YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KOOPERATİF GEZGİN ROBOTLARDA KONTROL TABANLI SEZGİSEL HAREKET PLANLAMA

Elektrik Mühendisi Özlem Feyza VAROL

FBE Elektrik Mühendisliği Anabilim Dalı Kontrol ve Otomasyon Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı: Doç. Dr. Haluk GÖRGÜN

İSTANBUL, 2009

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ....iv

KISALTMA LİSTESİ....v

ŞEKİL LİSTESİ....vi

ÇİZELGE LİSTESİ....vii

ÖNSÖZ viii

ÖZET ix

ABSTRACT....x

1. GİRİŞ....1

1.1 Problemin Tanımı....2
1.2 Tezin Genel Yapısı....3

2. LİTERATÜR TARAMASI ve ALT YAPI....4

2.1 Hareket Kontrolü için Kullanılan Yaklaşımlar....4
2.1.1 Klasik Teknikler....6
2.1.1.1 Potansiyel Alanlar Metodu....6
2.1.1.2 Hücrelere Bölme Metodu....7
2.1.2 Olasılıksal Tabanlı Yaklaşımlar....8
2.1.2.1 Yol Haritası Tabanlı Planlayıcılar....8
2.1.2.2 Ağaç Tabanlı Planlayıcılar....8
2.1.3 Sezgisel Teknikler....9
2.1.3.1 Genetik Algoritmalar....9
2.1.3.2 Bulanık Mantık....10
2.1.3.3 Yapay Sinir Ağları....10
2.2 Otonom Gezgin Robotlar....11
2.3 Notasyon ve Terminoloji....13

3. OTONOM GEZGİN ROBOTUN MODELLENMESİ....15

3.1 Sistemin Tanıtılması....15
3.2 Gezgin Robotun Kinematik ve Dinamik Denklemleri....16
3.2.1 Gezgin Robotun Kinematik Modeli....16
3.2.2 Gezgin Robotun Dinamik Modeli....18
3.3 DC Motorun Matematiksel Modeli....19
3.3.1 DC Motor Diferansiyel Denklemleri....19
3.3.2 DC Motor Transfer Fonksiyonu....20
3.3.3 DC Motor Parametrelerinin Belirlenmesi....21
3.4 Otonom Gezgin Robot ve DC Motorun Simulink Modelleri....22

4. ALGORİTMALAR....24

4.1 Hücrelere Bölme Metodu ve Arama Algoritmaları....24
4.1.1 A* (A-Yıldız) Arama Algoritması....27
4.1.1.1 A* (A-Yıldız) Arama Algoritmasının Özellikleri....29
4.1.1.2 A* Arama Algoritmasının Sözde Kodu....30
4.2 A* Arama Algoritmasıyla Yörünge Oluşturulması....31
4.2.1 Çevre Matrisi Oluşturma....32
4.3 Kontrol Algoritması....35
4.3.1 PID Denetleyici....35
4.3.2 PID Denetleyici ile Yörünge Takibi....37

5. SONUÇLAR....39

5.1 Benzetim Çalışmalarında Kullanılan Gezgin Robotun Parametreleri....39
5.2 Sistem Algoritmasının Tanımlanması....40
5.3 Benzetim Çalışmaları Sonuçları....41
5.4 Sonuçların Yorumlanması....49

6. GELECEK ÇALIŞMALAR İÇİN ÖNERİLER....50

6.1 Sistem Modelinin Oluşturulması....50
6.2 Yörünge Planlamada Kullanılan Algoritmalar....50
6.3 Yörünge Takibi (Hareket Kontrolü)....51
6.4 Kooperatif Kontrol Sisteminin Gerçeklenmesi....51
6.5 Sistem Ölçütleri....51

KAYNAKLAR....52

EKLER 55

Ek 1 A* Algoritmasına ait Matlab Kodları....56

ÖZGEÇMİŞ....76

SİMGE LİSTESİ

b	Motor sönüm katsayısı
D	Robotun tekerlekleri arasındaki mesafe
E	Düğümler arası kenar çizgileri
J	Toplam atalet
K	Motor tork sabiti
L	Motor endüktansı
N	Düğüm
P	DC Motor gücü
R	Tekerlek yarıçapı
T	Motor torku
V	Düğüm
v_c	Robotun merkez noktasına göre doğrusal hızı
v_L	Robotun sol tekerleğinin doğrusal hızı
v_R	Robotun sağ tekerleğinin doğrusal hızı
ω	Robotun açısal hızı

KISALTMA LİSTESİ

DVM	Destek Vektör Makineleri
EYM	Eğrilik Yarıçapı Merkezi
EMK	Elektromotor Kuvveti
GA	Genetik Algoritmalar
İHA	İnsansız Hava Araçları
LQR	Lineer Kuadratik Regülatör
YSA	Yapay Sinir Ağı

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 Yörünge oluşturma için kullanılan kara – kutu modeli....3
Şekil 2.1 Klasik yörünge planlama teknikleri....5
Şekil 2.2 Olasılıksal yörünge planlama teknikleri....5
Şekil 2.3 Sezgisel yörünge planlama teknikleri....5
Şekil 2.4 Hücresel bölümleme metodu....7
Şekil 2.5 Diferansiyel sürüş tekniği....11
Şekil 2.6 Senkron sürüş tekniği....12
Şekil 2.7 Ackermann sürüş tekniği....12
Şekil 2.8 Gezgin robot davranışlarının hiyerarşik dağılımı....12
Şekil 3.1 Tekerlekli gezgin robot....15
Şekil 3.2 İdealleştirilmiş tekerlek modeli....15
Şekil 3.3 Robotun koordinat düzlemindeki gösterimi....16
Şekil 3.4 DC motorun şematik gösterimi....19
Şekil 3.5 DC motor blok diyagramı....20
Şekil 3.6 DC motor simulink diyagramı....22
Şekil 3.7 Gezgin robot simulink diyagramı....23
Şekil 3.8 Alt sistem blok diyagramı....23
Şekil 4.1 *C* parametre uzayının hücrelere ayrılması....24
Şekil 4.2 Graf ve düğümler....25
Şekil 4.3 Düğüm komşulukları....25
Şekil 4.4 Arama algoritmaları....26
Şekil 4.5 Optimal yolun bulunması....27
Şekil 4.6 Bir düğümdeki maliyetin hesaplanması....29
Şekil 4.7 *h(.)* kabul edilebilir sezgisel fonksiyon....29
Şekil 4.8 Çalışma uzayı, hedefler ve engellerin tanımlanması....31
Şekil 4.9 Robotların başlangıç konumları, hedefler ve engellerin tanımlanması....32
Şekil 4.10 Çevre Matrisi....33
Şekil 4.11 2 robot – 2 hedef için optimal yörüngelerin bulunması....33
Şekil 4.12 3 robot – 2 hedef için optimal yörüngelerin bulunması....34
Şekil 4.13 4 robot – 2 hedef için optimal yörüngelerin bulunması....35
Şekil 4.14 Geri beslemeli bir sistemin blok diyagramı....36
Şekil 4.15 PID denetleyicinin blok diyagramı....36
Şekil 4.16 Sistemin birim basamak girişine verdiği cevap....37
Şekil 4.17 DC motorun kaskad PID kontrolü....38
Şekil 5.1 Sistemin blok diyagramı....39
Şekil 5.2 Gezgin robotun parametreleri....39
Şekil 5.3 Sistem algoritmasının akış diyagramı....40
Şekil 5.4 Durum 1 için optimal yörüngelerin hesaplanması....41
Şekil 5.5 Sistemin simulink blok diyagramı....42
Şekil 5.6 (a) Robot-1'in çıkış değerleri, (b) Robot-2'nin çıkış değerleri....43
Şekil 5.7 (a) Robot -1 için açı değişimi, (b) Robot-1'in konum bilgileri....44
Şekil 5.8 (a) Robot -1'in sol tekerlek hızı (b) Robot-1'in sağ tekerlek hızı....45
Şekil 5.9 (a) Robot -2 için açı değişimi, (b) Robot-2' nin konum bilgileri....46
Şekil 5.10 (a) Robot-2'nin sol tekerlek hızı (b) Robot-2'in sağ tekerlek hızı....47
Şekil 5.11 Durum 2 için optimal yörüngelerin hesaplanması....48
Şekil 5.12 (a) Robot-1'in çıkış değerleri, (b) Robot-2'nin çıkış değerleri....49

ÇİZELGE LİSTESİ

Sayfa

Çizelge 3.1 DC motor parametreleri....21
Çizelge 3.2 DC motorların dönüş yönüne bağlı olarak gezgin robotun hareketi....22
Çizelge 4.1 A* arama algoritmasının performansı....30
Çizelge 4.2 2 robot – 2 hedef için optimal yörünge uzunlukları....34
Çizelge 4.3 3 robot – 2 hedef için optimal yörünge uzunlukları....34
Çizelge 4.4 4 robot – 2 hedef için optimal yörünge uzunlukları....35
Çizelge 4.5 Oransal, integral ve türevsel denetleyicinin karakteristikleri....37

ÖNSÖZ

Robot teknolojilerinin gelişmesiyle birlikte insan gücünün yerini alan otonom robotlara olan ilgi gün geçtikçe artmaktadır. Özellikle otonom robotlardan oluşan sistemlerin bir amacı gerçeklemek üzere birlikte çalışması bu alanda öne çıkan konulardan biri haline gelmiştir. Bu sebeple tez çalışmasında otonom gezgin robotlardan oluşan bir sistemin hareket kontrolü problemi üzerinde durulmuştur. Problemin çözümünde "A* algoritması" ve doğrusal bir denetleyici birlikte kullanılmış ve amaca uygun sonuçlar elde edilmiştir.

Öncelikle tezimin her aşamasında bana destek olan ve ışık tutan çok değerli hocam, danışmanım Doç. Dr. Haluk GÖRGÜN'e, bu konuyu seçmemde beni cesaretlendiren sayın hocam Yrd. Doç. Dr. İbrahim B. KÜÇÜKDEMİRAL'a, tezin gelişmesinde katkılarını esirgemeyen Arş. Gör. Türker TÜRKER'e en derin teşekkürlerimi sunarım.

Ayrıca lisansüstü öğrenimim boyunca beni destekleyen TÜBİTAK'a teşekkürü bir borç bilirim.

Son olarak her an yanımda olan annem Füsun VAROL'a ve aileme sabırları, destekleri ve bana olan güvenlerinden dolayı teşekkür etmek isterim.

Yapılan bu tez çalışmasının otonom gezgin robotların kontrolü üzerine yapılacak çalışmalara kaynak olması dileğiyle.

ÖZET

Bu tezin amacı çoklu birimlerden oluşan bir sistemin kooperatif kontrolüne alt yapı sağlayacak bir algoritma oluşturmaktır. Günümüzde kooperatif sistemlerin kontrolü; kontrol alanında yapılan çalışmalarda öne çıkan konulardan biri haline gelmiştir. Kooperatif bir sistem; ortak bir amacı gerçekleştiren ve kendi aralarında görev dağılımı sağlayan dinamik birimlerden oluşur. Bu dinamik birimlere örnek olarak uçaklar, robotlar, deniz araçları verilebilir. Tek bir birim yerine birden çok birim kullanılması karmaşık görevlerin daha kısa sürede, daha dayanıklı ve verimli olarak yerine getirilmesinde büyük üstünlük sağlar.

Kooperatif bir sistemin kontrolü birimlerin dizilim düzeni kontrolü, hareket kontrolü ve birimler arası haberleşmenin sağlanması gibi alt başlıklara ayrılabilir. Bu çalışmada robotik alanında yaygın olarak kullanılan çoklu otonom gezgin robotların hareket kontrolü problemi ele alınmıştır. Otonom gezgin robotların hareket kontrolü ele alındığında yörünge planlaması gibi optimal kontrol problemleri, lineer olmayan sistemlerin kontrolü, pasiflik tabanlı kontrol gibi tekniklerle karşılaşılmaktadır.

Tez çalışması kapsamında çoklu gezgin robotlardan oluşan bir sistemin hareket kontrolü için iki aşamalı bir yapı oluşturulmuştur. Birinci aşamada optimal çözümü garantileyen "A* algoritması" geliştirilerek her bir robot için yörüngeler oluşturulmuş ve ikinci aşamada ise doğrusal bir denetleyici kullanılarak bu yörüngelerin takip edilmesi istenmiştir. Elde edilen sonuçlarda istenilen amaç ölçütlerine ulaşıldığı görülmüştür.

Anahtar kelimeler: kooperatif sistem, çoklu gezgin robot, hareket kontrolü, A* algoritması

ABSTRACT

The aim of this research is to create an algorithm which forms the basis of cooperative control of a system consisting multiple entities. Nowadays, the control of cooperative systems has become a highly active research area in control studies. A cooperative system is composed of multiple dynamic entities that share information or roles to accomplish a common purpose. The dynamic entity term is mostly used for aircrafts, robots and hovercraft vehicles. Complex tasks can be performed more efficiently, more robustly and much faster using multiple entities instead of employing one entity. These advantages of multiple entity systems make them more preferable.

The control of a cooperative system can be classified as follows: vehicle formation control, motion control and communication between entities. This research mainly deals with the problem of motion control of multiple autonomous mobile robots which are widely used in robotics area. Optimal control techniques such as trajectory generation, non linear control techniques and passivity based control are frequently used in the motion control of autonomous mobile robots.

In the context of the thesis, two phased approach is used for the motion control of multi mobile robot system. In the first stage "A* algorithm" that guarantees the optimal solution is developed and then the algorithm is used for creating optimal trajectories for each robot. In the second stage the objective is to use of a linear controller to follow the created trajectories in the first stage. Results are shown to agree with the desired criteria.

Keywords: cooperative system, multi vehicle mobile robot, motion control, A* algorithm

1. GİRİŞ

Günümüzde gerek endüstriyel gerekse askeri alanlarda yapılan çalışmalarda kooperatif olarak çalışan insansız sistemlerin kullanımı kaçınılmaz hale gelmiştir. Bir ortak amacı gerçekleştirmek üzere birlikte çalışan ve kendi aralarında görev dağılımı sağlayabilen çoklu robot sistemleri kooperatif sistemler olarak tanımlanır. Bu tür sistemlerin birçok uygulama alanı bulunmaktadır:

İnsansız Hava Araçları (İHA)

Havacılık alanında kaydedilen ilerlemeler ve uçuş kontrol tekniklerinin gelişmesi ile insansız hava araçları teknolojisi (İHA) önemli noktalara gelmiştir. Ryan ve arkadaşlarının (2004) yaptığı çalışmada insansız hava araçlarının kooperatif kontrolünde öne çıkan güncel konu başlıklarından bahsedilmiştir. Bu konulardan bazıları şunlardır: insansız hava araçlarının güvenliği (çarpışmadan ve yeri belirlenmemiş engellerden kaçınma), araçların belirli bir dizilimde uçması ve yörünge takibi.

İnsansız Denizaltı Araçları

Son zamanlarda geliştirilen otonom denizaltı araçları laboratuvardan çıkıp ticari ve askeri amaçlarla kullanılmaya başlanmıştır. Özellikle oşinografik (okyanus zemini) örnekleme ve mayın tarama gibi zorlu görevler gerçekleştirilmektedir.

Akıllı Otoyol (Ulaşım) Sistemleri

Tam otomasyonlu otoyol sistemleri konusu 90'lı yılların başından beri çalışılmakta olup başarılı simulasyon (benzetim) çalışmaları bulunmaktadır. Simulasyonların yanısıra test edilen prototipler bulunmaktadır. Hedrick vd.(1994) tarafından yapılan çalışmada otomasyonlu otoyol sisteminin kontrolü ele alınmıştır. Otomasyonlu (akıllı) ulaşım sistemlerinin hedefi temel olarak daha güvenli bir sürüş ortamının sağlanması,trafiğin azaltılması, enerji tüketiminin ve hava kirliliğinin azaltılması olarak özetlenebilir.

Robot Futbolu

Gezgin robotların gerçeklenmesi ve kontrolü büyük çaplı sistemlere (insansız uçak filosu, insansız deniz araçları..) göre daha kolay olduğundan kooperatif olarak çalışan futbol oynayan robotlar oldukça popüler hale gelmiştir. D'Andrea'nin (2000) yaptığı çalışmada birinci dereceden lineer olmayan, kendi ekseni etrafında dönebilen tekerlekli bir otonom araç modeli geliştirilmiştir. Daha sonra sistemin test edilebilmesi için birden fazla özdeş araç oluşturulmuş ve araçların verilen bir görevi grup halinde gerçekleştirmesi (futbol oynaması) istenmiş ve karşılaşılan sorunlara göre kontrol algoritmaları geliştirilmiştir.

Örnekleri verilen alanlarda kooperatif çalışan ve çoklu robotlardan oluşan sistemlerin kullanılması karmaşık görevlerin daha kısa sürede, daha dayanıklı ve verimli olarak yerine getirilmesinde büyük üstünlük sağlar.

Teorik anlamda birçok araştırmanın sonucunda elde edilen kontrol algoritmaları otonom gezgin robot modelleri üzerinde denenmektedir. Bunun en önemli sebebleri otonom gezgin robotların gerçeklenmesinin kolaylığı ve robotların 2.dereceden, lineer olmayan, eksik eyleyicili ve giriş sınırlı dinamikleri göz önüne alınarak geliştirilen kontrol tekniklerinin kolaylıkla diğer sistemlere uyarlanabilmesidir.

Otonom gezgin robotların kontrolü ele alındığında yörünge planlanması, dizilim düzeni kontrolü gibi optimal kontrol problemleri, lineer olmayan sistemlerin kontrolü, pasiflik tabanlı kontrol, kayan ufuklu kontrol gibi tekniklerle sıklıkla karşılaşılmaktadır.

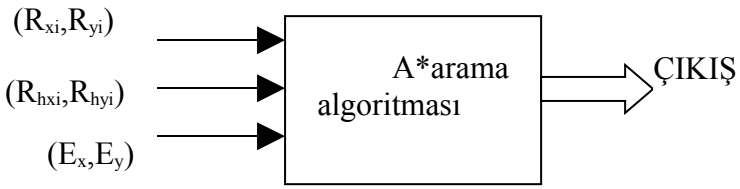
1.1 Problemin Tanımı

Bu tez çalışması kapsamında *2* serbestlik dereceli *n* tane otonom gezgin robottan oluşan bir sistemin kooperatif kontrolünü sağlayan bir algoritma gerçekleştirilmiştir. Böyle bir sistemin kooperatif kontrolü birçok alt başlığı içermektedir:

- Hareket kontrolü (Yörünge planlama/takip)
- Dizilim düzeni kontrolü
- Sistemi oluşturan birimler arası haberleşme
- Sistemin optimizasyonu
- Sistemin kararlığı

Bu çalışmada kooperatif kontrol probleminin alt başlıkları olan yörünge planlanması ve hareket kontrolü problemi ele alınmıştır. *N* tane otonom gezgin robotun matematiksel olarak modellenmesi, A* arama algoritması (Hart vd. 1968) yardımıyla yörüngesinin oluşturulması ve hareketinin kontrolü (yörünge takibi) gerçekleştirilmiştir.

Şekil 1.1’ de verilen yörünge oluşturma problemi kara-kutu modeliyle de tanımlanmış olup;



Şekil 1.1 Yörünge oluşturma için kullanılan kara – kutu modeli.

Kara-kutu modelinin giriş ve çıkış büyüklükleri sırasıyla:

R_{xi},R_{yi} : i.robotun kartezyen düzlemdeki başlangıç koordinatları,

R_{hxi},R_{hyi} : i.robotun kartezyen düzlemdeki hedef koordinatları,

E_x,E_y : Düzlemdeki engellerin koordinatları,

Çıkış: Başlangıç ve hedef koordinatları arasında optimal bir yörünge oluşturulmasıdır.

Daha sonra kara kutu modeliyle elde edilen optimal yörünge PID kontrolör ile takip edilmiştir.

1.2 Tezin Genel Yapısı

Bölüm 1' de giriş, tez probleminin tanımlanması ve tezin içeriğini gösterilmektedir.

Bölüm 2' de otonom gezgin robotların sürüş tekniklerinden bahsedilmiştir. Ayrıca hareket kontrolü ve yörünge takibi için kullanılan yaklaşımlar hakkında bir literatür taraması yapılmıştır. Sonraki kısımda ise tezin diğer bölümlerinde kullanılacak olan notasyon ve terminoloji belirtilmiştir.

Bölüm 3' te otonom gezgin robotun modeli ve robotta kullanılan motorların matematiksel modelleri çıkarılmıştır. Matematiksel modeller benzetim çalışmaları için Matlab®Simulink programına aktarılmıştır.

Bölüm 4'te otonom gezgin robotun yörüngesinin planlanması için kullanılan A* arama algoritması ve yörünge takibi için kullanılan PID kontrol algoritması anlatılmıştır. Bu bölümde ayrı ayrı tasarlanan iki algoritma birleştirilmiştir. Böylece A* arama algoritması ile oluşturulan optimal yörünge, PID kontrolör yardımıyla gezgin robotlar tarafından izlenmiştir.

Bölüm 5'te benzetim (simulasyon) çalışmalarının sonuçları gösterilmiştir.

Bölüm 6'da sonuçlar yorumlanmış ve gelecekte yapılabilecek yeni çalışmalar için öneriler verilmiştir.

2. LİTERATÜR TARAMASI ve ALT YAPI

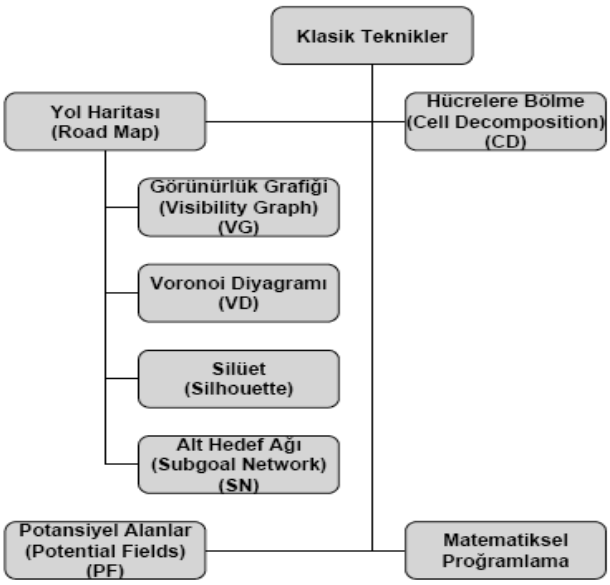
Bu bölümde tez çalışması kapsamında kullanılan hareket kontrolü algoritmaları ve otonom gezgin robotlar hakkında kapsamlı bir literatür taraması yapılmıştır.

2.1 Hareket Kontrolü için Kullanılan Yaklaşımlar

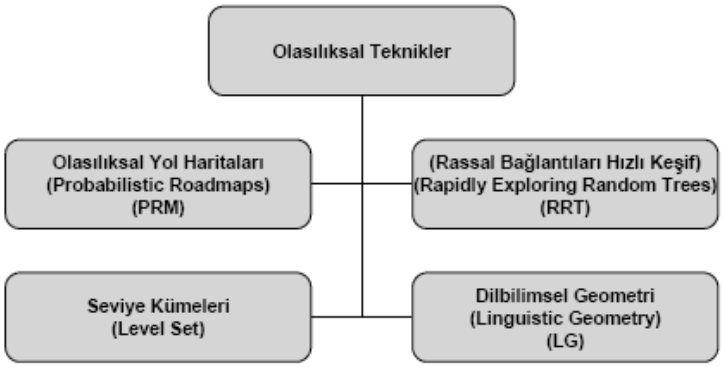
Son yıllarda, çoklu robot sistemlerinin önemi oldukça artmıştır. Bu sistemlerin yaygınlaşmasıyla birlikte çoklu robot sistemlerinin hareketlerinin planlanması üzerine yapılan çalışmalar artmıştır. Çoklu gezgin robotların hareketlerinin kontrolü, dizilim düzeni kontrolü ve yörünge planlaması olarak iki başlık altında toplanabilir. Dizilim düzeni kontrolünde üç temel yaklaşımdan bahsedilir: Lider-takipçi yaklaşımı, davranışsal yaklaşım ve sanal yapılar yaklaşımı. Bunlardan ilki olan lider-takipçi yaklaşımında robotların belirlenen bir dizilim yapısında hareket edebilmesi için gruptan bir robot lider olarak belirlenir ve grubunda bulunan diğer robotlar (takipçiler) için referans bir yörünge belirler. Diğer robotlar referans yörüngeyi takip ederek istenilen hedefe ulaşırlar. Lider robot takım davranışını belirlemiş olur. Davranışsal yaklaşımda ise robotların belirlenen görevi başarması istenirken yapması/kaçınması gereken davranışlar tanımlanır. Örneğin robotların engelden/çarpışmadan kaçınmasından sonra dizilimini tekrar oluşturması gibi. Sanal yapılar yaklaşımında ise çoklu robotlardan oluşan sistem tek bir birim olarak modellenir. Oluşturulan bu sanal yapının istenilen dinamik modeli tanımlandıktan sonra her bir robot için istenilen hareket elde edilmiş olur.

Çoklu robotlar için yörünge (hareket) planlama problemi ise robotların verilen başlangıç noktasından hedefe ulaşırken çarpışmadan kaçınarak ilerlemesini sağlayacak yörüngelerinin belirlenmesidir. Yörünge planlama süreci de kendi içinde katmanlar içerir. Bu katmanlar temel olarak evrensel (genel) ve yerel (lokal) planlama olarak tanımlanabilir. Evrensel (genel) yörünge planlamada çalışma uzayı tamamen bilinmeli ve statik olmalıdır, yani çevresel bilgiler önceden sağlanmalıdır. Bu yaklaşıma göre, robot hareketine başlamadan önce verilen başlangıç noktasından hedefe kadar olan tüm yörünge planlanır. Diğer yaklaşımda ise algılayıcılardan gelen anlık bilgilere bağlı olarak belirli bir alan için yörünge planlanır, yani planlama robot hareket halindeyken yapılır.

Yörünge planlanmasında kullanılan birçok teknik bulunmaktadır. Bu teknikler temel olarak klasik, sezgisel ve olasılığa dayalı teknikler olarak sıralanabilir. Şekil 2.1, 2.2 ve 2.3'te klasik, sezgisel ve olasılığa dayalı teknikler sınıflandırılmıştır.



Şekil 2.1 Klasik yörünge planlama teknikleri (Aksoy ve Kurnaz, 2009).



Şekil 2.2 Olasılıksal yörünge planlama teknikleri (Aksoy ve Kurnaz, 2009).

Şekil 2.3 Sezgisel yörünge planlama teknikleri (Aksoy ve Kurnaz, 2009).

Tez kapsamında, otonom gezgin robotların yörünge planlaması için klasik metodlardan olan hücrelere bölme metodu ve arama algoritmalarından A* arama algoritması kullanılacak olup aşağıda diğer tekniklerin bazılarından kısaca bahsedilmiştir.

2.1.1 Klasik Teknikler

Gerçek zamanlı cevap gerektiren sistemlerde hareket planlama problemi için klasik tekniklerin kullanılması oldukça yaygındır. Bu teknikler tek başına veya birbirlerinin kombinasyonu şeklinde kullanılmaktadır. Potansiyel alanlar metodu ve hücrelere bölme metodu klasik tekniklerin en bilinen metodlarındandır. Hwang ve Ahuja'nın (1992) çalışmasında klasik tekniklerle ilgili kapsamlı bir inceleme bulunabilir.

2.1.1.1 Potansiyel Alanlar Metodu

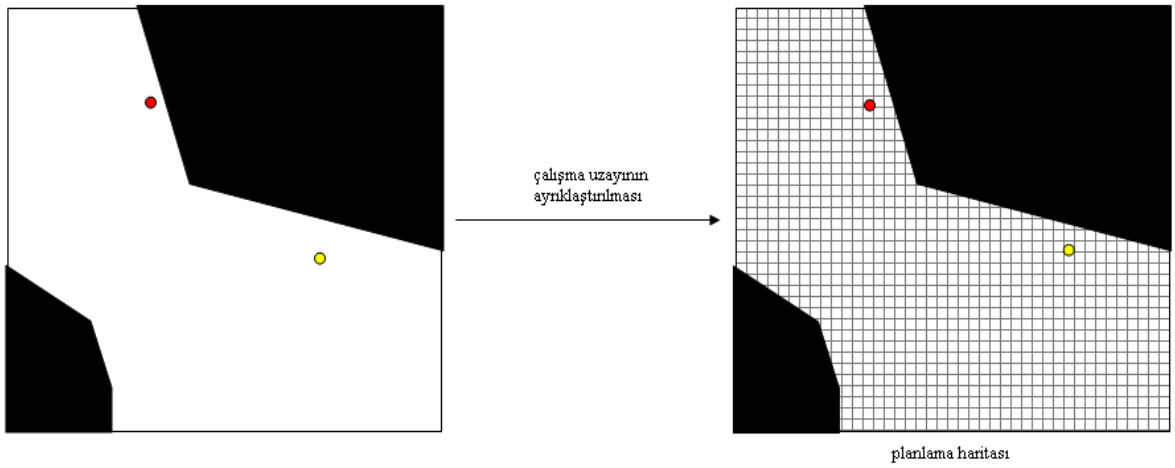
Yapay potansiyel alan metodunu ilk uygulayanlardan biri Khatib'dir (1986). Bu metodun temel prensibine göre çalışma alanı yapay potansiyel kuvvetler etkisindedir. Çalışma alanında bulunan engeller itme etkisi oluşturarak robotu engellerden uzaklaştırır. Robotun gitmesi gereken hedef noktası ise çekim etkisi oluşturarak robotu hedefe doğru çeker. Çalışma alanındaki herhangi bir noktada itim ve çekim kuvvetleri toplamının (net kuvvet) bulunmasıyla robot kontrolü sağlanır. Buradaki en önemli sorun potansiyel alanlar metodunun yerel minimumlara sahip olmasıdır. Yani robot hedefe ulaşmadan, hedefe ulaşmış gibi yerel minimum olan bir noktaya takılıp kalabilir. Graham ve Buckingham (1993) bu soruna yerel minimumu olmayan potansiyel alanlar oluşturarak çözüm getirmeye çalışmışlardır.

Reif ve Wang (1999) tarafından yapılan çalışmada ise yapay potansiyel alanlar metodu çoklu gezgin robotların hareket kontrolü için kullanılmıştır. Bu çalışma çoklu gezgin robot sistemlerinin kontrolünde potansiyel alanlar yönteminin kullanıldığı ilk çalışmalardan biridir. Çalışmada bir robotun hareketi diğer robotların konumlarına bağlı olarak belirlendiği için yazarlar bu yöntemi sosyal potansiyel alan metodu olarak adlandırmışlardır. Çoklu robot sistemlerinin potansiyel fonksiyonlarla hareketlerinin planlanmasında hala geliştilmesi gereken noktalar bulunmaktadır. Buna örnek olarak çoklu robotların birbirleriyle çarpışması veya karmaşık ortamda hareketlerinin modellenmesi verilebilir. (Fang ve Wei, 2007)

Literatüde potansiyel alanlar metodunun kullanıldığı birçok uygulama bulunmaktadır. Zhang vd. (2004) robot futbolunda yörünge planlamada, Baxter vd. (2007) çoklu robotlarla arama-kurtarma görevlerini gerçekleştirmede ve Khatib (1986) ise manipülatör ve mobil robot sistemlerinde engelden kaçınmada kullanmıştır.

2.1.1.2 Hücrelere Bölme Metodu

Hücrelere bölme metodu ilk kez Keil ve Sack (1985) tarafından kullanılmıştır. Bu yaklaşımın temel teorisi çalışma uzayının basit hücresel kümelere ayrılarak graf oluşturulması ve graf üzerinde bir yörüngenin bulunmasıdır.



Şekil 2.4 Hücrelere bölme metodu.

Öncelikle çalışma uzayı her biri aynı biçime sahip ve birbirine bağlı olan basit hücresel yapılara bölünür, yani ayrıklaştırılır. 2. adım olarak hangi hücrelerin serbest çalışma uzayında (engel içermeyen) bulunduğu belirlenir. Daha sonra bu hücreler için komşuluklar belirlenir. Örneğin robotun sadece aşağı, yukarı, sağa ve sola hareket ettiği düşünülürse 4 komşuluklu bir graf göz önüne alınarak bağlantılılık matrisi hesaplanacaktır. Bulunan bağlantılılık matrisine bağlı olarak başlangıç ve hedef noktaları arasında çarpışmadan kaçınan bir yörünge oluşturulur.

Katevas vd.'nin (1998) yaptığı çalışmada iki boyutlu uzayda çalışan gezgin robotlar için hücrelere bölme metodu temel alınarak yeni bir algoritma geliştirilmiştir. Algoritma gerçek zamanlı robot hareket problemine uygulanmıştır. Yapılan bir diğer çalışmada, çoklu robot sistemleri için kullanılan yörünge planlama teknikleri ile hücrelere bölme metodu hakkında yapılan çalışmaların sonuçları karşılaştırılmıştır. (Choset, 2001)

Hücrelere bölme metodu uygulama kolaylığı, tamamlanmış bir çözüm ve adaptif bir yapı (çözünürlüğe bağlı olarak) göstermesi sebebiyle robot hareketini planlama problemlerinde sıklıkla tercih edilmektedir. Fakat serbestlik derecesi 4 ve daha fazla olan sistemlerde büyük bellek gerektirdiğinden ve yavaş çalıştığından kullanımı uygun değildir. Ayrıca oluşturulan tüm çözümler (yörüngeler) optimallik (en iyi) göstermeyebilir. Bu sebeple tez çalışmasında hücrelere bölme metodu ve A* arama algoritması birlikte kullanılmıştır. A* arama algoritmasından bir sonraki bölümde bahsedilecektir.

2.1.2 Olasılıksal Tabanlı Yaklaşımlar

Son yıllarda olasılıksal (örnekleme) tabanlı yaklaşımlar kullanarak hareket planlama problemlerinin çözülmesi üzerinde çalışan birçok araştırmacı bulunmaktadır. Özellikle yüksek serbestlik dereceli robotların hareketlerinin planlanmasında diğer yöntemlere göre oldukça hızlıdır. Bu yaklaşımlardan en çok bilinen ve kullanılanları yol haritası-tabanlı ve ağaç tabanlı yaklaşımlardır.

2.1.2.1 Yol Haritası-Tabanlı Planlayıcılar

Yol haritası tabanlı planlayıcılar genellikle çoklu sorgusal yaklaşımlardır. Bu yaklaşımda, öğrenme aşaması ve sorgulama aşaması olmak üzere iki aşama bulunur. Öğrenme aşamasında çalışma uzayından engellerle kesişmeyecek şekilde örnekleme noktaları alınır. Alınan bu noktalar kendi aralarında yol haritaları oluştururlar. Sorgulama aşamasında ise oluşturulan yol haritası grupları birbirlerine en yakın noktalardan bağlanarak verilen başlangıç noktasından hedef noktasına nasıl gidilebileceği sorgulanır. Yol haritası tabanlı planlayıcıların en sık karşılaşılan yöntemi olasılıksal yol haritasıdır.

Olasılıksal yol haritası temelli planlamada örnekleme başarısına bağlı olarak çözümün başarısı artar. Fakat dar geçitli çevrelerde örnekleme az olduğundan bu planlayıcıların başarımı düşüş gösterir. Başarımı arttırmak için filtrelenmiş örnekleme, çalışma uzayı bilgisine göre örnekleme ve kademeli planlayıcı gibi yöntemler kullanılabilir.

Pettersson ve Doherty'nin (2006) yaptıkları çalışmada insansız bir hava aracı için çarpışmadan kaçınan ve verilen görevlerin otonom olarak gerçekleştirilmesini sağlayan bir yörünge planlayıcısı oluşturulmuştur. Yörünge planlama için olasılıksal yol haritası tabanlı planlayıcı ve ağaç tabanlı planlayıcı kullanılmıştır. Her iki planlayıcı da deneysel olarak gerçeklenmiş ve sonuçlar karşılaştırılmıştır.

2.1.2.2 Ağaç-Tabanlı Planlayıcılar

Ağaç tabanlı planlayıcılar ilk olarak LaValle (1998) tarafından geliştirilmiş ve hareket planlaması için kullanılmıştır. Tekli sorgusal yaklaşımlardan olan ağaç tabanlı planlayıcılarda örnekleme artımsal olarak var olan ağaç üzerine eklenerek yapılır. Çözüm döngüleri arttıkça çözüm ağacı genişler. Hızlı tarayıcı rastlantısal ağaç yaklaşımında uzay içerisinde rastlantısal olarak seçilen bir noktaya en yakın üye nokta, ilerlenecek nokta olarak belirlenir. Hızlı yayılma özelliğine bağlı olarak arama verilen başlangıç ve hedef noktalardan başlatılabilir. Aramanın artmasına göre çözüm zamanının azaldığı söylenebilir.

Franchi vd. (2007) gezgin robotlar için kooperatif bir araştırma stratejisi geliştirmişlerdir. Sensör temelli rastlantısal ağaçlar incelenen bölgeyle güvenli bölgeyi yol haritaları ile birbirine ilişkilendirmiştir. Bir başka çalışmada hızlı tarayıcı rastlantısal ağaç yöntemine farklı iki özellik eklenerek 'genişletilmiş hızlı tarayıcı rastlantısal ağaç metodu' olarak isimlendirilmiştir. Daha sonra geliştirilen bu yeni metod kooperatif gezgin robotların popüler bir uygulama alanı olan robot futbolu üzerinde denenmiştir. Gerçek uygulamada robot diğer yaklaşımlara göre daha yüksek bir performans göstermiştir. Örneğin engellerden kaçarken %40 daha hızlı hareket etmiş, osilasyon ve yerel minimum problemi azalmıştır.(Bruce ve Veloso, 2002)

2.1.3 Sezgisel Teknikler

Dinamik ortamlarda gerçek zamanlı çalışan çoklu robotlarda hareket planlama probleminin diğer tekniklerle çözülmesi her zaman mümkün olmayabilir. Çalışma ortamındaki engellerin ve ortam yapısının değişmesi önceden belirlenen yörüngenin değişmesine sebep olur. Bu durumda hızlı çözümler üretmek için sezgisel teknikler kullanılır. Bu algoritmalar her zaman en iyi çözümü vermeyebilir. Fakat sezgisel tekniklerin eniyileme yöntemleri ile birleştirilmesi sonucu oldukça iyi sonuçlar elde edilebilir.

2.1.3.1 Genetik Algoritmalar

Robotların hareket planlama probleminde genetik algoritmanın kullanılması fikri ilk olarak Davidor (1990) ve Parker vd.'nin (1989) çalışmalarında yer almıştır. GA'lar rastlantısal arama teknikleriyle iteratif olarak çözüm bulmaya çalışan bir makine öğrenmesi modelidir. Genetik algoritmalarla gerçekleştirilen evrim sürecinde genellikle üç aşama kullanılır. Bu aşamalar sırasıyla seçim, çaprazlama ve mutasyondur. Bu işlemlerin amacı, daha iyi özelliğe sahip yeni nesiller üretmek ve arama algoritmasının alanını genişletmektir. GA'lar çözümün nerede sonuçlanacağını bilmezler, yakınsamanın sağlandığı an çözüm olarak kabul edilir ve bu yüzden bulunan değer en iyi çözüm değil yerel optimum bir çözümdür.(Aksoy ve Kurnaz, 2009)

Gezgin robotlarda genetik algoritma yardımıyla yörünge planlama probleminin çözülmesi yörüngenin uygun bir 'kromozom' tarafından ifade edilmesi, yörünge yönlendirme mekanizmasının belirlenmesi, yörüngenin uzunluğunu minimize edecek bir kısıtın belirlenmesi gibi koşullara bağlıdır. Bu yaklaşımda çalışma ortamının statik ve önceden bilindiği kabul edilmektedir.

Li ve Jia'nın (2008) yaptığı çalışmada çoklu robotların yörüngelerinin planlanmasında

kooperatif temelli kayan ufuklu kontrol stratejisiyle genetik algoritmalar birleştirilerek belirlenen amaç fonksiyonunun ençoklanması için gereken optimal kontrol kanunu çözülmüştür.

2.1.3.2 Bulanık Mantık

Bulanık mantık teorisi ilk olarak Zadeh (1965) tarafından ortaya atılmıştır. Bulanık mantık yaklaşımına göre bir sistemin kontrolü için, uzman kişilerden dilsel ifadeler olarak alınan bilgilerin bulanık mantık kurallarıyla ifade edilir. Bu yaklaşımın en önemli avantajı matematiksel modeli tam olarak bilinmeyen sistemlere uygulanabilmesidir. Böylelikle karmaşık sistemlerin ifade edilmesi kolaylaşır.

Aoki vd.'nin (1994) yaptıkları çalışmada otonom gezgin robotların çoklu engellerden kaçması için hiyerarşik yapıda bulanık kurallar oluşturulmuştur. Literatürde bulanık mantık yaklaşımı ile yapay sinir ağları ve genetik algoritmaları birleştiren birçok çalışma bulunmaktadır. Son yıllarda yapılan bir başka çalışmada ise çok katmanlı yapıya sahip kooperatif robot sistemlerinin kontrolünde bulanık mantık kullanılmıştır. (Innocenti vd., 2007)

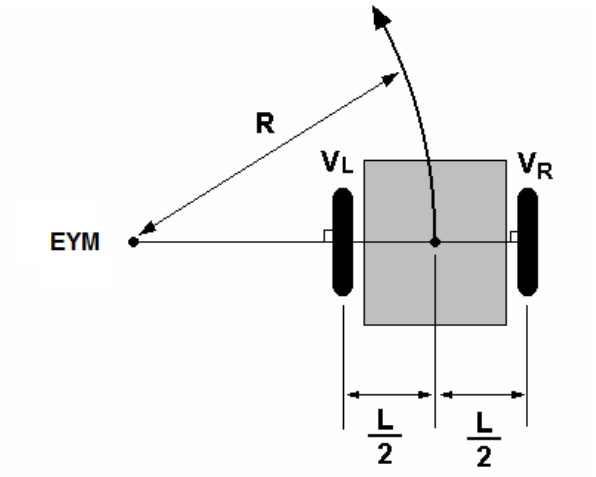
2.1.3.3 Yapay Sinir Ağları

Yapay sinir ağları, yapay sinir hücrelerinin birbirleri ile çeşitli şekillerde bağlanmasından oluşur ve genellikle katmanlar şeklinde düzenlenir. Yapay sinir ağları paralel bir yapıya sahiptir ve hatayı tolere etme kapasiteleri klasik tekniklere göre oldukça yüksektir. Ayrıca kendi kendini örgütleme ve adaptif yapısı nedeniyle de klasik tekniklerinden ayrılmaktadır. Eğiticili bir öğrenme algoritması olan yapay sinir ağları (YSA) yaklaşımı robot hareket planlama probleminde dinamik değişen ortamlarda gerçek zamanlı çarpışmadan kaçınan bir yörünge oluşturulması için kullanılmıştır (Zelinsky, 2004). Bu çalışmada geliştirilen model noktasal robotlara, manipülatör robotlara ve çoklu robot sistemleri olmak üzere çeşitli robot sistemlerine uygulanabilir. Parhia vd.'nin (2009) yaptıkları çalışmada kooperatif davranış gösteren çoklu robot sistemleri için akıllı yörünge planlanması ele alınmıştır. Çoklu robotların bilinmeyen veya kısmen bilinen çalışma ortamlarında geziniminin sağlanması için nöro-fuzzy temelli yaklaşım ile kural temelli yaklaşım analiz edilmiştir. Daha sonra robotların önceden belirlenen hedeflere ulaşmasını sağlamak amacıyla nöro-fuzzy ve kural temelli yaklaşımı birleştiren hibrid bir algoritma oluşturulmuştur. Bildirilen sonuca göre hibrid bir yapı oluşturulması karmaşık ve bilinmeyen çevrelerde robotların gezinim performansını arttırmıştır.

2.2 Otonom Gezgin Robotlar

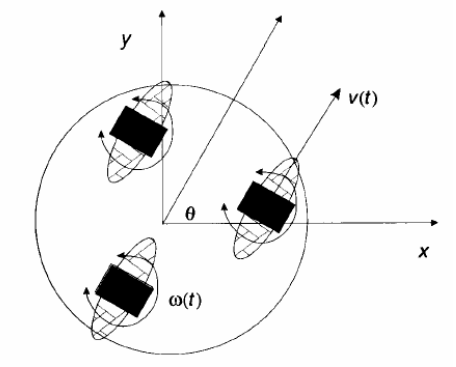
Bu bölümde tez çalışmasında kullanılan otonom gezgin robotlar ve uygulamaları hakkında genel bir bilgi verilmiştir. Kooperatif davranış içindeki otonom gezgin robotların modellenmesi aktif bir çalışma alanıdır (Hur vd., 2003; Borkowski vd., 2001). Gezgin robot birçok şekilde modellenebilir. Fakat modellemedeki en önemli faktör robotun kullanım alanı ve modelinin karmaşıklığıdır. Morin ve Samson (2004) tarafından yapılan çalışmada gezgin robotların modellenmesinde kullanılan farklı yaklaşımlar incelenerek birbirleriyle karşılaştırılmıştır. Bu tez çalışmasında kapalı alanlarda kullanılan tekerlekli gezgin bir robot modeli ele alınmıştır. Tekerlekli gezgin robotların modellenmesi, denetimi ve oluşturulması diğer robot tiplerine (çok ayaklı robotlar, denizaltı robotu vb.) göre daha kolaydır.

Tekerlekli gezgin robotlar sürüş tekniklerine 4 sınıfa ayrılabilir. (Dudek ve Jenkin, 2000) Bunlardan ilki diferansiyel sürüş tekniğidir. Genellikle robotik alanında yapılan çalışmalarda modelleme kolaylığı açısından "diferansiyel sürüş" sistemli robotlar kullanılmaktadır. Bu robotlar en az 3 tekerleğe sahiptir. Ortak eksende bulunan 2 sabit tekerlek sürüşü sağlarken, 1 ya da daha fazla bulunan serbest tekerlek ise robotun dengesini korumasını sağlar. Bu geometrik yapıya bağlı olarak robot ani yön değişimlerini kolaylıkla gerçekleştirebilir. Şekil 2.5'de gösterilen diferansiyel sürüş tekniğine göre robot iki tekerlek arasındaki hız farkına göre yönelimini gerçekleştirir. Burada EYM eğrilik yarıçapı merkezini göstermektedir.



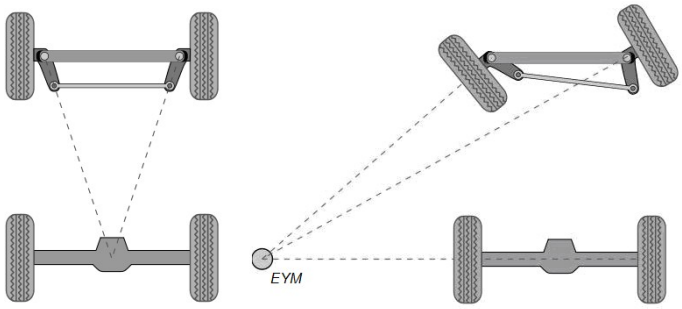
Şekil 2.5 Diferansiyel sürüş tekniği.

Şekil 2.6'da gösterilen senkron sürüş tekniğinde ise her bir tekerlek sürülebilir ve kontrol edilebilir. 3 tekerlekten oluşan bu modelde tekerleklerin dönüş yönü ve miktarı aynıdır. Bir başka teknik olan bisiklet sürüş tekniğinde kuvvet ve kontrol ön tekerlekten sağlanır.



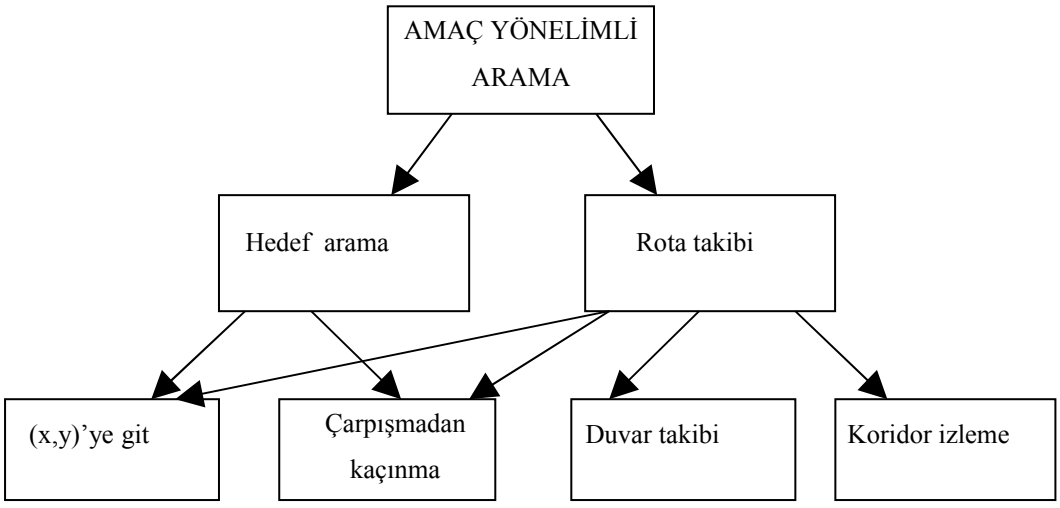
Şekil 2.6 Senkron sürüş tekniği.

Otomobillerde kullanılan ve diferansiyel sürüş tekniğinden sonra en bilinen teknik Ackermann sürüş tekniğidir. Ackermann sürüş tekniğinde ön tekerleklerin yönelim açısını kontrol eden bir direksiyon sistemi ve aracın hareketini sağlayan bir diferansiyel sistemi bulunur. Bu sistemde yol hareketi diğer sistemlere göre oldukça kararlıdır.



Şekil 2.7 Ackermann sürüş tekniği.

Gezgin robotların uygulama alanlarına göre duvar/çizgi takibi, koridor takibi, çarpışmadan kaçınma, hedef/amaç arama gibi çeşitli davranış biçimleri vardır.



Şekil 2.8 Gezgin robot davranışlarının hiyerarşik dağılımı (Jamshidi vd., 1997).

Gezgin robotlar genellikle basit kontrol yöntemleri ile denetlenmektedir. Bu robotların kontrol stratejileri açık çevrim ve kapalı çevrim kontrol olarak iki ana başlık altında toplanabilir. Açık çevrim kontrolde başlangıç ve hedef koordinatlar arasında bir yörünge oluşturulur ve buna bağlı olarak sisteme tork veya hız cinsinden girdi bilgileri gönderilir. Bu tip bir kontrolde model hataları ve bozucular dikkate alınmadığından çoğunlukla istenilen sonuç elde edilemez. Kapalı çevrim kontrolde ise girdiler sistemin gerçek (anlık) durumlarına göre belirlendiğinden bozucular ve hatalar kompanze edilir. Kapalı çevrim denetleyicilere örnek olarak P (oransal), PI (oran-integral), PID (oran-integral-türev), tam durum geri beslemesi ve bulanık mantık denetleyici verilebilir. Literatürde gezgin robot sistemlerinin kontrolü ve kararlılığı ile ilgili birçok kaynak bulunmaktadır. (Canudas de Wit vd., 1996; Craig, 2004)

Tezde kooperatif çalışan gezgin robotların hareket problemi ve kontrolü ele alınmıştır. Li ve D'Andrea'nin (2008) futbol oynayan robotların kontrolü üzerine yaptıkları çalışmada otonom robotların yörünge planlaması (hareket kontrolü) için iki katmanlı bir yapı kullanılmıştır. Alt katmanda, istenilen görevi gerçekleştiren yörüngeler optimal kontrol kanunları kabul edilerek belirlenmiştir. Üst katmanda ise istenilen görev özel bir maliyet fonksiyonuna dönüştürülerek robot için son yörünge oluşturulmuştur. Lineer olmayan kontrol teorisi de gezgin robotların kontrolünde önemli bir yer oluşturmaktadır. Aguiar vd. (2003)'nin yaptığı çalışmada eksik eyleyicili hoverkraft robot modeli için lineer olmayan Lyapunov-tabanlı bir kontrolör geliştirilmiş ve sistemin üstel kararlılığı sağlanmıştır. Lineer olmayan bu kontrolör Chung vd. (2002)'nin geliştirdiği Caltech Çoklu Araç Kablosuz Test Yatağında deneysel olarak gerçeklenmiştir. Davies ve Jnifene (2007)'nin yaptığı bir başka çalışmada ortak çalışan gezgin robotlar için hibrid bir kontrol yapısı geliştirilmiştir. Genetik algoritma kullanılarak her bir robot için optimize edilmiş yörüngeler bulunmuştur. Ayrıca robotların engellerden ve birbirleriyle çarpışmadan kaçınması için genetik algoritma ve yapay potansiyel alanlar yaklaşımı birleştirilip kinematik tabanlı bir kontrolör ile sistem kontrolü sağlanmıştır. Daha sonra geliştirilen hibrid kontrol mimarisi deneysel olarak gerçeklenmiştir.

Yapılan literatür taramasına göre bu konuda yapılan çalışmalar oldukça günceldir ve dünyada birçok grubun araştırmaları bu yöndedir. Bu sebepler, kooperatif kontrol teorisi için en uygun uygulama alanının otonom gezgin robotlar olduğunu göstermiştir.

2.3 Notasyon ve Terminoloji

Bu bölümde hareket kontrolü (planlanma) problemlerinde genel olarak kullanılan terminolojiden kısaca bahsedilmiştir. Hareket kontrolü probleminde amaç, başlangıç (B) ve

hedef (*H*) koordinatlarını ortamda bulunan engellerle (*E*) çarpışmadan kaçınarak birbirine bağlayan sürekli bir hareket sağlamaktır. Genellikle robotlar ve engeller 2 veya 3 boyutlu bir çalışma uzayında tanımlanırken, hareket daha yüksek boyutlu bir parametre uzayında gösterilir.

Parametre uzayı (*C*) Tez kapsamında robotların $W=R^2$ uzayında hareket ettiği kabul edilmiştir. Her bir robotun parametreleri belirli referans noktalara göre konumunu ve yönelimini içerir. Yani her bir robot (x,y,θ) parametreleri ile tanımlanır. *C* parametre uzayı ise olanaklı tüm parametrelerin olduğu kümedir. Örneğin *n* eklemli (her bir eklemin kısıtlı bir hareket bölgesi varken) bir robot kol için parametre uzayı R^n'dir.

Engel parametre uzayı (C_{eng})

Engel bölgesi (*O*), *W* uzayında bulunan bir veya daha fazla engele ait olan noktalar kümesinden oluşur. Robotların tüm parametrelerini içeren küme (*C*) ile engel bölgesinin (*O*) kesişimi engel parametre uzayıdır ve C_{eng} ile gösterilir. Çoklu robot sistemlerinde engel çarpışma denetimine ek olarak robotların birbirleriyle çarpışma denetimi de gerçekleştirilmelidir. Bu bölgelerin dışında kalan bölge ise serbest bölge olarak tanımlanır (C_{ser}).

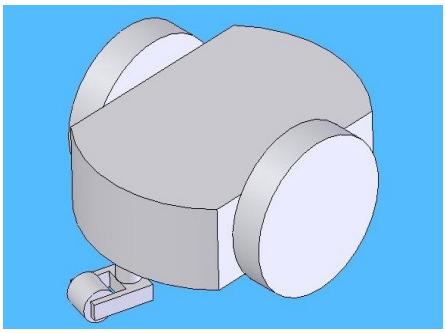
Bu bölümde verilen terminoloji hakkında daha kapsamlı bir bilgi için LaValle (2006)'in kitabına başvurulabilir.

3. OTONOM GEZGİN ROBOTUN MODELLENMESİ

Bu bölümde otonom gezgin robot ve bileşenleri tanıtıldıktan sonra matematiksel modelleri verilmiştir. Son kısımda ise benzetim çalışmaları için gerekli olan Simulink blok diyagram modelleri oluşturulmuştur.

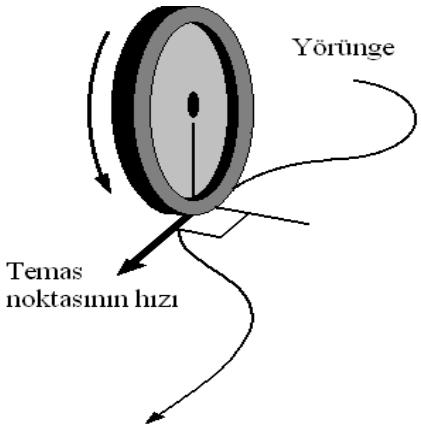
3.1 Sistemin Tanıtılması

Genellikle robotik alanında yapılan çalışmalarda modelleme kolaylığı açısından "diferansiyel sürüş" sistemli robotlar kullanılmaktadır. Tez çalışmasında kullanılan robot karesel bir gövdeye ve birbirine paralel olan özdeş iki tekerleğe sahiptir (Şekil 3.1). Bu tekerlekler bağımsız DC motorlar tarafından kontrol edilmektedir. DC motorun matematiksel modelinden sonraki bölümlerde bahsedilecektir.



Şekil 3.1 Tekerlekli gezgin robot.

Robotun matematiksel modeli oluşturulurken bazı kabuller yapılmıştır. Düşük hızlar için dönme hareketi baskın olduğundan tekerleklerin sadece dönme hareketi yaptığı kabul edilmiş ve kayma ihmal edilmiştir. Ayrıca tekerleklerin ağırlıkları, ataletleri ve oluşturdukları sürtünme ihmal edilmiştir (Şekil 3.2). Robotun merkez noktasının tekerlekleri birbirine bağlayan eksenin orta noktası olduğu kabul edilmiştir.



Şekil 3.2 İdealleştirilmiş tekerlek modeli.

3.2 Gezgin Robotun Kinematik ve Dinamik Denklemleri

Gezgin robotların modellenmesi için birçok yöntem bulunmaktadır. Burada x-y düzleminde hareket eden gezgin robotun hareket denklemleri verilecektir. Tez kapsamında kinematik model kullanıldığından dinamik modelin denklemlerinden kısaca bahsedilmiştir.

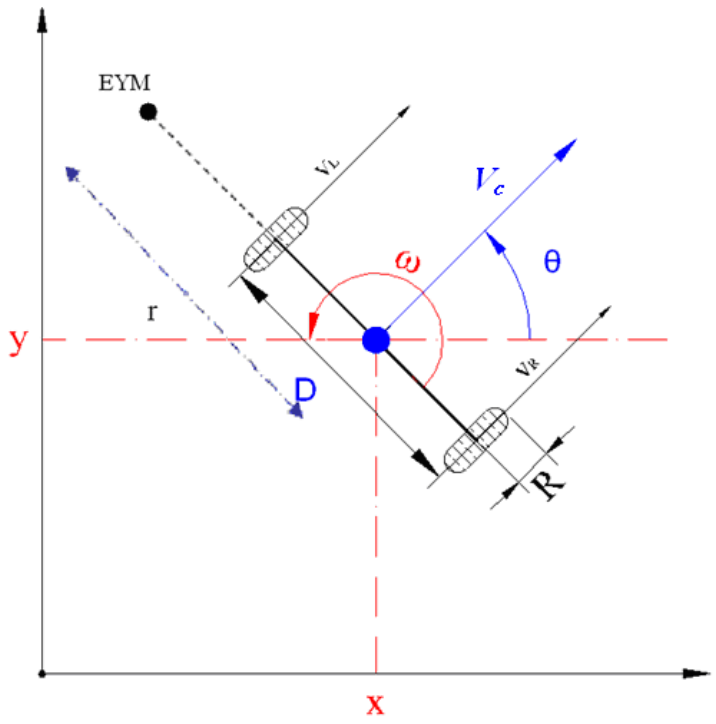
3.2.1 Gezgin Robotun Kinematik Modeli

Bir sistemin kinematik modeli çıkarılırken hareketi etkileyen kuvvetler dikkate alınmaz. Kinematik model sistemi oluşturan geometrik bağlantıları, kontrol değişkenleri ve durum uzayındaki sistem davranışı arasındaki ilişkiyi inceler.

Diferansiyel sürüş sistemli robotun durum değişkenleri aşağıdaki gibi gösterilebilir.

$$\vec{x} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

Burada x : robotun x koordinatını, y : robotun y koordinatını ve θ ise robotun yönelimini (oryantasyonunu) belirtir. Şekil 3.3'te gösterildiği gibi robotun sol ve sağ tekerleklerin doğrusal hızları sırasıyla v_L ve v_R olarak, ω ise açısal hız olarak tanımlanmıştır. D: robotun tekerlekleri arasındaki mesafe, EYM ise eğrilik yarıçapı merkezidir.



Şekil 3.3 Robotun koordinat düzlemindeki gösterimi.

Buna göre robotun merkez noktasına göre hızı (3.1)'deki gibi hesaplanır.

$$v_c = \frac{\left(v_L + v_R\right)}{2} \tag{3.1}$$

Problemin geometrisine göre robotun merkez noktasının x ve y koordinatlarına göre sahip olduğu hız ve yönelim açısı değişimi (3.2)'deki gibi ifade edilir.

$$\dot{x} = v_c \cos\theta$$

$$\dot{y} = v_c \sin\theta \tag{3.2}$$

$$\dot{\theta} = w$$

(3.2)'de verilen robotun kinematik denklemleri matrissel olarak da gösterilebilir.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_c \\ \omega \end{bmatrix} \tag{3.3}$$

Robotun üç durum değişkeni *(x,y,θ)* ve iki kontrol değişkeni bulunmaktadır. Kontrol edilecek durum değişkeni sayısının, kontrol değişkeni sayısından fazla olduğu sistemler holonom olmayan sistemler olarak adlandırılır. Bu sistemdeki kontrol değişkenleri sol ve sağ tekerlekleri süren DC motorların açısal hızları, ω_L ve ω_R'dir. Şekil (3.4)'te tekerleklerin yarıçapı R olarak gösterilmiştir. Böylece doğrusal hızlar açısal hızlar cinsinden ifade edilebilir.

$$v_L = \omega_L R$$

$$v_R = \omega_R R \tag{3.4}$$

Robotun doğrusal tekerlek hızları birbirinden farklı olduğunda robot Şekil 3.3'de gösterilen r yarıçaplı eğrinin çevresinde (EYM) döner. Yönelim açısının pozitif olduğu durumlarda EYM sol tekerleğin solunda oluşacaktır. EYM noktasındaki ω açısal dönme hızı her iki tekerlek için aynı olması gerektiğinden aşağıdaki denklemler yazılabilir.

$$\omega(r + D/2) = v_R$$

$$\omega(r - D/2) = v_L \tag{3.5}$$

(3.5)'teki denklemler kullanılarak açısal hız ω, doğrusal tekerlek hızları cinsinden elde edilir.

$$\omega = \frac{v_R - v_L}{D} \tag{3.6}$$

Matlab Simulink yazılımı altında robot modeli oluşturmak için kullanılacak olan denklem önceki denklemlerin yardımıyla oluşturularak matrissel formda gösterilmiştir.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\cos\theta & \frac{1}{2}\cos\theta & 0 \\ \frac{1}{2}\sin\theta & \frac{1}{2}\sin\theta & 0 \\ 1/D & -1/D & 0 \end{bmatrix} \begin{bmatrix} v_R \\ v_L \\ \theta \end{bmatrix} \tag{3.7}$$

3.2.2 Gezgin Robotun Dinamik Modeli

Bir sistemin dinamik modeli o sistemi etkileyen tüm kuvvetler göz önüne alınarak oluşturulur. Holonomik olmayan sistemlerin dinamik denklemlerinin oluşturulmasında birçok yöntem kullanılmaktadır. Bunlardan en sık kullanılanı Euler-Lagrange denklemleridir (Bloch vd., 1992). Gezgin robotun dinamik denklemleri şu şekilde verilebilir:

$$M(q)\ddot{q} = C(q,\dot{q})\dot{q} + G(q) = B(q)\tau + J^T(q)\lambda \tag{3.8}$$

Holonomik olmayan kısıt denklemi:

$$J(q)\dot{q} = 0 \tag{3.9}$$

Yukarıdaki denklemlerde *q* robotun durum değişkenleri vektörünü, *M(q)* $n \times n$ boyutlu pozitif tanımlı simetrik bir matrisi, $C(q,\dot{q})$ merkezcil tork ve coriolis (savrulma) momentlerini gösteren *n* boyutlu bir vektörü, *G(q)* yerçekimi momentlerini gösteren *n* boyutlu bir vektörü, *B(q)* $n \times r$ boyutlu girdi dönüşüm matrisini ($r<n$), τ *r* boyutlu girdi vektörünü ve λ ise kısıt koşullarının Lagrange çarpanlarını gösterir.

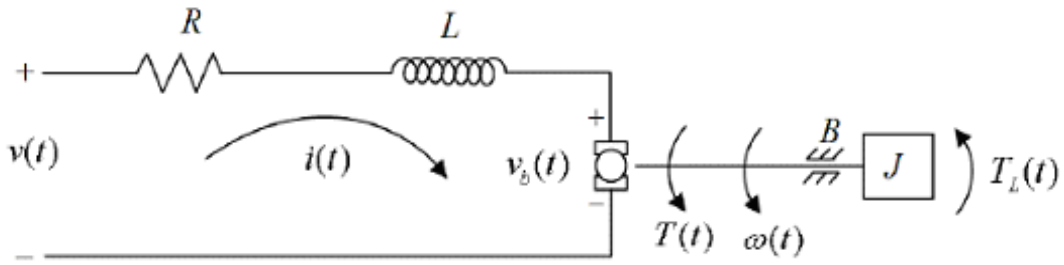
G(q) ve $C(q,\dot{q})$ 'nin 0 olduğu kabul edilerek denklem (3.8) ve (3.9) yeniden düzenlenirse

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{1}{R} \begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ L & -L \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} + \begin{bmatrix} \sin\theta \\ -\cos\theta \\ 0 \end{bmatrix} \lambda \tag{3.10}$$

elde edilir. τ_1, τ_2 sol ve sağ motorların momentlerini, *m* ve *I* robotun kütlesini ve ataletini belirtir. *R* tekerleklerin yarıçapını ve *L* ise tekerlekler arası mesafeyi gösterir. Bu kabuller yapıldıktan sonra holonomik olmayan kısıt eklenerek denklemler çözülür.

3.3 DC Motorun Matematiksel Modeli

DC motorlar birçok kontrol sisteminde yer almaktadır. Özellikle tekerlekli gezgin robotlarda eyleyici olarak görev yaparak robot hareketinin kontrolünü kolaylıkla sağlar. Benzetim çalışmalarını gerçekleştirmeden önce motor özelliklerine bağlı olarak uygun bir model geliştirilmelidir. Şekil 3.4' te DC motorun elektriksel devre modeli gösterilmiştir.



Şekil 3.4 DC motorun şematik gösterimi.

3.3.1 DC Motor Diferansiyel Denklemleri

T motor torku, *i(t)* armatür akımıyla doğru orantılıdır ve şu şekilde gösterilir:

$$T = K \cdot i \tag{3.11}$$

Rotor manyetik bir DC alan içersinde döndüğünde oluşan gerilim (ters emk) v_b, rotorun açısal hızına bağlı olarak değişir.

$$v_b = K_e \omega = K \frac{d\theta}{dt} \tag{3.12}$$

Şekil 3.4 'e göre Kirchoff'un gerilimler kanunu ve Newton'un kanunları kullanılarak aşağıdaki diferansiyel denklemler elde edilir.

$$\begin{aligned} J\frac{d^2\theta}{dt^2} + b\frac{d\theta}{dt} &= T + T_L = K \cdot i \\ L\frac{di}{dt} + R_m \cdot i &= V - K\frac{d\theta}{dt} \end{aligned} \tag{3.13}$$

Denklem (3.13)' te J toplam ataleti, b rotor viskoz sürtünme katsayısını, T_L motor ve yükün ters yönde etki eden sürtünme torkunu (gerektiği takdirde ihmal edilebilir), R_m ve L sırasıyla armatür direncini ve endüktansını, K ise tork sabitini vermektedir.

3.3.2 DC Motor Transfer Fonksiyonu

Laplace dönüşümü kullanılarak (3.13) şu şekilde yazılabilir.

$$Js^2\theta(s)+bs\theta(s)=KI(s)$$
$$LsI(s)+R_mI(s)=V(s)-Ks\theta(s) \tag{3.14}$$

s Laplace operatörünü gösterir. (Denklemler yazılırken ilk koşulların sıfır olduğu kabul edilmiştir.)

(3.14)' te verilen 2.eşitlikte *I(s)* yalnız bırakılıp elde edilen ifade 1.eşitlikte yerine konur.

$$I(s)=\frac{V(s)-Ks\theta(s)}{R_m+Ls} \tag{3.15}$$

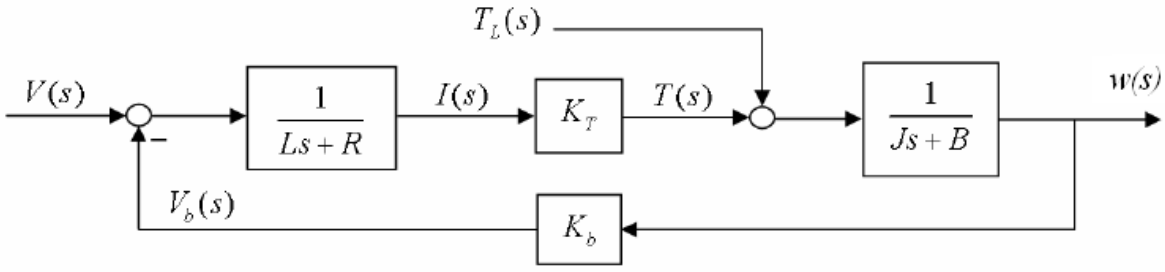
$$Js^2\theta(s)+bs\theta(s)=\frac{K(V(s)-Ks\theta(s))}{R_m+Ls} \tag{3.16}$$

Denklem (3.16)'dan DC motorun çıkış açısıyla *θ(s)*, giriş gerilimini *V(s)* ilişkilendiren transfer fonksiyonu şu şekilde oluşturulabilir.

$$G_a(s)=\frac{\theta(s)}{V(s)}=\frac{K}{\left\{s\left[(R_m+Ls)(Js+b)+K^2\right]\right\}} \tag{3.17}$$

(3.17)'den de görüldüğü gibi açısal hızla *ω(s),* giriş gerilimini *V(s)* ilişkilendiren transfer fonksiyonu kolaylıkla elde edilebilir.

$$G_w(s)=\frac{\omega(s)}{V(s)}=\frac{K}{\left\{\left[(R_m+Ls)(Js+b)+K^2\right]\right\}} \tag{3.18}$$



Şekil 3.5 DC motor blok diyagramı.

3.3.3 DC Motor Parametrelerinin Belirlenmesi

Gezgin robotun hareket kontrolünü gerçekleştirecek olan DC motorlar özdeş olarak seçilmiştir. DC motorların P = 8 watt gücünde, N= 5000 rpm hızında ve 12 V'luk olduğu kabul edilmiştir.

K tork sabitinin hesaplanması için hız bilgisi kullanılır.

$$\omega_m = \frac{V}{K} = \frac{2\pi N}{60} \tag{3.19}$$

K= 0.023 N m/A ve ω_m = 524 rad/s olarak bulunur.

Denklem (3.13)'te; ω ve I' nın kararlı hal durumları için $\frac{d\omega}{dt} = 0$ kabul edilir ve yeniden düzenlenir.

$$K \times i = b \cdot \omega \tag{3.20}$$

Verilen bilgilerden i= 0.666 A bulunur. Buna göre $T = Ki$ = 0.023*0.666 = 15.3 m Nm olarak hesaplanır.

K ve i bilindiğine göre b sönüm katsayısı denklem (3.20)' de yerine konarak elde edilir.

(0.023*0.666) − b*(524) = 0 → b = 0.00003 Nm s

Çizelge 3.1 DC motor parametreleri

V (volt)	J (kgm^2/s^2)	b (Nm s)	K (Nm/A)	R_m (Ω)	L (H)
12	0.01	0.00003	0.023	1	0.5

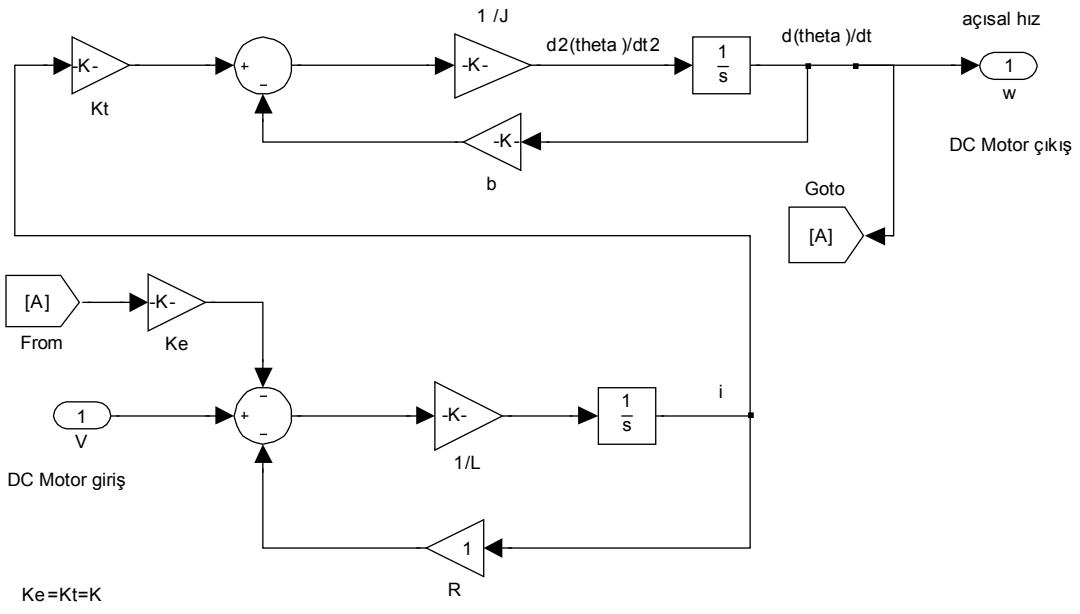
Çizelge 3.2 'de DC motorların durumlarına göre tekerlekli gezgin robotun hareketleri belirtilmiştir. Motorların dönüş yönleri belirlenirken robotun sağ tekerleğinin bulunduğu taraftan gözlem yapıldığı kabul edilmiştir. Tez kapsamında gezgin robot için sadece ileri yönlü hareket ve kendi ekseni etrafında dönme hareketi incelenecektir.

Çizelge 3.2 DC motorların dönüş yönüne bağlı olarak gezgin robotun hareketi

Sol Motor dönüş yönü	**Sağ Motor dönüş yönü**	**Gezgin Robotun Hareketi**
Saat yönü	Saat yönü	İleri yönlü hareket
Saat yönü	Saat yönü tersi	Kendi ekseni etrafında dönme
Saat yönü tersi	Saat yönü	Kendi ekseni etrafında dönme
Saat yönü tersi	Saat yönü tersi	Geri yönlü hareket
Durma	Saat yönü	Sağa dönme
Saat yönü	Durma	Sola dönme

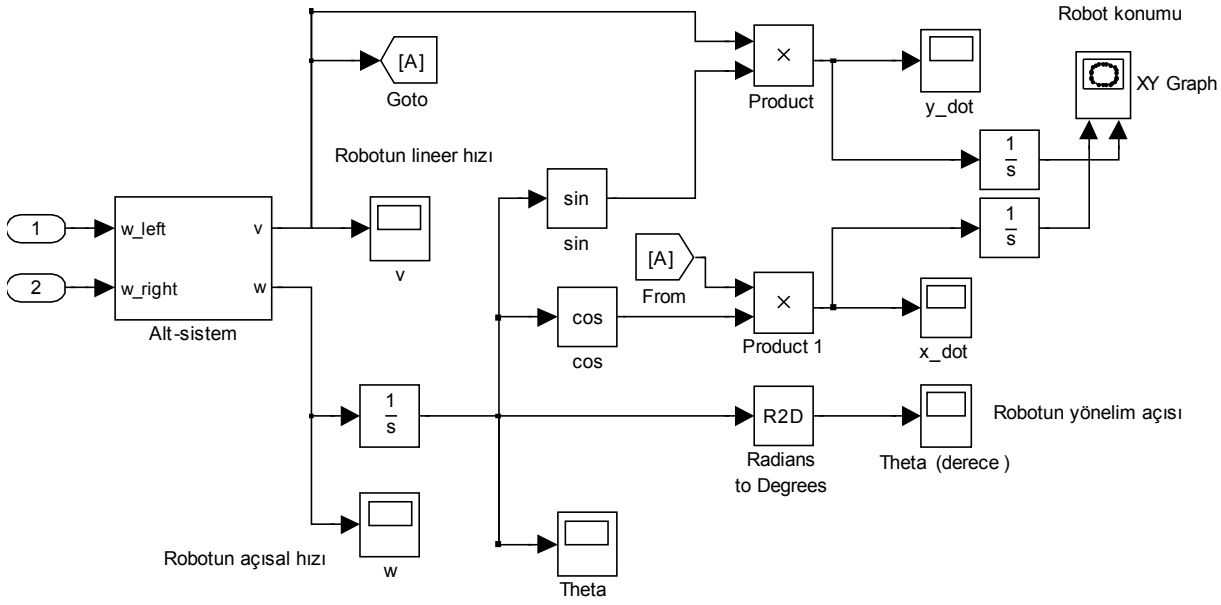
3.4 Otonom Gezgin Robot ve DC Motorun Simulink Modelleri

Önceki kısımlarda elde edilen denklemler yardımıyla Simulink yazılımı kullanılarak sisteme ait blok diyagramlar oluşturulmuştur.



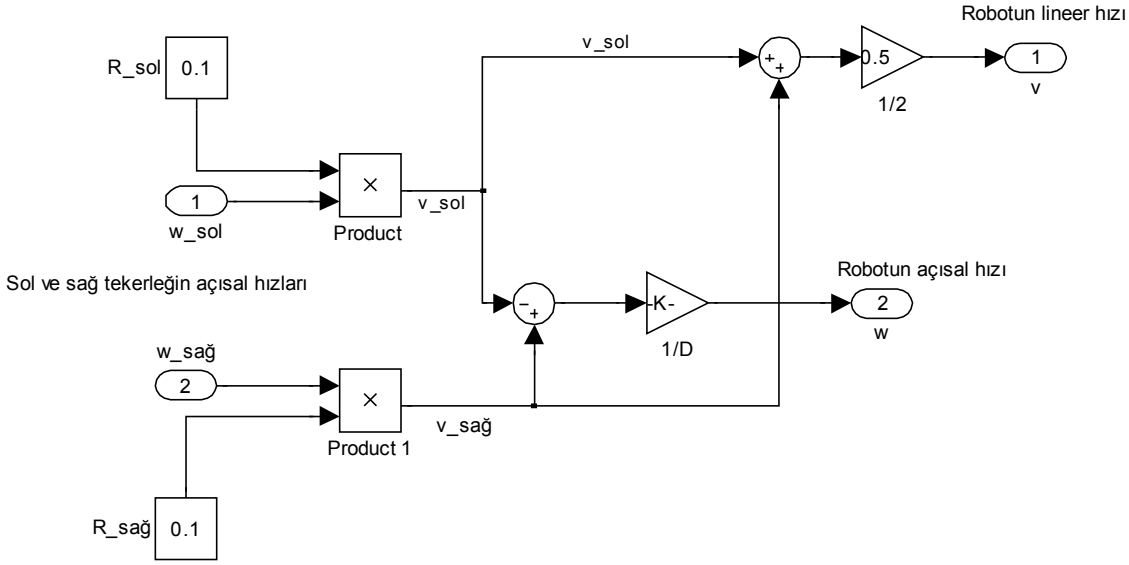
Şekil 3.6 DC motor simulink diyagramı.

Bir robota ait iki tekerlek ve bunları kontrol eden iki DC motor bulunmaktadır. Fakat DC motorlar özdeş olarak seçildiğinden Şekil 3.6' da sadece birine ait diyagram gösterilmiştir.



Şekil 3.7 Gezgin robot simulink diyagramı.

Şekil 3.7' de gösterilen alt-sistemin 1 ve 2 nolu girişleri sırasıyla sol DC motorun açısal hızını ve sağ DC motorun açısal hızını göstermektedir. Gezgin robot sistemi denklem (3.7) kullanılarak oluşturulmuştur. Şekilde gösterilen alt sistem tekerleklerin açısal hızları ile robotun lineer ve açısal hızı arasındaki bağlantıyı sağlamaktadır. Bu sistem denklem (3.4) ve (3.6) baz alınarak oluşturulmuştur.



Şekil 3.8 Alt sistem blok diyagramı.

4. ALGORİTMALAR

Tezin bu bölümünde gezgin robotların yörüngelerinin planlanmasında kullanılan A* algoritması detaylıca anlatılmıştır. Daha sonra A* yardımıyla oluşturulan optimal yörüngelerin takibi için kullanılan PID kontrolör açıklanmıştır.

Yörünge planlanması ve planlanan yörüngenin takibi tekerlekli gezgin robotlarla yapılan çalışmalarda önemli bir tutmaktadır. Bu alanda yapılan çalışmalarda parametrelerin (zaman, mesafe, enerji vb.) en iyilenmesi gibi problemler öne çıkmaktadır. (Stentz, 1994) Problemlerin çözümünde kullanılan algoritmaların zayıf yönlerinin geliştirilmesi veya alternatif çözümlerin oluşturulması aktif bir araştırma konusudur. (Howard, 2005)

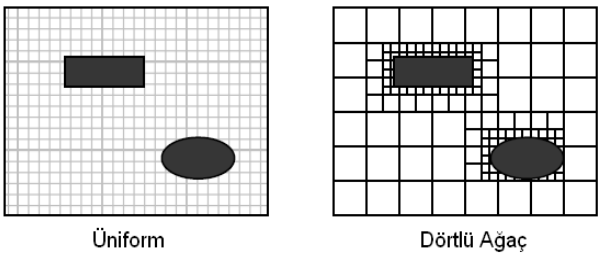
4.1 Hücrelere Bölme Metodu ve Arama Algoritmaları

Otonom gezgin robotların yörünge planlaması için algoritma geliştirilirken iki temel prensip kullanılmıştır. Bu temel prensipler şunlardır: iki nokta arasındaki en kısa mesafe o noktaları birleştiren bir doğrudur ve en basit çözüm, içinde en az matematik barındıran ve matematiksel olarak gerçeklenebilen çözümdür.

Bu prensiplerden hareketle bu çalışmada iki gezgin robotun verilen başlangıç noktalarından hedef noktalarına en kısa uzaklığı olan yörüngelerden gitmesi ve ortamdaki engellere çarpmadan ilerleyebilmesi için bir arama algoritması kullanılmıştır.

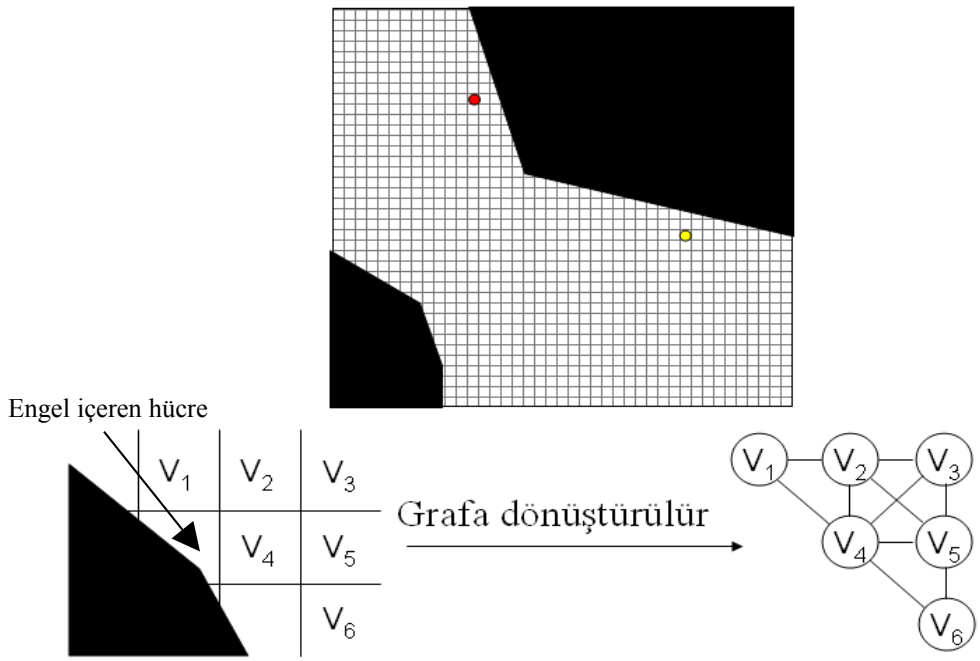
Arama algoritmalarının kullanılabilmesi için öncelikle çalışma uzayının tanımlanması ve ayrıklaştırılması gerekir. Yani bölüm 2.1.1.2'de bahsedilen hücrelere bölme metodu kullanılmalıdır. Hücrelere bölme metodu kullanıldıktan sonra ayrık uzay içinde optimal yörüngeler oluşturulur.

Hücrelere bölme metodu kendi içinde ikiye ayrılabilir. Bunlar tam hücrelere bölme ve yaklaşık hücrelere bölmedir. Yaklaşık hücrelere bölmede *C parametre uzayı* tekdüze (üniform) yapıda veya dörtlü ağaç yapısında hücrelere ayrılır. Tez çalışmasında tekdüze (üniform) yapı kullanılacaktır.



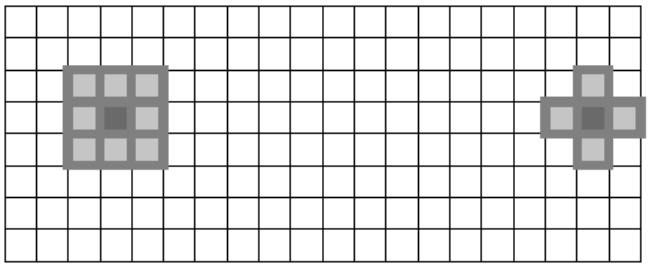
Şekil 4.1 *C* parametre uzayının hücrelere ayrılması.

Hücrelere ayrılmış olan parametre uzayından *G* grafı oluşturulur. Bu grafta $C_{ser} \subseteq C$ olan her bir hücre bir düğüme (*V*) denk gelir. Bu düğümler birbirlerine kenar çizgileri denilen yapılar (*E*) ile bağlıdır (Şekil 4.2). Hareket planlama problemlerinde düğümler genellikle parametre uzayındaki (*C*) noktaları ifade ederler.



Şekil 4.2 Graf ve düğümler.

Başlangıç noktasından (düğümünden) hedef noktasına (düğümüne) ulaşabilmek için bir düğümün etrafındaki düğümlerle bağlanması ve komşuluk matrisinin belirlenmesi gerekir. Şekil 4.3' te sol tarafta 8 komşuluklu bir düğüm görülürken sağ tarafta 4 komşuluklu bir düğüm görülmektedir.



Şekil 4.3 Düğüm komşulukları.

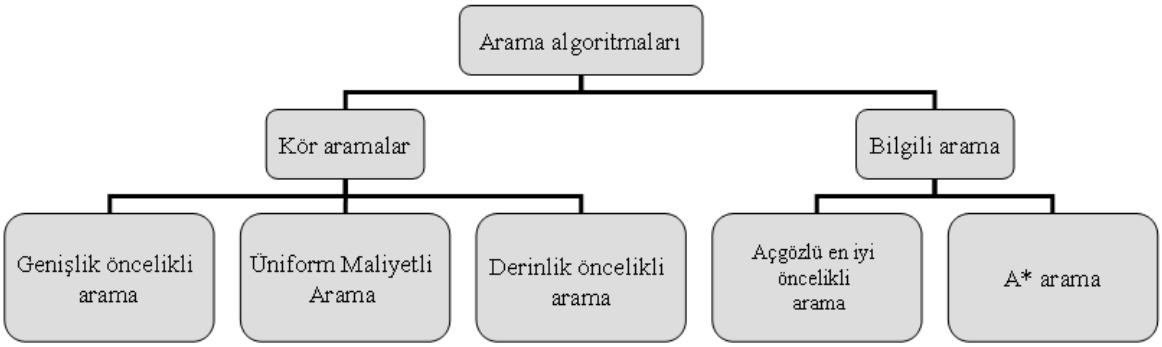
Şekil 4.2' de gösterilen düğümler *8* komşuluk bağlantısına sahiptir ve komşuluk matrisi şu şekilde ifade edilebilir. Burada *E* düğümler arası kenar çizgilerini ifade eder.

$a_{ij} = 1$ eğer $\left(v_i, v_j\right) \in E$

$a_{ij} = 0$ eğer $\left(v_i, v_j\right) \notin E$

Komşuluk matrisi $\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$ şeklinde gösterilir.

Başlangıç düğümünden hedef düğümüne ulaşmak için gerekli en kısa yolun bulunmasında bir arama algoritmasına ihtiyaç vardır. Arama algoritmaları temel olarak Şekil 4.4' teki gibi sınıflandırılabilir.



Şekil 4.4 Arama algoritmaları.

Arama algoritmaları genel olarak 4 kritere göre incelenir.

1) Bütünlük : Bir tane çözüm olduğunda kullanılan arama algoritmasının çözümü garantilemesi
2) Zaman karmaşıklığı : En kötü veya ortalama çözümün bulunması için gerekli zaman
3) Optimallik : Birkaç farklı çözüm bulunduğunda arama algoritmasının en iyi çözümü garantilemesi yani minimum maliyetli çözümün bulunması
4) Alan karmaşıklığı : Aramanın gerçekleştirilmesi için gerekli hafızanın boyutu

Arama algoritmalarının iki çeşidi vardır:

a) Kör (Bilgisiz) aramalar : Bu teknikler alternatif yolları ve bir karar çözümünün hipotezlerini tek tek araştırır.

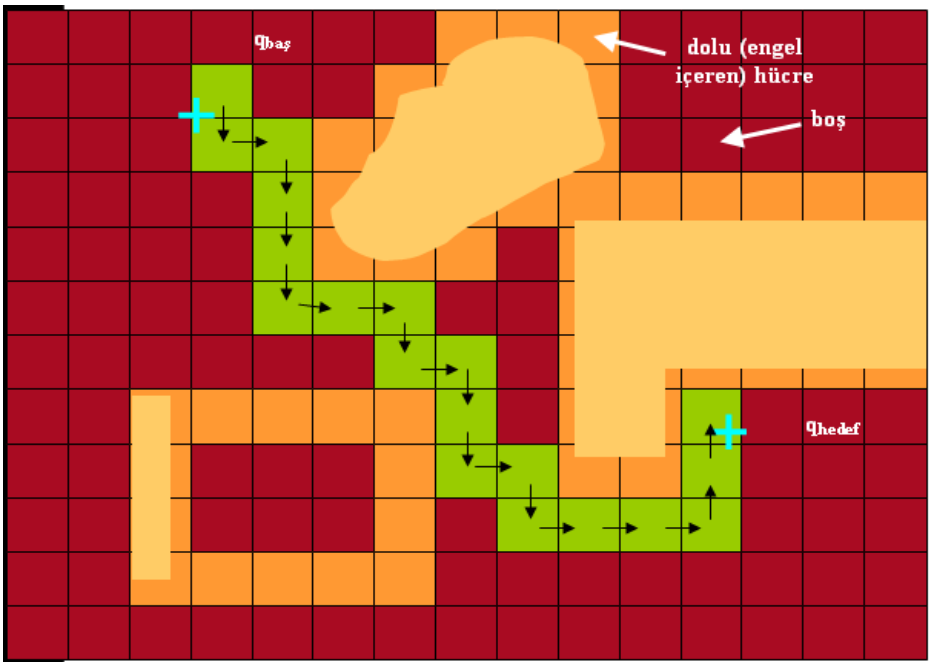
Kör aramalar da kendi arasında ikiye ayrılır: Ayrıntılı arama ve eksik arama. Ayrıntılı aramada tüm alternatifler değerlendirilir ve optimal bir çözüm bulunmaya çalışılır. Eksik aramada ise arama yeterli derecede iyi bir çözüm bulunana kadar sürer. Çözüm bulunabilmesine rağmen zaman ve hafıza kullanımı konusunda performansları kötü olduğundan büyük problemler için pratik değildir. Kör bir aramada değerlendirilen alternatifler üssel olarak artabilir. Genişlik öncelikli arama, derinlik öncelikli arama ve sınırlı derinlikte arama bilgisiz arama algoritmalarındandır.

b) Bilgili aramalar: Bu tip aramalarda çalışma uzayı (çevre) bilgisi önceden bilinmektedir.

Birçok uygulama için arama işlemini yönlendirecek bilgilere ihtiyaç duyulur. Bilgili aramalar 'sezgisel' bilgileri kullanarak problemin nasıl çözülmesi gerektiğine karar verir. Bilgisiz (kör) aramalara göre çok daha hızlıdır ve çözümü daha etkin olarak bulurlar. Bilgili arama algoritmalarına örnek olarak en iyi öncelikli arama, aç gözlü en iyi öncelikli arama ve A* arama verilebilir.

4.1.1 A* (A-Yıldız) Arama Algoritması

Bu algoritma çalışma uzayının hücresel yapılara bölünmesiyle elde edilen graf üzerinde $q_{baş}$ ve q_{hedef} düğümleri arasındaki en düşük maliyetli (optimal) yolu seçerek çalışır (Şekil 4.5). Kolaylıkla uygulanabilmesinden dolayı birçok alanda sıklıkla kullanılmaktadır.



Şekil 4.5 Optimal yolun bulunması.

Algoritma düğümler hakkında bilgiyi saklamak için iki tip listeye ihtiyaç duyar. Bunlardan

ilki açık listedir (*A*). Açık listede ilerletilecek düğümlerin bilgileri saklanır. Yani düğüm komşuluklarına göre engel içermeyen sonraki düğümlerin bilgileri bulunur. Diğer liste olan kapalı listede *(K)* ise ziyaret edilmiş düğümler veya engel içeren düğümler bulunur. Bir düğüm kapalı listeye eklenmiş ise o düğümün çocuk (sonraki) düğümleri yaratılmaz.

Örneğin *Q* araştırılacak düğümlerin listesi olsun. Bir *N* düğümü *Q* listesine ilk olarak eklendiğinde ziyaret edilmiş olur. Henüz çocuk (sonraki) düğümleri yaratılmamıştır. Bir *N* düğümü *Q*' dan çekildiği anda ise ilerletilmiş olur. Bu durumda *N* düğümünün bütün çocukları ziyaret edilmiş demektir. İlerletilen *N* düğümü kapalı listede bulunur.

A* algoritmasının yaklaşımına göre her düğüm için bir istenebilirlik (maliyet) fonksiyonu *f(n)* kullanılır.

$$f(n) = g(n) + h(n) \tag{4.1}$$

Denklem (4.1)' de *g(n)* başlangıç düğümünden o anda bulunulan düğüme gitmek için gerekli maliyeti tanımlar, *h(n)* ise bulunulan düğümden hedef düğüme gitmek için gerekli maliyeti tanımlayan sezgisel bir fonksiyondur. *h(n)* çalışma uzayının bilgilerini kullanarak o anda bulunulan düğümün hedef düğüme yakınlığını tahmin eder.

Bütün düğümler için $h(n) \geq 0$ 'dır. Eğer *h(n)*=0 ise *n* hedef düğümdür. $h(n) = \infty$ 'da ise hedefe ulaşılamayan bir düğüme gelindiği ifade edilir. İyi bir sezgisel fonksiyon hızlı hesaplama yapmalıdır. Yörünge planlamasında kullanılan en temel sezgisel fonksiyonlar Manhattan uzaklığı ve Euclidean uzaklığıdır. Tez çalışmasında *h(n)* fonksiyonu hesaplarken Euclidean uzaklığı kullanılmıştır.

1) Manhattan uzaklığı : $h(n) = |x_a - x_b| + |y_a - y_b|$ (4.2)

2) Euclidean uzaklığı : $h(n) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$ (4.3)

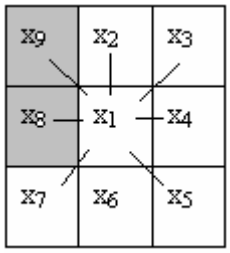
Şekil 4.6'da düğümler arası maliyetin hesaplanmasına ilişkin bir örnek gösterilmiştir. Robotun 8 yönde hareket ettiği kabul edilmiştir. x_1 düğümünde bulunan bir robot için;

$m(x_1,x_2)=1$ → x_1 düğümünden x_2 düğümüne giderken oluşan maliyet (x_2 boş bir hücredir.)

$m(x_1,x_3)=\sqrt{2}=1.4$ → x_1 düğümünden x_2 düğümüne giderken oluşan maliyet

$m(x_1,x_8)=10000$ → x_1 düğümünden x_8 düğümüne (engel içeren bir düğüm) giderken oluşan maliyettir.

Algoritma minimum maliyetleri seçerek ilerlediğinden robot engel içeren düğümlere uğramadan hedefe ilerler.



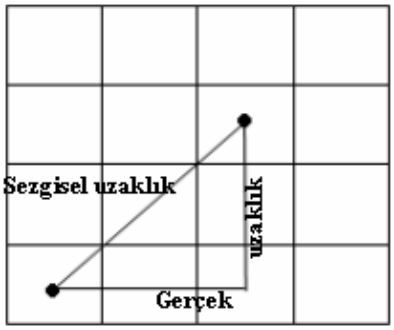
Şekil 4.6 Bir düğümdeki maliyetin hesaplanması.

4.1.1.1 A* Arama Algoritmasının Özellikleri

A* (A-Yıldız) arama algoritmasının performansı *h(n)* sezgisel fonksiyonunun ne kadar iyi seçildiğine bağlıdır. *h(.)* sezgisel fonksiyonu (4.4) koşulunu sağladığında kabul edilebilir yani tutarlıdır.

$$\forall N_i \in N : 0 \le h(N_i) \le h^*(N_i) \tag{4.4}$$

Bu eşitsizlikte $h^*(N_i)$ i.düğümden hedef düğüme ulaşmak için gerekli gerçek maliyeti belirtir. Denklem (4.4)'e göre kabul edilebilir bir sezgisel fonksiyonun hedefe ulaşmak için öne sürdüğü maliyet hiçbir zaman gerçek masraftan fazla olamaz. Şekil 4.7' de bu ifade basitçe gösterilmiştir.



Şekil 4.7 *h(.)* kabul edilebilir sezgisel fonksiyon.

A* arama algoritması (4.4) koşulu altında bütünlük sağlar ve optimal çözümü garantiler (Russell ve Norving, 2003). Bu algoritmayı değerlendirirken göz önüne alınması gereken önemli bir diğer nokta algoritmanın koşulma zamanıdır. Bu kriter en kötü durum senaryosu kabul edilerek yapıldığında genellikle üssel bir yapı gösterir. Burada 'üssel' herhangi bir polinom fonksiyon tarafından sınırlandırılmamış anlamına gelmektedir.

Bir diğer önemli kriter de algoritmanın kullanacağı bellek miktarıdır. Ziyaret edilmiş durumların bellekte tutulması algoritmanın çözüme ulaşma zamanını hızlandırır. Fakat büyük

ölçekli problemler için bellek sorunu oluşturabilir. Bu nedenle yüksek serbestlik derecesi olan robot uygulamalarında kullanılması tavsiye edilmez.

Çizelge 4.1 A* arama algoritmasının performansı

Bütünlük	Zaman	Bellek	Optimal Çözüm
Evet Sonsuz sayıda $f \leq f(N)$ sağlayan durum olmadığı sürece	Üssel	Üssel Tüm düğümler bellekte tutulur	Evet

4.1.1.2 A* Arama Algoritmasının Sözde Kodu

Öncelikle algoritmanın daha iyi anlaşılabilmesi için bazı tanımlar verilecektir. Buna göre;

- Başlangıç düğümü dışındaki N_s, hiçbir düğüm daha önce ziyaret edilmemiştir.
- Başlangıç anında açık listede sadece N_s bulunur.
- N_{eniyi} açık listede bulunan en düşük f değerine sahip bir düğümdür.
- Engel içeren düğümler kapalı listeye konulur.

Algoritmanın sözde kodu şu şekilde verilebilir.

Sadece N_s başlangıç düğümünü içeren bir G grafı oluştur ve bunu açık listeye koy, N_h hedef düğümünü oluştur

while (iken) döngüsüne gir

açık liste boş değilse listede bulunan N_{eniyi} düğümü bul, listeden çıkar ve $N_{güncel}$ olarak adlandır

if (eğer) $N_{güncel}=N_{hedef}$ ise while döngüsünden çık

else (değilse) $N_{güncel}$' den sonra gelebilecek olan takipçi (çocuk) düğümleri oluştur

for (her bir) güncel düğümün her bir takipçi düğümü için

$g(N)$ maliyetini hesapla

eğer (if) açık listede daha düşük bir maliyetli düğüm varsa bu takipçi düğümü at ve devam et

eğer değilse (else if) takipçi düğüm kapalı listedeyse daha düşük bir maliyetli düğüm varsa bu takipçi düğümü at ve devam et

değilse (else) takipçi düğümün oluşumlarını açık ve kapalı listeden kaldır.

takipçi düğümlerin atalarını güncel düğüm olarak belirle.

h(.) sezgisel fonksiyonunu hedef düğümüne olan tahmini mesafe olarak belirle.

takipçi düğümü açık listeye ekle

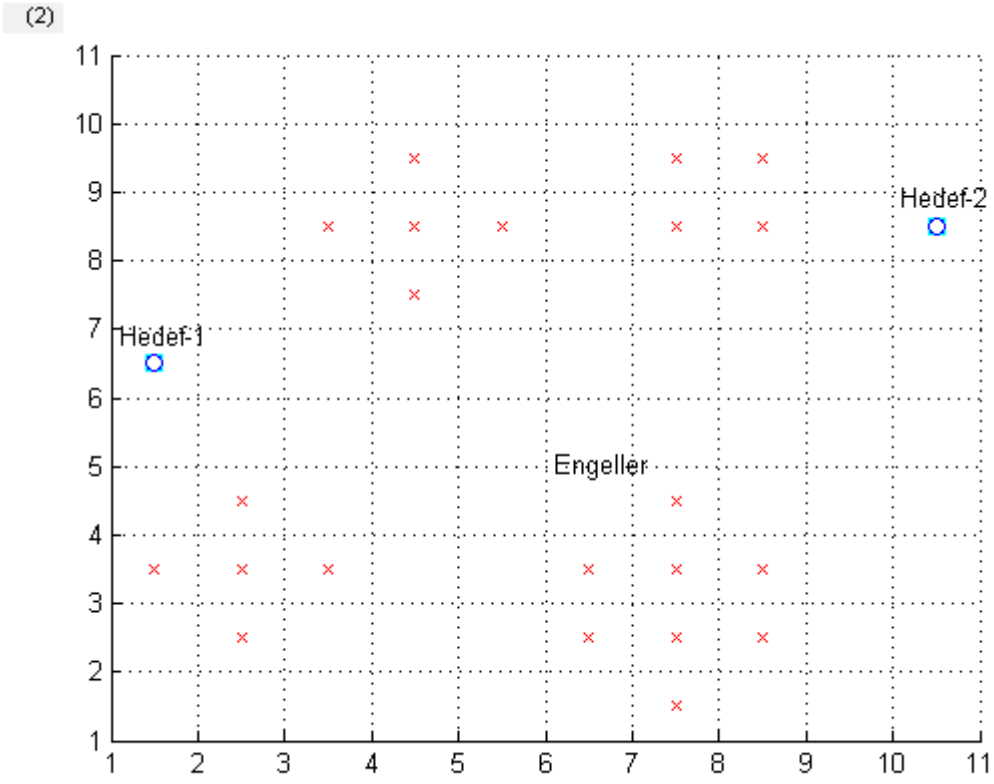
for döngüsü sonu

güncel düğümü kapalı listeye ekle

while (iken) döngüsü sonu

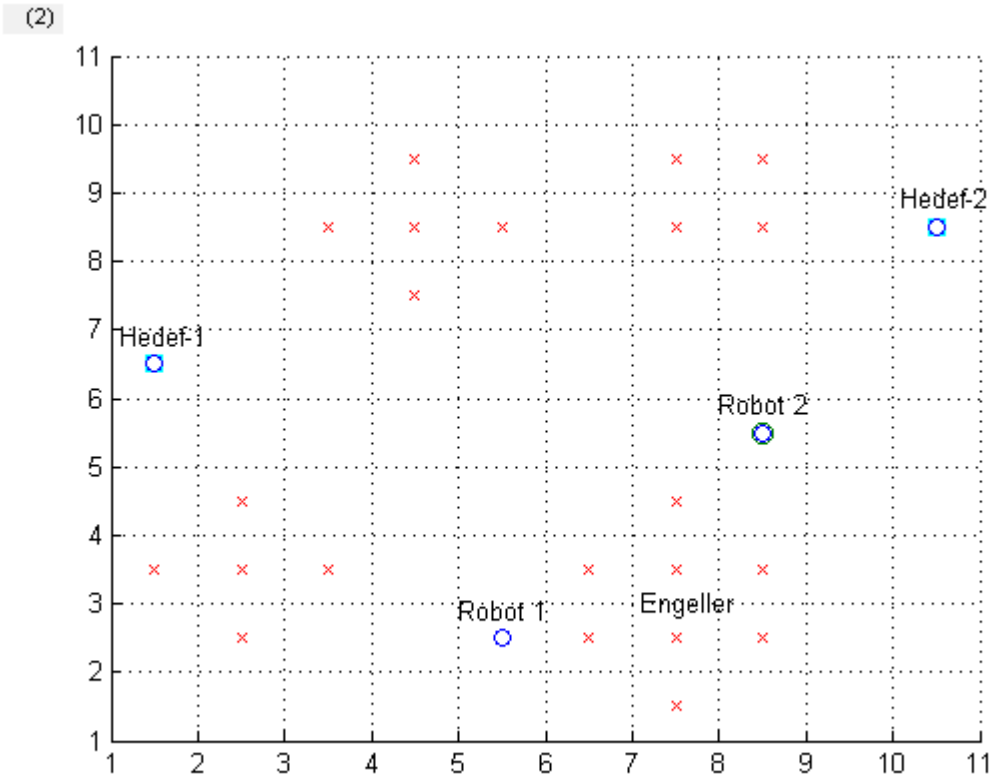
4.2 A* Arama Algoritmasıyla Yörünge Oluşturulması

Benzetim çalışmaları için oluşturulan harita 10×10 karesel hücrelerden oluşmaktadır. Her bir düğümün (karenin) *4* düz ve *4* tane de çapraz yönde olmak üzere *8* komşuluk bağlantısı bulunmaktadır. Yatay ve dikey geçişlerde maliyet 1 olarak belirlenmiştir. Çapraz geçişlerde ise maliyet $\sqrt{2}$ =1.4 olarak saptanmıştır.



Şekil 4.8 Çalışma uzayı, hedefler ve engellerin tanımlanması.

Yapılan tez çalışmasında, global bir planlama yöntemi olan A* arama algoritması kullanıldığından çalışma uzayı yani çevre önceden bilinmelidir (Şekil 4.8).

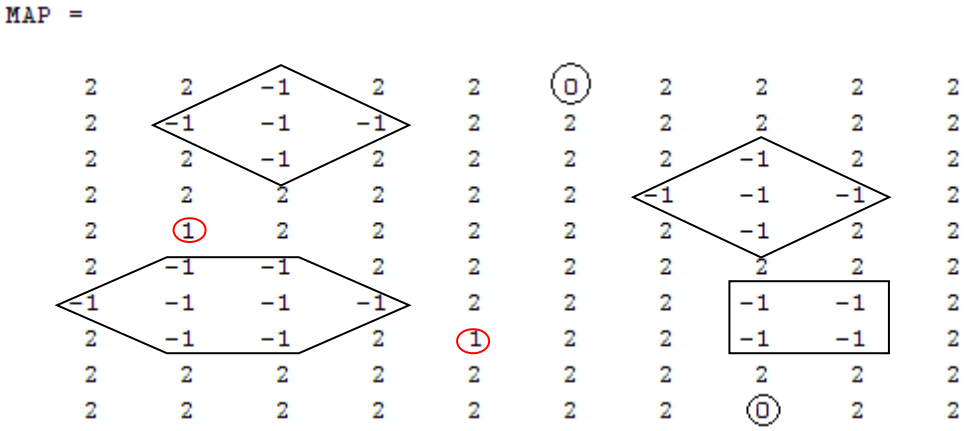


Şekil 4.9 Robotların başlangıç konumları, hedefler ve engellerin tanımlanması.

4.2.1 Çevre Matrisi Oluşturma

Algoritma, üç boyutlu sayısal bir çevreyi iki boyutlu bir matrise indirgeyerek kullanır. Oluşturulan çevre matrisindeki satır ve sütunlar robotların başlangıç koordinatlarını, engellerin ve hedeflerin koordinatlarını içermektedir. Bu matriste engeller *-1*, hedefler *0*, robotların başlangıç konumları *1* ve boş uzay (C_{ser}) da *2* sayısı ile tanımlanmıştır. Buna göre çevre matrisi Şekil 4.10'daki gibi oluşturulabilir. Çevre matrisi oluşturulurken Şekil 4.9' daki tanımlamalar göz önüne alınmıştır.

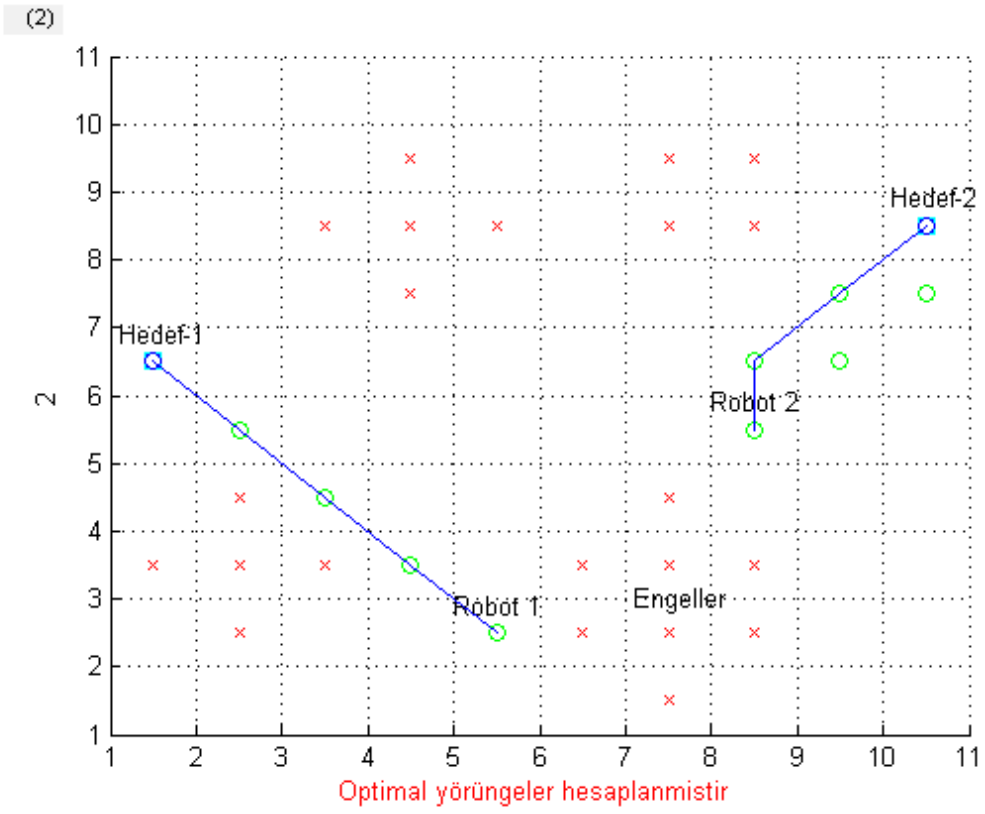
Çevre matrisi A* algoritmasının çalışmasında temel bir rol oynar. Bu matris dinamik durumlara göre tekrar oluşturulabilir veya yenilenebilir. Buna rağmen algoritma çevre matrisinin o anda bulunulan durumu gösteren kısmını kullanarak optimal yörünge hesaplar. Bunun sebebi gerçek zamanlı sistemlerde kullanılan sensörlerin algılamalarının sınırlı oluşudur.



Şekil 4.10 Çevre matrisi.

Yukarıdaki çevre matrisinde sütunlar y eksenini satırlar ise x eksenini göstermektedir. Örneğin x_5 ve y_2 de bulunan *1* sayısı robot-1'i ifade eder. Benzer şekilde x_{10}, y_8' de bulunan *0* ise hedefi belirtir.

Algoritma 2 veya daha fazla robot için yörünge planlama gerçekleştirmektedir. A* arama algoritmasının gerçekleştirilmesini sağlayan Matlab kodu ekler bölümünde verilmiştir. Şekil 4.11' de 2 robot - 2 hedef için optimal yörüngeler hesaplanmıştır.

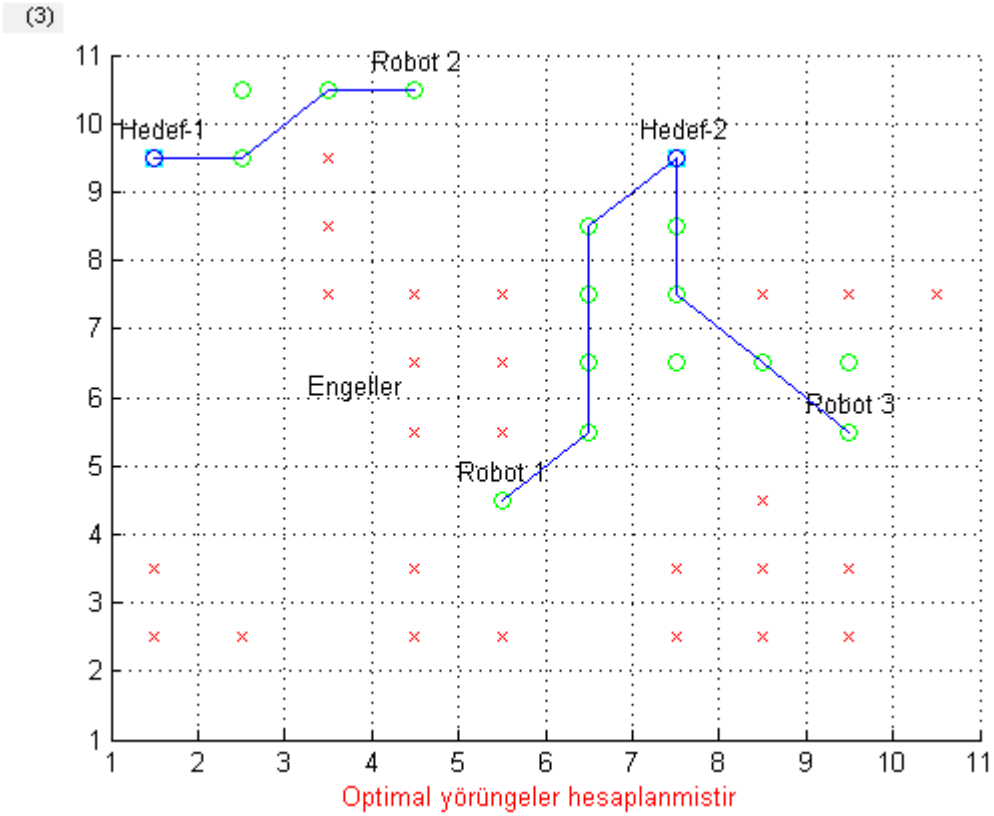


Şekil 4.11 2 robot - 2 hedef için optimal yörüngelerin bulunması.

Çizelge 4.2 2 robot - 2 hedef için optimal yörünge uzunlukları (m)

Robot 1 - Hedef 1	Robot 2 - Hedef 2
5.6569	3.82

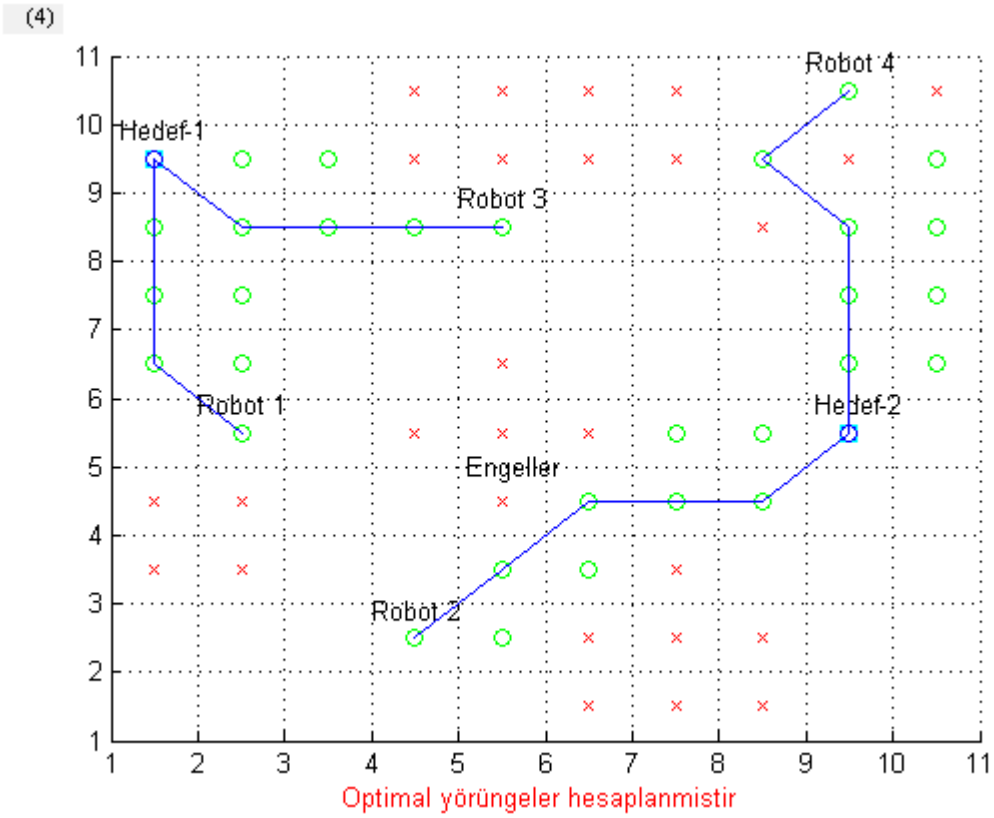
Algoritma 2' den fazla robotla da çalıştırılabilir. Şekil 4.12 ve 4.13' te sırasıyla 3 ve 4 robotlu uygulama için optimal yörüngelerin bulunması işlemi tekrar edilmiştir.



Şekil 4.12 3 robot - 2 hedef için optimal yörüngelerin bulunması.

Çizelge 4.3 3 robot -2 hedef için optimal yörünge uzunlukları (m)

Robot 1 - Hedef 2	Robot 2 - Hedef 1	Robot 3 – Hedef 2
5.82	3.41	4.82



Şekil 4.13 4 robot - 2 hedef için optimal yörüngelerin bulunması.

Çizelge 4.4 4 robot - 2 hedef için optimal yörünge uzunlukları (m)

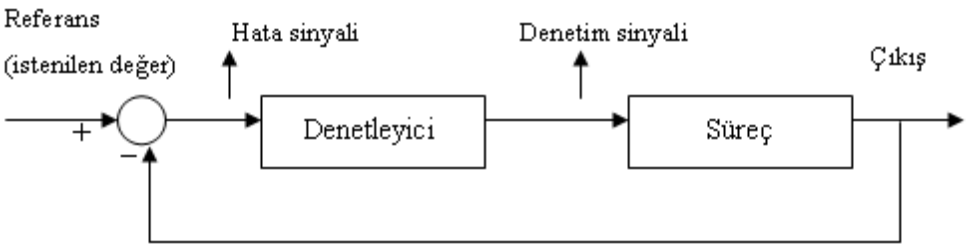
Robot 1 - Hedef 1	Robot 2 - Hedef 2	Robot 3 - Hedef 1	Robot 4 - Hedef 2
4.41	6.23	4.41	5.82

4.3 Kontrol Algoritması

Bu bölümde endüstride bulunan birçok sistemin kontrolünde yaygın olarak kullanılan üç terimli denetleyicinin (PID denetleyici) yapısı ve kullanılacağı denetim sistemi ele alınacaktır.

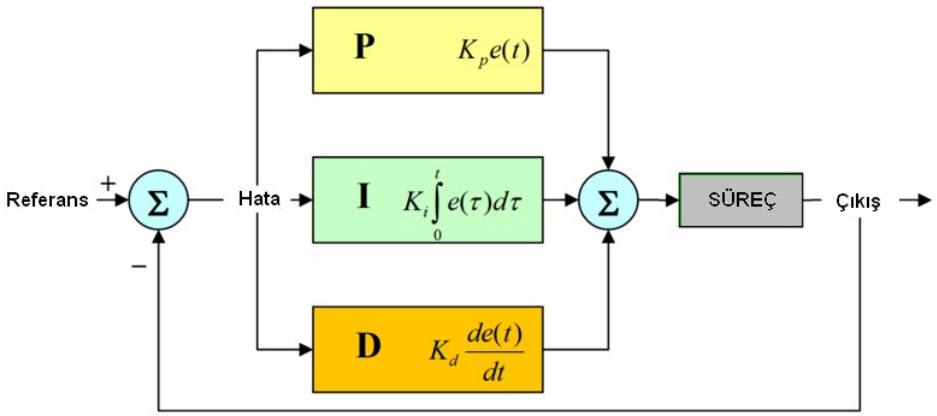
4.3.1 PID Denetleyici

Geri beslemeli bir denetim sisteminin hedefi girişe gerekli sinyalleri vererek çıkışın kontrol edilebilmesini sağlayacak bir denetleyici tasarlamaktır. (Şekil 4.14)



Şekil 4.14 Geri beslemeli bir sistemin blok diyagramı.

Bir PID denetleyici üç tip denetleyiciden oluşur. Bunlar oransal denetleyici K_p, integral denetleyici K_i ve türevsel denetleyici K_d' dir. Şekil 4.15' teki *e(t)* değişkeni hata sinyalini belirtir.



Şekil 4.15 PID denetleyicinin blok diyagramı.

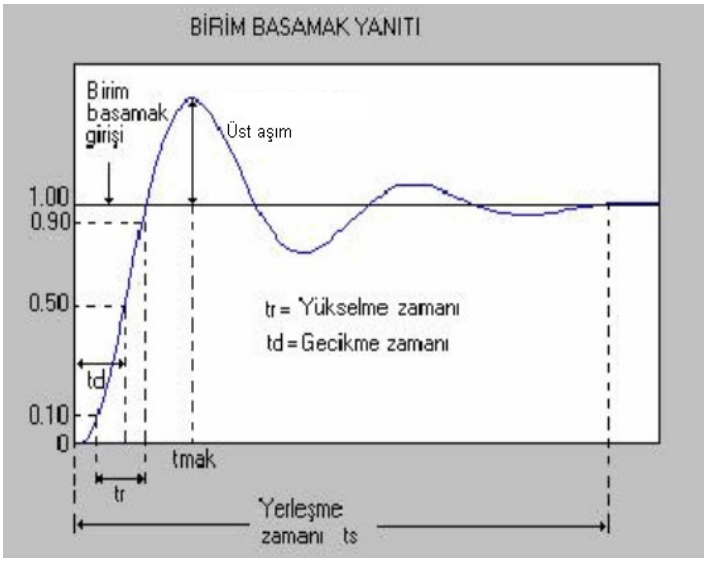
PID denetleyici hata sinyalinin türevini, integralini ve oransal kazancını hesaplar. Daha sonra bunların toplamını alarak sürecin girişine uygular ve çıkış işareti elde edilir. PID denetleyicinin çıkışı olan $u(t)$ sinyali denklem (4.5)' teki gibi ifade edilir.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de}{dt}(t) \tag{4.5}$$

Oransal denetleyici K_p yükselme zamanını azaltır ama kararlı hal hatasını hiçbir zaman ortadan kaldırmaz. Bir integral denetleyici K_i sistemin kararlı hal hatasını ortadan kaldırmada etkilidir fakat geçici cevabı kötüleştirebilir. Türevsel denetleyici K_d sistem kararlığının artmasını, üst aşımın azalmasını ve geçici cevabın düzelmesini sağlar. Her bir denetleyicinin kapalı çevrim bir sisteme etkisi Çizelge 4.5'teki gibi özetlenebilir. Şekil 4.16'da ise örnek bir sistemin birim basamak girişine verdiği cevap ve sistem tasarım kriterleri (t_r, t_d, t_s, üst aşım) gösterilmiştir.

Çizelge 4.5 Oransal, integral ve türevsel denetleyicilerin karakteristikleri

Denetleyici	Yükselme zamanı	Üst aşım	Yerleşme zamanı	Kararlı hal hatası
K_p	Kısalır	Artar	Çok az değişir	Azalır
K_i	Kısalır	Artar	Artar	Ortadan kalkar
K_d	Çok az değişir	Azalır	Kısalır	Çok az değişir

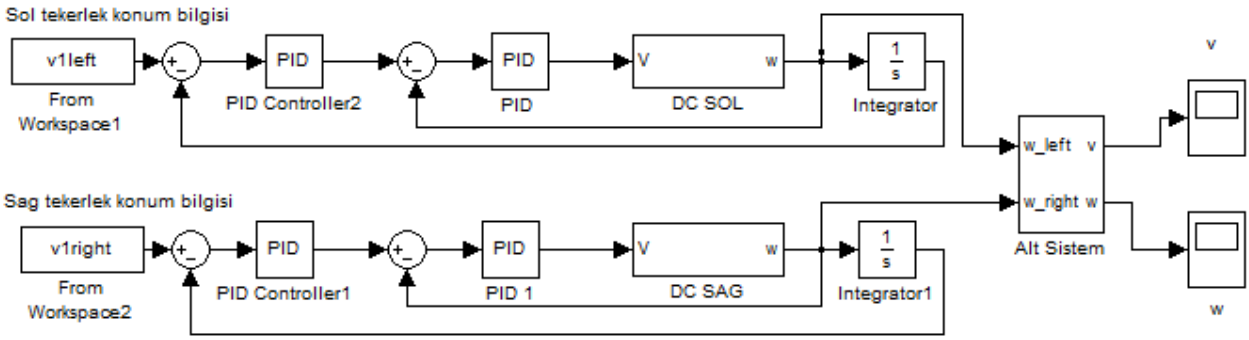


Şekil 4.16 Sistemin birim basamak girişine verdiği cevap.

4.3.2 PID Denetleyici ile Yörünge Takibi

PID denetleyicilerin en önemli avantajlarından biri, iki PID denetleyicinin birlikte kullanılmasıyla daha iyi bir performans elde edilmesidir. Bu tip kontrole kaskad PID kontrol adı verilir. Kaskad kontrolde bir PID denetleyici diğer bir PID denetleyicinin set değerini kontrol eder.

Yapılan tez çalışmasında iç içe iki PID denetleyici kullanılmıştır. Gösterim kolaylığı ve anlaşılırlık açısından sistem Matlab Simulink paket programı kullanılarak blok diyagramlar halinde modellenmiştir. Öncelikle; sistem giriş olarak A* algoritması tarafından oluşturulan yörünge bilgisini alır. Alınan yörünge bilgileri işlenerek motorlar için gerekli konum bilgisine dönüştürülür. Dıştaki PID denetleyici DC motorlar için gerekli konum bilgisini kontrol ederken içteki PID denetleyici ise hız bilgisini kontrol etmektedir.

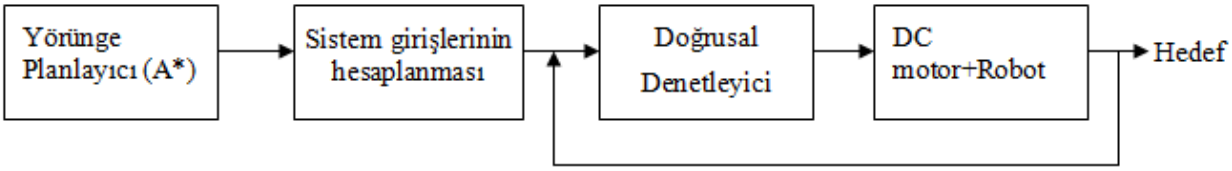


Şekil 4.17 DC motorun kaskad PID kontrolü.

Şekil 4.17' de Alt Sistem bloğunun çıkışı olan v, ω değişkenleri robotlar için denetim girdileridir. Gezgin robot sistemine ait çıkışlar ve grafikleri bir sonraki bölümde gösterilecektir.

5. SONUÇLAR

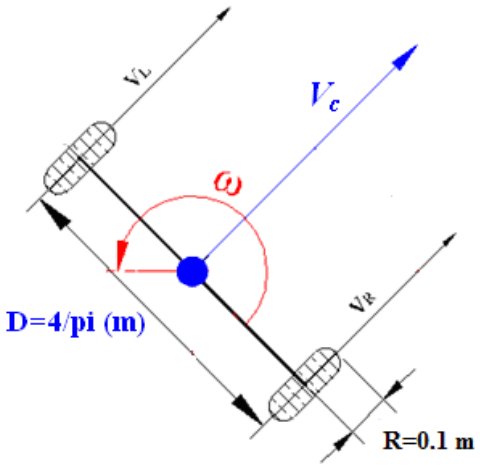
Bu bölümde benzetim çalışmalarında elde edilen sonuçlar verilmiş ve tartışılmıştır. DC motor ve gezgin robotlar için Bölüm 3.4' te oluşturulan Simulink modelleri ile Bölüm 4'te anlatılan yörünge planlama ve kontrol algoritmaları birleştirilmiştir. Şekil 5.1' de sistemi genel hatlarıyla tanımlayan bir blok diyagram gösterilmiştir.



Şekil 5.1 Sistemin blok diyagramı.

5.1 Benzetim Çalışmalarında Kullanılan Gezgin Robotun Parametreleri

Uygulama kolaylığı açısından gezgin robotun parametreleri Şekil 5.2' de gösterildiği gibi seçilmiştir. Ayrıca çoklu robot çalışmasında kullanılan robotlar özdeş olarak kabul edilmiştir.



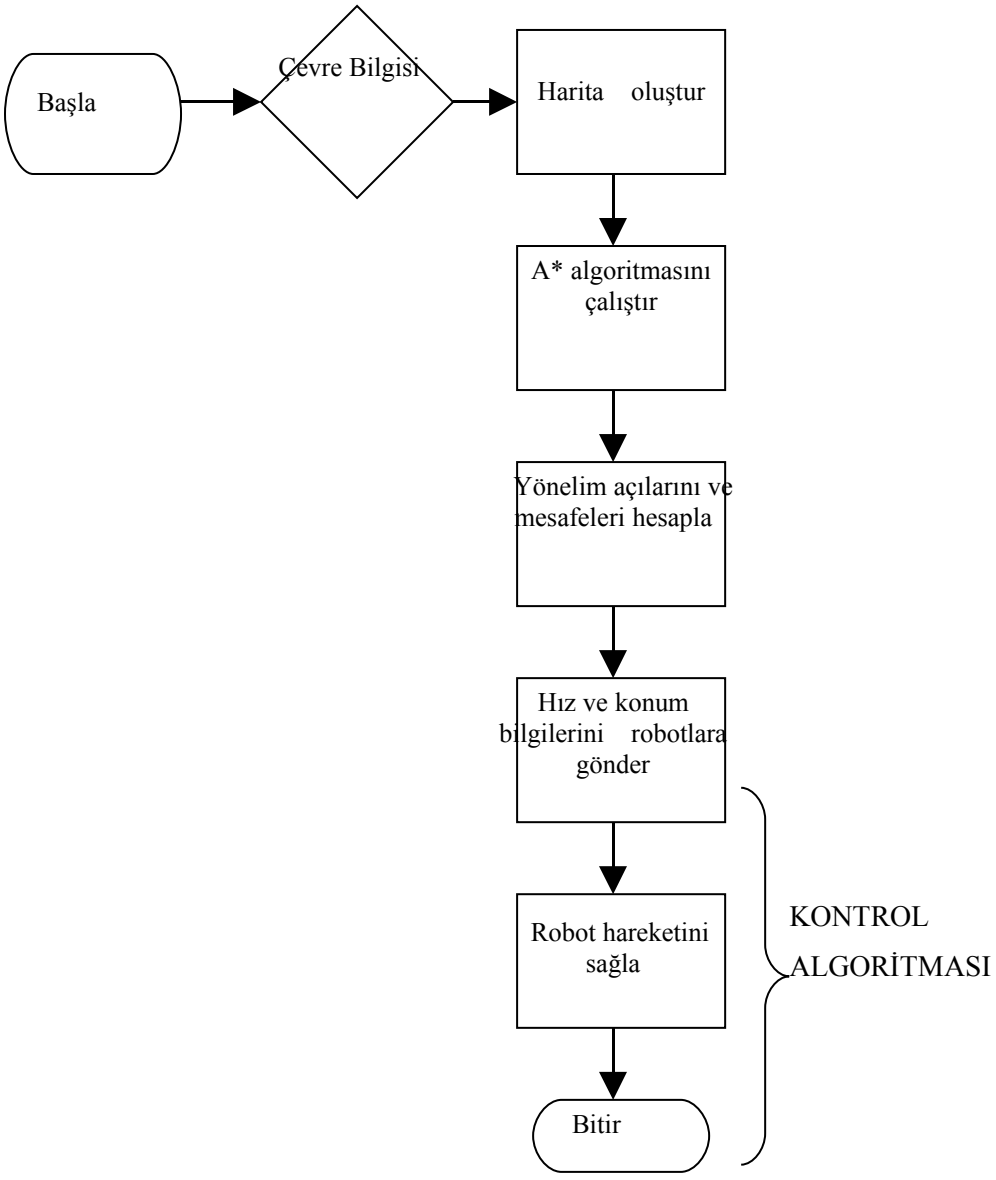
Şekil 5.2 Gezgin robotun parametreleri.

Tez çalışmasında kullanılan algoritmaya göre robotun iki temel hareket yaptığı düşünülmüştür. Bunlardan birincisi robotun sadece yönelim açısının değişmesi yani kendi ekseni etrafında dönmesidir. Buna göre robot kendi ekseni etrafında döndüğünde çapı (D) $4/\pi$ olan bir çemberi taramaktadır. Robotun kendi ekseni etrafında dönmesi için sol ve sağ tekerlekler birbirlerine ters yönde dönmelidir. Örneğin robotun sağ tekerleği ileri yönde dönerken sol tekerleği geri yönde dönerse robot çeyrek çember yayı yani 90°'lik bir açı taramış olur. İkinci temel hareket ise robotun düz gitmesidir. Robotun düz gidebilmesi içinse sol ve sağ tekerlekler aynı yönde dönmelidir. Örneğin her iki tekerlek 1 tam tur döndüğünde

$2 \cdot \pi \cdot R = 0.628$ m mesafe katetecektir.

5.2 Sistem Algoritmasının Tanımlanması

A* algoritması bütünlük sağladığı, optimal çözümü garantilediği ve düşük serbestlik dereceli sistemler için hızlı çözüm sağladığı için tez çalışmasında tercih edilmiştir (Çizelge 4.1). A* sonucu oluşturulan yörüngeler alt yörüngelere ayrılarak her biri için yönelim açıları ve mesafeler hesaplanır. Buna göre her bir robotun yönelim açısını değiştirmesi veya düz gitmesi için tekerleklerin ne kadar dönmesi gerektiği hesaplanır. Daha sonra Bölüm 4'te açıklanan PID denetleyici ile sistemin referans girişi takip etmesi sağlanır.



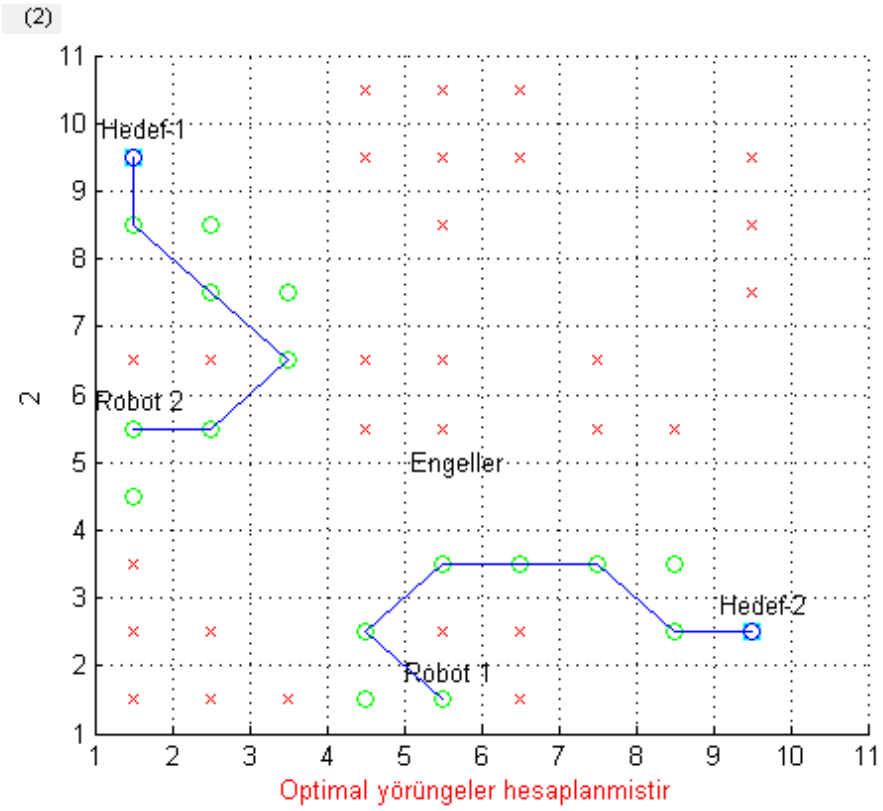
Şekil 5.3 Sistem algoritması akış diyagramı.

5.3 Benzetim Çalışmaları Sonuçları

Benzetim çalışmalarında 2 robot 2 hedef içeren bir uygulama gerçekleştirilmiştir. Yapılan uygulamada hedeflerin, engellerin ve robotların konumları rastgele seçilmiştir. Hedeflerin, engellerin ve robotların konumları kullanıcı tarafından problemin durumuna göre istenilen şekilde seçilebilir. Tez çalışmasında 2 durum çalışması yapılmıştır. Bunlar; robotların kendilerine yakın olan hedefe yönelmeleri ve robotların aynı hedefe yönelmeleridir.

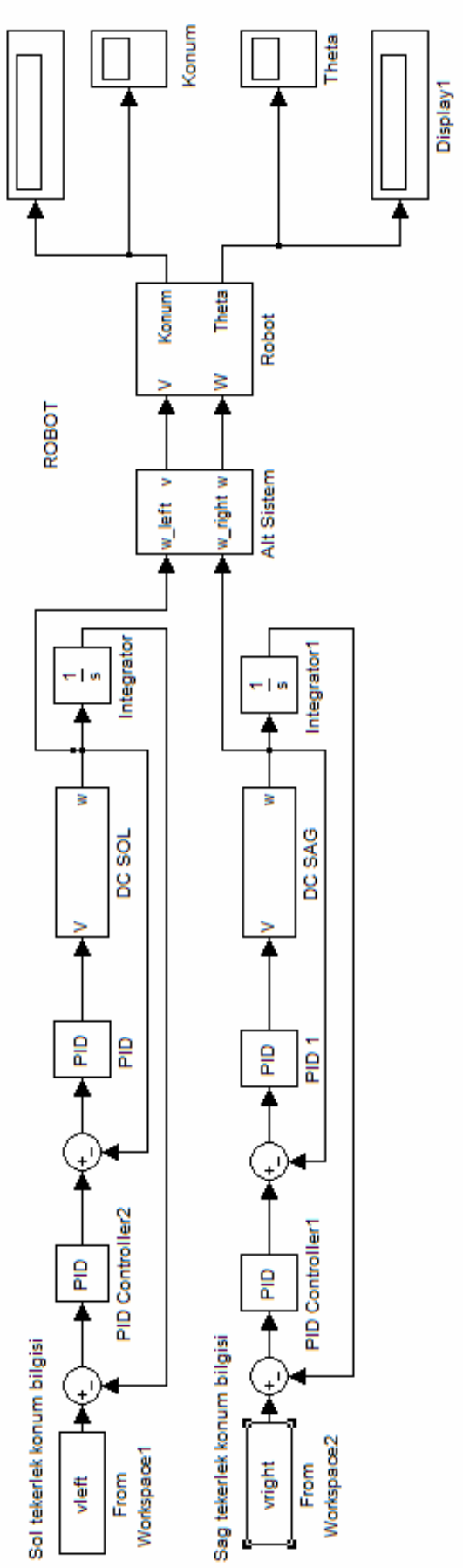
Benzetim çalışmaları verilmeden önce bazı kabullerin hatırlatılması gerekli görülmüştür. Buna göre robotlar ve robotların tekerleklerini süren DC motorlar tamamen özdeş kabul edilmiştir. Şekil 4.8'de de gösterilmiş olan çalışma ortamında hücreler arası yatay ve dikey geçişlerde karelerin arası 1 m olarak, çapraz geçişlerde ise 1.4 m olarak belirlenmiştir. Ayrıca robotların yönelim açıları başlangıçta 0°'dır. Saat yönü tersi açı değişimleri pozitif, saat yönündeki açı değişimleri negatif olarak tanımlanmıştır.

Durum 1: Robotların kendilerine yakın olan hedefe yönelmeleri



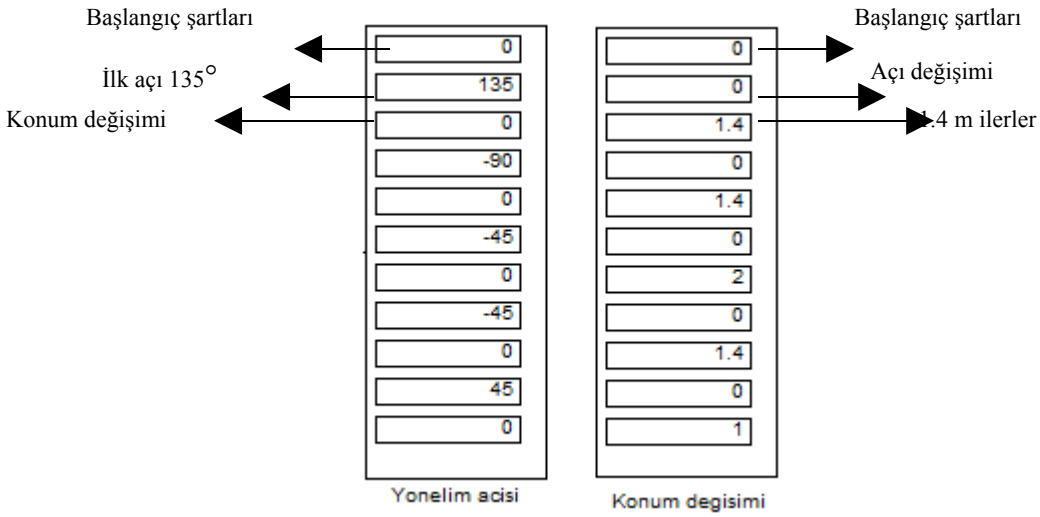
Şekil 5.4 Durum 1 için optimal yörüngelerin hesaplanması.

Sisteme giriş bilgisi olarak her bir aracın tekerlek tur sayısı (rad) verilmektedir. Bu nedenle A* algoritması sonucu oluşturulan yörüngelerden, robotların gitmesi gereken mesafe ya da açı hesaplanarak sisteme verilmektedir. Sistem çıkışları ise yönelim açısı değişimi, katedilen mesafeler ve tekerleklerin açısal hız değişimleridir.



Şekil 5.5 Sistemin simulink blok diyagramı.

Şekil 5.5'te tez çalışmasında kullanılan sistemin simulink blok diyagramı gösterilmiştir. Robotlar özdeş kabul edildiğinden tek bir blok diyagram kullanılmıştır. A* algoritması sonucu oluşturulan yörüngeler alt yörüngelere bölünmüştür. Bir başka deyişle ayrık planlama yapılmıştır. Her bir robot hedefe yönelirken bulunduğu konumda önce açısını değiştirecek ve daha sonra ilerleyecektir. Bu nedenle çıkışlardan da görüldüğü gibi robotlar açı değişimi ve ilerlemeyi aynı anda gerçekleştirmemektedir. Sistemin kontrolü için kullanılan PID denetleyicilerin kazançları istenilen ölçütleri (sıfır kararlı hal hatası, üst aşım yapmaması vb.) sağlayacak şekilde seçilmiştir. Robot 1 ve Robot 2 için yönelim açısı değişimleri ve konum değişimleri Şekil 5.6'daki gibidir. Daha anlaşılır olması açısından Robot 1'in açı ve konum değişimleri detaylı bir şekilde incelenmiştir.

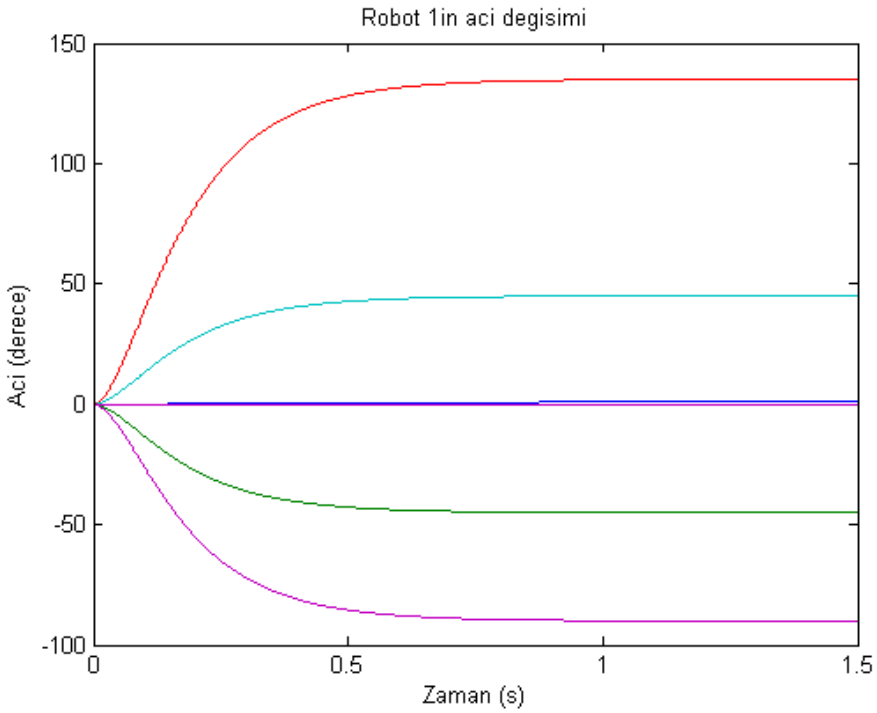


(a)

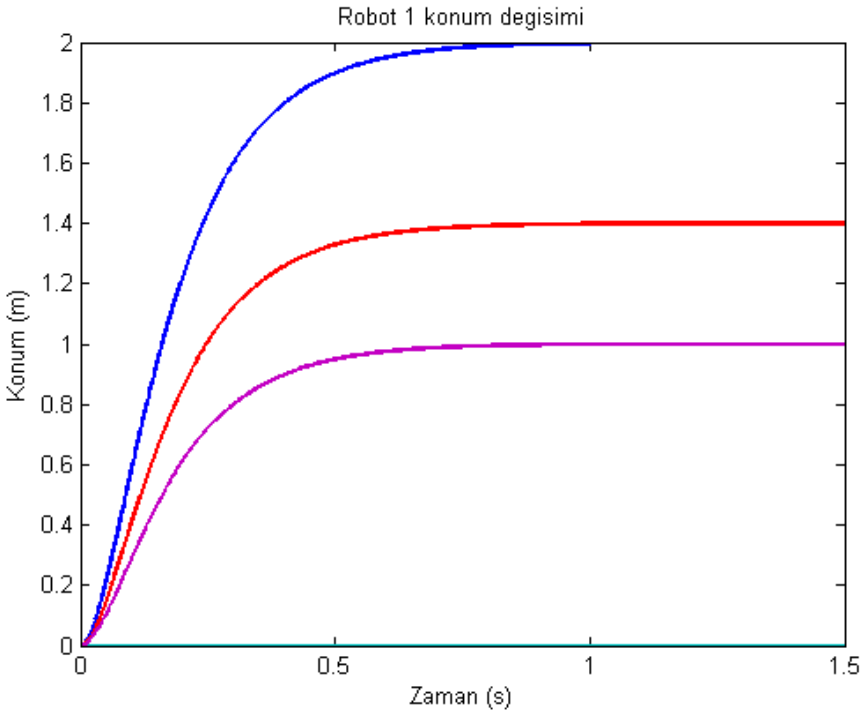
(b)

Şekil 5.6 (a) Robot 1'in çıkış değerleri , (b) Robot 2'nin çıkış değerleri.

Şekil 5.7 ve 5.9'da sırasıyla Robot 1 ve Robot 2'ye ait açı ve konum değişimleri çizdirilmiştir.

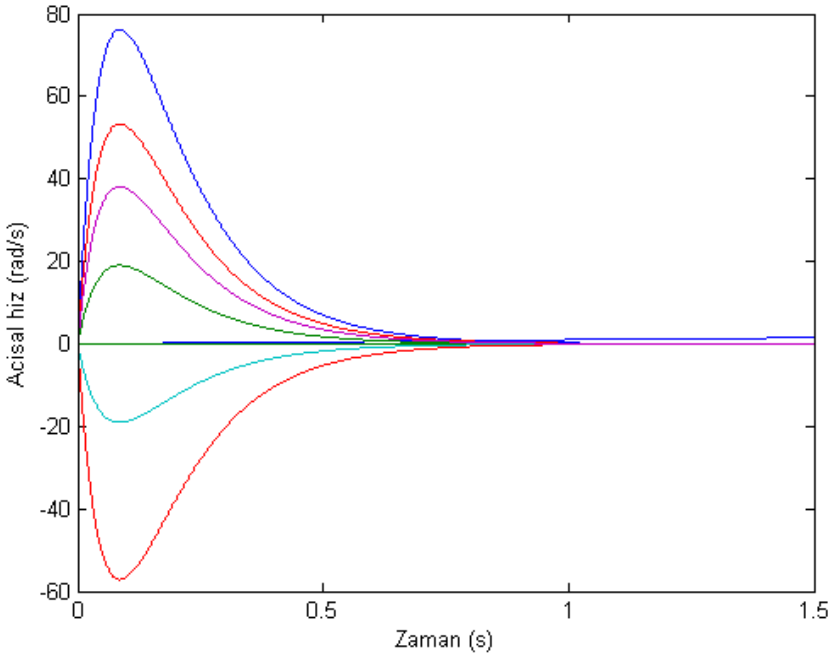


(a)

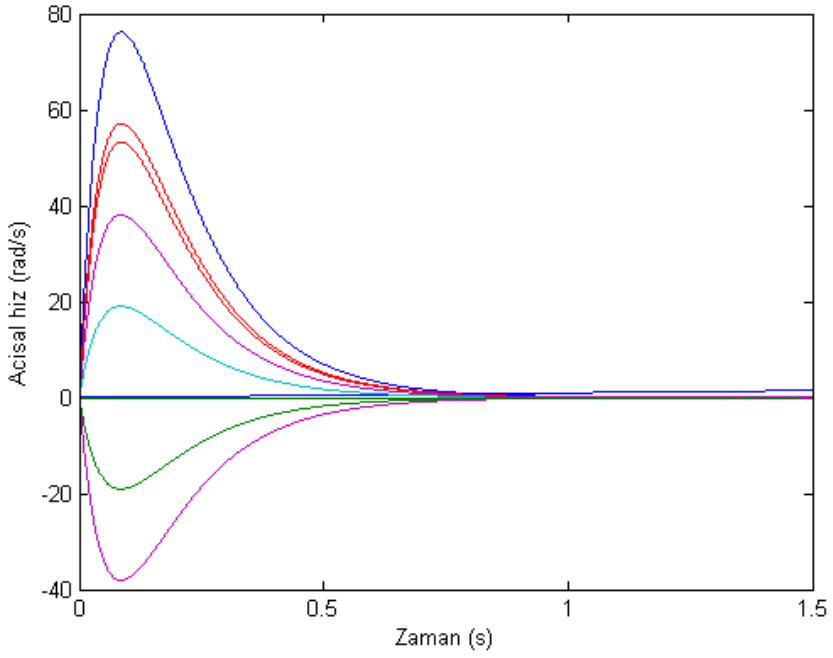


(b)

Şekil 5.7 (a) Robot 1 için açı değişimi , (b) Robot 1' in konum bilgileri.



(a)

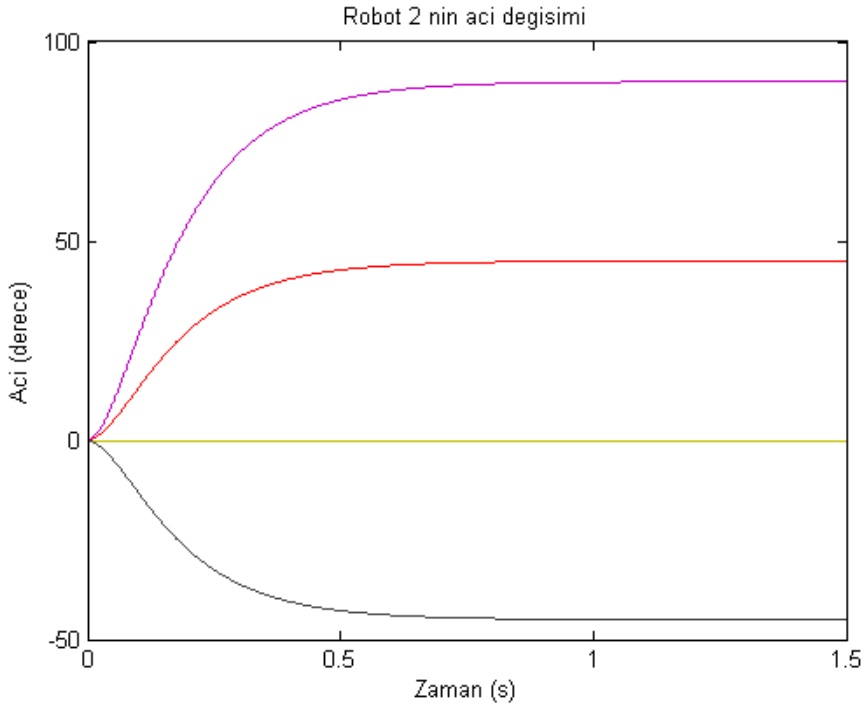


(b)

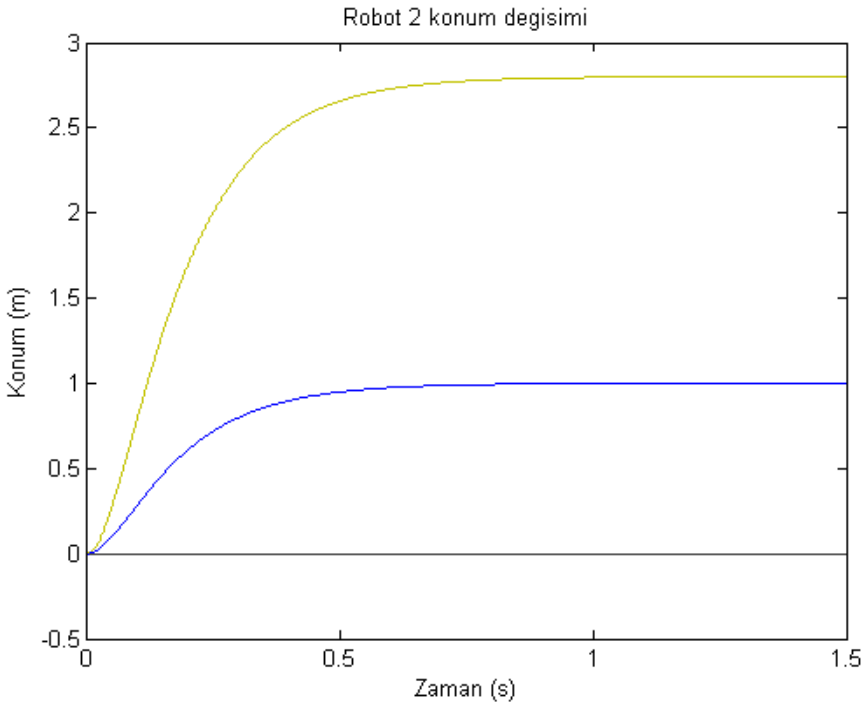
Şekil 5.8 (a) Robot 1'in sol tekerlek hızı , (b) Robot 1'in sağ tekerlek hızı.

Robot 1'in çıkış grafiklerinde görüldüğü gibi robotun kendi ekseni etrafında dönebilmesi yani yönelim açısını değiştirebilmesi için tekerleklerine birbirine eşit fakat ters işaretli hız bilgisi (ω_L=-ω_R) uygulanmıştır. Robotun düz gidebilmesi içinse birbirine eşit ve aynı işaretli hız bilgisi (ω_L=ω_R) uygulanmıştır.

Robot -2'nin çıkış grafikleri de benzer sonuçlar göstermektedir.

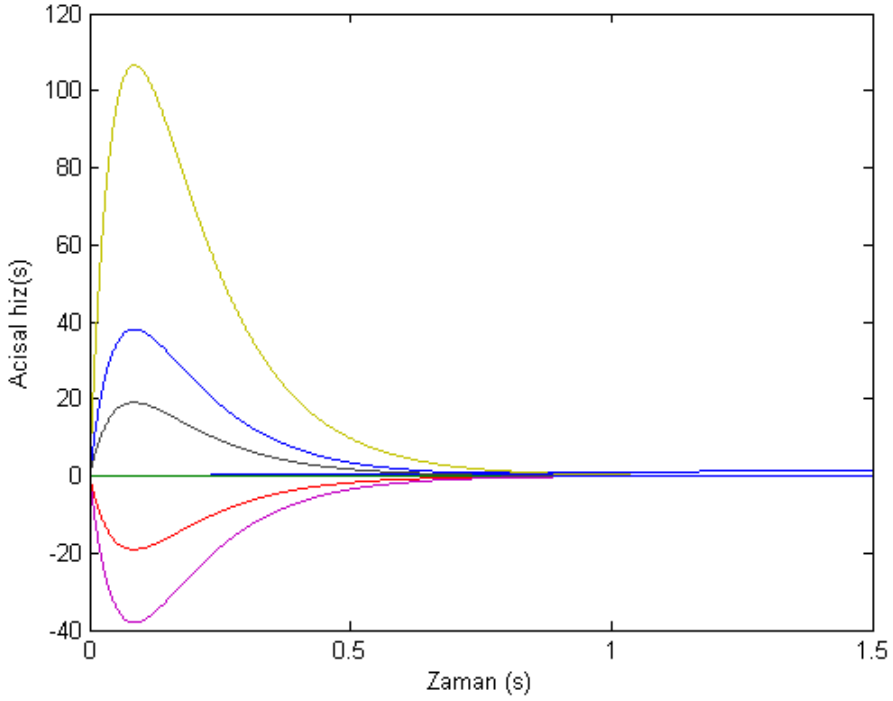


(a)

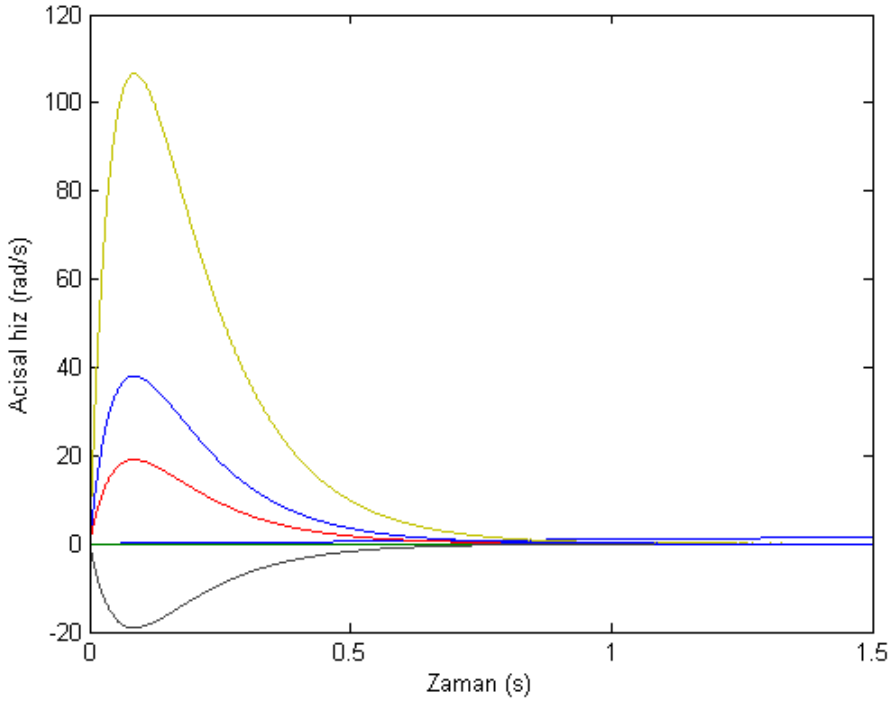


(b)

Şekil 5.9 (a) Robot-2 için açı değişimi , (b) Robot-2' nin konum bilgileri.



(a)

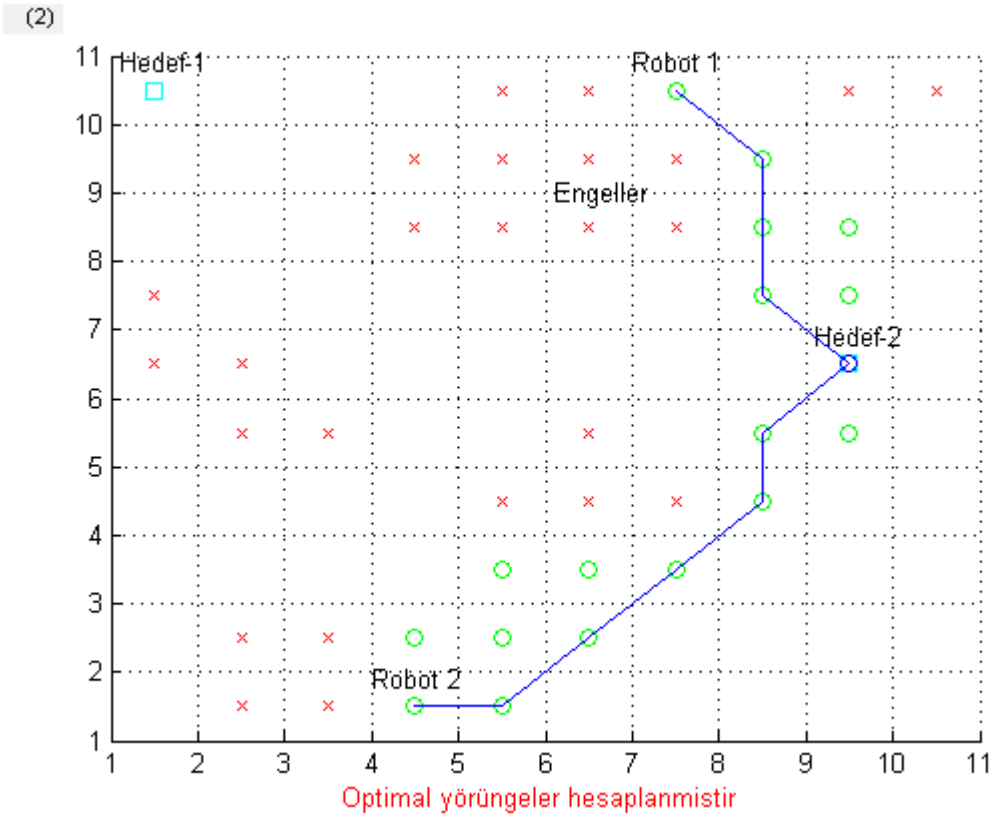


(b)

Şekil 5.10 (a) Robot-2'nin sol tekerlek hızı , (b) Robot-2' nin sağ tekerlek hızı.

Robotlar kullanılacakları problemin çözümüne göre aynı hedefe yönelebilirler. Bu durum Şekil 5.11'de gösterilmiştir.

Durum 2: Robotların kendilerine yakın olan (aynı) hedefe yönelmeleri



Şekil 5.11 Durum 2 için optimal yörüngelerin hesaplanması.

Şekil 5.12'de robotların çıkış değerleri gösterilmiştir. Robotların çıkış grafikleri benzer sonuçlar elde edildiğinden yeniden çizdirilmemiştir.

(a)

(b)

Şekil 5.12 (a) Robot 1'in çıkış değerleri , (b) Robot 2'nin çıkış değerleri.

5.4 Sonuçların Yorumlanması

Bu tezde çoklu robot çoklu hedef içeren kooperatif sistemler için yörünge izleme problemini çözecek bir algoritma oluşturulması amaçlanmıştır. Yörünge izleme probleminin çözümü için öncelikle Bölüm 2'de anlatılan klasik yörünge planlama tekniklerinden hücrelere bölme metodu uygulanmıştır. Hücrelere bölme metodu optimal sonucu garantilemediğinden optimal yörüngelerin oluşturulabilmesi için bilgili arama algoritmalarından A* algoritması kullanılmıştır (Çizelge 4.1). A* (A Yıldız) algoritması sonucu oluşturulan yörüngeler alt yörüngelere ayrılmış ve sistem için gerekli girişler hesaplanmıştır.

Daha sonra bu girişler robot sistemine verilmiş ve doğrusal bir denetleyici yardımıyla yörüngenin izlenmesi istenmiştir. Tez çalışmasında kullanılan robotlar diferansiyel sürüş sistemine sahiptir ve her bir tekerlek bağımsız DC motorlar tarafından sürülmektedir. Robot sisteminin kontrolünde PID denetleyici, sistem cevabında istenilen ölçütlerin sağlanması için yeterli bulunmuştur. Kaskad PID yapısına sahip denetleyici kullanıldığından sistem için ayrıca bir ivme değişkeninin tanımlanmasına gerek kalmamıştır.

1.5 s boyunca çalıştırılan simulasyonda sistemin 0.8 s'de istenilen referans değere oturduğu ve referansı başarıyla takip ettiği görülmüştür.

6. GELECEK ÇALIŞMALAR İÇİN ÖNERİLER

Bu tez çalışmasında elde edilen sonuçlar beraberinde birçok soruyu getirmekte ve değişik araştırma alanlarını ortaya çıkarmaktadır. Bu bölümde tezin geliştirilmesi ve sonraki araştırmalara yön vermesi amacıyla çeşitli öneriler verilmiştir.

Tez çalışması devamında yapılması düşünülen çalışmalar sırasıyla;

- Robot sisteminin ve çalışma ortamının dinamik modelinin kullanılması,
- Yörünge planlamada kullanılan A* algoritmasının dinamik ortamlarda çalışacak şekilde yeniden oluşturulması,
- Yörünge planlama algoritması sonucu elde edilen yörüngenin sürekli hale getirilmesi,
- Doğrusal denetleyici yerine doğrusal olmayan bir denetleyici kullanılması,
- Sistemin bozuculara karşı dayanıklılığının ve kararlılığının kontrol edilmesi,
- Çoklu robot (araç) sisteminin verilen bir görevi gerçekleştirecek şekilde gerçeklenmesi (donanımının kurulması) ve kooperatif kontrolünün sağlanması şeklinde özetlenebilir.

Sonraki araştırmalara yön vermesi amacıyla verilen öneriler 5 temel başlık altında toplanmıştır.

6.1 Sistem Modelinin Oluşturulması

Çoklu gezgin robotların yer aldığı birçok çalışmada olduğu gibi tez kapsamında da diferansiyel sürüş tekniğine sahip robotun kinematik modeli kullanılmıştır. Daha kararlı bir sistem yapısı elde edilmek istenirse robotun dinamik modeli kullanılabilir. Örneğin robotun engebeli ve virajlı bir çalışma ortamında yol aldığı düşünülürse dinamik modele sürtünme kuvvetleri, yol ve engel dinamikleri de eklenerek tam bir sistem modeli elde edilebilir. Bu durumda sistemin tamamı doğrusal olmayan bir sistem olarak ifade edilecektir.

6.2 Yörünge Planlamada Kullanılan Algoritmalar

Yörünge planlamada kullanılan A* algoritması bilinen ve tercih edilen bir yaklaşımdır. Bu yaklaşımın birçok avantajı bulunmakla birlikte dezavantajları da vardır. A* algoritması sonucu oluşturulan yörüngeler ayrık olarak düşünülebilir. Eğer sürekli bir yörünge elde edilmek istenirse B-spline eğrileri, polinomal eğriler, kübik spiraller vb. eğri ailelerinin kullanıldığı eğri uydurma teknikleri kullanılmalıdır. (Scheuer ve Xie, 1999)

Yörünge planlamada A* algoritması temel alınarak geliştirilen Theta*, D* gibi algoritmalar

da bulunmaktadır (Nash vd., 2007; Stentz, 1994). Bu algoritmalar çalışma ortamının bilinmediği problemlerde iyi bir alternatif olabilirler.

Değişik yaklaşımlardan biri de öğrenme algoritmalarının kullanılmasıdır. YSA ve Destek Vektör Makineleri gibi eğiticili öğrenme algoritmaları Bölüm 2.1.1'de anlatılan klasik yörünge planlama teknikleriyle birleştirilerek hibrid bir yapı oluşturulabilir.

6.3 Yörünge Takibi (Hareket Kontrolü)

Sistemin dinamik modeli oluşturulduktan sonra değişik tipte kontrol algoritmaları kullanılabilir ve performansları kıyaslanabilir. Yörünge takibi probleminin çözümünde en sık karşılaşılan denetleyiciler LQR denetleyici,doğrusal denetleyici, doğrusal olmayan denetleyici ve bulanık mantık denetleyicidir. Bu denetleyicilerin kazançları genetik algoritmalar yaklaşımı kullanılarak optimal bir şekilde ayarlanabilir.

6.4 Kooperatif Kontrol Sisteminin Gerçeklenmesi

Bir kooperatif kontrol sisteminin gerçeklenmesi için temel olarak çoklu dinamik birimlerin modellenmesi ve tasarlanması, birimler arası haberleşmenin sağlanacağı bir bilgi ağı oluşturulması ve sistemin kontrolü için dış merkezli ya da merkezi bir algoritma geliştirilmesi gerekmektedir. Kooperatif kontrolde çoklu dinamik birimler arası haberleşme ve haberleşmenin geliştirilen kontrol algoritmasıyla etkileşimi önemli bir konudur. Haberleşmenin geçici/kalıcı olarak kesilmesi veya birimin kaybı gibi durumlarda sistemin dayanıklılığı incelenebilir. Yapılan literatür taramasında birçok araştırma grubunun istenilen amaca yönelik kooperatif sistemler geliştirdiği görülmüştür. (Chung vd., 2002)

6.5 Sistem Ölçütleri

Geliştirilen kapalı çevrim sistemin bozuculara karşı dayanıklılığı (gürbüzlük) incelenmemiştir. H_2, H_∞ gibi yöntemler kullanılarak hem sistemin bozuculara karşı dayanıklılığı hem de sistem belirsizlikleri incelenebilir. Bunun dışında sistemin kararlılığının incelenmesi için evrensel bir metod olan Lyapunov metodu kullanılabilir. (Ackermann, 1993)

Anlatılan bölümlerde de görülebileceği gibi değişik yaklaşımların kullanılmasıyla birçok değişik problem oluşturulabilir.

KAYNAKLAR

Ackermann, J., (1993), Robust Control, Systems with Uncertain Physical Parameters, Springer-Verlag.

Aguiar, A. P., Cremean, L., Hespanha, J. P., (2003), "Position tracking for a nonlinear underactuated hovercraft: controller design and experimental results", Proceedings of the 42nd IEEE Conference on Decision and Control, 4(9):3858-3863.

Aksoy, R. ve Kurnaz, S. (2009), "Sektörel Grid Temelli Mihverleme Yöntemi ve Genetik Algoritmalarla İnsansız Kara Aracı Navigasyonu", Havacılık ve Uzay Teknolojileri Dergisi, 4(1):33-45.

Aoki, T., Matsuno, M., Suzuki, T. ve Okuma, S., (1994), "Motion planning for multiple obstacles avoidance of autonomous mobile robot using hierarchical fuzzy rules", IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems,265 –271.

Baxter, J. L., Burke, E. K. , Garibaldi, J. M. ve Norman, M., (2007), Autonomous Robots and Agents, Springer Berlin/Heidelberg, Chap. 2

Bloch, A. M., Reyhanoglu, M., McClamorch, H., (1992), "Control and stabilization of nonholonomic dynamic systems", IEEE Tran. On Automatic Control, 37(11):1746-1757.

Borkowski, A., Gnatowski, M. , Malec, J., (2001), "Mobile Robot Cooperation in Simple Environments". Proceedings of the Second International Workshop on robot Motion and Control, 109-114.

Bruce, J. ,Veloso, M., (2002), "Real-time randomized path planning for robot navigation", IEEE/RSJ International Conference on Intelligent Robots and System, 3: 2383-2388.

Canudas de Wit, C., Siciliano, B., Bastin, G., (1996), Theory of Robot Control, Springer-Verlag, London.

Choset, H., (2001), "Coverage for robotics – A survey of recent results", Annals of Mathematics and Artificial Intelligence, 31(1-4):113-126.

Chung, T., Cremean, L. , Dunbar, W. B. , Jin, Z., Moore, D. , Tiwari, A., Gogh, D. V., Waydo S., (2002), "A platform for cooperative and coordinated control of multiple vehicles: The Caltech multi-vehicle wireless testbed", Proc. of the 3rd Conference on Cooperative Control and Optimization.

Craig, J. J., (2004), Introduction to Robotics: Mechanics and Control, Pearson Prentice Hall, Upper Saddle River, NJ.

Davidor, Y., (1990), "Robot programming with a genetic algorithm" Proc. IEEE Int. Conf. on Computer Sys. & Soft. Eng, 186-191.

Davies, T. , Jnifene, A., (2007), "Path Planning and Trajectory Control of Collaborative Mobile Robots Using Hybrid Control Architecture", (2007), Journal of Systemics, Cybernetics and Informatics, 6(4):42-48.

D'Andrea, R., (2000), "Robot soccer: A platform for systems engineering.", Computers in Education Journal, 10(1):57-61.

Dudek, G., Jenkin, M. (2000), Computational Principles of Mobile Robots, Cambridge University Press,Newyork.

Fang, H. ve Wei, R., (2007), "Research of artificial potential field theory in motion of multi-robots.", Control Engineering of China, 14(2):115-117, 150.

Franchi, A., Freda, L., Oriolo, G., Vendittelli, M., (2007), "A Randomized Strategy for Cooperative Robot Exploration", 2007 IEEE International Conference on Robotics and Automation, 768 – 774.

Graham A. ve Buckingham R., (1993), "Real time collision avoidance of manipulators with multiple redundancy", Mechatronics, 3(1) :89-106.

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

Hedrick, J. K. , Tomizuka,M. and Varaiya,P., (1994), "Control issues in automated highway systems", IEEE Control Systems Magazine, 14(6):21-32.

Howard A., Seraji H., Werger B., (2005), "Global and regional path planners for integrated planning and navigation," Journal of Robotic Systems, 22(12): 767-778.

Hur, Y., Fierro, R., Lee, I., (2003) "Modeling Distributed Autonomous Robots using CHARON: Formation Control Case Study", Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 93-96.

Hwang, Y. K. ve Ahuja, N. (1992), "Gross motion planning-A survey", ACM Comp. Surv., 24(3): 219-291.

Innocenti, B., Lópeza, B. ve Salvia, J., (2007), "A multi-agent architecture with cooperative fuzzy control for a mobile robot", Robotics and Autonomous Systems,55(12):881-891.

Jamshidi, M. vd. (eds.), (1997), Applications of fuzzy logic: Towards high machine intelligence quotient systems, Prentice-Hall, NJ.

Katevas, N. I., Tzafestas, S. G., Pnevmatikatos, C. G., (1998), "The Approximate Cell Decomposition with Local Node Refinement Global Path Planning Method: Path Nodes Refinement and Curve Parametric Interpolation", Journal of Intelligent and Robotic Systems 22(3-4): 289-314.

Keil, J. M. ve Sack, J. R., (1985), "Minimum decomposition of polygonal objects" Comp. Geom., 197-216.

Khatib,O., (1986), "Real-time obstacle avoidance for manipulators and mobile robots", The International Journal of Robotics Research,5(1):90-98.

LaValle, S. M., (1998), "Rapidly-exploring random trees: A new tool for path planning.",In Technical Report No. 11, Computer Science Dept., Iowa State University.

LaValle, S. M. , (2006), Planning Algorithms, Cambridge University Press.

Li,K., D'Andrea,R., (2008), "Motion design and learning of autonomous robots based on primitives and heuristic cost-to-go',Robotics and Autonomous Systems,56(8):658-669.

Li, G. ve Jia, Q. ,(2008), "Cooperative receding horizon path planning of multiple robots by Genetic Algorithm", IEEE International Conference on Automation and Logistics,2449-2453.

Morin, P. ve Samson,C. (2004), "Trajectory tracking for non-holonomic vehicles:overview and case study", Proceedings of the Fourth International Workshop on Robot Motion and Control, 139-153.

Nash, A. , Daniel, K. , Koenig, S. ve Felner, A., (2007), "Theta*: Any-Angle Path Planning on Grids.", In Proceedings of the AAAI Conference on Artificial Intelligence, 1177-1183.

Parhia, D. R. , Pradhanb, S. K., Pandac, A. K. ve Beheraa, R. K., (2009), "The stable and precise motion control for multiple mobile robots", Applied Soft Computing, 9(2):477-487.

Parker, J. K., Khoogar, A. R. ve Goldberg, D. E., (1989), "Inverse kinematics of redundant robots using genetic algorithms" Proc. IEEE International Conference on Robotics and Automation, 1:271-276.

Pettersson, P. O. ve Doherty, P., (2006), "Probabilistic roadmap based path planning for an autonomous unmanned helicopter", Journal of Intelligent and Fuzzy Systems, 17 (4):395-405.

Reif, J. H. ve Wang,H., (1999), "Social potential fields: A distributed behavioral control for autonomous robots", Robotics and Autonomous Systems, 27(3):171-194.

Russell, S. ve Norving, P., (2003), Artificial Intelligence: A Modern Approach, 2nd Edition, Prentice Hall.

Ryan, A., Zennaro,M., Howell,A., Sengupta,R. ve Hedrick ,J. K.(2004) 'An Overview of Emerging Results in Cooperative UAV Control' 43rd IEEE Conference on Decision and Control December 14-17,2004.

Scheuer A. ve Xie M., (1999), "Continuous-Curvature Trajectory Planning for Manoeuvrable Non-Holonomic Robots", IEEE/RSJ Proc. of Int. Conference on Intelligent Robots and System, 3: 1675-1680.

Stentz, A., (1994), "Optimal and efficient path planning for partially-known environments," Proceedings of International Conference on Robotics and Automation, 4:3310–3317.

Zadeh, L.A., (1965), "Fuzzy sets", Information and Control, 8: 338-353.

Zelinsky, A., (1994), "Using path transforms to guide the search for find path in 2D", Int. Journal of Robotic Research, 13(4):315-325.

Zhang, P., Lü, T. ve Song, L., (2004) 'Soccer robot path planning based on the artificial potential field approach with simulated annealing', Robotica, 22(5):563-566. Cambridge University Press.

EKLER

Ek 1 Çoklu robot çoklu hedef içeren sistemler için geliştirilen A* Algoritması

Ek 1 Çoklu robot çoklu hedef içeren sistemler için geliştirilen A* Algoritması

```
clear all;clc;
%2 boyutlu harita (matris) dizisinin tanimlanmasi ve Baslangic degerlerinin atanmasi
MAX_X=10;
MAX_Y=10;
MAX_VAL=10;
OPEN_COUNT=0;
CLOSED_COUNT=0;
MAP=2*(ones(MAX_X,MAX_Y));
%elemanlari 2 rakamindan olusan 10 x 10 boyutlu bir matris
%Engellerin,Hedefin ve Robotun konumlarinin elde edilmesi
%Engeller=-1,Hedef=0,Robot=1,Bos uzay=2 sayıları ile tanimlanmistir.
%Robot baslangic, bitis ve engel pozisyonlarinin atanmasi
i=0;j=0;
x_val = 1;
y_val = 1;
n=0;%Engellerin sayisi (degisken tanitimi)
pause(5);
h = waitbar(0,'Program yuklenene kadar bekleyin...');
   for i=1:3000,
waitbar(i/3000)
   end
close(h)
prompt={'Lutfen robot sayisini giriniz...'};
defans={'2'};
fields={'value'};
info=inputdlg(prompt,'Robot #',1,defans);
  if ~isempty(info)
info=cell2struct(info,fields);
numberOfVehicle=str2num(info.value);
mycall = [ '  (' num2str(numberOfVehicle) ')'];
eval(mycall)
txt=uicontrol('Style','text','Position',[20 400 30 15],'String',mycall);
  end
axis([1 MAX_X+1 1 MAX_Y+1])
grid on;
```

```
hold on;
%%%%%%%%%%%%%%%%%%%%HEDEF-1%%%%%%%%%%%%%%%%%%%%
h=msgbox('Robotun gidecegi hedef kareyi sol mouse tusunu kullanarak secin ');
uiwait(h,5);
  if ishandle(h) == 1
delete(h);
  end
xlabel('Robotun gidecegi hedef kareyi sol mouse tusunu kullanarak secin','Color','black');
but=0;
%Sol mouse butonuna basilana kadar tekrarlar.
%Mouse ile harita uzerinden secimi saglar.
  while (but ~= 1)
%bu islem sol buton tiklanmadigi surece devam eder sol buton tiklandigi anda son tiklanan
%degerler alinip xval yval degiskenlerine atilir.
[xval,yval,but]=ginput(1); %bu while kod "but" 1'den farkli oldugu surece xval
%ve yval degiskenlerine figure'de tiklanan noktalarin kordinatlarini atar.
  end
%yuvarlatma islemleri
xval=floor(xval);
yval=floor(yval);
xTarget1=xval;%%Hedefin x koordinati
yTarget1=yval;%%Hedefin y koordinati
MAP(xval,yval)=0;%%Hedef=0 hedefin haritada tanimlanmasi
plot(xval+.5,yval+.5,'cs');
text(xval+.1,yval+.9,'Hedef-1')%hedef yazisinin sekle gore konumu
%%%%%%%%%%%%%%%%%%%%HEDEF-2%%%%%%%%%%%%%%%%%%%%
pause(0.25);
h=msgbox('Robotun gidecegi 2. hedefi kareyi sol mouse tusunu kullanarak secin ');
uiwait(h,5);
   if ishandle(h) == 1 %tiklandiysa kapa
delete(h);
   end
xlabel('Robotun gidecegi 2.hedef kareyi sol mouse tusunu kullanarak secin','Color','black');
but=0;
  while (but ~= 1)
%bu islem sol buton tiklanmadigi surece devam eder sol buton tiklandigi anda son tiklanan
```

```
%degerler alinip xval yval degiskenlerine atilir.
[xval,yval,but]=ginput(1); %bu while kod "but" 1'den farkli oldugu surece xval
%ve yval degiskenlerine figurde tiklanan noktalarin kordinatlarini atar.
  end
%yuvarlama islemleri
xval=floor(xval);
yval=floor(yval);
%Hedefin koordinati hedefin saklanacagi yere atiliyor
xTarget2=xval; % Hedef 2'nin x koordinati
yTarget2=yval; % Hedef 2'nin y koordinati
MAP(xval,yval)=0;
plot(xval+.5,yval+.5,'cs');%Hedef seklinin gridin icindeki yeri
text(xval+.1,yval+.9,'Hedef-2')%hedef yazisinin sekle gore konumu
%%%%%%%%%%%%%%%%%%%%%%ENGELLER%%%%%%%%%%%%%%%%%%%%
pause(0.2);%2.Hedef secildikten sonra yazi cikana kadar gecen sure 0.2 sn.
h=msgbox('Engelleri sol mouse tusu ile secin,son engeli secerken sag mouse tusunu kullanin');
xlabel('Engelleri sol mouse tusu ile secin,son engeli secerken sag mouse tusunu kullanin','Color','blue');
uiwait(h,10);
if ishandle(h) == 1 %tiklandiysa kapat
delete(h);
end
while (but == 1)
[xval,yval,but] = ginput(1);% sol mouse tiklandikca lokasyonlari al xval yvala at
xval=floor(xval);
yval=floor(yval);
MAP(xval,yval)=-1;
plot(xval+.5,yval+.5,'rx');
end
text(xval+.1,yval-1,'Engeller')
%%%%%%%%%%%%%%%%%%%%ROBOTLAR%%%%%%%%%%%%%%%%%%%%%%
vehicleCntr1=0;
vehicleCntr2=0;
VehicleMatrix=[];
for vehicleCntr1=1:numberOfVehicle
```

```
b = 'Robotun baslangic konumunu sol mouse tusunu kullanarak secin.';
xlabelStr=[num2str(vehicleCntr1),' ',b];
pause(0.5);%Son engel konduktan sonra yazinin cikmasi icin beklenecek sure
h=msgbox(xlabelStr);
uiwait(h,5);
if ishandle(h) == 1 %tiklandiysa kapat
delete(h);
end
xlabel(xlabelStr,'Color','black');
but=0;
while (but ~= 1) %Sol buton tiklanana kadar yap
[xval,yval,but]=ginput(1);
xval=floor(xval);
yval=floor(yval);
end
xStart=xval;%Starting Position
yStart=yval;%Starting Position
   MAP(xval,yval)=1;
   plot(xval+.5,yval+.5,'bo');
   a='Robot';
   carName=[a,' ',num2str(vehicleCntr1)];
   text(xval,yval+.9,carName);
   VehicleMatrix(vehicleCntr1,1)=vehicleCntr1;
   VehicleMatrix(vehicleCntr1,2)=xStart;
   VehicleMatrix(vehicleCntr1,3)=yStart;

end
 %Baslangic,Hedef ve engellerin haritada tanimlanmasi isleminin sonu
hk=0;
hl=0;
ArrayOfDistances=[];
for hk=1:numberOfVehicle
   OPEN=[];
   CLOSED=[];
    k=1;
for i=1:MAX_X
```

```
    for j=1:MAX_Y
        if(MAP(i,j) == -1)
            CLOSED(k,1)=i;
            CLOSED(k,2)=j;
            k=k+1;
        end
    end
end
CLOSED_COUNT=size(CLOSED,1);
%xNode ve yNode degiskenleri hangi aracta ise for dongusu onun koordinatlarini
%degiskenlere atar.
    xNode=VehicleMatrix(hk,2);
    yNode=VehicleMatrix(hk,3);
    OPEN_COUNT=1;
    path_cost=0;
    NoPath=1;
    %Herbir aracin herbir hedefe olan ayri ayri mesafesi hesaplaniyor
    for hl=1:2 %2 hedef oldugundan
    if hl==1
        goal_distance=distance(xNode,yNode,xTarget1,yTarget1);
    elseif hl==2
        goal_distance=distance(xNode,yNode,xTarget2,yTarget2);
    else
        goal_distance=distance(xNode,yNode,xTarget1,yTarget1);
    end
    ArrayOfDistances(hl,hk)=goal_distance;
    end
    %Array icinde array metodu kullaniliyor
OPEN(OPEN_COUNT,:)=insert_open(xNode,yNode,xNode,yNode,path_cost,goal_distance,
goal_distance);
    OPEN(OPEN_COUNT,1)=0;
    CLOSED_COUNT=CLOSED_COUNT+1;
    CLOSED(CLOSED_COUNT,1)=xNode;
    CLOSED(CLOSED_COUNT,2)=yNode;
    CELL_ARRAY(hk,1)={OPEN};
    CELL_ARRAY(hk,2)={CLOSED};
```

```
   CELL_ARRAY(hk,3)={OPEN_COUNT};
   CELL_ARRAY(hk,4)={CLOSED_COUNT};
end
Goal_Distances_Array=min(ArrayOfDistances,[],1);
gh=0;
for gh=1:numberOfVehicle
   OPEN=CELL_ARRAY{gh,1};
   CLOSED=CELL_ARRAY{gh,2};
   OPEN_COUNT=CELL_ARRAY{gh,3};
   CLOSED_COUNT=CELL_ARRAY{gh,4};
   xNode=VehicleMatrix(gh,2);
   yNode=VehicleMatrix(gh,3);
   xStart=xNode;
   yStart=yNode;
   goal_distance=Goal_Distances_Array(1,gh);
   if (ArrayOfDistances(1,gh)) < (ArrayOfDistances(2,gh))
      xTarget=xTarget1;
      yTarget=yTarget1;
   elseif (ArrayOfDistances(1,gh)) > (ArrayOfDistances(2,gh))
      xTarget=xTarget2;
      yTarget=yTarget2;
   else
      xTarget=xTarget1;
      yTarget=yTarget1;
   end
%%%%%%%%ALGORITMA BASLANGICI%%%%%%%%%%%%%%%%%%%%%
while((xNode ~= xTarget || yNode ~= yTarget) && NoPath == 1)
 plot(xNode+.5,yNode+.5,'go');
exp_array=expand_array(xNode,yNode,path_cost,xTarget,yTarget,CLOSED,MAX_X,MAX_
Y);
 exp_count=size(exp_array,1);
 %ACIK LISTENIN SONRAKİ DUGUMLERLE YENILENMESI
%ACIK LISTE YAPISI
 %--------------------------------------------------------------------
 %LISTEDE MI? 1/0 |X degiskeni |Y degiskeni |Ata X degiskeni |Ata Y degiskeni |h(n) |g(n)|
f(n)|
```

```
%--------------------------------------------------------------------------
%GENISLETILMIS DIZI YAPISI
%-------------------------------
%|X degiskeni |Y degiskeni ||h(n) |g(n)|f(n)|
%-------------------------------
for i=1:exp_count
  flag=0;
  for j=1:OPEN_COUNT
    if(exp_array(i,1) == OPEN(j,2) && exp_array(i,2) == OPEN(j,3) )
      OPEN(j,8)=min(OPEN(j,8),exp_array(i,5));
      if OPEN(j,8)== exp_array(i,5)
        OPEN(j,4)=xNode;
        OPEN(j,5)=yNode;
        OPEN(j,6)=exp_array(i,3);
        OPEN(j,7)=exp_array(i,4);
      end;%En kucuk fn kontrolu
      flag=1;
    end;%Dugum kontrolu sonu
  end;
  if flag == 0
    OPEN_COUNT = OPEN_COUNT+1;
OPEN(OPEN_COUNT,:)=insert_open(exp_array(i,1),exp_array(i,2),xNode,yNode,exp_array
(i,3),exp_array(i,4),exp_array(i,5));
   end;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%En kucuk fn e sahip dugum bulunur
 index_min_node = min_fn(OPEN,OPEN_COUNT,xTarget,yTarget);
 if (index_min_node ~= -1)
  xNode=OPEN(index_min_node,2);
  yNode=OPEN(index_min_node,3);
  path_cost=OPEN(index_min_node,6);
 %Dugumu kapali listeye yolla
 CLOSED_COUNT=CLOSED_COUNT+1;
 CLOSED(CLOSED_COUNT,1)=xNode;
 CLOSED(CLOSED_COUNT,2)=yNode;
```

```
 OPEN(index_min_node,1)=0;
 else
    NoPath=0;%Donguden cik
 end;
end;
i=size(CLOSED,1);
Optimal_path=[];
xval=CLOSED(i,1);
yval=CLOSED(i,2);
i=1;
Optimal_path(i,1)=xval;
Optimal_path(i,2)=yval;
i=i+1;
if ( (xval == xTarget) && (yval == yTarget))
   inode=0;
  parent_x=OPEN(node_index(OPEN,xval,yval),4);%dugumun indeksini verir
  parent_y=OPEN(node_index(OPEN,xval,yval),5);
  while( parent_x ~= xStart || parent_y ~= yStart)
        Optimal_path(i,1) = parent_x;
        Optimal_path(i,2) = parent_y;
        inode=node_index(OPEN,parent_x,parent_y);
        parent_x=OPEN(inode,4);
        parent_y=OPEN(inode,5);
        i=i+1;
   end;
 h=size(Optimal_path,1);
 Optimal_path(h+1,1)=xStart;
 Optimal_path(h+1,2)=yStart;
 j=size(Optimal_path,1);%Optimal yorungenin birinci kolonundaki eleman sayisi
if gh==1
   ylabel('1')
   OptimalArray1=Optimal_path;
elseif gh==2
   ylabel('2')
   OptimalArray2=Optimal_path;
end
```

```
%Optimal Yorungelerin Çizdirilmesi
 p=plot(Optimal_path(j,1)+.5,Optimal_path(j,2)+.5,'bo');
 j=j-1;
 for i=j:-1:1
  pause(0.5);
  set(p,'XData',Optimal_path(i,1)+.5,'YData',Optimal_path(i,2)+.5);
drawnow ;
  end;
   plot(Optimal_path(:,1)+.5,Optimal_path(:,2)+.5);
else
pause(1);
 h=msgbox('Hedefe giden bir yol bulunmuyor.!!','uyari');
 uiwait(h,5);
end
end
xlabel('Optimal yörüngeler hesaplanmistir ','Color','red');
%%%%%%%%%%%%%%%%%%%%ISLEMLER%%%%%%%%%%%%%%%%%%%%
 for nn=1:2
    switch nn
      case 1
      byt1=size(OptimalArray1,1);
      LocationArrayForVehicle1=[];
      for ctk=1:byt1
      LocationArrayForVehicle1(ctk,1)=OptimalArray1(byt1,1);
      LocationArrayForVehicle1(ctk,2)=OptimalArray1(byt1,2);
      byt1=byt1-1;
      end
      case 2
      byt2=size(OptimalArray2,1);
      LocationArrayForVehicle2=[];
      for ctk=1:byt2
      LocationArrayForVehicle2(ctk,1)=OptimalArray2(byt2,1);
      LocationArrayForVehicle2(ctk,2)=OptimalArray2(byt2,2);
      byt2=byt2-1;
      end
    end
```

```
 end
%%%%%%%%%%%%%%%%%%%%%%%%ARAC-1%%%%%%%%%%%%%%%%%%%%%%%%
angleOfV1=[];
angleOfV1(1,1)=0;
n1=size(LocationArrayForVehicle1,1);
%x1=1;
for i1=1:n1-1
      y=LocationArrayForVehicle1(i1+1,2)-LocationArrayForVehicle1(i1,2);
      x=LocationArrayForVehicle1(i1+1,1)-LocationArrayForVehicle1(i1,1);
      if (y<0 && x<0)||(x<0 && y==0)||(x<0 && y>0)
      angleOfV1(i1+1,1)=atan(y/x)+pi;
      angleOfV1(i1+1,2)=rad2deg(atan(y/x)+pi);
      elseif (x==0 && y<0)||(x>0 && y<0)
      angleOfV1(i1+1,1)=atan(y/x)+2*pi;
      angleOfV1(i1+1,2)=rad2deg(atan(y/x)+2*pi);
      else
      angleOfV1(i1+1,1)=atan(y/x);
      angleOfV1(i1+1,2)=rad2deg(atan(y/x));
      end
end
n1=size(angleOfV1,1);
angleOfV1(1,3)=0;
i1=2;
for i1=2:n1
   %onceki aci ile kiyaslama
   angleOfV1(i1,3)=angleOfV1(i1,2)-angleOfV1(i1-1,2);
   %normalizasyon
   if angleOfV1(i1,3)>180
      angleOfV1(i1,3)=angleOfV1(i1,3)-360;
   elseif angleOfV1(i1,3)<-180
      angleOfV1(i1,3)=angleOfV1(i1,3)+360;
   end
end
n1=size(angleOfV1,1);
x1=1;
k=1;
```

```
for xx=2:n1
        if ((angleOfV1(xx,2)==45) || (angleOfV1(xx,2)==-45) || (angleOfV1(xx,2)==135) || (angleOfV1(xx,2)==-135)|| (angleOfV1(xx,2)==225)|| (angleOfV1(xx,2)==-225)|| (angleOfV1(xx,2)==-315) || (angleOfV1(xx,2)==315))
       angleOfV1(xx,4)=2.226*2*pi;%1.4 metre
    else
       angleOfV1(xx,4)=1.59*2*pi; %1 grid ilerleme 1metre olarak atanir
    end
 end
sameAngle=0;
indis=[];
for xx=3:n1
    if(angleOfV1(xx,2)-angleOfV1(xx-1,2)==0)
       indis(xx-1,1)=xx-1;
       indis(xx,1)=xx;
       indis(xx-1,2)=2;
       indis(xx,2)=2;
       indis(xx-1,3)=angleOfV1(xx-1,2);
       indis(xx,3)=angleOfV1(xx,2);
    end
end
%%%%%%%%%%%%%%%%%%%%indis  duzeltme%%%%%%%%%%%%%%%%%%%%%%%%
sil=[];
n=size(indis,1);
for x=1:n
    if ((indis(x,1)== 0) && (indis(x,2)== 0) && (indis(x,3)== 0))
       sil(x,1)=x;
    end
end
sil(sil == 0) = [];
indis(sil(:,1),:)=[];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nIndis=size(indis,1);
counter=1;
indis(nIndis+1,3)=8;
nIndis=size(indis,1);
```

```
for x=1:nIndis-1
   if (indis(x,3)==indis(x+1,3)) && (indis(x,1)~=0)
      counter=counter+1;
      indis(x,4)=0;
   else
      indis(x,4)=counter*1;
      counter=1;
   end
end
nIndis=size(indis,1);
for ii=1:nIndis-1
angleOfV1(indis(ii,1),4)=angleOfV1(indis(ii,1),4)*indis(ii,4);
end
v1left=[];
v1left(1,1)=0;
v1right=[];
v1right(1,1)=0;
Even1=1;
   if (angleOfV1(2,2)==0) && (angleOfV1(2,3)==0)
      angleOfV1(2,3)=360;
   end
for d=2:size(angleOfV1,1)
   Even1d=(2*d)-2;
   Odd1d=(2*d)-1;
   v1left(Even1d+1,1)=angleOfV1(d,3);
   v1left(Odd1d+1,1)=angleOfV1(d,4);
   v1right(Even1d+1,1)=angleOfV1(d,3);
   v1right(Odd1d+1,1)=angleOfV1(d,4);
end
v1left(v1left == 0) = [];
v1left(v1left == 360) = 0;
v1right(v1right == 0) = [];
v1right(v1right == 360) = 0;
v1left=v1left';
v1right=v1right';
v1left = [0 0 v1left];
```

```
v1right = [0 0 v1right];
f1=size(v1left,2);
for xx=2:f1
   switch v1left(1,xx)
      case 0
         v1left(1,xx)=0;
         v1right(1,xx)=0;
      case 45
         v1left(1,xx)=-1.59*pi;
         v1right(1,xx)=1.59*pi;
      case -45
         v1left(1,xx)=1.59*pi;
         v1right(1,xx)=-1.59*pi;
      case 90
         v1left(1,xx)=-1.59*2*pi;
         v1right(1,xx)=1.59*2*pi;
      case -90
         v1left(1,xx)=1.59*2*pi;
         v1right(1,xx)=-1.59*2*pi;
      case 135
         v1left(1,xx)=-1.59*3*pi;
         v1right(1,xx)=1.59*3*pi;
      case -135
         v1left(1,xx)=1.59*3*pi;
         v1right(1,xx)=-1.59*3*pi;
      case 180
         v1left(1,xx)=-1.59*4*pi;
         v1right(1,xx)=1.59*4*pi;
      case -180
         v1left(1,xx)=1.59*4*pi;
         v1right(1,xx)=-1.59*4*pi;
   end
end
%%%%%%%%%%%%%%%%%ARAC-2%%%%%%%%%%%%%%%%%%%%%%%%%%
 angleOfV2=[];
angleOfV2(1,1)=0;
```

```
n2=size(LocationArrayForVehicle2,1);
x2=1;
for i2=1:n2-1
      y=LocationArrayForVehicle2(i2+1,2)-LocationArrayForVehicle2(i2,2);
      x=LocationArrayForVehicle2(i2+1,1)-LocationArrayForVehicle2(i2,1);
      if (y<0 && x<0)||(x<0 && y==0)||(x<0 && y>0)
      angleOfV2(i2+1,1)=atan(y/x)+pi;
      angleOfV2(i2+1,2)=rad2deg(atan(y/x)+pi);
      elseif (x==0 && y<0)||(x>0 && y<0)
      angleOfV2(i2+1,1)=atan(y/x)+2*pi;
      angleOfV2(i2+1,2)=rad2deg(atan(y/x)+2*pi);
      else
      angleOfV2(i2+1,1)=atan(y/x);
      angleOfV2(i2+1,2)=rad2deg(atan(y/x));
      end
end

n2=size(angleOfV2,1);
angleOfV2(1,3)=0;
i2=2;
for i2=2:n2
   angleOfV2(i2,3)=angleOfV2(i2,2)-angleOfV2(i2-1,2);
   if angleOfV2(i2,3)>180
      angleOfV2(i2,3)=angleOfV2(i2,3)-360;
   elseif angleOfV2(i2,3)<-180
      angleOfV2(i2,3)=angleOfV2(i2,3)+360;
   end
end
n2=size(angleOfV2,1);
x2=1;
k=1;
for xx=2:n2
       if ((angleOfV2(xx,2)==45) || (angleOfV2(xx,2)==-45) || (angleOfV2(xx,2)==135) ||
(angleOfV2(xx,2)==-135)|| (angleOfV2(xx,2)==225)|| (angleOfV2(xx,2)==-225)||
(angleOfV2(xx,2)==-315) || (angleOfV2(xx,2)==315))
      angleOfV2(xx,4)=2.226*2*pi;%1.4 metre
```

```
    else
        angleOfV2(xx,4)=1.59*2*pi; %1 grid ilerleme 1 metre
    end
 end
sameAngle=0;
indis=[];
for xx=3:n2
    if(angleOfV2(xx,2)-angleOfV2(xx-1,2)==0)
        indis(xx-1,1)=xx-1;
        indis(xx,1)=xx;

        indis(xx-1,2)=2;
        indis(xx,2)=2;

        indis(xx-1,3)=angleOfV2(xx-1,2);
        indis(xx,3)=angleOfV2(xx,2);
    end
end
%%%%%%%%%%%%%%%%%%%%indis duzeltme%%%%%%%%%%%%%%%%%%%%
sil=[];
n=size(indis,1);
for x=1:n
    if ((indis(x,1)== 0) && (indis(x,2)== 0) && (indis(x,3)== 0))
        sil(x,1)=x;
    end
end
sil(sil == 0) = [];
indis(sil(:,1),:)=[];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nIndis=size(indis,1);
counter=1;
indis(nIndis+1,3)=8;
nIndis=size(indis,1);
for x=1:nIndis-1
    if (indis(x,3)==indis(x+1,3)) && (indis(x,1)~=0)
        counter=counter+1;
```

```
        indis(x,4)=0;
    else
        indis(x,4)=counter*1;
        counter=1;
    end
end
 nIndis=size(indis,1);
for ii=1:nIndis-1
angleOfV2(indis(ii,1),4)=angleOfV2(indis(ii,1),4)*indis(ii,4);
end
v2left=[];
v2left(1,1)=0;
v2right=[];
v2right(1,1)=0;
Even1=1;
    if (angleOfV2(2,2)==0) && (angleOfV2(2,3)==0)
        angleOfV2(2,3)=360;
    end
for d=2:size(angleOfV2,1)
    Even1d=(2*d)-2;
    Odd1d=(2*d)-1;
    v2left(Even1d+1,1)=angleOfV2(d,3);
    v2left(Odd1d+1,1)=angleOfV2(d,4);
    v2right(Even1d+1,1)=angleOfV2(d,3);
    v2right(Odd1d+1,1)=angleOfV2(d,4);
end
v2left(v2left == 0) = [];
v2left(v2left == 360) = 0;
v2right(v2right == 0) = [];
v2right(v2right == 360) = 0;
v2left=v2left';
v2right=v2right';
v2left = [0 0 v2left];
v2right = [0 0 v2right];
f1=size(v2left,2);
for xx=2:f1
```

```
    switch v2left(1,xx)
        case 0
            v2left(1,xx)=0;
            v2right(1,xx)=0;
        case 45
            v2left(1,xx)=-1.59*pi;
            v2right(1,xx)=1.59*pi;
        case -45
            v2left(1,xx)=1.59*pi;
            v2right(1,xx)=-1.59*pi;
        case 90
            v2left(1,xx)=-1.59*2*pi;
            v2right(1,xx)=1.59*2*pi;
        case -90
            v2left(1,xx)=1.59*2*pi;
            v2right(1,xx)=-1.59*2*pi;
        case 135
            v2left(1,xx)=-1.59*3*pi;
            v2right(1,xx)=1.59*3*pi;
        case -135
            v2left(1,xx)=1.59*3*pi;
            v2right(1,xx)=-1.59*3*pi;
        case 180
            v2left(1,xx)=-1.59*4*pi;
            v2right(1,xx)=1.59*4*pi
        case -180
            v2left(1,xx)=1.59*4*pi;
            v2right(1,xx)=-1.59*4*pi;
    end
end
%Genisletilmis diziyi veren fonksiyon
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%expand_array%%%%%%%%%%%%%%%%%%%%%%
exp_array=expand_array(node_x,node_y,hn,xTarget,yTarget,CLOSED,MAX_X,MAX_Y)
 exp_array=[];
exp_count=1;
```

```
s_x=0;
s_y=0;
c1=1;
c2=size(CLOSED,1);%Kapali listedeki sifirlari iceren dizi
flag=0;%Dugumun cocuk dugum olup olmadigini belirler
for k= 1:-1:-1
   for j= 1:-1:-1
    if (k~=j || k~=0)  %Dugum kendisinin bir cocuk dugumu degil
       s_x = node_x+k;
       s_y = node_y+j;
       if( (s_x >0 && s_x <=MAX_X) && (s_y >0 && s_y <=MAX_Y))
flag=1;
c1=1;
         for c1=1:c2
       if(s_x == CLOSED(c1,1) && s_y == CLOSED(c1,2))
              flag=0;
       end;
         end;%Cocuk dugumun kapali listede olup olmadigini kontrol eden for dongusu
         if (flag == 1)
exp_array(exp_count,1) = s_x;
exp_array(exp_count,2) = s_y;
exp_array(exp_count,3) = hn+distance(node_x,node_y,s_x,s_y);%dugume gidisin maliyet
hesabi
exp_array(exp_count,4) = distance(xTarget,yTarget,s_x,s_y);%dugum ile hedef arasi mesafe
exp_array(exp_count,5) = exp_array(exp_count,3)+exp_array(exp_count,4);%fn
exp_count=exp_count+1;
          end
       end
       end
    end
   end
flag=0;
%%%%%%%%%%%%%%%%%%%%%insert_open%%%%%%%%%%%%%%%%%%%%
%Acik listeyi olusturan fonksiyon
function new_row = insert_open(xval,yval,parent_xval,parent_yval,hn,gn,fn)
%ACIK LISTE YAPISI
```

```
%--------------------------------------------------------------------------
 %LISTEDE MI? 1/0 |X degiskeni |Y degiskeni |Ata X degiskeni |Ata Y degiskeni |h(n) |g(n)| f(n)|
 %--------------------------------------------------------------------------
new_row=[1,8];
new_row(1,1)=1;
new_row(1,2)=xval;
new_row(1,3)=yval;
new_row(1,4)=parent_xval;
new_row(1,5)=parent_yval;
new_row(1,6)=hn;
new_row(1,7)=gn;
new_row(1,8)=fn;
%%%%%%%%%%%%%%%%%%%%distance%%%%%%%%%%%%%%%%%%%%
function dist = distance(x1,y1,x2,y2)
%Bu fonksiyon herhangi iki kartezyen koordinat arasi mesafeyi hesaplar
dist=sqrt((x1-x2)^2 + (y1-y2)^2);
%%%%%%%%%%%%%%%%%%%node_index%%%%%%%%%%%%%%%%%%%%
%Acik listedeki bir dugumun lokasyon indeksini veren fonksiyon
function n_index = node_index(OPEN,xval,yval)
i=1;
while(OPEN(i,2) ~= xval || OPEN(i,3) ~= yval )
   i=i+1;
end;
n_index=i;
%%%%%%%%%%%%%%%min_fn%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Dugumu minimum fn ile donduren Fonksiyon
% Bu fonksiyon Acik listeyi giris olarak alip en az maliyetli  dugumun indexini dondurur
function i_min = min_fn(OPEN,OPEN_COUNT,xTarget,yTarget)
temp_array=[];
j=1;
k=1;
flag=0;
goal_index=0;
temp_min=0;
```

```
for j=1:OPEN_COUNT
    if (OPEN(j,1)==1)
        temp_array(k,:)=[OPEN(j,:) j];
        if (OPEN(j,2)==xTarget && OPEN(j,3)==yTarget)
            flag=1;
            goal_index=j;%Hedef dugumun indeksinin saklanmasi
        end;
        k=k+1;
    end;
 end;%Acik listedeki butun dugumlerin toplanmasi
 if flag == 1
    i_min=goal_index;
 end
 %En kucuk dugumun indeksi
 if size(temp_array ~= 0)
 [min_fn,temp_min]=min(temp_array(:,8));
i_min=temp_array(temp_min,9);
else
    i_min=-1;
 end;
```

ÖZGEÇMİŞ

Doğum tarihi	11.06.1986	
Doğum yeri	Columbus/OH A.B.D.	
Lise	1996-2003	Edirne Anadolu Lisesi
Lisans	2003-2007	Yıldız Üniversitesi Mühendislik Fak. Elektrik Mühendisliği Bölümü (Yüksek Onur Öğrencisi, ikincilik derecesi)
Yüksek Lisans	2007-2009	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektrik Müh. Anabilim Dalı, Kontrol ve Otomasyon Programı