelerinden ayıran tarafları, yeniden programlanmak suretiyle, farklı maksatlarla farklı görevlerde kullanılabilmeleridir.

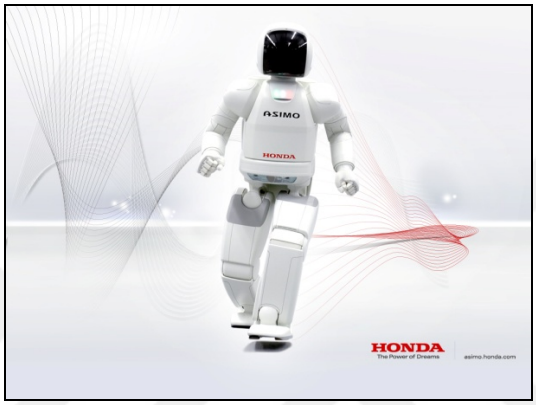
1.2.2. Mobil robotlar

Endüstriyel robotlar ve manipülatörler bir kaide üzerine monteli olarak, kısıtlı bir yer değiştirme imkanı ile çalışırlar. Ancak mobil robotlar, kullanım ihtiyacına göre fabrika içinde veya dünya üzerinde hareket ederek, yeteneklerini en iyi nerde gösterebileceklerse o yere gidebilecek imkana sahiptirler. Çölde aşırı sıcakların altında, suyun yüzlerce metre altında veya Mars gezegeni yüzeyindeki bir görev mobil robotlara olan ihtiyacı açıklamak için yeterlidir.

Robotlara mobil hareket yeteneği farklı şekillerde sağlanabilir. Bunların bir kısmı bu bölümde ele alınmaya çalışılmıştır.

1.2.2.1. Ayaklı robotlar

Doğada hayvanlarda yaygın olarak görülen bir hareket yöntemi olan ayaklar, yeryüzünde canlıların hareketi için büyük kolaylık sağlamıştır. İki ayaklı insanlardan, yüzlerce ayağa sahip kırkayaklara kadar farklı yapıları vardır. Ancak, canlıların gerek kas yapısı gerekse enerji sistemleri açısından efektif olan bu yöntem, robotlarda birçok sınırlamayı beraberinde getirmiştir. Mekanik ve güç karmaşıklığı nedeniyle robotlarda genelde az sayıda ayak tercih edilmiştir.

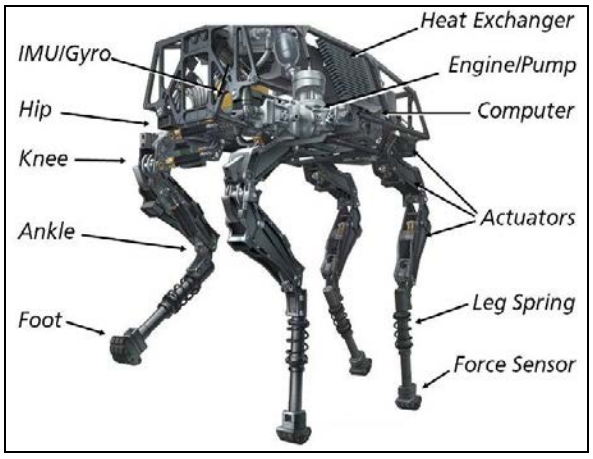


Şekil 1.2. İnsansı robot Asimo [35]

Mekanik ve güç karmaşıklığı nedeniyle robotlarda genelde az sayıda ayak tercih edilmiştir.



Şekil 1.3. Katır robot, Bigdog [36]



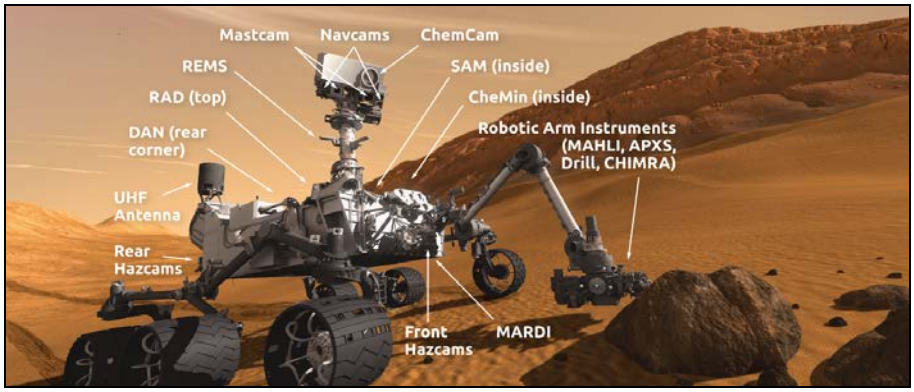
Şekil 1.4. Katır robot, Bigdog bileşenleri [36]

Japon Honda firmasının insansı robotu Asimo (Şekil 1.2) ile Amerikan Boston Dynamics firması tarafından geliştirilen katır robot BigDog (Şekil 1.3 ve 1.4), ayaklı robotların en bilinen örnekleridir.

Tekerleklerin düz bir satıh ihtiyacının bulunmasına rağmen, ayaklı robotların her türlü satıhta hareket kabiliyetlerinin olması, ayaklı robotların en önemli avantajını oluşturmaktadır.

1.2.2.2. Tekerlekli robotlar

Tekerleğin icadından itibaren, hareket ve ulaşım deyince akla ilk gelen teknoloji olan tekerlekler, mobil robotlarda da çok yaygın bir şekilde kullanılmaktadır. Efektif oldukları kadar, ayaklı robotlarda ihtiyaç duyulan karmaşık mekanizmalara bunlarda ihtiyaç duyulmamaktadır.

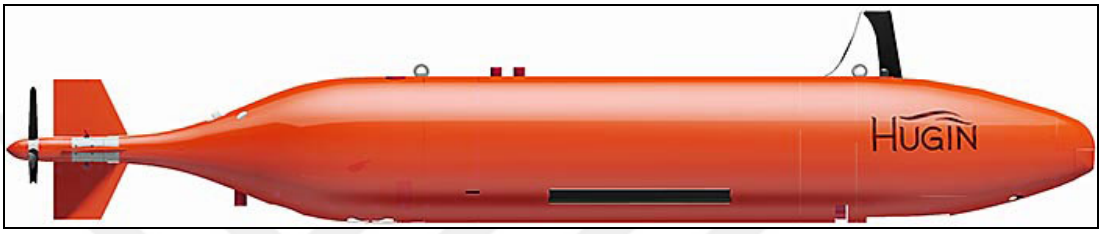


Şekil 1.5. Curiosity [37]

2012'de Mars yüzeyinde inceleme yapmak üzere gönderilen Curiosity (Şekil 1.5), başarı ile görevine devam etmekte olan bir robot olarak örnek verilebilir. Tekerlekli robotların çok yaygın kullanım alanları bulunmaktadır.

1.2.2.3. Otonom denizaltı araçları

Otonom denizaltı araçları, deniz araştırmaları amacıyla kullanılan, tasarımına göre; yüzebilen, dalabilen, akıntı ile sürüklenebilen, minimum operatör girdisi isteyen ve programlanabilir robotik araçlardır.



Şekil 1.6. Hugin [38]

Diğer robot türlerinde olduğu gibi, sivil ve askeri amaçlı kullanılabilen bu araçlar (Şekil 1.6), deniz içinde ve üstünde büyük hareket kabiliyetine sahiptirler.

1.2.2.4. İnsansız hava araçları

İnsansız hava araçları, uzaktan kumanda edilerek veya kendiliğinden belli bir uçuş planı üzerinden otomatik olarak hareket edebilen uçaklardır. Sivil kullanım alanları olsa da, yaygın olarak askeri alanda, keşif/gözetleme ve saldırı amaçlı kullanılmaktadırlar.

Çeşitli silahlı kuvvetler tarafından kullanılan, General Atomics firmasınca üretilen Predator (Şekil 1.7), saldırı amaçlı da kullanılabilen insansız hava araçlarının en bilinenlerinden biridir.



Şekil 1.7. Predator [39]

1.3. Robotların Sahip Olması Gereken Üç Yetenek

Bir robotun kendisine verilen görevi yerine getirebilmesi için aşağıdaki üç yeteneğin bulunması gereklidir:

1.Algılama: Robot alıcıları aracılığı ile gerek dış ortam ile ilgili gerekse kendi ile ilgili ortam bilgilerini algılayabilmelidir.

2.Planlama: Robot algılayıcılarından gelen bilgileri değerlendirebilmeli, yapacağı işi planlayabilmelidir.

3.Eylem: Robot kendisine verilen görevi, eyleyiciler aracılığı ile yerine getirmelidir.

Literatürde yaygın olarak "sense-plan-act" olarak da adlandırılan, bu üç özelliğin kullanımı hiyerarşik, reaktif veya hibrid şekillerde olabilmektedir. Hiyerarşik yapıda, bir sıra halinde, algılama, planlama ve eylem üçlüsü arka arkaya icra edilirken, reaktif yapıda, algılamaya karşılık direk eyleme geçilmekte, hibrid yapıda ise bu iki yapının birleşik şekli kullanılmaktadır [32].

Aşağıdaki altbölümlerde, bu üç temel özelliğin yerine getirilmesini sağlayan, algılayıcılar, kontrol bilgisayarı ve eyleyiciler hakkında özet bilgi verilecektir.

1.3.1. Algılayıcılar

Bir insanın sahip olduğu; gözler, kulaklar, burun, dil ve dokunma olarak adlandırdığımız beş duyu organı gibi, robotların da ortam bilgilerini almak için kullandıkları algılayıcıları bulunmaktadır.

Robotlarda algılayıcılar, robotun ilerleme hızı, pil durumu gibi bilgileri veren iç algılayıcılar ve etraftaki engeller veya ortam sıcaklığı gibi dış ortam hakkında bilgiler veren dış algılayıcılar olarak iki ayrı grupta sınıflandırılabilmektedir.

Bununla birlikte, algılayıcının dış ortama sinyal gönderme durumuna göre, aktif ve pasif olarak da sınıflandırılmaktadır. Sadece gözlem yaparak, çevredeki kaynaklardan gelen sinyalleri okuyan algılayıcılar pasif algılayıcı, çevreye ayrıca bir sinyal göndererek, bu sayede kaynaktan gelen sinyali algılayan algılayıcılara aktif algılayıcı denilmektedir. Işık, sıcaklık ve görüntü algılayıcıları genelde pasif algılayıcılar iken, ses sinyali göndermek suretiyle mesafe bulan ultrasonik algılayıcılar aktif algılayıcılardandır.

Günümüzde, pusula gibi manyetik algılayıcılar, ivmeölçerler, basınçölçerler, sıcaklık ölçerler, nemölçerler, ışık algılayıcıları, eğimölçerler, dönüş açısı ölçerler, koku algılayıcıları, ultrasonik algılayıcılar, jiroskoplar ve GPS gibi çok çeşitli algılayıcılar bulunmakta, ayrıca göreve göre ve farklı tip robotlarda, farklı algılayıcılara ihtiyaç duyulabilmektedir.

Tez kapsamında özellikle otonom mobil robotların ele alınması sebebiyle, bu bölümde mobil robotlarda yaygın olarak kullanılan bazı algılayıcılar ile ilgili ilave bilgi verilecektir.

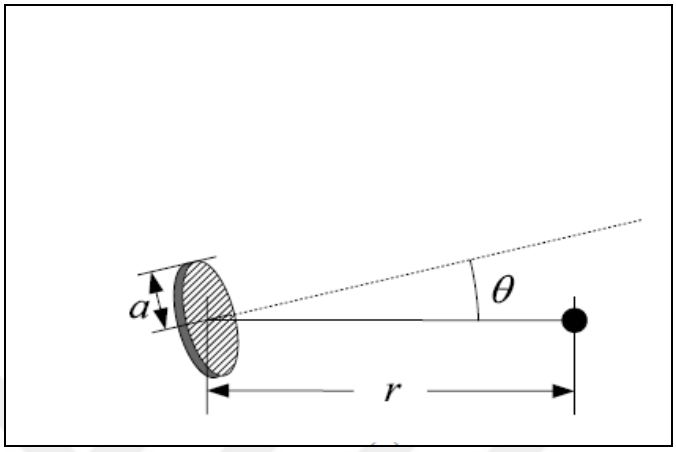
1.3.1.1. Ultrasonik mesafe bulucular

Ultrasonik mesafe bulucular sesin yayılması ve geri yansıması prensibi ile çalışan aktif algılayıcılardandır. Algılayıcı, bir ses dalgası paketi gönderir ve paketin geri gelişine kadar olan süre hesaplanır. Sesin hızı bilindiğinden, mesafe basit bir şekilde tespit edilebilir.

Literatürde sıklıkla Sonar olarak tanımlanan ve doğada yarasalar ve yunuslar tarafından da kullanılmakta olan bu yöntem, mobil robotlarda da sıkça tercih edilen bir algılayıcıdır. Sonar adını “Sound Navigation and Ranging” kelimelerinin baş harflerinden almaktadır.

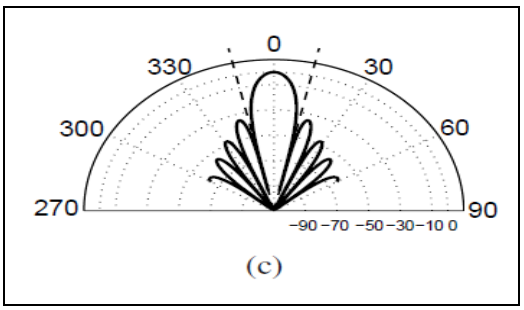
Mobil robotlarda, robot üzerindeki sonarların kullanımında öncelikle, engel ile sonar vericisinin aynı düzlemde olduğu varsayılmaktadır [18]. Bu varsayımla sonarın dikey

saçıntıları dikkate alınmamakta ayrıca sonarın bulunduğu yükseklikten daha aşağı ve daha yukarı engeller sonar tarafından algılanmamaktadır. Bu varsayımlar çerçevesinde bir nesnenin sonardan konumu r mesafesi ve θ azimut açısı olarak Şekil 1.8'deki gibi gösterilebilir.



Şekil 1.8. Bir nesnenin, sonara göre konumu [39]

Sonarda, ses dalgalarının transduserin merkezinden itibaren doğrusal bir şekilde yayımlandığı varsayılmış ve merkez noktada ses basıncının en yüksek, kenarlarda ise daha az olduğu tespit edilmiştir. Şekil 1.9'da sonar ses dalgalarının ses basıncına göre dağılımı görülmektedir.



Şekil 1.9. Sonar ses dalgalarının ses basıncına göre dağılımı [39]

Tez çalışması kapsamında sonarlar yoğun bir şekilde kullanılmıştır. Bölüm 3'te, tez çalışmasında kullanılan robot (Pioneer 3-DX) özellikleri anlatılırken, sonarların kullanımı hakkında daha detaylı bilgi verilecektir.

1.3.1.2. Lazer mesafe bulucular

Aktif algılayıcılardan biri olan lazer mesafe bulucular, sonar ile benzer bir çalışma prensibine sahiptir. Burada ses sinyali yerine hedefe lazer emisyonu gönderilmekte ve yansıyan emisyon tekrar optik algılayıcı tarafından alınmaktadır. Lazer emisyonu tek bir vuruş veya sürekli dalga olarak gönderilebilmektedir.

Sonarlara nazaran, arka plan gürültüsünden daha az etkilenmekte ve bu açıdan 2 ve 3 boyutlu çevre haritalamasında sıklıkla tercih edilmektedir.

Diğer algılayıcılara göre daha pahalı olmaları, sisli ve karlı havalarda kullanılamaması bazı dezavantajlarıdır [40].

1.3.1.3. Kızılötesi (Infrared) yakınlık algılayıcıları

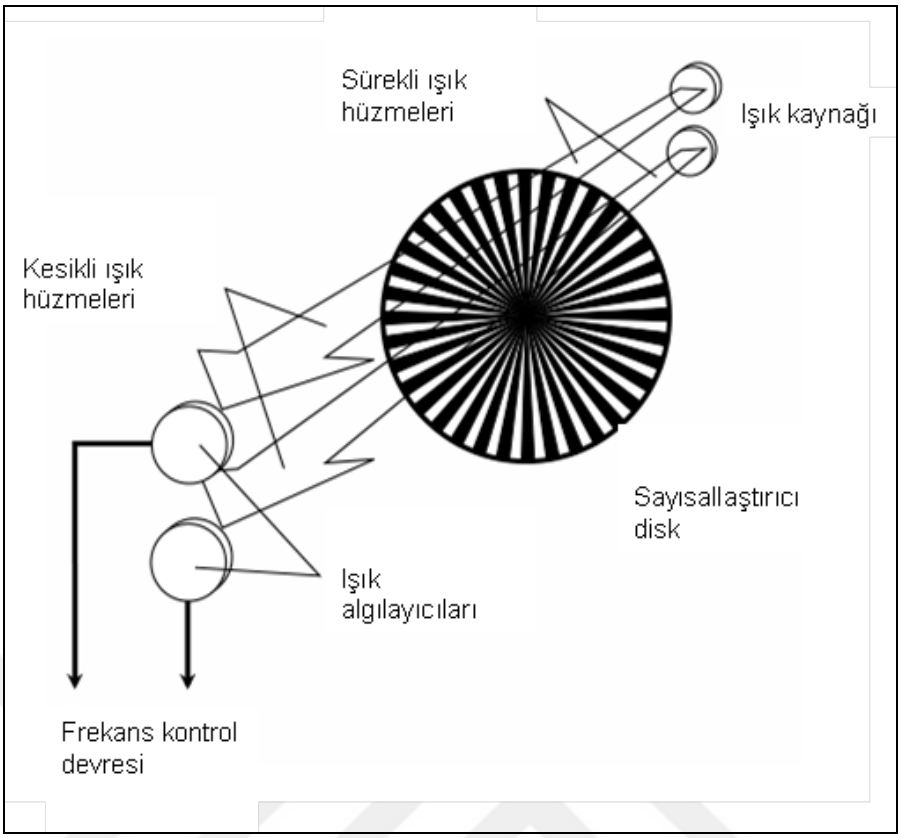
Diğer bir aktif algılayıcı olan kızılötesi algılayıcılarda, yoğun bir ışık huzmesi hedefe doğru gönderilmekte, hedeften yansıyan ışık, algılayıcı üstünde olan bir alıcı ile geri alınmakta, alınan ışık huzmesinin yoğunluğuna göre mesafe tespit edilmektedir. Bu açıdan, ultrasonik ve lazer mesafe buluculara nazaran daha karmaşık bir hesap gerektirmektedir.

1.3.1.4. Takometreler

Takometreler rotasyonel şaft süratini ölçmek için kullanılmaktadır. Şaft süratine göre, araç hızı ve ivmesi hesaplanmaktadır. Takometreler uygun fiyatlı olmalarına rağmen, hızlı aşınmaları ve gürültülü çıktıları nedeniyle, hassas olmayan kontrol problemlerinde tercih edilmektedirler.

1.3.1.5. Optik kodlayıcılar (Encoders)

Robotlarda lokalizasyon maksatlı kullanılan algılayıcıların başında gelen optik kodlayıcılar, şaft üzerinde dönen simgelerin okunması suretiyle çalışmaktadırlar. Üç parçadan oluşmaktadır. Bunlar; sayısallaştırıcı disk, ışık kaynağı ve ışık algılayıcıdır (Şekil 1.10). Şaft ile birlikte dönen sayısallaştırıcı disk üzerine düşen ışık kaynağından gelen ışık hüzmeleri, ışık algılayıcı tarafından sayılmakta ve ilerleme elde edilmektedir. Belirli bir referans noktaya göre, ilerleme üstüne eklenerek, pozisyon bilgisi alınabilmektedir [41].



Şekil 1.10. Optik kodlayıcının çalışma prensibi [41]

1.3.1.6. Cayrolar ve pusulalar

Cayrolar, açısal dengenin korunması ilkesi ile çalışan, yön ölçümü amacıyla kullanılan algılayıcılardır. Pusulalar, cayro mantığı ile çalışan daha basit yön bulmaya yarayan algılayıcılardır. Her ikisi de yön tayini için kullanılmaktadır.

1.3.1.7. Görüntü tabanlı algılayıcılar

Çevrenin görüntüsü çok zengin bir bilgi kaynağıdır. Bu açıdan görüntünün bilgisayar ile işlenmesi ve kullanımı, önemli araştırma alanlarından biri olmuştur. Görüntü tabanlı algılayıcıların merceklerinden geçen ışık fotonları algılayıcıların üzerine düşerek bıraktıkları enerji elektriksel sinyallere çevrilir. Bu işlem için kullanılan iki farklı çözüm bulunmaktadır. Bunlar CCD (Charged Couple Device) ve CMOS (Complementary Metal Oxide Semiconductor) algılayıcılardır. CCD algılayıcılar, CMOS algılayıcılara nazaran daha pahalı, daha fazla enerji harcayan, ancak daha kaliteli, düşük gürültü ve canlı renklere sahip bir algılayıcı türüdür. CMOS algılayıcılar CCD algılayıcılara göre daha ucuz olmaları ve daha az enerji sarfiyatlarından dolayı daha fazla tercih edilmektedir [32].

1.3.1.8. Global konumlandırma sistemi (GPS)

Özellikle dış ortamlarda çalışan mobil robotlar için olmazsa olmaz bir algılayıcı olan GPS, dünya çevresindeki uydular sayesinde, birkaç metrelik hata payı ile dünya üzerindeki konumumuzu veren bir algılayıcıdır. Amerikan Silahlı Kuvvetleri için tasarlanmış ve onun tarafından yönetilmesine rağmen, ticari ve kişisel uygulamalarda çok yoğun bir biçimde kullanılmaktadır. Üçgenleme esasına göre konum belirlemektedir. Bu kapsamda, dört uydudan aldığı bilgi ile dünya üzerindeki konum bilgisi tespit edilmektedir.

1.3.2. Kontrol bilgisayarı

Kontrol bilgisayarı, robotu ve eyleyicileri kontrol etmek için gerekli iş zekasını sağlamaktadır. Algılayıcılardan gelen bilgileri işlemekte, eyleyiciler için kontrol komutlarına dönüştürmekte ve robota verilen görevin icra edilmesini sağlamaktadır.

Robotu yöneten tüm yazılımların, gerekli hafıza birimlerinin, algılayıcılar ve eyleyiciler ile ilgili arabirimlerin bulunduğu yerdir.

1.3.3. Eyleyiciler

Robotu bir insana, algılayıcıları duyu organlarımıza, kontrol bilgisayarını beynimize benzetecek olursak, eyleyiciler insanın kaslarını oluşturmaktadır. Onlar sayesinde kontrol bilgisayarından gelen komutlara uygun olarak, robot ilerleyebilmekte ve görevini icra edebilmektedir.

Robotun gitmesini sağlayan ve tekerlerin dönmesini sağlayan motor gibi, bir çok elektro-mekanik cihazın kombinasyonundan oluşan eyleyiciler mevcuttur. Mobil robotların hareketlerini gerçekleştirebilmeleri için kullandıkları tekerlek, palet veya bacak gibi gezginliği sağlayacak parçalar ile endüstriyel robotlardaki kollar eyleyici elemanların bir kısmıdır.

2. YOL BULMA

Robotlarda yol bulma problemi, iki nokta arasında en kısa ve en az maliyetle gidilebilen yolun belirlenmesi olarak açıklanabilir. Basit bir cümle ile ifade edilen bu süreç, karmaşık bir çok işlemi içermektedir. Otonom bir mobil robot açısından, bir noktadan diğer bir noktaya gitme işlemi için aşağıdaki dört görevin başarı ile yerine getirilmiş olması gerekmektedir. Bunlar;

Algılama: Robot algılayıcıları aracılığı ile çevre hakkında bilgi toplamalı,

Lokalizasyon: Robot kendi pozisyonunu takip edebilmeli,

Yol bulma: Hedef noktaya en kısa ve en az maliyetle gidilecek yolu tespit edebilmeli,

Engelden sakınma: Robot karşılaşabileceği engellerden sakınma yapabilmeli.

Yukarıdaki görevlerden algılama ile ilgili bilgiler Bölüm 1'de verilmiştir. Bu bölümde lokalizasyon ve yol bulma ile ilgili algoritmalar hakkında bilgi verilmesi amaçlanmıştır. Engelden sakınma Bölüm 3'de detaylı olarak ele alınacaktır.

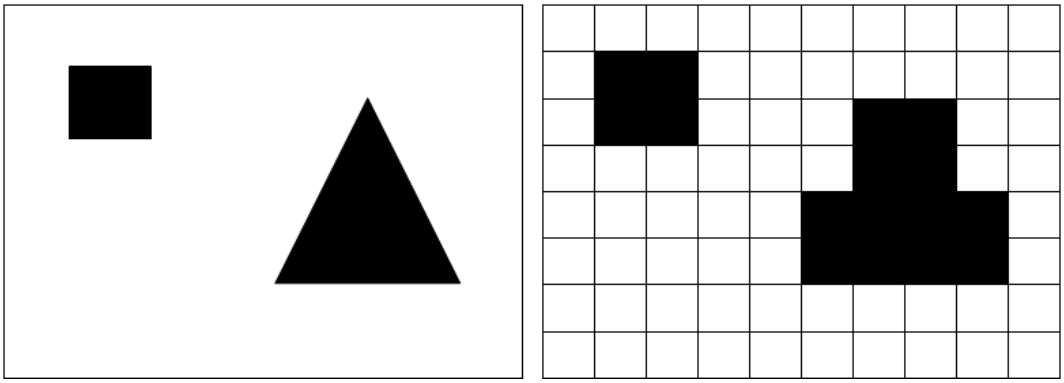
2.1. Lokalizasyon

Robotun kendi konumunu takip olarak özetlenebilecek lokalizasyonda iki girdi bulunmaktadır. Birincisi, robotun bulunduğu ortamın haritası ve ikincisi de robotun bu harita üzerindeki konumu.

Robot uygulamalarında, ızgara tabanlı ve öz nitelik tabanlı olarak iki tip haritalama yöntemi kullanılmaktadır [42].

2.1.1. Izgara tabanlı haritalama

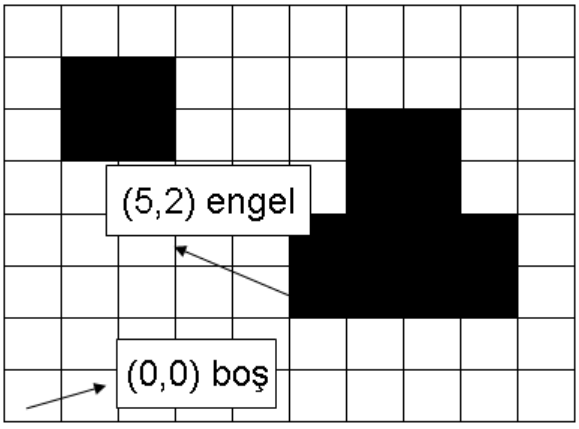
Izgara tabanlı haritalama metodunda, ortamdaki engeller ve robot, hücrelere bölünmüş bir geometrik alan üzerinde takip edilmektedir [43].



Şekil 2.1. Ortamın ızgara tabanlı haritaya aktarılması

Bu hücrelerin boş veya dolu olması, içindeki bilginin engel veya hedef gibi faydalı bilgiler ile dolu olması ızgara tabanlı haritalamada kullanılan çeşitli yöntemlerdir. Şekil 2.1'de ortamdaki engellerin bir ızgara tabanlı haritaya aktarılması gösterilmiştir.

Ortam haritasının hücresel bir ızgara tabanlı haritaya aktarılmasının ardından, ortam bilgileri bir matris olarak elde edilmiş olmaktadır. Bu özellik, ızgara tabanlı haritaların yol bulma problemlerinde yoğun bir şekilde kullanılmasına imkan sağlamaktadır. Şekil 2.1'deki ızgara tabanlı haritanın 8x10'luk bir matris olduğunu varsaydığımızda, bu matrisin (0,0) hücresinin boş ve (5,2) hücresinde engel bulunduğu rahatlıkla söylenebilmektedir.



Şekil 2.2. Izgara tabanlı haritada boş ve engel olan hücreler

Izgara tabanlı haritalamadaki hücrelerin büyüklükleri, haritanın hassasiyeti üzerinde büyük etkisi bulunmaktadır. Özellikle, robotun harita üzerinde konumlandırılması aşamasında, hassasiyet önem taşımaktadır. Bununla birlikte hassasiyet, yanında bir

de maliyet getirmektedir. Bu şekilde detaylı bir ızgara tabanlı harita için, çözüm uzayı büyüklüğü ve hafıza gereksinimi üssel olarak artış gösterecektir [44].

Tez çalışması kapsamında kullanılan algoritmalar, ızgara tabanlı haritalar üzerinde gerçeklenmiştir. Ayrıca, laboratuar ortamında robot üzerinde test edilmesi amacıyla, laboratuar tabanı 8x10'luk ızgara tabanlı bir harita olarak modellenmiştir. Ortamdaki engellerin ve robotun konum bilgilerinin tespiti amacıyla, laboratuar tavanında bulunan bir video kamera kullanılmış, yerdeki hedef, engeller ve robot çeşitli renk kodlarına göre tespit edilerek, harita üzerinde konumlandırılmıştır.

2.1.2. Öz nitelik tabanlı haritalama

Öznitelik tabanlı haritalamada, nesnelerin ortam içindeki şekli hakkında fikir veren ve bu nesnelerin konumlarına göre ortamın belirlendiği haritalardır. Öz nitelikler, robotun konumunun belirlenmesi aşamasında birer yer gösterici olarak kullanılmaktadır. Robot konumu bu yer göstericilere göre belirlenmektedir. Yer göstericileri kapı, pencere gibi doğal yer göstericiler olduğu gibi, GPS uydusu gibi yapay yer göstericiler de olabilir.

Bu yöntemde, yeteri kadar yer göstericinin bulunması ve ortamın genel haritasının biliniyor olması gerekmektedir [43].

2.2. Yol Bulma Algoritmaları

Yol bulma konusunda yapılan çalışmalar sonucunda, Dijkstra [3], A* (A-yıldız) [4], D* (D-yıldız) [5] ve potansiyel alan metodu [6] gibi çeşitli yol bulma amaçlı algoritma ve yöntemler geliştirilmiş ayrıca optimizasyon amaçlı geliştirilmiş genetik algoritma [7] ve karınca kolonisi [48] gibi algoritmalar da yine bu amaçla kullanılmışlardır. Bu algoritmalar genellikle graf teorisi ve yapay zeka tabanlı algoritmalardır.

Ayrıca, karşılaştırıldıklarında her bir metodun kendine özgü avantajları olduğu gibi, dezavantajlarının da bulunduğu görülebilmektedir. Bu nedenle, probleme yönelik en uygun algoritmanın seçilmesi oldukça önemlidir [59].

Yol bulma algoritmaları, ortam hakkındaki bilgiye göre çevrim içi (online)ve çevrim dışı (offline) olarak sınıflandırılabilmektedir. Eğer çevre hakkında tam bilgi mevcut

ise ve engellerin konumları önceden biliniyorsa, bu durumda kullanılan algoritmalara çevrim dışı yol bulma algoritmaları denmektedir. Çevre hakkında kısıtlı bilgi olması durumunda, yol bulma algoritması robotun algılayıcılarına dayalı bir şekilde çalışmak durumunda kalacaktır. Bu tür algoritmalara da çevrimiçi yol bulma algoritmaları denmektedir [45].

Yol bulma algoritmaları ayrıca, kesinlik durumlarına göre sezgisel ve tam algoritmalar olarak ayrılabilir. Sezgisel algoritmaların amacı, kısa zamanda kaliteli bir sonuç elde etmek için arama yapmaktır. Bununla birlikte, tam algoritmaların amacı, eğer mevcut ise optimum sonucu bulmak veya mümkün bir kesin sonuç olmadığını ispatlamaktır. Tam algoritmaların genellikle hesaplama maliyetleri yüksek olabilmekte, diğer tarafta, sezgisel algoritmalar da zor problemler için iyi yolu bulmakta yetersiz kalabilmektedirler [8].

Masehan ve arkadaşları tarafından 2007 yılında, yol bulma algoritmaları ile detaylı bir kronolojik inceleme yapılmıştır [46].

Tez kapsamında bu bölümde, algoritmalar hakkında genel bilgi verilmesi amacıyla; Dijkstra, A*, karınca kolonisi ve genetik algoritma ele alınmaktadır.

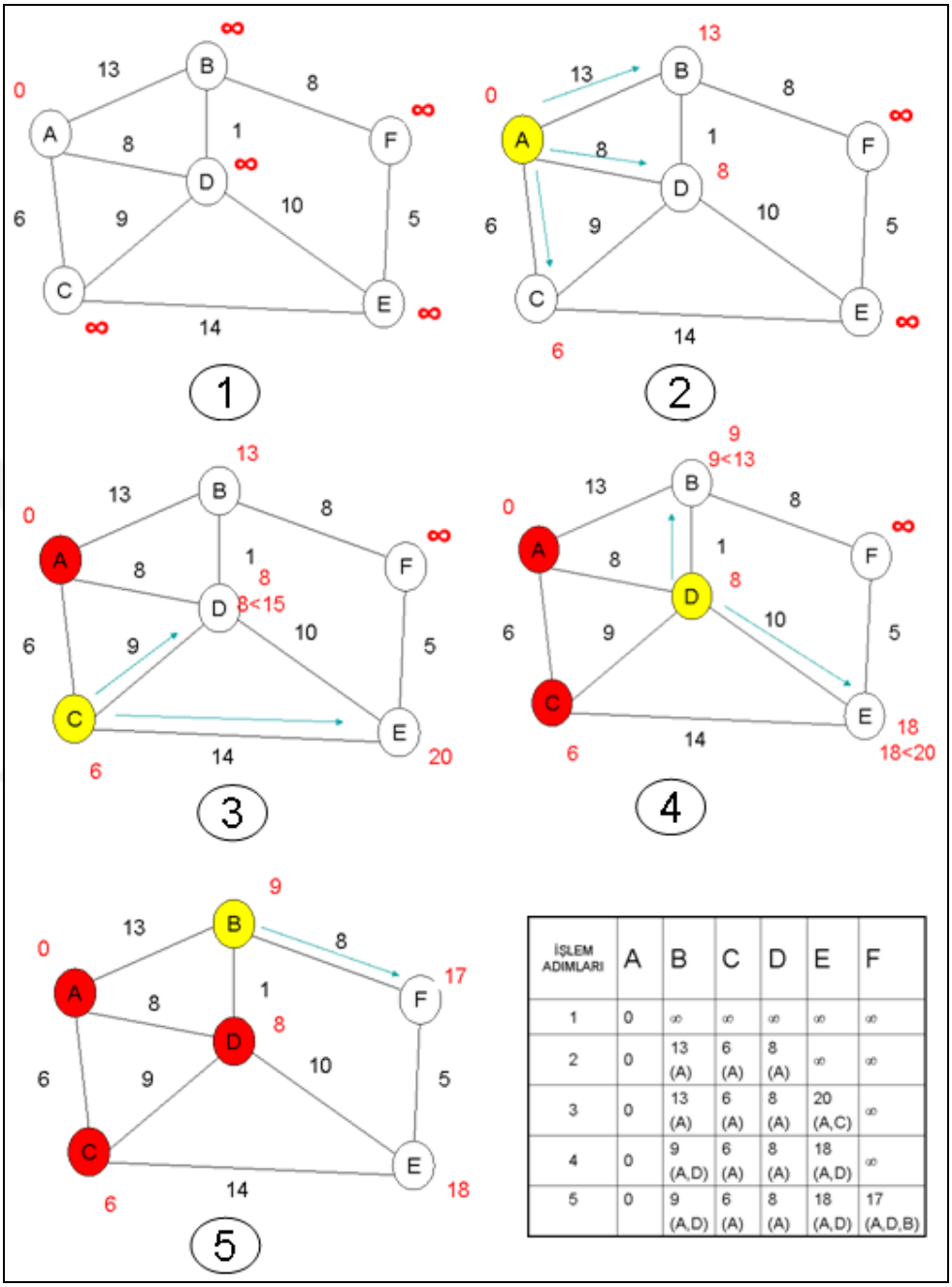
2.2.1. Dijkstra algoritması

Yol bulma problemi çözümlerinde yaygın olarak kullanılan algoritmalardan biri olan Dijkstra algoritması, 1959 yılında Dijkstra tarafından geliştirilmiştir [3]. Başlangıç düğümden hedef düğüme en kısa yolu bulmak için kullanılmaktadır.

Dijkstra algoritmasında, başlangıç düğümden itibaren hedef düğüme kadar tüm düğümler kontrol edilmektedir. Her bir düğüm noktası için hesaplanan mesafe değeri ve değere ulaşmak için gerekli adımlar kayıt edilmektedir. En kısa yolun bulunması garanti edilmektedir. Ancak hedefe olan mesafenin ve düğüm noktası sayısının artması durumunda verimsiz bir algoritma olarak değerlendirilebilmektedir [59].

En başta, düğümler için hesaplanan mesafe değeri sonsuz olarak seçilir. Birinci düğümden itibaren her bir düğüme olan mesafe değeri hesaplanır. Yeni değer eski değerden daha küçük ise, yeni değer, hesaplanan mesafe değeri olarak ayarlanır ve

bu değere ulaşmak için kullanılan düğümler kayıt edilir. En küçük mesafe değerli düğüm ile algoritma devam eder. İşlemi biten düğüm tekrar incelenmez.



Şekil 2.3. Dijkstra algoritmasının ve işlem adımlarının gösterimi

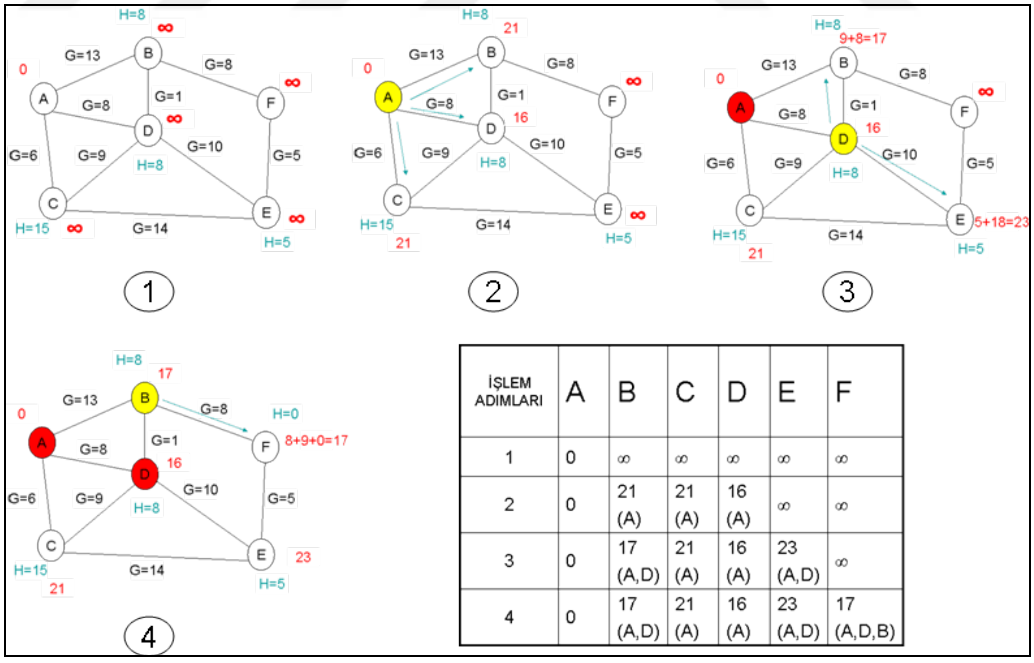
Şekil 2.3'de A'dan F'ye gitmek için en kısa yolun hesaplanması amacıyla Dijkstra algoritmasının işlem adımları görünmektedir. Ayrıca, her bir işlem adımında hesaplanan mesafe değerleri ve bu değerlere ulaşmak için kullanılan düğüm noktaları görülmektedir. Bu örnekte en kısa yolun, 17 birim mesafe ile A,D,B ve F yolu olduğu görülmektedir.

2.2.2. A* (A yıldız) algoritması

A* algoritması da başlangıç düğümü ile hedef düğüm arasındaki en kısa yolun bulunması amacıyla kullanılmaktadır. 1968 yılında geliştirilmiştir [4]. Sezgisel yol bulma algoritmalarından biridir. Toplam mesafe değerleri hesaplanırken, gerçek mesafenin yanı sıra, sezgisel mesafede dikkate alınmaktadır.

Dijkstra algoritmasında kullanılan graf, A* algoritması kullanarak çözülecek olursa, öncelikle her bir düğüm için sezgisel (H) değerlerin bilinmesi gerekmektedir. Bu örnekte, sezgisel değerler kuş uçuşu direk mesafe olarak alınmakta ve düğümlerin üzerine H değeri olarak gösterilmektedir. Düğümler arası gerçek mesafe (G) olarak tanımlanmaktadır.

İlk düğümden itibaren komşu düğümlerin mesafe değerleri, Mesafe = $G+H$ olacak şekilde hesaplanır. Hesaplanan mesafe değeri ve kullanılan düğümler kayıt edilir. Bir sonraki işlem adımında en küçük mesafe değerli düğümden devam edilir. Diğer düğümlere olan mesafe yine gerçek mesafe ve sezgisel değerin toplamı olarak hesaplanır.



İŞLEM ADIMLARI	A	B	C	D	E	F
1	0	∞	∞	∞	∞	∞
2	0	21 (A)	21 (A)	16 (A)	∞	∞
3	0	17 (A,D)	21 (A)	16 (A)	23 (A,D)	∞
4	0	17 (A,D)	21 (A)	16 (A)	23 (A,D)	17 (A,D,B)

Şekil 2.4. A* algoritmasının ve işlem adımlarının gösterimi

Şekil 2.4'te A'dan F'ye gitmek için en kısa yolun hesaplanması amacıyla A* algoritmasının işlem adımları görünmektedir. Ayrıca her bir işlem adımında

hesaplanan mesafe değerleri ve bu değerlere ulaşmak için kullanılan düğümler görülmektedir. Bu örnekte en kısa yolun, 17 birim mesafe ile A,D,B ve F yolu olduğu görülmektedir.

Dijkstra algoritması ile aynı sonuç beş işlem adımında bulunurken, A* algoritması ile dört işlem adımında bulunmuştur. Bu örnekte olduğu gibi, sezgisel yöntemler tüm olasılıkların denenmemesi nedeniyle diğer algoritmalardan daha performanslı sonuçlar verebilmektedir.

A* algoritmasının ızgara tabanlı bir harita üzerinde modellemesi Şekil 2.5'de gösterilmiştir. Burada, E engel hücreleri olmak üzere, K kaynak noktasından, H hedef noktasına ilerlemek için, önce K etrafındaki her bir hücrenin mesafe değeri hesaplanmaktadır. Mesafe değeri, hücrenin K'ya olan gerçek uzaklığı olan *G* değeri ile, hücrenin hedef noktaya olan sezgisel H değerinin toplamı ile elde edilmektedir. Harita üzerinde, hücreler arası mesafe 10'ar birim ve $\sqrt{2} \cong 1{,}4$ olarak alınmıştır [47].

	74 G=14H=60	60 G=10 H=50	54 G=14H=40	E			
	60 G=10 H=50	K	40 G=10 H=30	E		H	
	74 G=14H=60	60 G=10 H=50	54 G=14H=40	E			

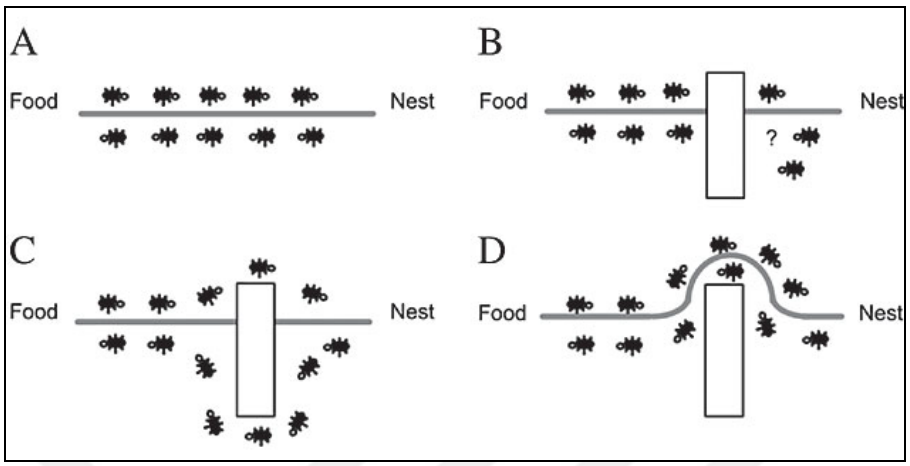
Şekil 2.5. A* algoritmasının ızgara tabanlı bir haritada gösterimi [47]

2.2.3. Karınca kolonisi algoritması

Karınca kolonisi algoritması, karınca kolonisi ve karıncaların davranışlarından esinlenerek geliştirilmiş bir yol bulma algoritmasıdır. Bu yaklaşım, hiçbir görsel ipucu olmaksızın, karıncaların gerçek yaşamda, yiyecek kaynağından yuvalarına kadar olan en kısa mesafeyi bulma yöntemlerini baz almaktadır [48].

Karıncalar, yiyecek veya yuvaya ilerlerken feremon adı verilen bir salgıyı yola salmaktadırlar. Bu feremon, arkadan gelen diğer karıncaları yol seçiminde etkilemektedir. Bir yolun birçok karınca tarafından seçilmesi durumunda, o yol üzerindeki feremon miktarı daha çok artacak, feremon miktarı nedeniyle, daha çok

karınca tarafından bu yol tercih edilecektir. Ayrıca, feremon miktarı zamanla birlikte ortamdan buharlaşarak kaybolmaktadır.



Şekil 2.6. Karıncaların engel ile karşılaşma durumları [49]

Şekil 2.6'da, karıncaların engel ile karşılaşma durumu canlandırılmıştır. Başlangıçta, yemek ve yuva arasında düz bir hat üzerindeki feremon boyunca ilerleyen karıncalar, bir engelle karşılaşılması durumunda iki yönden birini seçmekteler, bir süre sonra ise kısa yol yeni feremon yolu olarak oluşmaktadır.

Karınca kolonisi ile ilgili literatürdeki çalışmaların bir özeti [50] de bulunabilir.

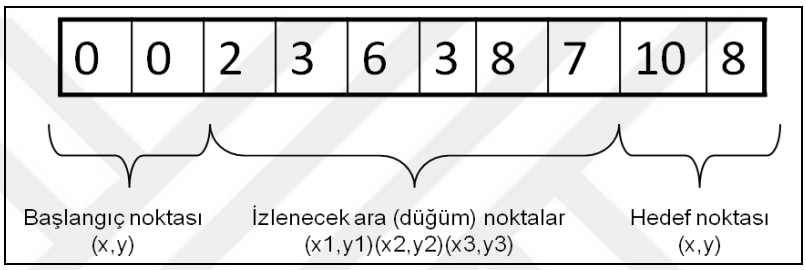
2.2.4. Genetik algoritma

Genetik algoritma, evrimsel algoritmalar içinde, genetik bilimine dayanan bir optimizasyon yöntemidir. Holland tarafından 1975 yılında ortaya atılmış [7], 1992 yılında Koza tarafından programlamada kullanılarak, genetik programlama adını almıştır. Genetik alanındaki çaprazlama, mutasyon ve doğal seçilim gibi biyolojik süreçlerden esinlenerek geliştirilen, bu süreçlerin matematiksel olarak modellenmesi ile fonksiyonları optimize eden bir algoritmadır [51].

Sezgisel bir yöntem olması sebebiyle, bulunan çözüm en iyi çözüm olmayabilir. Ancak, geleneksel yöntemler ile çok uzun sürebilecek problem çözümlerinin, başarı ile kabul edilebilir kısa bir sürede elde edilmesinde faydalı olmaktadır. Bununla birlikte, rastsal bir algoritma olarak da sınıflandırılmamalıdır. Doğadaki en iyi genlerin sonraki kuşaklara aktarılması kuralına paralel olarak, genetik algoritmada

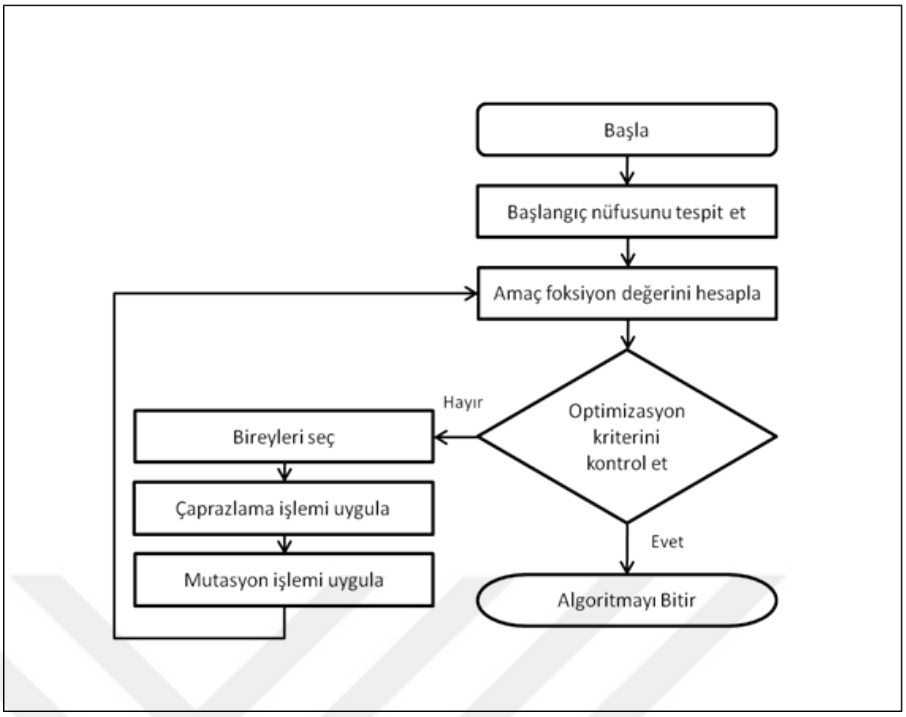
popülasyondaki en iyi genlerin, sonraki kuşaklara aktarılması yolu ile, geçmiş kuşakların yönlendirmesine dayalı bir arama yapılmaktadır.

Doğadaki tüm canlıların temelini oluşturan hücrelerin yapıtaşı olan genler ve kromozomlar, genetik algoritmada popülasyonu oluşturan kromozomlar olarak modellenmektedir [51]. Nasıl ki insandaki kromozomlar göz rengi, ten rengi gibi bilgileri kodlanmış olarak içermekteyse, genetik algoritmada kromozomlar, çözüm uzayı içindeki çözüm vektörlerini kodlanmış olarak içermektedir. Şekil 2.7'de tam sayı olarak, başlangıç noktasından hedef noktasına ilerleme aşamasındaki düğümleri içeren bir tam sayı kromozom örneği görülmektedir.



Şekil 2.7. Genetik algoritmada kullanılan kromozom örneği

Genetik algoritma akış şeması Şekil 2.8'de gösterilmekte olup, özetle; ilk adımda çözümler kromozom olarak kodlanır ve başlangıç nüfusu tespit edilir. Belirlenmiş olan amaç fonksiyonu aracılığı ile her bir bireyin amaç fonksiyonu değeri hesaplanır. Amaç fonksiyonun optimizasyon kriterini sağlaması durumunda algoritma sonlandırılır aksi halde yeni bireylerin üretimine geçilir. Seçim, çaprazlama ve mutasyon işlemleri ile yeni bireylerin üretimi yapılır. Yeni bireylerin amaç fonksiyon değerlerinin hesaplanmasıyla algoritmaya devam edilir.



Şekil 2.8. Genetik algoritma akış şeması

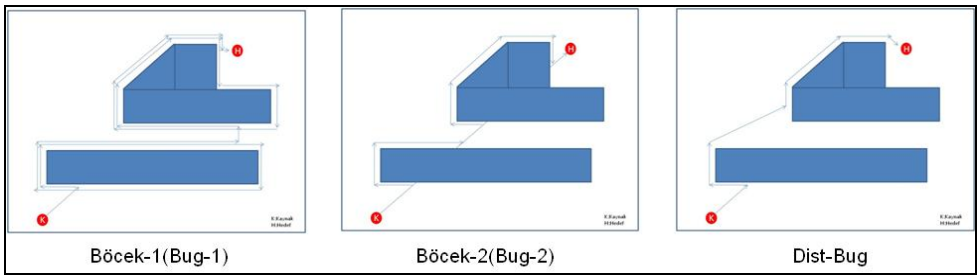
Tez kapsamında geliştirilen robot seyir sisteminde, yol bulma probleminin genetik algoritma ile çözülmesi nedeniyle, genetik algoritma konusu Bölüm 5'te daha geniş olarak ele alınacaktır.

3. ENGELDEN SAKINMA ALGORİTMALARI

Engelden sakınma veya çarpışma önleme, otonom robotun güzergahı üzerindeki öngörülebilen veya görülemeyen engellerden sakınma amacıyla yapılan bir faaliyetler dizisidir. Bu amaçla, çeşitli engelden sakınma algoritmaları geliştirilmiştir. Bu bölümde literatürde sıklıkla kullanılan bazı algoritmalar hakkında bilgi verilecektir.

3.1. Böcek Algoritmaları

Böcek algoritmaları, engelden sakınma konusunda ilk örnekler olmalarının yanı sıra, en çok kullanılan algoritmalar olma özelliğini de taşımaktadırlar [52]. Böcek algoritmalarının; Böcek-1, Böcek-2 ve Dist-bug olarak üç türü bulunmaktadır [53]. Ortak özellikleri, bir engelle karşılaşılması durumunda, robotun engel etrafında dolaşmasını sağlamalarıdır. Şekil 3.1'de,robotun böcek algoritmaları ile hareketleri görülmektedir.



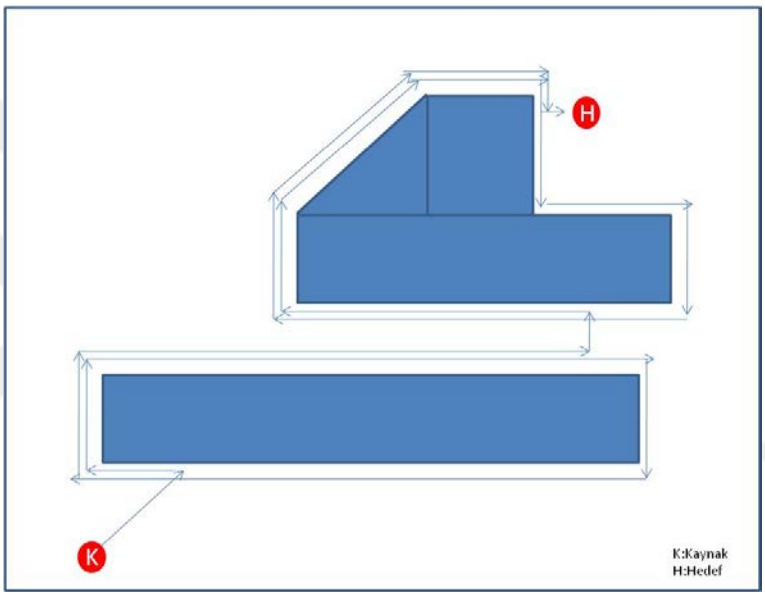
Şekil 3.1. Böcek algoritmalarında robotun hareketi

Literatürde bu üç böcek algoritmasının varyasyonu olarak, çok çeşitli (Alg1, Alg2, LeaveBug, TangentBug, VisBug-21, VisBug-22) böcek algoritmaları da geliştirilmiştir [54]. Temelde aynı algoritmalar esas alınmak kaydıyla, VisBug grubunda optik algılayıcıların kullanılması gibi aralarında farklar da bulunmaktadır.

Bu tez çalışması kapsamında, Böcek1, Böcek2 ve Dist-bug algoritmaları ele alınmış, bu algoritmalar yazılımsal olarak ayrıca gerçeklenmiştir. Bu bölümde, algoritmalar hakkında açıklayıcı bilgiler verilmiş, algoritmaların gerçeklenmesi ile ilgili detaylar Bölüm 6'da ele alınmaktadır.

3.1.1. Böcek-1 algoritması

Böcek-1 algoritması en basit engelden sakınma algoritmasıdır. Buradaki ana fikir, robotun yolu üstündeki her engelin dış çevresini takip etmek ve bu sayede çevresini dolaşmaktır [8]. Böcek-1 algoritmasıyla, robot önce engelin etrafını çepeçevre dolaşır, ilk tur esnasında hedefe en yakın mesafedeki noktayı tespit eder. İlk tur bittikten sonra, hedefe en yakın mesafedeki noktaya giderek engelden ayrılır ve hedefe doğru yoluna devam eder (Şekil 3.2). Bu yöntem elbette verimsiz bir yöntemdir ancak, robotun erişebilir her türlü hedefe ulaşmasını garantilemektedir.

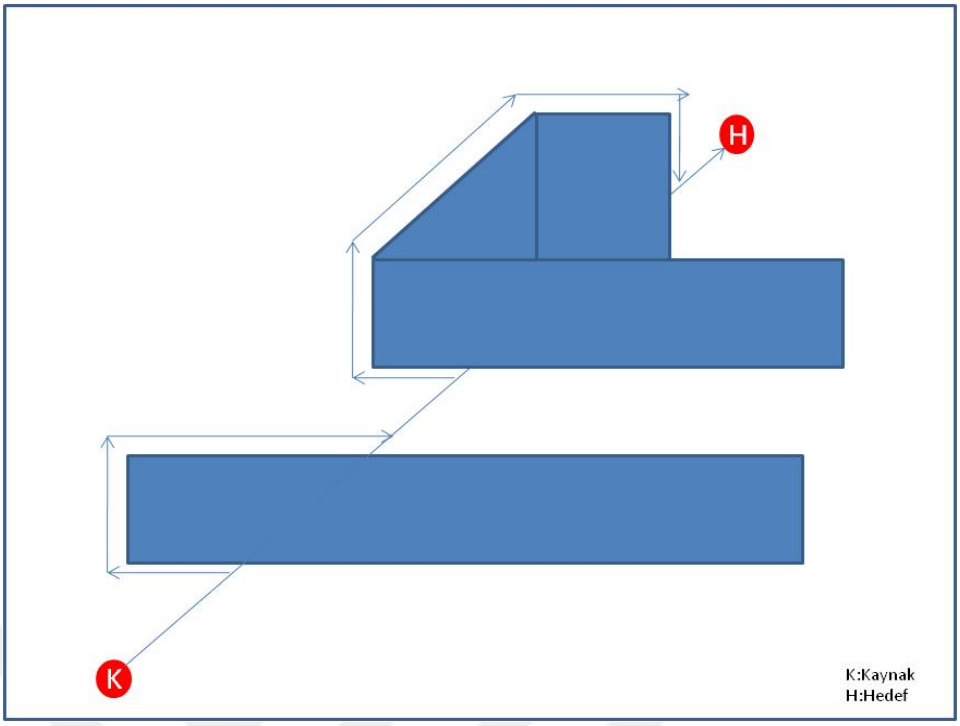


Şekil 3.2. Böcek-1 algoritması

3.1.2. Böcek-2 algoritması

Böcek-2 algoritmasında, robot hedefe doğru ilerlerken daha önceden planlanan bir yolu takip etmektedir. Bir engel ile karşılaşıldığında, engelin dış hattını takip etmeye başlar ancak, tekrar planlanan yol hattına ulaştığı ilk anda engelden ayrılır ve hedefe doğru yine planlanan yol üzerinde ilerlemeye devam eder [9].

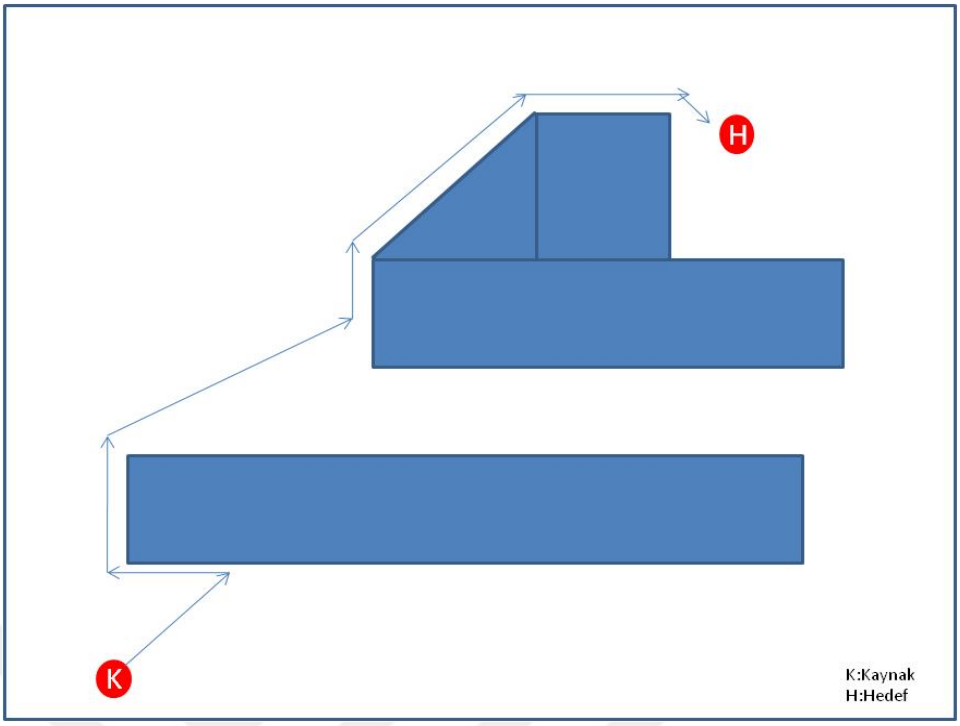
Genel olarak, Böcek-1 algoritmasının geliştirilmiş şekli olan bu algoritmada, robotun kat ettiği mesafe önemli ölçüde azalmaktadır (Şekil 3.3). Ancak, bu algoritmanın da verimsiz veya optimal çözümler üretmeyeceği örnekler olabilmektedir.



Şekil 3.3. Böcek 2 algoritması

3.1.3. Dist-bug algoritması

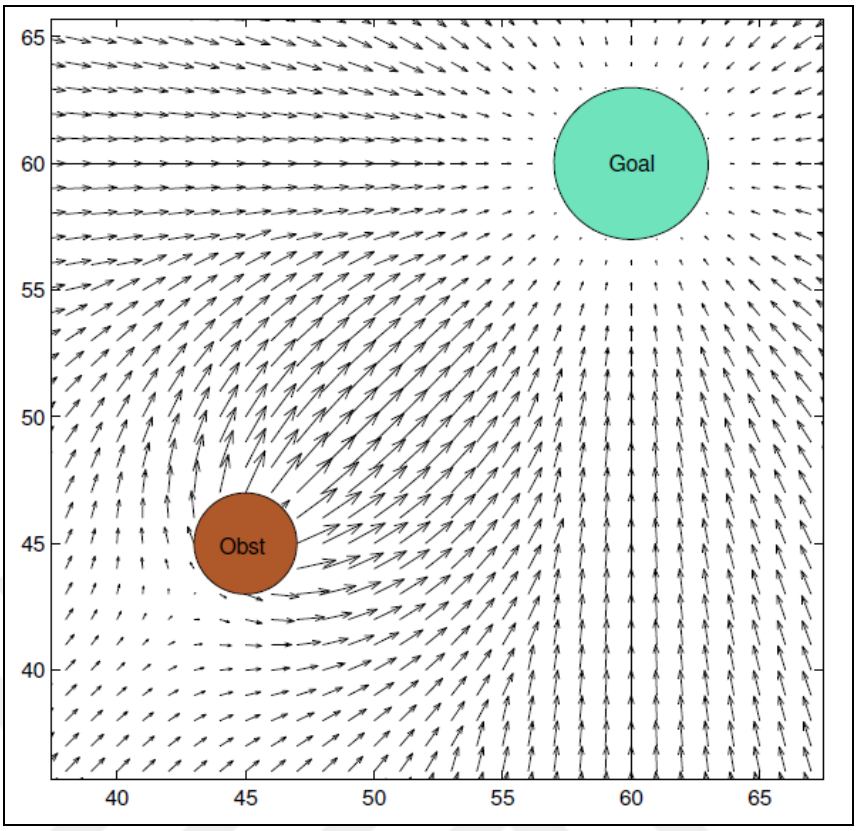
Dist-bug algoritmasında, robot hedefe doğru ilerlerken bir engel ile karşılaşması durumunda, diğer böcek algoritmalarında olduğu gibi, engelin dış hattını takip etmeyle başlar. Ancak, tam bir tur atmak yerine, sürekli olarak sonar değerlerini kontrol etmek suretiyle, hedefe doğru olan serbest bir alanın varlığını hesaplar [53]. Hedefe doğru serbest bir alanın varlığının tespiti durumunda, engelden ayrılır ve hedefe doğru ilerlemeye başlar (Şekil 3.4).



Şekil 3.4. Dist-bug algoritması

3.2. Potansiyel Alan Yöntemi

Potansiyel alan yöntemi (potential field method-PFM), otonom gezgin robotların engellerden sakınarak istenilen hedefe gitmelerinde sıkça kullanılan bir yöntemdir. Bu yöntemde [10], robotun çalışma alanının yapay bir potansiyel alanla dolu olduğu kabul edilmekte; hedef, çekici bir vektör oluşturarak robotun hedefe ilerlemesini sağlarken; engeller, itici bir vektör oluşturarak robotun engellerden uzaklaşmasını sağlamaktadır (Şekil 3.5). Potansiyel alan metodunun en bilinen problemi, robotun U harfi şeklindeki engellerin tuzağa düşmesidir [52].

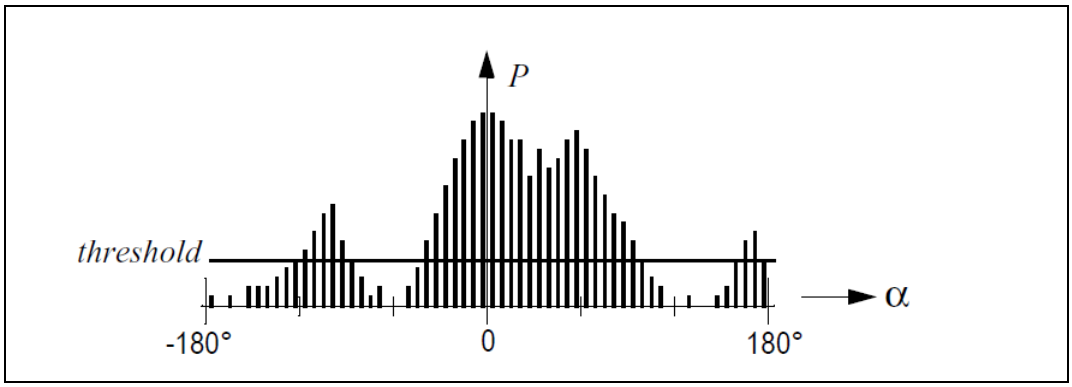


Şekil 3.5. Robot ve engel arasındaki itici ve çekici vektörler [55]

3.3. Vektör Alan Histogramı (Vector Field Histogram-VFH)

Böcek tipi algoritmaların eleştirildiği en önemli nokta, robot hareketinin, anlık algılayıcı değerlerini temel almasıdır. Bu durum, sağlıklı bir engelden sakınma için yeterli bilginin gelmediği durumlarda istenmeyen sonuçlar ortaya çıkartabilmektedir. Robotun etrafında yerel bir haritanın oluşturulması ile VFH [11] bu kısıtlamanın üstesinden gelebilmektedir. Bu yerel harita sadece, nispeten son algılayıcı ölçümlerinden oluşturulmuş bir doluluk gridi (occupancy grid) haritasıdır.

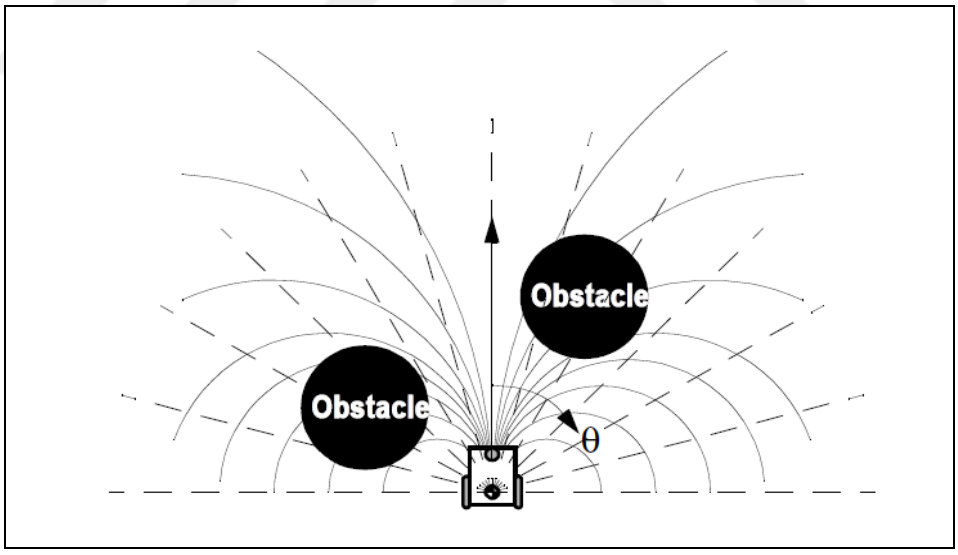
Engelden sakınma için VFH'da bir kutupsal histogram oluşturulmaktadır. X ekseni, engelin bulunduğu yönü/açıyı göstermekte, y ekseni ise bu yönde gerçekten bir engel olma olasılığını göstermektedir (Şekil 3.6).



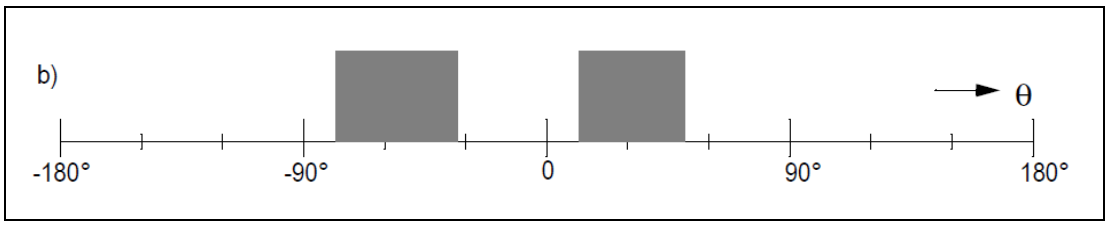
Şekil 3.6. Polar Histogram [11]

Bu histogramdan, robotun dönüş açısı hesaplanmaktır. Önce robotun geçebileceği kadar geniş açıklıklar tespit edilir, daha sonra her bir aday açıklık için maliyet fonksiyonu uygulanır. En düşük maliyetli açıklık seçilir. Maliyet fonksiyonu üç girdiye sahiptir. Bunlar; hedef yönü, aracın yeni yönü ve eski yönüdür.

Şekil 3.7 ve Şekil 3.8'de robot önündeki iki adet engel ve bu engellerin histogram olarak görüntüleri ele alınmıştır.



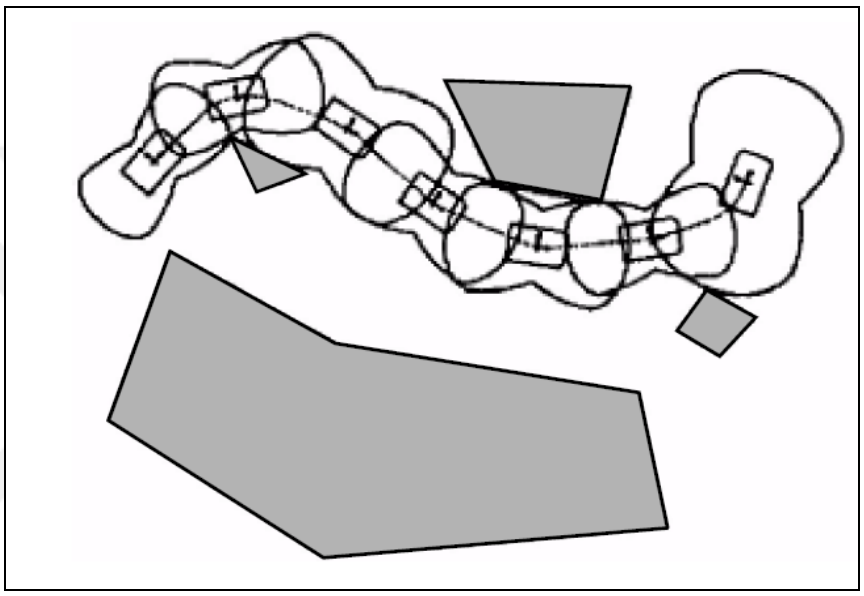
Şekil 3.7. Robot ve önündeki engeller [52]



Şekil 3.8. Robotun önündeki engellerin kutupsal histogram görüntüsü [52]

3.4. Kabarcık Bandı Tekniği (The Bubble Band Technique)

Kabarcık, bir robotun engellere çarpmaksızın her hangi bir yönde serbestce ilerleyebileceği boş alan olarak tanımlanmaktadır [7]. Kabarcık, robotun haritası ve mesafe bilgileri ile birlikte robotun basitleştirilmiş bir modeli olarak oluşturulur. Robotun fiziki şeklinin izdüşümü de kabarcığın büyüklüğünü belirlemektedir (Şekil 3.9). Oluşturulan kabarcıklar, robotun ilk pozisyonundan son pozisyonuna kadar ilerletilerek, robotun yol boyunca karşılaşacağı serbest alanlar tespit edilebilir.

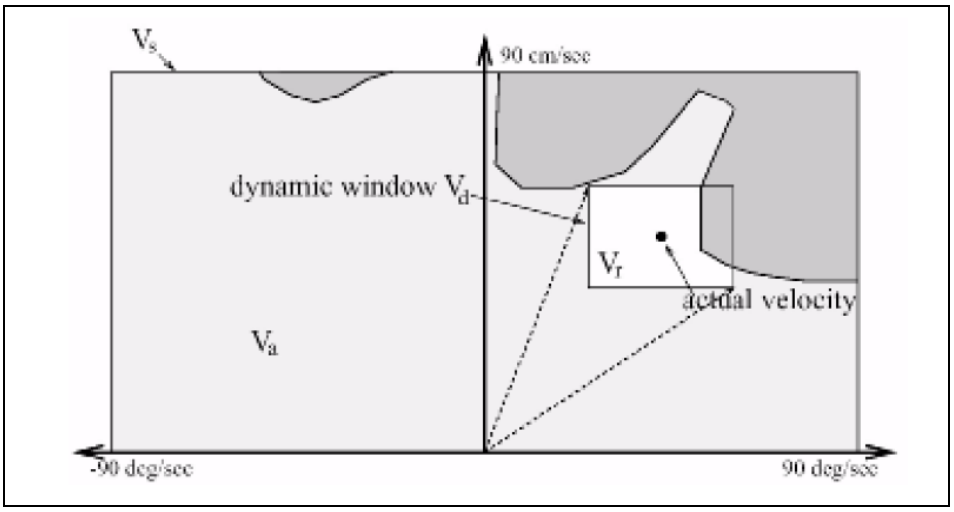


Şekil 3.9. Kabarcık dizisi tekniği [7]

3.5. Dinamik Pencere Yaklaşımı (Dynamic Window Approach)

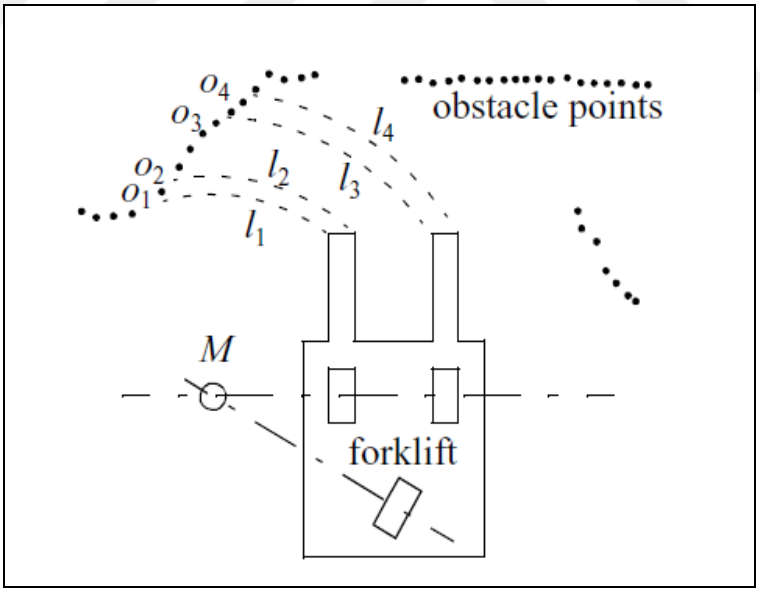
Dinamik pencere engelden sakınma yöntemi, robot kinematiklerini dikkate alan bir yöntemdir [14]. Bu yaklaşımda, en azından belirli bir zaman diliminde, robotların dairesel yaylar şeklinde hareket ettikleri varsayılmaktadır. İlk olarak, bir sonraki örnekleme periyodu içinde kullanılacak tüm veriler ele alınır. Bu verilere ivmelenme, robotun özellikleri ve periyodun zamansal uzunlukları da dahil edilmektedir.

Bu küçük zaman aralığı içindeki periyoda dinamik pencere adı verilmektedir. Bir sonraki adımda, bu dinamik pencere içinde robotun bir engele çarpmadan ilerleyebileceği noktaya kadar olan veriler dikkate alınmaktadır. Şekil 3.10'da bir dinamik pencere ve bu pencere içindeki engel görünmektedir.



Şekil 3.10. Dinamik pencere örneği [14]

Sonraki adımda, bu dinamik pencere içinde, robotun hızı ve kinematikleri dikkate alınarak yeni rota bulunmaktadır. Şekil 3.11'de bir forkliftin M merkezli olarak yapmakta olduğu dönme hareketi esnasında, engellere olan l uzaklıkları görülmektedir.



Şekil 3.11. Bir forkliftin dinamik pencere içinde engelden sakınması [52]

4. PIONEER 3-DX ROBOT

Bu bölümde, tez çalışması kapsamında kullanılan algoritmaların gerçeklenmesi amacıyla kullanılan Pioneer 3-DX robot hakkında bilgi verilmektedir. Öncelikle, robotun genel özellikleri açıklanmakta ardından robotun yönetilmesi maksadıyla kullanılan yöntemler hakkında bilgi verilmektedir. Tez çalışması kapsamında yoğun olarak kullanılan robot üstü sonarlar hakkında bilgi verilmektedir. Ayrıca, uygulama çalışmalarının benzetim ortamında denenmesi amacıyla kullanılan robot benzetim ortamı yine bu bölümde anlatılmaktadır.

4.1. Robotun Genel Özellikleri

Pioneer 3-DX akademik dünyada yaygın olarak kullanılan bir gezgin robottur. Adept MobileRobots firması tarafından üretilmiştir. Adept MobileRobots firması araştırma/geliştirme amaçlı olarak Pioneer robot ailesinin yanı sıra ayrıca AmigoBot, PeopleBot ve PowerBot robot türlerini de üretmektedir. Tüm bu robotların ortak özelliği araştırma/geliştirme faaliyetlerindeki olası darbelere karşı sert alüminyum gövdeye sahip olmalarıdır. Robotlar farklı tekerlek konfigürasyonuna sahip olabilmektedirler. Pioneer 3-DX robotu iki adet diferansiyel ve bir adet mobilya tekerleğe sahiptir.



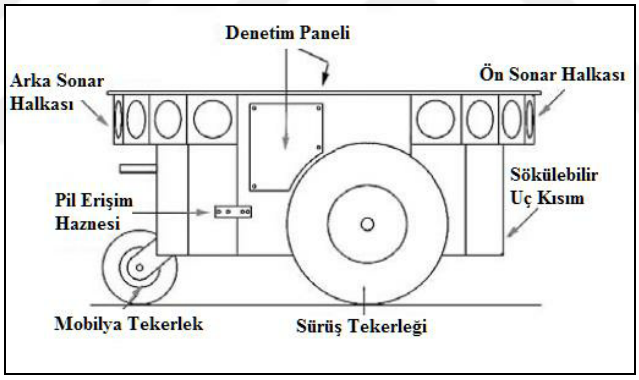
Şekil 4.1. Pioneer 3-DX gezgin robot [57]

Akademik dünyada yaygın olarak kullanılması nedeniyle, bir çok çalışmada referans olarak kabul edilen bir robot platformu olarak değerlendirilmektedir [58]

Pioneer 3-DX gezgin robotta bulunan standart özellikler aşağıda verilmektedir. Tez çalışmasında bu standart özelliklere sahip bir Pioneer 3-DX (Şekil 4.1) robot kullanılmıştır.

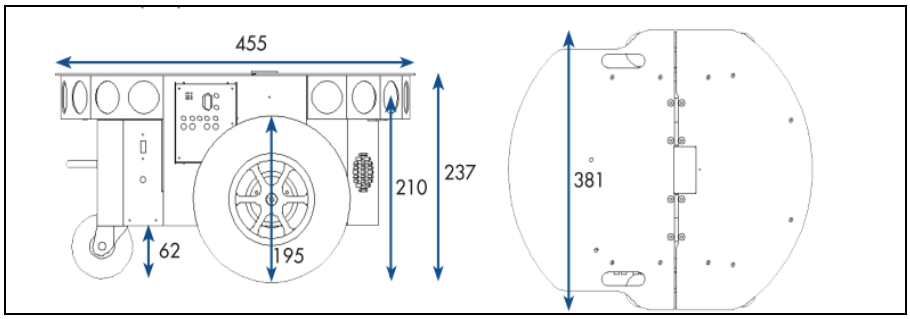
- Kullanıcı denetim paneli,
- Yüksek performanslı mikro denetleyici,
- Yüksek çözünürlükte kodlayıcı,
- 1 adet RS-232 seri haberleşme portu,
- 3 adet batarya,
- 8 adet ön sonar algılayıcı,
- Reset ve motor butonları.

Yukarıda belirtilen özelliklerin yanı sıra, opsiyonel olarak, arka sonar algılayıcılar, lazer mesafe ölçücü gibi ek özellikler yine MobileRobots firmasından temin edilebilmektedir.



Şekil 4.2. Pioneer 3-DX gezgin robot genel özellikleri [59]

Standart özellikteki bir Pioneer 3-DX robot 9 kg ağırlığı ile oldukça hafif bir gövdeye sahip olması nedeniyle çalışmalarda istenilen yere rahatlıkla taşınabilmektedir. Sert alüminyum gövdesi ile darbelere karşı dayanıklılığının yanı sıra, 23 Kg a kadar üstünde ilave yük taşıma kapasitesine sahiptir. Uzunluğu 44,5cm, genişliği 39,3cm, yüksekliği 23,7cm'dir. Şekil 4.2 ve 4.3'de Pioneer 3-DX robotun genel ve teknik özellikleri görülmektedir.



Şekil 4.3. Pioneer 3-DX gezgin robot teknik özellikleri [60]

Pioneer 3-DX robotlar 2 adet dolma tekere ve bir adet mobilya tekerleğe sahiptir. Dolma tekerler, yüksek hızlı, yüksek torklu ve çift yönlü doğru akım (DA) motorlar tarafından yönetilmektedir. DA motorların üzerinde yüksek çözünürlüklü optik şaft kodlayıcılar bulunmaktadır. Bu kodlayıcılar hassas konum, hız algılama ve gelişmiş konum tahmini için kullanılmaktadır [59].

4.2. Robotun Yönetilmesi

Pioneer 3-DX robot, üzerindeki ARCOS (Advanced Robot Control and Operations Software) istemci-sunucu arayüzü ile kontrol edilebilmektedir. ARCOS aracılığı ile robotun işleyişi ile ilgili, motorların çalıştırılması, sonarların tetiklenmesi, sonar değerlerinin rapor edilmesi, tekerlerdeki kodlayıcılardan verinin aktarılması gibi, tüm alt seviye görevler yönetilebilmektedir. Aynı zamanda, bir sunucu olarak çalışan ARCOS, istemci robot yönetim yazılımları ile haberleşmeyi de sağlamakta ve robotun etkin bir şekilde yönetilebilmesini sağlamaktadır.

Robota gönderilecek komutlar ve robottan alınabilecek veriler ile ilgili iki yöntem bulunmaktadır. Bunlar; ARCOS'a komutların direkt gönderilmesi yöntemi ile ARIA kütüphanesinde yer alan daha üst seviye komutların gönderilmesi yöntemleridir [58].

4.2.1. Robotun ARCOS komutları ile direkt yönetilmesi

Robot üzerindeki ARCOS sunucusuna RS-232 bağlantısı kurularak, veri paketleri kullanmak suretiyle, istenilen yazılım üzerinden haberleşme yapmak mümkündür. Veri paketleri; komut paketleri ve sunucu bilgi paketleri (server information packets-SIP) olarak ikiye ayrılmaktadır. Bu paketlerin toplam uzunluğu 207 Byte'ı geçemeyeceği gibi 204 Byte'dan daha uzun paketler sunucu tarafından göz ardı edilmektedir [58].

Komut paketleri; 2 byte uzunluğunda bir başlık, 1 byte paket uzunluğu, 1 byte komut numarası, argüman tipi ve argümanlar ve 2 byte uzunluğunda kontrol değeri olmak üzere 5 ana bileşenden oluşmaktadır.

Tablo 4.1. Komut paketi bileşenleri

Bileşen	Genişlik (bayt)	Açıklama
Başlık	2	Başlık bilgisidir. Sırasıyla 0xFA (250), 0xFB (251) değerlerini içerir.
Bayt sayısı	1	Başlık ve bayt sayısı haricinde paket içindeki bayt sayısıdır.
Komut numarası	1	Robotun yapacağı göreve ilişkin, 0-255 arasında bir değerdir. Örneğin;**move**(ilerle) komutu için bu değer 8, **rotate** (dön) komutu için bu değer 9'dur.
Argüman tipi	1	Argüman tipini gösterir.Argüman pozitif tam sayı ise 0x3B (59), negatif tam sayı ise 0x1B (27), string ise 0x2B (43) .
Argüman	N	Komut tipine göre; 2 byte uzunluğunda bir tam sayı veya string veridir.
Kontrol	2	Sunucu bilgi paketi bütünlük kontrol değeridir.

Aşağıda robota +525 mm ilerle komutu için gönderilen bir komut satırı gösterilmektedir. 250 ve 251 rakamları başlığı oluşturmaktadır. 6 rakamı, kendinden sonra 6 byte uzunluğunda veri olduğunu göstermektedir. 8 rakamı move komutunu, 59 rakamı pozitif tam sayı bir argüman olacağını göstermektedir. 525 mm'yi argüman olarak vermek için Şekil 4.4'deki matlab komutları çalıştırılır.

```
a=dec2hex(525,4);
H=hex2dec(a(1:2));
L=hex2dec(a(3:4));
```

Şekil 4.4. Bir argümana ait H ve L değerlerinin tespiti

Böylece, iki byte'lık komutun düşük byte'ı L=13 olarak ve yüksek byte'ı H=2 olarak elde edilir. Komut satırında argüman değerleri olarak, sırasıyla 13,2 değerleri yazılır. Kontrol değeri sırasıyla (13+8) 21 ve (59+2) 61 olarak hesaplanır.

Komut paketi; 250,251,6,8,59,13,2,21,61 olarak artık seri porttan robota iletilmeye hazırdır.

Bu işlem, yine Matlab da Şekil 4.5'deki komut dizisi ile yapılır;

```
X=[250,251,6,8,59,13,2,21,61];
[n, m]=size(x);
for k=1:m;
    fwrite(serialConnection,x(k));
end;
```

Şekil 4.5. Seri port üzerinden robota komut paketinin iletilmesi

Matlab aracılığı ile seri porttan haberleşme, şekil 4.6'daki komutlar yardımı ile yapılmaktadır;

```
serialConnection = serial('COM1');
set(serialConnection,'BaudRate',9600);
fopen(serialConnection);
```

Şekil 4.6. Seri port üzerinden haberleşme komutlar

Yukarıda belirtilen kontrol komutları gönderilmeden önce, istemci uygulama tarafından robot ile RS232 portundan bir bağlantı kurulması gerekmekte ve bağlantının hemen ardından sunucuya arka arkaya senkronizasyon paketleri (Sync0,Sync1,Sync2) gönderilmek zorundadır. Bu paketler, standart olarak tanımlanmış olup Şekil 4.7'de verilmektedir:

```
SYNC0:  250, 251, 3, 0, 0, 0
SYNC1:  250, 251, 3, 1, 0, 1
SYNC2:  250, 251, 3, 2, 0, 2
```

Şekil 4.7. Senkronizasyon paketleri

Robot senkronizasyon paketlerinin gönderilmesinden sonra, otomatik olarak seriport üzerinden istemciye bilgi paketleri göndermeye başlar. Bu paketler sunucu bilgi paketi (SIP) olarak adlandırılmaktadır. Sunucu bilgi paketleri aracılığı ile robotun durumu, konumu ve tüm algılayıcı değerleri elde edilebilmektedir. SIP paketleri her 100 milisaniyede bir gönderilmektedir.

Tablo 4.2'de sunucu bilgi paketinin bileşenleri liste halinde açıklanmaktadır.

Tablo 4.2. Sunucu bilgi paketi bileşenleri

Bileşen	**Genişlik**	**Açıklama**
Başlık	2 byte	Başlık bilgisidir. Sırasıyla0xFA (250), 0xFB (251) değerlerini içerir.
Bayt sayısı	1 byte	Paket içindeki, başlık ve bayt sayısı haricindeki bayt sayısıdır.
Motor durumu	1 byte	0x3s şeklinde bir bilgidir. Motor durmuşsa s=2 (32=50), robot hareket halinde ise s=3 (33=51)dir.
XPOS	int	Konum: X koordinatı
YPOS	int	Konum: Y koordinatı
THPOS	int	Açısal birim olarak oryantasyon
L VEL	int	Sol tekerlek hızı
R VEL	int	Sağ tekerlek hızı
Batarya	byte	Batarya voltajı.Volt olarak, on katı olarak gösterilir. (örnek124=12,4 volt)
Durmave tampon	uint	16 bitlik bir değerdir. Bit 0 sol tekerlek ani durma durumunu gösterir, 1-7 nolu bitler ise varsa birinci tampon (opsiyonel) bilgilerini içerir. 8 nolu bit sağ tekerlek durma durumunu, 9-15 nolu bitler ise varsa birinci tampon (opsiyonel) bilgilerini içerir.
Kontrol	int	Sunucunun açısal pozisyon servo ayar noktasını derece olarak göstermektedir.
Durum bayrakları	uint	16 bitlik bir veridir. Bit 0:motor durumu, bit 1-4:sonar dizisinin durumu,bit:5-6;stop, bit:7-8:IR sensoru, bit 9 joyistik fire düğmesi, bit 10 batarya şarjının durumu
Pusula	Byte	Elektronik pusula yönü.
Sonar sayısı	byte	Sunucu bilgi paketi içindeki yeni sonar değerlerinin miktarını gösterir. Eğer sonar sayısı>0 ise, sonraki bileşenler ilgili sonar numarası ve sonar değerini içerecektir.
Sonar no	byte	Eğer sonar sayısı>0 ise sonar numarası
Sonar mesafesi	int	Milimetre cinsinden sonar mesafesidir. Aşağıdaki formül ile hesaplanmaktadır. bitor(bitand(c1,hex2dec('FF')),bitshift(c2,8));
…..diğer sonar değerleri		Sonar no ve sonar mesafesi ikilisi olarak, sonar sayısında belirtilen sayıda sonar değeri bu kapsamda paket içinde yer almaktadır.
Kavrama durumu	byte	Robot üzerinde kavrama aleti (gripper) bulunuyorsa durumu
ANPORT	Byte	Seçilmiş analog port numarası
DIGIN	byte	Bayt olarak kodlamış, kullanıcı sayısal girdisi
DIGOUT	byte	Bayt olarak kodlamış, kullanıcı sayısal çıktısı
Batarya	int	Güncel batarya voltajı. Volt olarak, on katı olarak gösterilir. (örnek124=12,4 volt)
Batarya dolma durumu	byte	Otomatik batarya şarj cihazı var ise durumu -1=bilinmiyor,0= şarj etmiyor, 1=yükde,2=aşırı şarj,3=float
Rotvel	int	Rotasyonel hız (saniyede derece x10 olarak)
Hatabayrakları	int	Kullanılmıyor
Kontrol	int	Sunucu bilgi paketi bütünlük control değeri

Bu kapsamda, robottan gelen aşağıdaki örnek paketlerin açılımları Tablo 4.3, 4.4 ve 4.5'de özetlenmektedir. Örneklerde tüm bilgi alanları tespit edilmiş ancak hepsinin anlamlarından ziyade, sadece konuyla ilgili önemli alanların içerikleri açıklanmaktadır.

Örnek veri 1:

250,251,38,50,0,0,0,0,0,0,0,0,0,0,121,0,0,0,0,2,4,0,0,223,1,4,255,0,121,0,0,0,0,
0,0,0,0,128,0,152,246

Örnek veri 2:

250,251,41,50,0,0,0,0,0,0,0,0,0,0,121,0,0,0,0,2,4,0,1,0,136,19,223,1,4,255,0,121,0,0,
0, 0,0,0,0,0,128,0,194,105

Örnek veri 3:

250,251,47,50,57,1,164,252,157,4,2,0,2,0,121,0,0,0,0,3,4,0,3,5,136,19,6,247,6,7,80,3
,223,1,4,255,0,121,0,0,0,0,0,0,0,0,128,0,204,69

Tablo 4.3. Sunucu bilgi paketi çözümlemesi örnek 1

Bileşen	**Genişlik**	**Veri çözümlemesi Örnek 1**	**Veri anlamı**
Başlık	2 byte	250,251	
Bayt sayısı	1 byte	38	uzunluk 38 byte
Motor durumu	1 byte	50	50=0x32 motor durmuş
XPOS	int	0,0	
YPOS	int	0,0	
THPOS	int	0,0,	
L VEL	int	0,0	
R VEL	İnt	0,0	
Batarya	byte	121	12,1 volt
Durmave tampon	uint	0,0	Yok
Kontrol	int	0,0	0
Durum bayrakları	uint	2,4	
Pusula	Byte	0	
Sonar sayısı	byte	0	Bu pakette sonar bilgisi yok.
Sonar no	byte		
Sonar mesafesi	int		
…..diğer sonar değerleri			
Kavrama durumu	byte	223	
ANPORT	Byte	1	
ANALOG	Byte		
DIGIN	byte	4	
DIGOUT	byte	255	
Batarya	İnt	121,0	12,1 volt
Batarya dolma durumu	byte	0	Sarj etmiyor
Rotvel	int	0,0	0
Hata bayrakları	İnt	128,0	
Kontrol	int	152,246	

Tablo 4.4. Sunucu bilgi paketi çözümlemesi örnek 2

Bileşen	**Genişlik**	**Veri çözümlemesi Örnek 1**	**Veri anlamı**
Başlık	2 byte	250,251	
Bayt sayısı	1 byte	41	Uzunluk41 byte
Motor durumu	1 byte	50	Motorlar durmuş
XPOS	int	0,0	
YPOS	int	0,0	
THPOS	int	0,0,	
L VEL	int	0,0	
R VEL	İnt	0,0	
Batarya	byte	121	12,1 volt
Durmave tampon	uint	0,0	
Kontrol	int	0,0	
Durum bayrakları	uint	2,4	
Pusula	Byte	0	
Sonar sayısı	byte	1	Bir adet sonar bilgisi var
Sonar no	byte	0	Sonar no 0,a ait değer
Sonar mesafesi	int	136,19	Sonar değeri 136,19 yani 5000 mm dir.
…..diğer sonar değerleri			Başka sonar değeri yok
Kavrama durumu	byte	223	
ANPORT	Byte	1	
ANALOG	Byte		
DIGIN	byte	4	
DIGOUT	byte	255	
Batarya	İnt	121,0	12,1 volt
Batarya dolma durumu	byte	0	Şarj etmiyor
Rotvel	int	0,0	
Hatabayrakları	İnt	128,0	
Kontrol	int	194,105	

Tablo 4.4'de görüldüğü gibi, örnek 2'de bilgi paketi içinde sonar sayısı değeri 1 adet olarak belirtilmiş ve sonar no değeri ise 0 no'lu sonar olarak bildirilmiştir. Bu tek sonar için, sonar mesafesi sonraki bilgi alanları ile bildirilmiştir. İlgili aritmetik işlem sonrasında 0 nolu sonar için sonar mesafesinin de 5000 mm olarak tespit edildiği görülmektedir.

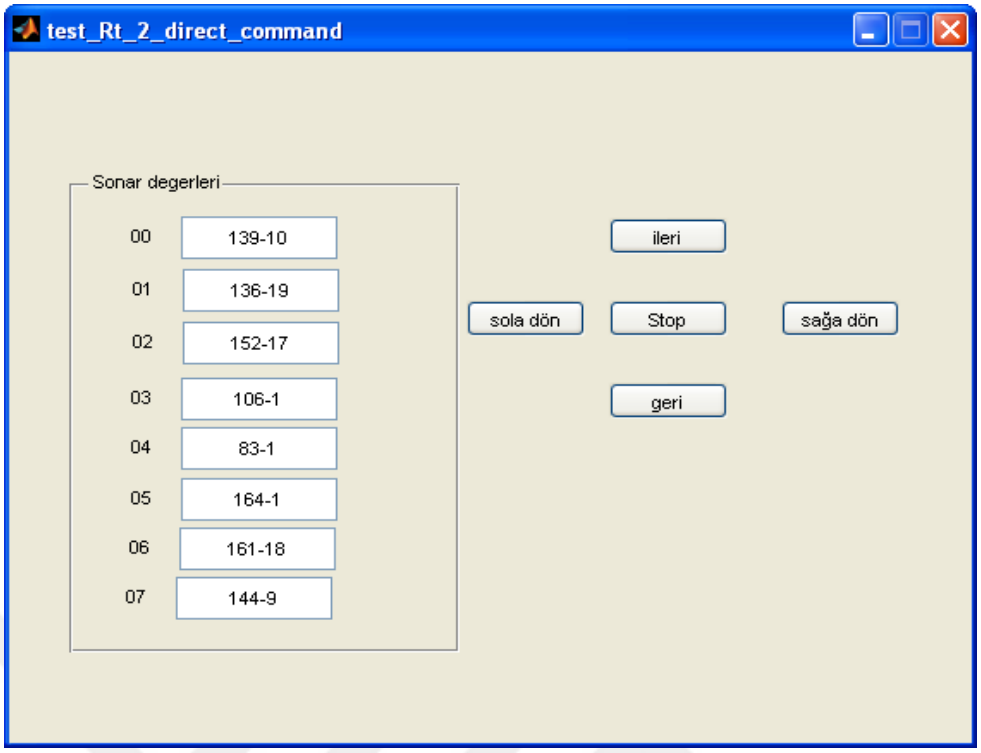
Örnek 2'de tek sonar değerinin olduğu belirtildiği için, bilgi paketindeki diğer bölümler sırasıyla kavrama durumu ile devam etmektedir.

Tablo 4.5'de belirtilen örnek 3'de üç adet sonar bilgisinin bulunduğu ve bunların sırasıyla 5,6 ve 7 nolu sonarlara ait sonar mesafeleri olduğu görülmektedir.

Tablo 4.5. Sunucu bilgi paketi çözümlemesi örnek 3

Bileşen	**Genişlik**	**Veri çözümlemesi Örnek 1**	**Veri anlamı**
Başlık	2 byte	250,251	
Bayt sayısı	1 byte	47	Uzunluk 47 byte
Motor durumu	1 byte	50	Motorlar durmuş
XPOS	int	57,1	
YPOS	int	164,252	
THPOS	int	157,4	
L VEL	int	2,0	
R VEL	İnt	2,0	
Batarya	byte	121	12,1 volt
Durmave tampon	uint	0,0	
Kontrol	int	0,0	
Durum bayrakları	uint	3,4	
Pusula	Byte	0	
Sonar sayısı	byte	3	Üç adet sonar bilgisi var
Sonar no	byte	5	Sonar no 5,a ait değer
Sonar mesafesi	int	136,19	Sonar değeri 136,19 yani5000 mm dir.
Sonar no	byte	6	Sonar no 6,a ait değer
Sonar mesafesi	int	247,6	Sonar değeri247,6 yani1783 mm dir.
Sonar no	byte	7	Sonar no 7,a ait değer
Sonar mesafesi	int	80,3	Sonar değeri 80,3yani 848 mm dir.
…..diğer sonar değerleri			Başka sonar değeri yok
Kavrama durumu	byte	223	
ANPORT	Byte	1	
ANALOG	Byte	4	
DIGIN	byte	4	
DIGOUT	byte	255	
Batarya	İnt	121,0	12,1 volt
Batarya dolma durumu	byte	0	Şarj etmiyor
Rotvel	int	0,0	
Hata bayrakları	İnt	128,0	
Kontrol	Int	204,69	

Tez çalışması kapsamında, MATLAB ortamı kullanılarak, ARCOS direkt komut yöntemi ile, robotun yönetilmesi ve sonar değerlerinin alınması amacıyla bir arayüz yazılımı geliştirildi.



Şekil 4.8. Direkt komutlar ile geliştirilen yazılım arayüzü

Bu arayüz aracılığı (Şekil 4.8) ile basit robot komutları ile robot yönetilebildiği gibi SIP paketleri de düzenli dinlenmekte ve içinden sonar değerleri çekilmektedir. Ekran görüntüsünde, SIP paketlerinden çekilen sonar değerleri tam sayıya çevrilmeden doğrudan gösterilmektedir.

4.2.2. Robotun ARIA Kütüphanesi ile yönetilmesi

Pioneer 3-DX, diğer MobileRobots robotlarında olduğu gibi, bir robot kontrol yazılımı (Advanced Robotics Interface for Applications - ARIA) ile birlikte gelmektedir. ARIA C++ tabanlı açık kaynak kodlu bir yazılım kütüphanesidir. Bu kütüphane kullanılarak ARCOS komut detaylarına girilmeden robota komutlar gönderilmekte ve SIP paketlerine ihtiyaç duyulmadan algılayıcı değerleri elde edilebilmektedir. ARIA'nın bir alt modülü olan ARNetworking TCP/IP tabanlı olarak robotla ağ üzerinden haberleşme görevini yerine getirmektedir [58].

ARIA C++ bazlı bir kütüphane olması nedeniyle, kullanıcıların C++'ı bilmeleri gerekmekte, pointerlar, Class yapıları gibi üst seviye programlama yeteneği istemektedir. Bu çalışmada, akademik dünyada nispeten çok daha yoğun kullanılan MATLAB ortamı tercih edilmiştir. Tez kapsamında, ARIA kütüphanesine ait

komutlar MATLAB üzerinden çalıştırılmıştır. Bu kapsamda, MATLAB'ın bir özelliği olan (MATLAB executable) MEX dosyaları kullanılmıştır. MEX dosyalarını oluşturan her bir komut, bundan sonra, normal bir MATLAB komutu gibi doğrudan MATLAB üzerinden kullanılabilmektedir. ARIA kütüphanesine ait MEX dosyaları MobileRobots firmasına ait internet sitesinden elde edilmiştir [61]. Yeni MEX dosyalarının oluşturulması ile ilgili detaylı bilgiler yine bu siteden elde edilebilmektedir.

ARIA kütüphanesi robota gönderilecek tüm komutları basit bir hale getirmiş ve detaylar ile uğraşılmasını engellemiştir. Basit bir karşılaştırma yapmak amacıyla, gönderilen bazı direkt komutların ARIA kütüphanesi aracılığı ile nasıl gönderilebileceği aşağıdaki tabloda özetlenmektedir.

Tablo 4.6. Direkt komut yöntemi ve ARIA komutlarının karşılaştırması

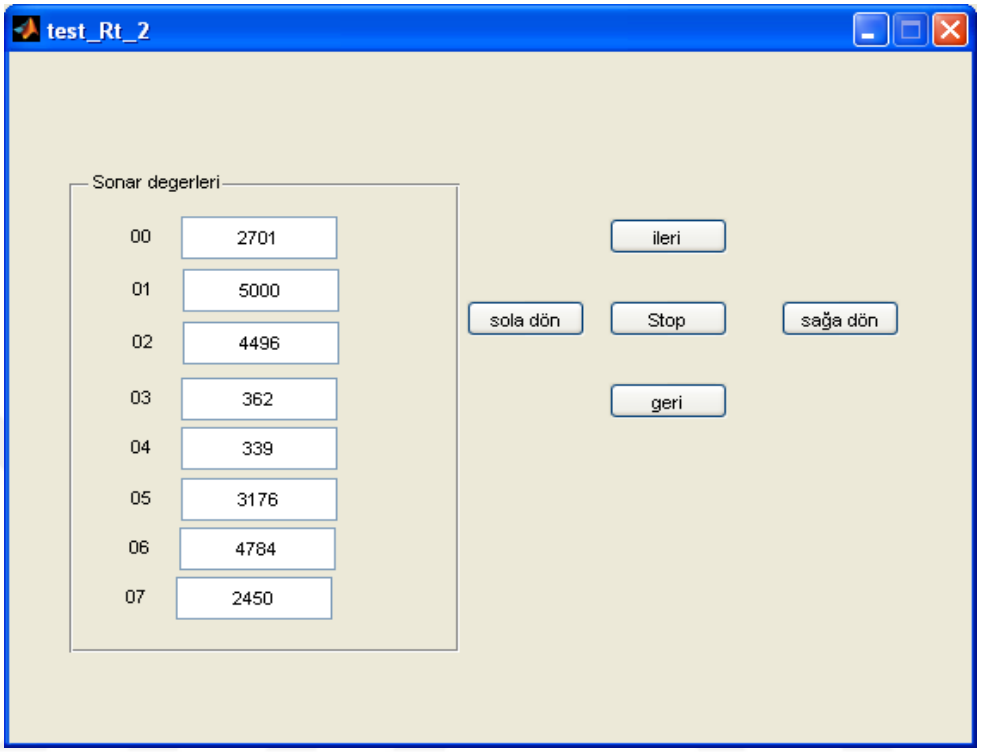
Komut	Direk komut yöntemi ile	ARIA kütüphanesi ile

525 mm ilerle	X=[250,251,6,8,59,13,2,21,61];
	
	[n, m]=size(x);
	
	for k=1:m;
	fwrite(serialConnection,x(k));
	end;
		arrobot_move(525)

1 nolu sonara ait sonar değerini al	veri_uzunlugu = length(SIP);
	
	if(veri_uzunlugu> 23)
	sonar_degeri_sayisi = veri(23);
	for i=1:sonar_degeri_sayisi
	sonarno = veri(24 + (i-1)*3);
	c1=int2str(veri(25+(i-1)*3));
	c2=int2str(veri(26+(i-1)*3));
	sonardegeri = bitor(bitand(
	c1,hex2dec('FF')),bitshift(c2,8));
	display(sonardegeri);
	end;
	end;
		s1 = arrobot_getsonarrange(1);

Karşılaştırma sonuçlarından da anlaşılacağı gibi direkt komut yöntemi ile yapılan karmaşık komut işlemleri, ARIA kütüphanesi aracılığı ile daha sade hale getirilmiştir.

Tez çalışması kapsamında, ARIA ile ünsiyetin kazanılması amacıyla direkt komutlar yardımı ile geliştirilen basit bir yazılım (Şekil 4.9) ayrıca ARIA kütüphanesi yardımı ile de geliştirildi.



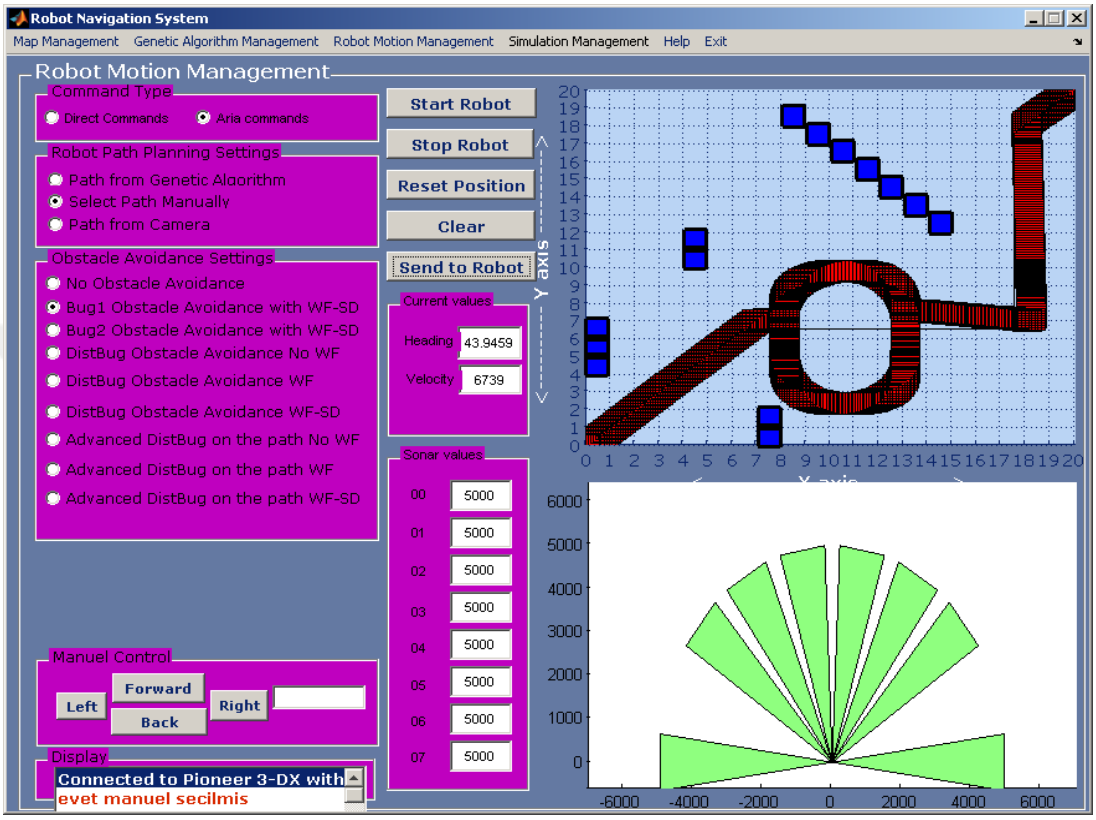
Şekil 4.9. Aria kütüphanesi yardımı ile geliştirilen yazılım arayüzü

Her iki yazılımın da MATLAB ortamında geliştirilmesi nedeniyle, ekran üzerinden hangi yöntemin kullanıldığının anlaşılması nerdeyse imkansızdır. ARIA kütüphanesi, yazılımcıya büyük kolaylık sağlamaktadır. Ancak, kullanılacak ortama ve kullanılacak programlama diline göre ilgili ARIA kütüphanesi dosyalarının önceden hazırlanmış olması gerekmektedir. Halihazırda, ARIA komutları C++ ve Java ortamında bulunmakta, ayrıca MATLAB için de MEX dosyaları olarak bulunmaktadır. Bunlar haricinde bir programlama dili kullanılması durumunda direkt komut yönteminin kullanılması gerekebilecektir.

Ayrıca gömülü sistemler ile uğraşanlar, mikroişlemci üzerinde ARIA kütüphaneleri olamayacağından dolayı, yine direkt komut yöntemini kullanmalıdırlar.

ARIA kütüphanesinin bir diğer avantajı da, robot benzetim ortamı ile tam uyumlu olarak çalışabilmesidir. Sonraki bölümlerde hakkında detaylı bilgi verilecek olan benzetim ortamı, araştırmacı için büyük kolaylık sağlamaktadır.

Bu tez çalışmasında, algoritmaların denenmesi amacıyla geliştirilen arayüzde her iki yöntemin de seçilebilmesine olanak sağlanmaktadır. Şekil 4.10'daki ekran görüntüsünde Command Type bölümünden direkt komut yöntemi veya ARIA komutları seçilebilmektedir.



Şekil 4.10. Algoritmaların denenmesi amacıyla geliştirilen yazılım arayüzü

Tez kapsamında, algoritmalara yoğunlaşmak amacıyla, direkt komut yöntemi sadece robotun manüel kontrolleri aşamasında kullanılmış olup, algoritmaların çalışması ve engelden sakınma gibi yoğun sonar bilgisi gerektiren işlemler için ARIA kütüphanesi zorunlu olarak kullanılmıştır. Ayrıca bu sayede, algoritmaların benzetim ortamında denenebilmesine de olanak sağlanmıştır.

4.3. Robot Benzetim Yazılımı

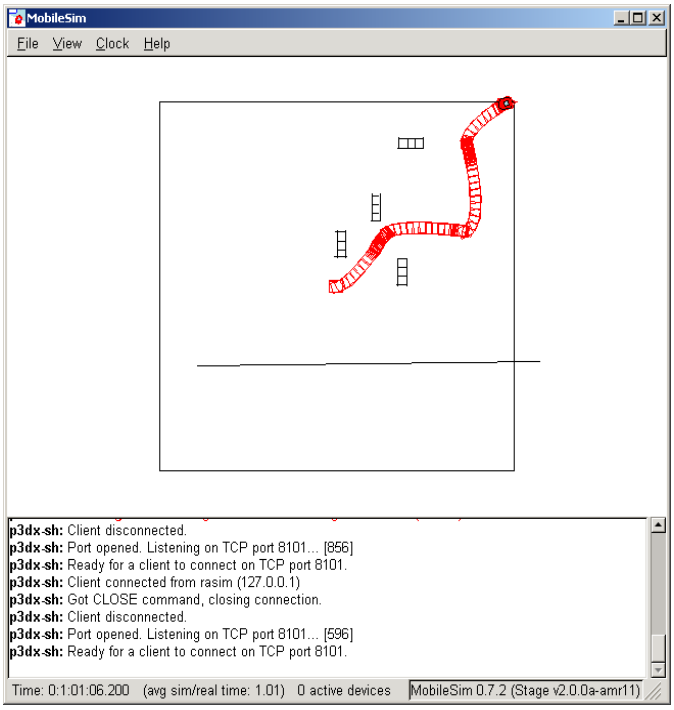
Robotlar ile ilgili yazılım ve algoritma geliştirme aşamalarında benzetim ortamları büyük önem taşımaktadır. Önceden test edilmiş bir algoritmanın, robot üzerinde denenmesi, birçok problemin önceden giderilmesine yardımcı olmaktadır. Tez çalışması kapsamında, MobileSim benzetim ortamından yoğun ölçüde

faydalanılmıştır. Robot seyir sistemi ve kullanılan algoritmalar ilk etapta MobileSim benzetim ortamında test edilmiş, daha sonra robot üzerinde gerçeklenmiştir.

MobileSim, açık kaynak kodlu player/stage projesi temel alınarak, Active Media şirketi tarafından geliştirilmiş olan bir benzetim yazılımıdır [62]. Player/Stage gibi açık kaynak kodludur. Robot kontrol programı Aria ile haberleşebilmektedir. Bu benzetim ortamı kullanılarak da çalışmalar robot olmadan yapılabilmekte, kullanılan algoritma ve programlar değişiklik yapılmaksızın, gerçek robot üzerinde uygulanabilmektedir.

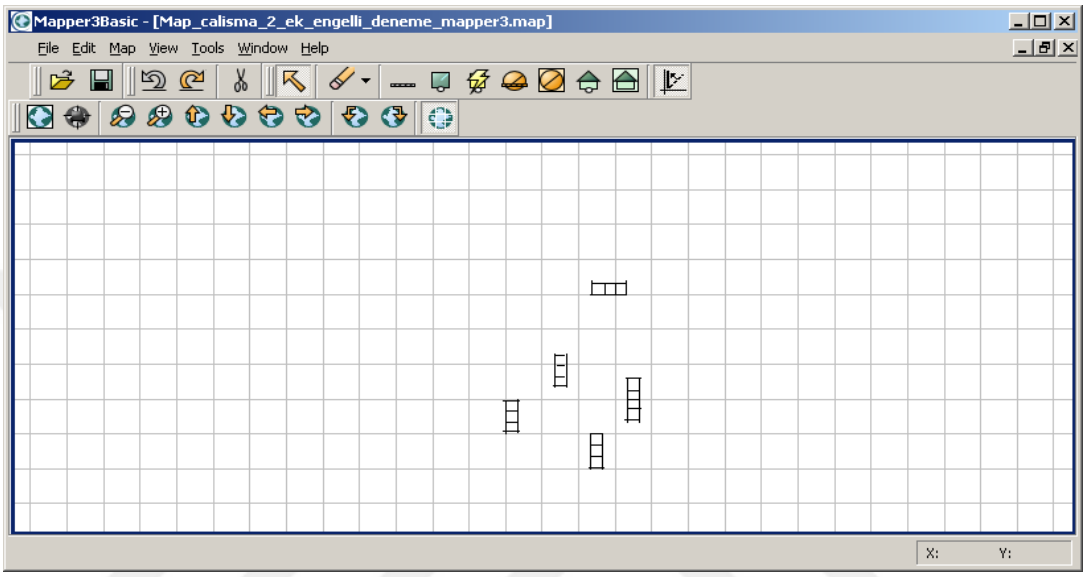
MobileSim yazılımı Pioneer robot ailesi ile tam uyumlu bir benzetim ortamı sunmaktadır. Benzetim esnasında hangi özelliklere sahip bir robot kullanıldığı sisteme tanıtılmaktadır. Kocaeli Üniversitesi Teknoloji Fakültesi laboratuarındaki Pioneer 3-DX'in, sadece ön sonarlarının bulunması gibi, standart özellikleri benzetim başlangıcında sisteme tanıtılmıştır. Böylece, gerçekçi testlerin yapılmasına imkan sağlanmıştır.

Benzetim ortamında, gerçeğe en yakın şekilde ortam oluşturmak amacıyla, MapperBasic uygulaması ile hazırlanmış ortam haritaları kullanılmıştır.



Şekil 4.11. MobileSim benzetim ortamında robotun hareket izleri

Bu sayede, benzetim ortamında sonar değerlerinin alınabileceği engeller tanımlanabilmiş, bunlar üzerinden yol bulma ve engelden sakınma algoritmaları test edilmiştir. Şekil 4.11 ve 4.12'de, MapperBasic yardımı ile hazırlanmış bir harita kullanan MobileSim benzetim ortamında ilerlemiş robotun hareket izleri görülmektedir.



Şekil 4.12. MapperBasic ile harita ve engellerin oluşturulması

MobileSim ve MapperBasic yazılımları MobileRobots firması internet sayfasından indirilebilmekte ve ücretsiz olarak kullanılabilmektedir.

MobileSim benzetim ortamı ile ilgili son bir özelliği vurgulamakta fayda bulunmaktadır. MobileSim ile ARCOS direkt komutları çalışmamakta, sadece ARIA kütüphanesi ile haberleşme sağlanabilmektedir.

4.4. Robot Üstündeki Sonarların Kullanımı

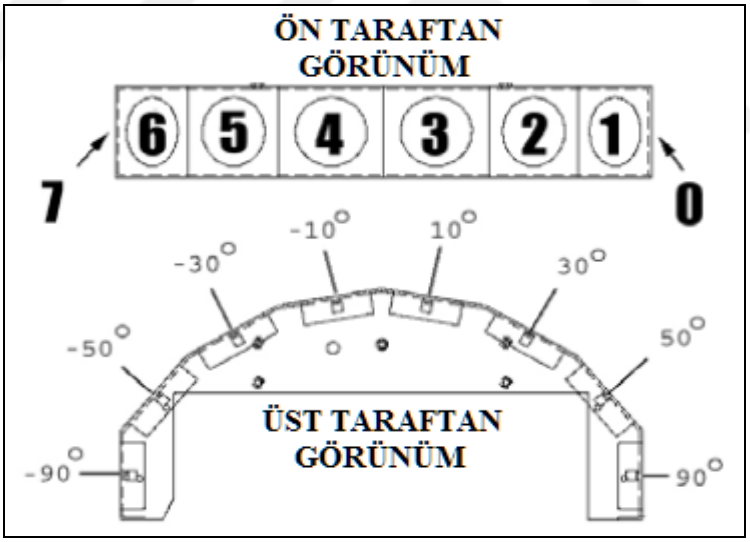
Pioneer robotlarında her birinde 8 adet algılayıcı olan sonar dizileri kullanılmaktadır ve toplamda 4 adede kadar sonar dizisi desteklenmektedir. Normal şartlarda, ön ve arkaya monte edilen 2 adet sonar dizisi ile robot etrafında 360 derecelik bir kapsama alanı elde edilebilmektedir.

Tez çalışması kapsamında kullanılan Pioneer 3-DX robotu üzerinde, önde 1 adet sonar dizisi yani 8 adet sonar algılayıcı bulunmaktadır (Şekil 4.13). Sonarlar soldan itibaren 0 dan 7 ye kadar numaralandırılmış durumdadır. Sonar algılayıcıların birer

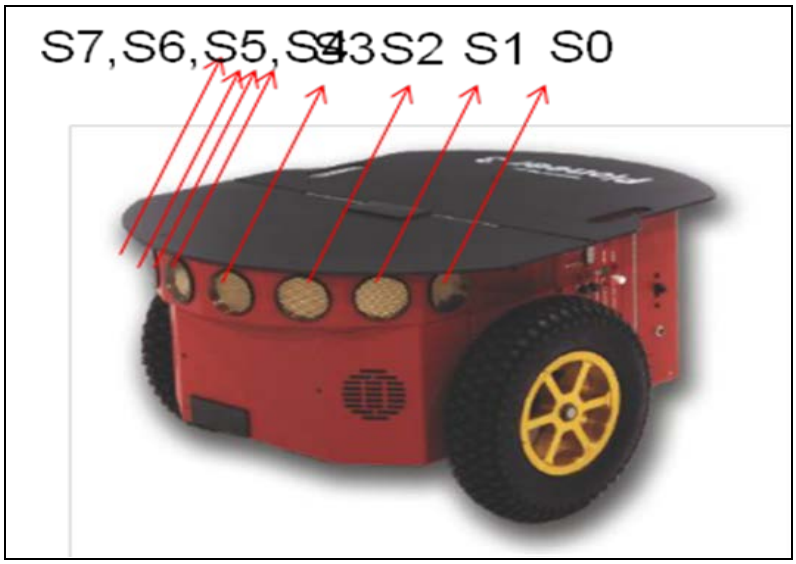
tanesi yanlarda, kalan 6 tanesi ise 20 şer derecelik açılar ile ön tarafa yerleştirilmiş bulunmaktadır (Şekil 4.14). Bu sayede, robotun ön kısmına ait çevrenin 180 derecelik kontrolüne imkan sağlanabilmektedir.

Robot devreye alındığında, aksi belirtilmez ise sonarlar aktif olarak çalışmaya başlamaktadır. İstenirse, direkt komutlar yardımı ile veya ARIA kütüphanesi ile tüm sonarlar veya bir kısmı aktif edilebilir veya kapatılabilir.

Sonarların veri algılama hızı 25 Hz olarak ayarlanmış durumdadır. Bu durumda, her 40 milisaniyede bir algılayıcı ölçüm yapabilmektedir. Hassas ortamlarda, istenirse bu hız ayarlanabilmektedir. Dizi içindeki algılayıcılar soldan sağa olmak kaydıyla her bir turda bir algılayıcı mesafe ölçmektedir. İstenirse bu sıra da değiştirilebilmektedir. SIP paketlerinin kaydedilmesi sonucunda, SIP paketlerinin her 100 milisaniyede bir gönderilmesi ve paket boyut kısıtlamaları nedeniyle, bir SIP paketi içine en fazla 3 adet sonar değerinin girdiği görülmüştür.



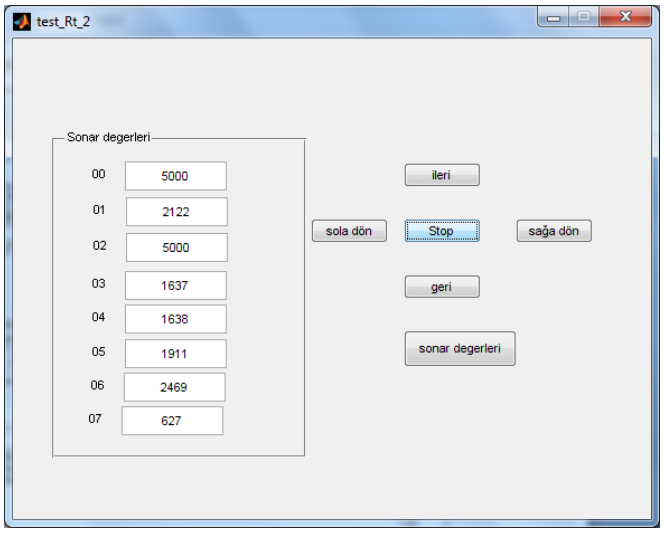
Şekil 4.13. Pioneer 3-DX gezgin robot sonar dizisi [59]



Şekil 4.14. Pioneer 3-Dx üzerindeki sonarlar ve numaraları

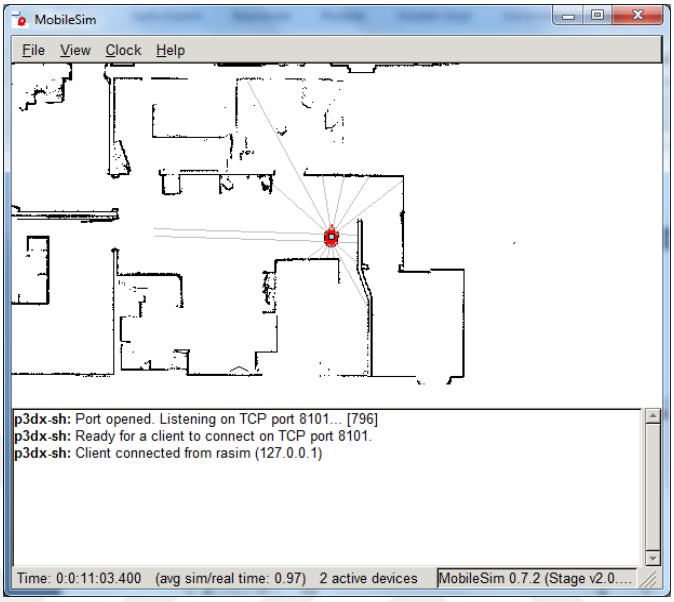
Sonar değerleri robottan mm olarak rapor edilmektedir. Bu kapsamda, maksimum 5 metreye kadar engelleri görebilirken, 17mm'den daha yakın nesneler görülememektedir.

Tez çalışması kapsamında, MATLAB üzerinden robotun yönetilmesi ve sonar değerlerinin alınması amacıyla bir uygulama yazılımı da geliştirilerek, robotun etkin kullanımı ve sonar değerlerinin analizinin yapılması sağlanmıştır.



Şekil 4.15. Benzetim ortamında uygulamanın elde ettiği sonar verileri

Yapılan yazılım gerek benzetim sistemine, gerekse seri porttan gerçek robota bağlanabilmektedir. Aşağıdaki şekilde, yazılımın MobileSim benzetim ortamına bağlandığında elde edilen ekran görüntüleri görülebilmektedir.



Şekil 4.16. MobileSim benzetim programı ekran görüntüsü

Uygulama yardımı ile, robota komutlar gönderilebilmekte ve robottan gelen sonar değerleri ekranda gösterilmektedir. Şekil 4.15 ve 4.16'da görülebileceği gibi, 0 ve 2 nolu sonarlar maksimum mesafede sonar bilgisi almışlardır. 7 nolu sonar en kısa sonar değerini içermektedir.

Sonar değerlerinin saniyede bir kez okunarak güncellenmesi amacıyla, MATLAB "timer" (zamanlayıcı) işlevi kullanılmıştır. MATLAB timer işlevi aracılığı ile, belirli periyotlarda bazı görevlerin thread mantığı ile çalışması sağlanabilmektedir. Aşağıdaki örnekte, MATLAB'da bir timer oluşturmak için gerekli komutlar gösterilmektedir (Şekil 4.17).

```
tmr = timer('Name','refresh_sonar_values_tmr',...
  'Period',1,...  % Update the time every .. seconds.
  'StartDelay',0.5,... % In seconds.
  'TasksToExecute',inf,...  % number of times to update
  'ExecutionMode','fixedSpacing',...%birinci bitince aradan sonra (period) ikinciye basla.
  'TimerFcn',{@refresh_sonar_values,handles});

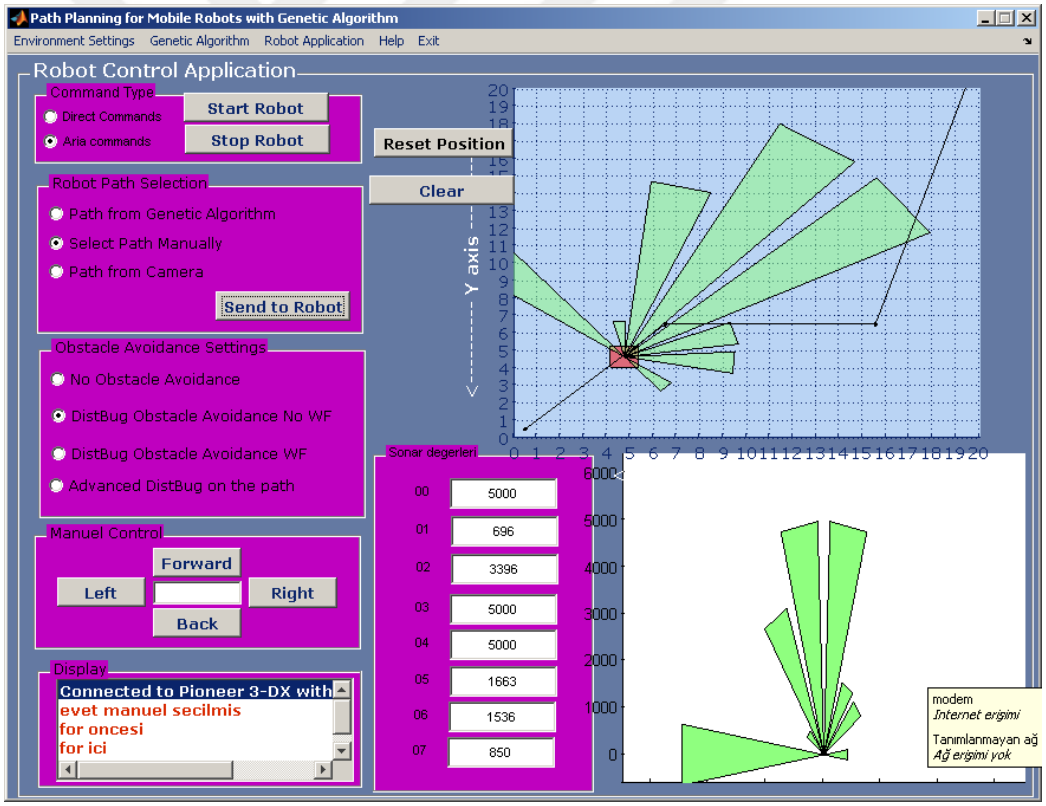
start(tmr);
```

Şekil 4.17. MATLAB'da timer oluşturma

Bu örnekte, refresh_sonar_values adlı fonksiyonunun birer saniye aralıklarla çalıştırılması için gerekli komutlar görülmektedir. Timer tanımlandıktan sonra, start komutu ile birlikte düzenli olarak çalışmaya başlamaktadır. Timer işlevinin kullanımı ile ilgili daha detaylı bilgi [63]'den elde edilebilir.

Tez çalışmasının başlangıcında, sonar değerleri sadece rakam ile ekranda gösterilirken, ilerleyen aşamalarda, sonar çizgileri şeklinde de gösterilmeye başlanmış (Şekil 4.18), tanımlanmış olan bu zamanlayıcı ile bu işlev yerine getirilmiştir.

Benzetim ortamındaki testlerin ardından, geliştirilen yazılım gerçek robot üzerinde de çalıştırılmış ve sorunsuz olarak robota kumanda edilebildiği ve sonar değerlerinin görülebildiği tespit edilmiştir.



Şekil 4.18. Yazılım üzerinde sonar bilgilerinin gösterimi

Sonardan gelen değerlerin doğruluğunu test etmek ve özellikle sonar numaralarına göre sonar bilgilerinin tutarlılığının tespiti için bazı fiziki engeller ile testler yapılmıştır. Gerçek robot ve yazılım kullanarak, robotun sol tarafındaki 0 numaralı

sonarın gerçek engelleri algılama testi Şekil 4.19'da, robotun önündeki 3 ve 4 numaralı sonarın gerçek engelleri algılama testi Şekil 4.20'da görülmektedir.



Şekil 4.19. Sol taraftaki 0 numaralı sonarın gerçek engelleri algılama testi



Şekil 4.20. Öndeki 3 ve 4 numaralı sonarların gerçek engelleri algılama testi

Yapılan testler sonucunda, yazılımın gerçek robot üzerinde de istediğimiz seviyede sonar değerlerini gösterebildiğini, milimetre cinsinden gelen sonar değerlerinin fiziki uzaklıklar ile uyuştuğu görülmüştür. Bu kapsamda, daha detaylı testlerin yapılmasına gerek duyulmamıştır.

Tez çalışması kapsamında, engelden sakınma algoritmalarının denenebilmesi amacıyla, sonar değeri döndürebilecek engeller olarak karton kutular kullanılmıştır (Şekil 4.21).



Şekil 4.21. Gerçek engel olarak kutuların kullanılması

5. ROBOT SEYİR SİSTEMİ VE TEST EDİLEN ALGORİTMALAR

Otonom bir mobil robotun hedefe ilerleyerek, kendisine verilen görevi yerine getirebilmesi için, sağlıklı bir yol planlaması yaparak en kısa yoldan hedefe ilerlemesi ve bu ilerleme esnasında yeni engeller ortaya çıkması durumunda aktif engelden sakınma yapabilmelidir.

Yol bulma ve engelden sakınma algoritmalarının birbirleriyle uyumlu bir şekilde çalışması robotun görevi açısından son derece önemlidir. Tez çalışması kapsamında;

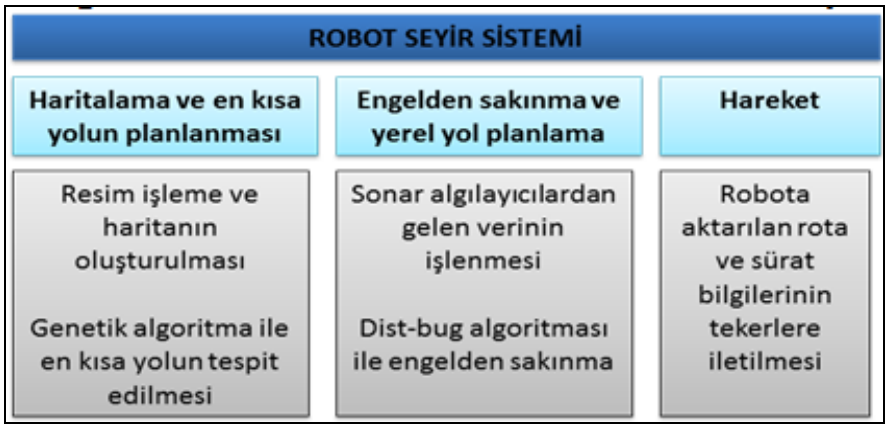
Mobil robotun yol bulma probleminin çözümünde genetik algoritmalar kullanıldı ve MATLAB'da kodlandı.

Engelden sakınma amacıyla, Böcek-1, Böcek-2 ve Dist-bug algoritmaları yazılımsal olarak gerçeklendi, benzetim ortamında ve gerçek robot üzerinde denenerek performansları incelendi,

Böcek algoritmalarında yapılan iyileştirmeler ile ilgili karşılaştırmalar yapıldı,

Yol bulma ve engelden sakınma aşamalarının tümleşik çalışması ele alınarak performans katkısı incelendi.

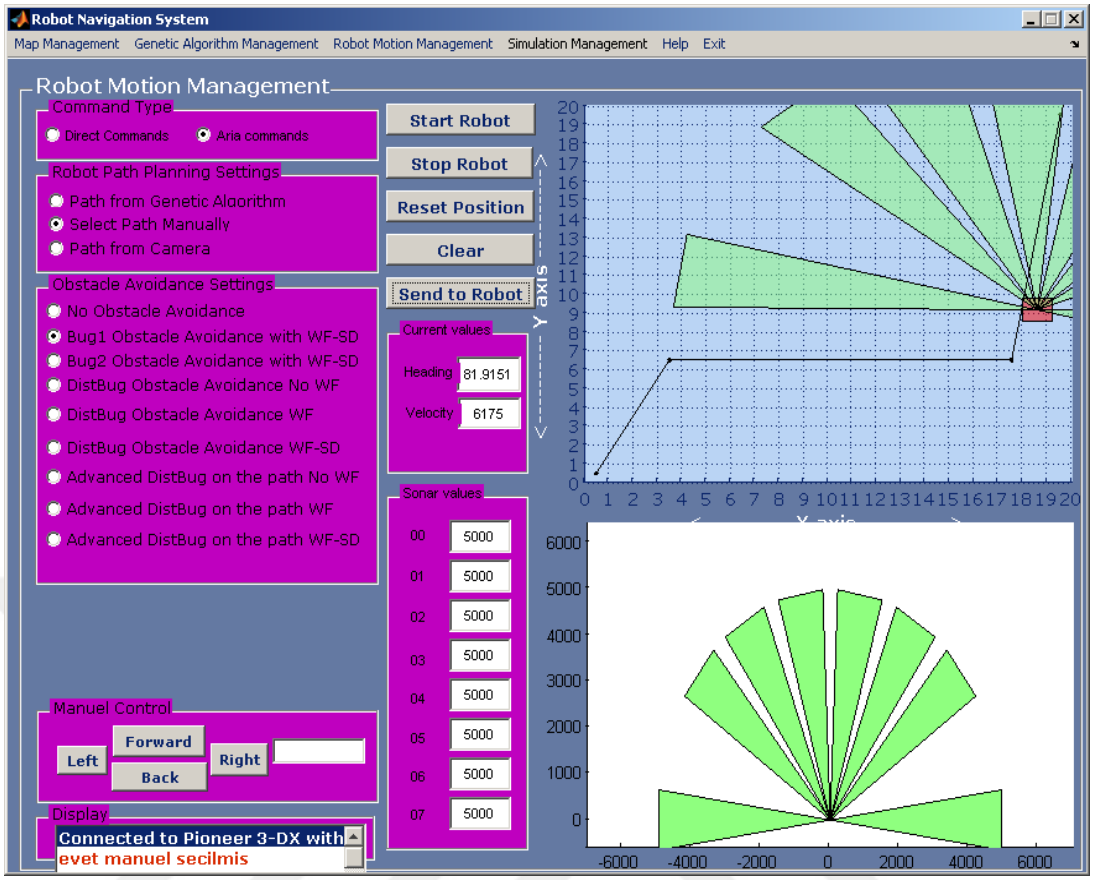
Tüm bu yapılan çalışmaların, tümleşik bir şekilde yazılımsal olarak gerçekleştirilebilmesi amacıyla, tez çalışması kapsamında, Şekil 5.1'de gösterilen bir robot seyir sistemi oluşturuldu. Bu sistemde özetle; iç mekan tavanına yerleştirilen bir kamera aracılığı ile ortamın resmi alınarak, görüntü işleme teknikleri de kullanılarak ortam haritası oluşturuldu ve ortamdaki engeller tespit edildi. Genetik algoritmalar ile, engelli bir ortamda hedefe ulaşmak için, engele çarpmayan en kısa yol planlanarak, rota üzerindeki ara düğüm noktaları tespit edildi. Bir sonraki aşamada, robotun dinamik engellerin bulunduğu bir ortamda sonar algılayıcılar yardımı ile engelden sakınması sağlandı. Son aşamada ise, engelden sakınma ve hedefe ilerleme amacıyla rota ve sürat bilgileri robota iletildi.



Şekil 5.1. Tasarlanan robot seyir sistemi yapısı

Konuyla ilgili ayrıca, Şekil 5.2'de ekran görüntüsü verilen bir kullanıcı arayüzü geliştirildi. Bu arayüz aracılığı ile ortam bilgileri ve yol bulma algoritması parametreleri değiştirilebilmekte, robottan gelen sonar bilgileri kullanıcıya gösterilebilmekte ve seçilen engelden sakınma yöntemine göre robotun en uygun şekilde hedefe ilerlemesi sağlanmaktadır. Robotun ilerlemesi ile ilgili bilgiler yine bu ekranda görüntülenerek, kullanıcının robotu en etkin bir şekilde takip ve kontrol edebilmesine imkan tanındı. Yazılım, MATLAB ortamında, ARIA kütüphanelerinin etkin kullanımı ile geliştirildi.

Engelden sakınma algoritmaları, MobileSim benzetim ortamında denenmiş, ayrıca Pioneer 3-DX robot üzerinde gerçeklenmiştir. Benzetim ortamında yapılan çalışmalar ve denenen senaryolar esnasındaki ekran görüntüleri ayrıca video olarak kayıt altına alınmıştır. Bu amaçla CamStudio yazılımından faydalanılmıştır. Camstudio genel kamu lisansı (GPL-General Public Licence) ile ücretsiz olarak [64]' den indirilebilmektedir.



Şekil 5.2. Geliştirilmiş arayüze ait ekran görüntüsü

5.1. Laboratuarda Oluşturulan Test Ortamı

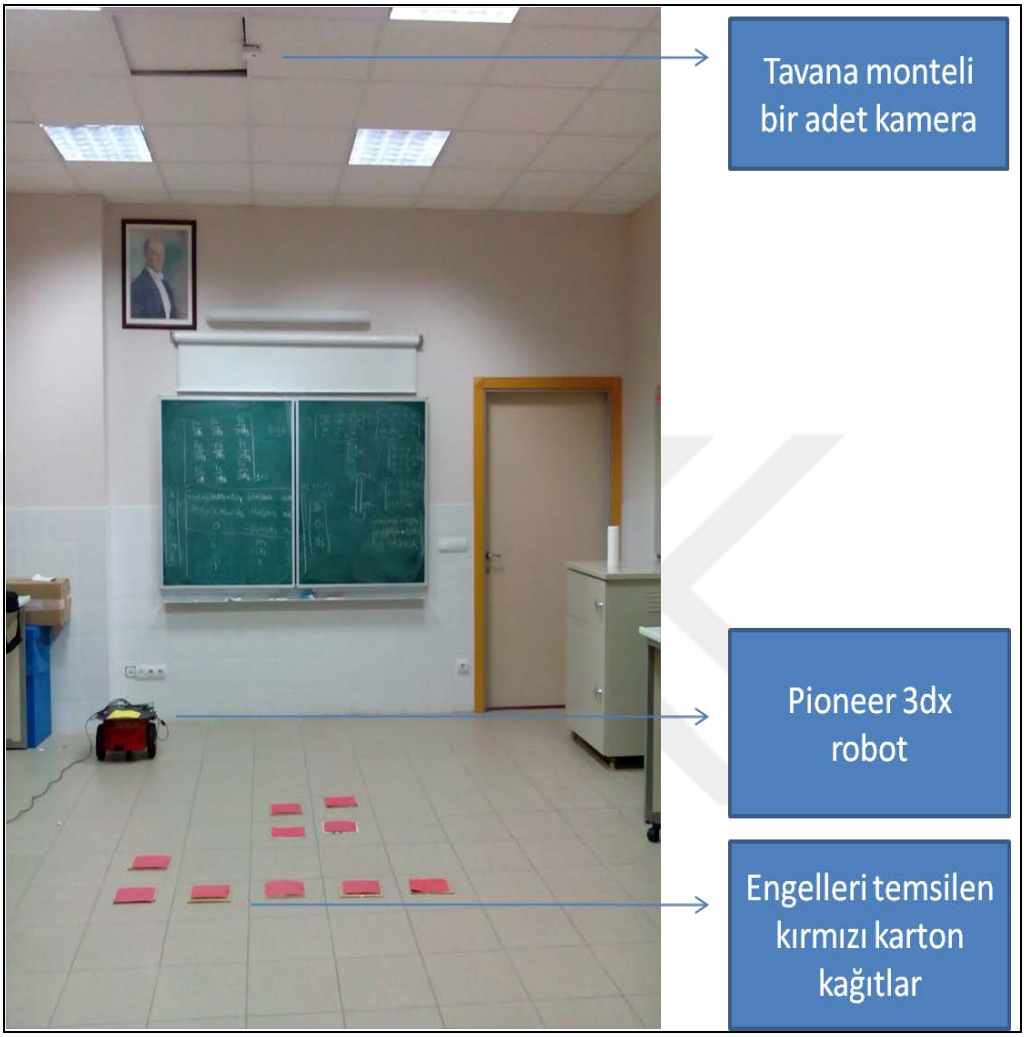
Bu bölümde, robot seyir sisteminin detayları hakkında bilgi verilecek, engelden sakınma algoritmalarının deneysel sonuçlarından alınan değerlerin karşılaştırması yapılacaktır.

Gerek engelden sakınma algoritmalarının robot üzerinde gerçeklenmesi, gerekse tez kapsamındaki çalışmaların gerçekçi bir senaryo üzerinde denenebilmesi amacıyla, Kocaeli Üniversitesi Teknoloji Fakültesi laboratuarında bir test ortamı düzenlendi.

Bu ortamda, tavana bir IP kamera yerleştirilip, laboratuarın yukarıdan görüntüsü elde edilerek ortam haritasının oluşturulmasına imkan sağlandı. Oluşturulan harita sayesinde yol bulma algoritmaları denendi.

Elde edilen görüntü üzerinden, laboratuar içindeki engellerin ve robotun bulunduğu yerin tespit edilmesi maksadıyla renkli kartonlar kullanıldı. Bu kapsamda, kırmızı

renkli kartonlar engelleri, sarı renkli karton ise robotu ifade edecek şekilde sistem yapılandırıldı. Şekil 5.3'de, laboratuardaki test ortamı görülmektedir.



Şekil 5.3. Laboratuardaki test ortamı

Çalışmada, ızgara tabanlı haritalama yöntemi tercih edilerek, laboratuarın ve yerdeki karoların büyüklüğü göz önüne alınarak 10 x 8 lik bir harita oluşturuldu. Her bir gridin yerdeki karolar ile örtüşmesi sağlandı.



Şekil 5.4. Laboratuarda engeller

Laboratuarda ayrıca, engelden sakınma algoritmalarının denenebilmesi amacıyla, sonar değeri döndürebilecek engeller olarak kutular kullanıldı (Şekil 5.4). Özellikle engelden sakınma ile ilgili testlerde, bu kutulardan faydalanıldı. Kutuların üzerine konulan kırmızı kartonlar aracılığı ile, tavan kamerasından alınan görüntülerle engellerin istenildiğinde tespit edilebilmesi sağlandı.

5.2. Görüntü İşleme Teknikleriyle Ortam Haritasının Tespiti

Yol bulma problemlerinde en önemli nokta bir haritanın oluşturulmasıdır. Oluşturulan haritada robot ve hedef noktaları bulunmalı, var ise engeller konumlandırılmış olmalıdır. Çalışmada, test ortamındaki tavan kamerası aracılığı ile laboratuarın yukardan görüntüsü elde edilerek ortam haritasının oluşturulması sağlandı. Elde edilen görüntü üzerinden, laboratuar içindeki engellerin ve robotun bulunduğu yerin tespit edilmesi maksadıyla renkli kartonlar kullanıldı. Bu kapsamda kırmızı renkli kartonlar engelleri, sarı renkli karton ise robotu ifade edecek şekilde sistem yapılandırıldı.

Harita oluşturmada ilk adım tavandaki kameradan zemin fotoğrafının çekilmesidir (Şekil 5.5).

Sonraki adımda, bu fotoğraf, görüntü işleme yöntemleri ile işlenerek, matris şeklinde bir harita bilgisine dönüştürülmektedir. Elde edilen matris üç boyutlu bir matristir. Bu matriste ilk iki boyut piksel koordinatı iken, üçüncü boyut RGB (red, green, blue)

değeridir. MATLAB'in zengin görüntü işleme kütüphaneleri yardımı ile birçok işlem kolaylıkla yapılabilmektedir.



Şekil 5.5. Tavan kamerası ile alınan zemin fotoğrafı

Tavandaki kameradan MATLAB yardımı ile görüntün alınması için imread komutu kullanıldı. Ayrıca, kamera görüntüsü içinde gereksiz alanların çıkartılması amacıyla, imcrop komutu yardımı ile resimdeki gereksiz dış alanlar temizlendi. Ardından resim boyutu 320x240 boyutuna ayarlandı.

Şekil 5.6'da bu kapsamda kullanılan MATLAB komutları görülmektedir.

```
img=imread('http://192.168.1.100/snapshot.jpg');%görütntünün alınması
img=imcrop(img,[62 42 512 384]);%gereksiz alanların kırpılması
img=imresize(img,0.63);%boyutun 320x240 a ayarlanması
```

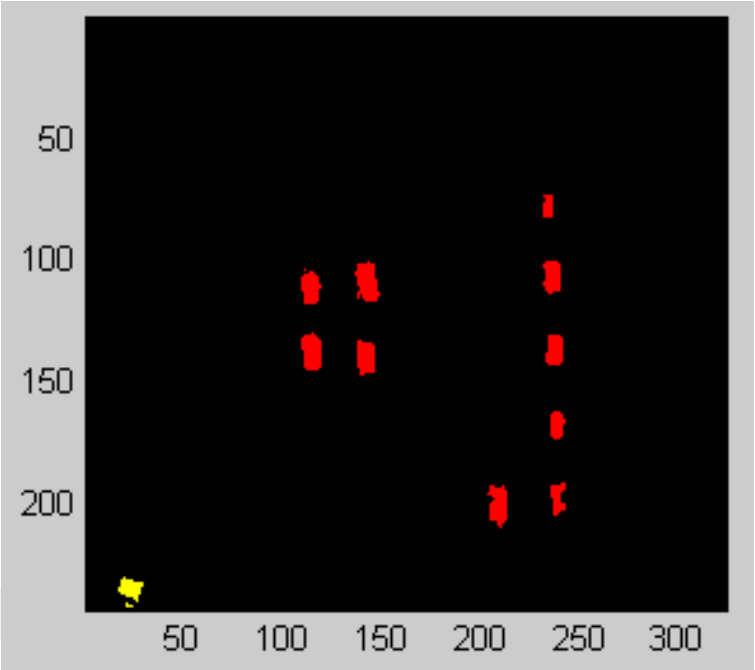
Şekil 5.6. Görüntü işleme ile ilgili kullanılan MATLAB komutları



Şekil 5.7. Temizlenmiş zemin fotoğrafı

Şekil 5.7'da zemin fotoğrafının kırpılarak, gereksiz bölgelerin temizlenmiş hali görülmektedir.

Bir sonraki adımda, fotoğraftaki renklerin yoğunlaştığı pikseller tespit edilip, bu piksellerin çalışma alanında hangi hücreye ait olduğu hesaplanmaktadır. Bu amaçla, öncelikle resim üzerinde kullanılan kırmızı ve sarı renklerin RGB aralıkları sisteme tanımlanmıştır. Ardından bir döngü ile resimdeki tüm koordinatların RGB değerleri, tespit ettiğimiz kırmızı ve sarı RGB değer aralığı ile karşılaştırılarak, resim üzerindeki anlamlı alanların gruplanması sağlanmış, bu sayede resimdeki gürültü/kirlilik gibi istenmeyen nesnelerin resimden temizlenmesi sağlanmıştır.



Şekil 5.8. Fotoğraftaki diğer bilgilerin süzülmüş hali

Şekil 5.8'de zemin fotoğrafının RGB değerleri kontrol edilerek, kırmızı ve sarı renkler haricindeki bölgelerden temizlenmiş hali görülmektedir. Görüntü üzerindeki kırmızı ve sarı tonları, laboratuarın aydınlık ve ışık durumuna göre değişkenlik göstermektedir. Bu nedenle çalışmalar öncesinde, kalibrasyon amaçlı olarak kırmızı ve sarı renklerin RGB değerleri tespit edilmiştir. Şekil 5.9'da çalışma kapsamında kullanılan kırmızı ve sarı renklerin RGB değerleri belirtilmiştir.

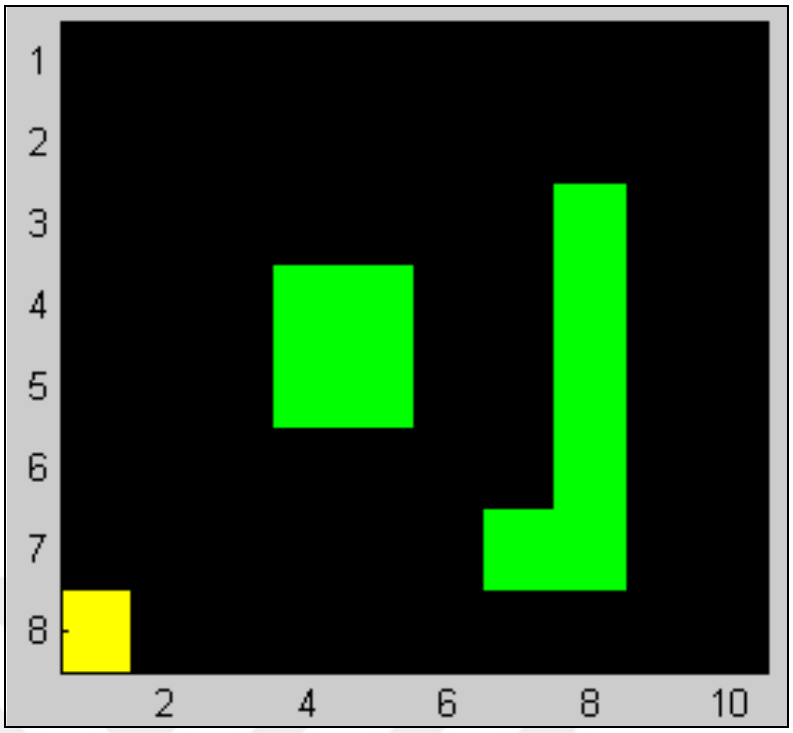
```
renk_kontrol_sari=[205,200,120]; %sari renk RGB
renk_kontrol_kirmizi=[195,85,100]; %kirmizi renk RGB
```

Şekil 5.9. Çalışmada kullanılan kırmızı ve sarı renklerin RGB değerleri

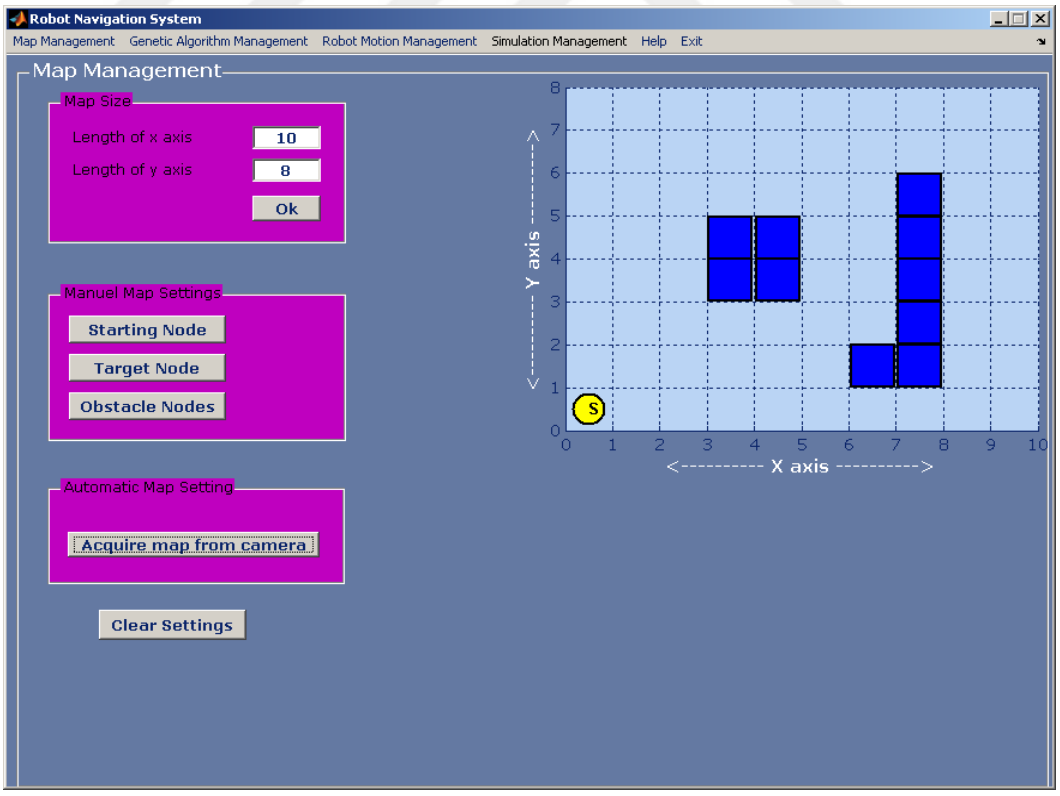
Fotoğraf üzerindeki her bir koordinattaki RGB değerleri, bu değerler ile kontrol edilmiş, bu değerlerin +20 ve -20 aralığı içinde olan RGB değerlerinin kırmızı veya sarı olduğu varsayılmıştır.

Sonraki adımda ise, pikseller, 10x8 harita üzerindeki ilgili karelere göre ayarlanmıştır (Şekil 5.10).

Son adımda, elde edilen 10x8'lik bir matris bilgisi, MATLAB üzerinde haritada kullanıcıya gösterilmektedir (Şekil 5.11).



Şekil 5.10. Fotoğraftaki engel ve robotun piksellerinin tespit edilmesi



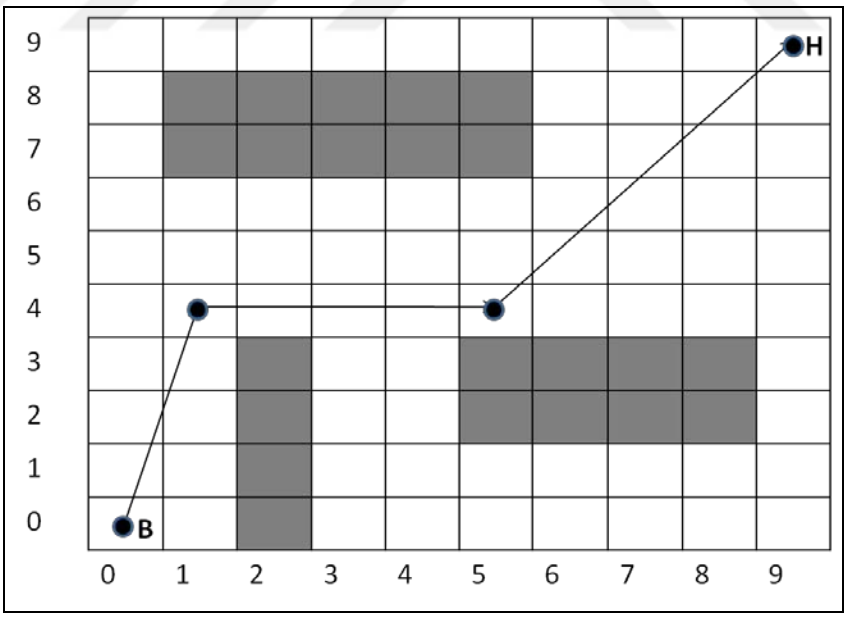
Şekil 5.11. Elde edilen 10x8 lik matrisin uygulama üzerinde gösterimi

5.3. Genetik Algoritmanın Yol Bulma Maksatlı Kullanımı

Tez çalışması kapsamında, ortamdan elde edilen harita bilgisi kullanılarak, genetik algoritma yardımı ile yol bulma problemi çözülmüştür. Bu bölümde, genetik algoritmanın yazılımsal olarak gerçeklenmesi kapsamında yapılan işlem adımları ele alınmaktadır.

5.3.1. Başlangıç nüfusunun tespiti

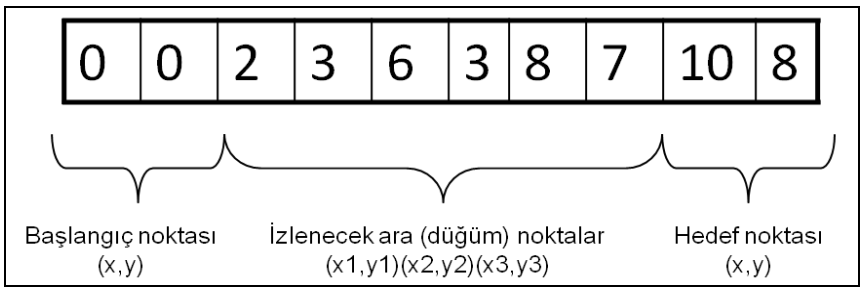
Doğadaki tüm canlıların temelini oluşturan hücrelerin yapıtaşı olan genler ve kromozomlar, genetik algoritmadaki popülasyona ait kromozomlar olarak modellenmektedir. Genetik algoritmada kromozomlar çözüm uzayı içindeki çözüm vektörlerinin kodlanmış halleridir [51]. Çözüm uzayı olarak ızgara tabanlı hücresel bir harita üzerindeki engellerin bulunduğu bir ortamda, başlangıç noktasından hedef noktasına ilerlemek için gerekli yol güzergahları genetik algoritma tarafından seçilmektedir. Şekil 5.12'de örnek bir çalışma alanı görülmektedir. Çalışma alanında gri hücreler engelleri ifade etmekte, boş hücreler, robotun kullanabileceği hücreleri göstermektedir.



Şekil 5.12. Çalışma alanı

Belirtilen çalışma alanı içinde kromozomlarımız, başlangıç noktasından, hedef noktasına ilerlerken kullanılacak düğüm noktalarını tanımlamaktadır. Bu kapsamda ikili, seri veya tam sayı kodlama yöntemleri kullanılabilmektedir. Tez çalışması kapsamında tam sayı kodlama yöntemi kullanıldı. Şekil 5.13'de tam sayı olarak,

başlangıç noktasından hedef noktasına ilerleme aşamasındaki düğüm noktalarını içeren bir tam sayı kromozom örneği görülmektedir.



Şekil 5.13. Tam sayı kromozom örneği

Başlangıç nüfusu tespit etme işlemi esnasında iki adet girdi bulunmaktadır. Bunlar istenen düğüm sayısı ve başlangıç nüfusu büyüklüğüdür. Başlangıç nüfusu bu iki değerin kullanıldığı iç içe döngüler yardımı ile doldurulmuş, kromozomlar rastgele seçilmiş ancak, düğüm noktalarında engellerin olmaması sağlanmıştır (Şekil 5.14).

```
for i=1:baslangic_nufusu_buyuklugu
    j=1;
    while(j~=dugum_sayisi+1)
        temp = 0;
        x = fix(rand(1,1)*harita_boyut_x)+1;
        y = fix(rand(1,1)* harita_boyut_y)+1;
        if(engel_varmi(x,y)==0)
            baslangic_nufus_matrisi(i,j)=x;
            baslangic_nufus_matrisi (i,j+1)=y;
            j=j+2;
        end
    end
end
```

Şekil 5.14. Başlangıç nüfusunun doldurulması

5.3.2. Amaç fonksiyon değerinin tespit edilmesi

Çalışmada amaç fonksiyon, engellere çarpmadan başlangıç ile hedef arasındaki en kısa yolu bulmak olarak tanımlanmıştır. Denklem (5.1) ve Denklem (5.2) çalışmada kullanılan amaç fonksiyonunu göstermektedir.

$$f = \begin{cases} \sum d(p_i, p_{i+1}) & \text{, engel yoksa} \\ \sum d(p_i, p_{i+1}) + \sum_{j=1}^{m} \text{ceza} & \text{, engel varsa} \end{cases} \quad (5.1)$$

$$d(p_i, p_{i+1}) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (5.2)$$

Burada; f amaç fonksiyonu, p kromozomdaki genleri, n kromozomdaki gen sayısını, d iki nokta arasındaki mesafeyi, m iki nokta arasındaki engel sayısını ifade etmektedir. İki nokta arasında herhangi bir engel var ise, amaç fonksiyona ceza puanı eklenmektedir. En iyi birey amaç fonksiyon değeri en küçük olan birey olarak tanımlandığı için, engel üzerinden geçen birey cezalandırılarak seçilme olasılığı azaltılmaktadır.

5.3.3. Optimizasyon kriterinin kontrolü

Bu adımda, optimizasyon kriteri kontrol edilmekte ve algoritma bitirilmekte veya yeni bireylerin üretimine devam edilmektedir. Çalışma kapsamında optimizasyon kriteri olarak jenerasyon sayısı kullanılmaktadır. Algoritma, jenerasyon sayısı kadar çalıştırılmakta ve en iyi değerlere sahip çözüm, olası en iyi çözüm olarak algoritma sonlandırılmaktadır.

5.3.4. Yeni bireylerin seçilmesi

Genetik algoritma, doğadaki güçlü bireyin özelliklerinin yeni kuşaklara aktarılması prensibini esas almaktadır. Nüfus içerisindeki bireylerin hangilerinin ve ne derecede iyi olduklarını tespit etmek amacıyla üç aşamalı bir seçim mekanizması kullanılmıştır. İlk aşamada, tüm bireylerin ne derecede iyi olduklarını gösteren amaç fonksiyon değerleri bulunmakta, ikinci aşamada, bireylere, amaç fonksiyon değerlerine bağlı olarak uygunluk değerleri atanmakta, son aşamada ise, bireyler uygunluk değerlerine göre seçilerek, yeni bireyler üretmek üzere eşleştirme havuzuna atılmaktadırlar [51].

Tez çalışmasında, amaç fonksiyonunda elde edilen değerler paralelinde, her bir bireye, amaç fonksiyon değeri sıralamasına göre eşit aralıklı uygunluk değeri atanmıştır. Buna göre, en kötü amaç fonksiyon değerli birey 1 puan alırken en iyi amaç değerli birey 10 puan olarak hesaplanmıştır.

Bireyler, uygunluk değerlerine göre, rulet tekerleği seçim yöntemi ile seçilerek eşleştirme havuzuna atılmışlardır. Rastlantısal bir yöntem olan rulet tekerleği seçiminde, bireylerin seçilme olasılıkları sürekli bir çizgi üzerine parçalar halinde yerleştirilirler. Parçaların toplam uzunluk aralığında rastgele bir sayı üretilerek, bu sayı hangi parça üzerine düşüyor ise o parçaya sahip birey seçilmiş olur [51].

Tablo 5.1'de, 10 bireyden oluşan, maksimum uygunluk değeri 10 olan bireylerin seçilme olasılıkları gösterilmektedir. Tabloda yer alan seçilme olasılıkları Denklem (5.3) ile belirlenmiştir [59].

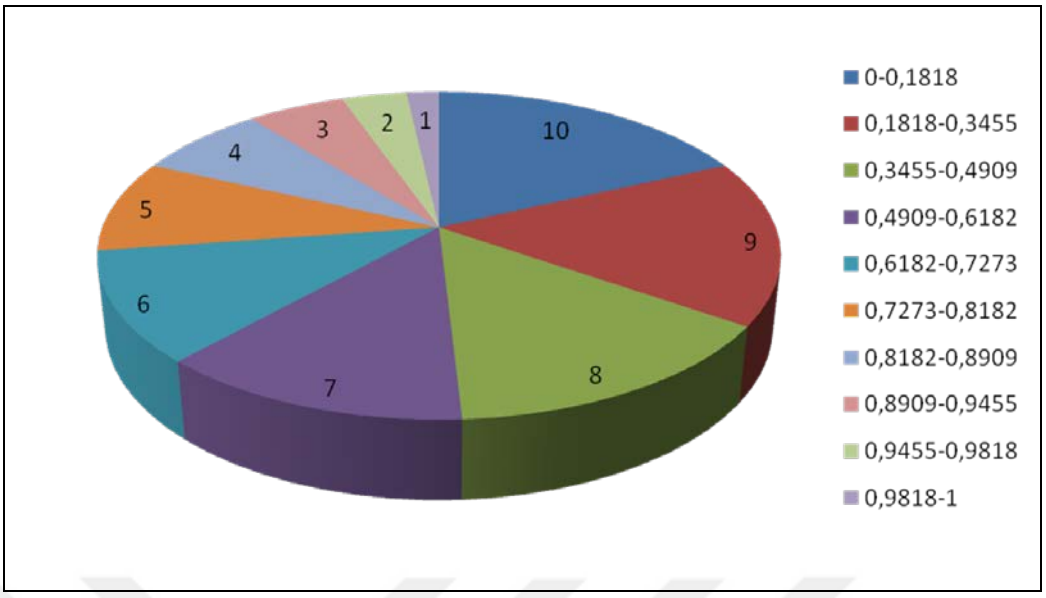
$$S_i = \frac{U_i}{\sum U} \tag{5.3}$$

Burada; S_i birey i'nin seçilme olasılığını, U_i birey i'nin uygunluk değerini, $\sum U$ nüfus içerisindeki bireylerin toplam uygunluk değerlerini göstermektedir.

Tablo 5.1. Rulet tekerleği seçilme olasılıkları.

Uygunluk değeri	10	9	8	7	6	5	4	3	2	1
Seçilme olasılıkları	0,1818	0,1636	0,1455	0,1273	0,1091	0,0909	0,0727	0,0545	0,0364	0,0182
Seçilme aralığı	0-0,1818	0,1818-0,3455	0,3455-0,4909	0,4909-0,6182	0,6182-0,7273	0,7273-0,8182	,8182-0,8909	0,8909-0,9455	0,9455-0,9818	0,9818-1

Tabloya göre, uygunluk değeri 10 olan birey, en kısa yola sahip ve nüfus içindeki diğer bireylere göre en uygun birey olması nedeniyle, rulet tekerleği kuralına göre en büyük aralığa (0-0,1818) sahip olarak, seçilme olasılığı en yüksek bireydir. Bu değerlerin nasıl kullanıldığına bakılacak olursa, rastgele 10 adet sayı seçildiğinde, bu 10 sayının büyük çoğunluğunun 0 ila 0,1818 arasında olabileceği, ve hiç veya çok az miktarda da 0,9818-1 arasında olabileceği gösterilmektedir.



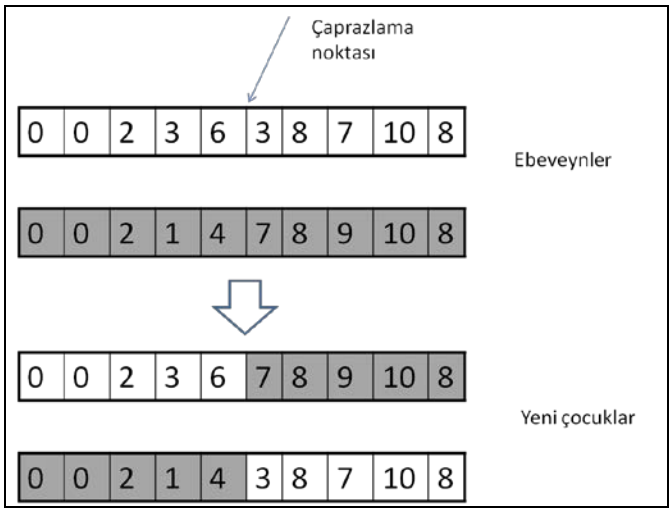
Şekil 5.15. Seçilme aralıklarının rulet tekerleği şekline gösterimi

Şekil 5.15'de seçilme olasılıkları bir rulet tekerleği üzerinde görülmektedir. Uygunluk değeri 10 olan birey, uygunluk değeri 1 olan bireye nazaran daha yüksek seçilme olasılığına sahiptir.

5.3.5. Çaprazlama

Genetik algoritmada, ebeveyn bireylerin iyi olan genlerinin aynı çocuk bireyde birleştirilerek, ebeveynlerden daha iyi bir birey üretilebilmesi amacıyla çaprazlama işlemi yapılmakta, bu amaçla ebeveyn birey çiftlerinin karşılıklı parçaları yer değiştirilerek çocuk bireyler üretilmektedir. Kaç tane parçanın değiştirileceğini belirleyen çaprazlama noktası sayısı, genellikle birkaçı geçmemekte, ayrıca çaprazlamaya tabi tutulacak olan ebeveyn bireyler nüfus içerisinden rastgele seçilmektedir. Bununla birlikte, ebeveyn bireyleri sağ ve sol iki parçaya ayıran çaprazlama noktası da rastgele belirlenmektedir [51].

Tez çalışması kapsamında, tam sayı kromozomlar kullanılması nedeniyle çaprazlamada tam sayı çaprazlama operatörü, rastgele ve tek noktalı olarak kullanıldı. Şekil 5.16'da ebeveynlerden çaprazlama ile yeni çocuk bireylerin oluşturulması görülmektedir.



Şekil 5.16. Çaprazlama ile ebeveynlerden yeni çocuk bireylerin üretilmesi

5.3.6. Mutasyon

Genetik algoritmada, başlangıç nüfusundaki bireyler, çaprazlama yöntemi ile yeni bireylere dönüştürülmektedir. Ancak benzer bireyler üzerine çözümün yoğunlaşması, çözümün hiç bulunamaması veya hatalı çözüme gitmesi gibi durumları beraberinde getirebilecektir.

Mutasyon operatörünün işlevi, bireylerin birbirlerine benzemeye başlamasını engellemek için kullanılan bir yöntemdir. Bu işlevde, bireylerin genlerini düşük bir olasılıkla rastgele değiştirme işlemi yapılmaktadır. Genlerin, ne kadarının değişeceği 1 ile 0 arasında değişen mutasyon oranı ile belirlenmektedir. 1 olduğunda nüfus içindeki bireylerin tamamı değişeme uğrarken, 0 olduğunda hiçbir birey değişime uğramayacaktır [51].

Çalışma kapsamında 0,01 gibi küçük mutasyon değerleri kullanılmış, bununla birlikte, kullanıcı tarafından bu değerin değiştirilebilmesine imkan tanınmıştır.

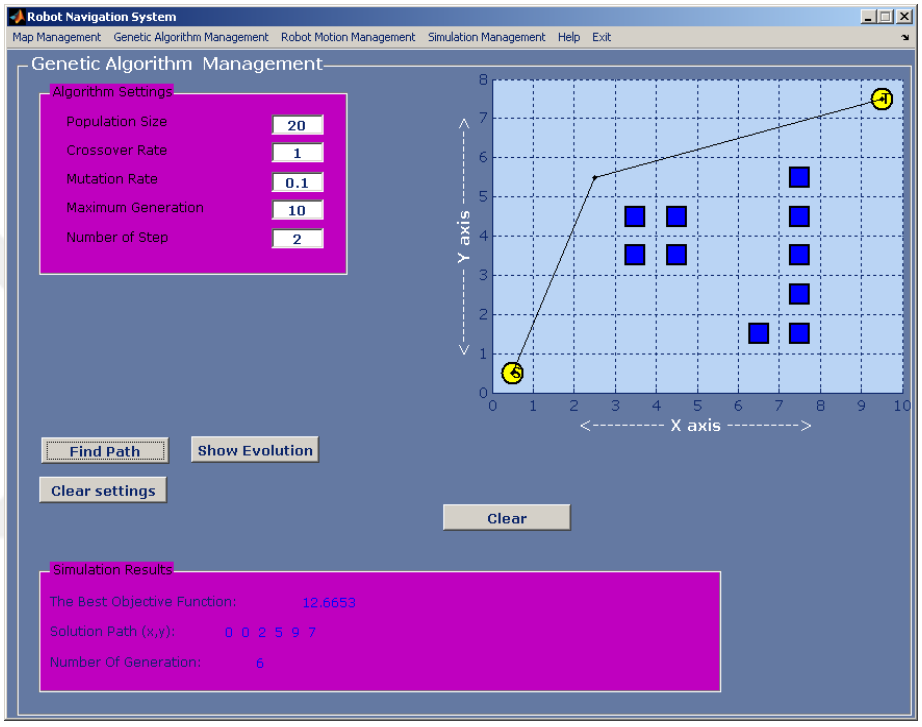
5.3.7. Elitizm

Elitizm, çocuk bireylerin oluşturulması aşamasında, iyi bireylerin sonraki kuşaklara aktarılması amacıyla kullanılmaktadır. Aksi halde, sonraki kuşaklar arasındaki bireylerden hiç biri bir önceki kuşaktaki en iyi bireyin sağladığı uygunluk değerine ulaşamayabilir. Tez çalışması kapsamında, eski kuşaklardan en iyi bireyin bir sonraki kuşaktaki en kötü bireyle değiştirilmesi yoluyla elitizm uygulanmıştır.

5.3.8. Genetik algoritma parametrelerinin değişimi

Genetik algoritmanın çalışması için, başlangıç nüfusu, çaprazlama ve mutasyon oranlarının tespit edilmiş olması ayrıca algoritmanın kaç jenerasyon çalışacağının belirlenmiş olması gerekmektedir.

Çalışma kapsamında bu değerlerin yol bulma probleminin çözümüne etkilerinin incelenmesi amacıyla bir görsel arayüz hazırlanmıştır.



Şekil 5.17. Genetik algoritma parametre giriş ekranı

Şekil 5.17'de, kamera yardımı ile alınan ortam fotoğrafı üzerinden oluşturulan haritada, genetik algoritma ile bulunan bir yol görülmektedir.

Bu ekran ve parametreler kullanılarak, çeşitli testler yapılmış, laboratuar ortamında 10x8 lik bir harita üzerinde, benzetim üzerinde ise 20x20 lik haritalarda, yol bulma probleminin genetik algoritma ile başarılı bir şekilde çözümü sağlanmıştır.

5.4. Robotun Düğüm Noktaları Üzerinden Hedefe İlerlemesi

Robotun elindeki bir harita yardımı ile tespit edilen düğüm noktaları üzerinden hedef noktaya gidişi standart bir algoritma gibi görülebilir. Ancak bu konuda da çeşitli yaklaşımlar bulunmaktadır.

5.4.1. Sadece rota ve mesafe bilgisiyle tek komutla ilerleme

Bu yaklaşımlardan belki en basiti, robot ile gidilmesi gereken düğüm noktası arasındaki mesafenin ve rotanın tespit edilerek, robota sadece belirtilen rotada belirtilen mesafe kadar git şeklinde tek bir komutun verilmesidir. Şekil 5.18'de bu algoritmadaki ana adımlar gösterilmektedir.

```
For (rota üzerindeki her bir adım için){
    hedef = yeni_düğüm_noktası;
    adım_hedefe_varildi_mi=0;
    Yeni_rota = Hedef_acisi_hesapla();
    Gidilecek_mesafe = Hedefe_mesafeyi_hesapla();
    Rotayı_değiştir(yeni_rota);
    İlerle(gidilecek_mesafe);
    While (ilerleme_bittimi () == false ) {
        Bekle;
    }
}
Exit();
```

Şekil 5.18. Sadece rota ve mesafe bilgisi ile ilerleme algoritması

Bu yaklaşımda, her bir düğüm noktası için rota ve mesafe hesap edilmekte ve robotun istenilen rotada belirtilen mesafe kadar ilerlemesi istenmektedir. Bu ilerleme esnasında ilerlemenin bittiği kontrol edilmekte ve ilerleme tamamlandı ise, bir sonraki düğüme geçilmektedir.

Burada kullanılan ilerleme_bittimi() fonksiyonu ilerlemenin tamamlanıp tamamlanmadığını tespit etmektedir. Pioneer 3-DX robotta, ilerle (move) komutu, istenilen mesafe kadar ilerleme tamamlanınca robotun hızını sıfıra getirmekte ve bu sayede ilerlemenin bittiği anlaşılmaktadır.

Basit komutların kullanıldığı, hızlı bir algoritma gibi görünse de robotun gerçek rotadan kayması durumunda, hata oranı kümülatif olarak artabilecektir. Ayrıca, olası bir dinamik engelin çıkması durumunda çarpışma kaçınılmaz olabilecektir.

Algoritmanın dinamik engeller için modifiye edilmesi de son derece zordur. Tek seçenek, yol bulma algoritmasının yeniden çalıştırılarak, yeni düğüm noktalarının tespit edilmesidir.

5.4.2. Dinamik olarak rota ve süratin ayarlanması

Bu yöntemde robot her düğüm noktasına ilerlerken, sürekli olarak hedefin konumunu kontrol etmekte, buna göre varsa gerekli rota düzeltmelerini dinamik olarak yapmaktadır.

```
For (rota üzerindeki her bir adım için){
    hedef = yeni_düğüm_noktası;
    adım_hedefe_varildi_mi=0;
    While (adım_hedefe_varildi_mi==0){
        hedefe_mesafe_hesapla();
        if (hedefe_mesafe < 150) {
            adim_hedef_varildi_mi=1;
            yeni_adima_gec;
        }
    Yeni_rota = Hedef_acisi_hesapla();
    Rotayı_değiştir(yeni_rota);
    Sürat_ayarla(ilerleme_sürati);
    }
}
Sürat_ayarla(0);
Exit();
```

Şekil 5.19. Dinamik rota ve sürat ayarlaması ile ilerleme algoritması

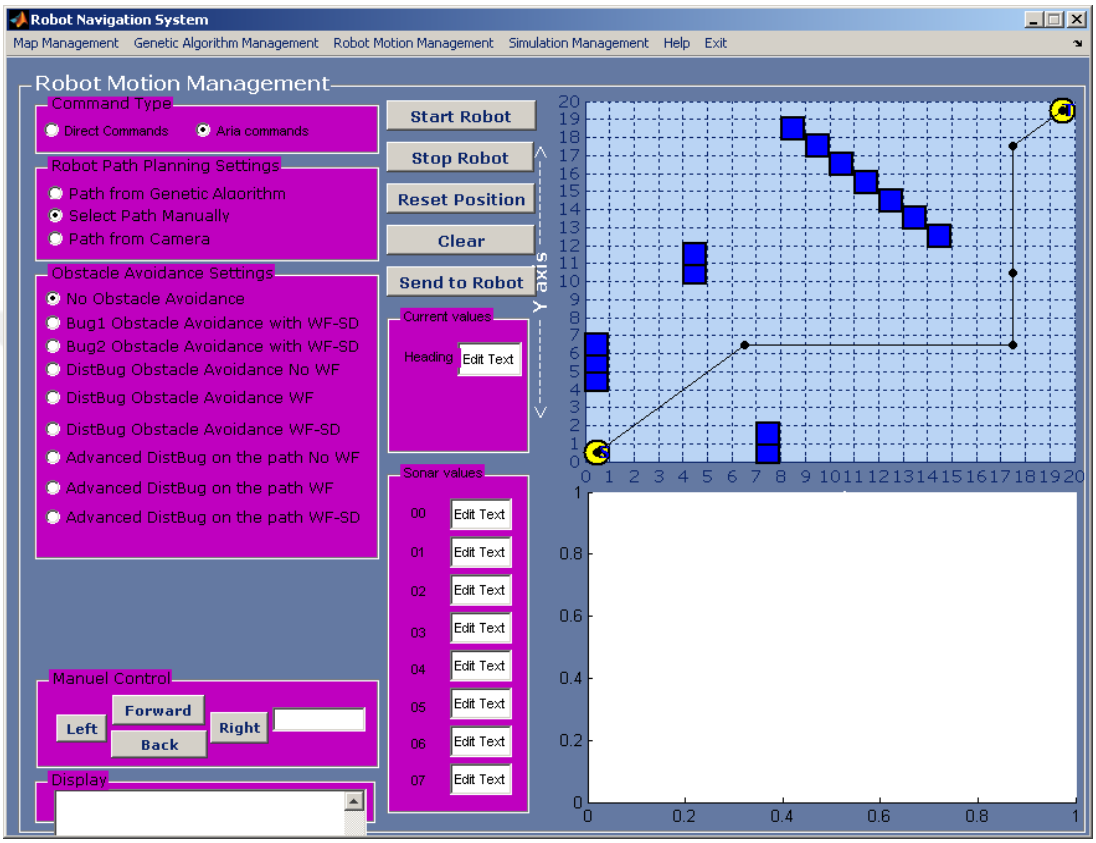
Bu yöntemde, her düğüm için dinamik bir şekilde, hedef ve robot arasındaki açı tespit edilerek, robota gerekli rota ayarı yaptırılmakta ve ilerleme süratinde gitmesi sağlanmaktadır (Şekil 5.19).

Sürekli olarak hedef ve robot arasındaki açının tespit edilmesi işlemci için küçük bir yük getirmektedir, ama bu hesaplama esnasında robotun ilerlemeye devam etmesi nedeniyle, zaman kaybı olmamakta ve işlemciye yüklenen yük gözardı edilebilmektedir.

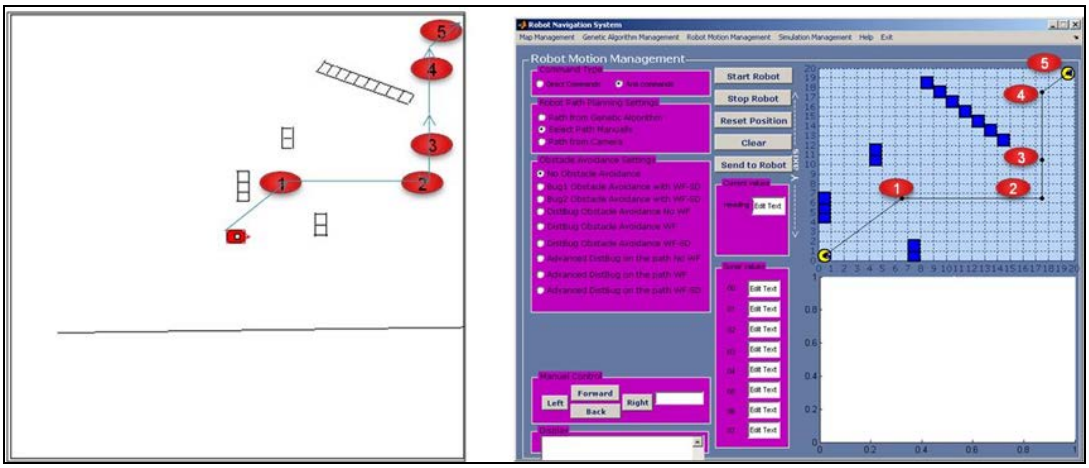
Ayrıca, ortaya çıkabilecek dinamik engel durumunda, algoritma kolaylıkla değiştirilebilmekte ve duvar_takip gibi özel bir algoritma rahatlıkla adapte edilebilmektedir.

5.4.3. Robot hareketinin robot seyir sisteminde gerçeklenmesi

Yol bulma problemi genetik algoritma ile çözülmüş ve tespit edilen düğümler üzerinden robotun hedef noktaya ilerlemesi benzetim ortamında ve gerçek robot üzerinde denendi ve başarı ile çalıştığı görüldü.



Şekil 5.20. Robot seyir sistemi ile oluşturulan yol güzergahı

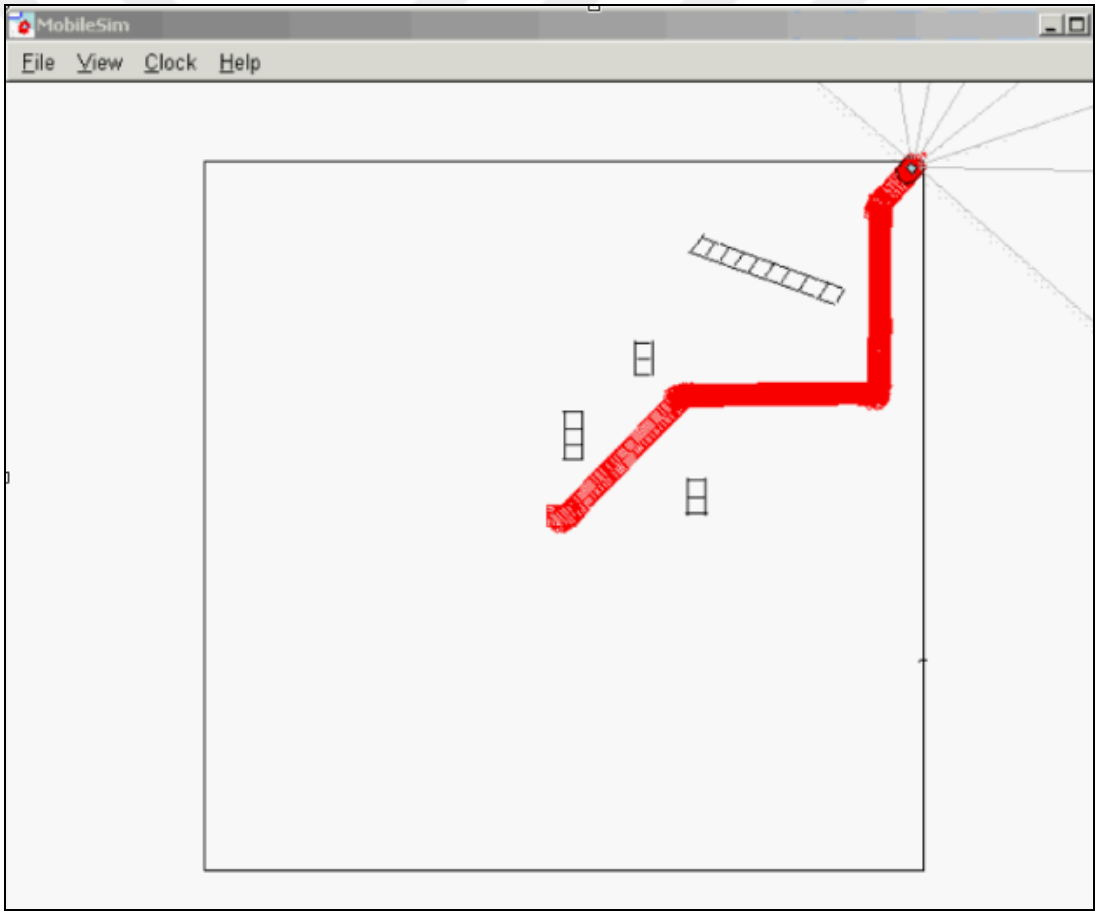


Şekil 5.21. MobileSim ve robot seyir sistemi üzerinde düğüm noktaları

Şekil 5.20'de,robot seyir sistemi ile engellerin olduğu bir ortam oluşturuldu ve yol bulma algoritması ile bir yol güzergahı belirlendi.

Bu güzergah beş düğüm noktasından oluşmaktadır. Robot kendisine verilen görev gereği düğüm noktaları üzerinden güzergahı takip edecektir. Şekil 5.21'de MobileSim ve robot seyir sistemi üzerinde düğüm noktaları görülmektedir.

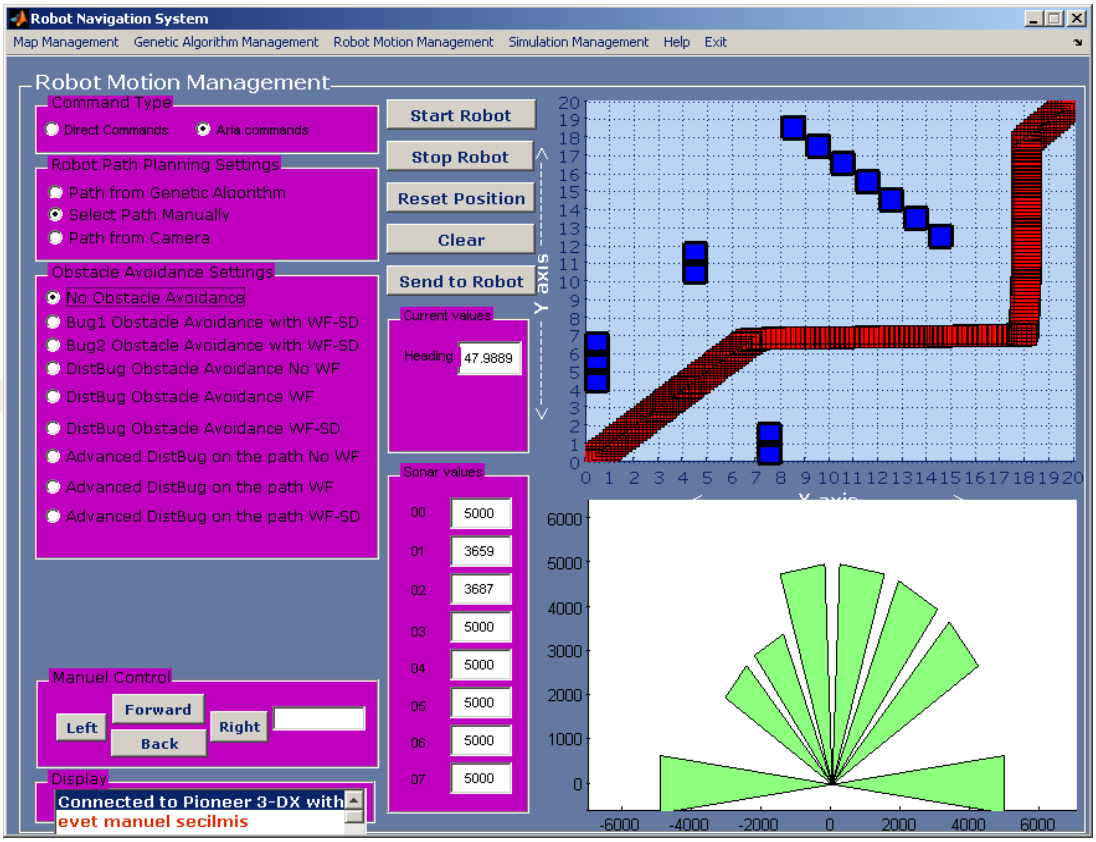
MobileSim benzetim ortamında algoritmanın çalıştırılması sonucunda, robotun başarı ile belirtilen güzergah boyunca her bir düğüm noktasını takip ederek hedefe ulaştığı görüldü (Şekil 5.22). Aynı güzergahın robot seyir sistemi üzerinde gösterimi ise Şekil 5.23'de verilmektedir.



Şekil 5.22. MobileSim üzerinde robotun ilerleyişi

Gerek MobileSim benzetim yazılımı üzerinden gerekse robot seyir sistemi üzerinden robotun hareketleri takip edildi ve başarı ile kendisine verilen güzergah üzerinde hedefe vardığı görüldü.

Bu senaryo, ayrıca laboratuar ortamında Pioneer 3-DX robot üzerinde, 10x8 lik bir alanda denendi ve robotun da başarı ile verilen güzergahta istenilen şekilde ilerlediği görüldü.



Şekil 5.23. Robot seyir sisteminde robotun ilerleyişi

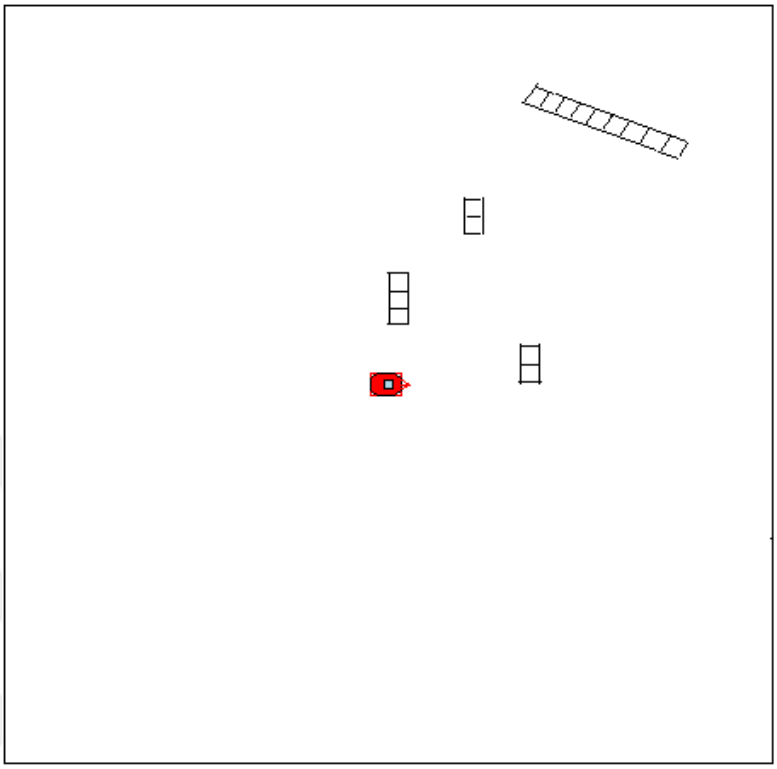
5.4.4. Engelden sakınma yeteneği olmadan hedefe ilerleme

Dinamik engellerin olmadığı bir ortamda, yapılan yol planlamasına istinaden oluşturulmuş düğüm noktalarını takip etmek suretiyle, hedefe ilerleme algoritmasının başarılı bir şekilde çalıştığı daha önceki senaryolarda görülmüştü.

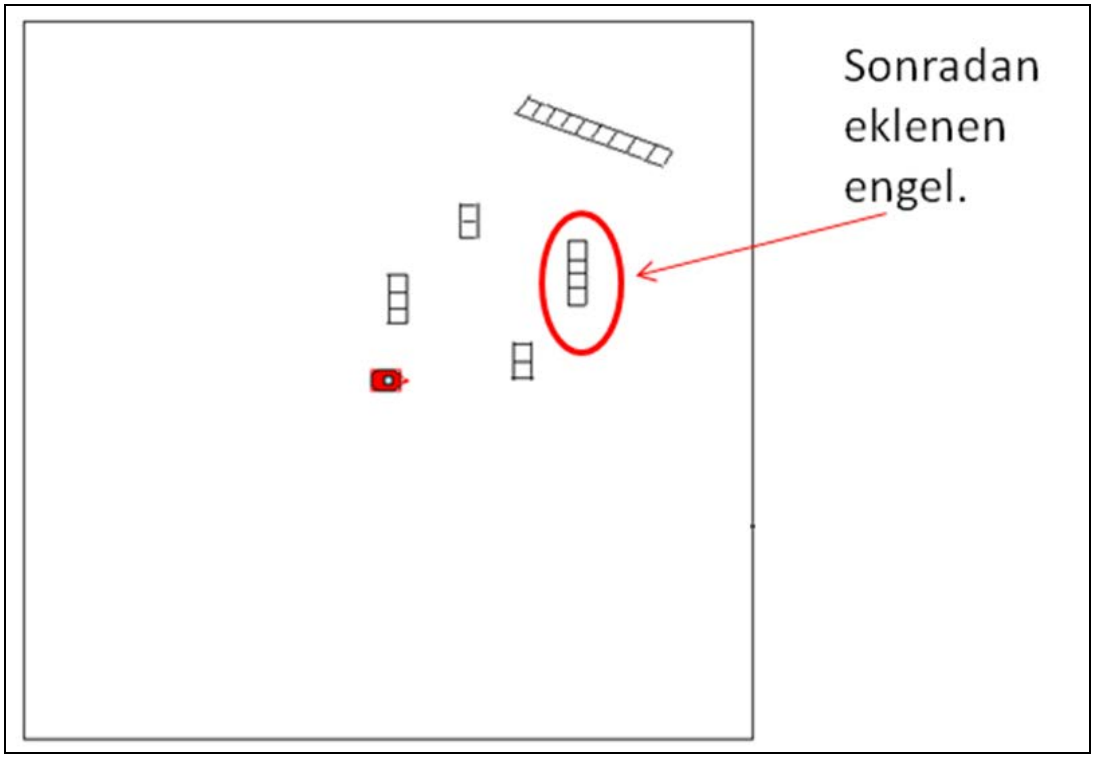
Dinamik engellerin olduğu bir ortamda ise, beklenmeyen bir engelin robotun karşısına çıkması durumunda ne olurdu? Sonuç elbette bellidir. Çarpışma kaçınılmaz olacak, gerek robotun gerekse ortamdaki engelin zarar görmesi gibi kötü bir durumla karşılaşılacaktır.

Bu durumun benzetimini yapmak için, Mapperbasic yardımı ile iki adet ortam oluşturuldu (Şekil 5.24 ve Şekil 5.25). İlk ortama göre yol bulma algoritması ile bir güzergahın belirlendiği varsayıldı. Sonradan çıkan bir engelin benzetimini yapmak

için ikinci ortamda ek bir engel eklendi. Tam da güzergah üstüne gelen bu engel ile karşılaştığında robotun hareketleri izlendi.



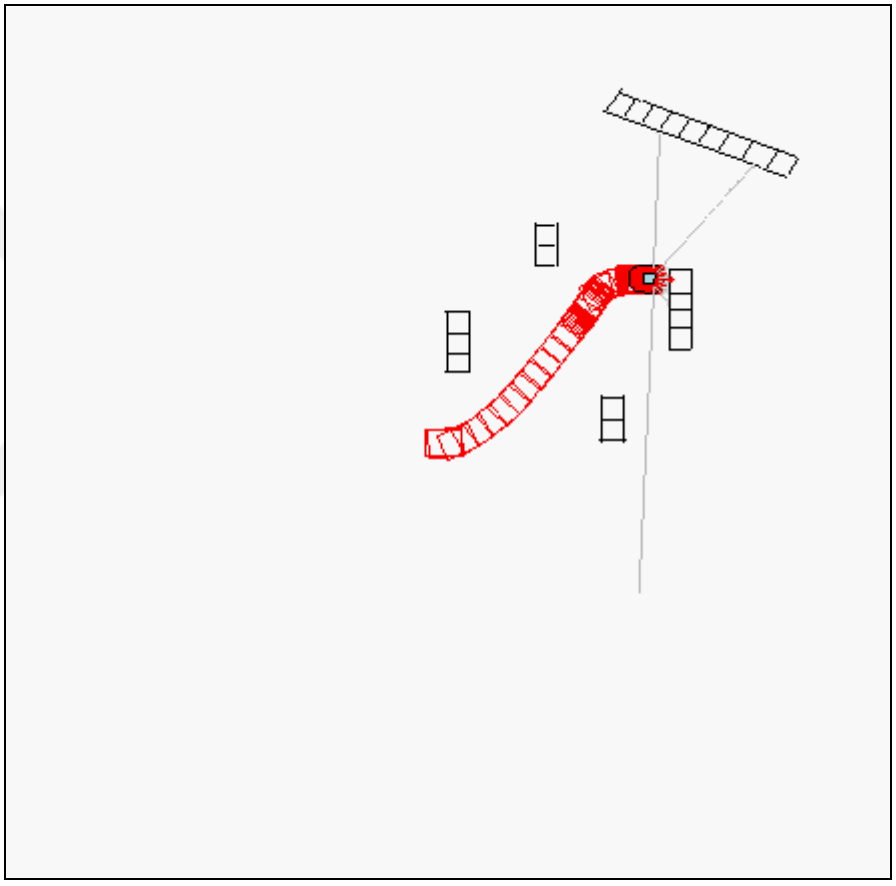
Şekil 5.24. Senaryo kapsamında hazırlanan ilk ortam



Şekil 5.25. Senaryo kapsamında hazırlanan ikinci ortam

Robota, Şekil 5.20'deki yol güzergahının iletilmesinin ardından, robot güzergah üzerinde ilerlemeye devam eder. Ancak sonradan eklenen engel hakkında bilgisi olmadığı ve engelden sakınma yeteneği bulunmadığı için çarpışma kaçınılmazdır (Şekil 5.26).

Bu basit senaryo ile engelden sakınmanın otonom robotlar için ne kadar hayati öneme sahip olduğu vurgulanmıştır. Sonraki bölümlerde, geliştirilen engelden sakınma yöntemi hakkında daha detaylı bilgi verilecektir.



Şekil 5.26. Sonradan eklenen engel ile robotun durumu

Geliştirilen algoritmaların test edilmesi aşamasında, sonradan engel eklenmiş Şekil 5.23'deki harita, temel harita olacaktır. Sonradan eklenen bu engelin, engelden sakınma algoritmaları aracılığıyla robota zarar vermeden etrafından nasıl dolaşıldığı örnekler ile açıklanacaktır.

5.5. Engelden Sakınma

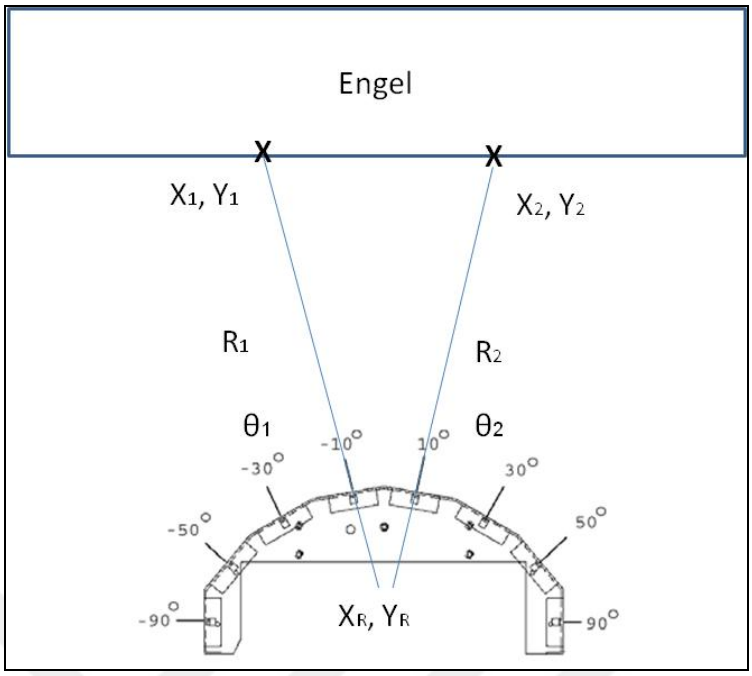
Bölüm 3'te literatürdeki engelden sakınma yöntemleri hakkında bilgi verilmişti. Tez çalışması kapsamında, engelden sakınma konusunda en yaygın olarak kullanılan Böcek-1, Böcek-2, Dist-bug ve geliştirmiş olduğumuz iyileştirilmiş Dist-bug algoritmalarının, MATLAB ortamında ARIA komutları yardımı ile gerçeklenmesi ve karşılaştırma yapılması sağlanmıştır. Bu bölümde, bu kapsamda yapılan çalışmalar hakkında bilgi verilecektir.

5.5.1. Böcek algoritmaları için geliştirilen duvar takibi

Böcek algoritmalarının ortak noktaları, engel ile karşılaştıklarında duvar takibi yapmalarıdır. Duvar takibi için çeşitli yöntemler bulunmaktadır. Sürekli olarak robotun yönü hedefe doğru döndürülmek suretiyle duvar takibi veya belli bir mesafede kalma gibi yöntemler izlenebilmektedir. Tez çalışması kapsamında geliştirmiş olduğumuz, duvar eğiminin hesaplanması yöntemi ile duvar takibi yapılmıştır.

Engel ile karşılaşıldığında, çevresinden en basit şekilde dolaşılması amacıyla, öncelikle engelin eğimi hesaplanmış ve bu eğime göre robot yönü belirlenerek, engele paralel bir şekilde engel etrafından dolaşım sağlanmıştır.

Engelin eğiminin hesaplanması için ilk olarak, Şekil 5.27'de görülen robotun, engelden almış olduğu sonar sinyalleri aracılığı ile X_1,Y_1 ve X_2,Y_2koordinatları hesaplanır. X_R, Y_R; robotun bulunduğu koordinatı, R_1ve θ_1 sırası ile sonar mesafeleri ve açıları olarak belirtilirse, X_1 ve Y_1koordinatları Denklem (5.4) ile hesaplanabilir.



Şekil 5.27. Engel ile karşılaşan robotun durumu

$$X_1 = X_R + \mathrm{COS}((\theta_1 - \theta_R) \times \pi \div 180) \times R_1$$
$$Y_1 = Y_R + \mathrm{SIN}((\theta_1 - \theta_R) \times \pi \div 180) \times R_1 \quad (5.4)$$

İkinci adımda ise, engele ait X_1,Y_1 ve X_2,Y_2koordinatları kullanarak engelin eğimi S_0 Denklem (5.5) ile hesaplanabilir.

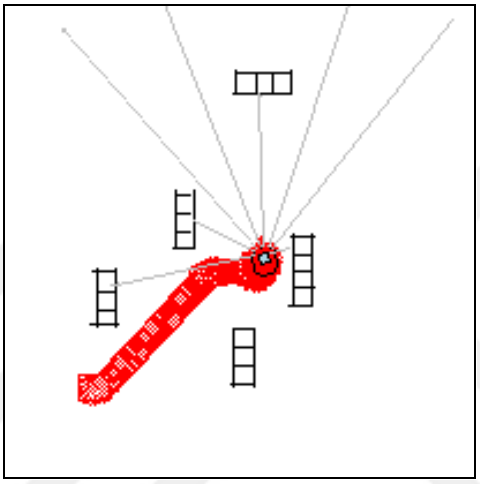
$$S_0 = \mathrm{ATAN}((Y_1 - Y_2)/(X_1 - X_2)) \times \pi \div 180 \quad (5.5)$$

Son adım olarak, önceden belirlenen eşik mesafe değerine ulaşıldığı anda, robotun yönü eğime paralel olarak ayarlanmakta ve engelin etrafında etkin bir dolaşım sağlanabilmektedir. Burada unutulmaması gereken önemli bir husus, engel etrafında dolaşırken düzenli olarak, yine engelin eğimi hesaplanmaya ve rota düzeltmesi yapılmaya devam edilmesidir. Ancak bu şekilde, farklı geometrik formlardaki engellerin de etrafından dolaşım sağlanabilmektedir.

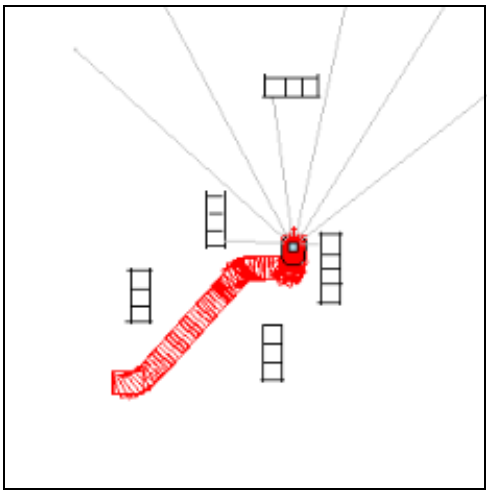
Algoritmadan da görüleceği üzere, duvar eğimini hesaplamak için en az iki sonar değerine ihtiyaç duyulmaktadır. Kullandığımız Pioneer 3DX robot üzerinde sadece ön sonarlar olması sebebi ile engelden tek sonardan veri alma gibi durumlar olabilecektir. Bu gibi durumlarda, tek sonarın eşik mesafesi kontrol edilmiş, rota değişimi ile ikinci sonardan da bilgi alınması sağlanmıştır.

Eğim hesaplanmadan engelin etrafında dolaşım yöntemi ele alındığında, robot engel ile belirlenen eşik mesafeye geldiğinde, rotasını sola çevirmekte ve sürekli olarak bir sonraki düğüm veya hedefe gitmek için tekrar rota düzenlemesi yapmak zorunda ve işlemlerin tekrarı yardımı ile engelin etrafında dolaşabilmekteydi (Şekil 5.28).

Engelin etrafından, engelin kenar eğiminin tespit edilerek dolaşımının sağlanması, algoritmalarda önemli ölçüde kazanç sağlamış, gereksiz rota değişikliği ihtiyacını ortadan kaldırmıştır (Şekil 5.29).



Şekil 5.28. Eğim hesaplı duvar takip algoritması yok iken robot



Şekil 5.29. Eğim hesaplı duvar takip algoritması ile robot

5.5.2. Böcek-1 algoritmasının gerçeklenmesi

Böcek-1 algoritması en basit engelden sakınma algoritması olup, ana fikir, robotun yolu üstündeki karşılaştığı engelin dış çevresini takip etmek ve bu sayede çevresini dolaşmaktır. Bu dolaşma esnasında, sürekli olarak bir sonraki düğüm noktasına olan en kısa mesafe hafızaya kayıt edilmek zorundadır. Engel etrafında bir tam tur tamamlandığında, en kısa mesafeye sahip noktaya giderek, bu noktadan başlayarak bir sonraki düğüm noktasına ilerler. Şekil 5.30'da böcek-1 algoritması verilmektedir.

```
For (rota üzerindeki her bir adım için){
    hedef = yeni_düğüm_noktası;
    adım_hedefe_varildi_mi=0;
    durum = YOL_BULMA;
    While (adım_hedefe_varildi_mi==0){
        hedefe_mesafe_hesapla();
        if (hedefe_varildi_mi==1) {
            adim_hedef_varildi_mi=1;
            yeni_adima_gec;
        }
        if(durum == YOL_BULMA)
            Yeni_rota = Hedef_acisi_hesapla();
                if(engel_varmi()==1)
                    durum=DUVAR_TAKIP;
                end;
        end;
        if(durum== DUVAR_TAKIP)
            ayrılış_noktasına_kadar_duvar_takip_yap();
            durum=YOL_BULMA;
        end;
    Rotayı_değiştir(yeni_rota);
    Sürat_ayarla(ilerleme_sürati);
    }
}
Sürat_ayarla(0);
Exit();
```

Şekil 5.30. Böcek 1 algoritması

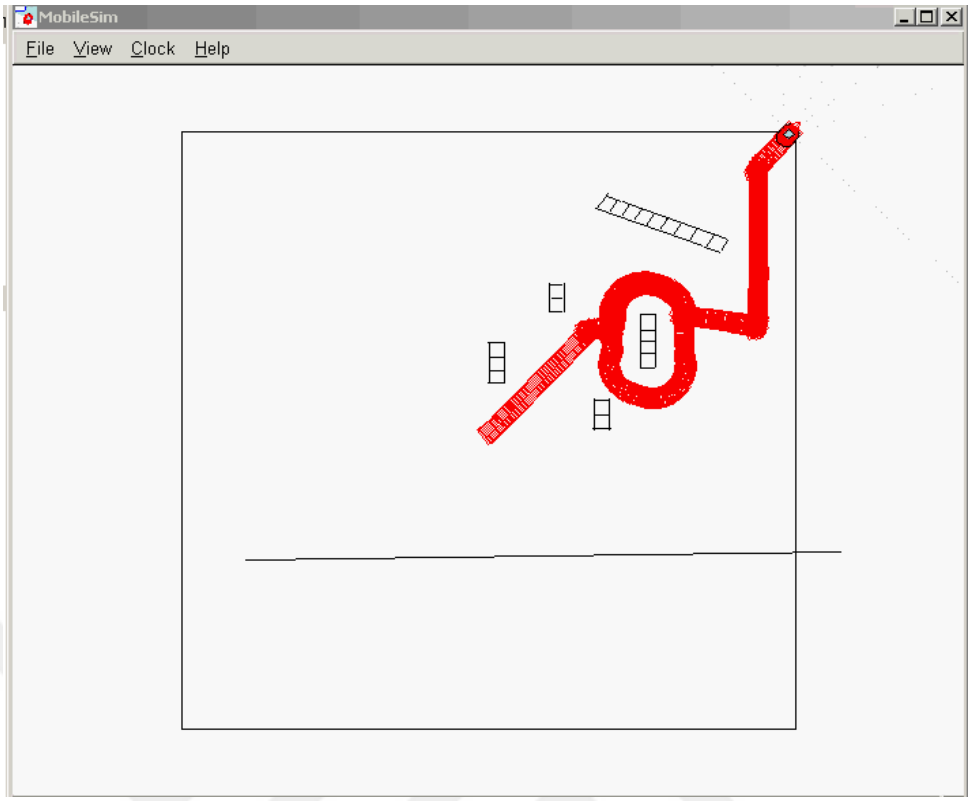
Böcek-1 algoritmasında kullanılan ve tez çalışması ile geliştirmiş olduğumuz ayrılış_noktasına_kadar_duvar_takip_yap fonksiyonu aşağıdaki şekilde çalışmaktadır (Şekil 5.31).

```
ayrılış_noktasına_kadar_duvar_takip_yap(){
    ayrilis_noktasina_varildi_mi=0;
    islem_tipi=baslangic_noktasina_kadar_duvar_takip;
    duvar_takip_başladımı=0;
    ilk_hedef_acisi=hedef_acisi_hesapla();
    while(ayrilis_noktasina_varildi_mi==0)
        if(duvar_takip_başladımı==0)
            if(ilk_hedef_acisi<>hedef_acisi_hesapla()
            duvar_takip_başladımı=1;
            end;
        end;
        if(islem_tipi==ayrilis_noktasina_gidelim&&
            ayrilis_noktasina_geldik_mi()==1)
            ayrilis_noktasina_varildi_mi=1;
        end;
        if(duvar_takip_başladımı==1&&
            islem_tipi==baslangic_noktasina_kadar_duvar_takip)
            if(baslangic_noktasina_geldik_mi()==1)
                islem_tipi==ayrilis_noktasina_gidelim;
            elseif(hedefe_en_yakın_mesafe>hedefe_mesafe_hesapla())
                hedefe_en_yakın_mesafe=hedefe_mesafe_hesapla();
                ayrılıs_noktası=suanki_konum;
                yeni_rota=duvar_takip_icin_rota_hesapla();
            end;
        else if(islem_tipi == ayrilis_noktasina_gidelim)
            yeni_rota=duvar_takip_icin_rota_hesapla();
        end;
    Rotayı_değiştir(yeni_rota);
    Sürat_ayarla(ilerleme_sürati);
    end;
}
```

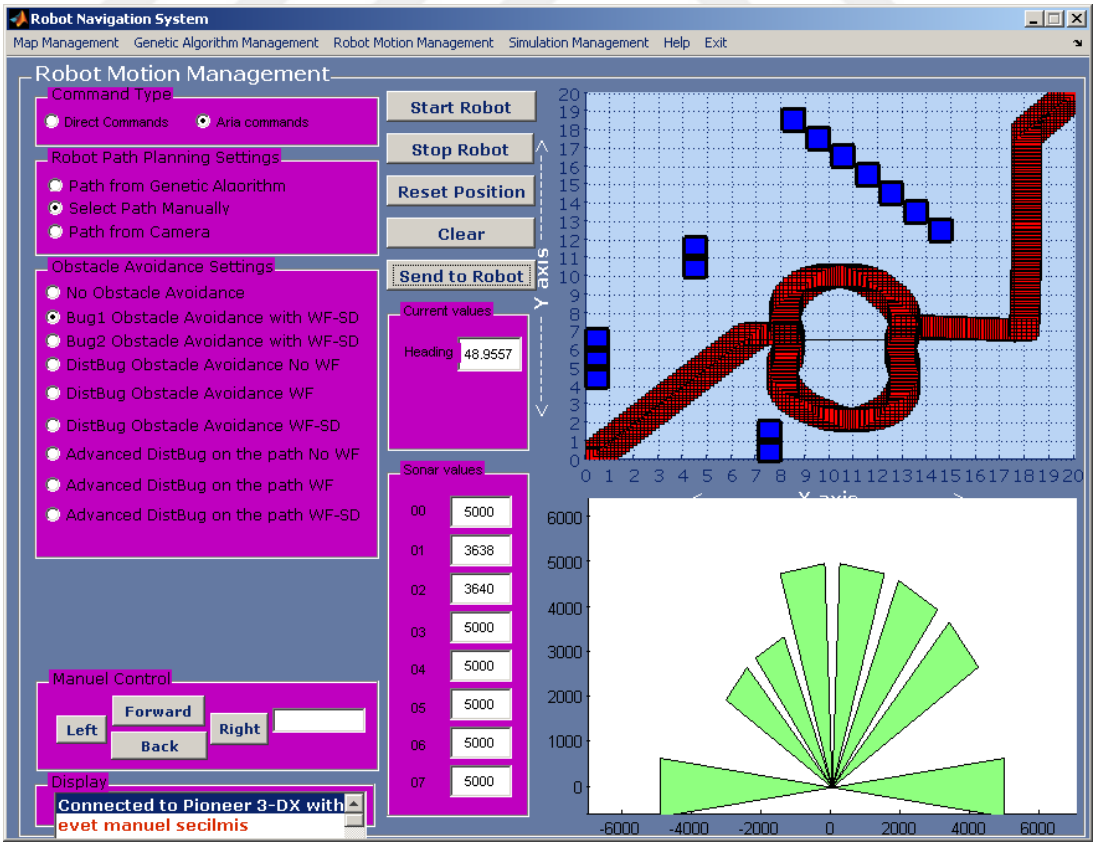
Şekil 5.31. Böcek 1 ayrılış noktasına kadar duvar takip yapma algoritması

Ayrılış noktasına kadar duvar takip yapma fonksiyonu böcek 1 için, engelin etrafında tam bir tur atılmasını sağlamakta, bu tam tur esnasında sürekli olarak bir sonraki düğüm noktasına en yakın noktayı (ayrılış noktası) hesaplamaktadır. Tam tur tamamlandıktan sonra, ayrılış noktasına yine duvar takip yaparak ilerleyerek, engelin etrafındaki tur tamamlanmaktadır.

Bu algoritmanın çalıştırılması neticesinde, engel etrafında başarı ile duvar takibi yapılabildiği ve engel etrafındaki bir tam tur sonrasında, bir sonraki düğüm noktasına en yakın mesafede engelden ayrılarak yoluna devam ettiği görülmüştür. Şekil 5.32 ve Şekil 5.33'de sırası ile, duvar eğimini takip eden Böcek-1 algoritmasının MobileSim'de benzetimi ve robot seyir sisteminde gerçeklenmesi gösterilmektedir.



Şekil 5.32. MobileSim'de duvar takibi yapan Böcek-1 algoritması



Şekil 5.33. Robot seyir sisteminde duvar takibi yapan Böcek-1 algoritması

5.5.3. Böcek-2 algoritmasının gerçeklenmesi

Böcek-2 algoritmasında, robot belirlenen yol güzergahında bulunan bir sonraki düğüme ilerlerken bir engelle karşılaştığında, Böcek-1'de olduğu gibi, engelin dış çevresini takip etmekte, ancak bir tam tur atmak yerine, tekrar yol güzergahına girdiğinde, engelden ayrılmakta ve bir sonraki düğüm noktasına doğru ilerlemektedir. Böcek-2 algoritması aşağıda verilmektedir (Şekil 5.34).

```
For (rota üzerindeki her bir adım için){
    hedef = yeni_düğüm_noktası;
    adım_hedefe_varildi_mi=0;
    durum = YOL_BULMA;
    While (adım_hedefe_varildi_mi==0){
        hedefe_mesafe_hesapla();
        if (hedefe_varildi_mi==1) {
            adim_hedef_varildi_mi=1;
            yeni_adima_gec;
        }
        if(durum == YOL_BULMA)
            Yeni_rota = Hedef_acisi_hesapla();
                if(engel_varmi()==1)
                    durum=DUVAR_TAKIP;
                end;
        end;
        if(durum== DUVAR_TAKIP)
            ayrılış_noktasına_kadar_duvar_takip_yap();
            durum=YOL_BULMA;
        end;
    Rotayı_değiştir(yeni_rota);
    Sürat_ayarla(ilerleme_sürati);
    }
}
Sürat_ayarla(0);
Exit();
```

Şekil 5.34. Robot seyir sisteminde duvar takibi yapan Böcek-2 algoritması

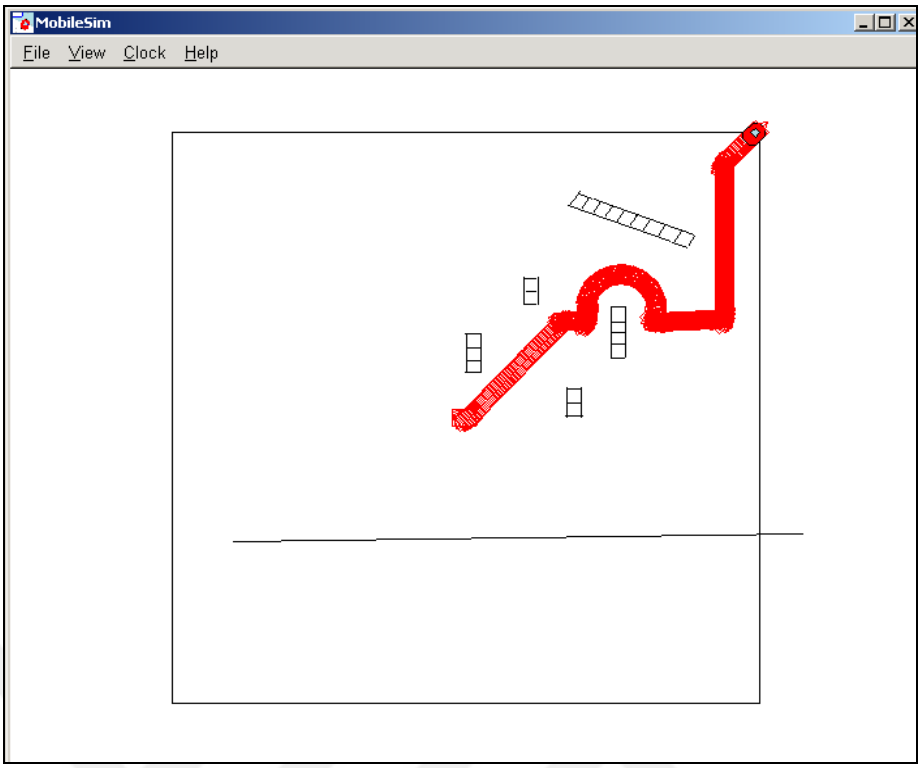
Görüldüğü üzere, algoritma her iki böcek algoritması için de aynı şekildedir. İki algoritma arasındaki fark, engelden ayrılma noktasının tespiti aşamasında olmaktadır. Böcek-2 algoritması kapsamında kullanılan ayrılış_noktasına_kadar_duvar_takip_yap fonksiyonu aşağıdaki şekilde çalışmaktadır (Şekil 5.35).

```
ayrılış_noktasına_kadar_duvar_takip_yap(){
    ayrilis_noktasina_varildi_mi=0;
    islem_tipi=baslangic_noktasina_kadar_duvar_takip;
    duvar_takip_başladımı=0;
    ilk_hedef_acisi=hedef_acisi_hesapla();
    while(ayrilis_noktasina_varildi_mi==0)
        if(duvar_takip_başladımı==0)
            if(ilk_hedef_acisi<>hedef_acisi_hesapla()
            duvar_takip_başladımı=1;
            end;
        end;
        if(islem_tipi==ayrilis_noktasina_gidelim&&
            ayrilis_noktasina_geldik_mi()==1)
            ayrilis_noktasina_varildi_mi=1;
        end;
        if(duvar_takip_başladımı==1 )
            if(ilk_hedef_acisi==hedef_acisi_hesapla())
                ayrilis_noktasina_varildi_mi=1;
            else
                yeni_rota=duvar_takip_icin_rota_hesapla();
            end;
        end;
    Rotayı_değiştir(yeni_rota);
    Sürat_ayarla(ilerleme_sürati);
    end;
}
```

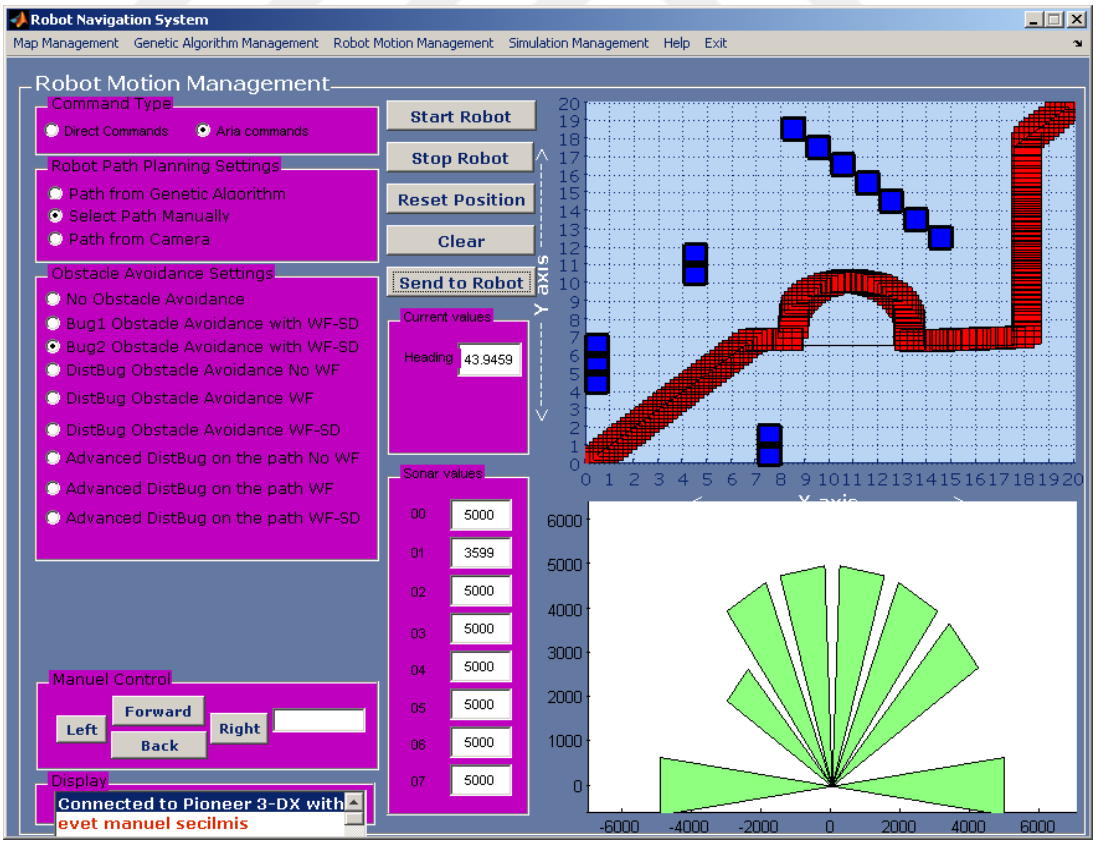
Şekil 5.35. Böcek 2 ayrılış noktasına kadar duvar takip yapma algoritması

Böcek-2 algoritmasında tam bir tur atmaksızın, hedef_hattı üzerine ilk varıldığı noktada engelden ayrılma gerekmektedir. Bu kapsamda ayrılış noktasına kadar duvar takip yap fonksiyonu duvar takip esnasında sürekli olarak bir sonraki düğüm noktasına olan hedef hattını kontrol etmektedir. Hedef hattına gelindiği noktada engelden ayrılmak üzere ana programa dönülmektedir.

Bu algoritmada Böcek-1'de olduğu gibi engel etrafından tam bir tur atılmaması nedeniyle Böcek-1'e göre büyük avantaj elde edildiği görülmektedir. Şekil 5.36 ve Şekil 5.37'de robotun MobileSim benzetim ortamı ve robot seyir sistemi üzerinde Böcek-2 algoritması iz bilgisi görülmektedir.



Şekil 5.36. MobileSim'de duvar takibi yapan Böcek-2 algoritması



Şekil 5.37. Robot seyir sisteminde duvar takibi yapan Böcek-2 algoritması

5.5.4. Dist-bug algoritmasının gerçeklenmesi

Dist-bug algoritmasında robot, bir sonraki düğüm noktasına doğru ilerlerken, karşısına bir engel çıkması durumunda, engelin etrafında dolaşmaya başlar. Engel etrafında ilerlerken tam bir tur atmak yerine, sürekli olarak sonar algılayıcıları ile bir sonraki düğüm noktası doğrultusunu kontrol etmektedir. Bir sonraki düğüm noktası doğrultusunda, yeteri kadar serbest bir alan tespit etmesi durumunda, engelden ayrılmakta ve düğüm noktasına doğru ilerlemektedir.

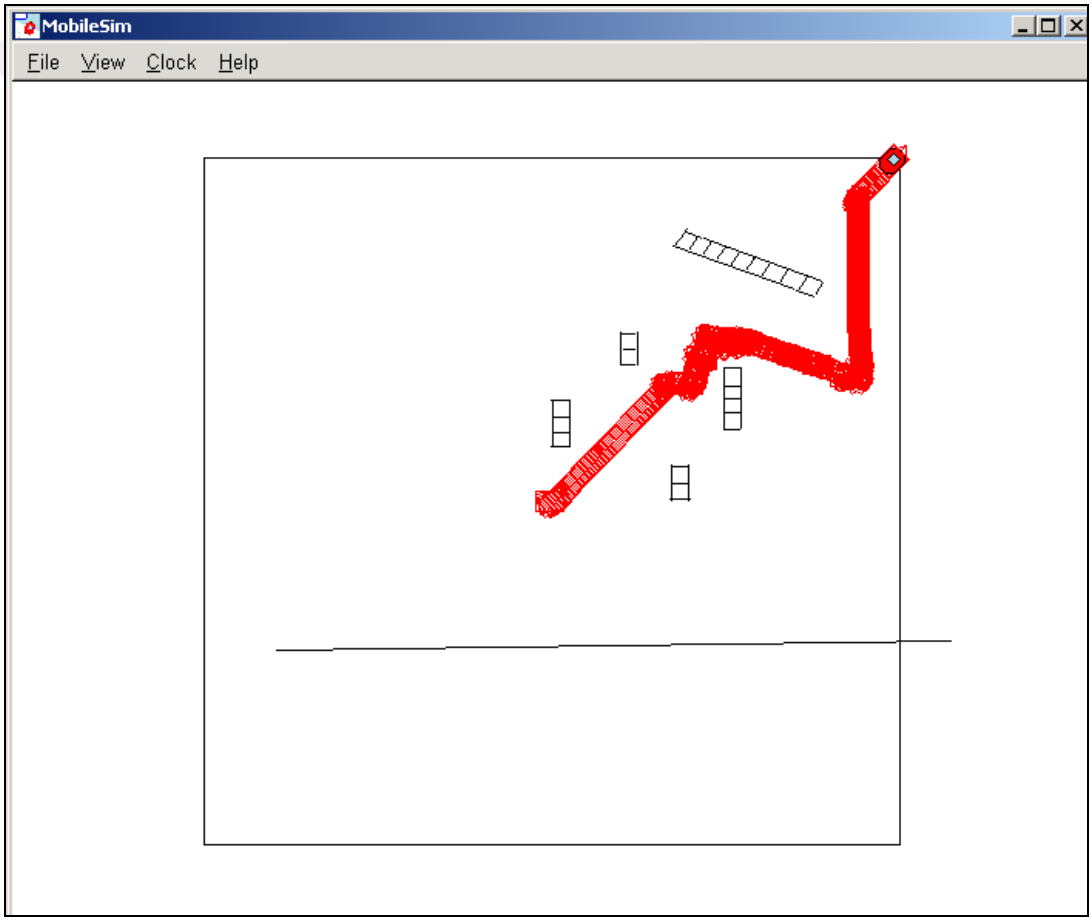
```
ayrılış_noktasına_kadar_duvar_takip_yap(){
    ayrilis_noktasina_varildi_mi=0;
    islem_tipi=baslangic_noktasina_kadar_duvar_takip;
    duvar_takip_başladımı=0;
    ilk_hedef_acisi=hedef_acisi_hesapla();
    while(ayrilis_noktasina_varildi_mi==0)
        if(duvar_takip_başladımı==0)
            if(ilk_hedef_acisi<>hedef_acisi_hesapla()
            duvar_takip_başladımı=1;
            end;
        end;
        if(islem_tipi==ayrilis_noktasina_gidelim&&
            ayrilis_noktasina_geldik_mi()==1)
            ayrilis_noktasina_varildi_mi=1;
        end;
        if(duvar_takip_başladımı==1)
            sonar_degerleri=[50,s1,30,s2,10,s3,-10,s4,-30,s5,-50,s6];
            if(hedef_acisi>=-90&&hedef_acisi<=90)
                for j=1:2:12
                    if(hedef_acisi>=(suanki_yonumuz+ sonar_degerleri(j)-7.5)&&
                    hedef_acisi<=(suanki_yonumuz+ sonar_degerleri(j)+7.5))
                    serbest_mesafe=sonar_degerleri(j+1);
                    end;
                end;
            end;
            if(serbest_mesafe> 1000)
                ayrilis_noktasina_varildi_mi=1;
            else
                yeni_rota=duvar_takip_icin_rota_hesapla();
            end;
        end;
    end;
    Rotayı_değiştir(yeni_rota);
    Sürat_ayarla(ilerleme_sürati);
    end;
}
```

Şekil 5.38. Dist-bug ayrılış noktasına kadar duvar takip yapma algoritması

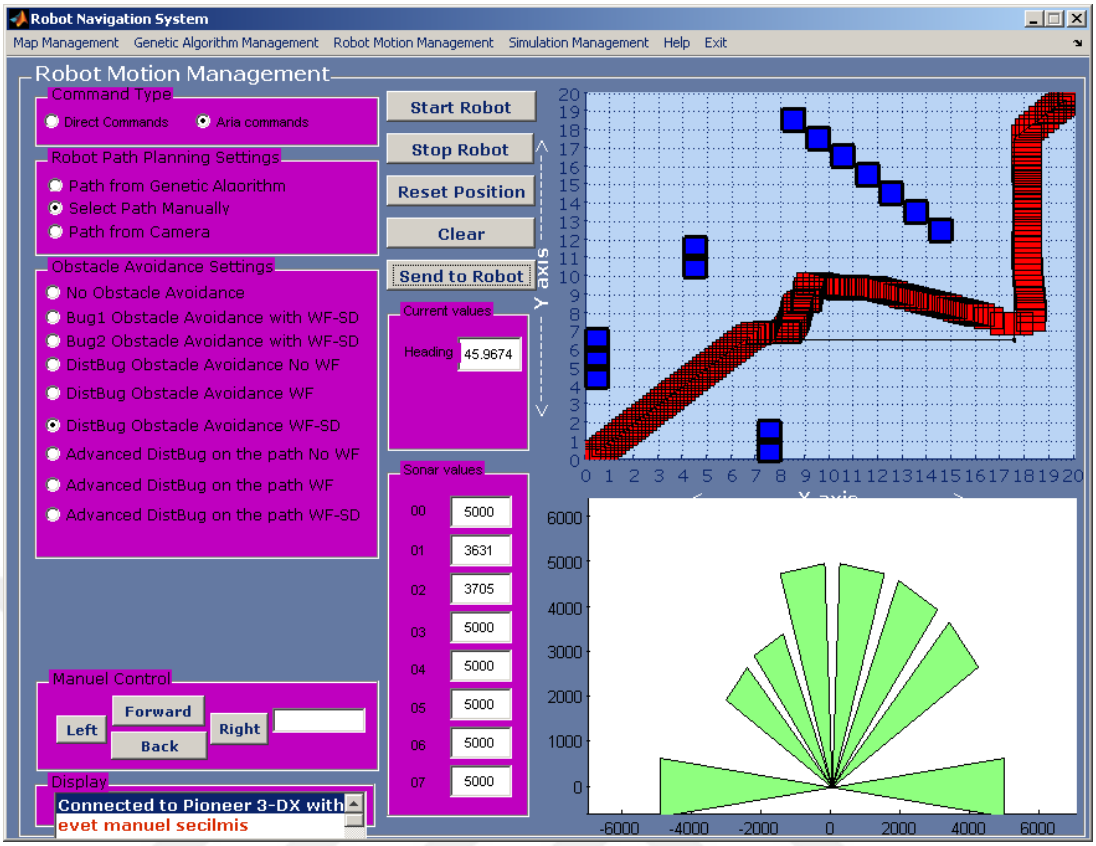
Ana algoritma Böcek-1 ve Böcek-2 de olduğu gibi Dist-bug algoritmasında da aynıdır. Farklılık ayrılış_noktasına_kadar_duvar_takip_yap fonksiyonunda bulunmaktadır. Dist-bug algoritmasında kullanılan ayrılış noktasına kadar duvar takip yapma algoritması yukarıdaki şekilde çalışmaktadır (Şekil 5.38).

Robot böcek-1 algoritmasındaki gibi tam bir tur atmamakta, böcek-2 algoritmasında olduğu gibi hedef hattı noktasını ayrılış noktası olarak almamaktadır. Bu sayede böcek-1 ve böcek-2 den daha etkin bir algoritma olarak karşımıza çıkmaktadır.

Robotun ilerleyişi Şekil 5.39 ve 5.40'da gösterilmektedir.



Şekil 5.39. MobileSim'de duvar takibi yapan Dist-bug algoritması



Şekil 5.40. Robot seyir sisteminde duvar takibi yapan Dist-bug algoritması

5.5.5. Böcek1, böcek2 ve dist-bug algoritmalarının karşılaştırması

Böcek algoritmaları MATLAB ortamında yazılımsal olarak gerçeklenmiş ve gerek benzetim ortamında gerekse de gerçek robot üstünde başarı ile çalıştıkları tespit edilmiştir. Yazılım üzerinden çeşitli süratler ile testler yapılabilmektedir. Sürat 50mm/sn olacak şekilde yapılmış testlerde, kat edilen mesafe ve süre değerlerinin kayıt edilmesi neticesinde aşağıdaki değerler elde edilmiştir.

Tablo 5.2. Böcek algoritmalarında elde edilen ölçüm değerleri

Algoritma	**Süre**	**Kat edilen Mesafe**
Engelsiz düz güzergah	242,96 saniye	12148 mm
Böcek1	408,10 saniye	20405 mm
Böcek2	281,49 saniye	14059 mm
Dist-bug	269,80 saniye	13490 mm

Tablo 5.2 deki veriler ışığında, Dist-bug algoritmasının Böcek-1 ve Böcek-2'ye nazaran daha performanslı sonuç verdiği tespit edilmiştir.

5.5.6. İyileştirilmiş Dist-bug algoritması

Distbug algoritmasında robot, bir sonraki düğüm noktasına doğru ilerlerken, karşısına bir engel çıkması durumunda, engelin etrafında dolaşmaya başlamaktadır. Engel etrafında ilerlerken tam bir tur atmak yerine, sürekli olarak sonar algılayıcıları ile bir sonraki düğüm noktası doğrultusunu kontrol etmektedir. Bu algoritmada yapılan bir iyileştirme ile engelden ayrılma noktası tespit edilirken, bir sonraki düğüm yerine mümkün olan en ileri düğüm noktası doğrultusunun kontrol edilmesinin performans açısından büyük fayda sağlayacağı değerlendirilmiştir. Tez çalışması kapsamında geliştirmiş olduğumuz iyileştirilmiş Dist-bug algoritması, sonarların kapsama alanı içerisinde kalan engelsiz en uzak düğümün, kestirme yol güzergahı olarak seçilmesi anlamına gelmektedir.

İyileştirilmiş Dist-bug algoritması aşağıda verilmektedir (Şekil 5.41).

Standart Dist-bug'a göre, algoritma akışındaki en önemli değişiklik, ayrılış_noktasına_kadar_duvar_takip_yap fonksiyonundan gelen yeni düğüm noktası bilgisidir. Hesaplanan yeni düğüm noktasının şu anki düğüm noktasından farklı olması durumunda, düğüm noktaları arasında ilerlemeyi sağlayan ana for döngüsünde continue komutu ile düğüm atlatma yapılmaktadır.

Bu sayede ulaşılabilecek durumdaki engele en yakın düğüm noktasına gidilmesi sağlanmakta, tespit edilen yeni düğüm noktasından itibaren güzergah üzerinde ilerlemeye devam edilmektedir.

```
For (rota üzerindeki her bir adım için)
    kacinci_dugum_noktasi = kacinci_dugum_noktasi +1;
    hedef = yeni_düğüm_noktası;
    adım_hedefe_varildi_mi=0;
    durum = YOL_BULMA;
    if (dugum_atlandimi ==1 &&
        kacinci_dugum_noktasi < yeni_dugum_noktasi)
        continue;
    end;
    if (dugum_atlandimi==1 &&
        kacinci_dugum_noktasi == yeni_dugum_noktasi)
        dugum_atlandimi=0;
        duvara_gelindi=0;
    end;
    While (adım_hedefe_varildi_mi==0 && dugum_atlandimi ==0)
        hedefe_mesafe_hesapla();
        if (hedefe_varildi_mi==1)
            adim_hedef_varildi_mi=1;
            yeni_adima_gec;
        end;
        if(durum == YOL_BULMA)
            Yeni_rota = Hedef_acisi_hesapla();
            if(engel_varmi()==1)
                durum=DUVAR_TAKIP;
            end;
        end;
        if(durum== DUVAR_TAKIP)
            yeni_dugum_noktasi =
            ayrılış_noktasına_kadar_duvar_takip_yap();
            if(yeni_dugum_noktasi>kacinci_dugum_noktasi)
                dugum_atlandimi =1;
            end;
        durum=YOL_BULMA;
        end;
    Rotayı_değiştir(yeni_rota);
    Sürat_ayarla(ilerleme_sürati);
    end;
end;
Sürat_ayarla(0);
Exit();
```

Şekil 5.41. Düğüm atlama özellikli Dist-bug algoritması

Ayrılış noktasına kadar duvar takip yap fonksiyonu, sürekli olarak ulaşılabilir sonraki düğüm noktası hesaplama fonksiyonu yardımı ile ulaşılabilir durumdaki hedefe en yakın düğüm noktasını bulmaya çalışmaktadır. Fonksiyon yeni düğüm noktası bilgisini ana programa döndürmektedir (Şekil 5.42).

```
yeni_dugum_noktasi  = ayrılış_noktasına_kadar_duvar_takip_yap(){
    yeni_dugum_noktasi = sonraki_dugum_noktasi;
    ayrilis_noktasina_varildi_mi=0;
    islem_tipi=baslangic_noktasina_kadar_duvar_takip;
    duvar_takip_başladımı=0;
    ilk_hedef_acisi=hedef_acisi_hesapla();
    while(ayrilis_noktasina_varildi_mi==0)
        if(duvar_takip_başladımı==0)
            if(ilk_hedef_acisi<>hedef_acisi_hesapla()
            duvar_takip_başladımı=1;
            end;
        end;
        if(islem_tipi==ayrilis_noktasina_gidelim&&
            ayrilis_noktasina_geldik_mi()==1)
            ayrilis_noktasina_varildi_mi=1;
        end;
        if(duvar_takip_başladımı==1)
            sonar_degerleri=[50,s1,30,s2,10,s3,-10,s4,-30,s5,-50,s6];
            if(hedef_acisi>=-90&&hedef_acisi<=90)
                yeni_dugum_noktasi = ulasilabilir_sonraki_dugumu_hesapla(
                kacinci_dugum_noktasi,x,y,suanki_yonumuz,sonar_degerleri)
            end;
            if(yeni_dugum_noktasi <> -1)
                ayrilis_noktasina_varildi_mi=1;
            else
                yeni_rota=duvar_takip_icin_rota_hesapla();
            end;
        end;
    end;
    Rotayı_değiştir(yeni_rota);
    Sürat_ayarla(ilerleme_sürati);
    end;
}
```

Şekil 5.42. Ayrılış noktasına kadar duvar takip yapma algoritması

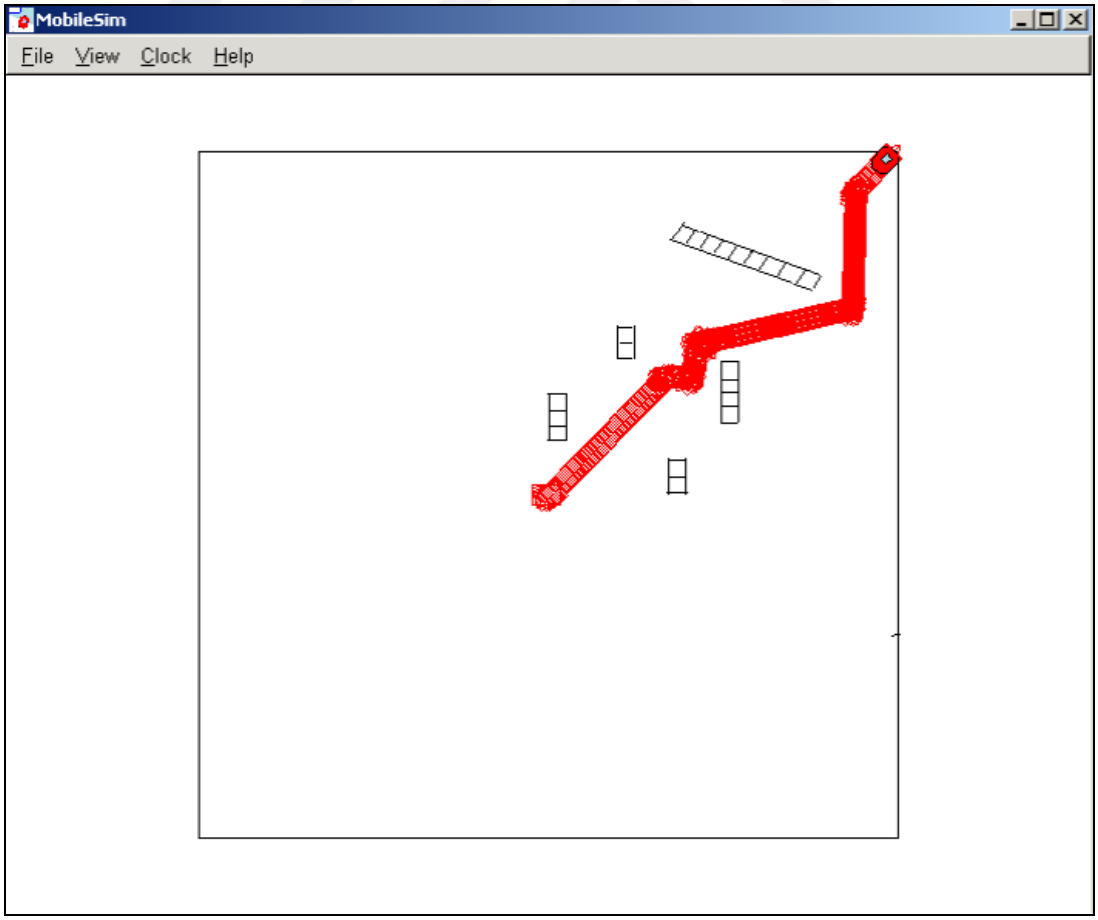
Ulaşılabilir_sonraki_dugumu_hesapla fonksiyonunda, hedef noktadan, bulunduğumuz düğüme kadar tüm düğüm noktaları incelenir. Sonra, değerlerine göre ulaşılabilir en uzak düğüm noktası tespit edilerek, yeni düğüm noktası olarak ayarlanır.

Bu amaçla fonksiyon içinde, bir döngü dahilinde, her bir düğüm noktasının +-7,5 derecesi içinde kalan alandaki engelsiz boş alan hesaplanır ve yeni düğüm noktası, ayrılış noktasına kadar duvar takip yap fonksiyonuna döndürülür (Şekil 5.43).

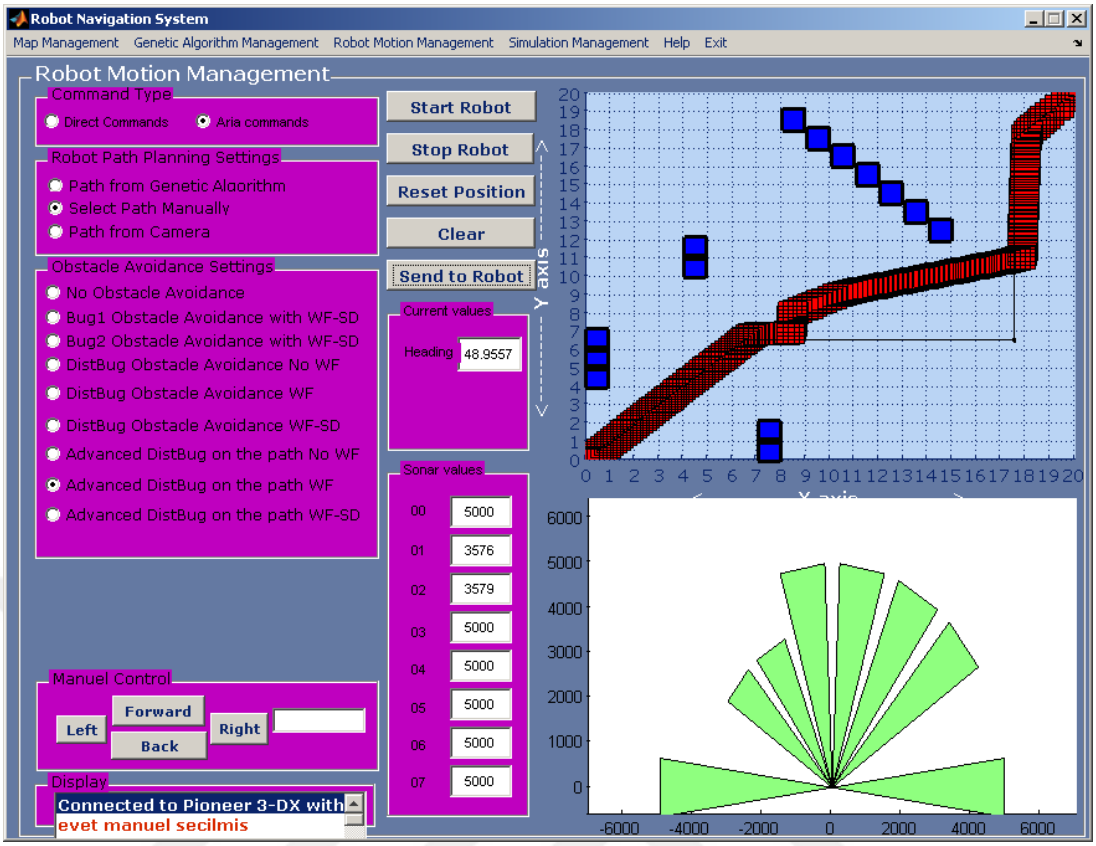
```
yeni_dugum_noktasi = ulasilabilir_sonraki_dugumu_hesapla(
   kacinci_dugum_noktasi,x,y,suanki_yonumuz,sonar_degerleri)
   global yol;
   yeni_dugum_noktasi =kacinci_dugum_noktasi;
       for i=length(yol):-2:kacinci_dugum_noktasi*2
           aday_hedef_y=yol(i)*333;
           aday_hedef_x=yol(i-1)*333;
           hedefe_mesafe_hesapla();
           hedef_acisi_hesapla();
           for j=1:2:16
               if(hedef_yonu >= (suanki_yonumuz + sonar_degerleri(j) - 7.5) &&...
                   hedef_yonu <= (suanki_yonumuz + sonar_degerleri(j) + 7.5) &&...
                   hedefe_mesafe < sonar_degerleri(j+1))
                   yeni_dugum_noktasi = i/2 - 1;
                   return;
               end;
           end;
       end;
   end;
```

Şekil 5.43. Ulaşılabilir sonraki düğümü hesapla fonksiyonu

Şekil 5.44 ve 5.45'de robotun izlemesi gereken yol ve atlanan düğüm noktası üzerinden robotun izlediği yol görülmektedir.



Şekil 5.44. MobileSim'de iyileştirilmiş Dist-bug algoritması



Şekil 5.45. Robot seyir sisteminde iyileştirilmiş Dist-bug algoritması

5.5.7. Dist-bug ve iyileştirilmiş Dist-bug karşılaştırması

Dist-bug ve düğüm atlama özellikli iyileştirilmiş Dist-bug algoritmaları MATLAB ortamında yazılımsal olarak gerçeklenmiş ve gerek benzetim ortamında gerekse de gerçek robot üstünde başarı ile çalıştıkları tespit edilmiştir. Kat edilen mesafe ve süre değerlerinin kayıt edilmesi neticesinde aşağıdaki değerler elde edilmiştir.

Tablo 5.3. Dist-bug ve iyileştirilmiş Dist-bug algoritmalarının karşılaştırması

Algoritma	**Süre**	**Katedilen Mesafe**
Dist-bug	269,80 saniye	13490 mm
İyileştirilmiş Dist-bug	226,31 saniye	11315 mm

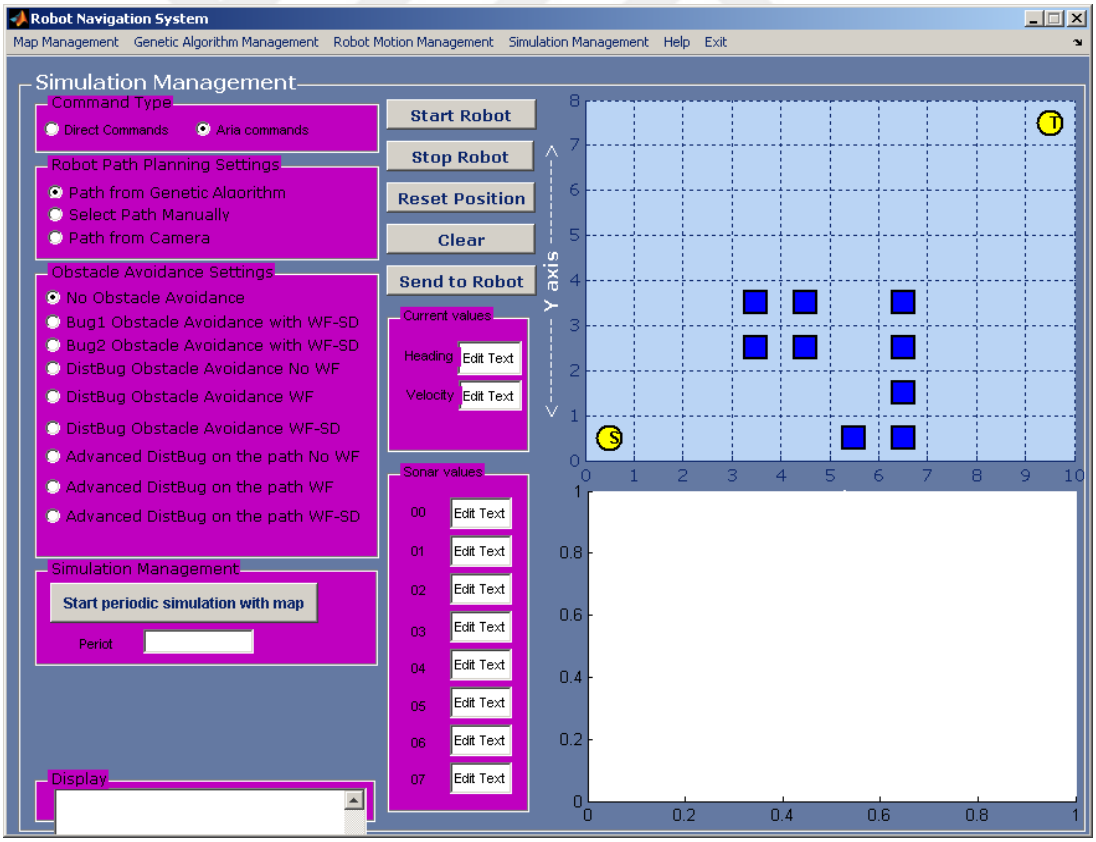
Tablo 5.3 deki veriler ışığında, iyileştirilmiş Dist-bug algoritmasının standart Dist-bug'a nazaran daha performanslı sonuç verdiği tespit edilmiştir. Hatta Tablo 5.3'deki engelsiz düz mesafe ile karşılaştırıldığında dahi daha performanslı olduğu

gözlemlenmiştir. Adeta engelden ayrıldıktan sonra yeni bir yol planlaması yapılmış gibi iyi bir sonuç vermiştir.

5.6. Engelden Sakınma Yerine Sadece Yol Bulma İle Seyir Emniyeti

Sadece yol bulma algoritmasına güvenerek seyir emniyetini sağlamak, tamamen statik ortamlarda ancak mümkün olabilir. Engelden sakınma yeteneği olmayan bir robot için, karşısına çıkabilecek öngörülmeyen dinamik engeller durumunda, çarpışmanın kaçınılmaz olacağı ve robot ile etrafındaki nesnelere zararların verilebileceği örnek senaryo ile gösterilmişti.

Bununla birlikte, tez çalışması kapsamında ayrıca, yol bulma algoritmasının periyodik olarak hesaplanması durumu da incelenmiştir. Geliştirilen robot seyir sisteminde, periyodik olarak yol bulma işlemi yapılarak, robotun başlangıç noktasından hedef noktasına kadar ilerlemesi test edilmiştir.



Şekil 5.46. Periyodik yol bulma maksatlı benzetim yönetim ekranı

Şekil 5.46'da robot seyir sisteminde periyodik yol bulma maksatlı benzetim yönetim ekranı görülmektedir.

Tavandaki kameradan yeni görüntünün alınması, robotun ve engellerin lokasyonunun tespit edilerek haritanın oluşturulması, oluşturulan haritadaki bilgiler ışığında genetik algoritma ile yol bulma faaliyetinin icrası ve son olarak, bu işlemler esnasında robotun kendi emniyeti için hareketsiz kalması, periyodik yol bulma sürecinin efektif olmayacağını göstermiştir.

Laboratuar ortamındaki testler esnasında tutulan kayıtlara göre, harita oluşturma ve genetik algoritmanın Tablo 5.4'de belirtilen süreler içinde icra edildiği tespit edilmiştir.

Tablo 5.4.Yol bulma işlemine ait süreler

İşlem	**Süre**
Tavandaki kamera yardımı ile haritanın oluşturulması	1 sn
Oluşturulmuş 10x 8 harita üzerinde genetik algoritma ile yol bulma	6 sn

Benzetim yönetim ekranı, periyodik yol bulma benzetimleri için periyot parametresi girilebilecek şekilde geliştirildi. Bu sayede, 40 veya 30 sn gibi farklı periyotlarda periyodik yol bulma benzetimi için laboratuar ortamında testler yapıldı. Yapılan testlerde, her ne kadar işlem süreleri toplamda 7 sn gibi kısa bir süre tutsa da, robotun bu süre içinde beklemede kalması, seyir süresini her 30 sn lik periyot için 7 sn uzatmış, bu da normalde 2 dk sürecek olan seyir süresinin sadece işlem beklemesi nedeniyle 28 sn gibi önemli oranda uzatmıştır. Elde edilen sonuçlar paralelinde, laboratuardaki test ortamında, periyodik yol bulma işleminin, engelden sakınma algoritmasından daha etkin olmadığı tespit edilmiştir.

Bu nedenle, test sonuçları paralelinde, engelden sakınma yeteneğinin, otonom bir robot için gerek kendi bekası gerekse çevre emniyeti için olmazsa olmaz özelliklerden biri olduğu değerlendirilmiştir.

6. SONUÇLAR VE ÖNERİLER

Otonom robotlar günümüzde birçok alanda önemli görevler icra etmektedirler. Robot, kendisine verilen bir görevi yerine getirirken, hedefe ilerlemek için yol bulma problemini çözmeli ayrıca kendi bekasını korumak için, algılayıcıları yardımıyla çevreden elde ettiği verileri kullanarak, varsa olası bir çarpışmayı önlemek için statik veya dinamik olarak engellerden sakınma yapabilmelidir.

Sonar algılayıcıları ekonomik olmaları ve pratik kullanımları sayesinde otonom robotlarda oldukça yaygın kullanım alanına sahiptirler.

Bu tez çalışmasında, sonar algılayıcılar ve sezgisel yöntemler kullanarak, otonom robotların yol bulma ve engelden sakınma probleminin hibrid bir şekilde çözümü sağlanmıştır. Bu kapsamda, robotların engelden sakınma ve yol bulma probleminin çözümüne ilişkin bir otonom robot seyir sistemi geliştirilerek, yeni algoritmaları bu sistem üzerinde gerçeklenmiştir.

6.1. Sonuçlar

Tez çalışması kapsamında, geliştirilen algoritmaların daha kolay test edilmesi amacıyla MATLAB de bir arayüz geliştirilmiştir. Geliştirilen arayüz aracılığı ile harita ve yol bulma ile ilgili parametreler değiştirilebilmekte, istenen engelden sakınma yöntemi seçilebilmekte, robot sonarları görsel olarak takip edilebilmekte, çeşitli senaryolar icra edilebilmektedir.

Geliştirilen algoritmalar benzetim ortamında test edildiği gibi, Kocaeli Üniversitesi Teknoloji Fakültesi laboratuarında bulunan Pioneer 3-DX robot üzerinde de başarılı bir şekilde çalıştıkları görülmüştür. Bu amaçla laboratuarda bir test ortamı oluşturulmuştur. Tavandaki bir kamera yardımı ile zemin fotoğrafı çekilmekte, fotoğraf üzerinde robot ve engellerin yerleri 10x8'lik bir haritaya işlenmekte ve harita bilgisi üzerinden yol bulma algoritması ile hedefe en kısa yol bulunmaktadır.

Geliştirilen algoritmalar ile ilgili ilk tespit, engelden sakınma olmaksızın sadece yol bulma yeteneği ile otonom bir robotun etkin bir görev icra edemeyeceğidir.

Gerçekleştirilen test senaryolarında, engelden sakınma yeteneği bulunmayan robot için, karşısına çıkan beklenmedik ilk engel durumunda çarpışma kaçınılmaz olmuştur.

Engelden sakınma algoritmalarından en yaygın olarak kullanılan böcek algoritmaları olan Böcek-1, Böcek-2 ve Dist-bug algoritmaları MATLAB ortamında yazılımsal olarak gerçeklenmiştir. Bu algoritmaların MobileSim benzetim ortamında ve Pioneer 3-DX robot üzerinde başarı ile çalıştığı görülmüştür. Bu üç algoritmanın çalıştırılması esnasında yapılan ölçümler paralelinde, Dist-bug algoritmasının daha iyi performans verdiği tespit edilmiştir.

Böcek algoritmalarının ortak yönü, karşılaşılan bir engelin etrafında duvar takibi yapılmasıdır. Tez çalışması kapsamında, duvar eğiminin tespit edilmesi yöntemi ile duvar takibi yapılmıştır. Bu yöntemde öncelikle engelin eğimi hesaplanmış ve bu eğime göre robot yönü ayarlanarak, engele paralel bir şekilde engel etrafından dolaşım sağlanmıştır. Bu şekilde yapılan duvar takip işlemi ile, rotanın sürekli olarak hedefe doğru çevrilmesi suretiyle rotanın ayarlandığı duvar takip yöntemine kıyasla gereksiz rota değişimlerinin engellendiği ve performansın arttırıldığı gözlemlenmiştir.

Düğüm noktaları üzerinden hedefe doğru ilerlerken, Dist-bug algoritmasında engelden ayrılma noktasından sonra, bir sonraki düğüm noktasına gitmekteydi. Bunun yerine, tez çalışmasında geliştirilen düğüm atlama özellikli iyileştirilmiş Dist-bug algoritmasında, sonar algılayıcılar yardımı ile engelsiz bir şekilde ulaşılabilecek, hedefe en yakın düğüm noktasına erişim imkanı sağlanmıştır. Bu şekilde performans açısından kazanç elde edildiği yapılan ölçümlerde görülmüştür.

Hiçbir engelden sakınma yeteneği olmadan sürekli olarak yol bulma algoritmasının çalıştırılması da senaryolar kapsamında denenmiştir. Haritanın oluşturulması ve oluşturulan haritanın düzenli aralıklar ile genetik algoritma ile çözülmesi sağlanmıştır. Bu senaryo sonucunda, yol bulma algoritmasının zaman açısından maliyetli bir süreç olduğu ve sürekli yol bulma hesaplanmasındansa, Dist-bug gibi bir engelden sakınma yönteminin kullanılmasının daha efektif olduğu tespit edilmiştir.

Hibrit bir yapıda kullanılan yol bulma ve engelden sakınma yeteneklerinin otonom bir robot için, hedefe ilerleme ve verilen görevin icrası açılarından büyük öneme sahip olduğu yapılan testler ve senaryolar ile görülmüştür. Geliştirilen robot seyir sistemi ile robotların otonom bir şekilde görevlerini icra edebilmeleri sağlanmıştır.

6.2. Öneriler

Tez çalışması kapsamında geliştirilen algoritmalar ve robot seyir sistemi gelecekte bu konuda çalışacak araştırmacıların çalışmalarına hız kazandıracaktır.

Elimizdeki robotta sadece sonar algılayıcılarının bulunması nedeniyle, diğer algılayıcı tiplerinin engelden sakınma konusundaki avantajları ele alınamamıştır. Robota eklenecek ilave özellikler ile daha etkin engelden sakınma yöntemleri geliştirilebilecektir.

KAYNAKLAR

[1] Bajracharya M., Maimone M. W., Helmick D., Autonomy for Mars Rovers: Past, Present, and Future, *Computer*, 2008, **41**(12), 44-50.

[2] Philippsen R., Siegwart R., Smooth and Efficient Obstacle Avoidance for a Tour Guide Robot, *IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, 14-19 September 2003.

[3] Dijkstra E. W., A Note On Two Problems in Connexion With Graphs, *Numerische Mathematik*, 1959, **1**(1), 269-271.

[4] Hart P. E., Nilsson N. J., Raphael B., A Formal Basis for The Heuristic Determination of Minimum Costpaths, *Systems Science and Cybernetics*, *IEEE Transactions on*, 1968, **4**(2), 100-107.

[5] Stentz A., Optimal and Efficient Path Planning for Partially-Known Environments, *IEEE International Conference on Robotics and Automation*, San Diego, California, 8-13 May 1994.

[6] Rimon E., Koditschek D. E., Exact Robot Navigation Using Artificial Potential Functions, *IEEE Transactions on Robotics and Automation*, 1992, **8**(5), 501-518.

[7] Holland J. H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Michigan, 1975.

[8] Lumelsky V., Skewis T., Incorporating Range Sensing in the Robot Navigation Function, *IEEE Transactions on Systems, Man and Cybernetics*, 1990, **20**(5), 1058–1068.

[9] Lumelsky V. J., Stepanov A. A., Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape, *Algorithmica*, 1987, **2**(4)**,** 403-430.

[10] Koren Y., Borenstein J., High-Speed Obstacle Avoidance for Mobile Robotics, *IEEE Symposium on Intelligent Control*, Arlington, Virginia (USA), 24-26 August 1988.

[11] Borenstein J., Koren Y., The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots, *IEEE Journal of Robotics and Automation*, 1991, **7**(3), 278–288.

[12] Khatib O., Quinlan, S., Elastic Bands: Connecting, Path Planning and Control, *IEEE International Conference on Robotics and Automation*, Atlanta, Georgia (USA), 2-6 May 1993.

[13] Simmons R., The Curvature Velocity Method for Local Obstacle Avoidance, *IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota (USA), 22-28 April 1996.

[14] Fox D., Burgard W., Thrun S., The Dynamic Window Approach to Collision Avoidance, *IEEE Transactions on Robotics and Automation*, 1997, **4**(1), 23–33.

[15] Philippsen R., Siegwart R., Smooth and Efficient Obstacle Avoidance for a Tour Guide Robot, *IEEE Int. Conf. on Robotics and Automation*, Taipei, Taiwan, 14-19 September 2003.

[16] Savage J., Matamoros M., Osorio R., Obstacle Avoidance Behaviors for Mobile Robots Using Genetic Algorithms and Recurrent Neural Networks, *IFAC Conference on Management and Control of Production and Logistics*, Ceara, Brazil, 1-13 September 2013.

[17] Stone P., Intelligent Autonomous Robotics: A Robot Soccer Case Study, *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2007, **1**(1), 1-155.

[18] Li S., *Mobile Robot Obstacle Avoidance Using Learning Classifier Systems*, University of Alberta, Edmonton, 2007.

[19] Zhu Y., Zhang T., Song J., Li X., A New Hybrid Navigation Algorithm for Mobile Robots in Environments with Incomplete Knowledge, *Knowledge-Based Systems*, 2012, **27**, 302-313.

[20] Xi L. , Byung-Jae C., Design of Obstacle Avoidance System for Mobile Robot Using Fuzzy Logic Systems, *International Journal of Smart Home*, 2013, **7**(3), 321-328.

[21] Faisal M., Hedjar R., Al Sulaiman M., Al-Mutib K., Fuzzy Logic Navigation and Obstacle Avoidance by a Mobile Robot in an Unknown Dynamic Environment, *International Journal of Advanced Robotic Systems*, DOI: 10.5772/54427.

[22] Lee J. H., Hashimoto H., Intelligent Space-Concept and Contents, *Advanced Robotics*, 2002, **16**(3), 265-280.

[23] Bell D. A., Borenstein J., Levine S. P., Koren Y., Jaros L., An Assistive Navigation System for Wheelchairs Based Upon Mobile Robot Obstacle Avoidance, *IEEE International Conference on Robotics and Automation*, San Diego, California (USA), 8-13 May 1994.

[24] Shoval S., Borenstein J., Koren Y., Mobile Robot Obstacle Avoidance in A Computerized Travel Aid for the Blind, *IEEE International Conference on Robotics and Automation*, San Diego, California (USA), 8-13 May 1994.

[25] Conte G., Zanoli S., A Sonar Based Obstacle Avoidance For AUVS, *Autonomous Underwater Vehicle Technology (AUV 94)*, Cambridge, Massachusetts (USA), 19-20 July 1994

[26] Lee Y. I., Kim Y. G., Kohout L. J., An Intelligent Collision Avoidance System for AUVS Using Fuzzy Relational Products., *Information Sciences*, 2004, **158**, 209-232.

[27] Schultz A. C., *Using a Genetic Algorithm to Learn Strategies for Collision Avoidance and Local Navigation*, Naval Research Lab, Washington DC (USA), 1990.

[28] Hockstein N. G., Gourin C. G., Faust R. A., Terris D. J., A History Of Robots: From Science Fiction to Surgical Robotics, *Journal of Robotic Surgery*, 2007, **1**(2), 113-118.

[29] Čapek K., *RUR (Rossum's Universal Robots)*, Penguin, London, 2004.

[30] Romdhane L., Zeghloul S., AL-JAZARI (1136–1206), *Distinguished Figures in Mechanism and Machine Science*, 2009, **7**, 1-21.

[31] Kumar V., 50 Years of Robotics [From the Guest Editors], *IEEE Robotics & Automation Magazine*, 2010, **17**(3), 8-8.

[32] Murphy, R., *Introduction to AI Robotics*, MIT press, Cambridge, 2000.

[33] IFR Statistical Department, World Robotics Report 2015, http: //www.worldrobotics.org, (Ziyaret tarihi: 5 Ocak 2016).

[34] KUKA-robotics, http://www.kuka-robotics.com, (Ziyaret tarihi: 5 Ocak 2016).

[35] Robot ASIMO, http://asimo.honda.com, (Ziyaret tarihi: 10 Ocak 2016).

[36] Raibert M., Blankespoor K., Nelson G., Playter R., Team T. B., Bigdog, The Rough-Terrain Quadruped Robot, *17th IFAC World Congress*, Seoul, Korea, 6-11 July 2008.

[37] https://www.nasa.gov/mission_pages/msl/index.html, (Ziyaret tarihi: 11 Ocak 2016).

[38] http://www.km.kongsberg.com, (Ziyaret tarihi: 12 Ocak 2016).

[38] http://www.ga.com/unmanned-aircraft-systems-and-sensors, (Ziyaret tarihi: 16 Ocak 2016).

[39] Burguera A., González Y., Oliver G., Sonar Sensor Models and Their Application To Mobile Robot Localization, Sensors, 2009, **9**(12), 10217-10243.

[40] Carmer D. C., Peterson L. M., Laser Radar in Robotics, *Proceedings of the IEEE*, 1996, **84**(2)**,** 299-320.

[41] Gibilisco S., *Concise Encyclopedia of Robotics*, McGraw-Hill, New York, 2002.

[42] Alp O., Gezgin Robotlarda Eş Anlı Haritalama ve Konum Belirleme, Doktora Tezi, Başkent Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 2008, 216718.

[43] Elfes A., Using Occupancy Grids for Mobile Robot Perception and Navigation, *Computer*, 1989, **22**(6), 46-57.

[44] Guan-Zheng T., Huan H., Sloman A., Ant Colony System Algorithm for Real-Time Globally Optimal Path Planning of Mobile Robots, *Acta Automatica Sinica*, 2007, **33**(3), 279-285.

[45] Raja P., Pugazhenthi S., Optimal Path Planning of Mobile Robots: A review, *International Journal of PhysicalSciences*, 2012, **7**(9), 1314-1320.

[46] Masehian E., Sedighizadeh D., Classic and Heuristic Approaches in Robot Motion Planning-a Chronological Review, *World Academy of Science, Engineering and Technology*, 2007, **29**(1), 101-106.

[47] AStar tutorial, http://www.policyalmanac.org/games/aStarTutorial.htm, (Ziyaret tarihi: 16 Nisan 2016).

[48] Dorigo M., Maniezzo V., Colorni A., *The Ant System: An Autocatalytic Optimizing Proces, Technical Report 91-016*, Dipartimentodi Elettronica, Politecnicodi, Milano, 1991.

[49] Perretto M., Lopes H. S., Reconstruction of Phylogenetic Trees Using the Ant Colony Optimization Paradigm, *Genetics and Molecular Research*, 2005, **4**(3), 581-589.

[50] Dorigo M., Stützle T., Ant Colony Optimization: Overview and Recent Advances, *Handbook of Metaheuristics*, 2010, **146**, 227-263.

[51] Yıldırım M., Genetik Algoritmalar ve Benzetilmiş Tavlama ile Uzun Dönem Üretim Genişletme Planlaması, Doktora Tezi, Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Kocaeli, 2003, 135971.

[52] Nourbaskhsh I. R., Siegwart R., *Introduction to Autonomous Mobile Robots*, MIT Press, Cambridge, 2011.

[53] Yufka A., Parlaktuna O., Performance Comparison of Bug Algorithms for Mobile Robots, *International Symposium on Innovations in Intelligent Systems and Applications* (INISTA'09), Trabzon, Türkiye, 29 Haziran-1 Temmuz 2009.

[54] Ng J., Bräunl T., Performance Comparison of Bug Navigation Algorithms. *Journal of Intelligent and Robotic Systems*, 2007, **50**(1), 73-84.

[55] Goodrich M.A., *Potential Fields Tutorial*, Brigham Young University, Brovo Utah (USA), 2002.

[56] Koren Y., Borenstein J., Potential Field Methods and Their Inherent Limitation For Mobile Robot Navigation, *IEEE International Conference on Robotics and Automation*, 1991, **2**, 1398-1404.

[57] http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx (Ziyaret tarihi: 6 Nisan 2016).

[58] MobileRobots Inc., *Pioneer 3 Operations Manual*, Version 6, September 2010.

[59] Tuncer A., Otonom Araçlar İçin Yol Bulma Probleminin Genetik Algoritmalar ve FPGA Ile Çözümü ve Gerçekleştirilmesi, Doktora Tezi, Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Kocaeli, 2013, 335451.

[60] MobileRobots Inc., *Pioneer 3-DX Datasheet*, 09366- Rev A., 2010.

[61] http://robots.mobilerobots.com/wiki/Aria_Matlab_Interfaces (Ziyaret tarihi:6 Nisan 2016).

[62] http://www.mobilerobots.com/software/mobilesim.aspx (Ziyaret tarihi: 1 Nisan 2015).

[63] http://www.mathworks.com/help/matlab/matlab_prog/use-a-matlab-timer-object.html (Ziyaret tarihi: 1 Nisan 2015).

[64] http://www.camstudio.org/ (Ziyaret tarihi: 10 Haziran 2014).

KİŞİSEL YAYIN VE ESERLER

[1] **Topuz R.**, Yıldırım M., Sonar Algılayıcılar ve Sezgisel Yöntemler İle Otonom Robotlarda Engelden Sakınım ve Yol Bulma, *El-Cezeri Journal of Science and Engineering*, 2016, **3**(2), 363-371

ÖZGEÇMİŞ

1974 yılında Gölcük'de doğdu. İlk ve orta öğrenimini Kocaeli'de tamamladı. 1996 yılında Deniz Harp Okulu Kontrol ve Bilgisayar Mühendisliği bölümünden mezun oldu. 2002 yılında Naval Post Graduate School Monterey/Kaliforniya, ABD'de Bilgisayar Bilimleri (Computer Science-CS) ve Bilgi Teknolojileri Yönetimi (Information Technology Management-ITM) dallarında Yüksek Lisans öğrenimini tamamladı. Halen Deniz Kuvvetleri Komutanlığı'nda görevine devam etmektedir. Evli ve 2 çocuk babasıdır. İngilizce bilmektedir.