T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

LABİRENTLERDE YAPAY ZEKA TABANLI YÖN BULMA ALGORİTMALARI KULLANAN BİR GEZGİN ROBOT GELİŞTİRİLMESİ

AYDIN GÜLLÜ

DOKTORA TEZİ

MAKİNE MÜHENDİSLİĞİ ANABİLİM DALI

Tez Danışmanı: Doç. Dr. Hilmi KUŞÇU

EDİRNE-2017

T.Ü. Fen Bilimleri Enstitüsü onayı

Prof. Dr. Murat YURTCAN
Fen Bilimleri Enstitüsü Müdürü

Bu tezin Doktora tezi olarak gerekli şartları sağladığını onaylarım.

Prof. Dr. Ayşegül ÖZTÜRK
Anabilim Dalı Başkanı

Bu tez tarafımca okunmuş, kapsamı ve niteliği açısından bir Doktora tezi olarak kabul edilmiştir.

Doç. Dr. Hilmi KUÇU
Tez Danışmanı

Bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından Makine Mühendisliği Anabilim Dalında bir Doktora tezi olarak oy birliği ile kabul edilmiştir.

Jüri Üyeleri(Unvan, Ad, Soyad)	İmza
Prof. Dr. Nihat AKKUŞ	.....
Prof. Dr. Selim KARA	.....
Doç. Dr. Hilmi KUŞÇU	.....
Doç. Dr. Sezgin ERSOY	.....
Yrd. Doç. Dr. Cenk MISIRLI	.....

Tarih: 17/01/2017

T.Ü. FEN BİLİMLERİ ENSTİTÜSÜ
MAKİNE MÜHENDİSLİĞİ DOKTORA PROGRAMI
DOĞRULUK BEYANI

İlgili tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin kaynak gösterilerek ilgili tezde yer aldığını beyan ederim.

17.01.2017

Aydın GÜLLÜ

Doktora Tezi

Labirentlerde Yapay Zeka Tabanlı Yön Bulma Algoritmaları Kullanan Bir Gezgin Robot Geliştirilmesi

Trakya Üniversitesi Fen Bilimleri Enstitüsü

Makine Mühendisliği Anabilim Dalı

ÖZET

Bu tez kapsamında, otonom çalışan bir gezgin robot ile daha önceden bilinmeyen gerçek bir ortamın tanımlanması sağlanacaktır. Ortamın tanımlanmasından sonra, ortamdan seçilen iki nokta arasındaki en kısa yolun bulunması için çeşitli yazılımlar geliştirilmiştir.

Bilgisayar üzerinde gerçekleştirilen yazılım ile ortamın analizi için yapay zeka tabanlı arama ve çözümleme fonksiyonları geliştirilmiştir. Ortam olarak çizgi labirent kullanılmıştır. Analiz için, duvar takibi, derinlik öncelikli arama, genişlik öncelikli arama algoritmaları ile testler yapılmıştır. Ortamın gerçek zamanlı taranmasında derinlik öncelikli arama ve genişlik öncelikli arama algoritmaları, Dijkstra en kısa yol algoritması ile bütünleşik çalıştırılmıştır. Bu hibrid çalışma gerçek ortamın keşfi için daha verimli sonuçlar vermektedir. Ortamın keşfi sonucunda tüm labirent grafik yapısına dönüştürülmüştür. Labirentin tanımlanması geliştirilen algoritmalar ile yapılabileceği gibi, labirentin görüntüsünün analizi ile de yapılabilmektedir. Bunun için görüntü işleme yazılımı geliştirilmiş ve labirentin yapısı bilgisayara aktarılmıştır. Robot, labirent üzerinde seçilen bir hedef nokta için en kısa mesafeyi kullanarak ulaşabilmektedir. Bu işlem için A Yıldız (A*) veya açgözlü en iyi öncelikli arama algoritmaları kullanılmıştır. Bu algoritmalardan A* yapılan testlerde her zaman en kısa yolu vermiştir. Test labirentlerinin oluşturulması derinlik öncelikli arama algoritması ile rastgele yapılmıştır.

Yıl : 2017

Sayfa Sayısı : 82

Anahtar Kelimeler : Gezgin Robot, Labirent Analizi, Yapay Zeka, Mekatronik Sistem, Makine Mühendisliği

Doctoral Thesis
Development of Mobile Robot Based on Artificial Intelligence for Navigation Algorithms in Mazes
Trakya University Institute of Natural Sciences
Department of Mechanical Engineering

ABSTRACT

In this thesis, a previously unknown real environment will be able to identify with an autonomous mobile robot. After the environment is identified, the shortest path between the two selected points will be found.

Artificial intelligence based search and solve functions have been developed for the analysis of the environment with the designed software on the computer. Line maze was used as environment. For analysis, wall-followed, breadth first search algorithms are used. In real-time scanning of the environment, depth-first search and breadth-first search algorithms are integrated with the Dijkstra shortest path algorithm. This hybrid work gives more efficient results for the discovery of the real environment. As a result of the discovery of the environment, the whole maze has been transformed into a graph structure. The identification of the maze can also be done by analyzing the image of the maze. Image processing software has been developed for this and the maze structure has been transferred to the computer. The robot is able to reach the selected target point on the maze using the shortest distance. A star (A *) or greedy best-prioritized search algorithms are used to find the shortest path. It has been observed that A * always gives the shortest path as a result of the tests. The test mazes were randomly generated with a depth-first search algorithm.

Year : 2017
Number of Pages : 82
Keywords : Mobile Robot, Maze Solving, Artificial Intelligence, Mechatronic System, Mechanical Engineering

ÖNSÖZ VE TEŞEKKÜRLER

Robotlar, teknolojinin gelişmesi ile hayatımızda daha fazla yer almaktadır. Kendi kedilerine çalışan, karar veren robotlar, belirli bir işi yapmak için insanların en büyük yardımcısı olarak gelecekte de kullanılacaktır. Otonom çalışan bu robotlar için yazılımlar ve donanımlar geliştirilmektedir. Bu çalışma kapsamında da otonom çalışan gerçek bir gezgin robot için ortamın keşfedilmesi ve en kısa yol probleminin çözülmesi için bazı yazılımlar geliştirilmiştir. Geliştirilen bu yazılımlar uygulanarak test edilmiştir. Hayatımıza giren ve otonom olarak çalışan, elektrik süpürgesi, çim biçme makinesi, mayın tarama robotu gibi birçok robotun alan taramada temelini oluşturan bu yazılımlar, ticari robotlara uygulanarak, insanlar için daha faydalı işlerin yapılabilmesi sağlanabilir.

Bu temenniler ile;

Görüntü işleme aşamasında vermiş olduğu destek için mesai arkadaşım M. Ozan AKI'ya, akademik çalışmalarımı destekleyen okul müdürüm Yrd. Doç. Dr. Hayati ARDA'ya Teşekkür Ederim.

Bu yolda bana desteklerini esirgemeyen, her zaman yanımda olan en büyük destekçim eşim Didem GÜLLÜ'ye enerji kaynağım kızım Duru GÜLLÜ'ye teşekkür etmeyi bir borç bilirim.

Eğitim hayatım boyunca her türlü desteği veren ve üzerimde büyük emekleri olan Anne ve Babama Teşekkür Ederim.

Beni bu tez ile ilgili çalışmaya yönlendiren, tez çalışmalarım boyunca her zaman yanımda olan, gösterdiği bilgi, tecrübe ve pratik çözümler ile ufkumu açan danışmanım Doç. Dr. Hilmi KUŞÇU'ya Teşekkürlerimi Sunarım.

Tez çalışmalarımı yürütebilmem için sağlamış olduğu maddi destekten ötürü Trakya Üniversitesi Bilimsel Araştırmalar Projeler Birimine (Proje No: 2014/04) Teşekkürlerimi Sunarım.

Aydın GÜLLÜ

İÇİNDEKİLER

SİMGELER LİSTESİ

x, y	:Eksen takımı
θ	:Robotun dönme açısı
v	:Tekerlek hızı (v_R: Sağ teker için, v_L:Sol teker için)
r	:Tekerlek yarıçapı
ω	:Tekerlek açısal hızı (ω_R: Sağ teker için, ω_L:Sol teker için)
r_c	:Robot yarıçapı
G	:Grafik (Graph) Veri Tipi
V	:Düğüm (Vertex), Her son nokta ya da ara geçiş alanları
E	:Kenar (Edge), iki düğüm arasındaki mesafe
$h(n)$	:Sezgisel tahmin n durumundan hedef duruma kadar
$g(n)$	:n durumunda hedefe kadar yol maliyeti
$f(n)$	:n durumunda hedef duruma kadar hesaplanmış sezgisel fonksiyon
P	:Oransal denetleyici
I	:İntegral denetleyici
D	:Türevsel denetleyici
K_P	:Oransal denetleyici kazanç katsayısı
K_I	: İntegral denetleyici kazanç katsayısı
K_D	:Türevsel denetleyici kazanç katsayısı
$e(t)$	:Hata değeri
L	:Gezgin robotun sola dön komutu
R	:Gezgin robotun sağa dön komutu
S	:Gezgin robotun düz git komutu
B	:Gezgin robotun geri dönüş komutu
T	:Görüntünün siyah beyaz geçişi için eşik değeri

KISALTMALAR LİSTESİ

2D	:İki boyutlu
3D	: Üç boyutlu
BFS	: Genişlik Öncelikli Arama (Breadth first search)
DFS	: Derinlik Öncelikli Arama (Depth-First Search)
GBFS	: Açgözlü öncelikli arama (Greedy Best First Search)
BestFS	: En iyi öncelikli arama
A*	:A Yıldız (A Star)
MLM	: Değiştirilmiş Çizgi Labirent (Modified Line-Maze)
PSO	: Parçacık Sürü Optimizasyonu
GA	:Genetik algoritma
NSGA	:Sıralama olmayan genetik algoritma
YSA	: Yapay Sinir Ağları (Neural Networks NN)
GRNN	: Genel Regresyon Yapay Sinir Ağları (GRNN)
RF	: Radyo Frekansı
pdf	: Taşınabilir Belge Biçimi (Portable Document Format)
PWM	:Darbe Genişlik Modülasyonu
ROS	: Robot işletim sistemi

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

BÖLÜM 1

GİRİŞ

1.1. Konu

Robot, istenilen görevleri yerine getirmek için tasarlanmış mekatronik sistemdir. Robotlar önceden programlanabileceği gibi otonom olarak çalışarak da istenilen görevleri yerine getirebilirler. Robotlar için çeşitli sınıflandırmalar vardır. Bunlar kullanıldığı yer, robotun sabit veya hareketli oluşu ya da çalıştığı ortam olabilir. Hareketli robotlar gezgin veya mobil robot olarak isimlendirilmektedir. Belirli bir ortamda hareket ederek istenilen görevleri yeri getirebilirler. Bu robotların çalıştığı ortam ise hava, kara ve su olabilir. Geliştirilen bazı özel robotlar ise bu ortamların birkaçında çalışabilecek şekilde tasarlanmıştır. Mobil robot yapı itibari ile mekanik bir şasi, hareket etmesi için eyleyici, çevresini algılayan algılayıcılar ve tüm bu donanımı kontrol eden denetleyici elemandan oluşur. Kullanılacağı iş için tasarlanan mekanik şasi üzerine tüm donanımlar eklenir. Hareketi sağlayan eyleyici olarak çoğunlukla doğru akım elektrik motoru kullanılır. Robotun otonom olarak hareket etmesi için bir karar mekanizmasına gerek vardır. Bu işlem bir denetleyici eleman tarafından sağlanır. Denetleyici olarak, mikrodenetleyici entegreler, bilgisayarlar ve özel tasarlanan bütünleşik devreler kullanılabilir. Bu kontrolörler mobil robot üzerinde bulunan motorları kontrol ederek hareketi sağlayabilir. Hareketin verimli ve düzgün olabilmesi için ortamdan bazı sinyaller alınması gerekmektedir. Robotun yönünün, konumunun, etrafındaki nesnelerin algılanması için çeşitli algılayıcılar geliştirilmiştir. Robotun yerine getireceği işleme göre seçilen algılayıcılar şasi üzerine monte edilir. Robot üzerinde bulunan denetleyici ise verilen görev çerçevesinde algılayıcılardan aldığı bilgileri yorumlayarak motorları hareket ettirir. Yukarıda sayılan donanımlar bir mobil robotun otonom çalışabilmesi için gereklidir.

Verilen görevin karmaşık olması robotun çalışmasını zorlaştırır. Bu durumda robotun donanımlarının kontrolü için mikrodenetleyici üzerinde iyi bir yazılım algoritmasının geliştirilmesi gerekmektedir. İnsan gibi, karşılaştığı durumlardan mantıksal veya öğrenilmiş kazanımlarla çıkarımlar üretmesi gerekmektedir. Bu kapsamda yapay zeka kavramı ortaya çıkmaktadır. İnsan gibi hareket etme veya davranmak amacı ile çeşitli yapay zeka algoritmaları geliştirmiştir.

Bu tez kapsamında gezgin tip bir robotun otonom olarak hareket etmesi incelenecektir. Robotun önceden bilmediği bir ortamda kullanacağı yapay zeka algoritmaları ile ortamı tanımlaması ve yönünü bulması sağlanmıştır.

1.2. Amaç

Bu çalışmada, otonom olarak mobil robotlar tarafından önceden bilinmeyen bir ortamın tanımlanması amaçlanmıştır. Ortamın haritasının çıkartılması için yapay zeka tabanlı algoritmalar gerçek mobil robot üzerinde kullanılabilecek şekilde tasarlanmıştır. Bu algoritmaların verimlikleri test edilerek, alan tarama faaliyetlerinin daha az mesafe harcanarak yapılabilmesi için algoritmaların optimize edilmesi amaçlanmıştır. Ortamın tanımlanmasında gerçek robot ile taramanın yanı sıra görüntü işleme ile ortamın algılanması için bir yazılım geliştirilmesi hedeflenmiştir. Nihai olarak ortamın algılanmasından sonra belirtilen iki nokta arasına en kısa yoldan gidilebilmesi için rota planlanması amaçlanmıştır.

1.3. Kapsam

BÖLÜM 1.2'deki belirtilen amaçları gerçekleştirebilmek için ayrıntılı bir literatür taraması yapılmıştır(BÖLÜM 2). Literatür taraması bilgisayar yazılım ve mobil robotlar için iki farklı alanda ayrı ayrı yapılarak en uygun çalışmalar incelenmiştir.

BÖLÜM 3'de belirten amaç ve kaynak araştırmasında bulunan bilimsel bulgular çerçevesinden tez kapsamında kullanılacak yöntem ve fiziksel donanımlar açıklanmıştır. Donanım olarak gerçek bir mobil robot geliştirilmiş ve fiziksel test ortamı seçilmiştir. Sonrasında hem mobil robot için hem de çözüm için kullanılacak yazılımlardan bahsedilmiştir. Bu bölümde yapay zeka yazılımlarından grafik arama ve sezgisel arama algoritmaları anlatılmıştır.

BÖLÜM 4’de tezin uygulanması ve uygulama sonucunda elde edilen veriler incelenmiştir. Uygulama aşamasında robotun ve bilgisayar yazılımlarının çalışması detaylı olarak anlatılmıştır. Tez kapsamında tanımlanan algoritmalar çalıştırılmıştır. Test sonuçları yine bu bölümde sunulmuştur.

Elde edilen verilerin sonuçları ve genel sonuçlar BÖLÜM 5’de ele alınmıştır. Ayrıca bu bölümde uygulama alanları ve gelecek çalışmalara değinilmiştir.

BÖLÜM 2

KAYNAK ARAŞTIRMASI

Yapay zeka ile yapılan çalışmalara 50'li yıllarda başlanmıştır[1-2]. Yapay zeka algoritmalarının mobil robotlar için uygulanması ise 80'li yılları bulmuştur. Bu tez kapsamında yapılan literatür taramasında mobil robotların önceden bilmediği ortamlarda yön bulmaları ile ilgili çalışmalar çeşitli veri tabanlarında taranmıştır. Yapılan araştırmalarda kullanılan anahtar kelimeler mobil robot, yapay zeka, robot yörünge planlama, bilinmeyen ortamlarda yön bulma olarak seçilmiştir. Tez kapsamında yapılan araştırma iki kapsamda incelenmiştir. Birinci araştırma mobil robotların olduğu ve mobil robotların yön bulma ve yörünge planlanmada kullandığı yöntemlerdir. Ek olarak mobil robotlar için geliştirilen bilinmeyen ortamlarda yapay zeka algoritmaları ile yön bulma için yapılan çalışmalar da incelenmiştir. İkincil olarak yapay zeka kavramı ve yapay zeka yazılımları araştırılmıştır. Bu kapsamda robotlara literatürde uygulanan yapay zeka yazılımları ve diğer uygulanabilecek yazılımlar incelenmiştir. 80'li yıllardan günümüze kadar yapılan çalışmalar taranarak derlenmiştir. Tez konusuna yakın olarak yapılan çalışmalar aşağıdaki gibi özetlenmiştir.

Corowley'in çalışmasında, robotların yön bulması için algılayıcı teknolojisinin önemi vurgulanmıştır. Bir mobil robot geliştirilmiş ve üzerine dönen ultrasonik mesafe algılayıcısı monte edilmiştir. Yön bulma algoritması, ön öğrenme ve çalışma olmak üzere iki aşamadadır. Ön öğrenme, algılayıcılardan alınan bilgiler ile sağlanmıştır, çalışma esnasında kısa sürede yön tayini yapabilmektedir [3].

Zelinsky bilinmeyen ortamlarda hedefi bulmak için bir algoritma geliştirmiştir. Ortamın algılanması temaslı bir algılayıcı vasıtası ile algılanmış ve haritalama yapılmıştır.

En kısa yolun bulunması için robotun tüm ortamı taraması ve bir optimizasyon yapması için algoritma geliştirilmiştir. Optimizasyon için kaçınma optimizasyonu seçilmiş ve dördün ağaç veri yapısı kullanılmıştır[4].

Rao, bilinmeyen ortamlarda yön bulma stratejilerinin ele alındığı bu çalışmada iki metot üzerinde durmaktadır. Birinci metot robotun sezgisel olarak yön tayin etmesidir. İkincisi ise geliştirilen algoritmalar vasıtası ile yön tayinidir. Bu çalışmada bir labirent için çıkış yolunun belirlenmesi deneme yanılma yolu ile tespit edilmiş ve tekrarlanma problemi en aza indirilerek iyi bir çözüm geliştirilmeye çalışılmıştır[5].

Oommen ve benzer olarak Rao ve arkadaşlarının yaptığı bilinmeyen ortamlarda robot yön bulma algoritmalarının denendiği bu çalışmalarda sezgisel olmayan metotlar üzerine çalışılmıştır. Çalışmada algılayıcı ve görüntü tabanlı giriş bilgileri ile bir yön bulma sağlanmış ve en kısa yön üzerine bir çalışma yapılmıştır. Bu çalışmalarda literatürde bulunan labirent tarama algoritmalarından yararlanılmıştır [6-7].

Liscano ve arkadaşları, bir mobil robotun tanımını ve kullanım amacının; verilen bir problemi çözüm getirilebilecek bir kabiliyette olması olarak tanımlamışlardır. Bu tanım çerçevesinden değişken koşullarda hedefe varabilmek için bir algoritma geliştirilmiştir[8].

Botelho ve arkadaşları, robotlarda yön bulma algoritmalarında yapay sinir ağları kullanmışlardır. Temel bir yapay sinir ağı yapısı kullanan çalışma, mantıksal işlemlerin yapılmasından oluşmaktadır. Çalışma fiziksel bir sistemde kullanılmıştır. Çalışma ayrıca bulanık mantık ve çok katmanlı sinir ağları ile karşılaştırılmıştır[9].

Lee ve arkadaşları, mobil robotlarda bulanık bir yön bulma stratejisini bulanık mantık kontrol algoritması ile çözümlemiştir. Çözümlemede çeşitli engellerin olduğu labirent benzeri bilinmeyen bir ortamda başlangıç hedefinden başlayıp sonuca gitmesi incelenmiştir. Burada engellerin dış bükey olmasının çözüm için en kısa yolu bulmasında verimli sonuçlar verdiğinden bahsedilmektedir. İç bükey olması ise çözümde zorluk sağlamaktadır. Problemin çözümü için bulanık mantık kontrol algoritması ile teğet (tanjant) algoritması kullanılmıştır [10].

Lu ve arkadaşları, bu çalışmada yön bulma problemini iki boyutlu (2D) lazer mesafe algılayıcısı ile çözmüşlerdir. Çözümde sadece mesafeyi algılamak, engel şekillerinin farklı açı ve yönlerde olmasıyla zorluk oluşturmuştur. Bu sebeple çalışmada mevcut ve bir önceki konumlar arasında bir ilişki kurularak bir algoritma oluşturulmuştur.

Böylelikle bilinmeyen ortamlarda 2D mesafe algılayıcısı ile pozisyon bulma sağlanmıştır[11].

Dain, genetik algoritma tabanlı duvar takibi yapan mobil robot algoritmasının uygulandığı çalışmasını bilgisayar ortamında çalıştırmıştır. Çalışma genetik algoritmalar hakkında bilgi verildikten sonra duvar takip eden mobil robotlar incelenmiştir. Sonunda duvar takibi işlemi genetik algoritmalar ile çözümlenmiştir[12].

Ryu, yörünge planlamada farklı bir yaklaşımın sergilendiği çalışmasında; ısı dağılımlarından faydalanarak robotun hedefe ulaşması sağlanmıştır. Yön bulma probleminde başlangıç durumunu; ısı kaynağını, hedefi; ısı işareti, engelsiz alan; ısıl iletken, engelleri; ısıl yalıtkan ve optimal yolu; ısı yolu olarak tanımlamıştır. Tüm formüllerini ısıl olarak düzenlemiş ve çözümlemiştir[13].

D++ diye yeni bir algoritmanın tanımlandığı çalışmada, robotlarda yörünge bulma probleminin çözümü amaçlamaktadır. Dijkstra'nin algoritmasında üretilerek oluşturulan bu algoritmanın performansı A* algoritması ve D* algoritmaları ile karşılaştırılmalı olarak ortaya konulmuştur. Hem simülasyon hem de gerçek mobil robot üzerinde denenen çalışma hareketli engellerde çabuk tepki vermesi bakımından olumlu sonuçlar vermiştir[14].

Birden fazla nesne bulunan ortamda hedefe ulaşmak için bir algoritma geliştirilmiştir. Çözüm için hedefe uzaklık ve yolun açıklığı temel alınmıştır. Nesnelerin şekilleri birbirlerinden farklı olduğu ve hedefe ulaşmak için birden fazla yol olduğu çalışmada, Genetik algoritma tabanlı sıralama olmayan genetik algoritma 2 (NSGA II) algoritması kullanılmıştır. Elde edilen sonuçlar A*(A yıldız) ve PSO(parçacık sürü optimizasyonu) yöntemleri ile karşılaştırılmıştır [15].

Bu çalışmada geometrik bir yaklaşım ile engelden kurtularak hedefe ulaşması için yörünge planlaması yapılmıştır. Veriler çevrim içi (on-line) olarak işlenmiştir. Deneysel çalışma Matlab ortamında simülasyon ile yapılmıştır. Yapılan algoritma dış bükey, dış bükey olmayan ve labirent olmak üzere üç farklı ortam için uygulanmıştır[16].

Taşırma (Flood-Fill) algoritması ile labirent çözen mobil robot uygulaması ve gerçekleştirilmesi isimli çalışma, literatürde bu algoritmanın mobil robotlar için kullanıldığı ilk çalışmalarından biridir. Ultrasonik algılamanın yapıldığı çalışma duvar labirent çözümü yaptırılmıştır.

Labirentin öğrenmesinin de gerçekleştirildiği çalışmada en kısa yol bulunarak problem çözdürülmüştür. Mobil robot, Arduino Uno ile yapılmış ve denenmiştir[17].

Birden fazla robotun yol planlamasının yapıldığı çalışmada çözüm genetik algoritmalar ile sağlanmıştır. Java tabanlı gerçekleştirilen uygulamalarda birçok veri elde edilmiş ve sonuçlar karşılaştırılmıştır. Çalışmada genetik algoritmanın kullanılması ve kooperatif çalışma yapılmıştır[18].

Görüntü işleme tabanlı gerçek zamanlı çizgi takibinin yapıldığı yüksek lisans tezinde bir mobil robot yapılmıştır. Denemeler Virtual DSP 5.0 tabanlı bir platformda hazırlanan algoritmalar ile sağlanmıştır. Bu tezde, görüntü çeşitli işlemlerden geçirilip çizginin bulunması ve mobil robotun çizgi takip edebilecek şekilde yönlendirilmesini içermektedir[19].

Mae'nin çalışmasında, konsept olarak bir yangın söndürme robot tasarımı yapılmıştır. Robot Atmel entegresi ile uygulanmış ve yanan mumları söndürmek üzerine programlanmıştır. Programda bir labirent kullanılmış ve labirentte hedefe ulaşmak için değiştirilmiş çizgi labirent (Modified Line-Maze -MLM) algoritması ve derinlik öncelikli arama (Depth First Search - DFS) algoritmalarından yararlanılmıştır. Çalışmada ultrasonik algılayıcı, pusula gibi birçok algılayıcı kullanarak denemeler yapılmış ve geçmiş sonuçlarla karşılaştırılmıştır. Yangını bulma ve söndürme alanlarında iyi sonuçlar verebilmektedir [20].

Bir mobil robot tasarımı PIC ile yapılmıştır. BUG algoritması kullanılarak engelleri aşması sağlanmış ve hedefe ulaştırılmıştır. Hedefe ulaşan robotun hedef noktadaki sıcaklık bilgisinin RF ile istenilen noktaya gönderilmesi sağlanmıştır [21].

Enine arama algoritması(BFS) kullanılan çalışma Lego MindStorm RIS mobil robotu üzerinde denenmiştir [22].

Genel labirentler için yol planlanmasının yapıldığı tezde, karınca koloni optimizasyonunu, yapay sinir ağlarını ve A* algoritması ayrı ayrı denenmiştir. Sonuçlar bilgisayar ortamında farklı labirentlere uygulanarak farklı sonuçlar bulunmuştur. Bu algoritmalar sırası ile düzenlenerek en iyi sonuçlar için test edilmiştir [23].

2D lazer mesafe algılayıcısı kullanılan çalışmada, bilinmeyen bir ortamda çizgi tanımlama yöntemi ile yön bulma stratejileri geliştirilmiştir. Yön bulma algoritmasında Hough dönüşümü ve özyinelemeli bölünmüş biçimcilik yöntemlerinden yararlanılmıştır.

NAJI II kurtarma robotu ile gerçek zamanlı olarak denenmiş olan algoritma başarılı bir şekilde işletilmiştir [24].

Song ve arkadaşları, kendi kendine yön bulan bir mobil robot algoritması geliştirmişlerdir. Geri yayılım yapay sinir ağları algoritmasının kullanıldığı yapay zeka tabanlı bu çalışmada bir makine öğrenmesi kullanılmıştır. Bilinmeyen ortamlarda kendi kendine yön ve hız ayarlayan bir sistem geliştirilmiştir. Denemeler fiziksel sisteme uygulanmış ve başarılı sonuçlar elde edilmiştir [25].

Algılayıcı tabanlı bir yön bulma algoritması kullanılan çalışmada, LIDAR algılayıcısından alınan uzaklık bilgisi engellerden kaçma ve yörünge takibi için işlenmiştir. Uzaklık miktarı en fazla olan yer ve yön seçimi için kullanılmıştır [26].

Yeni hibrid yön bulma algoritması, önceden bilinmeyen bir ortam için kullanılmıştır. Klasik yön bulma algoritmalarına ek olarak iki katman oluşturulmuş, bunlardan birisi bilinçli (düşünen) katman, diğeri ise algılayıcı katmanıdır. Algılayıcı katman sürekli algılayıcılar ve bilinçli katman ile iletişim halinde olup çıkışları kontrol etmektedir. Bilinçli katman algılayıcılardan aldığı bilgileri değerlendirerek değişen çevre şartlarına göre algılayıcı katmana bilgi akışı sağlamaktadır. Çalışmada farklı çevre şartlarına göre çeşitli denemeler yapılmıştır. Deneylerin simülasyonu yapılarak sonuçlar karşılaştırılmıştır. Bilinçli katman burada A* algoritmasını, duyarlı katman ise DH-Bug algoritmasını kullanmıştır [27].

“Voronoi Fast Marching” yol planlaması metoduna gürbüz bir algoritma uygulayarak bir yön bulma stratejisi denenmiştir. Lider yörünge takibi yaptırarak sonuçları simülasyonda incelenmiştir. Lider yörünge takibinde “Fast Marching” algoritması kullanılmıştır [28].

Melendez ve arkadaşları bulanık mantık yöntemi ile yön bulma algoritması tanımlanmışlardır. Yapılan testler Matlab Mobile Robot araç kutusu yardımı ile denenmiştir [29].

Gerçek bir farenin yön bulma çalışmalarının robota nasıl aktarılacağı ile ilgili bazı çalışmalar yapılmıştır. Bir fare üzerine elektronik devre monte edilmiş ve insan ile etkileşimi sağlanmıştır. Sonraki aşamada insan faktörü yerine bir yapay zeka algoritması eklenmiştir. (Genel Regresyon YSA- GRNN)[30].

”A hybrid navigation strategy for multiple mobile robots” isimli çalışma birden fazla robotun aynı çalışma ortamında yön bulma stratejilerinin incelenmiş ve denenmiştir. Çalışmada geleneksel yukarı aşağı (top-down) mimarisi ile aşağı yukarı (down-up) mimarisi karşılaştırılmıştır. Burada aşağı yukarı mimarisinin uygulanması çalışmanın temelini oluşturmuştur. Bir başlangıç noktası ve hedef noktalar belirlendikten sonra robotlar hedefe ulaşmak için harekete geçirilmiştir. 2D yön bulma işleminde robotlar üzerinde bulunan yön mesafe algılayıcılarından alınan bilgiler birbirlerine gönderilmektedir. Ayrıca hız, koordinat bilgileri de paylaşılmaktadır. Algoritmada yüksek seviye katman ön hafızaya sahip planlı katmanıdır. Fakat öncelikli aktif olarak alt gerçekleştirme katmanı kullanılır. Çözümsüzlük durumlarında üst katmana başvurulur. Bu sayede üst katmanda yoğun işlemler azaltılmış ve daha çabuk bir karar verme sağlanmış olur. Algoritma alttan üstte doğru çalışmaktadır. Ayrıca bu çalışmada klasik kullanılan Bug algoritmaları hakkında bilgiler verilmiştir. Tek çalışan mobil robotlarda kullanılan bu algoritmalar sabit çevreler için kullanılmaktadır [31].

Ostafew ve arkadaşlarının çalışmasında öğrenme tabanlı doğrusal olmayan model öngörülü kontrol algoritması ile mobil otonom arazi aracı için yol planlaması yapılmaktadır. Değişen hız ve yük şartlarına göre testler yapılmıştır. Deneyler bilgisayar ortamında 3D olarak test edilmiştir ve başarılı sonuçlar elde edilmiştir [32].

Mobil robotların yörünge planlanması için kullanılan diğer bir yöntem ise yapay arı kolonisi yöntemidir. Bu yöntem ile klasik olasılık yaklaşım metodu karşılaştırılmalı olarak incelenmiştir. Algoritma hem bilgisayar ortamında hem de gerçek bir mobil robot üzerinde test edilmiştir. Test sonuçlarında robotun başarılı bir şekilde engellerden kaçarak sonuca ulaşması sağlanmıştır [33].

Literatürde mobil robotlarla yapılan çalışmalarda ilk önceleri, otonom robotların hareket yönlerinin algılayıcılardan alınan bilgiler tarafından belirlenmekte olduğu gözlemlenmiştir. Zaman içerisinde çok disiplinli çalışmalar ile algılayıcı verileri işlenerek yol planlaması yapılmıştır. Sonraları karar mekanizmasında kullanılan olasılıksal ya da mantıksal çözümler yerini gelişmiş algoritmalara bırakmıştır. Son zamanlarda ise mobil robot çalışmalarında çözümler yapay zeka algoritmaları ile çözümlenmeye başlanmıştır.

BÖLÜM 3

MATERYAL VE METOD

3.1. Mobil Robotlar

Mobil robot, her hangi bir noktaya sabitlenmeden, hareket edebilen mekatronik bir sistemdir. Hareket ortamı hava, kara veya su olabilir. Bu robotlar biri tarafında kontrol edilerek hareket edebileceği gibi, belirtilen görev neticesinde kendi kendilerine de hareket edebilirler. Kendi karar mekanizması olan bu robotlara otonom robotlar denilmektedir. Bir mobil robot; mekanik şase, hareketi sağlayan donanım, çevreyi algılayan algılayıcılar, denetleyici eleman ve elektronik devreden oluşur. Robotlar kullanılacağı yere ve işlevine göre çeşitli boyutlarda tasarlanırlar [34-37].

Önceleri askeri alanda kullanılan mobil robotlar, günümüzde birçok yerde kullanılmaktadır. Bu kapsamda ticari olarak birçok mobil robot geliştirilmiştir. Çim biçen, temizlik yapan, ürün taşıyan mobil robotlar bunlara örnektir. Mobil robotlar konusunda birçok akademik çalışma da yapılmaktadır. Robotların dinamiği, tasarımının yanı sıra yörünge planlanması, alan tanımlama gibi konularda akademik çalışmalar yapılmıştır [38-40].

Yapılan literatür taramasında incelenen çalışmalarda, deney düzeneği için üç farklı metot kullanılmıştır. Bunlar;

- Bilgisayar üzerinde yapılan mobil robot simülatörleri,
- Ticari olarak temin edilen bir mobil robot üzerinde yapılan deneyler
- Özgün tasarımı olan robotlarda yapılan deneylerdir.

Ticari olarak üretilen robotlar fiziksel ve kinematik özellikleri üreticiler tarafından verilmektedir [41]. Bu bilgiler ile simülasyon programları üzerinde robot testleri gerçek ortama yakın olarak yapılabilir. Mobil robot simülatörleri ticari firmalar tarafından sunulduğu gibi ücretsiz açık kaynak kodlu olarak da karşımıza çıkmaktadır.

Bu platformlar mobil robotların yörünge planlama, problem çözme, hareket analizi gibi birçok test ve analizlerini yapabilmektedir. Bu platformlar aynı zamanda ticari ve veri tabanında bulunan gerçek mobil robotların doğrudan programlanmasına ve testlerin yapılmasına olanak sağlamaktadır.

Bu tez kapsamında 5'i ticari olmak üzere toplam 10 adet mobil robot test simülasyon programı incelenmiştir. Bu simülatörlerin birbirlerine göre avantaj-dezavantajları ve akademik çalışmalarda kullanım sayıları gibi özellikleri Tablo 3.1'de gösterilmiştir [42].

Tablo 3.1 Mobil Robot Simülatörlerini Karşılaştırma Tablosu

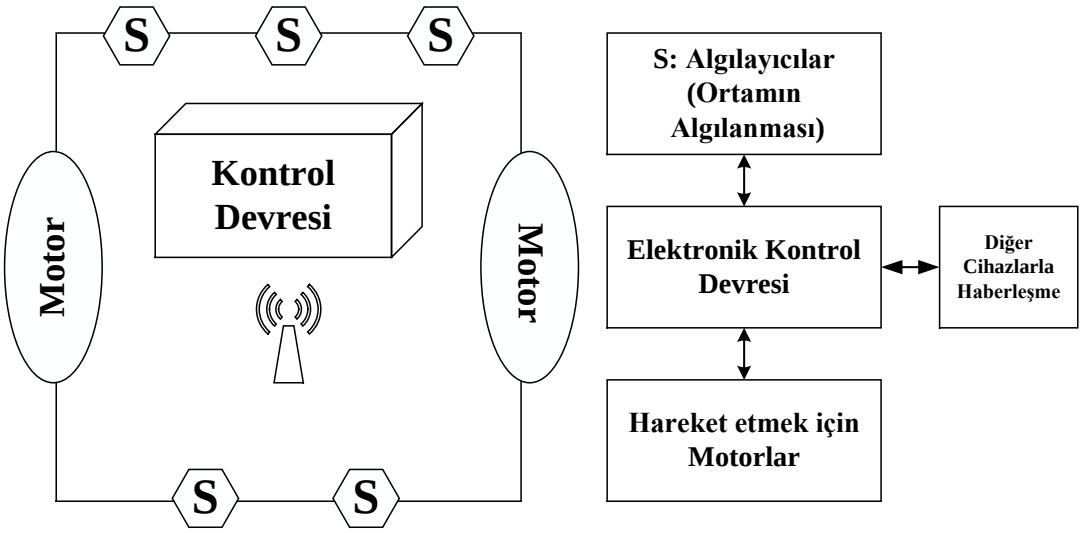
Simulatör Programları	Akademik Veritablalarında Yapılan Taramalar					Ücret[1]	Kullanım Kolaylığı[2]		Fizisel		Dökümantasyon Web Tabanlı İçerik[3]
	IEEE Yayınları	ACM Yayınları	Springer Yayınları	MIT Yayınları	ScienceDirect Yayınları		Simülatör	Kontrolör	Çevre	Robot	
Marilou	3	7	7	0	6	●	●●●●●	●●●●	3D	Var	●●●●
robotSim	1	4	3	0	0	●●	●●●●	●●●	3D	Var	●
Robotics Dev.St.	10	43	40	152	11	Ücretsiz	●●●●●	●	3D	Var	●●
V-REP PRO	7	19	7	6	5	●●●●	●●●●●	●●	3D	Var	●●●●
Webots	65	43	220	7	77	●●	●●●●	●●●●●	3D	Var	●●●●●
Robotics toolbox	97	56	32	7	38	Ücretsiz	●●	●●●	2D	Yok	●●●●●
STDR Simulator	0	0	0	0	4	Ücretsiz	●●	●●●	2D	Yok	●●●●
Simbad 3d Robot	1	17	17	0	4	Ücretsiz	●●	●	3D	Yok	●●●
miniBloq	0	5	1	0	0	Ücretsiz	●	●●●●●	Yok	Var	●●●
Gazebo	25	206	170	7	37	Ücretsiz	●●●●●	●	3D	Var	●●●●

[1] Fiyatlandırmada ● en düşük, ●●●●● en yüksek fiyatı temsil etmektedir. [2] Kullanım Kolaylığında ● en zor, ●●●●● en kolay kullanımı temsil etmektedir.

[3] Dökümantasyonda ● en az, ●●●●● en çok veri ve belge bulunduğunu temsil eder.

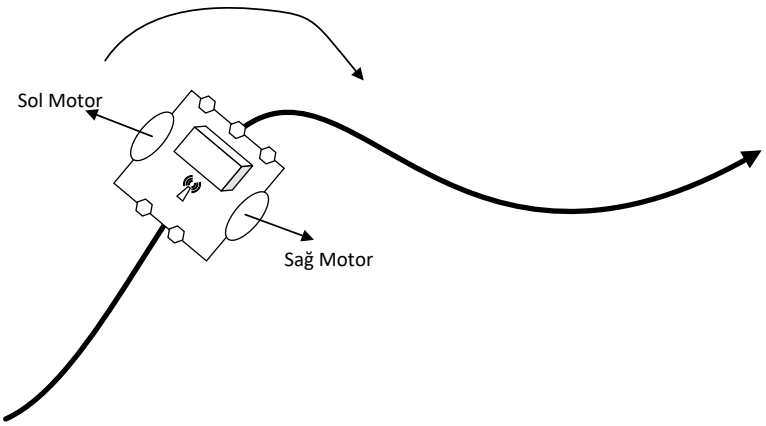
Bu simülatörlerin incelenmesi sonucunda, bir çoğunda esnek yazılım ile geliştirilmeye imkan vermediği görülmüştür. Esnek programlamaya imkan veren platformlar ise ileri seviye programlama dili bilgisine ihtiyaç duymaktadır. Ayrıca tez kapsamında kullanılan robotun bu platformlarda mekanik özellikleri de bulunmamaktadır. Bu sebep ile mobil robot deneylerin test edilmesi için Microsoft Visual Studio 2013 platformu C# programlama dili ile bir simülatör ve deney yazılımı geliştirilmiştir(BÖLÜM 4).

Bir mobil robot yapısal olarak; bir motor, kontrol devresi, algılayıcı devresi ve mekanik şaseden oluşur. Algılayıcıdan elde edilen veriler işlenerek motorlar kontrol edilir [43-44]. Şekil 3.1'de robot bileşenleri yer almaktadır.



Şekil 3.1 Bir Mobil Robotun Yapısı ve Donanımlarının Etkileşimi

Şekil 3.1'de temel bileşenlerinin olduğu bir mobil robot yapısı gösterilmiştir. Tasarım ve donanımların yapısı kullanılacak yere göre çeşitlilik gösterebilir. Temel hareket mekanizması bir elektrik motoru ile sağlanmaktadır. Seçilen uygun güçteki motorlar, robot için tasarlanan mekanik gövdeye bağlanır. Hareket ileri geri olabileceği gibi sağa sola da olmaktadır. Bu sebep ile robotun dönme hareketi için bir yapıya ihtiyaç duyulmaktadır. Bu yapı iki bağımsız motor bulunan robotlarda diferansiyel olarak sağlanır. Bağımsız kontrol edilen iki motor yoksa dönüş için ayrıca bir motora ihtiyaç duyulabilir. Diferansiyel dönüş mekanizması iki tekerlekli veya paletli robotlarda yaygın olarak kullanılır. Bu mekanizmanın kontrolü de kolaydır [45-46].



Şekil 3.2 Diferansiyel Sürüş Örnek Dönme Hareketi

Diferansiyel sürüş mekanizmasında robot x ve y olmak üzere iki düzlemde hareket edebilir. Ayrıca bu koordinatlar arasında dönüş yapabilir. Bu, mobil robotun 3 serbestlik derecesine sahip olduğunu gösterir. Serbestlik dereceleri denklem 3.1'deki matris ile gösterilir.

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{3.1}$$

Bu yapıda, x ve y düzlemleri θ ise dönme açısını göstermektedir. Dönme sağlanması için motorların farklı hızlarda hareket etmesi gerekir. Motorların dönüş hızlarının yanı sıra tekerlek çapları da önemlidir. Denklem 3.2 ve 3.3'de motor hızlarının bağıntısı gösterilmiştir.

$$v_R = r\ \omega_R \tag{3.2}$$

$$v_L = r\ \omega_L \tag{3.3}$$

Burada r tekerin yarıçapı, ω ilgili motorun açısal hızıdır. Bağımsız teker hareketlerinden robotun doğrusal ve açısal hareketi denklem 3.4 ve 3.5'de gösterilmiştir.

$$v(t) = \frac{v_R(t) + v_L(t)}{2} \tag{3.4}$$

$$\omega(t) = \frac{v_R(t) - v_L(t)}{r_c} \tag{3.5}$$

Açısal hız formülünde r_c mobil robotun yarıçapıdır. Yani iki tekerlek arasındaki mesafenin yarısıdır. Buna göre robotun matematiksel modeli denklem 3.6'daki gibidir.

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta(t)) & v(t) \\ \sin(\theta(t)) & v(t) \\ \omega(t) & \end{bmatrix} \tag{3.6}$$

Mobil robotun kinematik denklemi ise aşağıdaki gibidir;

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \dot{q} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} x \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = j(\theta)v \tag{3.7}$$

Burada $\dot{q}$ durum değişkeni vektörünün türevidir. x ve y eksenlerindeki doğrusal ve açısal hız değişimlerinden oluşur[47-49].

Şekil 3.2'de robot, çizgiyi takip ederek ilerlemek istediği zaman sağa doğru bir dönüş yapması gerekmektedir. Robot üzerindeki algılayıcılar çizgiyi algılayarak dönmesi için gerekli sinyalleri kontrol kartına iletirler. Kontrol kartı ise robotun sağ tarafa dönmesi için sağ motoru yavaştır veya durdur. Sol motoru ise normal ilerleme hızında bırakır. Bu, mevcut araçlarda dönme için kullanılan mekanizmanın diferansiyel sürüş sisteminde sağlanmasıdır. Sol motora bağlı tekerlek daha fazla dönerek yol yapacağı için sağa doğru bir yönelme sağlanır. Tam tersi dönüş için sol motorun yavaşlatılması veya durdurulması gerekmektedir.

Sürüş mekanizması robotun hareket etmesi için önemlidir. Fakat robotun nereye gideceği, algılayıcılar ve kontrol kartı tarafından belirlenir. Robot verilen görev çerçevesinde ortamı algılayarak elde edilen verileri kontrol kartına iletir. Kontrol kartında veriler değerlendirilir ve dönüş yönü seçilir. Dönüş yönü belirlendikten sonra uygun motorlar kontrol edilir. Kullanılacak yer ve işlev için farklı algılayıcılar kullanılabilir. Ayrıca robotların bir merkez veya birbirleri ile haberleşebilmesi için bir haberleşme aygıtı da kullanılmaktadır. Bu işlevleri yeri getirebilecek tüm bileşenler mobil robot donanımını oluşturmaktadır. Bu donanımlar internet üzerinde satış yapan sitelerden temin edilebilirler. Temin edilen parçalar geliştirilen mekanik gövde üzerine monte edilerek mobil robot hazır hale gelir. Ancak çalışması için bir yazılım geliştirilmesine ihtiyaç vardır. Mobil robotların çalışmaya hazır olarak satışa sunulan modelleri de mevcuttur. Farklı firmalar farklı işlevleri yerine getirmek için çeşitli tasarımlar yapmışlardır.

Omron firması 1995 yılından itibaren Pioneer mobil robotlarını geliştirmektedir. Geliştirdiği robotların hareketli manipülatörler, iç mekanlarda hareket edebilen robotlar, dış mekanda çalışabilen robotlar gibi bir çok çeşidi vardır. Firma araştırma robotlarının satışını ve geliştirmesini yapmaktadır. Bu robotların birçoğu robot işletim sistemi (ROS – Robot operation system), Matlab, Webots gibi platformlarda çalıştırılabilir[41]. Firmanın ürettiği bazı robotlar Şekil 3.3'de gösterilmektedir.

Arduino, Atmel mikroişlemcisi tabanlı yeni nesil bir fiziksel programlama platformudur. Bu platform ticari olarak satılan bir elektronik devre ve kod geliştirme için ücretsiz açık kaynak kodlu bir ara yüzden oluşmaktadır. Kullanım kolaylığı sebebi ile birçok çalışmada kullanılmaktadır. Kullanıcı ve ticari donanım üreten firmalar tarafından geliştirilen kütüphaneleri ile hızlı ve kolay bir şekilde kodlama yapmak ve çalıştırmak mümkündür[50-53].

(a) (b) (c)

Şekil 3.3 a-Pioneer LX Araştırma Robotu
b- Pioneer Manipülatörler
c- PowerBot Araştırma Robotu

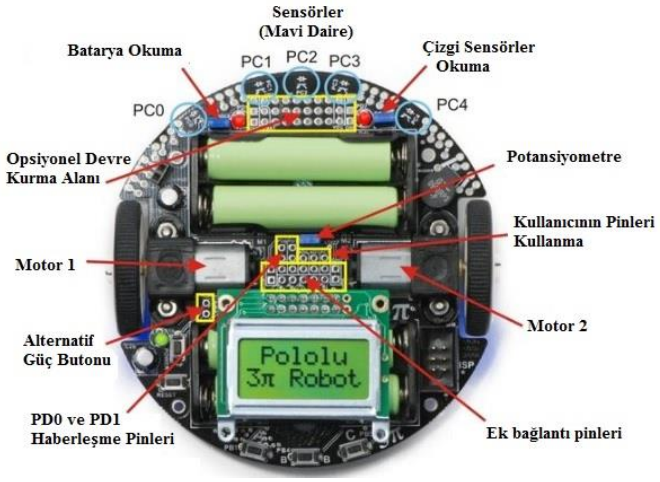
Arduino markası altında birçok kontrol kartı üretilmektedir. 2013 yılından itibaren şasesi baskılı devre kartı olan bir mobil robot üretilmiştir. Mobil robot, üzerinde iki adet mikrodenetleyici bulunan iki katlı olarak üretilmiştir (Şekil 3.4). Bu robotun yazılımı ücretsiz olarak sunulan açık kaynak kodlu Arduino yazılımı ile yapılabilmektedir. Kart üzerinde ek giriş çıkış modülleri mevcuttur. Mevcut donanımları ile çizgi takibi yapabilmektedir[52].

Şekil 3.4 Arduino Robot

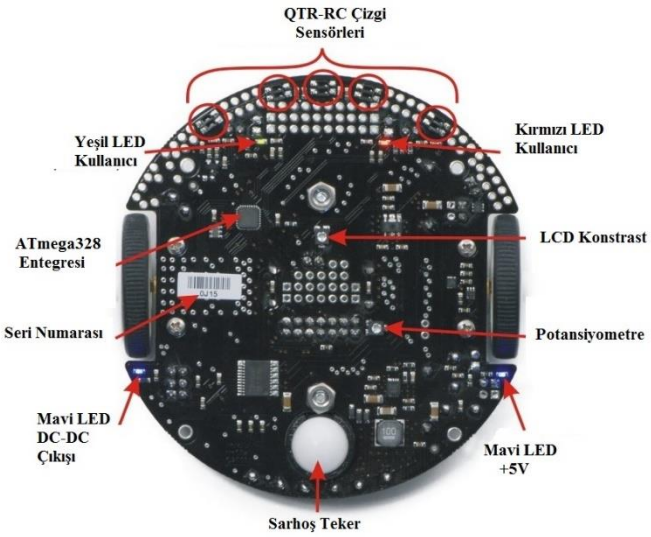
Şekil 3.5 Pololu 3pi Robotu

Yukarıda bahsi geçen ticari robotlara ek olarak farklı robotlar da akademik çalışmaların testleri için sunulmaktadır. Bu tez kapsamında Pololu firmasın ürettiği 3pi mobil robotu kullanılmıştır. Robot küçük yapısı (9.5cm çapında) olması sebebi ile tercih edilmiştir. Robotun şasesi Arduino robotta olduğu gibi baskılı devre kartı olarak tasarlanmıştır. Pololu firması bu robot için motor kontrol ve algılayıcı okuma kütüphanelerini hazır olarak sunmaktadır[54]. Robotun yapısı Şekil 3.5'de gösterilmiştir.

Robot üzerinde 5 adet çizgi algılayıcısı, 3 adet buton, motor sürücü entegresi, DC motor ve 2x8 karakter LCD ekran gibi donanımlar hazır olarak sunulmaktadır. Motor yapısı ve üzerinde bulunan donanımlar Şekil 3.6'da gösterilmiştir.



(a)



(b)

Şekil 3.6 (a)Pololu 3pi Üst Görünüşü
(b)Pololu 3pi Alt Görünüşü

3.2. Yapay Zeka

Yapay zeka kavramı ilk olarak 1956 yılında ortaya çıkmıştır[2, 43]. Ortaya atılmasındaki fikir makinelerin insan gibi düşünebileceğidir. Makinelerin insan gibi hareket etmesi fikri ile geliştirilmeye başlanmıştır. Ünlü İngiliz matematikçi ve bilgisayar bilimcisi Alan Turing bu kapsamda makinelerin zeki olup olmadığını ölçmek için bir test geliştirmiştir. Teste göre bir insan ve makineye başka bir insan tarafından sorular sorulmaktadır. Soru soran insanın aldığı cevaplara göre makine ile insanı ayırt etmesine dayalı bir testtir[1, 55].

İlk yapay zeka çalışmaları, mantıksal ve duygusal olmak üzere iki kavram halinde karşımıza çıkmaktadır. Bazı bilim adamları yapay zeka kavramını şu şekilde tanımlamaktadır[2].

- Sistem insan gibi düşünebilir mi?
- Sistem insan gibi hareket edebilir mi?
- Sistem mantıksal düşünebilir mi?
- Sistem mantıksal hareket edebilir mi?

Yukarıdaki çıkarımlardan, yapay zeka mantıksal veya sezgisel olmak üzere iki kısımda incelenmiştir. Günümüzde yapılan yapay zeka çalışmaları da sezgisel yaklaşımlar(A*, Yapay Sinir Ağları) veya olasılık yaklaşımlar(Bazı Makine Öğrenmesi algoritmaları, BFS, DFS) olarak karşımıza çıkmaktadır.

Yapay zeka birçok bilim dalının ortak çalışması sonucunda şekillenmektedir. Bunlar felsefi, matematik (algoritma hesapları, tahmin mantık), ekonomi (karar verme, oyun, işlem aritmetiği), nöroloji(insan gibi biyolojik etmenler), psikoloji (davranışlar, kavramlar) ve mühendislik (Bilgisayar bilimi, algoritmaların işlenmesi) olarak özetlenebilir[2].

Yapay zeka altında geliştirilen birçok algoritma bulunmaktadır. Bir makinenin bir problemle karşılaştığı zaman problemi analiz etme, yönelme, çözme ve çözümü öğrenme aşamalarının her biri için ayrı ayrı algoritmalar kullanılabilir. Bu tez kapsamında bilgisiz arama algoritmaları ve sezgisel arama algoritmaları ile labirent çözümü ve analizi gerçekleştirilmiştir.

3.3. Labirent Analizinde Kullanılan Algoritmalar

Labirent yapısının doğuşu Yunan mitolojisine dayanmaktadır. İlk desen ve figür olarak yapılan labirentler ilerleyen zamanlarda deney ve test amaçlı kullanılmaya başlanmıştır[56].

Bu tez kapsamında mobil robotun bilinmeyen ortamlarda yön bulma testlerinin yapılması amacı ile labirentler kullanılmıştır. Labirent yapısı bilgisayar üzerinde rastgele olarak üretilerek fiziksel olarak uygulanmıştır. Gerçek mobil robot daha önceden hiç bilmediği bu ortamda seçilen bir başlangıç noktasına bırakılarak labirentin tamamını tanımlamıştır. Labirentin öğrenilmesi için aşağıda belirtilen algoritmalar ile gezgin robotun gerçek zamanlı gezinerek öğrenmesi istenmektedir. Diğer bir öğrenme metodu olarak bölüm 4.4'de anlatılan görüntü işleme ile labirent tanımlanmasıdır.

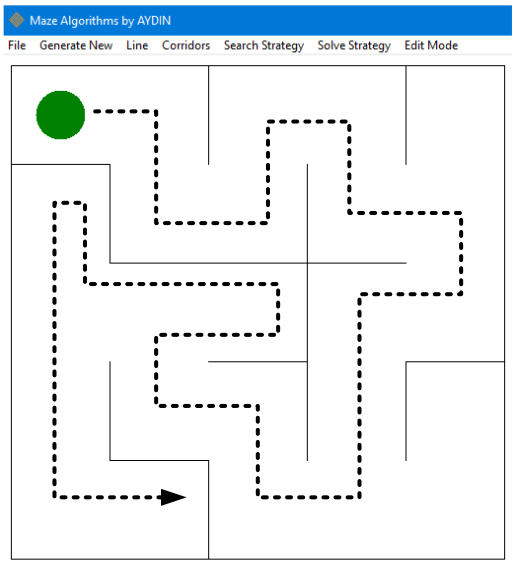
3.3.1. Duvar Takip Algoritmaları

Duvar takip algoritmaları, bir referans yüzeyin belli bir koşulda takip edilmesi ile gerçekleştirilir. Herhangi bir veri kaydetmeye, geçmiş verileri incelemeye ve kıyaslama yapmaya ihtiyacı olmadığı için kodlaması basittir.

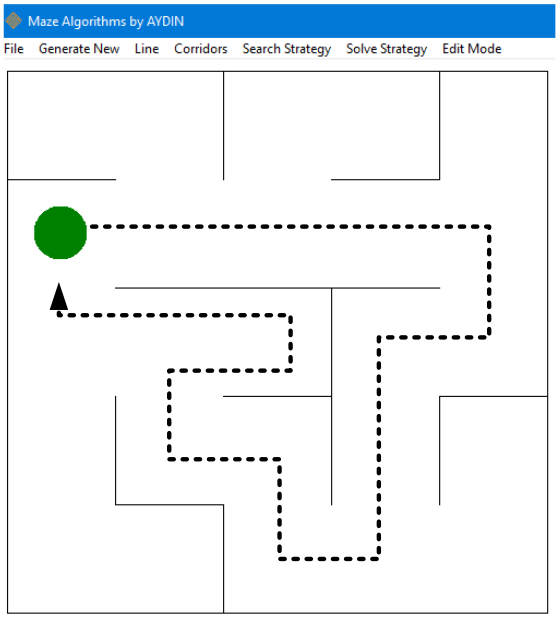
Sağ duvar takip ve sol duvar takip olmak üzere iki çeşidi vardır. Genel olarak, genetik algoritmalar, bulanık mantık gibi diğer algoritmalar ile birlikte kullanılır. Birden fazla çözümü olan karmaşık labirentlerin çözümü ve tanımlanması yapılamaz. Çok çözümlü labirentlerde tekrarlanan döngülere girebilir [57-60].

3.3.1.1. Sağ Duvar Takip Algoritması

Robot, başlangıç noktasında harekete bağlandığı zaman sürekli olarak sağa dönerek hareket eder. Herhangi bir sapak ile karşılaşıldığı zaman ilk olarak sağ yön, yoksa düz olarak ilerlenir. Şekil 3.7'de sağ duvar algoritmasının başlangıç noktasından başlayarak ilerleyeceği hareket alanı gösterilmiştir. Şekilde hareket belli bir noktaya kadar gösterilmiştir. Eğer algoritma koşturulmaya devam edilirse geri dönülerek çalışma devam eder ve dönüşte tekrar sağ yön tercih edilir. Şekil 3.7'deki labirent için labirentin tamamı tanımlanabilir. Ancak bu durum sadece Şekil 3.7'deki ve buna benzer tek çözümlü labirentler için geçerlidir.



Şekil 3.7 Sağ Duvar Takip Algoritması Hareket Gösterimi

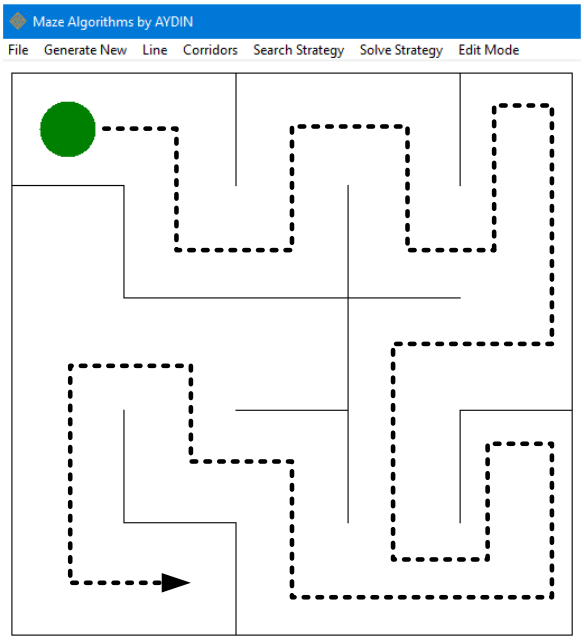


Şekil 3.8 Sağ Duvar Takip Tekrarlanan Döngü

Şekil 3.8'deki gibi başlangıç noktasına konumlanırsa yörünge kendini sürekli olarak tekrarlar. Labirentin diğer kısımlarına giremez. Eğer çözüm bu rota üstünde değilse bulunamaz. Aynı şekilde labirent taraması yapılıyorsa tüm labirenti tanımlanamaz.

3.3.1.2. Sol Duvar Takip Algoritması

Yapısal olarak sağ duvar takip algoritmasının aynısıdır. Tek değişen nokta bir düğümle karşılaştığı zaman bu algoritmada soldaki yönü seçer. Sola doğru yönelme yoksa düz devam edilir. Şekil 3.9'da sol duvar takip algoritması uygulandığı zaman hareket doğrultusu gösterilmiştir. Şekildeki labirent aynı zamanda sağ duvar takip algoritmasının uygulandığı labirenttir. Bu algoritmada da tek çözümlü labirentlerin analizi ve çözümü yapılabilmektedir. Fakat birbirleri ile bağlantılı yollar bulunan çok çözümlü labirentlerin çözümü ve analizi her zaman yapılamayabilir.



Şekil 3.9 Sol Duvar Takip Algoritması Hareket Gösterimi

3.3.1.3. Akıllı Duvar Takip Algoritması

Bu algoritmada sol ya da sağ duvar takibi gibi sabit bir seçim yapılmamaktadır. Robot harekete başladığı andan itibaren ilk olarak bir düğüme geldiği zaman bir seçim yapılır. Yapılan seçim ve düğüm numarası kaydedilir. Sonra tekrar o düğüme gelindiği zaman aynı seçim hatırlanarak farklı bir seçim yapılması sağlanır. Robot bazen sağ duvar takibi yaparken bazen de sol duvar takibi yapar. Bu sayede, labirent yapısı çok çözümlü ve karmaşık dahi olsa tüm labirentin analizi yapılabilir. Labirent taraması yapılırken yeni bulunan düğümler kaydedilerek ortamın haritası çıkartılmıştır. Robot düğümleri ve seçilen yönleri kaydetmiş ve herhangi bir döngüyü girmesi halinde döngüden çıkarak diğer yolların tespit edilmesini sağlamıştır. Akıllı duvar takip algoritmasının çalışması Ek- C Akıllı Duvar Takip Algoritması bölümünde verilmiştir.

3.3.2. Derinlik Öncelikli Arama (DFS - Depth-First Search)

Derinlik öncelikli arama(DFS) algoritması bir grafik arama algoritmasıdır. Yapay zeka algoritmalarındandır ve mantıksal olarak işlem yapar. 19. Yüzyılda Fransız matematikçi Charles Pierre Trémaux tarafından labirent çözümünde kullanılmıştır [61-62].

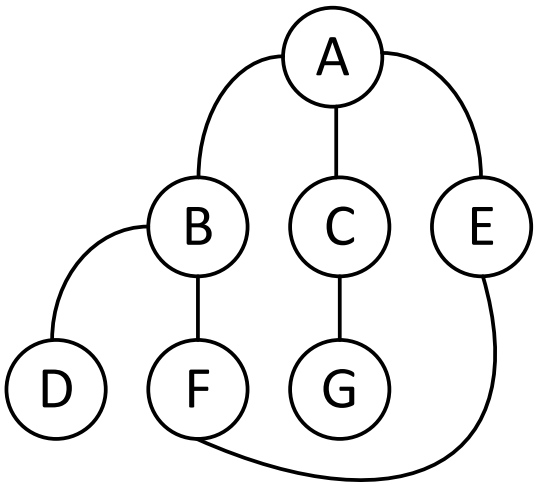
Algoritma, bir grafik yapısındaki verinin derinlik öncelikli olarak taranmasını içerir. Tarama esnasında en üstteki başlangıç noktasından en sondaki düğüm noktasına kadar ilerlenir. Son düğüm noktasına gelindiği zaman bir önceki düğüm noktasına gidilir ve arama oradan son düğüme kadar devam eder. İsmini, arama yaparken hep son düğüme doğru tarama yapmasından alır. Kod olarak iki şekilde kullanılabilir. Bunlardan biri tekrarlamalı programlanma (recursive programming) ile diğeri yinelemeli programlama (iterative programming) ile yapılabilmektedir. Şekil 3.10'da bir grafik tipi veri gösterimi verilmiştir. Bu veri üzerinde DFS algoritması ile arama yapılmak istendiğinde arama sonucu şu şekilde işletilir;

A→B→D→B→F→E→A→C→G

Fiziksel olarak çalışma sırası yukarıdaki gibi olsa da bilgisayar üzerinde bir veriden diğerine geçmek için zaman kaybı olmaz. Bu sebeple program çıktısı;

A→B→D→F→E→C→G

Şeklinde olur.



Şekil 3.10 Grafik Tipi Veri Gösterimi

Programın kodlanması için kod parçaları aşağıdaki gibi verilmiştir.

Burada G: Grafik yapısını, v:düğümleri temsil etmektedir.

Tekrarlamalı fonksiyon için kod parçası;

```
procedure DFS(G,v):
    label v as discovered
    for all edges from v to w in G.adjacentEdges(v) do
        if vertex w is not labeled as discovered then
            recursively call DFS(G,w)
```

Yinelemeli fonksiyon için kod parçası;

```
procedure DFS-iterative(G,v):
    let S be a stack
    S.push(v)
    while S is not empty
        v = S.pop()
        if v is not labeled as discovered:
            label v as discovered
            for all edges from v to w in G.adjacentEdges(v) do
                S.push(w)
```

3.3.3. Genişlik Öncelikli Arama (BFS - Breadth-first search)

Genişlik öncelikli arama algoritması(BFS) yapısal olarak grafik arama algoritmasıdır [63-64]. 1959 yılında labirent çözümü için kullanılmıştır[65-66]. Algoritma bir grafik verisinde belirtilen başlangıç noktasından itibaren kademe kademe inerek tarama yapar. Bu işlem gerçek ortamda uzun sürse de bilgisayar ortamında verilerin işlenmesi için çok zaman almaz. Örnek olarak Şekil 3.10'daki veri ağacı BFS ile tarandığı zaman aşağıdaki işlenir.

A→B→A→C→A→E→A →B→D→B→F→B →A→C→G→C→A→E→F

Fiziksel olarak yukarıdaki gibi düğümler arasında gezinerek çözümü bulur. Ama bilgisayar ortamında düğümler arasında geçiş yapılabildiği için çıktısı aşağıdaki gibi olur.

A→B→C→E→D→F→G

Program yazılması için aşağıdaki kod parçası kullanılabilir.

Graph: Grafik verisi, root: Başlangıç düğümü

```
Breadth-First-Search(Graph, root):
    for each node n in Graph:
        n.distance = INFINITY
        n.parent = NIL
    create empty queue Q
    root.distance = 0
    Q.enqueue(root)
    while Q is not empty:
        current = Q.dequeue()
        for each node n that is adjacent to current:
            if n.distance == INFINITY:
                n.distance = current.distance + 1
                n.parent = current
                Q.enqueue(n)
```

3.3.4. Dijkstra En Kısa Yol Algoritması

En kısa yol için Dijkstra algoritması kullanılmıştır. 1956 yılında Edsger W. Dijkstra tarafından geliştirilen bu algoritmada, grafik yapıda bir ortamda bir düğümden diğer düğüme gitmek için en kısa yolu vermektedir[67-69]. Dijkstra algoritması yazılım tarafından oluşturulan veri yapısına uygulandığı zaman;

Her bulunan düğüm V(Vertex) düğümler arası mesafe E (Kenar) isimlendirilmesine göre ve matematiksel "O" ile gösterimi ile aşağıdaki gibi formüle edilir.

$$O(V^2) \tag{3.8}$$

Döngü içerisinde arama yapılarak en kısa yolun bulunmasında

$$O(V + E) \tag{3.9}$$

Olarak çalıştırılır ve işlem iç içe döngüler ile işletildiği için zamanla işlem sayısı azalır. Dolasıyla,

$$O(\log V) \tag{3.10}$$

Olur.

Tüm bu dönüşümlerle;

$$O((\mathrm{V}+\mathrm{E})\log V) = O(E\log V) \quad (3.11)$$

Şeklinde gösterilir. Problem esnasında, en kısa yolun, başlangıç ve bitiş noktaları dinamik olarak değişmektedir. Bu yukarıdaki gibi bir zaman karmaşıklığı olarak gösterilir.

3.4. Labirent Çözüm Algoritmaları

Ortam tümüyle tanımlandıktan sonra seçilen hedef noktasına en kısa yoldan gidiş hesaplanmıştır. Sonrasında robot tekrar başlangıç noktasından bırakıldığında hızlı bir şekilde çözümün yapılması sağlanmıştır. Bu amaçla robotun labirenti çözmesi için bazı algoritmalar kullanılmıştır. Bu algoritmalar aşağıdaki gibidir.

3.4.1. Açgözlü Öncelikli Arama Algoritması (GBFS)

Sezgisel arama algoritmasıdır. Grafik arama algoritmalarından farklı olarak bir tahmin değerlendirmesi vardır. Arama yapılırken bir öngörü gerekmektedir. Tahmin değerinde doğruluk önemli değildir. Ancak sonuç veya yön ne kadar iyi tahmin edilirse daha verimli ve hızlı bir çalışma sağlanır[70].

En iyi öncelikli arama algoritması (BestFS), Açgözlü Öncelikli Arama Algoritması (GBFS – Greedy Best First Search) algoritmasının temelini oluşturur. BestFS algoritmasında her durum için bir tahmin belirlenir. GBFS'de ise her durum için değil hedefe ulaşmak için bir tahmin belirlenir. Bu sebeple, sonuca ulaşmak için GBFS, BFS'ye göre daha hızlı çalışmaktadır[71-73].

GBFS algoritması aşağıdaki gibi gösterilir.

$$f(n) = h(n) \quad (3.12)$$

Burada;

$f(n)$ =n durumundan hedef duruma kadar hesaplanmış sezgisel fonksiyon.

$h(n)$ =Sezgisel tahmin n durumundan hedef duruma kadar.

Yukarıda görüldüğü gibi sezgisel GBFS fonksiyonun maliyeti hedef durumuna göre yapılmaktadır. BestFS algoritmasında ise hesap her bir düğüm noktasına göre tek tek hesaplatılmaktadır.

3.4.2. A Yıldız Algoritması (A* - A Star)

A* ilk olarak 1968 yılında kullanılmıştır[74]. GBFS'nin olduğu gibi sezgisel arama algoritmasıdır. A yıldız(A*) algoritmasında farklı olarak bilinen bir gerçek değer vardır. Bu değer sezgisel tahmin değerine eklenerek sezgisel fonksiyon hesaplanır[75].

$$f(n) = g(n) + h(n) \quad (3.13)$$

Burada;

$f(n)$ =n durumundan hedef duruma kadar hesaplanmış sezgisel fonksiyon.

$h(n)$ =Sezgisel tahmin n durumundan hedef duruma kadar.

$g(n)$ =n durumundan hedefe kadar yol maliyeti.

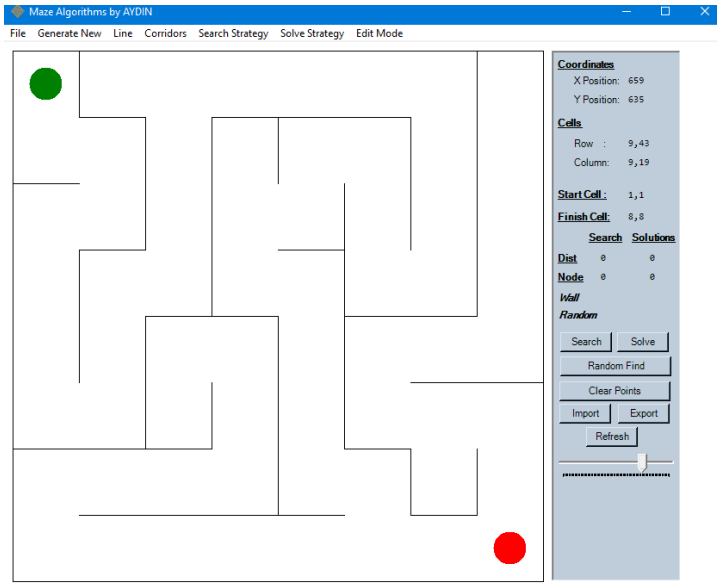
BÖLÜM 4

UYGULAMA VE VERİLERİN ALINMASI

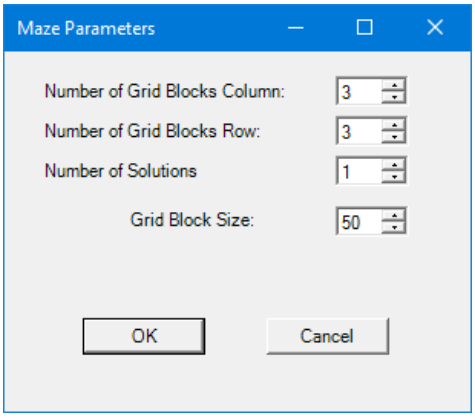
4.1. Geliştirilen Yazılım Arayüzü

Mobil robot uygulamalarının test ve analizi için bir bilgisayar yazılımı geliştirilmiştir. Yazılım C# programlama dili kullanılarak kodlanmıştır. Yazılım ile labirent üretimi, keşfedilmesi ve çözülmesi gibi işlemler yapılabilir. Ayrıca görsel çizim ve arşivlemede kullanılacak fonksiyonlar da geliştirilmiştir. Geliştirilen yazılım arayüzü Şekil 4.1'deki gibidir.

Yazılım üzerinde bulunan Yeni Üret (Generate New) butonuna basıldığı zaman rastgele olarak yeni bir labirent üretilir. Üretilen rastgele labirent boyutları, dosya (file) menüsü altında bulunan labirent parametreleri (Maze Parameters) butonu ile ayarlanır. Bu butona tıklandığı zaman Şekil 4.2'deki pencere açılır.

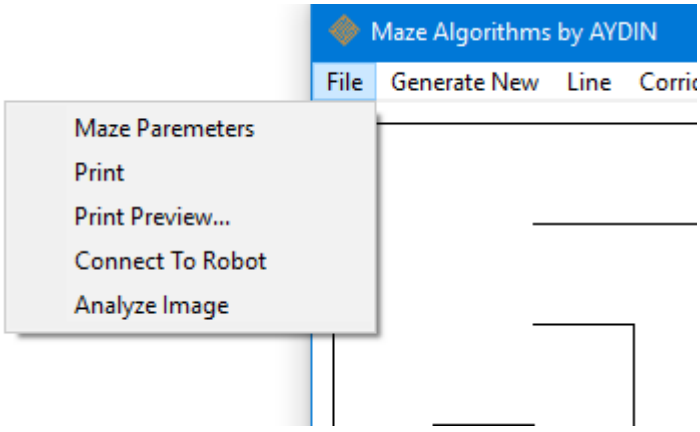


Şekil 4.1 Geliştirilen Labirent Algoritmaları Test Arayüzü

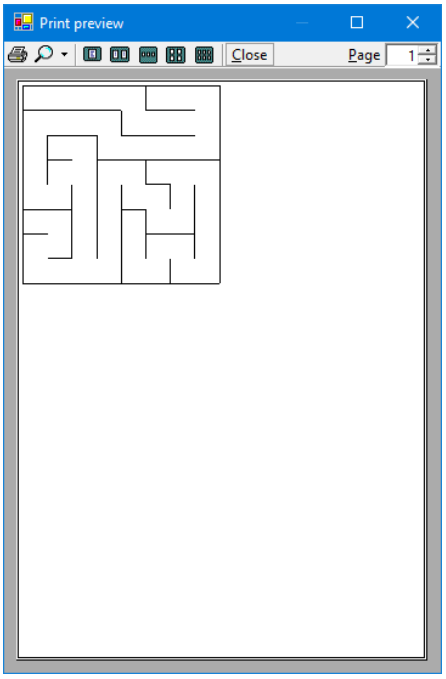


Şekil 4.2 Labirent Parametre Giriş Ekranı

Labirent parametreleri menüsünde labirentin satır ve sütun boyutları ayrı ayrı girilerek ayarlanabilir. Ayrıca bu menüden labirent üzerindeki çözüm sayısı da ayarlanabilir. Normalde üretilen labirentlerin en az bir çözümünün olması sağlanmaktadır. "Number of Solutions" parametresi arttırıldığında labirentin çözüm sayısı da artar. Görüntülenen labirent satır ve sütunlardan oluşmaktadır. Labirentin hücre büyüklüğü "Grid Block Size" kısmından girilerek ayarlanabileceği gibi yazılım arayüz üzerindeki ana ekranın alt tarafında bulunan kaydırma çubuğu ile de ayarlanabilir. Dosya menüsünde ayrıca yazdırma ve yazdırma ön izleme gibi standart fonksiyonlar da çalıştırılabilmektedir. Yazdır (Print) butonuna basıldığı zaman oluşturulan labirent, .pdf formatında kaydedilmektedir. Yazdırma ön izleme (print preview) ise kaydedilecek dosyayı görüntülemektedir.



Şekil 4.3 Dosya Menüsü



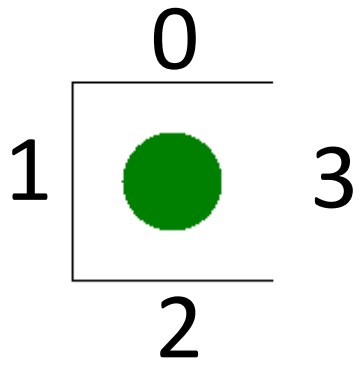
Şekil 4.4 Baskı Ön İzleme Ekranı

Labirent üret butonuna basıldıktan sonra girilen parametreler doğrultusunda rastgele bir labirent üretilir. Labirent üretimi rastgele olmasının yanı sıra en az bir çözümünün de olması gerekmektedir[76]. Bu nedenle labirent üretimi için DFS algoritması kullanılmıştır[77]. Bu algoritma ile mutlaka en az bir çözümü olan ve tüm yolların birbirleri ile bağlantılı olduğu bir labirent üretilmiştir. İstenildiği zaman bu çözüm sayısı arttırılabilir.

Labirent üretim algoritması aşağıdaki gibidir.

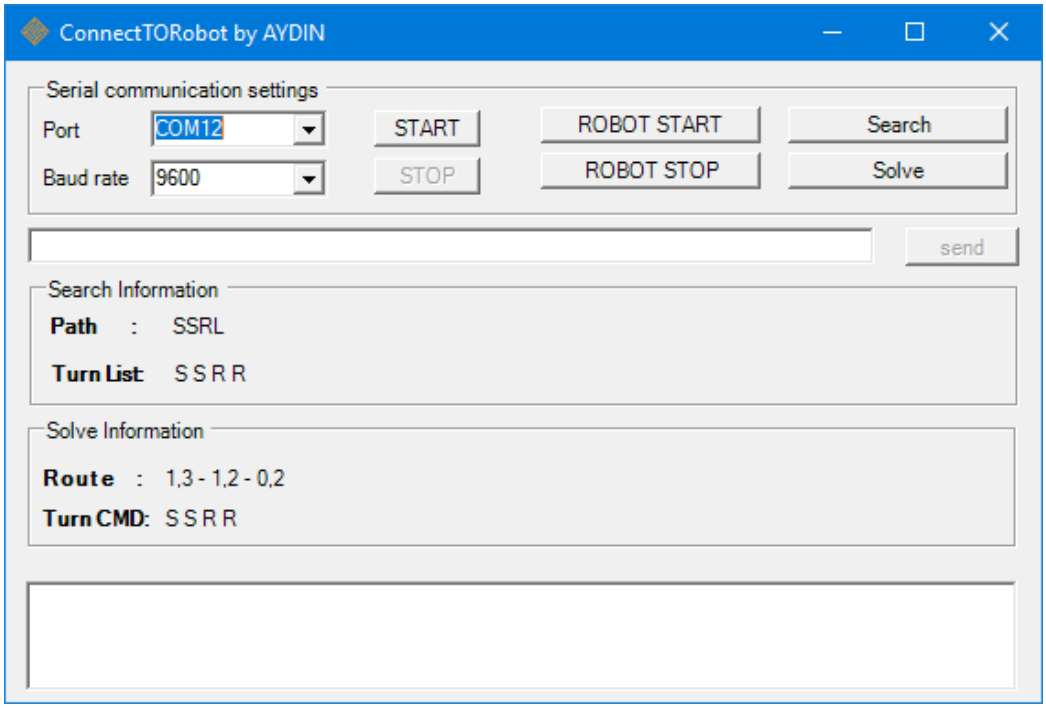
- Labirent Üret
- Parametreleri Oku
- Girilen Satır ve Sütun Kadar Matris Oluştur
- 0,0 Hücresini başlangıç noktası olarak ata
- Mevcut hücreden rastgele bir şekilde satır veya sütun sayısını arttırarak veya azaltarak bir sonraki hücreye geç
- Matris sınırına kadar ilerle
- Bir önceki düğüme gel ve tüm labirent tanımlanana kadar bir önceki adıma atla,

Şeklinde çalışmaktadır.



Şekil 4.5 Labirentin Bir Hücresi

Labirent hücrelerden oluşmaktadır. Her hücrenin dört adet duvarı bulunmaktadır. Yazılım eğer duvar varsa 1, yok ise 0 bilgisini vermektedir. Bir hücre dört elemanlı tek boyutlu bir dizidir. Birinci eleman hücrenin üst duvarını ikinci eleman sol duvarını üçüncü eleman alt duvarını ve son eleman ise sağ duvarını göstermektedir. Şekil 4.5'de gösterilen labirent hücresinde sadece 3. elemanın değeri 0 diğerleri ise 1 olarak tanımlanmıştır.



Şekil 4.6 Robota Bağlanma ve Robot ile Bilgi Alışverişinin Yapıldığı Ekranı

Arayüz ekranının dosya menüsünden, labirent parametreleri ve yazdırma fonksiyonlarının yanı sıra labirentin analizinde kullanılacak robot ile bağlantı ayarları ve görüntü işleme ile labirent analizi fonksiyonları da yapılabilmektedir. Dosya menüsünden robota bağlan "Connect To Robot" seçimi yapılırsa Şekil 4.6'daki ekran açılmaktadır.

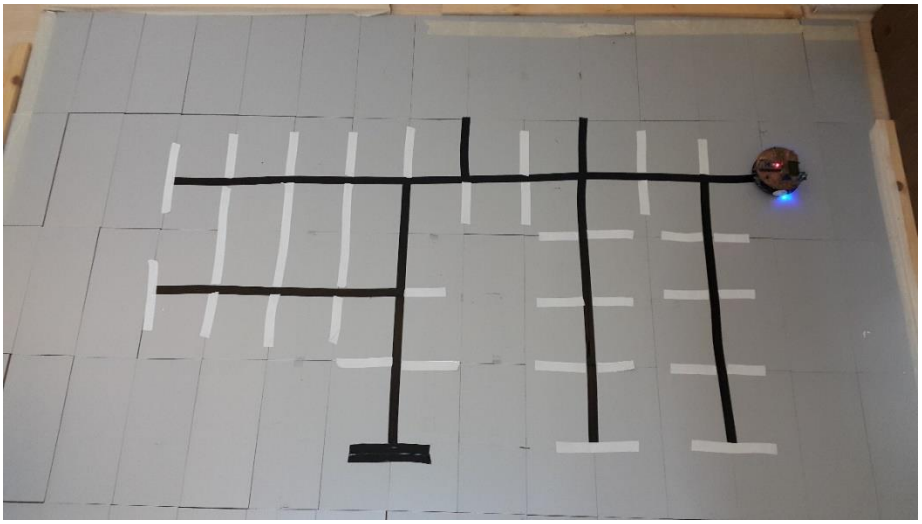
Ekran üzerinden robotun bağlantısının yapılabileceği bağlantı portu adresi ve bağlantı hızı seçimleri yapıldıktan sonra "Start" butonuna basılır ve robot ile bilgisayar arasındaki bağlantı kurulmuş olur. Artık robot bilgisayardan komut almaya hazır hale gelmiştir. Robota, başla komutu "Robot Start" butonu verilmektedir. Bu aşamadan sonra robot ile çözüm veya arama işlevlerinden biri seçilerek yaptırılabilir.

"Arama" seçimi yapılırsa robot bir düğüm noktası tespit edilene kadar ilerlemektedir. Ayrıca her adım bilgisi ortamdan alınarak bu adımlara denk gelen dönüş komutları da kaydedilmektedir. Robot düğüm noktasına geldiği zaman düğüme gelene kadar her adım için kaydettiği dönüş komutlarını bilgisayara gönderilir. Bu bilgiler "Search Information" grubunun altından yol bilgisi olarak gösterilir. Bu komutlara denk gelen koordinat da bilgisayar üzerinde çizdirilir. Düğüm noktasına gelindiği zaman ayrıca düğüm noktasından gidilecek yönler de gösterilmektedir. Bu bilgiler de uygun formatta kaydedilir. Robot, kullandığı algoritmaya göre mevcut düğüm noktasında, gidilecek yönlerden birini seçerek arama işlemine devam etmektedir.

Eğer seçim, çözüm "Solve" olursa bu kez robotun başlangıç ve bitiş noktaları arasında gideceği düğümler sırası ile gösterilir. Ayrıca düğümlerde seçmesi gereken yönler de belirtilir.

4.2. Fiziksel Test Ortamı

Fiziksel test ortamı olarak 8mm kalınlığında 25cmx12cm boyutlarında suntalar kullanılmıştır. Bu suntalar zemine yerleştirilmiştir. Bir sunta iki hücreyi temsil edebilecek şekilde planlanmıştır. Bu kapsamda geliştirilen bilgisayar yazılımında üretilen labirent fiziksel olarak uygulanabilmiştir. Kullanılan mobil robot üzerinde bulunan algılayıcılar çizgi takibi yaparak hareket etmektedir. Bu sebeple test ortamında labirent tipi olarak çizgi labirent seçilmiştir. Bilgisayarda geliştirilen labirent üzerindeki yollar siyah bir elektrik bandı ile suntaların üzerine çizilmiştir. Labirent değiştiği zaman çizgiler sökülerek yeni labirent çizimi yapılmıştır. Çizgi labirent Şekil 4.7'de gösterilmiştir.



Şekil 4.7 Çizgi Labirent Ortamı

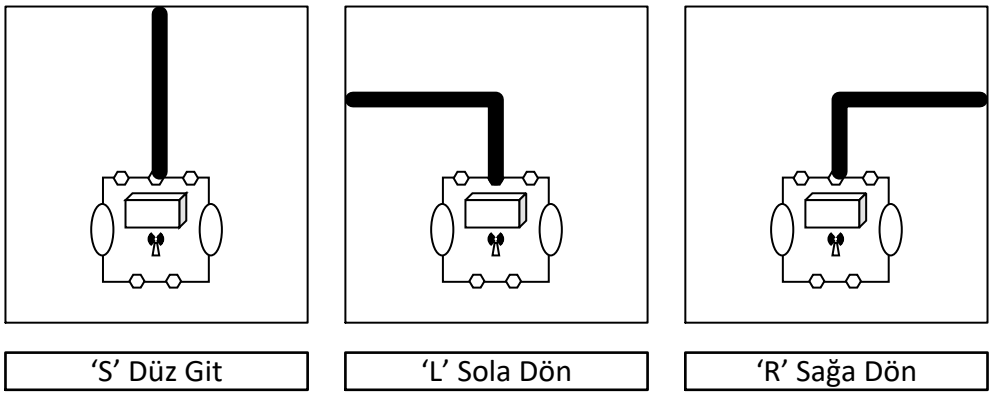
Fiziksel test ortamı alüminyum sigma profiller ve mevcut sunta parçaları kullanılarak duvar labirent şeklinde de kurulabilmektedir. Her iki yapı da modüler olarak tasarlanmıştır. Tasarlanan labirentin esnek olarak taşınması, kurulması için labirent direkleri 3x3x12cm sigma profil kullanılarak yapılandırılmıştır. Temin edilen mobil robotun boy ve en oranları ile doğru orantılı olarak duvar genişlikleri 8mm MDF lavhadan 25x12cm'lik kesitler halinde düzenlenmiştir. Şekil 4.8'de tasarlanan labirentin kurulmuş hali gözükmektedir. Labirent ortamı iki şekilde de planlanmıştır. Koridor labirent fiziksel olarak çalışma ortamına kurulduğu zaman en çok 4x6 boyutlarında bir labirent oluşturmuştur. Bu büyüklük, testlerin yapılması için çok yetersiz olduğundan çizgi labirent kullanılmıştır.

Şekil 4.8 Koridor Labirent Gösterimi

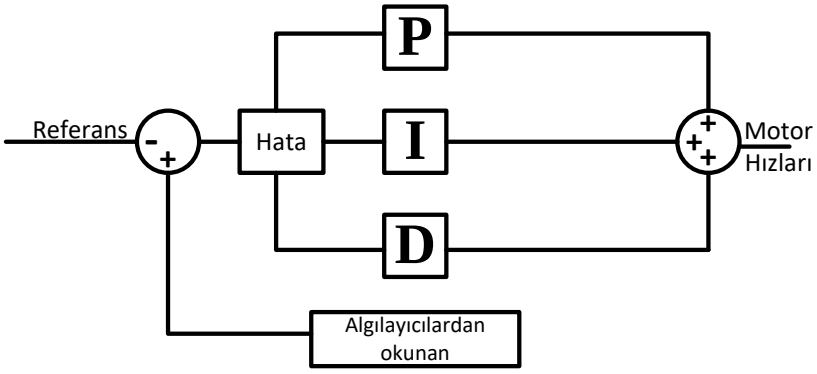
Daha küçük bir mobil robot ile çizgi labirent en çok 13x25'lik matris boyutlarında oluşturulmuştur. Bu sebeple algoritmaların test işlemleri için çizgi labirent ortamı kullanılmıştır.

4.3. Kullanılan Mobil Robot ve Elektronik Donanım

Ticari olarak birçok mobil robot sunulmaktadır. Bunlardan bir kısmı akademik ihtiyaçları karşılayacak şekilde tasarlanmıştır. Bu robotlar bölüm 3.1'de anlatılmıştır. Tez kapsamında pololu 3pi mobil robotu geliştirilerek kullanılmıştır. Bu robot çizgi takibi için algılayıcılara ve donanımlara sahiptir. Şekil 3.5 ve Şekil 3.6'da bu robot ve donanımları gösterilmiştir. Robot üzerinde bulunan beş adet çizgi algılayıcısı bulunmaktadır. Bu algılayıcılar ile çizgi takibi yapılarak çizgi üzerinde bulunan çatallanmalar algılanmıştır. Çizgi algılayıcısında iki grup bilgi alınmaktadır. Birinci grup alınan bilgiler ile robot kendi kendine hareket eder. Herhangi bir yön seçimi veya karar yoktur. Birinci grupta ortamdan alınan bilgiler tek komutlu; sola dön ('L'), sağa dön ('R') ve düz git ('S') olur. Bu komutlar adım adım kaydedilir ve işletilirler. Birinci grupta alınan veriler Şekil 4.9'da gösterilmiştir. Birinci grup verileri geldiği zaman robot her adım için gelen veriyi kaydeder. Gelen komut bilgisine göre motorlar diferansiyel olarak kontrol edilerek yönelme sağlanmıştır. İki DC motor için de aynı hız bilgisi verilmiş olsa dahi motorların veya tekerlerin üretiminden kaynaklı ya da ortamın kusursuz olmayışından kaynaklı olarak robot çizgiyi verimli bir şekilde takip edemez. Çizgiden sapmalar meydana gelebilir. Bu sebeple robotun çizgiyi iyi bir şekilde takip edebilmesi için PID kontrolör kullanılmıştır[78-79]. PID kontrolörde P,I, D aşağıdaki gibi hesaplanmıştır[80-82].



Şekil 4.9 Birinci Grup Algılayıcı Verileri



Şekil 4.10 Tasarlanan PID Kontrolör

$$P = Kp\ e(t) \tag{4.1}$$

$$I = Ki\ \int e(t)\,dt \tag{4.2}$$

$$D = Kd\ {de(t)}/{dt} \tag{4.3}$$

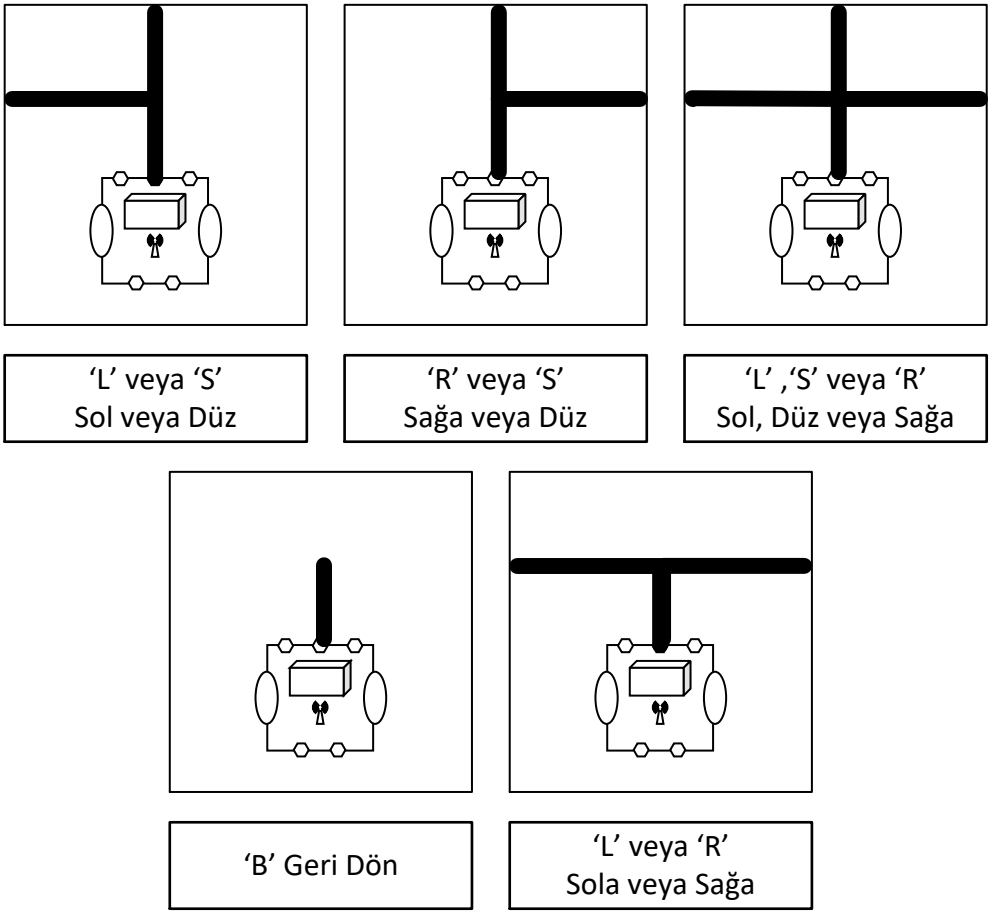
Denklemlerle hesaplanan P, I ve D parametreleri çizgi takibi için Şekil 4.10'daki blok diyagramda belirtilen kontrolör tasarımına uygulanmıştır. Kp, Ki ve Kd katsayıları mobil robot üretici tarafından sunulan katsayılardır[54]. Ancak sistemin kararlılığını arttırmak için katsayılarda robotun daha istikrarlı çizgi takibi yapabilmesi için küçük ayarlamalar yapılmıştır. Aynı zamanda robot hız parametreleri de ortama uygun olarak tekrar düzenlenmiştir. Sonuç olarak robotun çizgi takibi kod parçası aşağıdaki gibidir.

```
     unsigned int position = robot.readLine(sensors, IR_EMITTERS_ON);
         int proportional = ((int)position) - 2000;
         int derivative = proportional - last_proportional;
        integral += proportional;
         last_proportional = proportional;
         int   power_difference   =   proportional/20   +   integral/10000   +
derivative*3/2;
```

Algılayıcılardan robotun çizgi üzerindeki konumuma göre 0-4000 arasında analog olarak bir sayı okunmaktadır. Eğer algılayıcılardan 2000 değeri okunursa bu değer robotun çizginin üstünde olduğunu göstermektedir. Bu sebeple referans olarak 2000 değeri alınmış ve hatanın devamlı sıfır olması üzerine PID hesaplatılmıştır. PID sonucuna göre motorların hızları azaltılmakta eğer yeterli gelmezse motorlar ters döndürülmektedir. Bu sayede düzgün bir şekilde çizgi takibi yaptırılmıştır.

Robot algılayıcılar eğer sola dön ya da sağa dön şeklinde bir dönüş bulduğu zaman, çizgi takip kodu yerini dönme koduna bırakır. Bunun için robotun tekerlekleri 90 derece istenilen yöne doğru döner. Robotun çizgi takibi veya dönme hareketi için motorların hızının kontrolü mikrodenetleyici tarafından darbe genişlik modülasyonu (PWM) ile sağlanmıştır[83-84]. PWM sinyalleri motor sürücü entegresini uygulanarak motor hızları bağımsız olarak kontrol edilebilmektedir.

İkinci grup algılayıcı bilgisi bize robotun bir kavşak (düğüm) noktasına geldiğini ya da bir son noktaya (düğüm) geldiğini belirtir. Bu aşamada bir karar verilmesi gerekmektedir. Robotun seçeceği yön veya ortamının tanımlanmasının tamamladığı kararı seçilen algoritma doğrultusunda bilgisayar tarafında robota verilir. Robotun ikinci grup olarak alınan bilgiler Şekil 4.11'deki gibidir. Bu bilgiler geldiği zaman robotun bir karar vermesi gerekmektedir.

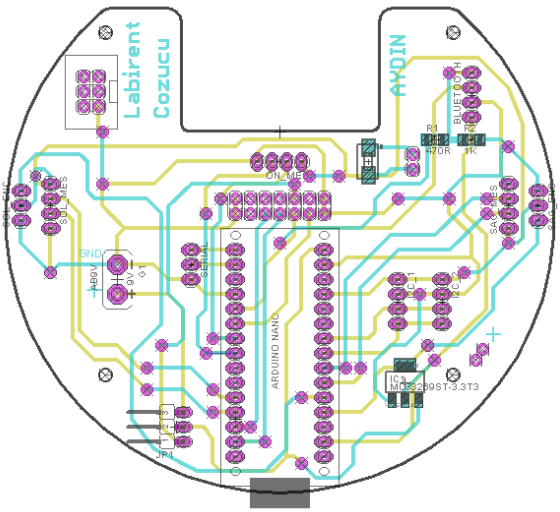


Şekil 4.11 İkinci Grup Algılayıcı Verileri

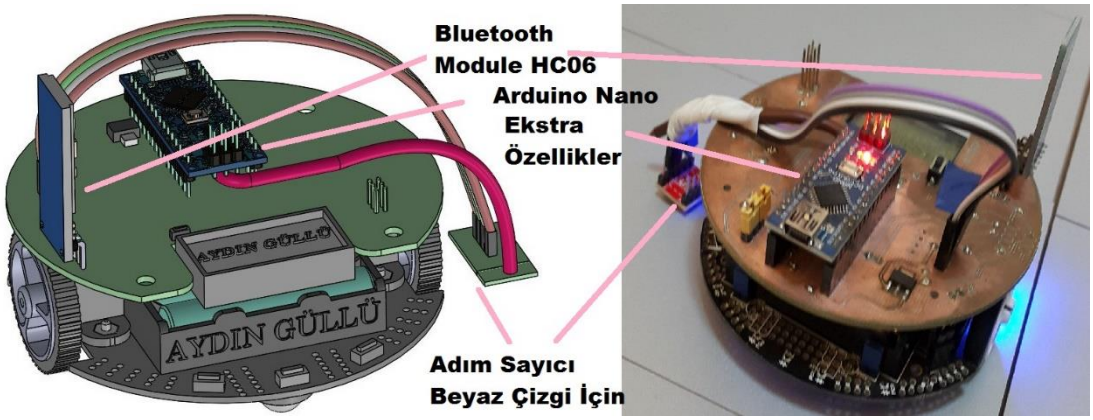
Eğer algılayıcılar herhangi bir çizgi bulamadıysa bu muhtemelen bir son noktadır ve yazılım için bir düğüm noktasıdır. Bu nokta eklenir ve eğer tüm labirent tanımlandıysa robot bu noktada işlemi bitirir. Diğer komutların hepsinde bir seçim söz konusudur. Algoritmaların çalışmasına göre robot bir yönü seçer.

Bu çalışmada robotun hareketi birim hücre bazında ölçülmüştür. Şekil 4.7'deki örnek çizgi labirent üzerinde gösterilen her düğüm noktasını ve her beyaz çizgi bir birimi ifade etmektedir. Bu işaretler başka bir algılayıcı ile algılanarak robotun gerçekte kaç birim yol gittiği bulunmaktadır. Her bir birimin boyutu 12cm'dir. Aynı zamanda birim bir hücreyi temsil etmektedir. Bu sayede bulunan düğümlerin adresleri hesaplanır. Robot birinci grup verileri ve adım bilgisini kaydeder. Bir düğüme ulaşınca bu bilgileri bilgisayar gönderir. İki boyutlu bir matris gibi indisler kaydedilir. İndislerin arttırılıp azaltılması ile robotun hangi hücrede olduğu tespit edilir.

Bu robota ek olarak bir elektronik devre tasarımı yapılmıştır. Bu tasarım sayesinde giriş çıkış sayıları, işlem hacmi arttırılmıştır. Tasarlanan devrenin baskılı devre şeması Şekil 4.12'de verilmektedir. Bu devre robot üzerine pratik bir şekilde monte edilmektedir. Devre üzerinde 4 adet algılayıcı, 2 adet I^2C modülü, 1 adet Arduino Nano, 1 Bluetooth modülü ve ayrı besleme devresi bulunmaktadır. Bu devre mevcut mobil robot üstüne uygun soket ile bağlanarak kullanılmaktadır. Tasarlanan elektronik devrenin devre şeması Ek-A'da verilmiştir. Mobil robot üstüne yerleştirilen ek devre Şekil 4.13'de gösterilmiştir.



Şekil 4.12 Tasarlanan Ek Elektronik Devre



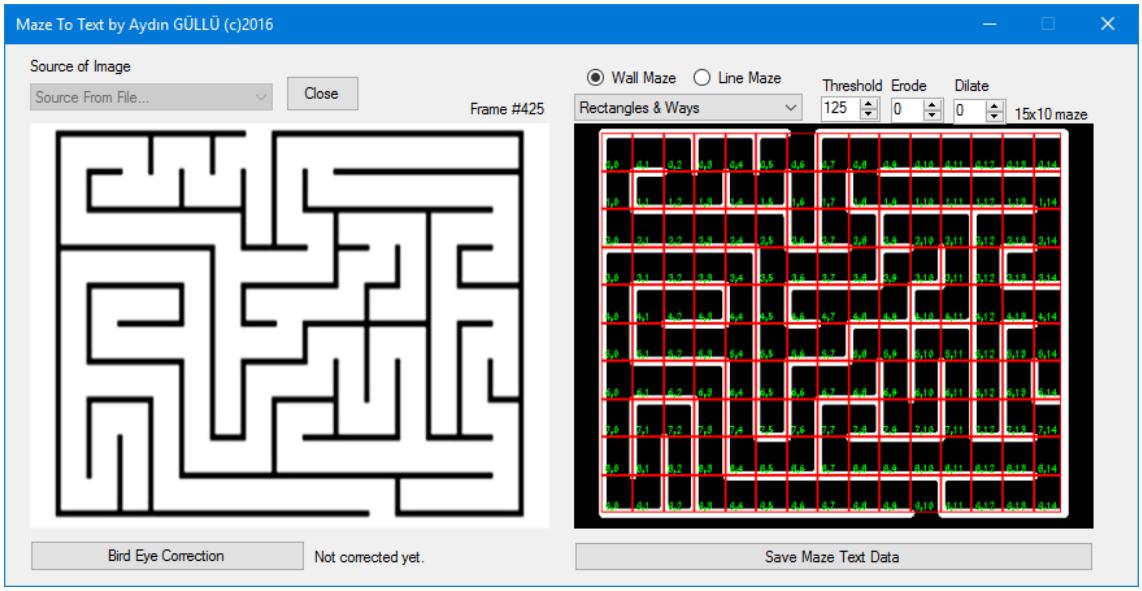
Şekil 4.13 Tasarlanan Ek Elektronik Devre ve Robot

4.4. Ortamın Görüntü İşleme ile Tanımlanması

Bu tez kapsamında mobil robot önceden bilmediği ortamda hareket ederek hedefi bulmayı amaçlamaktadır. Bu amaçla yazılım ve donanım için çeşitli tasarımlar yapılmıştır. Bazı durumlarda mobil robotun fiziksel olarak keşfedeceği ortamlar, önceden tahmin edilebilmektedir. Örnek olarak, planı bilinen bir bina verilebilir. Bu planın bilgisayara yüklenebilmesi çözüm için hız kazandırmaktadır. Bu tez kapsamında yapılan çalışmalarda kullanılan labirentin görüntü işleme yöntemi ile bilgisayara öğretilmesi de mümkün olmuştur. Önceden oluşturulmuş labirentin fotoğrafı çekilerek görüntü işleme ile analizi yapılmıştır. Sonuçlar labirent data tipine dönüştürülerek çözdürülmüştür. Çözüm sonucunda robotun güzergah planı robota gönderilerek robotun istenilen hedefe başlangıçtan itibaren gitmesine olanak sağlamaktadır.

Bu işlem için Visual Studio C# yazılımı kullanılmıştır. Görüntü işleme için arka planda OpenCV kütüphanelerinden yararlanılmıştır. Bunu için EmguCV kullanılmıştır[85-86]. Labirentin görüntüsü bir fotoğraftan alınabileceği gibi bir kamera üzerinden de alınabilmektedir.

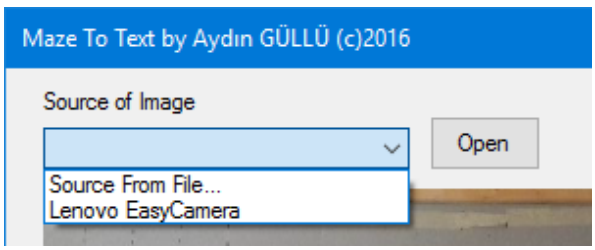
Görüntü işleme için ilk olarak alınan görüntünün düzenlenmesi gerekmektedir. Açılı çekilmiş görüntülerin üzerinde direk olarak işlem yapmak zordur. Bu sebeple eğer görüntü açılı çekilmiş ise bunun dik konuma çevrilmesi gerekmektedir[87]. Bu dönüşüm bir resim üzerinde yapılır. Resim seçimi tamamlandığı zaman dönüşüm için bir dönüşüm matrisinin girilmesi istenir.



Şekil 4.14 Görüntü İşleme ile Labirent Tanımlama

Dönüşüm matrisi resim içinde kuş bakışı dönüşümü yapılacak alanın X,Y koordinatlarının tanımlandığı matrisdir. Matris oluşturması tamamlandığı zaman Bird's Eye(Kuş Bakışı) butonuna tıklanarak dönüşüm yapılmış olur. Program her aşamasında kullanıcılar yönlendirilerek doğru sonuca ulaşılması sağlanmıştır. Tasarlanan görüntü işleme yazılımının ekran görüntüsü Şekil 4.14'deki gibidir.

Program çalıştırıldığı zaman üst tarafta 'open' butonu belirir (Şekil 4.15). Bu buton görüntü yüklemek için kullanılır. Burada iki farklı seçenek sunulmuştur. Bunlardan biri görüntünün bir dosyadan seçimi, diğer(ler)i ise bilgisayar bağlı görüntü kaynağından alınacak labirent görüntüsüdür. Seçim işlemi yapıldığı zaman görüntü hemen altındaki karede belirir.



Şekil 4.15 Görüntü Yükleme Seçimi

Görüntü belirmesinin ardında görüntünün renginin herhangi bir önemi yoktur. Bu sebeple görüntünün daha kolay işlenebilmesi için ilk olarak gri tonlarına dönüştürülmesi gerekmektedir. Görüntüdeki her piksel için, denklem 4.4'deki eşitlikten yararlanılarak renklerin grilik seviyesi bulunur.

$$y = 0.2126R + 0.7152G + 0.0722B \tag{4.4}$$

Gri dönüşümünün ardından görüntüde bazı bozulmalar oluşmaktır. Bu bozulmaları düzelterek daha iyi bir görüntü elde etmek için bir yumuşatma fonksiyonu kullanılır. Gauss yumuşatma fonksiyonu bu işlem için seçilmiştir. Gürültüleri bastırmak için seçilen Gauss Kernel 3x3 fonksiyonu kullanılmıştır. Kullanılan fonksiyon denklem 4.5'de gösterilmiştir[88-89].

$$p = \frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.5}$$

Gri skala dönüşümü ve görüntünün düzeltilme işleminden sonra görüntünün siyah dönüşümünün yapılması için bir eşik (Threshold) fonksiyonu kullanılmıştır. Bu fonksiyon denklem 4.6'da gösterilmiştir.

$$q(x,y) = \begin{cases} p(x,y) \geq T, 1 \\ p(x,y) < T, 0 \end{cases} \tag{4.6}$$

Görüntünün siyah beyaz dönüşümü Şekil 4.14'deki ekranın hemen yanında gösterilir. Görüntü kaynağına göre iyi bir dönüşüm sağlanması için labirentin sınır noktalarının iyi bir şekilde algılanması gerekir. Bunun için gerekli ise ilk olarak kuş bakışı dönüşümü yapılır. Bu dönüşümün ardından görüntü iyileştirilmesi için dönüşümde belirtilen eşik seviyeleri ayarlanır burada üç farklı parametre vardır. Bunlar; Eşik seviyesi (Threshold), Çıkartma (Erode) ve Doldurma (Dilate) parametreleri ihtiyaca göre ayarlanır.

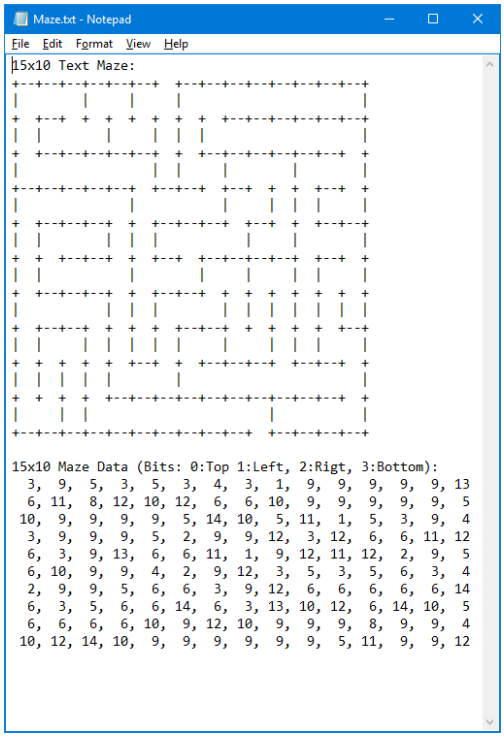
Bu parametreler; Threshold: Resim ikilik dönüşümü yapılırken eşik seviyesini belirtir. Bu eşik seviyesine göre labirentin duvar veya yolları bir matris biçiminde görüntülenmeye çalışılır.

Erode: Görüntü üzerinde gürültü varsa, bu gürültü görüntüde bazı yolların birbirlerine bağlı olmadığı halde bağlıymış gibi gözükmesine neden olabilir. Bunun için erode komutunun değeri ile görüntü üzerinde çıkarma işlemi yapılır.

Dilate: Bu erodenin tam tersine görüntü üzerinde olan boşlukları doldurma için kullanılır.

Tüm bu parametreler uygun görsel için ayarlandığı zaman labirent dönüştürülmeye hazır hale getirilir. Dönüştürme işlemi için 'Save to Text' butonuna basılır ve labirent bir metin olarak kaydedilir. Kaydedilen metin üzerinde labirentin görseli karakterler ile çizdirilir. Bu sayede bir kontrol yapılabilir. Aynı metin dosyası altında labirentin hücrelerinin özellikleri verilmiştir. Metin dosyası ile hazır tanımlanmış labirent üzerinde çözüm komutları uygulanarak robot fiziksel olarak hareket etmeden çözüm işlemine geçebilir.

Şekil 4.16'da dönüştürülmüş bir labirentin çıktısı gösterilmektedir. Bu gösterimde aynı zamanda labirent karakterler ile çizdirilmiştir. Labirent iki boyutlu bir matris şeklinde tanımlanmıştır. Her hücrenin duvar bilgisi de ayrı ayrı tanımlanmıştır. Bir hücrede en çok dört adet duvar bulunmaktadır. Eğer duvar yoksa, bu o hücreden diğer hücreye bir geçiş olduğunu gösterir.



Şekil 4.16 Görüntü İşleme ile Tanımlanan Labirentin Çıktısı

Tanımlanmış labirent duvarları dört bitlik bir sayı gibi kodlanmıştır. Hücrenin üst tarafı 0. bit, sol tarafı 1. bit, sağ tarafı 2. bit ve alt tarafı 3.bittir. Yukarıda tanımlanan labirentin ilk hücresi onluk olarak 3 sayısal değerine karşılık gelmektedir. Bu sayıyı ikilik kodda dönüştürürsek sayı 0011 şeklinde yazılır ve burada hücrenin üst ve sol tarafında duvar olduğunu tanımlar. Labirentin tamamı bu şekilde tanımlanmış hücrelerden oluşmaktadır.

4.5. Ortamın Gerçek Gezgin Robot ile Tanımlanması

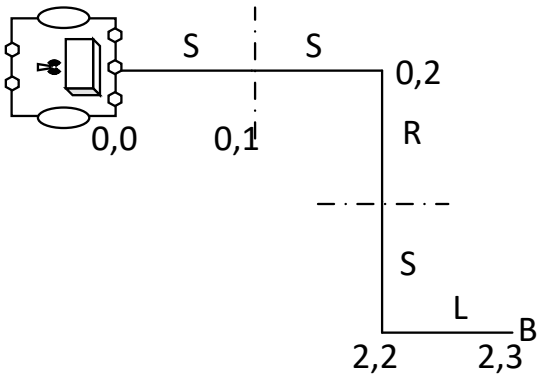
Bölüm 4.4'de ortamın analizi görüntü işleme yöntemi ile sağlanmıştır. Görüntü işleme için ortamın görselinin olması gerekmektedir. Bu bilgi bir fotoğraf veya bir kamera vasıtası ile alınabilir. Ancak bazı ortamlar için bu mümkün değildir. Bir enkaz, maden veya önceden bilinmeyen geniş bir labirent gibi ortamların görüntüsünü almak zor olabilir. Bu sebeple ortamın keşfinin gerçek bir robot ile ortamda gezinerek yapılması gerekmektedir. Robot, ortamda dolaşarak ortamın haritasını çıkartır ve bilgileri bağlı olduğu bilgisayara gönderir. Tüm ortamın tanımlanmasından sonra, ortamdan, seçilen yere gidilebilmek için yol rotası robot için planlanmıştır. Ortamın analizi için duvar takip ve grafik arama algoritmaları kullanılmıştır. Bunlar içerisinde, verimli çalışan grafik arama algoritmalarının verimini arttırmak için hibrid bir algoritma geliştirilmiştir. Algoritma grafik arama ile dijkstra algoritmasının eş zamanlı çalıştırarak tarama esnasına daha kısa mesafe ile tüm ortamın analizini sağlanmıştır.

Keşif için kullanılan algoritmalardan BFS 1950'li yıllarda[90], DFS ise 1900'lü yıllarda [91] ve dijkstra algoritması [92] 1956 yılında bulunmuştur. Bulunan bu algoritmaların mobil robotlar ve mevcut donanımlar için uygulanması gerekmektedir. Bunun için algoritmaları bulan bilim insanlarının ifade ettiği matematiksel veya sistematik yaklaşımların istenilen hedef veya amaç doğrultusunda yeniden kodlanması gerekir. Bu tez çalışmasında bu algoritmaların gerçek bir gezgin robot ile kullanılabilmesi için yazılımlar geliştirilmiştir. Robot, fiziksel ortamından aldığı veriler ile hareket ederek algoritmaları çalıştırır.

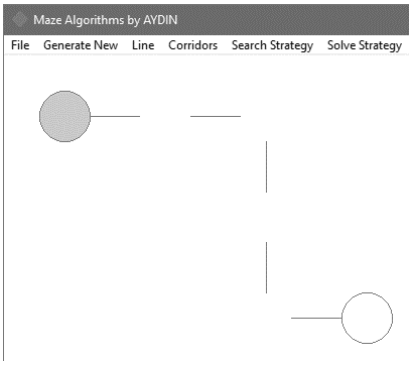
Robotun çalışma algoritması Ek-B'de verilmiştir. Robot ile alan taramaya başlanınca robot ilk olarak çizgiyi takip eder. Çizgi takibi bölüm 4.3'de anlatılan PID kontrol yöntemine göre sağlanmıştır. Ayrıca labirent üzerinden mesafe bilgisi ve labirent ortamındaki işaretler algılanarak tespit edilir.

Bir karar noktasına gelinene kadar robot dönüş bilgisini ve mesafe bilgisini depolanmıştır. Şekil 4.17'de robot 0,0 koordinatından harekete başlar. Sonrasında 2 birim düz ilerlemiştir. Bu esnada robot S,S şeklinde rota bilgisini depolamıştır. Sonra tek kademeli bir dönüşe geldiği için sağa doğru dönmüştür. Bu durumda robotun konumu 0,2 olmuştur. Gelen 'R' sağa dönme komutu ile sağa dönmüş ardından bir birim düz devam etmiştir. Sonrasında sola dönerek ve bir düğüm noktasına ulaşmıştır. Robot düğüm noktasına gelince hareket düzlemini SSRSL olarak göndermiştir. Ayrıca düğüm noktasının koordinatı 2,3 olarak belirlenmiştir. Bu düğüm noktasına ait herhangi başka bir sapak varsa bu bilgide gönderilir. Eğer burada olduğu gibi sapak yoksa 'B' geri komutu gönderilir.

Bilgisayar tarafından koşturulan algoritma durum değerlendirmesi yaparak ya robotu geri dönüş bilgisi gönderir ya da labirentin tanımlaması tamamlandı diyerek programı sonlandırır. Bilgisayar üzerinde ise gönderilen yol bilgisine göre istenilen ölçekte robotun yörüngesi çizdirilir (Şekil 4.18). Ayrıca düğüm noktaları, düğüm noktalarının dönüş bilgileri ve düğüm noktaları arasında mesafe bir grafik yapısı şeklinde depolanır.



Şekil 4.17 Robotun Birinci Grup Bilgiler ile Hareketi

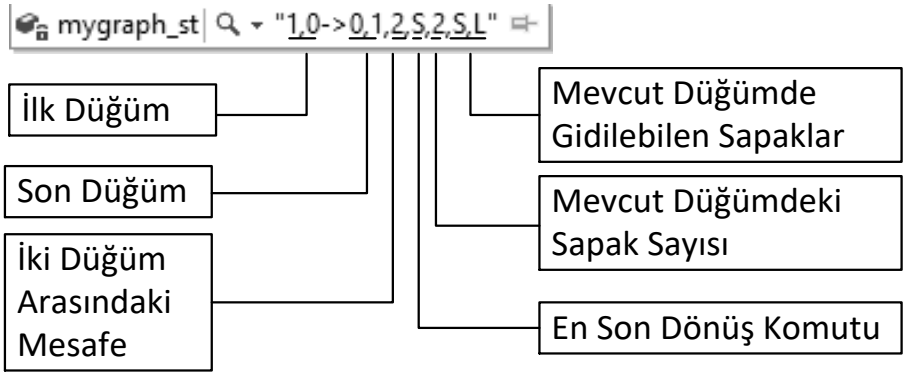


Şekil 4.18 “SSRSL” Komutlarının Bilgisayara Aktarılması

Robotun bir düğümden diğer bir düğüme hareketi ile; bu iki düğüm mesafesi, gelinen düğümden hangi komutla gelindiği ve mevcut düğümün dönüş bilgileri Şekil 4.19’daki gibi depolanır. Ortamda tüm düğümler ve düğümlere bağlı tüm yollar tarandığı zaman labirentin analizi tamamlanmış olur.

4.5.1. Duvar Takip Algoritmaları ile Labirent Analizi

Duvar takip algoritmalarının çalışma yapısı bölüm 3.3.1’de anlatılmıştır. Bu bölümde algoritmaların mobil robota uygulanmasından bahsedilmişrir. Duvar takip algoritmaları sağ, sol ve akıllı duvar takibi olmak üzere üç başlıkta incelenmiştir. Ancak çalışma yapısının benzerliği bakımından sağ ve sol duvar takibi, basit duvar takip algoritmaları başlığı altında anlatılmıştır.



Şekil 4.19 Tasarlanan Grafik Yapısının Bilgileri

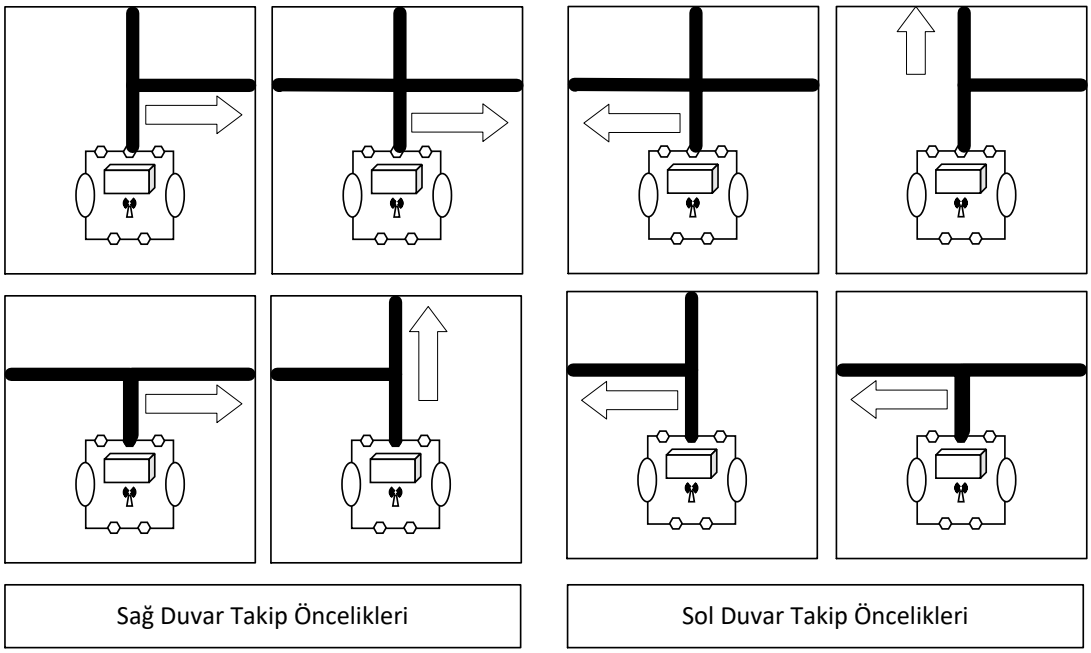
4.5.1.1. Basit Duvar Takip Algoritmaları ile Labirent Analizi

Bilgisayar üzerinden sağ veya sol duvar takibi olarak seçilirse bu algoritmalar çalıştırılır. Robot bir başlangıç noktasına yerleştirildiği zaman bir sonraki düğüm noktasına doğru ilerlenir. Bu aşamaya kadar robot bir önceki bölümde anlatıldığı gibi çalışmaktadır. Düğüm noktasına gelindiği zaman robot karşısına çıkan yönlerden çalıştırılan algoritmaya göre Şekil 4.20' de gösterilen önceliklerle tercih yapar. Robot düğümleri hafızada tutmamaktadır. Her karşılaştığı düğüm için öncelikli yön kavramına göre yönünü seçerek hareket eder. Bu tek çözümlü kompleks yapıda olmayan ortamların taranmasında kullanılabilir. Ancak kompleks grafik yapısında alanların taranmasında kesin çözüm vermeyebilir.

Bu öncelikler bilgisayar yazılımında aşağıdaki kod parçası ile tanımlanmıştır. Kodda ilk olarak dönüş yapılabilecek sapaklar tespit edilmektedir. Robot tarafında yapılan bu tespit bilgisayara gönderilir. Bilgisayarda ise sırası ile bulunan sapaklar için dönüş komutu atanır. En son işletilen komut uygulanır. Eğer sağ duvar takibi için sağ yol varsa sağa dönülür.

```
if (follow == 'L'){
if (found_right == '1') turncmd = 'R';
if (found_straight == '1') turncmd = 'S';
if (found_left == '1') turncmd = 'L';
}
else if (follow == 'R'){
if (found_left == '1') turncmd = 'L';
if (found_straight == '1') turncmd = 'S';
if (found_right == '1') turncmd = 'R';
}
```

Bu algoritmanın en büyük dezavantajı robotun döngüye girmesi ve döngüye girdiğini algılayamamasıdır. Bu sebeple akıllı duvar takip algoritması geliştirilmiştir.



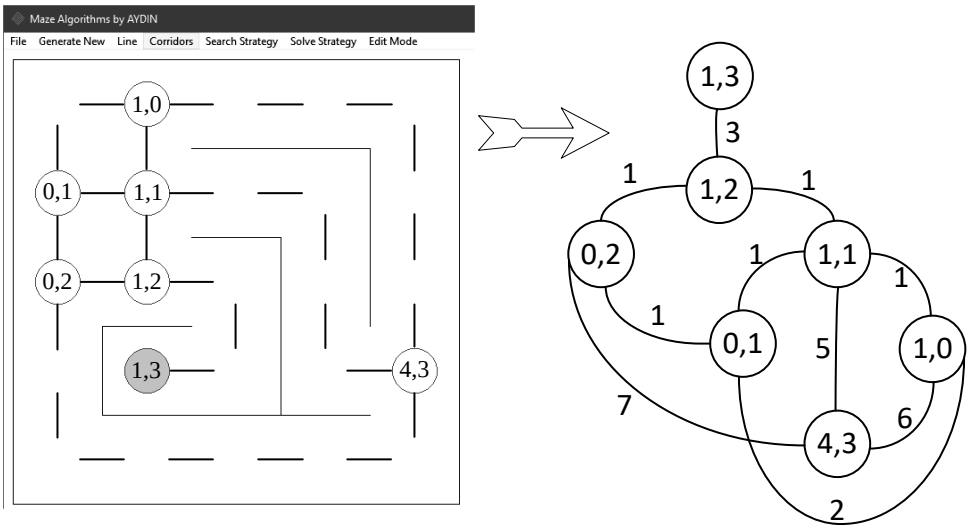
Şekil 4.20 Duvar Takip Algoritması Karar Noktalarında Tercih Öncelikleri

4.5.1.2. Akıllı Duvar Takip Algoritması ile Labirent Analizi

Duvar takibi temelli bu algoritmanın çalışma yapısı sağ veya sol duvar takibine benzemektedir. Bu algoritmanın akıllı olmasının sebebi geçtiği yolları ve tercihlerini hafızasında tutmasıdır. Bu sayede döngüye girmeyerek ve tüm labirentin tanımlanmasını sağlanmıştır. Algoritmanın çalışmasında herhangi bir duvar takip algoritması ile aramaya başlanmıştır. Bulduğu her yeni düğümü kaydederek çalışmayı sürdürmektedir. Ayrıca robot iki düğüm arasındaki mesafeyi de kaydederek grafik şeklinde depolamıştır. Eğer robot önceden kullandığı bir yolu tekrar kullanırsa bu tespit edilmiştir. Klasik duvar takibi ile aynı karar verilirse robot döngüye girebilmektedir. Bu bilindiği için dönüş yönü, duvar takip algoritmasının öncelikli dönüş yönü değil, tespit edilen bir başka yön seçilmiştir. Bu sayede robotun döngüye girmesi engellenmiş ve yeni düğümlerin ve yeni yolların tespit edilmesi sağlanmıştır. Algoritma çalışması Ek-C'de verilmiştir.

4.5.2. Grafik Arama Algoritmaları ile Labirent Analizi

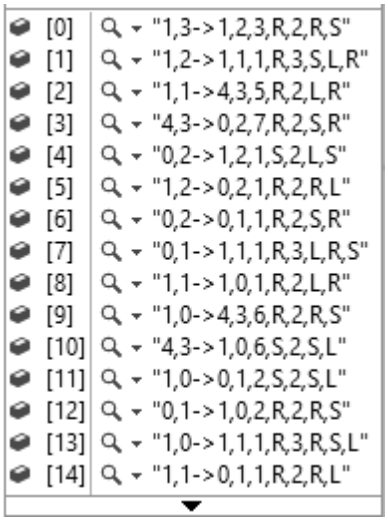
Bu çalışma kapsamında BFS ve DFS olmak üzere iki adet grafik arama algoritması ortamın keşfinde kullanılmıştır. Bu iki algoritma gerçek ortamın analizi için yapılandırılmıştır.



Şekil 4.21 Labirentin Grafik Yapısına Dönüştürülmesi

Bu algoritmalar ağaç veya grafik tipinde verilerin analizinde kullanılmaktadır. Bu sebeple ortamın grafik yapısına dönüştürülmesi gerekmektedir. İki algoritma için de ortak olan bu yapının dönüştürülmesi için robot labirent ortamında dolaşarak bulduğu düğümleri ve yolları kaydedilir. Bu işlemden sonra ortamın grafik yapısı çıkartılmıştır. Eş zamanlı olarak ortamın taranması DFS veya BFS ile gerçekleştiriltmişir.

Şekil 4.21'de labirent çözümlenmesinden sonra grafik yapısına dönüştürülmüştür. Bu yapının oluşabilmesi için bilgisayarda oluşturulan veriler kullanılmıştır. Bilgisayar üzerinde tutulan veriler Şekil 4.22'deki gibidir.



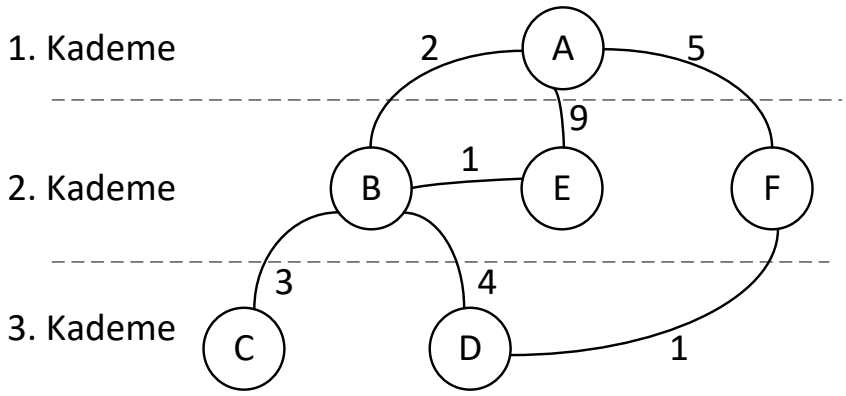
Şekil 4.22 Bilgisayarda Tutulan Grafik Tipindeki Veriler

Bu veri tipi grafik yapısı için; kaynak düğüm, hedef düğüm ve aralarındaki mesafe bilgilerini vermektedir. Ayrıca bu yapıda robot için ortamdan seçilen dönüş yönü bilgisi ve seçilmeyen tüm yönlerin bilgileri de yer almaktadır. Şekil 4.21'deki yapı üzerinde algoritmalar koşturulmuştur.

4.5.2.1. BFS Algoritması ile Labirent Analizi

BFS algoritması bölüm 3.3.3'de anlatılmıştır. Bu algoritma gerçek mobil robot için yeniden kodlanmıştır. Algoritma basit ağaç yapısında kademe kademe taranarak çalışmaktadır. Bu çalışmada, robot ortamda dolaşarak ortamda bulduğu düğümleri ve yolları kaydetmiştir. Bu kayıt işlemi sonucunda depolanan veriler anlık olarak bir grafik yapısına dönüştürülerek BFS algoritması çalıştırılmıştır.

Grafik yapısı Şekil 4.23'deki bir ortamın BFS ile taranması sağlanırsa robot ilk olarak A noktasından hareket başlar. A başlangıç noktasının özelliklerini kaydedilerek taramaya başlanmıştır. A noktasında çıkan 3 farklı yön bulunmaktadır. Bu yönlerden birini seçerek bir sonraki düğüme gidilmiştir. Sonra bulduğu B düğümü yeni bir düğüm olduğu için eklenmiştir. Algoritma genişlik öncelikli olduğu için 1. Kademede sadece A fakat A'dan çıkan 3 farklı yol bulunmaktadır. Algoritma A'dan çıkan bu faklı yolları 2. kademe olarak algılamış ve tüm yolların tanımlanması için hareket edilmiştir. Bu durumda B noktasında tekrar A düğümüne gelinir, A → E → A → F'ye gidilerek ikinci kademe tamamlanmıştır. Artık 3. kademe için arama çalışmalarına geçilmiştir. Bu durumda F düğümünden tekrar sırası ile A, B düğümlerine gelinerek ve üçünce kademe arama çalışmalarına yapılmıştır.



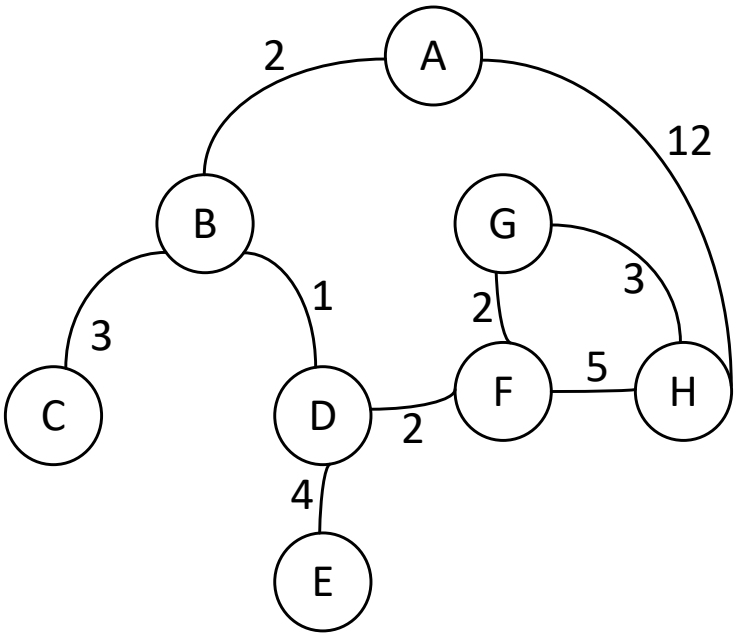
Şekil 4.23 BFS Algoritması Çalışma Örneği

Bu işlem için B → C → B → D → B → E düğümüne gelinince B'ye bağlı üçüncü kademede tamamlanmıştır. F düğümüne gidilerek, bilinmeyen yoldan taramaya devam edilmesi gerekmektedir. Bu işlem için A → F → D yolu izlenerek tüm düğümler ve yollar tanımlanmıştır. Bu güzergah hesabı robotun gerçek ortamda hareket analizine göre yapılmıştır. Robot düğümlerde gidebileceği yönleri, seçtiği yönleri kaydederek bilgisayara göndermektedir. Bu bilgiler bilgisayarda işlenerek BFS ile tarama yapabilmesi için robota rota bilgisi gönderilmektedir. Algoritma işletilmesinde gezilen mesafe birim olarak kaydedilir ve verimlilikleri hesaplanır.

4.5.2.2. DFS Algoritması ile Labirent Analizi

DFS algoritması bölüm 3.3.2'de anlatılmıştır. BFS'de olduğu gibi grafik tipide verilerde arama yapmak için kullanılır. Şekil 4.24'de grafik yapısı için gezgin robotla arama faaliyetleri açıklanmıştır.

Bu algoritma derinlik öncelikli arama yapar. Robot A noktasından harekete başlar ve en son düğüme veya daha önceden tanımlanmış bir düğüm noktasına gelinceye kadar ilerler. Hareket esnasında, yeni düğümleri, düğümler arasındaki mesafeyi ve düğümlerin özelliklerini kaydedilir.



Şekil 4.24 DFS Algoritması için Örnek Gösterim

Robot bir düğüme geldiği zaman düğümden hangi noktalara dönebileceği bilgisini kaydetmektedir. A noktasında 2 adet yol olduğu bilgisi kaydedilir ve bir adet yol seçilerek ilerlenir.

İlerleme A → B → C şeklinde olur. C bir son düğümdür ve 'B' komutu üretilir. Geri dönülmesi gereklidir. Bu aşamada C düğümünden hemen önce olan düğüme gidilir. Bu düğümde gidilebilecek bir yol varsa bu yol tercih edilir ve taramaya devam edilir. Eğer gidilecek yol yoksa bir önceki düğüme daha gelinir. Örnekte B düğümüne bağlı bir yol daha bulunmaktadır. Bu sebeple C'den B'ye ve oradan sırası ile D → E gidilir. Aynı şekilde E son düğümünden geri dönülerek E →D → F →H →A düğümüne gelinir. A düğümüne bağlı 2 yol vardı ve bu aşamada bu iki yol da tespit edilmiş oldu. Tekrar H düğümüne geri gelinir. Sonrasında G → F 'ye gidilir ve rota tamamlanır. Tüm labirent çözümlenmiş olur. Bu güzergah doğrultusunda harcanan yol maliyeti kaydedilir. Labirent tanımlanması grafik şeklinde oluşturulmuş olur.

4.5.3. Optimize Edilmiş Grafik Arama Algoritmaları

Ortamın analizi BFS veya DFS algoritmaları ile sağlanmıştır. Bu iki algoritmanın da gerçek ortam üzerine uygulanmasında bir optimizasyon işlemi gerekmektedir. Bu sayede robotun arama faaliyetlerinde kat edeceği mesafe en aza indirilmiştir.

4.5.3.1. Optimize Edilmiş BFS ile Arama

Şekil 4.23'deki grafik yapısında arama yapılırken. Robot birinci kademede sırası ile; A → B →A → E → A → F düğümlerine giderek birinci kademe tamamlanırtır. İkinci kademe için B düğümüne dönülür F → A → B sonrasında B → C→ B → D → B → E düğümüne gidilir. Bu aşamadan sonra F düğümüne gidilmesi ve tespit edilmeyen bir yolun bulunması gerekmektedir. Yukarıdaki çalışan algoritmalar için her düğümün komşuluk matrisleri çıkartılmaktadır.

Tablo 4.1 BFS için Komşuluk Matrisi Taranan -1

	A	**B**	**C**	**D**	**E**	**F**
A	X	2			9	5
B	2	X	3	4	1	
C		3	X			
D		4		X		
E	9	1			X	
F	5					X

Şu ana kadar yapılan tarama için elde edilen komşuluk matrisi Tablo 4.1'de verilmiştir. Bu matris her yeni düğüm veya yeni bir yol eklendiği zaman güncellenmektedir. En son E düğümünden F düğümü gidilmesi gerekmektedir. Bu durumda robot bilinen düğüm ve yollardan en kısa yolu seçerek F düğümüne gidebilmesi için Dijkstra algoritmasını kullanılmıştır. Bu algoritma Tablo 4.1'deki komşuluk matrisinde tekrarlanmalı çözüm ile en kısa rota çizdirilmiştir. E → A → F şeklinde gidilebilecek olan rotanın maliyeti E → A =9 + A → F =5 olmak üzere toplam 14 birimdir. Bu en kısa rota değildir. Dijkstra ise bize E → B → A → F rotasını sunmaktadır. Bu durumda yol maliyeti 8 birim olmaktadır. Bu sayede labirentin tanımlanmasındaki yol maliyeti azaltılmıştır. Aramaya devam edildiği zaman tüm labirent tanımlandığında son kalan yol F →D mesafesi bulunarak tüm düğüm ve yolların tanımlanması sağlanmıştır.

Gerçek robot ile yapılan çalışma için bir düğümden başka bir düğüme gidilirken robotun seçtiği yön bilgisi de kaydedilmektedir. Grafik yapı çift yönlüdür. Ancak bir düğümden diğerine sağ yolu seçerse dönüş için simetriği seçilmiştir. Nihai oluşturulan komşuluk matrisi Tablo 4.2'deki gibidir.

Tablo 4.2 BFS için Komşuluk Matrisi Nihai Tarama

	A	**B**	**C**	**D**	**E**	**F**
A	X	2			9	5
B	2	X	3	4	1	
C		3	X			
D		4		X		**1**
E	9	1			X	
F	5			**1**		X

4.5.3.2. Optimize Edilmiş DFS ile Arama

BFS algoritmasının gerçek arama için kullanılması bölüm 4.5.3.1'deki gibi Dijkstra algoritması ile optimize edilerek tarama için kat edilen mesafe azaltılmıştır. Tarama yol maliyetinin azaltılması aynı zamanda geçen süreyi de azaltmaktadır. Aynı optimizasyon işlemini DFS için uyguladığımızda da sonuç benzer çıkmaktadır. Şekil 4.24'deki grafik yapısına DFS algoritması uygulandığında çalışma;

A → B → C → B → D → E → D → F → H → A şeklinde olmaktadır. Bu durumda bilinen düğümler arasında komşuluk matrisi çıkartılırsa oluşan matris Tablo 4.3'deki gibidir. Tanımlanan düğümler ve düğümler arasındaki ilişki bu tabloda gözükmektedir.

Robot geldiği A düğümünden en son düğüme yani H düğümüne dönmesi gerekmektedir. Normalde geldiği yoldan geri dönerek H düğümüne ulaşılabilir. Ancak yol maliyet 12 birimdir. Bu geri dönüş rotası Dijkstra algoritması ile oluşturulursa tabloda A'da H düğümüne A → B → D → F → H rotası çizdirilmiştir. Bu rotanın maliyeti ise 10 birimdir. Bu mesafe daha kısa olduğu için daha verimli çözüm sunulmaktadır. Nihai olarak oluşturulan komşuluk matrisi ise Tablo 4.4'de verilmiştir.

Tablo 4.3 DFS için Komşuluk Matrisi Taranan -1

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
A	X	2						12
B	2	X	3	1				
C		3	X					
D		1		X	4	2		
E				4	X			
F				2		X		5
G							X	
H	12					5		X

Tablo 4.4 DFS için Komşuluk Matrisi Nihai Tarama

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
A	X	2						12
B	2	X	3	1				
C		3	X					
D		1		X	4	2		
E				4	X			
F				2		X	**2**	5
G						**2**	X	**3**
H	12					5	**3**	X

Tablo üzerinde tüm düğümler ve düğümlere bağlı yollar gösterilmektedir. Grafik yapısında toplam 8 düğüm ve 9 adet yol bulunmaktadır.

4.6. Labirent Üzerinde En Kısa Rotanın Bulunması

En kısa yol probleminin çözümü için ortamın tümüyle bilinmesi gerekmektedir. Analizi yapılan labirent ortamının tümüyle tespiti için tüm düğümlerin ve düğümlere bağlı olan yolların parametreleri ile birlikte bulunması gerekir. Bu parametreler bölüm 4.4 veya bölüm 4.5'de tanımlanan algoritmalar ile bulunmaktadır. Tanımlama işleminin ardından kullanıcı tarafından seçilecek iki nokta arasındaki en kısa rotanın yapay zeka tabanlı iki farklı algoritma ile bulunması ve rotanın robota yüklenmesi sağlanmıştır.

4.6.1. Duvar Takip Algoritmaları ile En Kısa Yolun bulunması

Bölüm 4.5'de anlatılan duvar takip algoritmaları ile tarama esnasında robota belirtilen hedef doğrultusunda gittiği yanlış yolları çıkartarak sadece doğru rotayı hafızada tutarak en kısa yol çözümünü sağlanmaktadır. Çalışma aşamasında tarama yapılırken çözümün de bulunmasına olanak sağlamaktadır.

Bu kısa yol algoritmasının avantajları;

- Robot tarama esnasında çözüm rotasını çıkartabilmektedir.
- Robot ve bilgisayar üzerinde yazılımın kodlaması ve kodun çalışması daha basittir.

Bu algoritmanın dezavantajları;

- Bu yöntem, sadece çözümün önceden bilindiği konumlarda yapılabilir.
- Çözüm konumu tarama sonrasında değiştirilirse yeni rota hesaplanamaz.
- Eğer labirentin tümüyle taranamaması ya da hedef noktanın bulunamaması söz konusu olursa çözüm rotası çizdirilemez.
- En kısa yolun bulunmasını her zaman sağlayamaz.

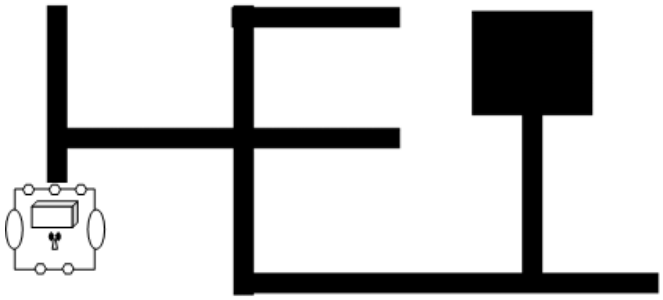
Çalışması için robot başlangıç noktasında itibaren hedef olarak belirtilen çözüm noktasına doğru taramaya başlatılır. Tarama esnasında eğer bir "B" komutu (Geri Dönüş) gelirse bu konum yanlış bir rota olduğunu gösterir. Bunun için robot en son düğüm noktasından seçimini yaptığı yönün hatalı olduğunu kabul ederek başka bir yön seçmesi sağlanır. Bu işlem için robotun her dönüş hareketine açısal bir karşılık değeri verilmiştir. Bu hesapla Düz 0°, Sağ 90° Geri 180° ve Sol 270° olarak tanımlanmıştır.

Şekil 4.25'de gösterildiği gibi robot düz giderek ilerlerse, sonrasında bir geri dönüş ile karşılaşılır. Sonrasında geri dönüş işlemi yaparak diğer seçimi yapılır. Son seçimi sol olur. Bu durumda gittiği yönde sonuç bulunmamaktadır. Bu sebeple robotun geri dönüşten bir önceki ve bir sonraki işlettiği komutlar;

$$Yol = S + B + L \tag{4.7}$$

Denklem 4.8 deki gibi sayısal olarak hesaplanırsa;

$$Yol = 0° + 180° + 270° = 450° \tag{4.8}$$



Şekil 4.25 Duvar Takibi ile En Kısa Rota için Örnek Labirent

Yol dönüş açısı 450° olarak bulunmuştur. Bu işlem 360°'ye bölünerek kalan bulunursa.

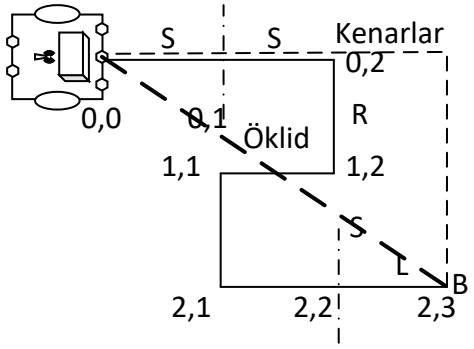
$$Yol = 450° \% 360° = 90° \quad (4.9)$$

90° olarak hesaplanır burada yanlış gidilen rota için bu üç komutun yerine R = 90° olarak tanımlanır. Rota üzerinde SBL = R şeklinde dönüştürülür. Bu sayede yanlış gidilen yönler sadeleştirilerek sonuca giden en kısa rota hesaplanır.

4.6.2. GBFS Algoritması ile En Kısa Rotanın Bulunması

Açgözlü en iyi öncelikli arama algoritması (GBFS) bölüm 3.3.3'de açıklanmıştır. Bu algoritma, daha önceden tanımlanmış labirent üzerinde seçilen iki nokta arasındaki en kısa yol rotasını hesaplayabilmektedir. Bu algoritmanın çalışmasında bir tahmin bilgisinin girilmesi gerekmektedir. Tahmin bilgisi için ortamın tümüyle tanımlanmış olması gerekmektedir. Tahmin kullanıcı tarafından belirtilen başlangıç ile bitiş noktası arasında olan tahmini bir mesafedir. Bu mesafe gerçek ulaşılabilecek mesafeden daha uzak olmayacak şekilde seçilmelidir. Bu çalışmada tahmin mesafesi iki farklı yöntem ile seçilmiştir. Çalışılan ortam, labirent ve yönelme X,Y düzlemlerinde olduğu için birinci tahmin mesafesi kenar uzaklığı X mesafesi + Y Mesafesi olarak seçilmiştir. İkinci tahmin ise Öklid uzaklığıdır. Bunun için iki dik kenar mesafelerinin Pisagor teoremi ile Öklid uzaklığı hesaplanarak eklenmiştir [74, 93].

Şekil 4.26'da gösterilen robot hareketinin ardından 2,3 koordinatına ulaşılması istenmektedir. Robot ortamın tanımlanmasını iki boyutlu bir matris şeklinde yapmaktadır. Her nokta bir X ve Y koordinatları ile gösterilmektedir.



Şekil 4.26 Tahmin Mesafesinin Bulunması

Bu altyapı da iki farklı tahmin mesafesi çıkarmak mümkündür. Robot 0,0'dan 2,3 koordinatına gönderilmek istendiğinde tahmin mesafesi kenar uzunlukları olarak

Tahmin Kenarlar 2+3=5 (4.10)

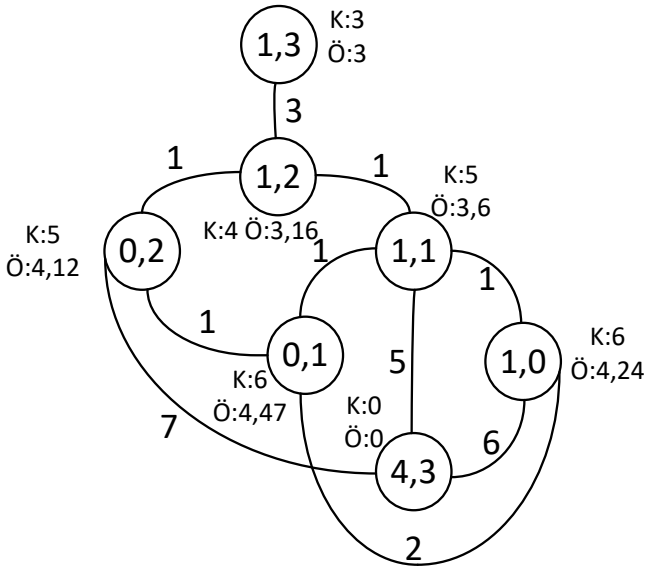
Olarak hesaplanır eğer Öklid kullanılacaksa

Tahmin Öklid $\sqrt{2^2 + 3^2} = 3{,}605$ (4.11)

Olarak hesaplanır. Bu iki yöntemde ayrı ayrı kullanılarak tahmin mesafesi ayrı ayrı hesaplatılmıştır.

Denklem 3.12'de gösterilen formülde h(f) fonksiyonunun yukarıda belirtilen tahminler ile her düğüm için ayrı ayrı hesaplatılmıştır. Bu hesaplama sonucunda her düğümün tahmin maliyeti bulunmuştur. GBFS algoritması ile en kısa yolun bulunma süreci çalıştırıldığında;

Şekil 4.27'de gösterilen labirent yapısında tüm noktaların hedef gösterilen 4,3 koordinatına göre tahmini mesafeleri iki farklı yöntem içinde ayrı ayrı hesaplatılmıştır. Yapılan hesaplamalar sonucunda robotun hedefe ulaşmak için uğrayacağı düğümler sırası ile 1,3 → 1,2 robot buradan kenar tahmini kullanarak hareket ederse 0,2 veya 1,1 rotalarından herhangi birini seçmek durumundadır.



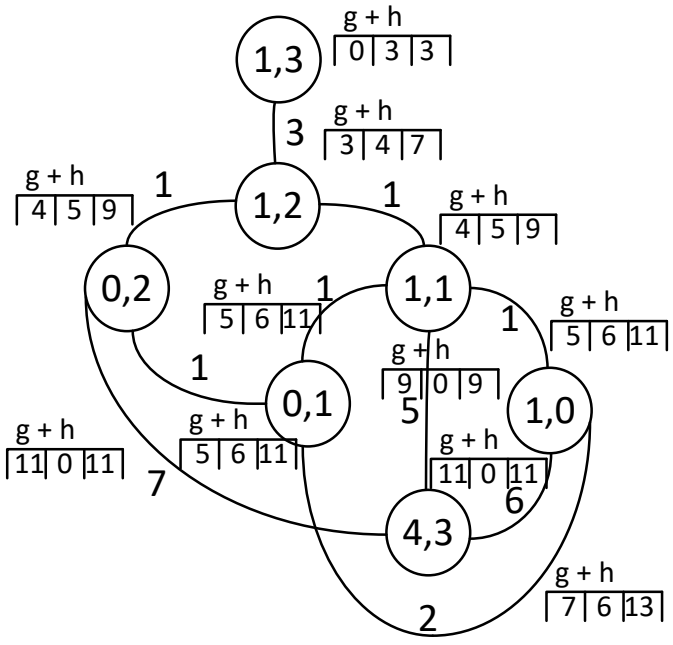
Şekil 4.27 Düğümlerin 4,3 Koordinatına Göre Tahmini Mesafeleri

Bu seçim rastgele yapılabilir. Fakat Öklid tahminine göre ise rota 1,3 → 1,2 → 1,1 → 4,3 olarak belirlenir. Düğümlerde dönüş komutları ise S → R → R şeklinde olur.

GBFS ile en kısa yolun bulunmasına göre yapılan çalışmada sonuca ulaşabilmek için mobil robota belirlenen düğümlere uğrayarak gidebilmesi için gereken yönler bulunmuştur. Fakat bu rota her zaman en kısa yolu veremeyebilir.

4.6.3. A* Algoritması ile En Kısa Rotanın Bulunması

Bu algoritmanın çalışması GBFS'nin çalışmasına benzemektedir. Aynı şekilde düğümlerden hedefe olan bir tahmin mesafesi bulunmaktadır. Fakat robot sadece tahminlerle hareket etmemektedir. Ayrıca robot sayısal olarak kat edilen mesafeleri de hesaplayarak formüle eklemektedir. Algoritma denklem 3.13'de gösterilmektedir. Formülde yer alan tahmin için bölüm 4.6.2'de anlatılan tahmin mesafesinin hesaplama metotları kullanılarak hesaplama yaptırılabilir. Şekil 4.28'deki grafik yapısındaki bir labirent üzerinde en kısa yolun bulunması için A* çalıştırılırsa maliyetler hesaplanır.



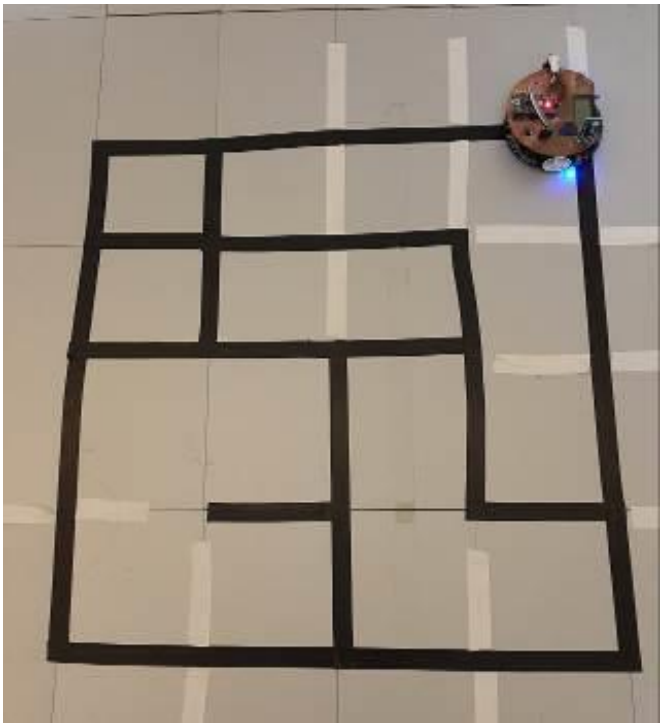
Şekil 4.28 A* Algoritmasına Göre Maliyet Hesabı

Yapılan maliyet hesabı 4,3 düğümüne göre her düğüm ve her yol için ayrı ayrı hesaplatılır. Bu hesaplama sonucunda, sonuca ulaşmak için üç farklı alternatif bulunmaktadır. Bu alternatiflerden ikisinin maliyeti 11 birimdir. Diğeri ise 9 birimdir. Bu en kısa yoldur. Sonuçtan başlayarak en kısa yoldan başlangıç noktasına hareket ile gidilirse robot rotası 1,3 → 1,2 →1,1 → 4,3 olarak hesaplanır. Bu rota için robota S → R → R dönüş komutları yüklenir.

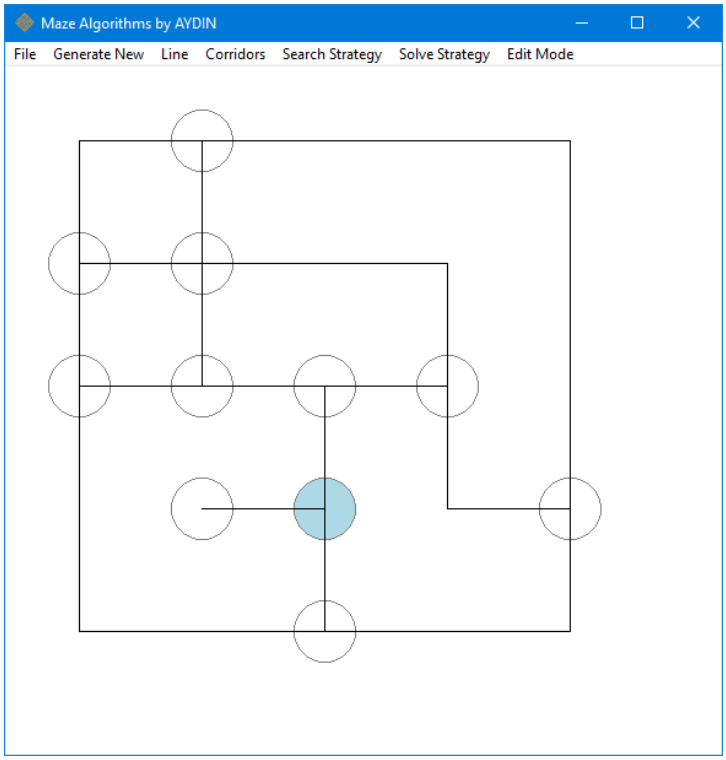
4.7. Labirentin Tanımlanmasında Kullanılan Algoritmaların İncelenmesi

Labirent tanımlanması tez kapsamında iki şekilde yapılmaktadır. Bunlardan biri görüntü işleme ile tanımlamadır. Daha önceden bilinmeyen ve görüntüsü (video veya fotoğraf) olmayan ortamların analizinde robotun gezinerek tanımlama yapması için çeşitli algoritmalar geliştirilmiştir. Bu algoritmalar bölüm 4.5'de uygulamalı olarak anlatılmıştır. Dört farklı, çok çözümlü labirent için yapılan arama çalışmaları sonucu robotun labirenti tanımlamada kat ettiği mesafeler bulunmuştur. Bu mesafeler Tablo 4.5'de gösterilmiştir. Test edilen labirent ortamları DFS algoritması ile rastgele olarak üretilmiştir. Üretilen labirentler fiziksel olarak uygulanmış ve mobil robot ile arama yapılmıştır.

Mobil robot arama esnasında gittiği yolu, kaç adım mesafe kat ettiğini, bulduğu düğümleri ve düğümler arasındaki yolları kaydetmiştir. Labirent üzerindeki tüm noktalar tanımlanana kadar algoritmalar çalıştırılmıştır. Labirentlerin çözümlenmesi sonucunda bilgisayar arayüzü üzerinde tamımlanan çıktıların haritaları Şekil 4.30, Şekil 4.32, Şekil 4.34 ve Şekil 4.36'da gösterilmiştir. Fiziksel ortam fotoğrafları ise Şekil 4.29, Şekil 4.31, Şekil 4.33 ve Şekil 4.35'de gösterilmiştir.

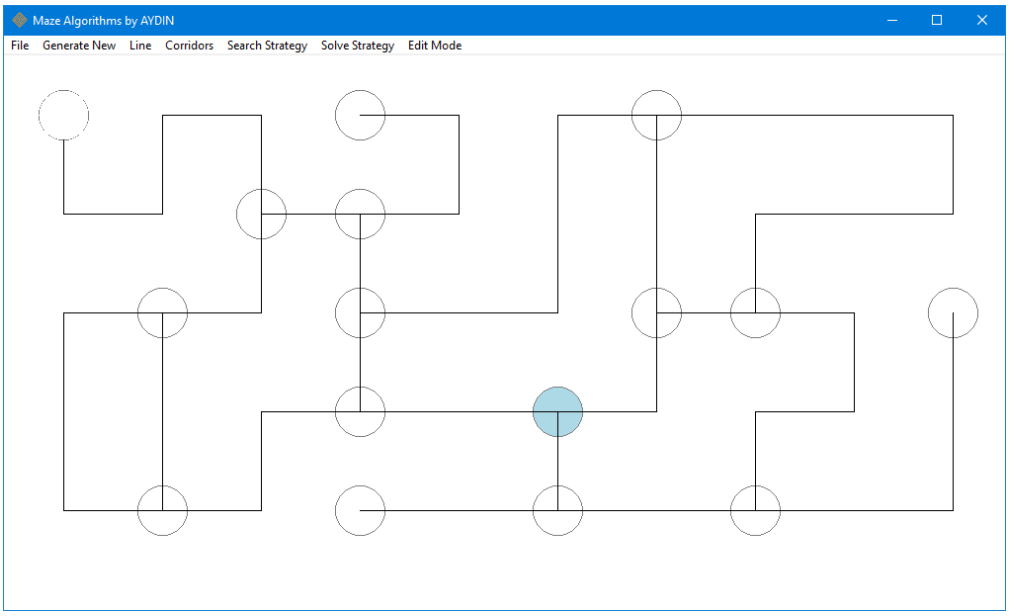


Şekil 4.29 Test Labirenti 1 (L1)

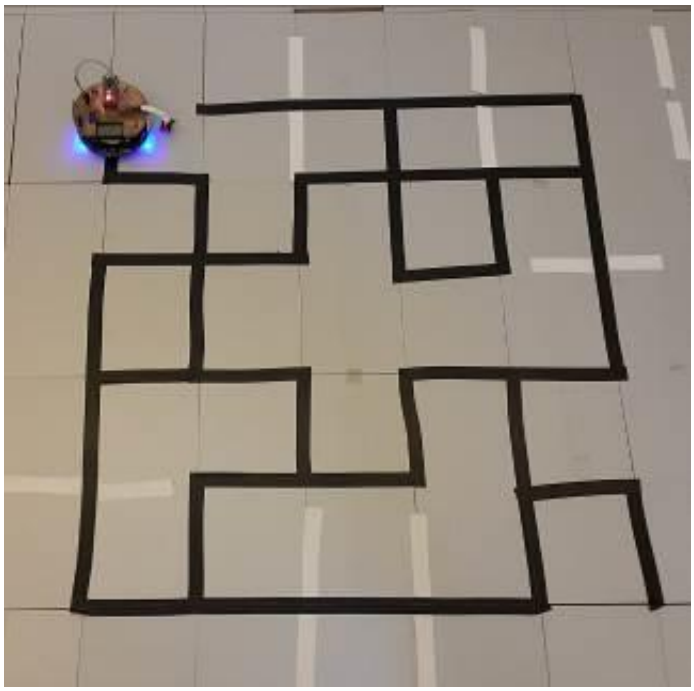


Şekil 4.30 Test Labirenti 1 Bilgisayar Çıktısı (L1)

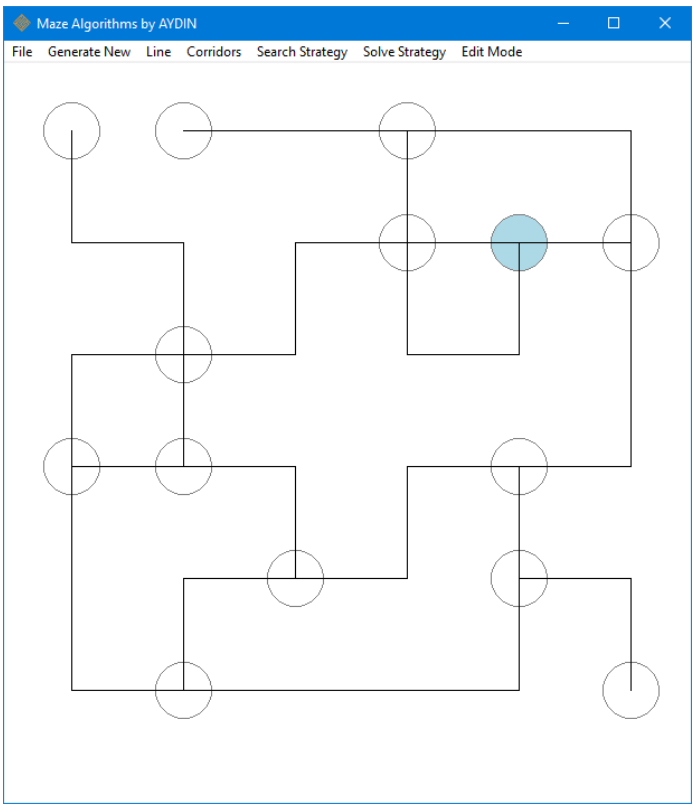
Şekil 4.31 Test Labirenti 2 (L2)



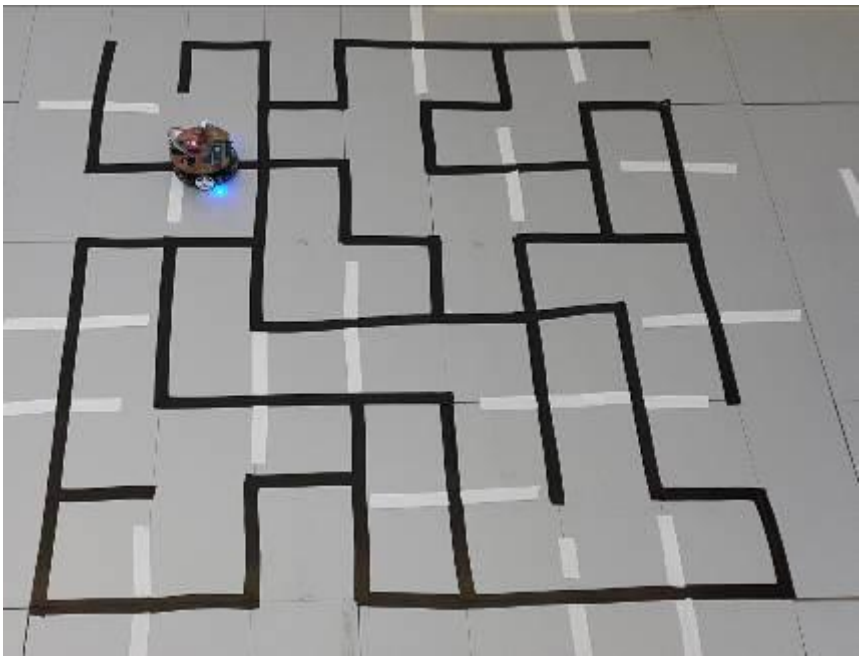
Şekil 4.32 Test Labirenti 2 Bilgisayar Çıktısı (L2)



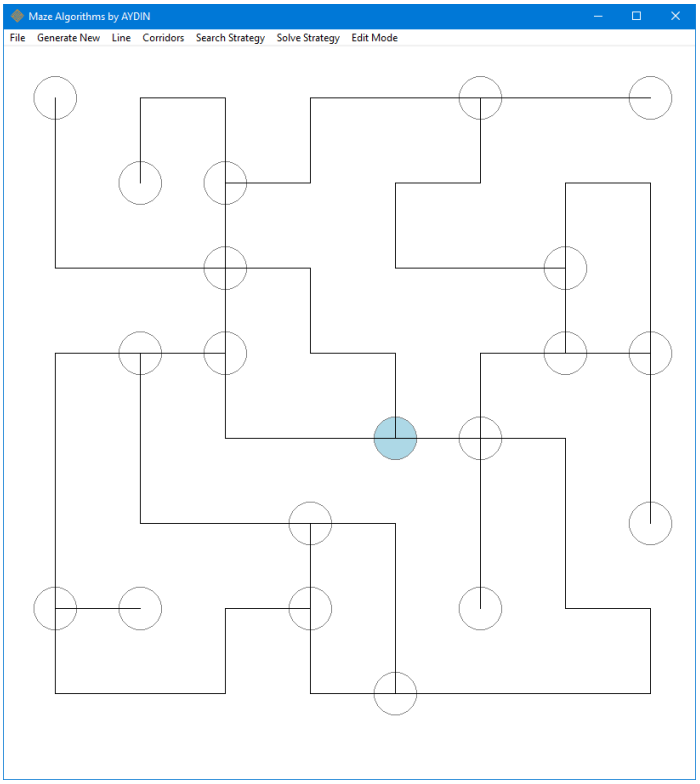
Şekil 4.33 Test Labirenti 3 (L3)



Şekil 4.34 Test Labirenti 3 Bilgisayar Çıktısı(L3)



Şekil 4.35 Test Labirent 4 (L4)



Şekil 4.36 Test Labirent 4 Bilgisayar Çıktısı (L4)

Test ortamlarının analizinde sol ve sağ duvar takip algoritmaları ortamın tamamının tanımlanmasında başarılı olamamışlardır. Ancak tek çözümlü ve basit ortamların tanımlanmasında başarılı olmuşlardır. Bu sebeple kıyaslama tablosunda yer verilmemiştir. BFS algoritması ile tanımlama yapılırken robot aynı düğüme çok fazla kez geldiği için tanımlama zamanını çok uzatmaktadır. Optimizasyon sonucunda ise bu azaltılabilmektedir. Kullanılan algoritmalar arasında en verimli sonucu DFS algoritması vermektedir. DFS + Dijkstra'nın birlikte çalışması ile alan tarama daha da kısa mesafede yapılabilmektedir. Fiziksel ortamın kısıtlı olması sebebi ile testlerde kullanılan labirentler çok büyük değildir. Bu sebeple hibrid algoritmaların sonuçları ile yalın algoritmaların sonuçları yakın çıkmıştır. Daha büyük ortamlarda bu farkın daha da artması beklenmektedir. Ancak tüm ortamlar için hibrid algoritmaların daha verimli olduğu gözükmektedir. Akıllı duvar takip algoritmasında tüm labirentin tanımlanması sağlamıştır. Ancak labirentin taranmasında duvar takibi kullanıldığı için aynı düğümlere birden fazla kez uğrayabilmektedir. Bu sebeple sonuçlar DFS algoritmasından iyi değildir. Tablo 4.5'de alan tarama algoritmaları ile tüm ortamın analizi sonucunda kat edilen birim mesafeler verilmiştir. Bir birim mesafe 12cm'ye denk gelmektedir.

Tablo 4.5 Labirentin Tanımlanmasında Kullanılan Algoritmaların Kıyaslanması

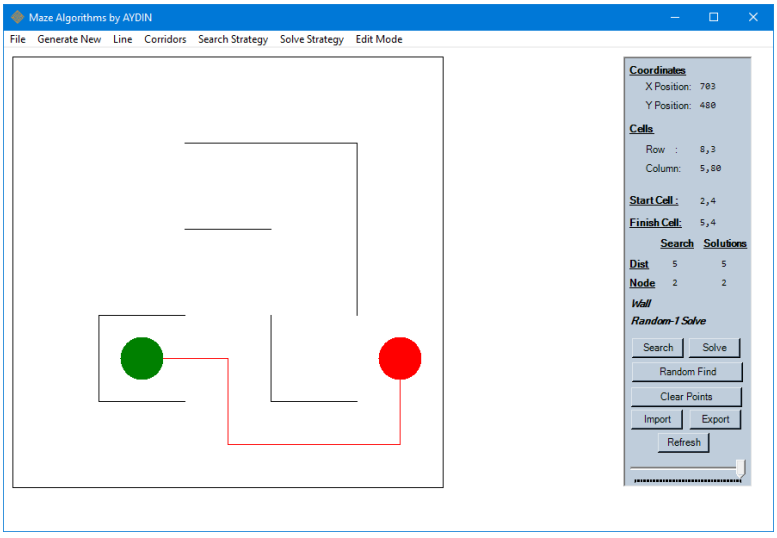
	L1	**L2**	**L3**	**L4**
DFS	39	76	58	112
BFS	90	169	156	292
DFS+DİJKSTRA	37	72	56	94
BFS+DİJKSTRA	81	137	121	233
AKILLI DUVAR	66	117	95	195

4.8. Belirtilen İki Nokta Arasındaki En Kısa Yolun Bulunması

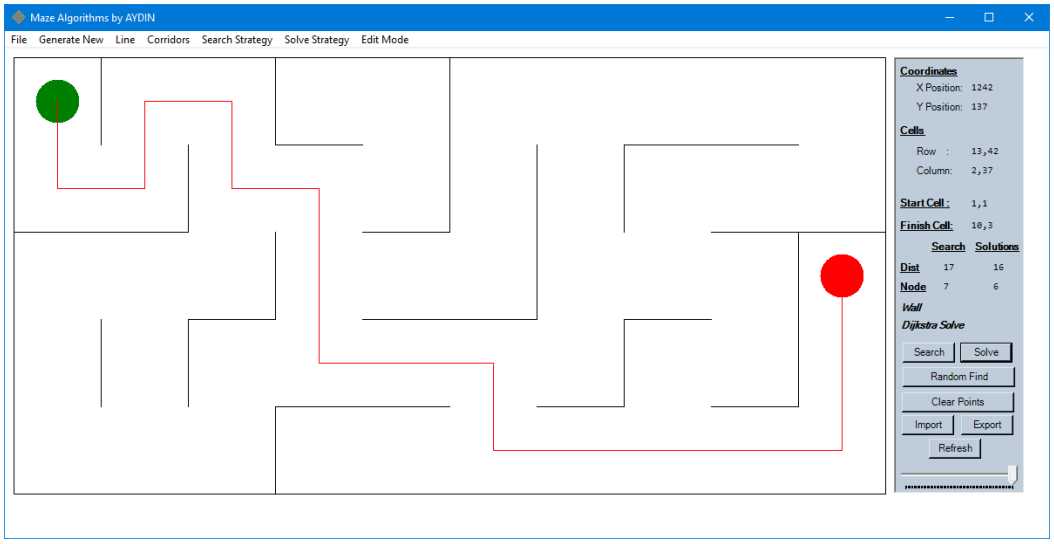
Labirentin tanımlanmasının ardından kullanıcının seçimini yapacağı iki nokta arasındaki en kısa yolu bulmak için iki farklı algoritmadan yararlanılmıştır. Bu algoritmalar sezgisel A* ve GBFS algoritmalarıdır. Bu iki algoritma da tanımlama sonrasında seçilen iki nokta arasındaki rotayı çıkarabilmektedir. Eğer tanımlama aşamasında hedef nokta seçilecek olursa duvar takip algoritmaları ile çözüm yaptırılabilmektedir. Duvar takip algoritmaları ile çözüm birden çok çözümlü labirentler için her zaman verilemeyebilir. A* ve GBFS algoritmaları ise her ortam için bir çözüm sunabilmektedir. Sezgisel olan bu algoritmalarda sezgisel mesafe fonksiyonu dikey kenarlar olarak hesaplanmıştır.

Bölüm 4.7'de kullanılan test ortamları için seçilen iki nokta üzerinde en kısa yol algoritması çalıştırıldığında en kısa yol olarak test labirenti 1 için Şekil 4.37'deki rota bulunur. Grafik yapısı üzerinde bu rota robota yüklenerek robotun hareket etmesi sağlanır. Robot iki farklı algoritma ile en kısa rotanın bulunması için test edilmiş ve test sonuçları tüm labirent için Tablo 4.6'da gösterilmiştir.

Test labirenti 1 için seçilen başlangıç noktası 1,3 bitiş noktası ise 4,3 olarak seçilmiştir. Bu seçime göre çözüm sağlanmıştır.

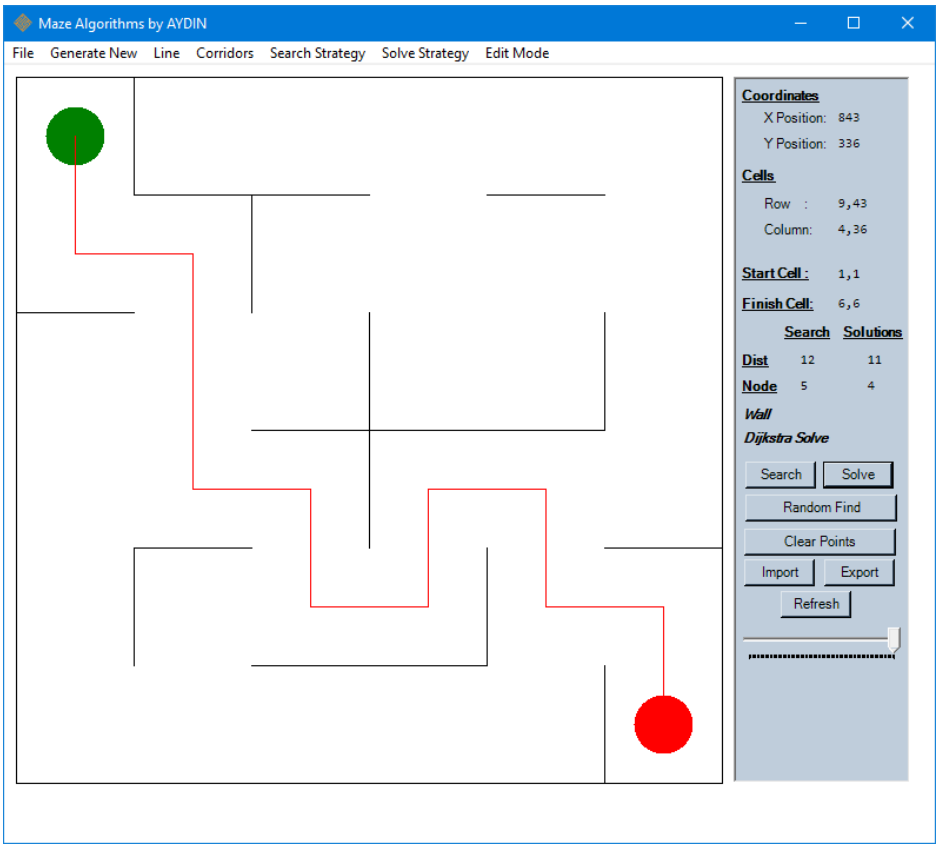


Şekil 4.37 Test Labirenti 1 (L1) için En Kısa Rota Bulunması

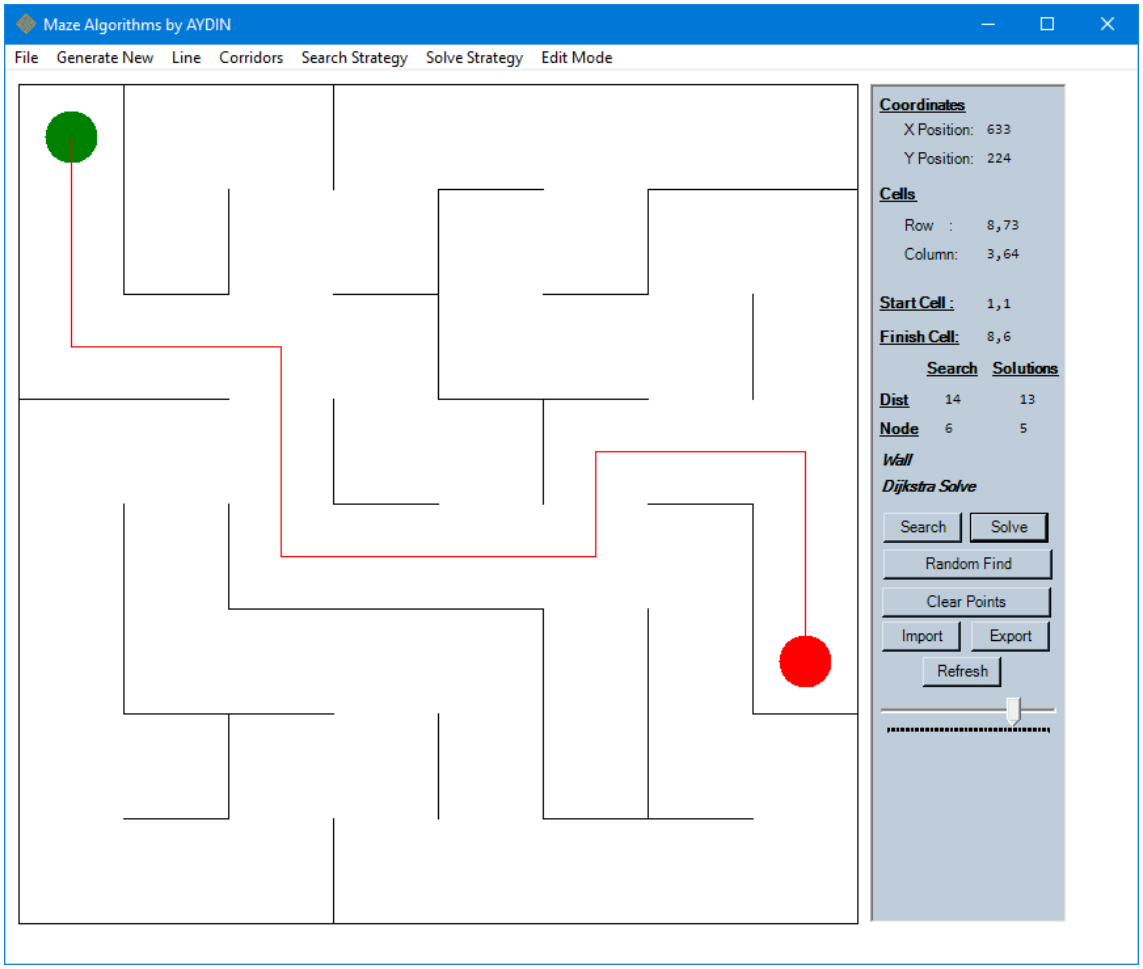


Şekil 4.38 Test Labirenti 2 (L2) için En Kısa Rota Bulunması

Test labirenti 2 için başlangıç noktası 0,0 ve hedef noktası 9,2 seçilmiştir. Bu seçim noktaları için en kısa rota çizdirilmiş ve düğümlerdeki dönüş yönleri tespit edilmiştir.



Şekil 4.39 Test Labirenti 3 (L3) için En Kısa Rota Bulunması



Şekil 4.40 Test Labirenti 4 (L4) için En Kısa Rota Bulunması

Test labirenti 3 için başlangıç noktası 0,0 ve bitiş noktası 5,5 olarak seçilmiştir. Benzer şekilde son test ortamının başlangıç noktası 0,0 ve bitiş noktası 7,5 olarak seçilmiştir. Tüm labirentlerin belirtilen noktaları için en kısa yol hesabı yapıldığında, robotun sonuca ulaşabileceği mesafe birim adım olarak verilmiştir. Aynı şekilde robot düğümlere geldiği zaman sonuca ulaşabilmesi için seçeceği dönüş yönleri de tespit edilmiştir. Bu dönüş yönleri ile robotun hedefe ulaşabilmesi için başlangıç noktasından hareket ederek uğrayacağı düğümler sırası ile Tablo 4.6'da gösterilmiştir.

İki farklı yöntem için sonradan belirtilen noktalar üzerinde en kısa yolun bulunması için yapılan hesaplamalarda A* algoritması her zaman en kısa mesafeyi vermektedir. GBFS algoritması ise sonuca ulaşmakta fakat en kısa yolu garanti etmemektedir. Ancak yapılan iş yükü, hesaplama ve kodlama olarak GBFS daha basit ve hızlı çözüm vermektedir.

Tablo 4.6 En Kısa Yol Hesabı için Alınan Veriler

Labirent	Mesafe (Birim Yol)	Dönme Komutları	Rota	Algoritma
L1	5	S,R,L	1,3 - 1,4 - 2,4 – 4,3	A*
	5	S,R,L	1,3 - 1,4 - 2,4 – 4,3	GBFS
L2	17	S,L,R,S, L,R,L,S	0,0 - 2,1 - 3,1 - 3,2 - 3,3 - 5,3 - 5,4 - 7,4 - 9,2	A*
	35	S,L,R,L,S, R,L,L,L,S	0,0 - 2,1 - 3,1 - 3,2 - 6,0 - 7,2 - 6,2 - 5,3 - 5,4 - 7,4 - 9,2	GBFS
L3	12	S,S,L, L,R,L,	0,0 - 1,2 - 1,3 - 2,4 - 4,3 - 4,4 - 5,5	A*
	14	S,L,S,S, R,L,L	0,0 - 1,2 - 1,3 - 3,1 - 4,1 - 5,1 - 4,3 - 4,4 - 5,5	GBFS
L4	14	S,R,S,S, L,S,R	0,0 - 2,2 - 2,3 - 4,4 - 5,4 - 6,3 - 7,3 - 7,5	A*
	14	S,R,S,S, L,S,R	0,0 - 2,2 - 2,3 - 4,4 - 5,4 - 6,3 - 7,3 - 7,5	GBFS

BÖLÜM 5

SONUÇLAR VE TARTIŞMA

5.1. Sonuçlar

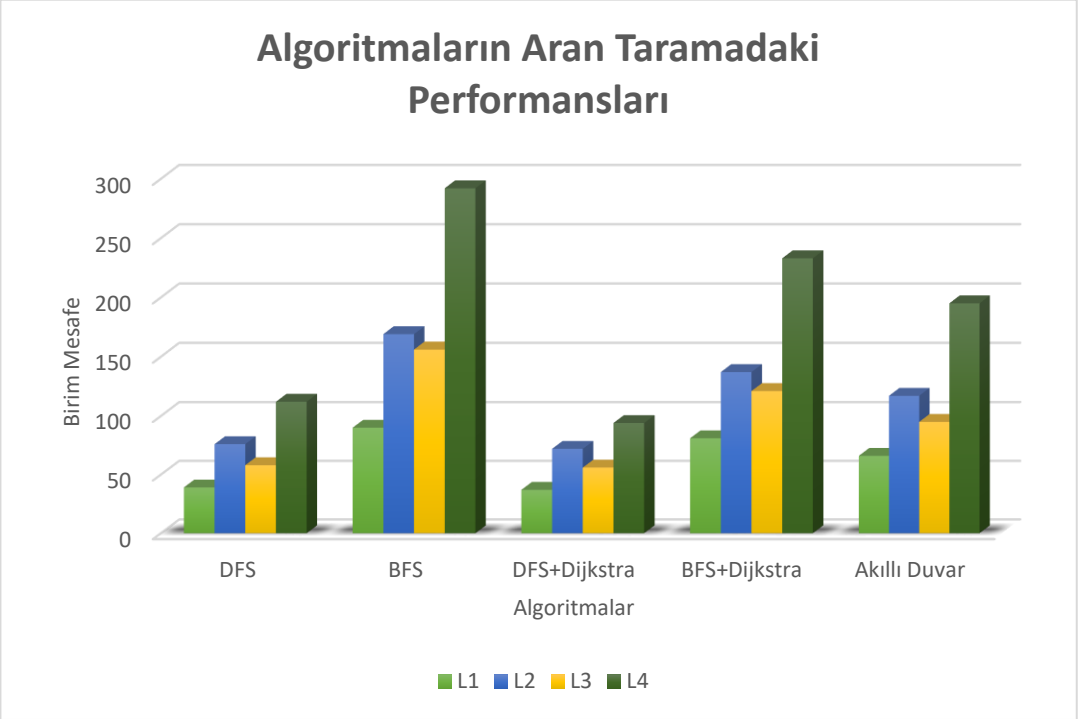
Gezgin robotun bilinmeyen bir ortamda otonom olarak hareket edebilmesi için geliştirilen bu tez çalışmasında iki ana konu üzerinde çalışılmıştır. Bunlardan ilki ortamın tanımlanmasıdır. Bunun için gezgin robot önceden bilmediği ortamı keşfedebilmesi için iki farklı metot kullanılmıştır. Bunlardan biri görüntü işleme ile ortamın tanımlanmasıdır. Bu aşamada bir arayüz geliştirilmiştir. Geliştirilen arayüz ile görüntüsü girilen ortamın haritasını çıkartılmaktadır. İkinci kısım ise robotun ortamda gezinerek ortamı analiz edebilmesidir. Bu amaçla ortamın analizi için robotun ortamda fiziksel olarak gezinerek ortamı keşfedebilmesi sağlanmıştır. Gezinerek ortamın taranması için duvar takip algoritmalar ve grafik arama algoritmaları ayrı ayrı test edilmiştir. Ayrıca grafik arama algoritmalarında arama için kat edilen mesafeyi optimize etmek için Dijkstra algoritması kullanılmıştır. Ortamın fiziksel robot ile keşfi için sol duvar takip, sağ duvar takip, akıllı duvar takip, DFS, BFS, DFS+Dijkstra ve BFS+Dijkstra algoritmaları ile ayrı ayrı testler yapılmıştır.

Sol ve Sağ Duvar Takip algoritmaları, hedefe birden fazla yol ile gidilen, karmaşık ortamların analizinde kullanılması neredeyse imkansızdır. Bu algoritmalarda tarama yapılırken robotun döngüye girdiği tespit edilmiştir. Ancak bu algoritmalar tek çözümlü basit yapıların taranmasında kullanılabilirler.

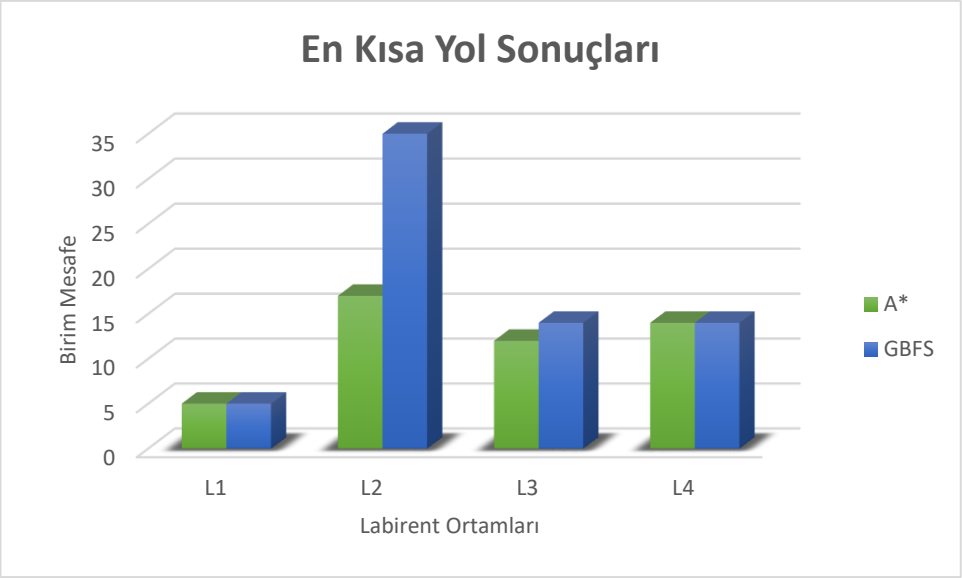
Akıllı duvar takip algoritması, sol ve sağ duvar takip algoritmalarında oluşan olumsuzluklar için geliştirilmiştir. Klasik duvar takip algoritmalarından farklı olarak yol ve düğüm bilgisi kaydedilmektedir. Robot döngüye girdiğini anlayarak duvar takip stratejisini değiştirilmiş ve tüm ortamın tanımlanması sağlanabilmiştir.

Bu algoritmanın dezavantajı, ortamın taramasında kat edilen mesafenin en kısa mesafe olmayışıdır. Gerçek robotun DFS algoritması ile arama yapması en iyi şekilde sonuç veren algoritmalardan birisidir. Tek robotlu yapılan testlerde ortamın tanımlanması için iyi sonuçlar vermektedir. DFS'nin çalışması esnasında geliştirilen Dijkstra optimizasyonu ile DFS algoritması daha da verimli hale dönüştürülmüştür. Bu sayede en verimli ortam analizi sağlanmıştır.

BFS algoritması genişlik taraması yaptığı için robotun ilerleme sürecinde aynı düğümden birden fazla kez geçmesine neden olmaktadır. Bu tarama esnasında kat edilen mesafeyi oldukça arttırır. Algoritma ile ortamın tamamıyla tanımlanması sağlanmaktadır. Aynı şekilde, BFS algoritması tarama esnasında Dijkstra ile optimize edilerek daha kısa mesafede ortamın analiz edildiği gözlemlenmiştir. L1, L2, L3 ve L4 test ortamlarının tanımlanmasında elde edilen sonuçlar Şekil 5.1'deki grafikte gösterilmiştir.



Şekil 5.1 Ortamın Tanımlanmasında Algoritmaların Kıyaslanması



Şekil 5.2 En Kısa Yol Algoritmalarının Kıyaslanması

Ortamın tanımlanmasının ardından robotun kullanıcı tarafından seçilen iki nokta arasına en kısa yoldan gidilebilmesi için bazı algoritmalar kullanılmıştır. Tarama esnasında kullanılan duvar takip algoritmaları ile sonuç belirlenerek ve gidilen yanlış yönler optimize edilerek sonuca giden rota çizdirilmektedir. Fakat bu tarama esnasında yapıldığı için bazı dezavantajları vardır. Her zaman en kısa rotayı vermeyebilir.

En kısa rotanın tanımlanmış ortam üzerinde bulunması için sezgisel çalışan GBFS ve A* algoritmaları kullanılmıştır. Bu iki algoritma ile yapılan testlerde A* her zaman en iyi sonucu verdiği gözlemlenmiştir. GBFS ise sonuca her zaman ulaşmakta fakat bazı ortamlar için en kısa rotayı vermemektedir. Bu iki algoritmanın test ortamları için elde edilen en kısa rota sonuçları birim mesafe bazında Şekil 5.2'deki grafikte gösterilmiştir.

Çalışmada pratik uygulama için Pololu 3pi robotu geliştirilerek kullanılmıştır. Ortam olarak çizgi labirent kullanılmıştır. Ortam DFS algoritması ile rastgele çizgi labirentler şeklinde üretilmiş ve fiziksel olarak uygulanmıştır. Bilgisayar üzerinde geliştirilen bir arayüz vasıtası ile robot ile haberleşerek ortam bilgileri alınmıştır.

Geliştirilen algoritmalar gerçek gezgin robot ile test edilmiştir. Testler sonucunda robotun otonom olarak ortamı tanımlaması ve seçilen iki nokta arasına en kısa yoldan ulaşabilmesi sağlanmıştır.

5.2. Tartışma

Bu tez kapsamında geliştirilen algoritmaların testleri iki boyutlu gerçek çevre ile denenmiştir. Testler 35 m^2'lik laboratuvarın bir kısmında yapılmıştır. Bu sebeple test labirentlerinin boyutları kısıtlı olmuştur. Daha büyük ortamlarda testlerin yapılma imkanı olursa arama kısmında geliştirilen optimizasyon işlemlerinin daha faydalı olduğu görülebilir. Ancak ortam büyüdüğü zaman robotun, enerji, haberleşme gibi fonksiyonlarını sağlayan devrelerin de geliştirilmesi gerekmektedir.

Tez kapsamında geliştirilen yazılımlar daha verimli bir şekilde kodlanabilirse daha hızlı çalışma veya bilgisayar olmadan robot üzerinde çalışma sağlanabilir.

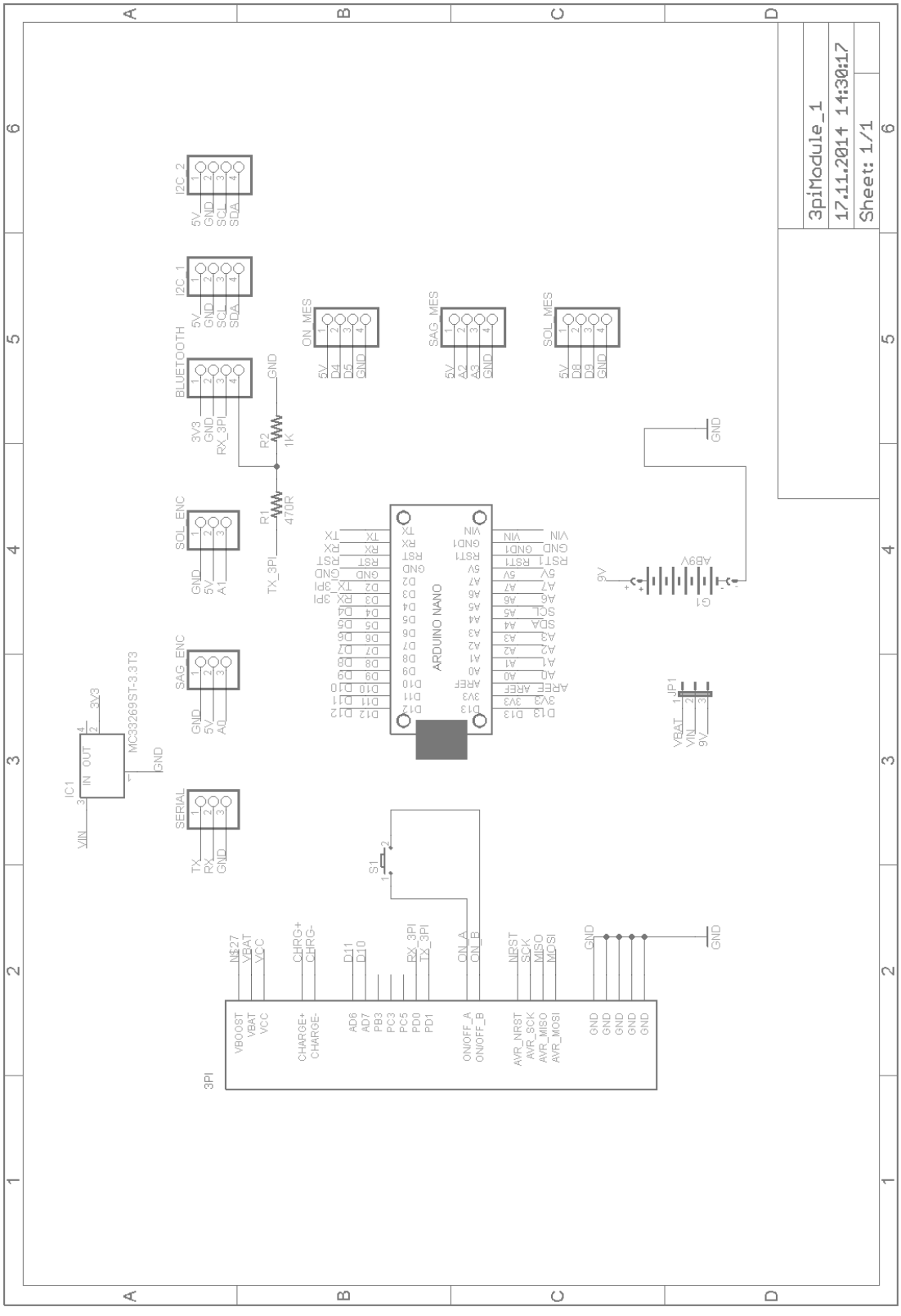
5.3. Gelecek Çalışmalar ve Öneriler

Geliştirilen algoritmaların gerçek 3D çevre için test edilmesi planlanmaktadır. Bu çalışma kapsamında kullanılan iki boyutlu ortamın yerine, robota farklı algılayıcılar yerleştirilerek gerçek bir ortamın tanımlanması ve verilen işlevi yerine getirilmesi çalışmalarının yapılması planlanmaktadır. Bu kapsamda ticari bir temizlik veya çim biçme robotunun, ortamın haritasının çıkartarak daha verimli çalışabilmesi sağlanabilir. Yine bu tez kapsamında geliştirilen tüm yazılımların robot üzerinde çalıştırılması için optimizasyon çalışmaların da yapılması planlanmaktadır.

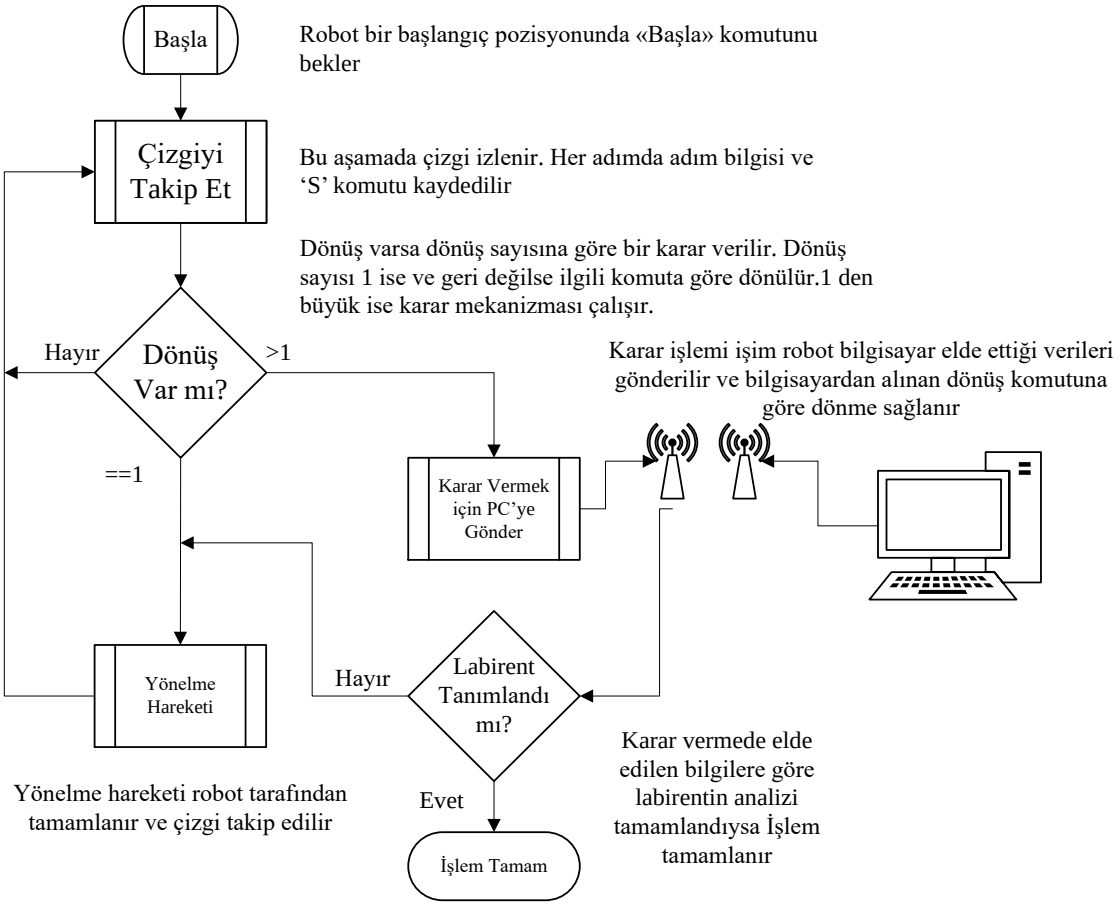
Bu fiziksel geliştirmelere ek olarak çevrim içi arama yapan A*, Yapay sinir ağları, makine öğrenmesi gibi farklı yapay zeka algoritmaları kullanılarak da tarama ve hedefe ulaşma çalışmalarında iyileştirmelerin yapılması hedeflenmektedir.

EKLER

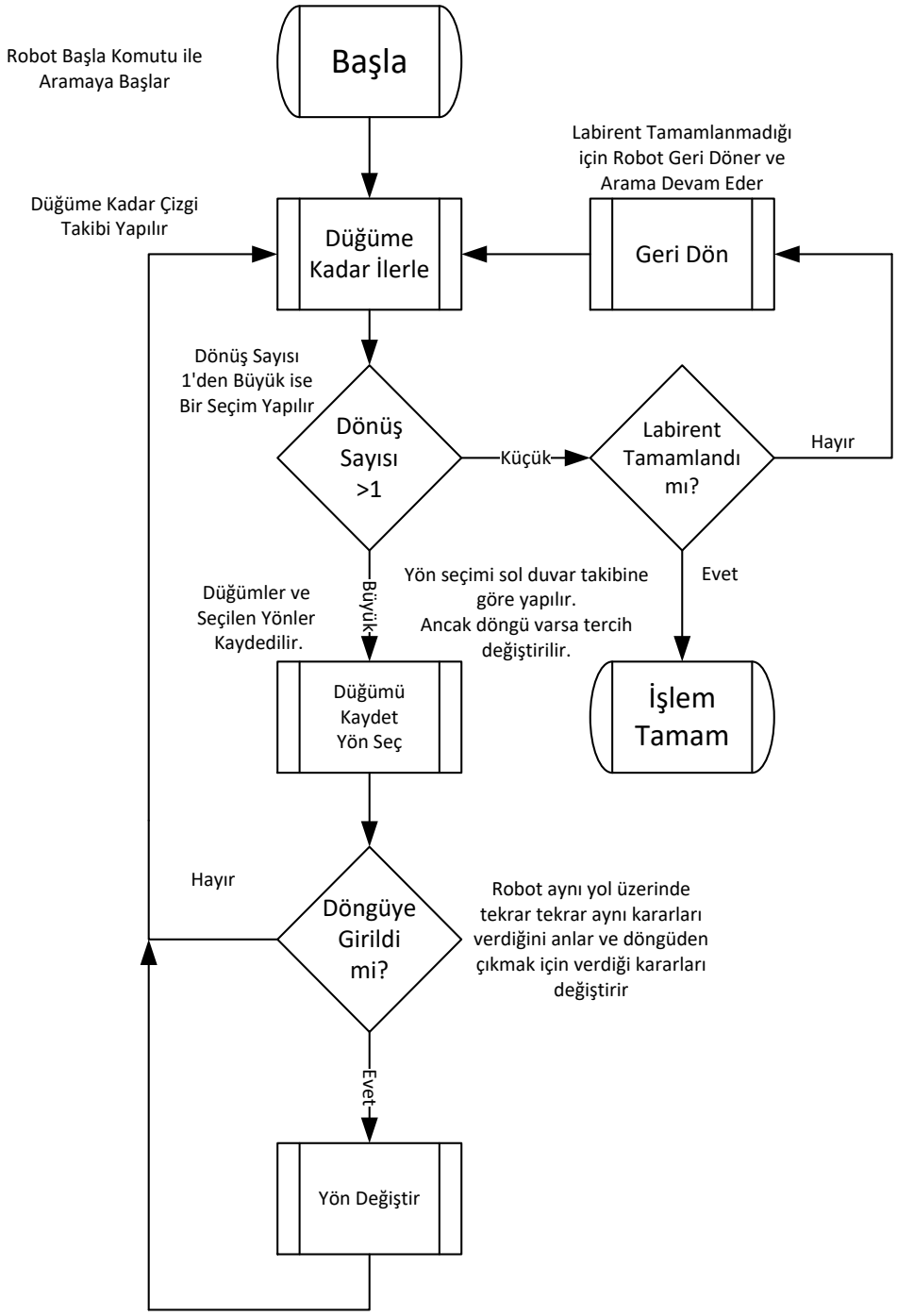
Ek-A Tasarlanan Ek Elektronik Devre



Ek- B Robot Çalışma Algoritması



Ek- C Akıllı Duvar Takip Algoritması



KAYNAKLAR

[1] Turing, A., ve arkş., ***Can automatic calculating machines be said to think?(1952)***. B. Jack Copeland, 2004: p. 487.

[2] Russell, S.J., ve arkş., ***Artificial intelligence: a modern approach***. Vol. 2. 1995: Prentice hall Englewood Cliffs.

[3] Crowley, J.L., ***Navigation for an Intelligent Mobile Robot***. IEEE Journal on Robotics and Automation, 1985. VOL. RA-I, NO. 1.

[4] Zelinsky, A., ***A Mobile Robot Exploration Algorithm***. Ieee Transactions on Robotics and Automation, 1992. 8(6): p. 707-717.

[5] Rao, N.S.V., ***Algorithmic Framework for Learned Robot Navigation in Unknown Terrains***. Computer, 1989. 22(6): p. 37-43.

[6] Oommen, B., ve arkş., ***Robot navigation in unknown terrains using learned visibility graphs. Part I: The disjoint convex obstacle case***. Robotics and Automation, IEEE Journal of, 1993. 3(6): p. 672-681.

[7] Rao, N.S., ve arkş., ***Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms***. 1993: Citeseer.

[8] Liscano, R., ve arkş., ***Using a blackboard to integrate multiple activities and achieve strategic reasoning for mobile-robot navigation***. IEEE expert, 1995. 10(2).

[9] Botelho, S.S., ve arkş. ***High speed neural control for robot navigation***. in *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*. 1996. IEEE.

[10] Lee, S., T.M. Adams, and B.-y. Ryoo, ***A fuzzy navigation system for mobile construction robots***. Automation in Construction, 1997. 6(2): p. 97-107.

[11] Lu, F. and E. Milios, ***Robot pose estimation in unknown environments by matching 2d range scans***. Journal of Intelligent and Robotic Systems, 1997. 18(3): p. 249-275.

[12] Dain, R.A., ***Developing mobile robot wall-following algorithms using genetic programming***. Applied Intelligence, 1998. 8(1): p. 33-41.

[13] Ryu, J.C., F.C. Park, and Y.Y. Kim, ***Mobile robot path planning algorithm by equivalent conduction heat flow topology optimization***. Structural and Multidisciplinary Optimization, 2011. 45(5): p. 703-715.

[14] Cheng, P.-Y. and P.-J. Chen, ***Navigation of mobile robot by using D++ algorithm***. Intelligent Service Robotics, 2012. 5(4): p. 229-243.

[15] Davoodi, M., ve arkş., ***Multi-objective path planning in discrete space***. Applied Soft Computing, 2012.

[16] Hun, C.-W., H.-C. Chen, and M.-F. Hwang, ***On-line algorithm for optimal 3D path planning***. Computer Applications in Engineering Education, 2012. 20(4): p. 713-720.

[17] Ibrahim Elshamarka, A.B.S.S., ***Design and Implementation of a Robot for Maze-Solving using Flood-Fill Algorithm***. International Journal of Computer Applications (0975 – 8887), 2012. Volume 56– No.5.

[18] Kala, R., ***Multi-robot path planning using co-evolutionary genetic programming***. Expert Systems with Applications, 2012. 39(3): p. 3817-3831.

[19] KÜÇÜKYILDIZ, G., ***Mobil Robot Platformu Üzerinde DSP Tabanlı Gerçek Zamanlı Çalışan Şerit Tespiti Algoritmalarının Geliştirilmesi ve Optimizasyonu***. Kocaeli Üniversitesi Yüksek Lisans Tezi, 2012.

[20] Mae, J., ve arkş., ***Modified Line-Maze Algorithm for Mobile Robot Navigation***. Procedia Engineering, 2012. 50: p. 740-747.

[21] Murat, Y. and İ. Serhat, ***Bir Mobil Robotun Hedef Noktaya Erişimi ve Toplanan Verilerin RF ile Transferi***. FEEB 2012: p. 270-274.

[22] Orhan KÜÇÜKCEYLAN, T.Y., Abdullah SEZGİN, ***Enine Arama Algoritmasını Kullanarak En Kısa Yol Probleminin Çözümünün Lego Mindstorm ile Gerçekleştirilmesi*** EMO, 2012.

[23] Rivera, G.G.A., ***Path planning for general mazes***, in *Electrical Engneering*. 2012, Missouri University of Science and Tecnology. p. 45.

[24] Seyed Mohammad Mavaei, H.R.I., ***Line Segmentation and Slam For Rescue Robots In Unknown Environments***. World Applied Sciences Journal, 2012: p. 1627-1635.

[25] Song, X., ve arkş. ***Autonomous mobile robot navigation using machine learning***. in *Information and Automation for Sustainability (ICIAfS), 2012 IEEE 6th International Conference on*. 2012. IEEE.

[26] Suat Karakaya, G.K., Hasan Ocak, Zafer Bingül, ***Mobil Robot Platformu Üzerinde Engel Algılanması ve Optimal Yönün Belirlenmesi***. 2012.

[27] Zhu, Y., ve arkş., ***A new hybrid navigation algorithm for mobile robots in environments with incomplete knowledge***. Knowledge-Based Systems, 2012. 27: p. 302-313.

[28] Garrido, S., ve arkş., ***General Path Planning Methodology for Leader-Follower Robot Formations***. International Journal of Advanced Robotic Systems, 2013. 10.

[29] Meléndez, A., ve arkş., ***Optimal Design of the Fuzzy Navigation System for a Mobile Robot Using Evolutionary Algorithms***. International Journal of Advanced Robotic Systems, 2013. 10.

[30] Sun, C., ve arkş., ***Automatic Navigation for Rat-Robots with Modeling of the Human Guidance***. Journal of Bionic Engineering, 2013. 10(1): p. 46-56.

[31] Zhu, Y., ve arkş., ***A hybrid navigation strategy for multiple mobile robots***. Robotics and Computer-Integrated Manufacturing, 2013. 29(4): p. 129-141.

[32] Ostafew, C.J., ve arkş., ***Learning-based Nonlinear Model Predictive Control to Improve Vision-based Mobile Robot Path Tracking***. Journal of Field Robotics, 2016. 33(1): p. 133-152.

[33] Contreras-Cruz, M.A., V. Ayala-Ramirez, and U.H. Hernandez-Belmonte, ***Mobile robot path planning using artificial bee colony and evolutionary programming***. Applied Soft Computing, 2015. 30: p. 319-328.

[34] Bishop, R.H., ***The Mechatronics Handbook, -2 Volume Set***. 2006: CRC Press.

[35] Auslander, D.M., ***What is mechatronics?*** Mechatronics, IEEE/ASME Transactions on, 1996. 1(1): p. 5-9.

[36] Bradley, D.A., ve arkş., ***What is mechatronics?*** 1991: Springer.

[37] Arkin, R.C., ***Motor schema—based mobile robot navigation***. The International journal of robotics research, 1989. 8(4): p. 92-112.

[38] Walter, W.G., ***mitation of Life***. 1950.

[39] Kortenkamp, D., R.P. Bonasso, and R. Murphy, ***Artificial intelligence and mobile robots: case studies of successful robot systems***. 1998: MIT Press.

[40] Ergezer, H. and K. Leblebicioglu, ***Path Planning for UAVs for Maximum Information Collection***. IEEE Transactions on Aerospace and Electronic Systems, 2013. 49(1): p. 502-520.

[41] ***Pioneer LX Research Platform***. 2016 [Erişim Tarihi 2016 09.10.2016]; Erişim Yeri: http://www.mobilerobots.com/ResearchRobots/PioneerLX.aspx.

[42] Aydın, G., ve arkş., ***Mobil Robotların Deneysel Simülasyonunda Kullanılan Platformların İncelenmesi***, in *TOK 2014*, P.D.Z. Bingül, Editor. 2014: Kocaeli. p. 8-13.

[43] Kortenkamp, D., *Artificial Intelligence and Mobile Robots Successes and Challenges*. 2000.

[44] Brooks, R.A., ***A robust layered control system for a mobile robot***. Robotics and Automation, IEEE Journal of, 1986. 2(1): p. 14-23.

[45] Dudek, G. and M. Jenkin, ***Computational principles of mobile robotics***. 2010: Cambridge university press.

[46] Nehmzow, U., ***Mobile robotics: a practical introduction***. 2012: Springer Science & Business Media.

[47] Ribeiro, M.I. and P. Lima, ***Kinematics models of mobile robots***. Instituto de Sistemas e Robotica, 2002: p. 1000-1049.

[48] Mireles Jr, J., ***Kinematic models of mobile robots***. Automation and Robotics Research Institute, University of Texas at Austin, 2004.

[49] Siegwart, R., I.R. Nourbakhsh, and D. Scaramuzza, ***Introduction to autonomous mobile robots***. 2011: MIT press.

[50] Junior, L.A., ve arkş., ***A Low-Cost and Simple Arduino-Based Educational Robotics Kit***. 2013.

[51] Grover, R., ve arkş. ***A competition-based approach for undergraduate mechatronics education using the arduino platform***. in *Interdisciplinary Engineering Design Education Conference (IEDEC), 2014 4th*. 2014. IEEE.

[52] ***Arduino Web***. 2016 [Erişim Tarihi 2016 14.10.2016]; Erişim Yeri: https://www.arduino.cc/.

[53] Barrett, S.F., ***Arduino Microcontroller Processing for Everyone!*** Synthesis Lectures on Digital Circuits and Systems, 2013. 8(4): p. 1-513.

[54] Electronics, P.R.a. ***Pololu 3pi Robot***. 2016 [Erişim Tarihi 2016 02.01.16]; Erişim Yeri: https://www.pololu.com/product/975.

[55] Turing, A.M., ***Intelligent machines***. Ince, DC (Ed.), 1992. 5.

[56] O'Donnell, J.J. and P.R. Doob, ***The Idea of the Labyrinth from Classical Antiquity through the Middle Ages***. 1992, JSTOR.

[57] Cai, Z., L. Ye, and A. Yang. ***FloodFill Maze Solving with Expected Toll of Penetrating Unknown Walls for Micromouse***. in *High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*. 2012. IEEE.

[58] Hanafi, D., Y.M. Abueejela, and M.F. Zakaria, ***Wall follower autonomous robot development applying fuzzy incremental controller***. Intelligent Control and Automation, 2013. 4(01): p. 18.

[59] del Rosario, J.R.B., ve arkş. ***Modelling and Characterization of a Maze-Solving Mobile Robot Using Wall Follower Algorithm***. in *Applied Mechanics and Materials*. 2014. Trans Tech Publ.

[60] Donnarumma, F., ve arkş., ***Learning programs is better than learning dynamics: A programmable neural network hierarchical architecture in a multi-task scenario***. Adaptive Behavior, 2015: p. 1059712315609412.

[61] Tarjan, R., ***Depth-first search and linear graph algorithms***. SIAM journal on computing, 1972. 1(2): p. 146-160.

[62] Wikipedia. ***Depth-first search***. 2016 [Erişim Tarihi 2016 5.10.2016]; Erişim Yeri: https://en.wikipedia.org/wiki/Depth-first_search.

[63] Leiserson, C.E. and T.B. Schardl. ***A work-efficient parallel breadth-first search algorithm (or how to cope with the nondeterminism of reducers)***. in *Proceedings of the twenty-second annual ACM symposium on Parallelism in algorithms and architectures*. 2010. ACM.

[64] Subramanian, M.B., K. Sudhagar, and G. RajaRajeswari, ***Optimal Path Forecasting of an Autonomous Mobile Robot Agent Using Breadth First Search Algorithm***. 2014.

[65] Moore, E.F., ***The shortest path through a maze***. 1959: Bell Telephone System.

[66] Skiena, S.S., ***The algorithm design manual: Text***. Vol. 1. 1998: Springer Science & Business Media.

[67] Dijkstra, E.W., ***A note on two problems in connexion with graphs***. Numerische mathematik, 1959. 1(1): p. 269-271.

[68] Crowley, J., ***Navigation for an intelligent mobile robot***. IEEE Journal on Robotics and Automation, 1985. 1(1): p. 31-41.

[69] Noh, S.W., ve arkş., *Implementation of Autonomous Navigation Using a Mobile Robot Indoor*, in ***Advances in Computer Science and Ubiquitous Computing***. 2015, Springer. p. 751-756.

[70] Kanal, L. and V. Kumar, ***Search in artificial intelligence***. 2012: Springer Science & Business Media.

[71] Bonet, B. and H. Geffner, *Planning as heuristic search: New results*, in ***Recent Advances in AI Planning***. 2000, Springer. p. 360-372.

[72] Rios, L.H.O. and L. Chaimowicz, *A survey and classification of a* based best-first heuristic search algorithms*, in ***Advances in Artificial Intelligence–SBIA 2010***. 2011, Springer. p. 253-262.

[73] Guo, J.H. and K.L. Su, *Greedy Algorithm Based Multiple Target Searching for Mobile Robots*, in ***Frontiers of Manufacturing and Design Science, Pts 1-4***, R. Chen, Editor. 2011, Trans Tech Publications Ltd: Stafa-Zurich. p. 1335-1339.

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

[75] Yao, J., ve arkş. ***Path planning for virtual human motion using improved A* star algorithm***. in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*. 2010. IEEE.

[76] Didier, K. and D. Jo. ***A flexible path generator for a mobile robot***. in *Advanced Robotics, 1991.'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*. 1991. IEEE.

[77] Kozlova, A., J.A. Brown, and E. Reading. ***Examination of representational expression in maze generation algorithms***. in *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. 2015. IEEE.

[78] Anh, H.P.H., T.T. Huan, and N.T. Nam, ***Novel Robust Walking for Biped Robot Using Adaptive Neural PID Controller***. 2014.

[79] Khan, A.A.M., ve arkş. ***Design and implementation of a robot for maze-solving with turning indicators using PID controller***. in *Informatics, Electronics & Vision (ICIEV), 2013 International Conference on*. 2013. IEEE.

[80] Yüksel, İ., ***Otomatik kontrol: sistem dinamiği ve denetim sistemleri***. 2011: Nobel Akademik Yayıncılık Danışmanlık.

[81] Kuo, B.C. and A. Bir, ***Otomatik kontrol sistemleri***. 2005: Literatür Yayınları.

[82] Åström, K.J. and T. Hägglund, ***Advanced PID control***. 2006: ISA-The Instrumentation, Systems and Automation Society.

[83] Barr, M., ***Pulse width modulation***. Embedded Systems Programming, 2001. 14(10): p. 103-104.

[84] ***Pulse width modulation***. 1951, Google Patents.

[85] Emgu, C., ***Emgu CV: OpenCV in .NET (C#, VB, C++ and more)***. Online: http://www. emgu. com, 2013.

[86] Emgu, C., ***Emgu cv***. URL http://emgucv. sourceforge. net, 2011.

[87] Venkatesh, M. and P. Vijayakumar, ***A simple bird's eye view transformation technique***. Int J Sci Eng Res, 2012. 3(5).

[88] Bradski, G. and A. Kaehler, ***Learning OpenCV: Computer vision with the OpenCV library***. 2008: " O'Reilly Media, Inc.".

[89] Bradski, G., A. Kaehler, and V. Pisarevsky, ***Learning-Based Computer Vision with Intel's Open Source Computer Vision Library***. Intel Technology Journal, 2005. 9(2).

[90] Skiena, S., ***The Algorithm Design Manual: Springer***. 2008, London.

[91] Trémaux, C.P. ***École Polytechnique of Paris (X: 1876)***. in *Proc. French Engineer of the Telegraph in Public Conference*. 2010.

[92] Misa, T.J. and P.L. Frana, ***An interview with edsger w. dijkstra***. Communications of the ACM, 2010. 53(8): p. 41-47.

[93] Liu, X. and D. Gong. ***A comparative study of A-star algorithms for search and rescue in perfect maze***. in *Electric Information and Control Engineering (ICEICE), 2011 International Conference on*. 2011. IEEE.

ÖZGEÇMİŞ

1986 yılında İstanbul'da doğdu. İlk ve ortaokulu İstanbul'da tamamladı. Liseyi İstanbul Bayrampaşa İstanbul Ticaret Odası Anadolu Teknik Lisesi Elektronik Bölümünde tamamladı. 2005 yılında başladığı Marmara Üniversitesi Teknik Eğitim Fakültesi Mekatronik Eğitimi Bölümünden 2009 yılında bölüm birincisi olarak mezun oldu. Aynı yıl Marmara Üniversitesi Fen Bilimleri Enstitüsü Mekatronik Programında Yüksek Lisans eğitimine ve proje mühendisliği olarak özel sektörde iş hayatına başladı. 2010 yılında yüksek lisans eğitimini tamamlayarak Trakya Üniversitesi İpsala Meslek Yüksekokulu elektronik programında öğretim görevlisi olarak çalışmaya başladı. Aynı yıl Trakya Üniversitesi FBE makine mühendisliği bölümünde doktora eğitimine başladı. 2011 yılından beri evli olan Aydın GÜLLÜ evli ve bir kız çocuğu babasıdır.

OCAK 2017

TEZ İLE İLGİLİ BİLİMSEL FAALİYETLER

[1] GÜLLÜ A, AKI M.O., KUŞÇU H., ARDA M., "***Mobil Robotların Deneysel Simülasyonunda Kullanılan Platformların İncelenmesi***", Otomatik Kontrol Ulusal Toplantısı (TOK 2014), Kocaeli (Eylül 2014)

[2] GULLU A., KUŞÇU, H.:" ***Maze Generation for Real Mobile Robot Application***", International Scientific Conference UNITECH'16 Gabrovo, Proceedings Volume-III, pp. 237-238, 18-19 November, Bulgaria, 2016

[3] GÜLLÜ A., KUŞÇU H., "***Mechanical and Software Design of the Warehouse Robot***", Conference on Advances in Mechanical Engineering Istanbul 2016 – ICAME2016, Yildiz Technical University, pp: 281-285, 11-13 May 2016, Istanbul, Turkey (Mayıs 2016)

[4] KUŞÇU, H. , A. GÜLLÜ, "***Mobil Robotlarda Yapay Zeka Tabanlı Yön Bulma Algoritmaları Geliştirilmesi***" (TUBAP Projesi No:2014/04)