YAPAY POTANSİYEL FONKSİYONLAR VE PANEL METODU KULLANARAK ROBOT DİZİLİM KONTROLÜ VE SEYRÜSEFERİ

ABDEL-RAZZAK MERHEB

YÜKSEK LİSANS TEZİ

ELEKTRİK VE ELEKTRONİK MÜHENDİSLİĞİ

TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Ağustos 2010

ANKARA

Fen Bilimleri Enstitüsü onayı

_____

Prof. Dr. Ünver Kaynak

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Prof. Dr. M. Önder Efe

Anabilim Dalı Başkanı

Abdel-Razzak MERHEB tarafından hazırlanan YAPAY POTANSİYEL FONKSİYONLAR VE PANEL METODU KULLANARAK ROBOT DİZİLİM KONTROLÜ VE SEYRÜSEFERİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

_____

Doç. Dr. Veysel Gazi

Tez Danışmanı

_____

Yrd. Doç. Dr. Nilay Sezer Uzol

Tez İkinci Danışmanı

Tez Jüri Üyeleri

Başkan : Prof. Dr. M. Önder Efe _____

Üye : Doç. Dr. Veysel Gazi _____

Üye : Yrd. Doç. Dr. Nilay Sezer Uzol _____

Üye : Yrd. Doç. Dr. Ayşe Melda Yüksel _____

Üye : Yrd. Doç. Dr. Yiğit Taşçıoğlu _____

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Abdel-Razzak MERHEB

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri Enstitüsü**
Anabilim Dalı : **Elektrik ve Elektronik Mühendisliği**
Tez Danışmanı : **Doç. Dr. Veysel Gazi**
Tez Türü ve Tarihi : **Yüksek Lisans - Ağustos 2010**

ABDEL-RAZZAK MERHEB

YAPAY POTANSİYEL FONKSİYONLAR VE PANEL METODU KULLANARAK ROBOT DİZİLİM KONTROLÜ VE SEYRÜSEFERİ

ÖZET

Bu tez çalışmasında, potansiyel akış hesaplamaları robot sürülerine çarpışmasız seyrüsefer algoritması geliştirmek için kullanılmıştır. Akışkanlar mekaniğinde iyi bilinen, sıkıştırılmayan viskoz olmayan potansiyel akış genel denklemlerini akışın yüzeyine paralel olma koşulunu sağlayarak çözen panel metodu kullanılarak, sürü erkinleri için hedefe giden güvenli yollar bulunmaktadır. Ayrıca, yapay potansiyel fonksiyonları robot sürüsünün önceden belirlenmiş bir geometrik şekli koruması için kullanılmaktadır. İki tür algoritma geliştirilmiştir, çevrimdışı ve çevrimiçi algoritmalar. Çevrimdışı algoritmada, ortamın haritasının önceden bilindiği kabul ederek geliştirilmiştir ve deneyler karmaşık engeller içeren ortamda e-puck küçük robotlar kullanılarak yapılmıştır. Çevrimiçi algoritmada ortamda bulunan engelleri algılamak için lazer algılayıcı ile donatılmış üç adet kheperaIII gezgin robot kullanılmıştır. Daha sonra, karmaşık şekilli rijit cisimler etrafındaki potansiyel akış tabanlı çizgilerini hesaplamak için gerçek zamanlı panel metodu kullanılarak, robotları hedef noktasına götüren güvenli yolları bulmuştur. Robot sürüsünün önceden belirlenmiş bir geometrik şekli koruması için yapay potansiyel fonksiyonlar kullanılmıştır. Potansiyel akış algoritmalar hedefe doğru düzgün yollar ile seyrüseferi sağlar. Çevrimdışı algoritma ortamda olan değişiklikleri algılayabildiği için gerçek zamanda değişen ortamlarda da kullanılabilir. Son olarak, algoritmadaki hataları azaltmak için bağıl konumlandırmaya dayalı bir yöntem kullanılmıştır.

Anahtar Kelimeler: Robot Seyrüseferi, Sürü Robot Sistemleri, Potansiyel Akış, Panel Metodu, Yapay Potansiyel Fonksiyonlar, Lazer Algılayıcısı, Göreceli Konumlandırma

University	**: TOBB University of Economics and Technology**
Institute	**: Institute of Natural and Applied Sciences**
Science Programme	**: Electrical and Electronics Engineering**
Supervisor	**: Associate Professor Veysel Gazi**
Degree Awarded and Date	**: M.S. - August 2010**

ABDEL-RAZZAK MERHEB

ROBOT FORMATION CONTROL AND NAVIGATION USING ARTIFICIAL POTENTIAL FUNCTIONS AND PANEL METHOD

ABSTRACT

In this thesis, potential flow calculations are used to develop collision free navigation algorithms for a robot swarm. The well known panel method in fluid mechanics is used to solve the governing equations for inviscid incompressible potential flow around rigid objects with tangency boundary conditions, providing the robots with safe trajectories to the target. In addition, artificial potential functions are used to force the swarm to achieve a predefined geometrical shape. Two kinds of algorithms are developed, offline and online algorithms. In the off-line algorithm the environment is assumed to be previously known, and experimental results are obtained using e-puck mini-robots navigating in an environment with complex shaped obstacles. In the online algorithm, three KheperaIII robots are equiped with laser scanners which are used to detect the obstacles in the environment. Then, a real-time panel method is used to calculate the streamlines of the potential flow around the complex shaped rigid objects, providing the robots with safe trajectories to the target. The swarm of robots is also forced to keep a desired formation during navigation using potential functions. Potential flow algorithms provide navigation with smooth paths to the target. The algorithm can be used in dynamic environments in real-time as the changes in the medium are detected. Finally, a method based on relative position sensing is developed to minimize the error in the algorithms.

Keywords: Robot Navigation, Swarm Systems, Potential Flow, Panel Method, Artificial Potential Functions, Lazer Sensor, Relative Positioning

TEŞEKKÜR

Bana doğru yolu gösteren ve beni güzel şekilde yetiştirene; çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren değerli hocalarıma Doç. Dr. Veysel GAZİ ve Yrd. Doç. Dr. Nilay Sezer Uzol'a, katkılarından dolayı Prof. Dr. M. Önder Efe, Yrd. Doç. Dr. Yiğit Taşçooğlu, Yrd. Doç. Dr. Ayşe Melda Yüksel'e TOBB Ekonomi ve Teknoloji Üniversitesi Elektrik ve Elektronik Mühendisliği Bölümü öğretim üyelerine,

Sürü Sistemler Araştırma Laboratuvarı'ndaki çalışma arkadaşlarım olan Yunus Ataş, Engin Karataş, Salih Burak Akat, Ömer Çayırpunar, Mirbek Turduev, Murat Kırtay, Esma Gül ve Sabahat Duran'a,

Beni destekleyen İlim Yayma Cemiyeti ve Mehmet Akif Ersoy yurdun sorumluları, özellikle Halim Altunkal hoca, Selim Cerrah hoca, Uğur Elaman hoca ve müdür Hasan keklik abiye, yurttaki arkadaşlarım Ebubekir Sıddık Solmaz, Hukuk Ünaldı, Üsame Ölmez, Yusuf Ünal, Abdullah Metin ve tüm arkadaşlara, beni gurbette destekleyen kardeşlerim Avukat Ebubekir Dere, Yusuf Erdem Dere ve Mühendise Hatice Pak Dere'ye,

Ve her zaman beni destekleyen çok sevdiğim aileme ve eşime teşekkürlerimi sunarım. Ayrıca Andaç Töre Şamiloğlu ve Engin Karataş arkadaşlarıma deney sonuçları çıkarmasında yardım ettikleri için, Ömer Çayırpunar arkadaşıma e-puck'larda yardım ettiği için ve Yunus Ataş arkadaşıma KheperaIII kullanımında yadım ettiği için teşekkür ederim.

Bu çalışma Avrupa Komisyonu tarafından 045269 sözleşme numaralı 6. Çerçeve Programı özel amaçlı araştırma projesi kapsamında desteklenmiştir.

İÇİNDEKİLER

Sayfa

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

BÖLÜM 1

1. GİRİŞ

1.1. Sürü Sistemler

Sürü zekası kavramı hayvan ve böcek topluluklarındaki toplumsal davranış alanında yapılan keşifler ve araştırmalardan esinlenilmiştir. Yaklaşık elli sene önce, biyologlar böcek, balık, kuş ve memeliler toplumlarından yeni zeka türü çıkabileceğini farkettiler. Sürü sistemlerin karar verme özelliği bilim adamların dikkatini çekmiştir. Bu özellik, basit ve ilkel erkinlerden oluşan sürülerin büyük yapay zeka sahibi olan bir erkinin sergilediği zeka davranışlarını sergilebildiğini göstermiştir. Adi erkinlerin topluluğu karmaşık kararları verebilme kabiliyetini göstermiştir [1].

Toplumsal davranışın ilk ciddi teorik açıklaması fransız biyolog Pierre-Paul Grassé tarafından sunulmuştur. 1950'de Grassé incelediği termit kolonilerinde karıncaların arasındaki dolaylı eşgüdüm[1] ve etkilerini anlatmak için "stigmetri" kavramını ortaya koydu. Bir karınca etkisiyle ortamda yapılan bir değişim, aynı ya da başka bir karınca vasıtasıyla yeni bir değişime yol açar; böylece ardarda gelen etkiler ve değişimler görünüşe göre sistematik bir aktivite sergilemiş olacaktır. Grassé'nin sözleriyle, "stigmetri, işçi karıncaların kendi aktiviteleriyle kendi kendilerini uyarmalarıdır". Stigmetri çok basit, belleksiz, zeki olmayan ve de birbirinden haberi olmadan erkinlerden oluşan bir öz-organizasiyon[2] şeklidir. Dahası da, direkt iletişim, denetim ya da planlama olmadan karışık ve zeki vasfı taşıyabilen (yapılar) aktiviteler yaratılmaktadır [1].

1990'da Jean-Louis Deneubourg ve arkadaşları bir karınca topluluğunun yuva ve yem bölgesini bağlayan iki farklı uzunluktaki yolların arasından en kısa yolun seçilme olasılığının deneylerle büyük olduğunu gösterdi. Bu davranış, Deneubourg tarafından olasılık modeli şeklinde açıklanmıştır. Bu olasılık modelindeki her karınca önceki karıncaların tarafından bırakılan feremon yoğunluğunu kullanarak rastgele karar verir. Karar verme algoritması çok basit olmasına rağmen karınca topluluğun davranışı "zeki" vasfını taşır. Deney sonunda karıncalar yeme giden en kısa yolu bulmuştur [2].

[1] ing.:coordination
[2] ing.:self-organization

1.2. Çok Erkinli Sistemler

Çok erkinli sistemler[3] birbirleri ile etkileşen elemanlardan oluşur. Her erkinin iki özelliği vardır: birincisi her erkin özerktir, ya da tasarının arkasındaki amacına ulaşmak için gereken kararları alma ve eylem kabiliyeti vardır. İkincisi ise, her erkin gruptaki diğer erkinleri etkileyebilir ve onlardan etkilenebilir. Bu etkileşme sadece bilgi paylaşımından ibaret değil, üstelik toplumsal aktiviteler (yardımlaşma, koordinasyon ve muzakere v.s.) gibi davranışlar da gözlemlenebilir [3].

Yakın zamana kadar çok erkinli sistemlerin yapay zekanın bir alt alanı olduğu kabul ediliyordu. Dağıtık yapay zekanın[4] amacı çok erkinli sistemleri inceleme, modelleme ve uygulama bilimi olduğu tanımlanıyor. Ancak son zamanlarda, çok erkinli sistemlerde araştırma yapanlar, yapaynın zeka çok erkinli sistemlerin bir parçası olduğunu iddia ediyorlar. Dahası, Stuart Russell ve Peter Norvig yapay zekanın asıl hedefinin zeki bir erkini üretmek olduğunu savunuyorlar [3], [4], [5].

Çok erkinli sistemler zeki sistemlere doğal ve tabii özellikler katar. Zeka ve etkileşim ayrılmaz ikilidir, ve çok erkinli sistemler bunu yansıtmaktadır. İnsanlar gibi doğal, zeki, tek yaşamazlar; en azında kendileri ve diğer zeki sistemlerin yaşadıkları ortamın bir parçasını oluştururlar. İnsanlar değişik şekillerde ve düzeylerde etkileşirler, ve insanların başarısının sırrı bu etkileşimdir [3], [4].

Çok erkinli sistemler ilgi çekici özellikler sunmaktadır; Hız ve verimlilik, erkinlerin eşzamanlı ve paralel bir şekilde çalışma imkanları vardır; böylece sistemin hızı erkinlerin sayısına doğrudan bağlı olur. Gürbüzlük ve güvenilirlik, bir ya da birden fazla erkinin bozulması sistemin çöküşüne yol açmayabilir. Sistemde bulunan diğer erkinler bozulan erkinlerin işini görebilir. Ölçeklenirlik[5] ve esneklik: sisteme erkin ekleyerek daha büyük problemler çözmeye ayarlanabilir. Eklenen erkinler diğer erkinlerin işini aksatmaz. Fiyat, basit ve ucuz birimlerden oluştuğu için bu sistemler, merkezileştirilmiş sistemlerden[6] çok daha hesaplı olabilir. Geliştirilebilme ve yeniden kullanılırlık[7], erkinler tek tek donanımsal ve yazılımsal anlamda geliştirildiğinde sistemde değişiklik yapmadan çalışır, sistem kolay bir şekilde düzeltilip test edilebilir, üstelik erkinleri farklı ve değişik uygulamalarda kullanılabilir [3], [4].

[3] ing.:Multi agent systems
[4] ing.:Distributed Artificial Intelligence
[5] ing.:scalability
[6] ing.:centralized system
[7] ing.:Development and Reusability

1.3. Gezgin Robotlar ve Seyrüsefer

Gezgin robot, bulunduğu ortamda hareket edebilen özerk makinedir. Alt seviyeli robotlar, bulundukları ortamlarda basit bir etki/tepki davranış ve rastgele arama metodları kullanacak şekilde gezerler. Öte yandan daha gelişmiş işler yapan robotlar, işlerini tamamlayabilmek için bir nevi seyrüsefer ve planlama ister [6].

Seyrüsefer, robot alanında gerçekleşmesi zor uygulamalardan biridir. Seyrüsefer yapabilmek için bir robotun çevreyi algılaması, güzergah planlaması, ve karar verebilmesi gereklidir. Alt seviyeli etki/tepki robotları engellere takılmadan ve çarpmadan gezebilir, fakat seyrüsefer robotun gitmek istediği yere en kısa yoldan, en düşük maliyetle ve emniyetli bir şekilde nasıl gidebileceği planlanmasını içermelidir. Seyrüsefer algoritmaları dört soruda toparlanabilir: "nereye gidiyorum?", "oraya gitmek için en iyi yol hangisidir?", "önceden neredeydim ve şimdi neredeyim?". Seyrüseferde iki kategori vardır: topolojik ya da niteleyici[8] ve metrik ya da nicel[9] seyrüsefer. Topolojik seyrüsefer güzergah üzerinde ya da ortamda bulunan özel işaretleri[10] ya da farklı özellikleri taşıyan noktaları kullanarak robota yol buldurmaya çalışır. Topolojik seyrüsefer robotu hedefe yaklaştıracak ve belli işaretli bir yol gösterir. Metrik seyrüsefer robotun bir sonraki adımı ya da konumu hedefe yakın olan ve aynı anda maliyeti en düşük olan noktayı belirler [7].

1.4. Tezin Amacı ve Konusu

Bu tez çalışmasında, akışkanlar mekaniği kullanılarak gezen bir robotun güzergah belirleme yöntemi geliştirilmiştir. Potansiyel akış teorisine dayalı sayısal bir teknik olan ve aerodinamik analizlerde geniş çapta kullanılan "panel metodu" tekniği geliştirilen yöntemin temelini oluşturmaktadır. Panel metodu engel içeren ortama uygulanarak elde edilen akış çizgileri robotun güzergahları olarak kullanılmıştır. Akış çizgilerin özellikleri gereği otomatik olarak robot ya da robot sürüsü elemanlarının birbirleri ile ve etraftaki engeller ile çarpışmadan hedef noktasına gitmeleri sağlanmiştir.

Robot sürüsünün önceden belirlenmiş bir dizilimi sağlaması için yapay potansiyel fonksiyonlarda kullanılmıştır. Erkinlerin arasındaki mesafeler ve önceden belirlenmiş sürünün istenen geometrisini kullanarak erkinlerin birbirine olan itim-çekim kuvvetleri

[8] ing.:topological/qualitative
[9] ing.:metric/quantitative
[10] ing.:landmarks

bulunur. Robotlara hem panel metodundan hem de yapay potansiyel fonksiyonlardan gelen hız bileşenleri uygulanarak, robot sürüsünün istenilen dizilimi sağlayarak hedef noktasına engellere çarpmadan ulaşması sağlanır.

Bu çalışmada iki farklı yöntem kullanılmıştır. Birinci yöntemde ortam hakkında eksiksiz bilgi olduğunu varsayarak panel metodu ortamın haritasına uygulanmış, başlangıçtan hedefe kadar akış çizgileri belirlenmiş ve robotlara hız olarak verilmiştir. Bu yönteme çevrimdışı gezinme denebilir. Gezinme sırasında ortamda oluşabilecek değişiklikler algılanmaz ve robotlar eski haritaya göre gezinir. İkinci yöntemde ise, robotlar ortam hakkında herhangi bilgileri olmadan yola çıkar. Robotların üzerinde bulunan lazer tarayıcısının yardımıyla ortamın anlık harita karesi alınır, panel metodu uygulanır ve bulunan akış çizgileri bir süre gezinmek için kullanılır. Ortam tekrar lazer algılayıcı ile taranır ve yeni harita yeni akış çizgileri bulmak için kullanılır; hedef noktasına ulaşana kadar bu işlem tekrarlanır. Bu yönteme çevrimiçi ya da gerçek zamanlı gezinme denir. Ortamda herhangi bir değişiklik olursa bir sonraki taramada algılanır ve işleme geçilir.

Deneylerde elde edilen ve bu tezde gösterilen sonuçlar geliştirilen yöntemin gerçek robot sürüleri üzerine uygulanabileceğini göstermektedir. Gerek ortamın haritası önceden bilindiğini varsayarak gerekse de keşfedilmeyen ortamlarda yapılan deneyler başarıyla sonuçlanmıştır. Önceden bilinen ortamlarda panel metodu robot sürüsü harekete geçmeden önce uygulandığı için, çevrimdışı yöntemi gerçek zamanlı yöntemden daha hızlıdır. Öte yandan, gerçek zamanlı yöntem ortamda oluşan değişiklikleri aniden algıladığı için daha gürbüzdür. Çevrimdışı yöntemde robotların konumları tepe kamerasından alındığı için potansiyel fonksiyonların güncellemesinde gecikme yaşanmamaktadır. Gerçek zamanlı yöntem ise sürü robotların birbirlerini lazer tarayıcısıyla ya da iletişim vasıtasıyla algıladığı için konumlar gecikme ile güncellenmektedir. Bu da potansiyel fonksiyonların uygulamalarında gecikme oluşturmuş ve salınımlara yol açmıştır. Sürü robotların işlemcileri geliştirilir hızları yükseltilirse gerçek zamanlı yöntemin her türlü ortamlarda hatta sürekli değişen ortamlarda da uygun sonuçlar vereceği öngörülmektedir.

BÖLÜM 2

2. PANEL METODU

2.1. Giriş

Panel metodları karmaşık şekilli cisimlerin etrafında potansiyel akış problemi çözmek için kullanılan sayısal yöntemdir. Analitik çözümlerin tersine, Panel metodu problem doğrusal denklem sistemine dönüştürülerek, direkt veya interaktif sayısal yöntemlerle çözülebilir. Bunun için panel metodları aerodinamik analizlerde ve akışkanlar mekaniğinde geniş çapta kullanılır.

Kısaca panel metodu şu şekilde uygulanır. Potansiyel akışa maruz kalan bir cisim "panel" adlı küçük çizgilere ayrılır ve her panele tekillik elemanı eklenir. Cismin yarattığı potansiyel alanı bulmak için sınır koşulu şartlarından çıkan denklemler çözülür [8].

2.2. Sınır Koşulu

Katı bir cisim bir sıvı içine batırıldığında bu cismin kabuğu (dış yüzeyi) sıvı akışının içine geçmesini engeller. Panel metodunda, sıvı ile cismin dış yüzeyi arasındaki bağıl hızın dik bileşenin sıfır olduğu; başka deyişle, akışın cismin yüzeyine teğet olduğu sınır koşulu kullanılır.

$$\overrightarrow{\mathbf{Q}} \cdot \overrightarrow{\mathbf{n}} = 0 \tag{2.1}$$

Burada $\overrightarrow{\mathbf{Q}}$ sıvı ile cismin dış yüzeyi arasındaki bağıl hız ve $\overrightarrow{\mathbf{n}}$ cismin yüzeyine olan dikey vektördür [8].

Sıvı ile cismin arasındaki bağıl hızın dört bileşeni olduğu durumu ele alalım. Hızlar şunlardır: serbest akış hızı $\overrightarrow{\mathbf{Q}}_{serbest}$, cisimlerin sıvı içinde bulunmalarından indirgenmiş hız $\overrightarrow{\mathbf{q}}_{ind}$, başlangıç noktasında bulunan bir kaynak elemanından çıkan hız $\overrightarrow{\mathbf{q}}_{kaynak}$, ve hedef noktasına çizgilerini çekmek üzere konulan kuyu noktasından uyarlanan hız $\overrightarrow{\mathbf{q}}_{kuyu}$. Bu hızlar kullanılırsa sınır koşulu şu şekilde olur:

$$(\overrightarrow{\mathbf{Q}}_{serbest} + \overrightarrow{\mathbf{q}}_{kaynak} + \overrightarrow{\mathbf{q}}_{kuyu} + \overrightarrow{\mathbf{q}}_{ind}) \cdot \overrightarrow{\mathbf{n}} = 0 \tag{2.2}$$

Bilinen parametreleri denklemin sağ tarafına alındığında, sınır koşulu ile elde edilen denklem şu şekilde olur,

$$-\overrightarrow{\mathbf{q}}_{ind} \cdot \overrightarrow{\mathbf{n}} = (\overrightarrow{\mathbf{Q}}_{serbest} + \overrightarrow{\mathbf{q}}_{kaynak} + \overrightarrow{\mathbf{q}}_{kuyu}) \cdot \overrightarrow{\mathbf{n}} \tag{2.3}$$

Denklemin sol tarafı engelleri göstermek için kullanılan tekillik elemanına göre değişir. Dahası, akışın başka bileşeni varsa, örneğin ikinci serbest akış, bu bileşenin etkisi denklemin sağ tarafına eklenmesi gerekir [8].

2.3. Panel Metodun Uygulanması

Yukarıda söz edildiği gibi, panel metodu uygulamasında engeller küçük düz çizgilere (panellere) ayrılır, her panele şiddeti bilinmeyen bir tekillik elemanı (kaynak ya da girdap) eklenir, ve her panelde sınır koşulu (2.3) sağlanması beklenir. Problem her panelde yerleştirilen tekillik elemanın şiddetini bulmaya indirgenmiştir. Daha sonra tekillik elemanların şiddetleri, engellerin akışta bulunmalarından dolayı indirgenmiş hızın uzayda herhangi noktada hesaplanması için kullanılır. İndüklenen hız veya indüklenmiş hız serbest akış hızı, kaynak hızı ve kuyu hızına eklenerek toplam akışın hızı ve yönü herhangi bir noktada bulunur. Tekillik elemanlarının şiddetlerini bulmak için ortamda bulunan tüm panallerin etkisinin yanı sıra (tüm panellerin tekillik elemanlarının etkileri) (2.3) denkleminde bulunan dış akışın etkileri (serbest akış, kaynak ve kuyu etkileri) dikkate almak gerekir. Başka bir deyişle, (2.3) denkleminin sol tarafı tüm panellerin etkisini içerecek şekilde her panel için yazılır. En sonunda tekillik elemanlarının şiddetlerini bilinmeyen olarak içeren doğrusal denklemler sistemi elde edilir [9].

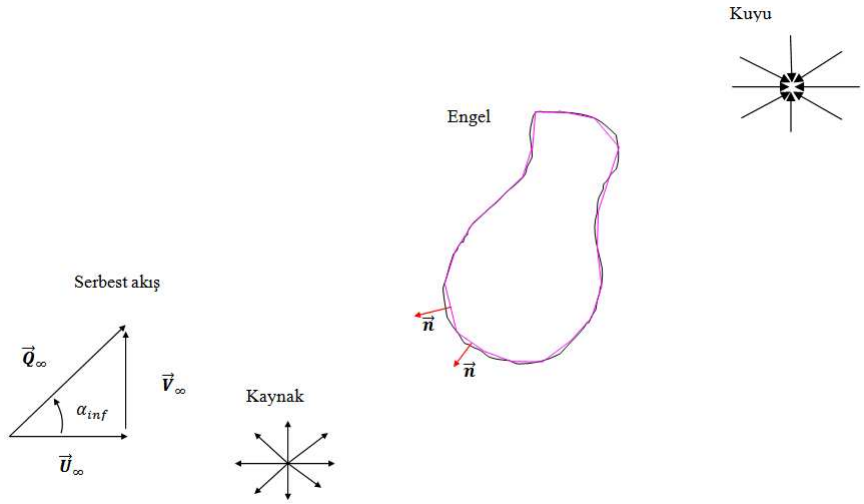
İki boyutlu uygulamalarda karmaşık şekilli engeller küçük çizgilere (panellere) ayrılır. Her panelin orta noktasında (kontrol noktası) tekillik elemanı olarak şiddeti bilinmeyen bir girdap yerleştirilir. Bir (x, y) noktasına, Γ_k şiddetli ve (x_k, y_k) noktasında bulunan girdap tarafından indirgenmiş hız aşagıdaki denklemdeki gibi yazılır

$$\overrightarrow{\mathbf{q}}_{ind} = u_{ind}\overrightarrow{\boldsymbol{i}} + v_{ind}\overrightarrow{\boldsymbol{j}} \tag{2.4}$$

Burada u_{ind} and v_{ind} indirgenmiş hızın x ve y bileşenleridir ve girdap için bu şekilde yazılır

$$u_{ind} = \frac{\Gamma_k}{2\pi} \frac{y - y_k}{(x - x_k)^2 + (y - y_k)^2} \tag{2.5}$$

$$v_{ind} = -\frac{\Gamma_k}{2\pi}\frac{x - x_k}{(x - x_k)^2 + (y - y_k)^2} \tag{2.6}$$



Şekil 2.1. Panel metodunun uygulanması

Şiddeti Γ_{kaynak} olan ve (x_s, y_s) noktasında bulunan bir kaynağın (x, y) noktasına etki eden hız etkisinin bileşenleri bu şekilde yazılır

$$u_{kaynak} = \frac{\Gamma_{kaynak}}{2\pi}\frac{x - x_s}{(x - x_s)^2 + (y - y_s)^2} \tag{2.7}$$

$$v_{kaynak} = \frac{\Gamma_{kaynak}}{2\pi}\frac{y - y_s}{(x - x_s)^2 + (y - y_s)^2} \tag{2.8}$$

Şiddeti Γ_{kuyu} olan ve (x_f, y_f) hedef noktasına yerleştirilen bir kuyunun (x, y) noktasına etki eden hız etkisinin bileşenleri bu şekilde yazılır

$$u_{kuyu} = -\frac{\Gamma_{kuyu}}{2\pi}\frac{x - x_f}{(x - x_f)^2 + (y - y_f)^2} \tag{2.9}$$

$$v_{kuyu} = -\frac{\Gamma_{kuyu}}{2\pi}\frac{y - y_f}{(x - x_f)^2 + (y - y_f)^2} \tag{2.10}$$

Panel i için sınır koşulu denklemi (2.3) iki boyutta, matris formatında bu şekilde yasılır

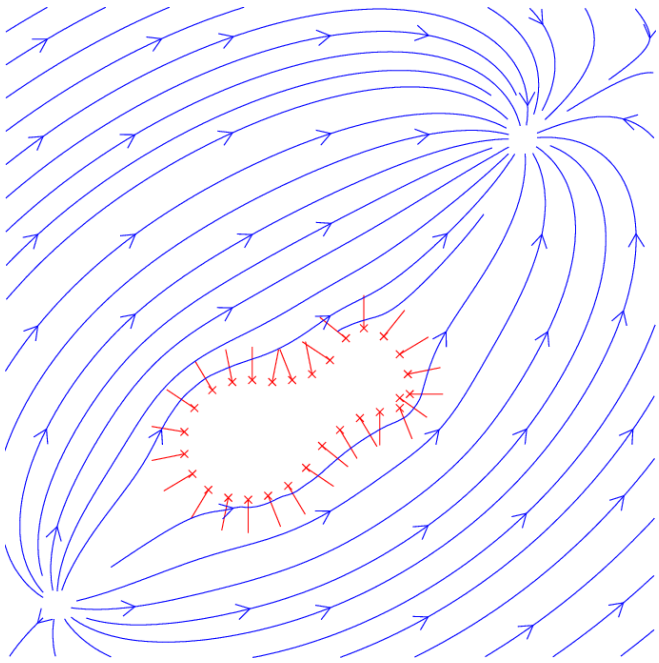
$$-\sum_{j=1}^{N}\begin{bmatrix} u_{ind_{ij}} \\ v_{ind_{ij}} \end{bmatrix} \cdot \begin{bmatrix} n_{x_i} \\ n_{y_i} \end{bmatrix} = \left(\begin{bmatrix} U_{serbest} \\ V_{serbest} \end{bmatrix} + \begin{bmatrix} u_{kaynak} \\ v_{kaynak} \end{bmatrix} + \begin{bmatrix} u_{kuyu} \\ v_{kuyu} \end{bmatrix}\right) \cdot \begin{bmatrix} n_{x_i} \\ n_{y_i} \end{bmatrix} \tag{2.11}$$

Burada $U_{serbest}$ ve $V_{serbest}$ serbest akış hızının x ve y bileşenleri, ve n_x ve n_y cismin yüzeyine dikey vektörün $\overrightarrow{\mathbf{n}}$ x ve y bileşenleridir.

Ortamda N panel olduğunu kabul edersek, (2.11) denklemi panellerin kontrol noktalarındaki indirgenmiş hızları bulmak için uygulanır. Bu hızların yüzeye teğet olması gerekir. Tüm panellerin etkisini göz önünde bulundurursak bu şekilde bir doğrusal denklem sistemi elde edilir

$$-\begin{bmatrix} K_{11} & \dots & K_{1N} \\ \vdots & & \vdots \\ K_{N1} & \dots & K_{NN} \end{bmatrix} \begin{bmatrix} \Gamma_1 \\ \vdots \\ \Gamma_N \end{bmatrix} = \begin{bmatrix} RHS_1 \\ \vdots \\ RHS_N \end{bmatrix} \tag{2.12}$$

Burada K_{ij} panel j'nin panel i'ye olan etkisidir (panel j'de yerleştirilen girdap'ın panel i'nin kontrol noktasına olan etkisi), Γ_i panel i'de yerleştirilen girdap'ın şiddeti, ve RHS_i (2.11) denklemin sağ tarafıdır (kaynağın, kuyunun ve serbest akışın etkileri). Panel i için $K_{ii} = -0.5$'tir.



Şekil 2.2. Ortamda akış çizgileri

Doğrusal denklemler sistemini çözerek bulunan girdapların şiddetleri, robotlar engellere çarpmadan hedefe ulaştıran hızları bulmak için kullanılır. Hızların bileşenleri

(2.5)'ten (2.10)'a kadar denklemleri kullanarak herhangi noktada bu şekilde bulunur

$$u_{robot} = U_{serbest} + u_{kaynak} + u_{kuyu} + \sum_{i=1}^{N} u_{ind} \tag{2.13}$$

$$v_{robot} = V_{serbest} + v_{kaynak} + v_{kuyu} + \sum_{i=1}^{N} v_{ind} \tag{2.14}$$

Burada *N* toplam panel sayısıdır, $\sum u_{ind}$ ve $\sum v_{ind}$ robot konumunda tüm panellerin toplam indirgenmiş hızlarının *x* ve *y* bileşenleridir.

Şekil 2.2.'de MATLAB çizdirme fonksiyonunu kullanmadan çizilen akış çizgileri gösterilmektedir. Bu programda akış çizgileri hesaplandıktan sonra $ode23$ fonksiyonu kullanarak bir robot sürüsüne takip ettirilmiş. Resimde çıkan çizgiler sürü robotların konumlarını çizdirerek gösterilmiştir.

Panel metodları uzayda bulunan karmaşık şekilli engellerin ayrıklaştırılmasından sonra uygulanan tekniklerdir. Böylece, hesaplamaların tüm uzayda olmaktansa sadece cisimlerin yüzeylerinde olmaları sağlanır. Bu yöntem seyrüsefer algoritmalarında kullanılması çok önemli ve kritik olan kolaylık ve hız sağlar. Ayrıca, panel metodları bir yaklaştırma yöntemidir. Daha iyi tahminde bulunmak için daha fazla panel sayısı kullanmak gerekir; bu da tabii ki daha karmaşık ve uzun hesaplamalar anlamına gelir.

BÖLÜM 3

3. ORTAMIN HARİTASININ BİLİNDİĞİ DURAĞAN ORTAMLARDA GEZİNME

Kör gezinme algoritması robotlarda ortamın haritasının bilindiğini varsayarak uygulanır. Akış çizgileri kullanarak gezinme yapmak için, elde bulunan haritaya başlangıç noktasına kaynak noktası, hedef noktasına kuyu noktası ve uzaktan gelen serbest bir akış ekleyerek panel metodu uygulanır. Engeller panellere ayrılır, her panelin kontrol noktasına şiddeti bilinmeyen bir girdap eklenir ve her panelin öbür panellere olan etki denklemleri yazılır. Sınır koşulu uygulandığında her panelde bulunan girdapların şiddetlerini çözmek için bir doğrusal denklem sistemi elde edilir. Bu sistemi çözerek girdapların şiddetleri ve de engellerin akışa olan etkisi bulunur. Yüzeyin herhangi noktasında akışın hızını ve yönünü bulmak için o noktada kaynak, kuyu, serbest akış ve engellerin etki hızlarını toplamak yeterlidir.

Bu çalışmada kör gezinme algoritmasının iki uygulaması ele alınmıştır. İlk uygulamada bilgisayar üzerinde benzetim uygulamasıdır. Altı adet erkin içeren bir robot sürüsü, haritası bilinen bir ortamda önceden belirlenmiş dizilimde hedef noktasına gezinmesi beklenir. İkinci uygulamada laboratuvar ortamında üç erkinli gerçek robot sürüsü kullanılmıştır.

3.1. Benzetim Uygulaması

Benzetim uygulaması MATLAB programı kullanılarak yapılmıştır. Robot dinamikleri, benzetimde kullanılan yapay potansiyel fonksiyonlar, benzetim programı, robot denetimi ve benzetim sonuçları aşağıda anlatılmaktadır.

3.1.1. Robot Dinamikleri

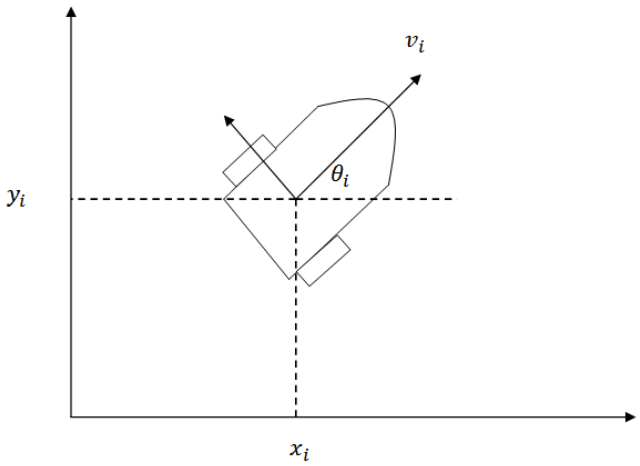
Benzetimler MATLAB programı yardımıyla bilgisayar ortamında yapılmıştır. Bu uygulamada altı adet hız kısıtlı[1] erkinden oluşan ve iki boyut alanında hareket eden

[1] ing.:non-holonomic

robot sürüsü kullanılmıştır. Her erkinin dinamikleri

$$\dot{x}_i(t) = v_i(t)\cos(\theta_i(t)) \tag{3.1}$$
$$\dot{y}_i(t) = v_i(t)\sin(\theta_i(t)) \tag{3.2}$$
$$\dot{\theta}_i(t) = w_i(t) \tag{3.3}$$

şeklinde gösterilir. Burada $x_i(t)$ ve $y_i(t)$ değişkenleri $p_i(t) = [x_i(t), y_i(t)]^\top$ konumun kartezyen koordinatlarında bileşenleridir, ve $\theta_i(t)$ i'ninci erkinin t zamandaki yönelim açısıdır. Bu erkinin kontrol girişleri doğrusal ve açısal hızlarıdır, sırasıyla $v_i(t)$ ve $\omega_i(t)$'dir. Erkinlerin şematiği Şekil 3.1.'de gösterilmektedir.



Şekil 3.1. Hız kısıtlı robot yapısı

3.1.2. Benzetimde Kullanılan Yapay Potansiyel Fonksiyonlar

Robot sürüsünün istenilen dizilimi sağlaması için yapay potansiyel fonksiyonlar erkinlerin arasındaki itim-çekim kuvvetlerini tasarlamak için kullanılmıştır. Dizilimi sağlayan potansiyel fonksiyonun şöyle bir etkisi olması beklenir: erkin, sürüde bulunan başka erkine istenilen mesafeden daha fazla yaklaşırsa potansiyel fonksiyonu itim fonksiyonuna dönüşür ve erkinin uzaklaşmasını sağlar; böylece erkinlerin çarpışması önlenmiş olur. Erkin komşu erkinden önceden belirlenen mesafeden fazla uzaklaşırsa potansiyel fonksiyonu çekim fonksiyonuna dönüşerek komşu erkinlerin yakınlaşmaları sağlanır; sürünün dağılması bu şekilde önlenmiş olur. Bu şartları

sağlayan yapay potansiyel fonksiyonu

$$\boldsymbol{J}(p) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left[\frac{a_{ij}}{2} \|p_i - p_j\|^2 + \frac{b_{ij} c_{ij}}{2} \exp\left(\frac{\|p_i - p_j\|^2}{2} \right) \right] \tag{3.4}$$

şeklinde olabilir [25]. Buradaki *n* sürüde erkinlerin sayısı, $p^\top = [p_1^\top, p_2^\top, ..., p_n^\top]$ tüm erkinlerin konumu içeren vektör, a_{ij} ve b_{ij} ve c_{ij} yapay potansiyel fonksiyonun şiddetini değiştirmek için kullanılan pozitif kazançlar.

Yapay potansiyel fonksiyonları robot gezinmesinde kullanmak için, robot fonksiyonun negatif eğiminin yönünde gitmesi gerekir. Robot sürüsünün dizilimi istenilen şekli sağlanması için sürü erkinleri, (3.4)'te görülen potansiyel fonksiyonun negatif eğiminini kullanır. (3.4)'ün gradyanı şu şekilde gösterilir

$$\nabla_{p_i} J(p) = \sum_{j=i+1, j\neq i}^{n} (p_i - p_j) \left[a_{ij} - b_{ij} \exp\left(\frac{\|p_i - p_j\|^2}{c_{ij}} \right) \right] \tag{3.5}$$

Burada i ve j erkinlerin arasındaki itim-çekim kuvvetleri aşağıda bulunan mesafede dengelenir

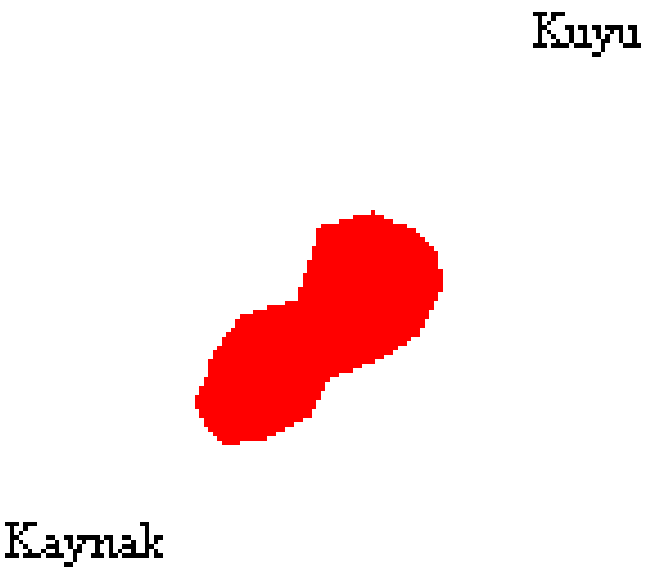
$$\delta_{ij} = \sqrt{c_{ij} \ln\left(\frac{b_{ij}}{a_{ij}} \right)} \tag{3.6}$$

δ_{ij} istenilen dizilimi sağlamak için erkinlerin arasındaki gereken mesafedir; başka deyişle, *i* ve *j* erkinlerin arasındaki itim-çekim kuvvetlerinin birbirini dengelediği mesafedir

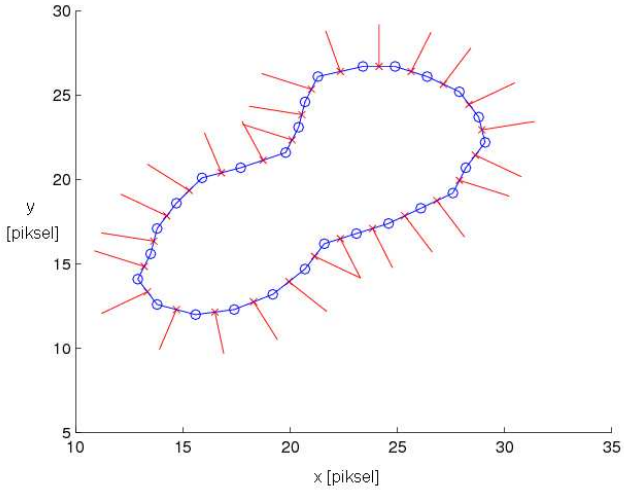
3.1.3. Programın Mantığı

Benzetim programı üç alt programdan oluşmaktadır. Birinci program, ortamın önceden çekilmiş harita resmini inceleyen programdır. Bu program görüntü işleme algoritmaları kullanarak haritada bulunan engelleri tanıyarak arka planından ayırır ve kenar algılama algoritmaları kullanarak engellerin kenarlarını bulur. Bu kenarlar aralarında belli mesafe bulunan noktalara ayrılır. Bu noktalar panellerin başlangıç ve son noktalarını oluşturur. Panellerin kontrol noktalarını bulmak için her panelin ilk ve son noktalarının orta noktası hesaplanır. Son olarak, her panelin açısı ve normal vektörü bulunur. Bu alt program, panel metodu alt programında kullanılan ve hesaplamalarda gereken panel bilgilerini bulma programıdır. Şekil 3.2.'de Kullanılan

ortamın haritası ve Şekil 3.3.'te işlenmiş harita programın çıktısı gösterilmektedir.



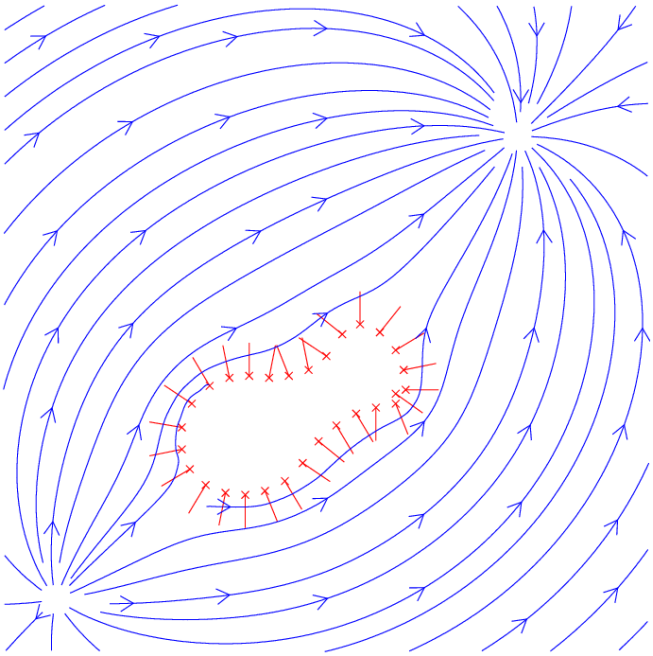
Şekil 3.2. Ortamın haritası



Şekil 3.3. Harita inceleyen programın çıkışı

İkinci alt program panel metodu programıdır. Bu program, kullanıcının istediği başlangıç noktasına bir kaynak noktası, hedef noktasına bir kuyu noktası koyarak, uzaktan serbest akış uygularak ve harita inceleyen programdan gelen engel bilgileri kullanarak panel metodunu uygular. Her panelin kontrol noktasına şiddeti bilinmeyen girdap noktası koyduktan sonra her panelin diğer panellere olan etkisi hesaplanır. En sonunda, sınır koşulu uygulanarak ve çıkan doğrusal sistem çözülerek her panele

konulan girdap'ın şiddeti bulunur. Kaynak noktası, kuyu noktası, serbest akış, ve engellerin yarattığı hızlar toplanarak. Bulunan akış çizgileri robotlara güvenli gezinme yolları oluşturur.



Şekil 3.4. Robotların güvenli gezinme çizgileri

Üçüncü alt program robotların ve robot sürüsünün dizilimini denetleyen programdır. Bu alt program her erkinin diferansiyel denklemlerini, konum ve yönünü, panel metodundan ve yapay potansiyel fonksiyonlardan gelen hızlardan tamamlar. Program, $ode23$ fonksiyonunu kullanarak diferansiyel denklem sistemini çözer ve her robotun gezinmesi için gereken ve sürünün dizilimini sağlayan hız ve açıyı bulur. Bulunan hız ve açılar robot sürüsünün, dizilimi koruyarak ve engellerden kaçarak başlangıç noktasından hedef noktasına gezinmesini sağlar.

3.1.4. Robot Kontrolü

Yukarıda söz edildiği gibi, istenilen dizilimin sağlanması için sürü erkinlerinin yapay potansiyel fonksiyonun negatif eğiminin yönünde hareket etmeleri gerekir. Ayrıca, güvenli bir seyrüsefer gerçekleşmesi için, erkinlerin akış çizgilerini takip etmeleri gerekir. Bu nedenle, sürünün erkinleri güvenli seyrüsefer yaparken istenilen dizilimi sağlamak için aşağıda bulunan fonksiyona göre hareket etmeleri gerekiyor

$$\dot{p}_i(t) = \begin{bmatrix} \dot{x}_i(t) \\ \dot{y}_i(t) \end{bmatrix} = -G_i(p) = \begin{bmatrix} -G_{x_i}(p_i) \\ -G_{y_i}(p_i) \end{bmatrix} \tag{3.7}$$

Buradaki $G_{x_i}(p_i)$ ve $G_{y_i}(p_i)$, $G_i(p)$'nin x ve y bileşenleridir

$$G_{xi}(p) = \nabla J_{xi}(p) - u_i \tag{3.8}$$

$$G_{yi}(p) = \nabla J_{yi}(p) - v_i \tag{3.9}$$

Buradaki u_i ve v_i panel metodu kullanarak bulunan i'ci erkinin hız bileşenleridir. Robotların doğrusal ve açısal hızları aşağıdaki şekildedir

$$v_{id}(t) = \|G_i(p)\| \tag{3.10}$$

ve

$$\theta_{id} = \mod \left(\arctan\left(\frac{G_{y_i}(p_i)}{G_{x_i}(p_i)}\right), 2\pi\right) \tag{3.11}$$

Bu hızları sağlamak için, i'ninci erkinin kontrol girişi şu şekilde seçilebilir

$$v_i(t) = v_{id}(t) \tag{3.12}$$

ve

$$\omega_i = -K_i(\theta_i - \theta_{id}) + \dot{\theta}_{id} \tag{3.13}$$

Burada $K_i > 0$ bir oransal kazançtır.

Üstelik, erkinler istenilen açıya dönerken küçük açı yönünden dönmelerini sağlamak için [25, 27]'de geliştirilen denetleyici kullanılmıştır

$$\omega_i = -K_i(\mod(\theta_i - \theta_{id} + \pi, 2\pi) - \pi) + \dot{\theta}_{id} \tag{3.14}$$

Bu denetleyici ileri besleme[2] terimli ($\dot{\theta}_{id}$) bir oransal denetleyicidir[3].

3.1.5. Benzetim Sonuçları

Benzetimlerde sürü erkinlerin ilk konumları ve yönleri rastgele atanmıştır. Seyrüsefer halindeyken korunması gereken dizilim ise köşelerinde ve her kenarın ortasında bir robot içeren eşkenar üçgendir. Bu dizilimi sağlamak için erkinlerin arasındaki

[2] ing.:feedforward
[3] ing.:proportional controller

mesafeler aşağıda bulunan matrise göre atanmıştır

$$d_{ij} = \begin{bmatrix} 0 & d_1 & d_1 & d_2 & d_3 & d_2 \\ d_1 & 0 & d_1 & d_1 & d_1 & d_3 \\ d_1 & d_1 & 0 & d_3 & d_1 & d_1 \\ d_2 & d_1 & d_3 & 0 & d_1 & d_2 \\ d_3 & d_1 & d_1 & d_1 & 0 & d_1 \\ d_2 & d_3 & d_1 & d_2 & d_1 & 0 \end{bmatrix} \tag{3.15}$$

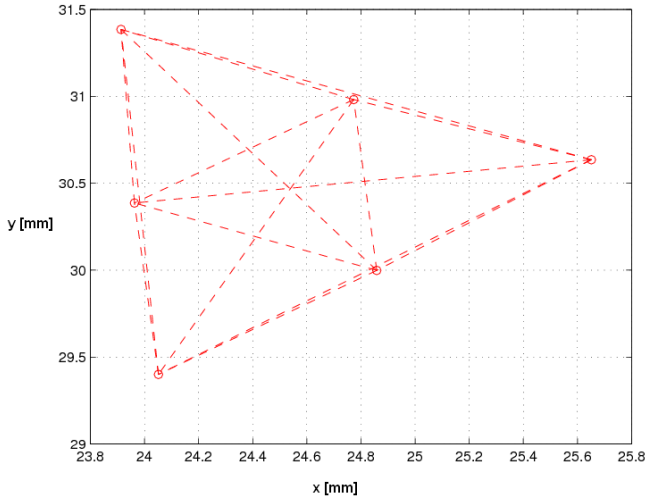
Yapay potansiyel fonksiyonda bulunan pozitif terimler $b_{ij} = 10$, $c_{ij} = 1$ seçilirken, a_{ij} sabiti yukarıdaki matrisin içindeki mesafeler sağlanması için aşağıdaki denkleme göre seçilmiştir

$$a_{ij} = b_{ij} \exp\left(-\frac{d_{ij}^2}{c_{ij}}\right) \tag{3.16}$$

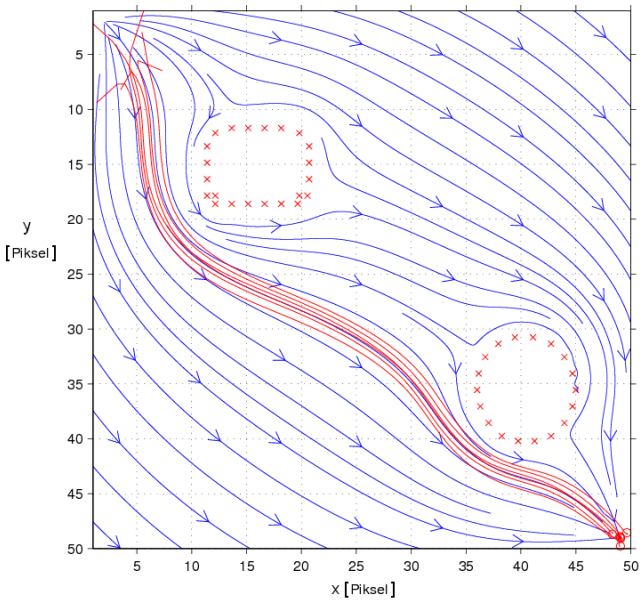
Başlangıç noktası olarak (2,2) koordinatlı nokta seçilmiştir. Bu noktaya 5 şiddetli bir kaynak noktası yerleştirilmiştir. Hedef noktası ise, şiddeti 5 olan bir kuyu noktası (40,40) koordinatlı noktasında belirlenmiştir. Erkinlerin düzgün dönüşler sağlamaları için (3.14)'de bulunan pozitif katsayı tüm erkinler için $K_i = 100$ olarak seçilmiştir.

Şekil 3.6., Şekil 3.7., Şekil 3.8. ve Şekil 3.9.'da dört farklı haritada benzetim sonuçları gösterilmektedirler. Robot sürüsü istenilen dizilimi sağlarken ve engellerden kaçınırken başarıyla hedef noktasına ulaşmıştır. Şekillerden görüldüğü gibi sürünün güzergahı erkinlerin ilk konum ve yönlerine bağlıdır. Dağınık başlayan sürüde erkinlerin arasındaki mesafeler çok büyük olduğu için çekim kuvvetleri akıştan gelen kuvvetleri yener ve sürünün toplanmasını sağlar. Toplanan ve dizilimi sağlayan sürü, akış çizgilerini takip ederek hedefe ulaşır. Şekil 3.5.'te başlangıçta sağlanan ve seyrüsefer esnasında korunan dizilim gösterilmektedir.

Aşağıdaki sonuçlarda görüldüğü gibi, robot sürüsü problem yaşamaksızın hedef noktasına ulaşmıştır. Benzetim ortamı ideal bir ortamdır, robot donanım hataları, ortamdaki değişiklikler, kaymalar ve gecikmeler benzeri beklenmeyen durumlarda bozulma etkisi yoktur. Gerçek uygulamalarda etkili olan belirsizlikler ve kararsızlıklar algoritmanın çıkışını bozabilirler. Bozulmaların etkisini azaltmak için dış etkenleri ve donanımda oluşabilecek değişiklikleri göz önünde bulundurmak gerekir.



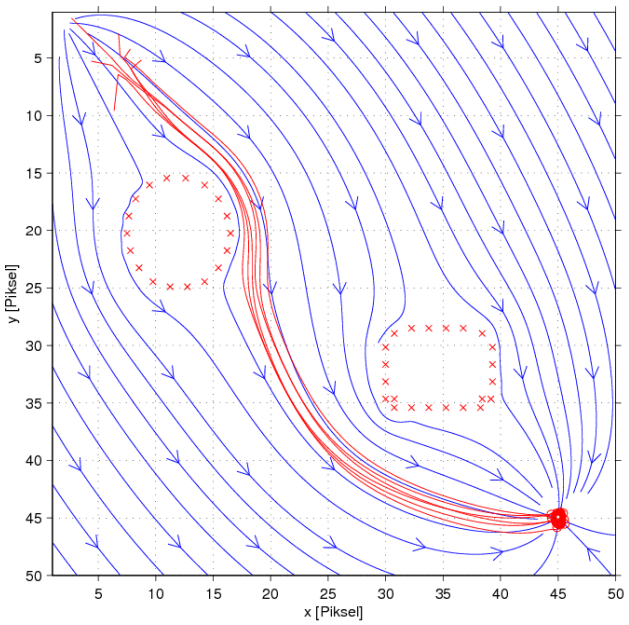
Şekil 3.5. Robot sürüsünün seyrüsefer esnasında dizilimi



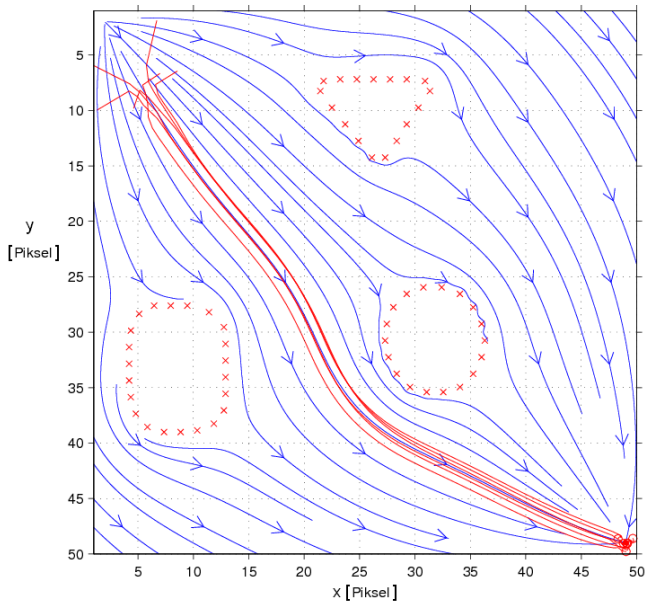
Şekil 3.6. Robot sürüsünün seyrüseferi (Harita 1)

3.2. Robot Üzerinde Uygulama

Bu bölümde, geliştirilen algoritma laboratuvar ortamında gerçek robot üzerinde uygulanmıştır. Üç robottan oluşan sürünün birkaç engel içeren ortamda engellere takılmadan ve önceden belirlenen dizilimi koruyarak hedef noktasına gitmesi beklenmektedir. İstenilen dizilim eşkenar bir üçgen olarak seçilmiştir.



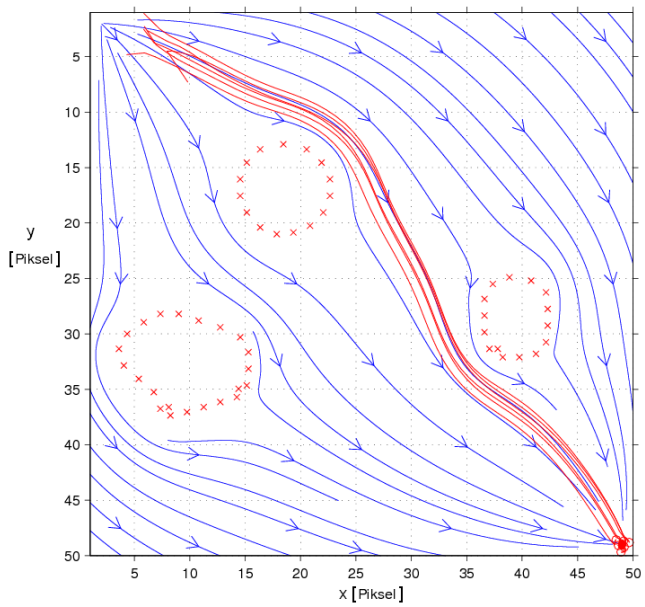
Şekil 3.7. Robot sürüsünün seyrüseferi (Harita 2)



Şekil 3.8. Robot sürüsünün seyrüseferi (Harita 3)

3.2.1. Deney Ortamı

Uygulamalar Şekil 3.10.'da görülen 120x180 cm genişliğinde bir arenada gerçekleştirilmiştir [17]. Uygulama alanı bir masaüstü bilgisayar, iletişim için



Şekil 3.9. Robot sürüsünün seyrüseferi (Harita 4)

Bluetooth arayüzü donanımlı ve 7 cm çapında üç adet e-puck robotu, konum geri beslemesini sağlamak için bir tepe kamerası ve sistemin kontrolü sağlamak için MATLAB programıdır. E-puck robotları diferansiyel sürüş ile hareket eden hız kısıtlamalı robotlardır. Bu robotlar bluetooth arayüzü yardımıyla bilgisayardan sağ ve sol tekerleklerin hızlarını alarak kontrol edilir. Tepe kamerası robotların konumları, yönleri ve kimlik bilgilerini bulmak için kullanılmıştır. Deney ortamı hakkında daha fazla bilgi için [17] numaralı kaynağa başvurulabilir.

Tüm görüntü tanıma ve kontrol programları MATLAB programında masaüstü bilgisayar üzerinde çalışmaktadır. Programların toplam devir zamanı 0.03 saniyedir. Başka deyişle, robot motorlarının hızları her 0.03 saniyede bir güncellenmektedir.

3.2.2. Programın Akışı

Bu uygulama çevrimdışı gezinme grubuna girdiği için harita bilgisi tamamen önceden bilinmektedir. Tepe kamerası ve MATLAB programı kullanarak ortamın haritası çekilir, görüntü işleme teknikleri kullanarak engeller ayıklanır ve küçük panellere ayrılır. Sınır koşulu kullanarak her panele konulan girdabın şiddeti (3.19)'da bulunan doğrusal denklem sistemi çözülerek bulunur. Ortamın her noktasında serbest akış, kaynak ve kuyu noktalarının etkisi indirgenmiş hızlara eklenerek robotlara gereken güvenli gezinme yolları elde edilmiştir.

Şekil 3.10. Deney Ortamı [17]

Yukarıda geçen işlem çevrimdışıdır yani robotlar çalışmadan önce yapılır. Robotlar harekete geçmeden önce hedef noktasına emniyetle götüren tüm yolların biliniyor olması gereklidir. Robotlar harekete geçtiğinde onların konumları, yönleri ve kimlikleri tepe kamerası yardımıyla bilgisayara gönderilir. Bu bilgileri kullanarak her robot için kendi konumunda panel metodun hızı bulunur ve robotların arasındaki mesafeleri hesaplayarak dizilimi sağlamak için gereken hareketlerin büyüklükleri ve yönleri bulunur. Sürünün erkinleri hem panel metodundan gelen hız hem de yapay potansiyel fonksiyonlardan gelen hızı sürüş için kullanırsa, dizilimi koruyarak hedef noktasına götüren seyrüsefer sağlanmış olur.

3.2.3. Kullanılan Yapay Potansiyel Fonksiyonlar

Benzetim uygulamasında olduğu gibi robotlar üzerinde yapılan uygulamada robotların arasında itim-çekim kuvvetleri ile sürü dizilimi sağlamak için yapay potansiyel fonksiyonlara başvurulmuştur. Yakın mesafede itimi sağlayan ve uzun mesafede çekim

gücü uygulayan fonksiyon aşağıdaki şekilde tasarlanmıştır

$$J(p) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} a_{ij} \Big[\|p_{ij}(t)\| - d_{ij} \ln \left(\|p_{ij}(t)\| \right) \Big] \tag{3.17}$$

Buradaki *N* sürüde bulunan erkin sayısıdır, $p^\top = [p_1^\top, p_2^\top, ..., p_N^\top]$ erkinlerin konumlarını içeren vektör, $p_{ij}(t) = p_i(t) - p_j(t)$ erkin *i* ve erkin *j*'nin konumlarını bağlayan vektör, a_{ij} iki erkinin arasındaki fonksiyonun etkisini değiştirmek için kullanılan pozitif kazanç ve d_{ij} *i* ve *j* erkinlerin arasında itim ve çekim kuvvetlerin birbirini dengelediği mesafedir. Bu fonksiyonun p_i konumundaki gradyanı aşağıda bulunmaktadır

$$\nabla_{p_i} J(p) = \sum_{j=i+1, j\neq i}^{N} a_{ij} \big(p_i(t) - p_j(t) \big) \times \left[\frac{1}{\|p_{ij}(t)\|} - \frac{d_{ij}}{\|p_{ij}(t)\|^2} \right] \tag{3.18}$$

Yukarıda geçtiği gibi, robotların istenilen dizilimi sağlaması için potansiyel fonksiyonun negatif eğimin yönünde hareket etmeleri gerekir.

3.2.4. Robot Kontrolü

Bu uygulamada, hız kısıtlamalı üç e-puck robotu kullanılmıştır. Sürüde bulunan her erkinin Şekil 3.1.'de gösterilen dinamiğe sahiptir. Robotların dinamikleri (3.3) denkemlerinde gösterilmiştir. Robotlar istenilen dizilimde ve engellere takılmadan hedef noktasına ulaşmak için hem yapay potansiyelin negatif gradyanını hem de panel metodundan gelen hızları kullanmaları gerekir. Robotların hareketlerinin bileşenleri (3.8) ve (3.9)'da gösterildiği gibidir. Robotların doğrusal ve açısal hızları benzetimdeki hızlar gibi (3.12) ve (3.14) olur. Bulunan kontrol girişleri v_i ve ω_i robotlara uygulamak için sağ ve sol motor hızlarına dönüştürülmesi gerekir. Dönüştürme işlemi robotların donanımsal özelliklerini kullanarak yapılmıştır. Dönüştürmeden sonraki sağ ve sol motorun hızları aşağıdaki doğrusal sistemde gösterilmektedir

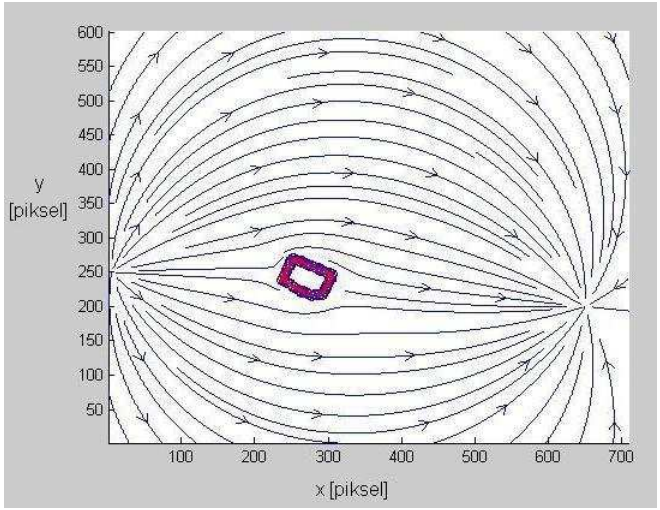
$$\begin{bmatrix} V_{Ri} \\ \\ V_{Li} \end{bmatrix} = \begin{bmatrix} \frac{1}{R} & \frac{-L}{R} \\ \\ \frac{1}{R} & \frac{L}{R} \end{bmatrix} \begin{bmatrix} v_i \\ \\ \omega_i \end{bmatrix} \tag{3.19}$$

Buradaki V_{Ri} sağ motor ve V_{Li} sol motorun hızıdır. $2L$ robotun iki tekerleklerin arasındaki mesafe ve R tekerleklerin yarı çapıdır. E-puck robotları için $L = 26.5$ mm ve $R = 20$ mm'dir.

3.2.5. Uygulama Sonuçları

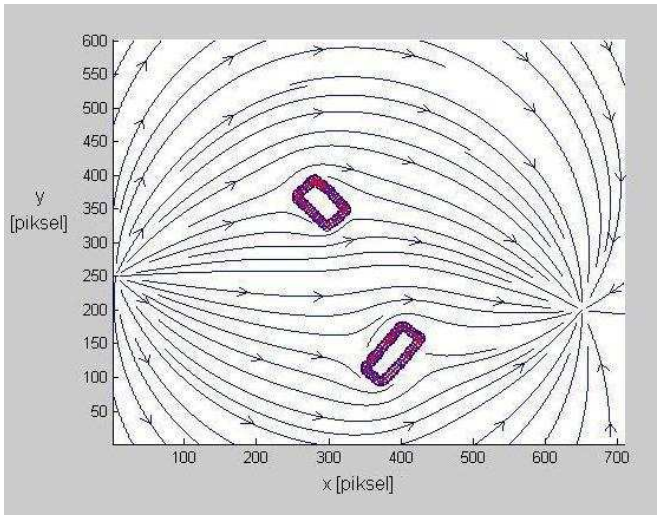
Robotlar rastgele başlangıç noktalarından harekete başlar, ve hem akış çizgilerini takip ederek hem de potansiyel fonksiyonlara göre hareket ederek eşkenarlı üçgeni gerçekleştirip istenilen dizilimi sağlar. Toplanan sürü oluşturduğu dizilimi sağlayarak hedef noktasına engellere çarpmadan gezinir.

Erkinlerin arasındaki mesafe 100 piksel olarak seçilmiştir. Tepe kamerasının arenaya olan uzaklığında beş piksel 2 santim'e tekabul eder, ve eşkenar üçgenin kenarlarının uzunluğu 40 cm'dir. Bu uzaklığı ayarlamak için, yapay potansiyel fonksiyonların kazancı $a_{ij} = 3000$ olarak seçilmiştir. (3.14)'teki açısal hızın kazancı ise $K_i = 15$ olarak seçilmiştir. Deneyler farklı hedef noktaları ve farklı engel dağıtımıyla yapılmıştır. İlk deneyler ortamda tek engel varken yapılmıştır. Şekil 3.11.'da tek engel varken akış çizgileri gösterilmektedir. Şekil 3.12.'de ise iki engel varken akış çizgileri gösterilmektedir.



Şekil 3.11. Harita-1 için akış çizgileri

Şekil 3.13. ve Şekil 3.14.'te Harita-1'de yapılan deneylerin sonuçları gösterilmektedir. Her sonuç, erkinlerin izlediği yol ve onların arasındaki mesafeleri içermektedir. Sonuçlardan görüldüğü gibi robotlar başlangıç noktasından hedef noktasına başarıyla ulaşmıştır. Üstelik, erkinlerin arasındaki mesafeler 100 pikselde sabitlenerek istenilen dizilim seyrüsefer boyunca sağlanmıştır. Her iki sonuçta bulunan küçük hatalar, geri beslemede kullanılan görüntü tanıma işlemlerinden gelen gecikmelerden oluşmuştur. İlk deneyde sürünün tüm erkinleri engelin bir tarafından geçmiştir. Seyrüsefer istenilen dizilim bozulmadan gerçekleştirilmiştir. İkinci deneyde ise robotların güvenli bir

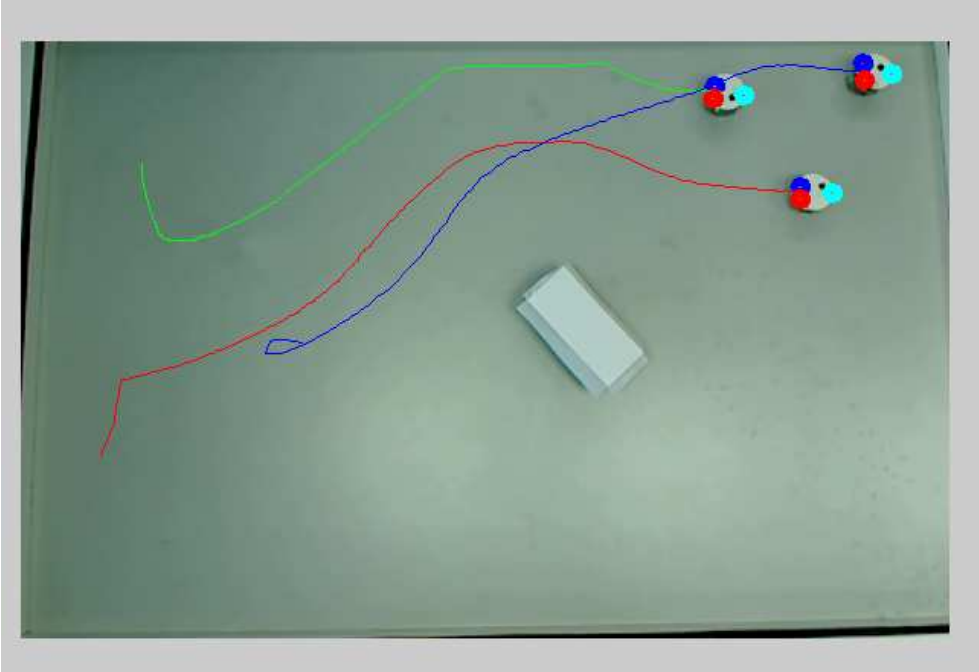


Şekil 3.12. Harita-2 için akış çizgileri

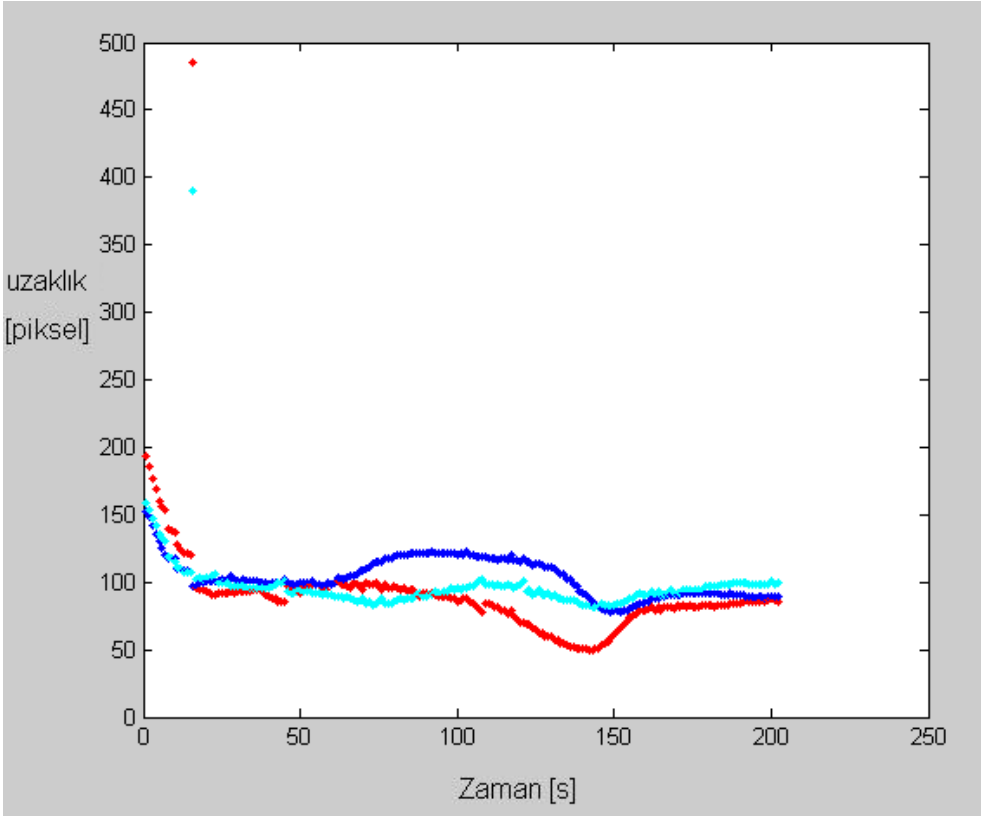
şekilde hedefe ulaşabilmeleri için dizilimde geçici bozulma gerçekleştirilmiştir. Ancak Şekil 3.14.'te gösterildiği gibi bir robot engeli diğer taraftan geçer geçmez dizilim tekrar sağlanmıştır.

Şekil 3.15. ve Şekil 3.16.'da iki engelli haritada yapılan deneylerin sonuçları gösterilmektedir. Şekillerde görüldüğü gibi, robot sürüsü dizilimini koruyarak hedef noktasına güvenli seyrüsefer yapabilmektedir. Sürünün güzergahı robotların başlangıç konumlarına bağlıdır. Her deneyde robotlar farklı konumlar ve yönler ile başlamaktadır. Yapılan ilk işlem sürünün toplanmasıdır; robotlar istenilen dizilimi sağlamak üzere birbirlerine yaklaşmaktadır. İstenilen dizilim sağlandıktan sonra robotlar hedefe doğru döner ve yönteme göre hareket etmeye başlamaktadır. Akış çizgileri takip etmek ve dizilimi sağlamak üzere uygulanan kontrol girdileri sürünün güzergahını belirtmektedir. Deney sonuçlarından görüldüğü gibi sürü bazen engellerin yan tarafından, bazen engellerin arasından ve hatta dizilimi bir süre bozarak ve engelleri aştıktan sonra tekrar toplanarak hedefe ulaşmıştır.

Ortamda çok engel bulunuyorsa ya da hedef noktasına açılan yollar dar ise sürünün istenilen dizilimi korunması zordur. Bu durum Şekil 3.17.'de gösterilmektedir. Emniyetli bir geçiş sağlayabilmesi için sürü dizilimi bozmaya mecbur kalmıştır. Başka bir seçenek ise dar geçişlere girmeden önce sürünün dizilimini değiştirmek te olabilir (üçgenden düz çizgiye). Sürünün dizilimi önemli ve değiştirilmez ise geçişi sağlayabilecek başka yollar bulunması gerekir.

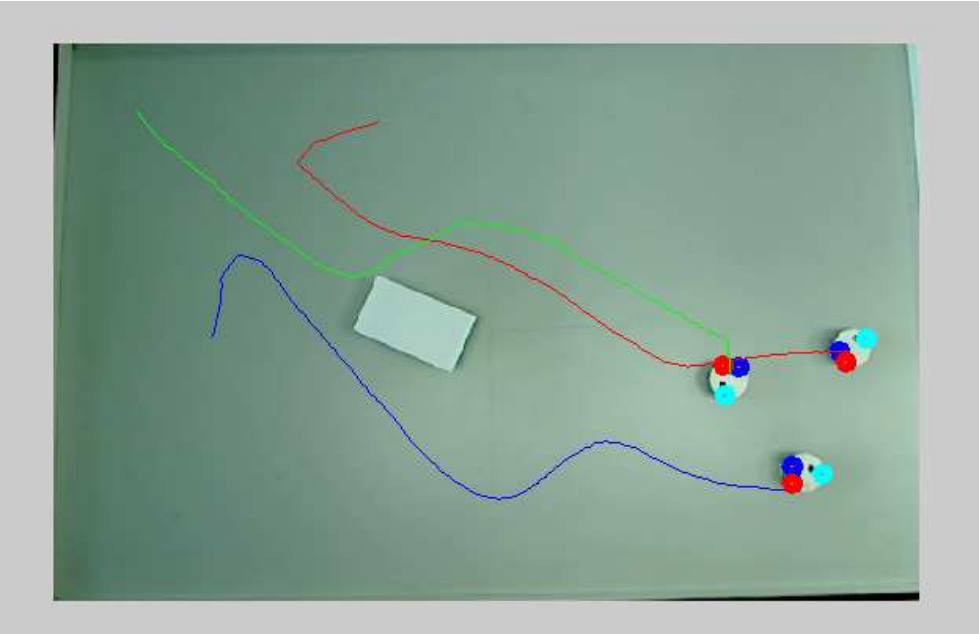


a- Robotların güzergahları

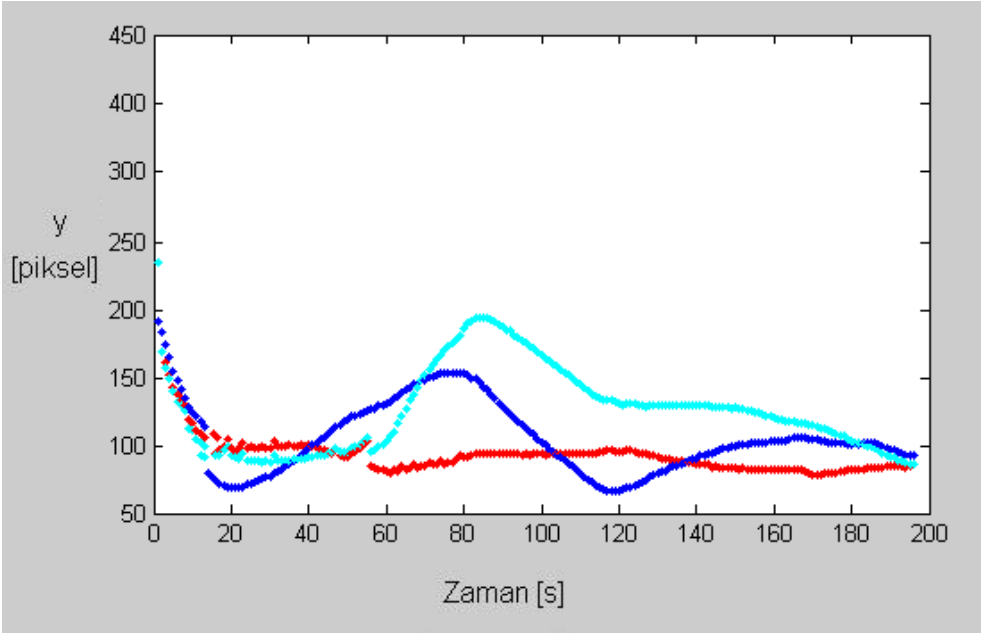


b- Robotların arasındaki mesafeler

Şekil 3.13. Robot sürüsünün seyrüseferi - Harita-1

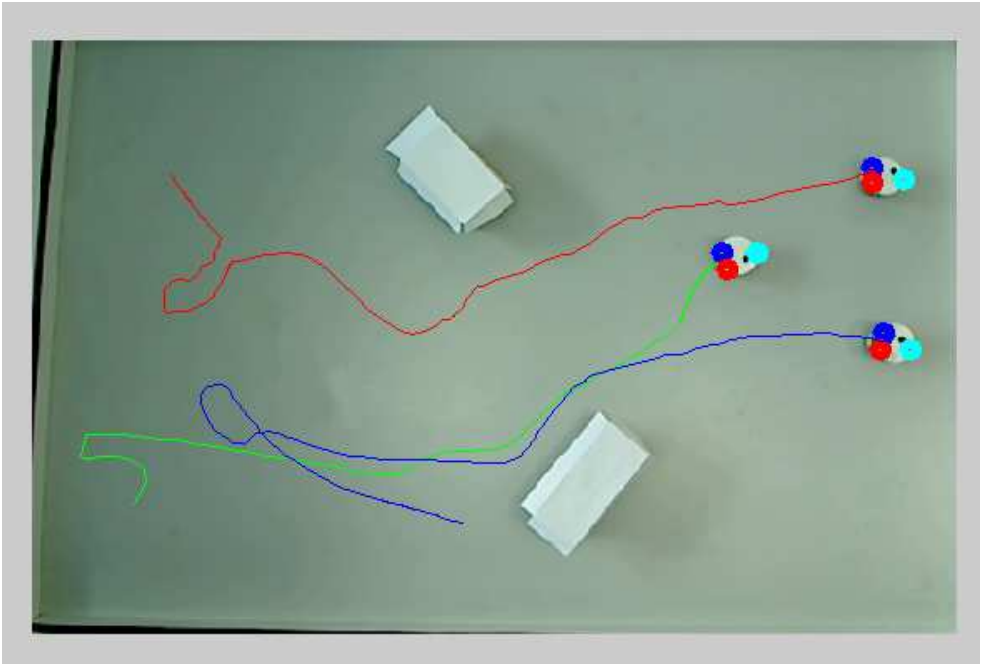


a- Robotların güzergahları

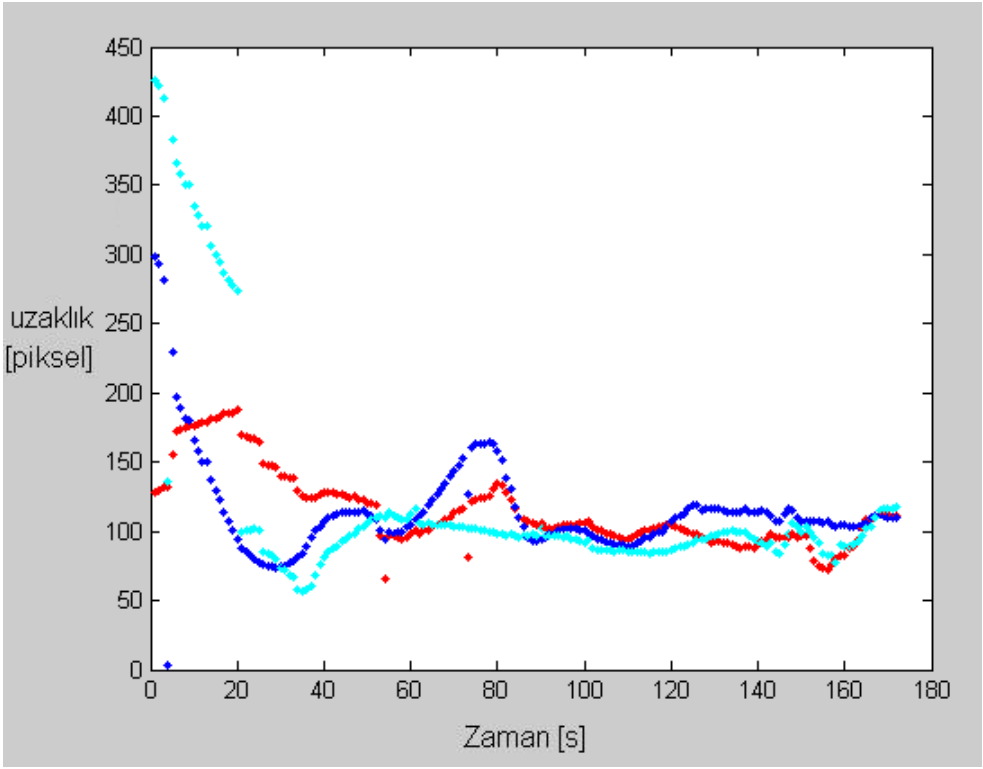


b- Robotların arasındaki mesafeler

Şekil 3.14. Robot sürüsünün seyrüseferi - Harita-2

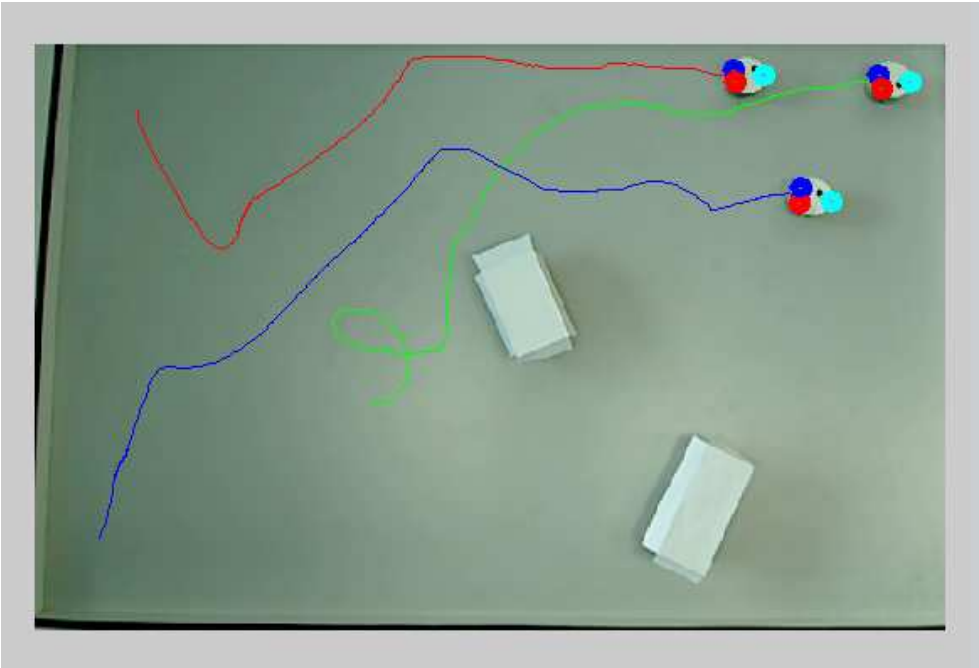


a- Robotların güzergahları

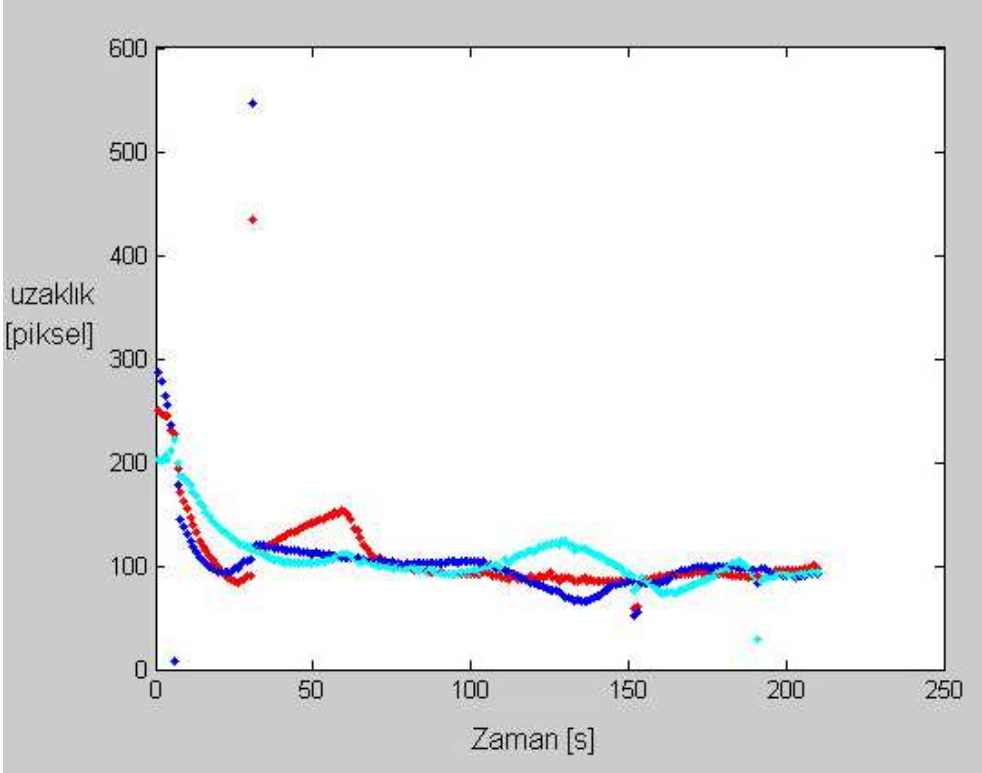


b- Robotların arasındaki mesafeler

Şekil 3.15. Robot sürüsünün seyrüseferi - Harita-3

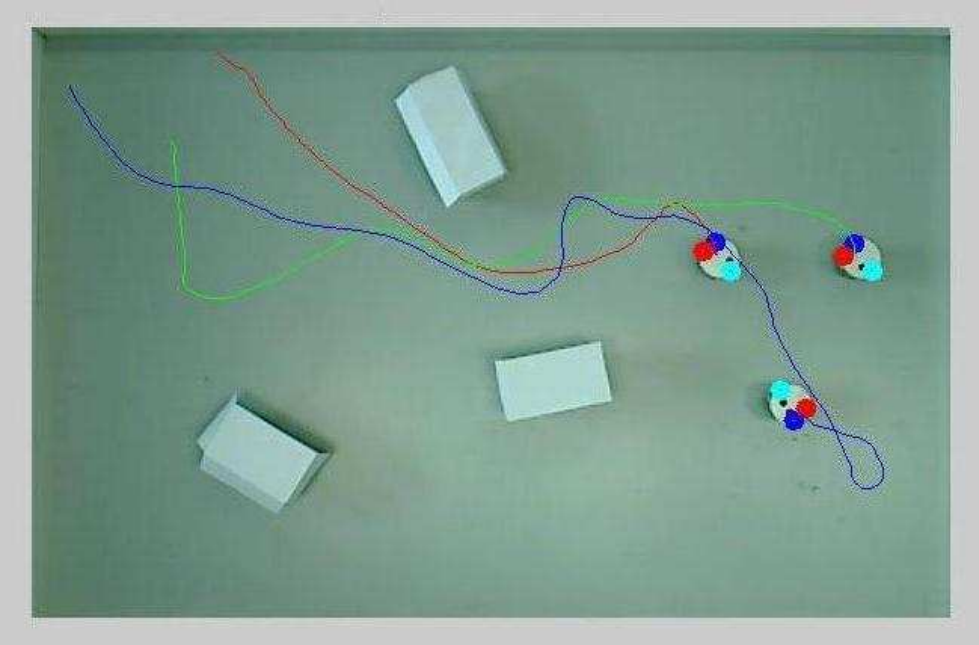


a- Robotların güzergahları

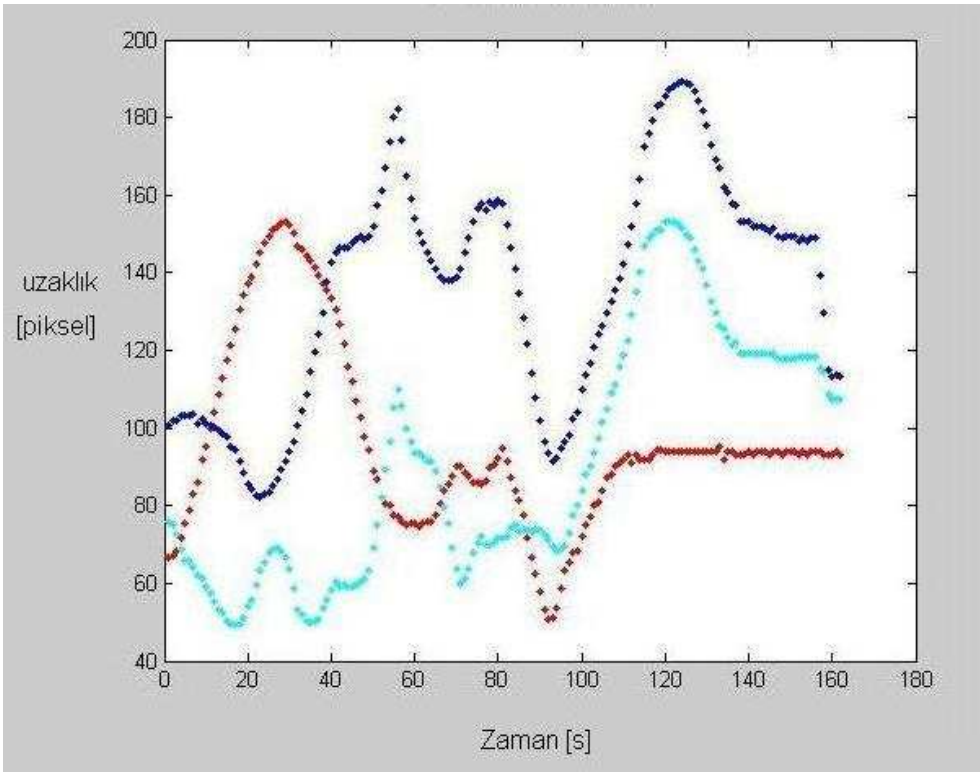


b- Robotların arasındaki mesafeler

Şekil 3.16. Robot sürüsünün seyrüseferi - Harita-4



a- Robotların güzergahları



b- Robotların arasındaki mesafeler

Şekil 3.17. Birden fazla engelli ortamda seyrüsefer

BÖLÜM 4

4. HARİTANIN BİLİNMEDİĞİ ORTAMLAR VE GERÇEK ZAMANLI PANEL METODU

Bu bölümde, geliştirilen algoritma bilinmeyen bir ortama uygulanmıştır. Robotlar ortamı tanıyabilmek için lazer tarayıcı sistemlerle donatılmıştır. Robotlar hareket ederken ortamdan gördükleri alana panel metodu uygulayarak akış çizgileri hesaplayıp takip etmeye çalışırlar. Yeni görüş alanı açılınca ve yeni engeller ortaya çıktıkça panel metodu tekrar uygulanır ve akış çizgileri güncellenir. Bu çevrimiçi yöntem dinamik ortamlara uygulanır ve robotlara sürekli değişen ortamlarda güvenli yolları verebilir.

Bu uygulamada üç KheperaIII robotu kullanılmaktadır. Bu üç robot sürü halinde gezer ve akış çizgileri takip ederken önceden belirlenmiş geometrik şekli koruyarak sürü halinde gezer. Sürünün şeklini korumak için yine yapay potansiyel fonksiyonlar kullanılmaktadır.

4.1. Deney Ortamı ve Programın Çalışma Mantığı

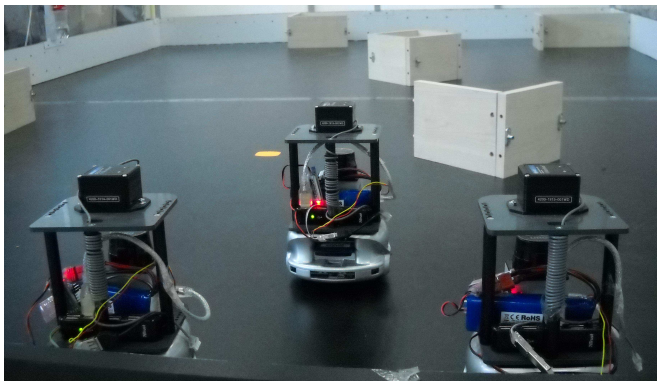
4.1.1. Deney Ortamı

Deneyler 240×340 cm genişliğinde, 15 cm yüksekliğinde duvarları olan arena içinde, üç KheperaIII robot ve masaüstü bilgisayarla yapılmıştır. KheperaIII robotların uygulamada engelden kaçmak için kullanılan 9 kızılötesi algılayıcıları, 5 ses ötesi algılayıcıları ve enkoderle donatılan iki DC motoru vardır. Robotların işlemci kısmı KoreBot LE platformudur. KoreBot LE platformu 400 MHz'lik işlemcisi, 32 MBaytlık flaş belleği, 64 Mbaytlık RAM belleği ve Linux kernel 2.6 işletim sistemi olan platformdur. Robotların arasındaki ve bilgisayar ile olan iletişimleri IEEE 802.1 kablosuz ağ uzerinden saglanmaktadır.

Robotların ortamı görüp tanıyabilmeleri ve engelleri ayıklamaları için Şekil 4.1.'de gösterilen Hokuyo URG-04LX lazer tarayıcısı ile donatılmıştır. Tarayıcının küçük boyutları (50mm $\times 50$mm $\times 70$mm), düşük ağırlığı, büyük performansı (menzili 4m'ye kadar çıkabilir ve 240^o görüş açısını 0.36^o çözünürlükle tarayabilir) ve düşük güç harcaması (2.5W) küçük robot uygulamalarında sahip oldugu bu ozelliklerden dolayı tercih edilmiştir. 240^o görüş açısı ve 0.36^o çözünürlük ile tarayıcı, ortamı 682 nokta (örnek) ile ifade eder; panel metodu için ve robotların düşük özellikleri olan

Şekil 4.1. Hokuyo lazer tarayıcısı

işlemcileri için oldukça büyük panel sayısı verir. Robot işlemcisi 600'e yakın paneli işlemek için oldukça uzun zamana ihtiyacı vardır. Ayrıca algoritmanın çevrimiçi ya da gerçek-zamanlı olmasından dolayı panel metodu hesaplarının tekrarlanması gerekir. Deneylerimize göre robot eğer 56 panel görürse (3.19)'da bulunan denklemleri çözmek için 0.69 saniyeye ihtiyacı vardır. Panel sayısı 120'ye çıkarsa bu zaman dilimi 3.385 saniyeye çıkar. Bu sorunu çözmek için sadece hedef noktası yönünde ve robotun 45^o görüş açısı içerisinde bulunan noktalar alınmaktadır. Bu sezgiseldir çünkü robot için sadece hedef noktasına ulaşmasına engel olan noktalar önemlidir. Deneye hazır robotlar Şekil 4.1.'de gösterilmektedir.



Şekil 4.2. Khepera III gezgin robotları ve uygulama düzeneği

4.1.2. Programın Çalışma Mantığı

Deneyin başlangıcında lider robot ortamı tarar, engelleri ayıklar ve sadece 45^o bölgesinde bulunan noktaları panellere çevirir ve kullanır. Sonra da, (3.19)'da bulunan denklemleri çözer ve panellerde bulunan girdap şiddetleri ve engel bilgisini diğer robotlara kablosuz ağ yardımı ile yollar. Tüm sürü elemanları (lider ve takipçi robotlar) engel bilgisi ve girdap şiddetlerini alınca akış çizgilerini hesaplar. Lider robot akış çizgileri kullanarak hareket eder. Takipçi robotlar hem akış çizgilerine göre hem de dizilimi koruyarak hareket etmesi gerekir. Bunun için takipçi robotlar lider ve kendi aralarında haberleşir ve birbirlerinden konum bilgilerini alırlar. Bu konumları potansiyel fonksiyonlarda kullanarak sürü elemanları arasındaki itim- çekim kuvvetleri hesaplanır. Takipçi robotlar hem akış çizgilerinden gelen hızları hem de potansiyel fonksiyonlardan gelen hızları kullanır; bu şekilde robot sürüsü hedef noktasına yaklaşırken önceden belirlenmiş dizilimi korur. Bu algoritmada gerçek zamanlı panel metodu uygulanmaktadır. Teoride lider robot sürekli ortamı tarar, etrafta oluşan değişiklikleri ayıklar ve panel metodu kullanarak akış çizgilerini güncellemektedir. Sürüde bulunan robotların hesap gücü düşük olduğundan her tarama gezinmede zaman kaybına yol açmaktadır. Algoritmadaki yavaşlamaları azaltmak için tarama işlemi her adımda yapılmamaktadır; lider robot bir tarama yapar, elde edilen engel bilgilerini gezinmek için bir süre kullanır, sonra da yeni tarama yapar ve güncellenen harita bilgilerini kullanır.

4.2. Uygulamada Kullanılan Yapay Potansiyel Fonksiyonlar

Literatürde, yapay potansiyel foksiyonlar robot seyrüseferi için kullanılmıştır. Geçmiş yıllarda bazı araştırmacılar yapay potansiyel fonksiyonları çoklu robot sistemlerinde erkinler arasındaki mesafeleri ayarlamak için kullanmıştır (detaylı bilgi için bknz. [25]).

Erkin i'nin t zamanındaki konumu $p_i(t)$ olsun, ve $p^\top = \left[p_1^\top, \ldots, p_N^\top\right]$ olsun, *N* sürünün içindeki erkin sayısıdır. Uzak mesafede çeken ve yakın mesafede iten bir potansiyel fonksiyonu şu şekilde tanımlanabilir

$$J(p) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} a_{ij} \left[ln\left(\|p_{ij}(t)\|\right) + \frac{d_{ij}}{\|p_{ij}(t)\|^2} \right] \tag{4.1}$$

Burada $p_{ij}(t) = p_i(t) - p_j(t)$ robot *i* ve robot *j* arasındaki vektördür, a_{ij} robotlar arasındaki potansiyel fonksiyonu ağırlaştırmak için bir pozitif katsayıdır, ve d_{ij} erkin *i*

ve *j* arasındaki kuvvetlerin dengelediği mesafedir.

Sadece toplanma isteniyorsa tüm çift robotlar (i, j) için $d_{ij} = d$ olarak seçilebilir; $d > 0$'dır. Eğer sürünün belirlenen bir şekli koruması isteniyorsa, d_{ij} istenilen geometrik şekilde erkinlerin arasındaki mesafeyi temsil eder. d_{ij} değerleri dikkatli seçilirse, sürüye istenilen geometrik şekil verilebilir. Potansiyel fonksiyonların yerel minimum problemi vardır. Yani yukarıdaki potansiyel fonksiyon gibi fonksiyonlarda her zaman ve her başlangıç koşulu için istenilen geometrik şeklin oluşması garanti edilemez.

4.3. Robot Denetimi

Bu uygulamada kullanılan KheperaIII robotları, (3.3)'te gösterilen dinamik denkemlerine sahiptir. Burada geliştirilen çevrimiçi algoritma lider tabanlıdır. Lider ortamı keşfeder ve takipçi robotlara engel bilgisini yollar. Takipçi robotlar gezinmek için hem akış çizgileri hem de potansiyel fonksiyonları kullanır. Lider robot gezinmek için akış çizgilerini hesaplar ve aşağıdaki denklemlere göre hareket eder

$$v_l(t) = \|[v, u]^\top\| \tag{4.2}$$

$$\omega_l = -K_l \mod ((\theta_l - \theta_{ld} + \pi, 2\pi) - \pi) \tag{4.3}$$

Burada u ve v lider robotun konumundaki akış hızının x ve y bileşenleridir, K_l pozitif kazançtır, θ_l lider robotun gerçek yönüdür, ve θ_{ld} onun istenilen yönüdür

$$\theta_{ld} = \mod \left(\arctan\left(\frac{v}{u}\right), 2\pi\right) \tag{4.4}$$

Takipçi robotlar akış çizgileri kullanırken sürünün şeklini korumak için, akış hızıyla potansiyel fonksiyonun negatif gradyanın toplamını takip etmeleri gerekir

$$\dot{p}_i(t) = \begin{bmatrix}\dot{x}_i(t)\\ \dot{y}_i(t)\end{bmatrix} = G_i(p) = \begin{bmatrix}u_{if} - \nabla J_{x_i}(p)\\ v_{if} - \nabla J_{y_i}(p)\end{bmatrix} \tag{4.5}$$

Burada $\nabla J_{x_i}(p)$ ve $\nabla J_{x_i}(p)$ potansiyel fonksiyonun gradyanın x ve y bileşenleridir, ve u_{if} ve v_{if} robot i konumundaki akış hızın x ve y bileşenleridir. Takipçi robotların denetim girişleri aşağıdaki fonksiyonlarda gösterilmektedir

$$v_i(t) = \|G_i(p)\| \tag{4.6}$$

$$\omega_i = -K_i \mod ((\theta_i - \theta_{id} + \pi, 2\pi) - \pi) \tag{4.7}$$

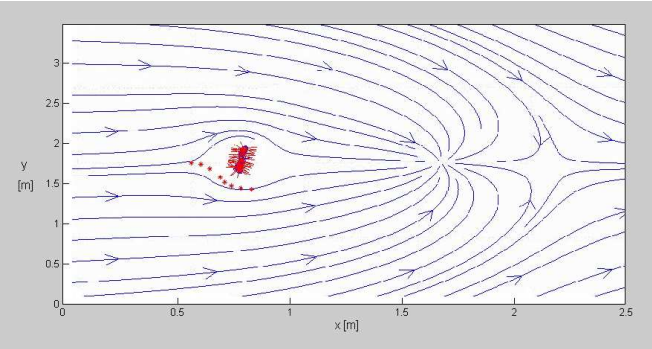
Burada $K_i > 0$ bir katsayıdır. Robotların istenilen yönleri aşağıdaki denklemle bulunur

$$\theta_{id} = \mod \left(\arctan \left(\frac{G_{yi}(p)}{G_{xi}(p)} \right), 2\pi \right) \tag{4.8}$$

4.4. Uygulama Sonuçları

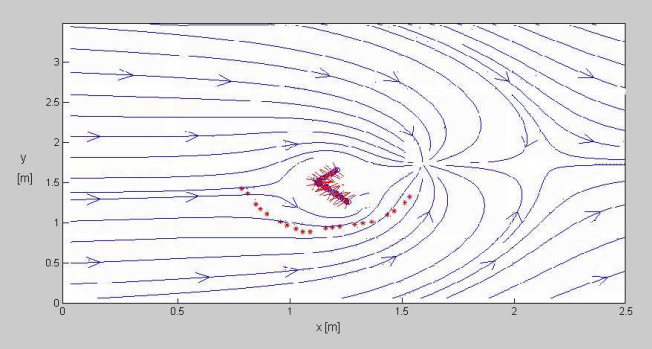
Bu uygulamada üç robot kullanılmaktadır. Lider robot ortamı tanır, takipçi robotlara engel bilgisini yollar ve panel metodu kullanarak akış çizgilerini bulur ve onları gezinmek için kullanır. Takipçi robotlar liderden gelen engel bilgileri ve konum bilgilerini hem akış çizgileri kullanarak gezinmek hem de yapay potansiyel fonksiyonlar kullanarak sürünün şeklini korumak için kullanır.

Robot sürüsünün ikizkenar bir üçgen oluşturması beklenmektedir. Üçgenin eşit kenarları (lider-takipçi1 ve lider-takipçi2) 35 cm uzunluktayken kalan kenarın uzunluğu (takipçi1-takipçi2) 30 cm'dir. Bu şekli sağlamak için lider ve takipçilerin arasındaki potansiyel fonksiyonun kazancı $a_{ij} = 4$ iken, takipçilerin arasındaki potansiyel fonksiyonun kazancı $a_{ij} = 1$ olarak ayarlanmıştır. Dahası, (4.3)'teki lider açısal hızın kazancı $K_l = 1$ olarak seçilirken, (4.7)'de bulunan takipçi açısal hızın kazancı $K_l = 0.225$ olarak seçilmiştir.

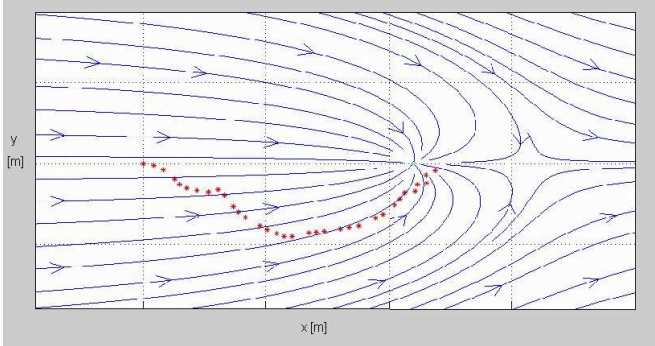


Şekil 4.3. Birinci engeli aşarken robot güzergahı

İlk denemeler tek robot ile iki engel içeren ortamda yapılmıştır. Şekil 4.3.'de robotun gördüğü ilk engelin etrafındaki hesaplanan akış çizgileri gösterilmektedir. Robot kendisine en yakın çizgiyi takip ederek engeli aşmayı garantiler. İlk engeli aşmadan onun arkasında bulunan ikinci engel gözükmediği için ilk engelin arkası bilinmeyen bölgedir. Engeli aştıktan sonra ikinci engel ortaya çıkar ve robot yeni ortam haritasına göre Şekil 4.4.'te gösterilen akış çizgilerini çizer. İkinci engeli aştıktan sonra akış



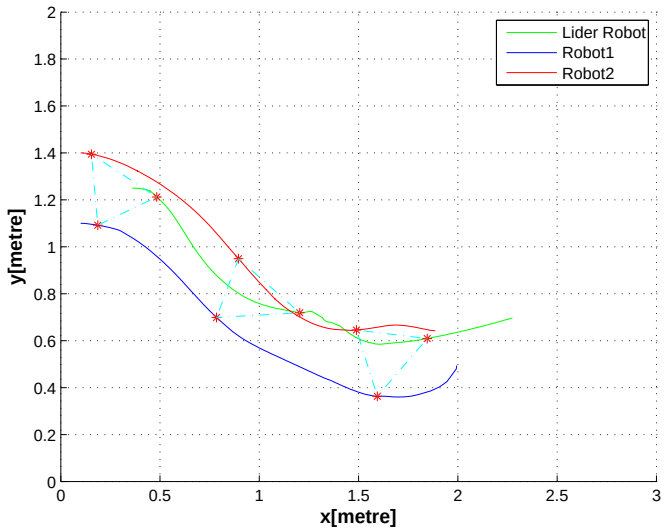
Şekil 4.4. İkinci engel algılanmış ve güzergah değişmiş



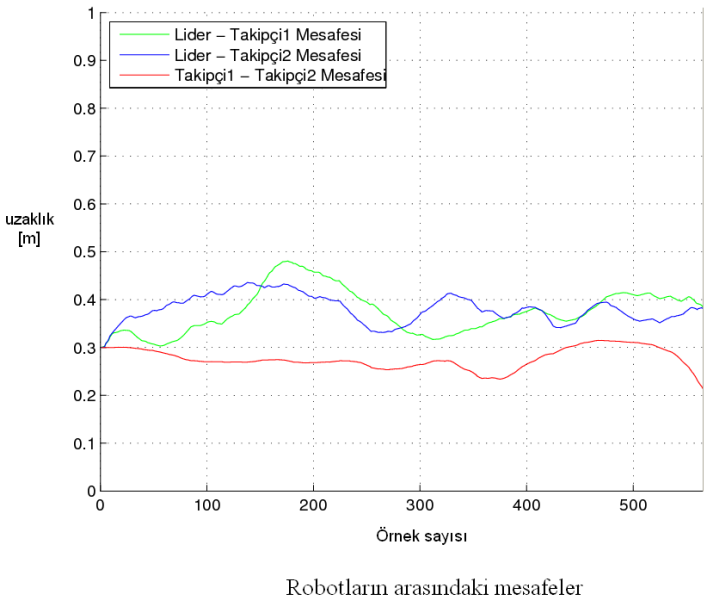
Şekil 4.5. İkinci engelden sonraki güzergah (engel algılanmamış)

çizgileri Şekil 4.5.'teki gibi olmuştur. Şekillerdeki kırmızı noktalar robotun başlangıç noktasından hedef noktasına takip ettiği yolu gösterir.

Şekil 4.6.'dan Şekil 4.9.'a kadar olan şekiller iki deneyin sonuçları gösterilmektedir. Şekil 4.6. ve Şekil 4.7. robotların kör konumlandırmalarını kullanarak çizilmiştir. Şekil 4.8. ve Şekil 4.9.'daki çizimler ise tepe kamerası kullanarak çizilmiştir. Şekil 4.6.'da erkinlerin belli zaman dilimlerinde alınan konumlarından takip edilen yolları gösterir. Şekil 4.7. erkinlerin arasındaki mesafeleri örnek sayısına göre gösterilmiştir. Şekillerden görüldüğü gibi robotların arasındaki mesafeler ve sürünün üçgen şekli küçük hatalar ile gezinme sırasında korunmuştur. Hatalar erkinlerin arasındaki mesafeler doğrudan ölçülmediği, kablosuz iletişim ile alınan konum bilgilerinden hesaplandığından kaynaklandıgı öngörülmektedir. Kablosuz iletişimde gecikmeler bulunmaktadır. Laboratuvarda yapılan deneyler, iletişim zamanının 0.007 ile 0.5 saniye arasında değiştiğini gözlemlenmiştir. Bir deneyin toplam zamanı 150 saniye civarında olup robotların ortalama hızı 0.016 m/s'dir.

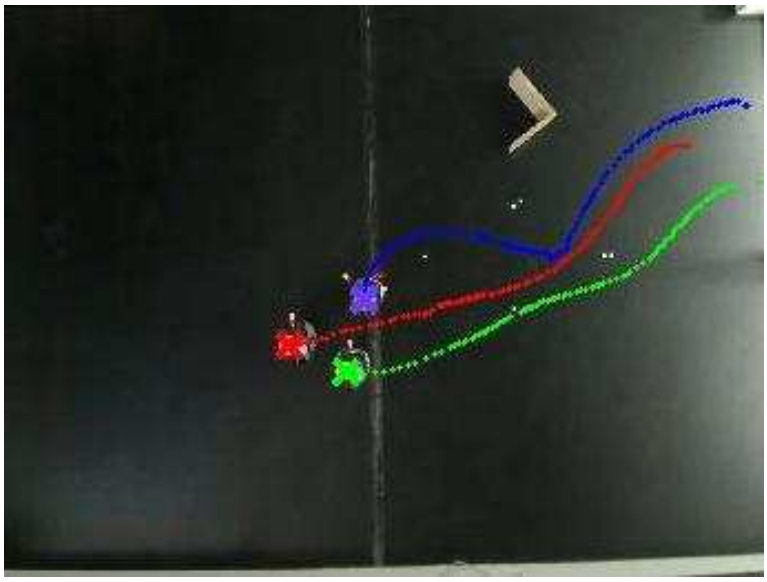


Şekil 4.6. Robotların güzergahları

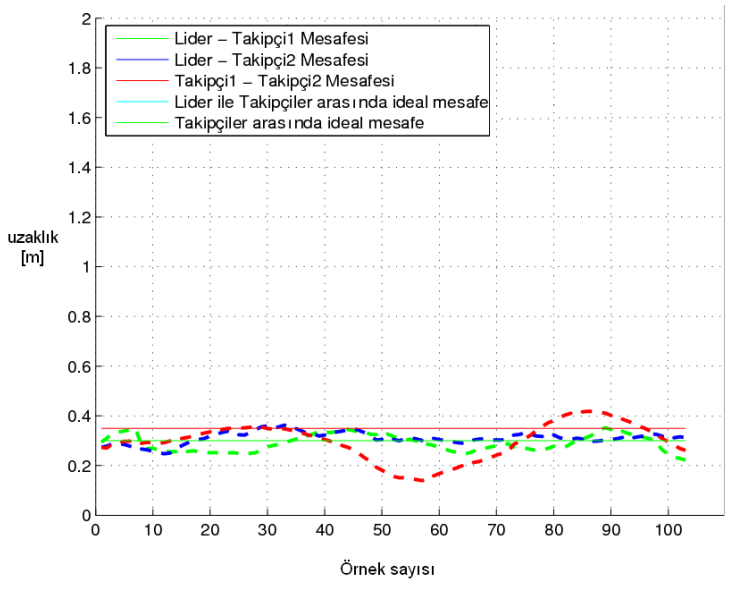


Şekil 4.7. Erkinlerin arasındaki mesafeler

Şekil 4.8.'de robotların gezinme yolları tepe kamerasından çekilen arenada gösterilmektedir. Bu yollar deneyin çekilen videosunu işleyerek çıkarılmıştır. Şekil 4.9.'da bu deneydeki erkinlerin arasındaki mesafeleri göstermektedir. Sürü elemanlarının arasındaki mesafeler ve sürünün genel şeklinin korunduğu görülebilmektedir.



Şekil 4.8. Robotların güzergahları



Şekil 4.9. Erkinlerin arasındaki mesafeler

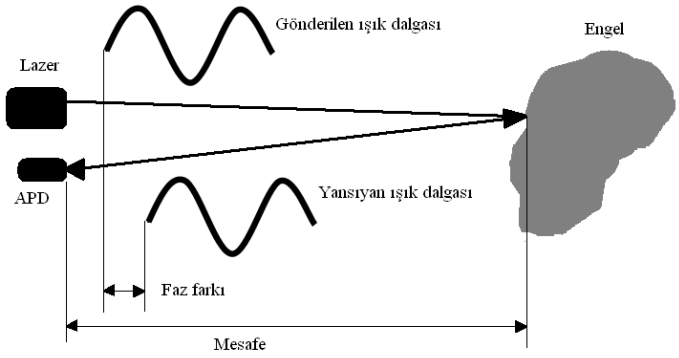
BÖLÜM 5

5. ALGORİTMA GELİŞTİRME YÖNTEMLERİ

Seyrüsefer algoritmaları geliştirmek için iki yöntem kullanılmıştır. Birinci yöntem bağıl konumlandırma tabanlıdır. Sürü erkinleri eğer birbirlerini görebilirlerse iletişime ihtiyaç kalmaz ve bu nedenle oluşan gecikmelerden kaynaklanan hatalar önlenmiş olunur. Bağıl konumlandırmada robotlar etrafı görecek, diğer sürü arkadaşlarını tanıyacak ve onların konumlarını kendilerine göre bulacaklardır. Gözlemci robotun konumu bağıl konumlara eklenirse, görünen robotların evrensel konumu bulunmuş olur. İkinci yöntemde sürünün her elemanına bir tekillik noktası eklenmiştir. Her robota bir girdap veya kaynak noktası eklenmiştir, ve akış çizgileri birbirleriyle kesişmediği için robotların arasındaki çarpışmalar daha etkili önlenmiş olur.

5.1. Bağıl konumlandırma

Hokuyo URG-04LX lazer tarayıcısı mini-robot uygulamaları için uygun ve iki boyutlu mesafe algılayıcıdır. Bu algılayıcının menzil tespiti ışık dalgalarının genlik modülasyonu (AM) ve gönderilen ışık ile yansıyan ışık arasındaki faz farkını algılamaya dayalıdır. Bu algılayıcı hakkında daha fazla bilgi için [28]'e bakılabilir. Lazer tarayıcısı 46.55 ve 53.2 MHz'lik olmak üzere iki farklı ışık dalgası kullanmaktadır. Yansıyan ışığın şiddeti çok yüksek olursa tarayıcının içindeki algılayıcı doyum noktasına ulaşabilir ve gelen ışık şiddeti hissedilmeyebilir. Doyma durumunu önlemek için ışık algılayan Avalanş Fotodiyod'un kazancı (APD) AGC devresi (Auto-Gain Control) yardımıyla düşürülür. Böylece daha geniş algılama erimi sağlanmış olur. Öte yandan, yansıyan çok düşük şiddetli ışıkların görülebilmesi için AGC devresi APD'nin kazancını düşürür. Başka deyişle, lazer tarayıcının AGC kazancı yansıyan ışığın şiddetini yüksek olup olmadığını söyler. Tarayıcının bu özelliği sürüde bulunan başka robotları tanımak ve konumlarını bulmak için kullanılmıştır. Normalde URG-04LX lazer tarayıcısı sadece mesafe taraması yapmaktadır ve yansıyan ışık siddeti hakkında bilgi vermemektedir. Lazer algılayıcısının APD kazancı bir şekilde okunabilirse yansıyan ışığın şiddeti hakkında bir fikir edinilebilir. APD kazancını okuyablimek için lazeri geliştiren Hokuyo şirketiyle temasa geçilmiş ve kazancı elde etmek için gereken komutlar elde edilmiştir. Lazer sürücüsü programları değiştirilmiş ve yansıyan ışık şiddetini veren bir fonksiyon geliştirilmiştir. Mesafe ölçme algoritmasının çalışma mantığı Şekil 5.1.'de gösterilmektedir.



Şekil 5.1. Mesafe ölçme algoritmasının görselleştirilmesi([28]'den uyarlanmıştır)

Sürüde bulunan diğer robotları algılamak için ilk önce robotlar ortamda bulunan başka nesnelerden ayrılmıştır. Ondan sonra robot kimlikleri kullanarak robotlar tanınmıştır. Her robota bir kimlik sağlamak için geri-yansıtıcı[1] barkodlar kullanılmıştır. Geri-yansıtıcı maddeler gelen bir ışığın geliş açısından bağımsız bir şekilde ve en az dağılmayla kaynağına doğru yansıtır. Bu maddelerden yansıyan ışık yüksek şiddete sahiptir; robotları etraflarda bulunan nesnelerden ayırmak için bu özellik kullanılmıştır.

Sürüde bulunan her robot kendi kimliğini oluşturan geri-yansıtıcı barkoda sahiptir. Etrafta bulunan robotları tanımak için her robot lazer tarayıcısını kullanarak ortamda yoğunluk taraması yapar, yüksek şiddetli nesneleri ayırır, barkodları okur ve sürüde bulunan komşu erkinlerini önceden bilinen kimliklerini kullanarak tanır (Lazer taraması hakkında daha fazla bigi [28] ve EK1'de bulunmaktadır). Görünen sürü robotların konumlarını bulmak için yine lazer tarayıcısı kullanılarak mesafe taraması yapılır. Robotların evrensel konumları gören robotun konumunu bağıl konumlara ekleyerek hesaplanmıştır. Bağıl konumlandırma fonksiyonu seyrüsefer algoritmasına çok büyük katkısı olmuştur. İletişim kullanarak yapılan konumlandırma fonksiyonunun zamanı düzensiz bir şekilde 0.1 ile 0.55 saniye arasında değişirken, bağıl konumlandırma fonksiyonunun ortamın taraması, robotların tanınması ve evrensel konumların hesaplaması için gereken zaman görünen barkodun uzunluğuna göre 0.11 ile 0.16 saniye arasında değişmektedir. Şekil 5.2.'de robotların kimliklerini belirlemek için kullanılan geri-yansıtıcılı barkodlar gösterilmektedir. Lider robot ortada farklı kimliği ile görünürken, iki tarafta aynı kimliğe sahip olan takipçi robotlar görülmektedir.

[1] ing.:retro-reflective

Şekil 5.2. Robotların geri-yansıtıcılı barkodları

Barkodlar robotların 360 derece etrafını sarmaktadır. Bu robotların tanınması için belli açıda görünmesi sağlamaktadır. Ayrıca, takipçi robotların lazer tarayıcıları ve barkodları birbirlerini görecek şekilde ama öbür takipçinin lazer ışıklarını engellemeyecek şekilde yerleştirilmiştir. Lazer tarayıcıları 0.36 derece hassasiyet ile 240 derece tarama yapabilir. Bu da robotların ölü bölgeleri olduğunu göstermektedir; robotlardan birisi bu bölgeye girerse robot algılanmaz ve bağıl konumu hesaplanamaz olur. Bu sorunu çözmek için özel durum algoritması geliştirilmiştir. Robotlardan birisi lazer tarafından algılanmazsa ya da hatalı bir konumlandırma olursa bu robotun konumu iletişim bilgisinden alınmaktadır. Bu iletişim bilgisi doğrudur ama gecikmeli olabilir ama sürünün dağılmasını engellemek için kulanılabilecek yeterli bilgidir. Robot tekrar görüldüğü zaman hemen bağıl konumu hesaplanır ve sürü şeklinde oluşan deformasyon hemen düzeltilir.

AGC kazancını okumak için lazer tarayıcısı yoğunluk taraması yapmaktadır. Yoğunluk taraması, mesafe taramasını tanımlamak ve gerçekleştirmek için gereken komutlardan farklı parametreler kullanarak yapılmaktadır. Yoğunluk taraması hakkında daha fazla bilgi EK1'de bulunmaktadır.

5.2. Robotlar Üzerine Kaynak Veya Girdap Potansiyel Fonksiyonu Eklemek

Tek sürüde bulunan insansız hava araçlarının çarpışmalarını önlemek için [10]'da her araca bir kaynak noktası konulmuştur. Gerçek zamanlı algoritmada, her araç kaynak

noktası olarak tanımlanmıştır ve (2.8)'de bulunan kaynak noktasının etkisi (2.3) denkleminin sağ tarafına eklenmiştir. Akış çizgileri birbirleri ile hiç kesişmediği için erkinlerin arasındaki olası çarpışmalar önlenmiş olur. Potansiyel akış robotların ortamda olmalarından aşağıda gösterildiği gibi değişmektedir.

$$u_{robot} = U_{serbest} + u_{kuyu} + \sum_{i=1}^{N} u_{ind} + \sum_{i=1}^{N} u_{robotlar} \tag{5.1}$$

$$v_{robot} = V_{serbest} + v_{kuyu} + \sum_{i=1}^{N} v_{ind} + \sum_{i=1}^{N} v_{robotlar} \tag{5.2}$$

Burada $\sum_{i=1}^{N} u_{robotlar}$ ve $\sum_{i=1}^{N} v_{robotlar}$, her robotta yerleştirilen kaynak noktaları tarafından indirgenmiş hızların x ve y bileşenleri toplamıdır. Bu şekilde, sürüde bulunan robotların çarpışma olasılığını düşürmek için her robota kaynak noktası eklenmiş ve etkisi deneysel bir şekilde gözlemlenmiştir.

Daha sonra, kaynak noktasının yerine her robota girdap noktası eklenmiş ve yapılan deneyler ile girdap eklenmenin etkisi incelenmiştir.

Şiddeti Γ_{girdap} olan ve (x_v, y_v) noktasında bulunan bir girdap fonksiyonun, (x, y) noktasındaki indirgenmiş hız etkisinin bileşenleri bu şekilde yazılır:

$$u_{girdap} = \frac{\Gamma_{girdap}}{2\pi} \frac{y - y_s}{(x - x_s)^2 + (y - y_s)^2} \tag{5.3}$$

$$v_{girdap} = -\frac{\Gamma_{girdap}}{2\pi} \frac{x - x_s}{(x - x_s)^2 + (y - y_s)^2} \tag{5.4}$$

5.3. Sonuçlar

Geliştirilen algoritma üç erkinli robot sürüsüne uygulanmıştır. Sürü hedefe doğru ilerlerken bir eşkenar üçgen şeklini korunması beklenmektedir. Sürünün istenilen şeklinde takipçi robotların görüş alanı lider robot tarafından kısıtlandığı için lider tabanlı gezinme uygulanmıştır. Lider robot önde bulunduğundan engel için ortamı tarayacak ve görülen engeller panel metodu kullanılmak üzere diğer robotlara tasarsız[2] kablosuz ağ yardımı ile gönderilmektedir. Robotların hesap gücü ve hızları düşük olduğu için sadece tek bir engel kullanılmıştır. Robotlarda daha güçlü ve hızlı işlemci kullanılırsa daha karışık ortamlar da kullanılabilir. Üstelik algoritma gerçek zamanlı olduğu için değişen ortamlarda da kullanılma kabiliyeti vardır. Burada gösterilen

[2] ing.:ad hoc

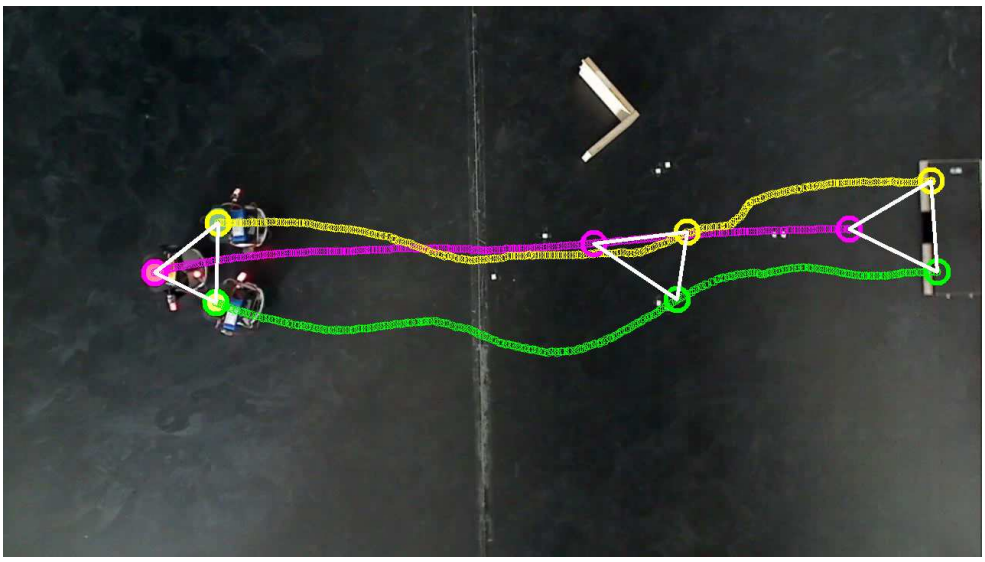
sonuçlar deneyler sırasında çekilen videyoların işlenmesinden elde edilmiştir. Görüntü işleyen program C programlama dili ile yazılmıştır ve görüntü tanıma tabanı olarak OpenCV kütüphanesi kullanılmıştır. Sonuçlarda her robotun hedef noktasına kadar izlediği yol ile birlikte robotların arasındaki mesafeler gösterilmektedir.

Şekil 5.3., Şekil 5.4., Şekil 5.5. ve Şekil 5.6.'da bağıl konumlandırma ile geliştirilmiş panel metodu seyrüsefer algoritması sonuçları gösterilmektedir. İlk deneyde robotlar sürü halinde engeli aşarak önde bulunan hedef noktasına kadar ulaşmıştır. Üstelik sürü erkinlerin arasındaki mesafeler neredeyse istenilen mesafeleri korumaya başarmıştır. Sürüdeki erkinlerin komşularını geç algılaması mesefelerde gorünen salınımlara yol açmıştır. Sürüde bulunan bir robot arkadaşlarının uzaklaştığını biraz geç farkeder, sürünün şeklini korumak için hızlı bir şekilde denetimi uygulanmıştır. Hızlı kontrol robotu sürüdeki arkadaşlarına çok yaklaştırır ve robot sürünün şeklini korumak için tekrar uzaklaşmaya başlar. Mesafelerde gorulen salınımları önlemek için bagıl konumlandırma algoritması hızlandırıması gerekmektedir.

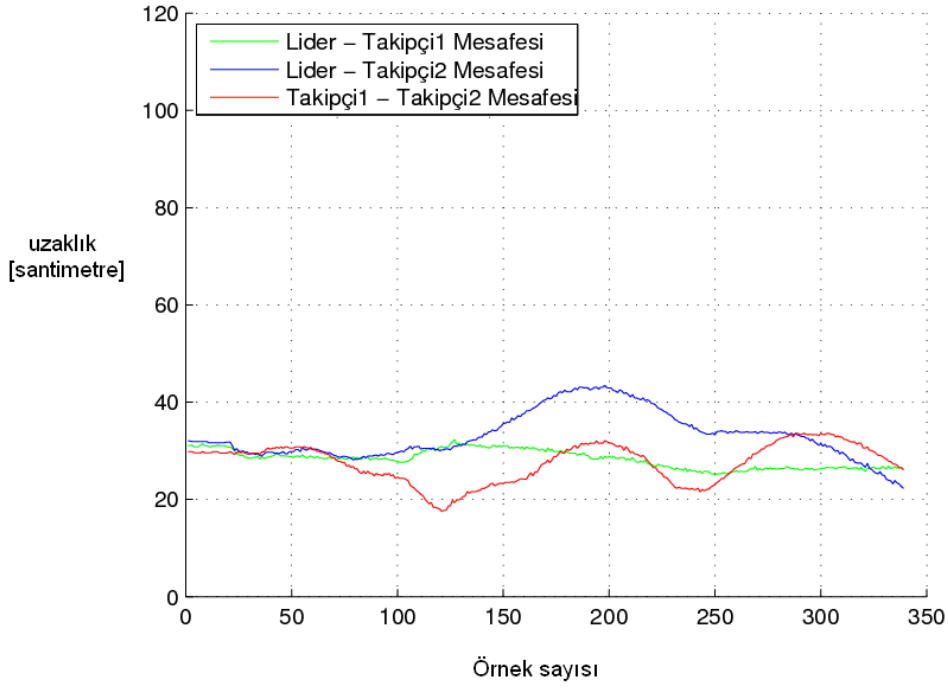
Şekil 5.3., Şekil 5.4. ve Şekil 5.5.'de gösterilen sonuçlarda deneylerde kullanılan potansiyel fonksiyonun kazancı yüksek seçilmiştir. Denklem (4.5)'teki potansiyel hızlar 1 kazancı ile çarpılmıştır. Kazancın yüksek olması sürünün istenilen şekle hemen ulaşmasını ve deney sırasında bu şeklin korunmasını sağlamıştır, ama öbür yandan da salınımlara yol açmıştır.

Şekil 5.6.'de gösterilen sonuçlarda, potansiyel fonksiyonun kazancı düşük seçilmiştir. Denklem (4.5)'teki potansiyel hızlar 0.1 ile çarpılmıştır. Robotlar, sürü şeklinde oluşan bir bozukluğu yavaş bir şekilde toparlamaya başlamıştır. Robotların arasındaki mesafeler kısmından görüldüğü gibi salınımlar azalmıştır, ama engelden sonra sürünün istenilen şekle dönmesi için çok uzun zamana ve bu sebeple de mesafeye ihtiyaç duyulmuştur.

Robotlara kaynak ve girdap potansiyel fonksiyon eklenen deneylerin sonuçları Şekil 5.7. ve Şekil 5.8.'de gösterilmektedir. Şekil 5.7.'de sürünün erkinlerine kaynak noktaları eklenmiştir. Denklem (4.5)'teki potansiyel hızlar 0.5 ile çarpılmıştır. Bu durumda salınımlar az görülmektedir. Robot sürüsü istenilen şekli koruyarak ve önceki deneylerde (kaynak ve girdap eklemeden) görülen daha az salınımlar ile hedef noktasına ulaşmıştır. Şekil 5.8.'da robotlara girdap noktaları eklenmiştir. Bu durumda robotlar arasındaki mesafelerde bulunan salınımlarda bir azalma görülmemiştir. Kaynak noktası eklemek daha etkili olmaktadır. Robotlar kaynak etkisi ile çok yaklaştıklarında birbirlerini itmektedirler.

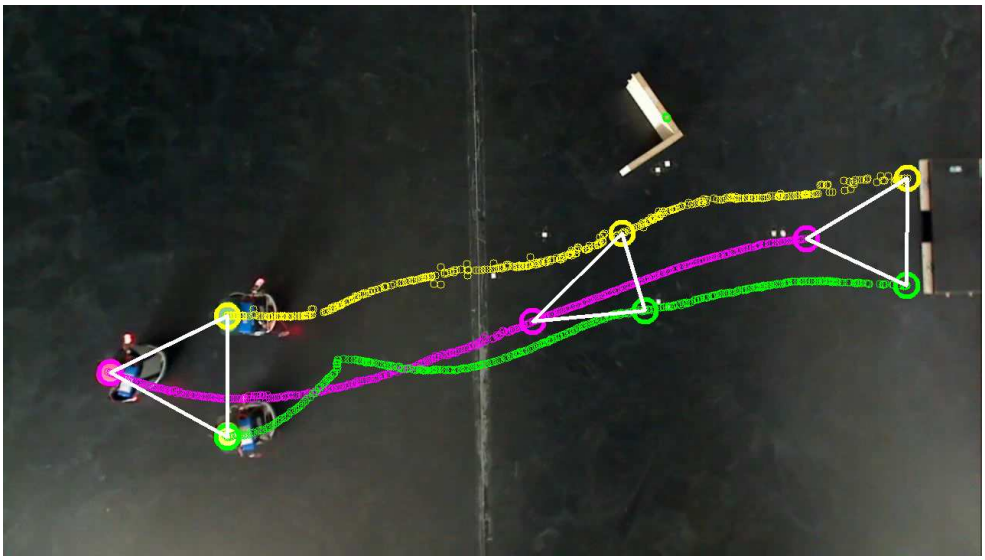


a- Robotların güzergahları

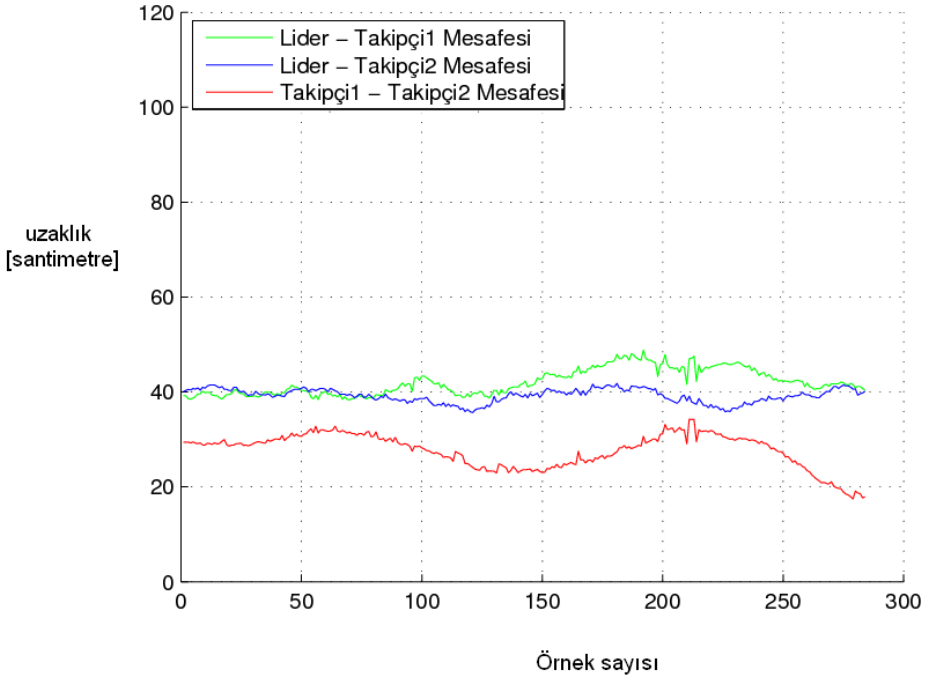


b- Robotların arasındaki mesafeler

Şekil 5.3. Bağıl konumlandırma - deney 1

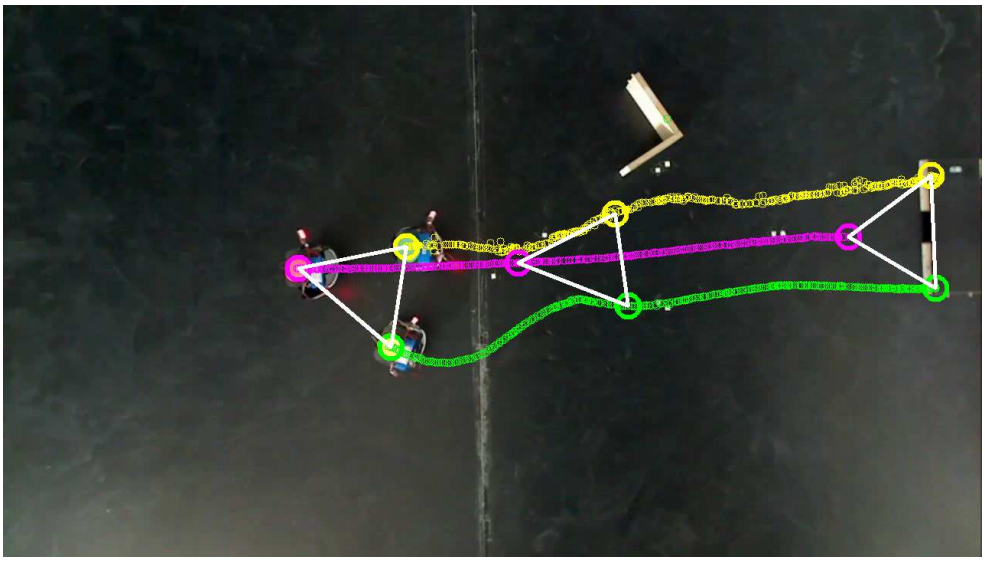


a- Robotların güzergahları

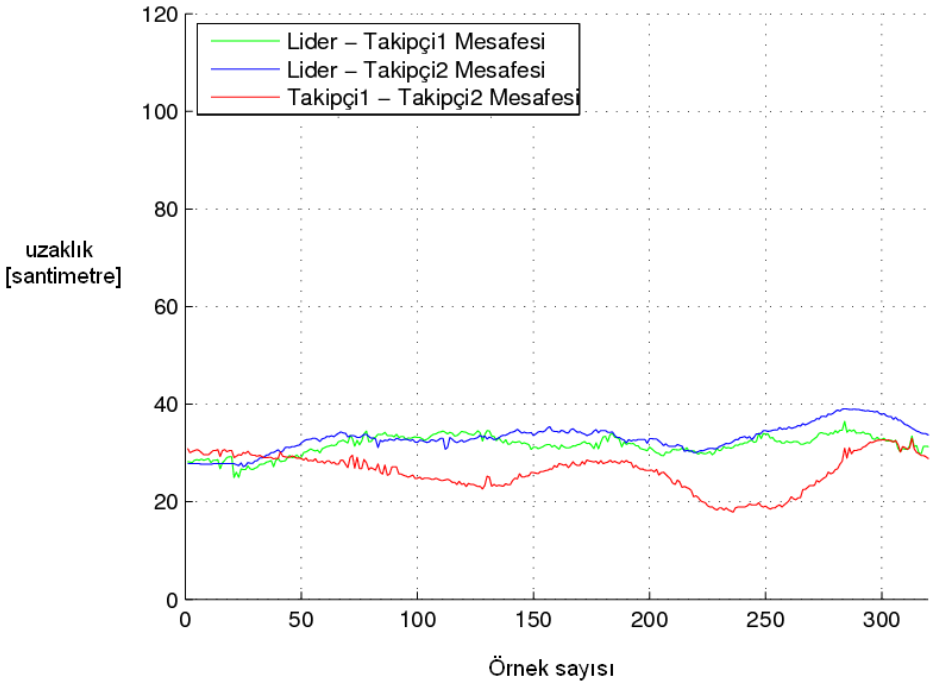


b- Robotların arasındaki mesafeler

Şekil 5.4. Bağıl konumlandırma - deney 2

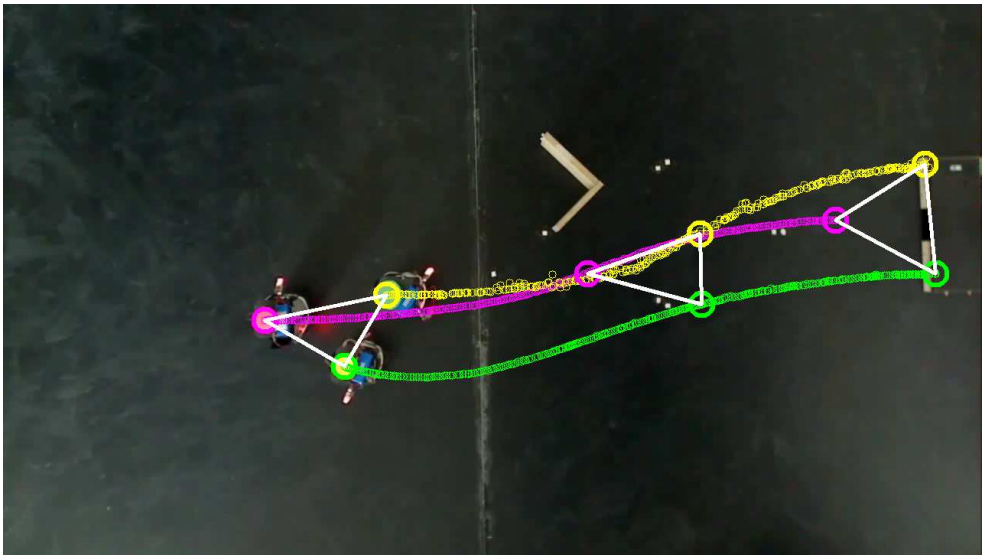


a- Robotların güzergahları

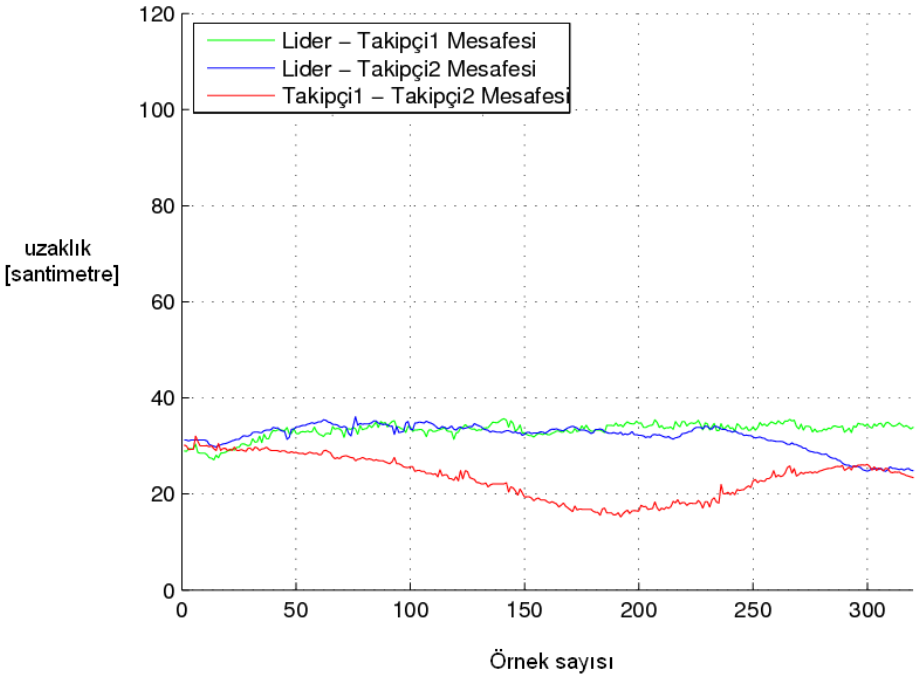


b- Robotların arasındaki mesafeler

Şekil 5.5. Bağıl konumlandırma - deney 3

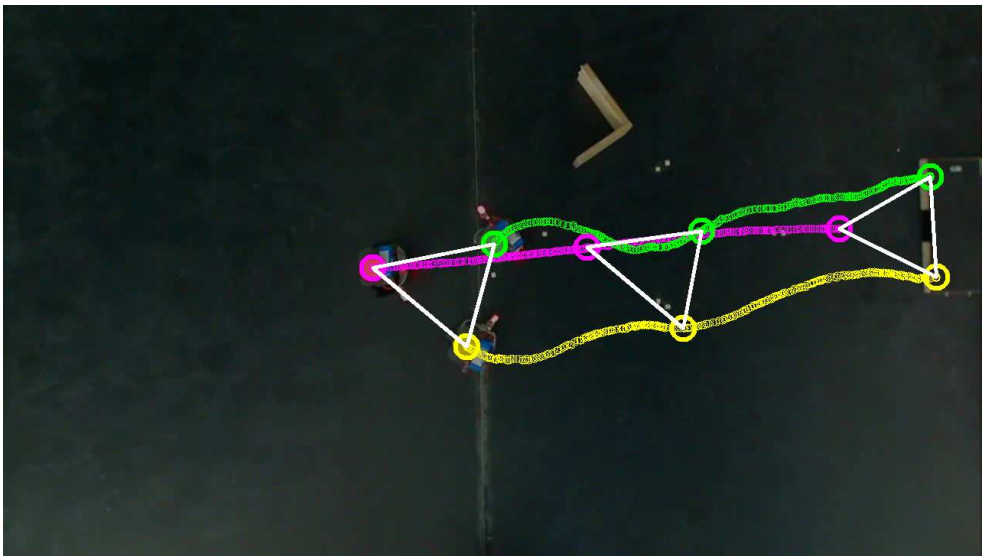


a- Robotların güzergahları

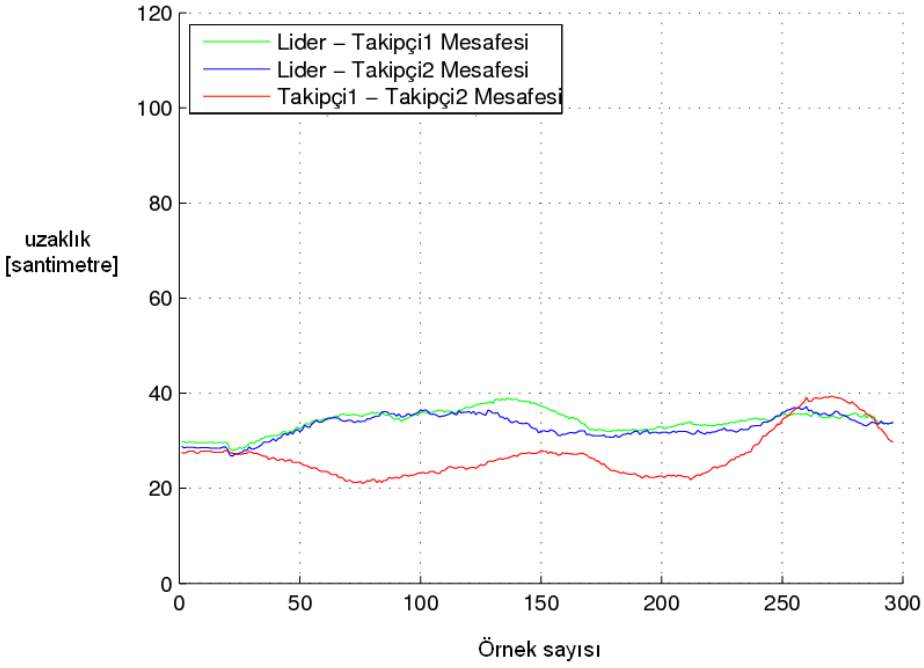


b- Robotların arasındaki mesafeler

Şekil 5.6. Bağıl konumlandırma - deney 4

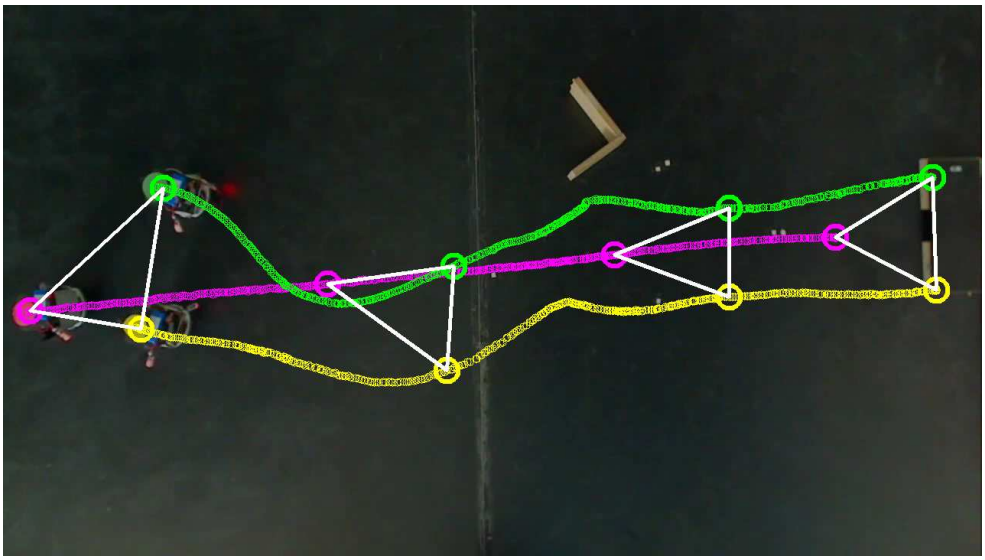


a- Robotların güzergahları

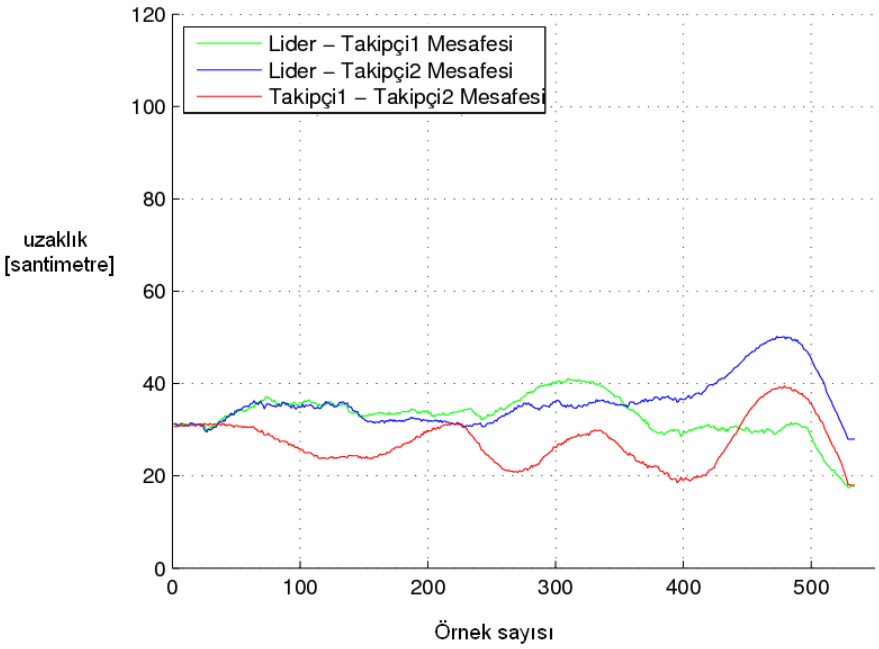


b- Robotların arasındaki mesafeler

Şekil 5.7. Kaynak noktası deneyi



a- Robotların güzergahları



b- Robotların arasındaki mesafeler

Şekil 5.8. Girdap noktası deneyi

BÖLÜM 6

6. SONUÇ VE DEĞERLENDİRME

Bu tez çalışmasında potansiyel akış çizgilerinin robot sürüsü seyrüseferinde yol bulma algoritması olarak kullanılabildiği deneysel bir şekilde gösterilmiştir. Akışkanlar mekaniğinde potansiyel akış analizlerinde kullanılan panel metodu robot sürüsü seyrüsefer algoritma geliştirmesinde kullanılmıştır. Robot sürüsü bulunduğu ortamda bulunan engellere takılmadan istenilen hedef noktasına ulaşmıştır. Panel metodundan gelen akış çizgileri robotların yolları olarak kullanılmıştır. Potansiyel akışlar gerçek akışkanların hareketlerini gösterdiği için robotlara çok doğal ve düzgün yollar vermektedir. Bu algoritma ile hesaplanan yollar insansız hava araçları sürülerinin gezinmesi için kullanılabilir.

Robot sürüsünün hedef noktasına doğru ilerlerken önceden belirlenmiş bir şekli oluşturması ve koruması için potansiyel fonksiyonlar da kullanılmıştır. Potansiyel fonksiyonlar robotların arasındaki mesafeleri algılayarak sürünün istenilen şekli oluşturması için uygulanması gereken denetleyiciyi hesaplamak için kullanılır. Sürü erkinlerine hem potansiyel fonksiyonlardan hem de panel metodundan gelen denetim uygulandığında sürü istenilen dizilimi koruyarak ve akış çizgilerini kullanarak hedefe doğru seyrüsefer yapmıştır.

Algoritmanın çevrimdışı ve çevrimiçi olarak iki versiyonu geliştirilmiştir. Çevrimdışı versiyonunda ortamın haritasının önceden bilindiği varsayılmıştır. Deneylerde e-puck mini robotlar kullanılmıştır. Robotların kendi ve birbirlerine göre konumları tepe kamerasından alınmıştır. Çevrimdışı deneylerinde masaüstü bilgisayar kullanıldığı için hesaplama problemleri yaşanmamıştır. Panel metodu çok işlem gerektiren bir yöntem olmasına rağmen bilgisayarın işlemcisi tüm işlemlere yetişmiştir. Ayrıca sadece başında deneylerin bir defa hesaplanıyor. Deneylerde robotların konumları MATLAB görüntü tanıma yöntemleri ile hesaplanmıştır. Sürü erkinlerin arasındaki mesafeler, sürünün şeklini korumak için kullanılan potansiyel fonksiyonlar etkileri, akış çizgileri ve onlardan gelen kontrol girdileri bilgisayar tarafından hesaplanmıştır. Robotlara kontrol girdileri bluetooth vasıtasıyla yollanırken gecikme yaşanmıştır. Bu gecikme robotların şekil oluşturmalarını biraz etkilerken engellerden kaçınmalarını etkilememiştir.

Çevrimiçi algoritmada ortamın hakkında önceden hiç bilgi bulunmadığı varsayılmıştır. Robotlar tamamen bilinmeyen bir bölgede kendi algılayıcılarını kullanarak, sürü şeklini bozmadan engeller arasında hedefe kadar yolu bulmaları gerekmektedir. Bu algoritmayı kullanan deneylerde KheperaIII robotları kullanılmıştır. Ortamı keşfetmek ve bulunan engelleri algılamak için Hokuyo URG-04LX lazer tarayıcısı kullanılmıştır. Sürünün istenilen şeklinden kaynaklanan arkada bulunan robotların görüş alanların daralmasının önüne geçmek için lider tabanlı tarama kullanılmıştır. Lider robot ortamı tarar ve takipçi robotlara görünen engellerin bilgilerini yollamaktadır. Lider robot engel bilgilerine panel metodu uygulayarak hedefe kadar giden akış çizgilerini bulmaktadır. Bu akış çizgileri lideri ortamda bulunan engelleri aşarak hedefe doğru götürmektedir. Lider robot ortamda ilerlerken yeni bir tarama yapar, engelleri ayıklar ve akış çizgilerini güncellemektedir. Algoritmanın çevrimiçi özelliği bu işlemden kaynaklanmaktadır. Takipçi robotlar liderden gelen engel bilgisinin yanı sıra birbirlerinin arasındaki mesafeleri kullanarak hem akış çizgileri hem de sürü dizilimini korumak için potansiyel fonksiyonları kullanmaktadır. Sürü robot önceden belirlenmiş eşkenar üçgen şeklini koruyarak hedef noktasına ulaşmayı başarmaktadır. İletişim gecikmelerinden ve robot işlemcilerin sınırlı performanslarından bir miktar hata oluşmasına rağmen sonuçlar başarılı olmuştur.

Algoritmayı geliştirmek için ve hataları azaltmak için iki yönteme başvurulmuştur. Birinci yöntem iletişimden kaynaklanan gecikmeleri önlemek için geliştirilmiştir. Bu yöntemde bağıl konumlandırmaya dayanarak sürüde bulunan öbür robotların konumları birbirlerine göre hesaplanmıştır. Robotların birbirlerini görmek için yine üzerlerinde bulunan lazer tarayıcısı kullanılmıştır. Her robota bir geri-yansıtıcı barkod kimliği tasarlanmıştır. Robot, sürü arkadaşlarını tanıması için lazer tarayıcısı kullanarak ortamın yoğunluk taramasını yaptıktan sonra barkodları çevrede bulunan engellerden ayırır ve tanıma işlemine geçer. Robotlar tanındıktan sonra robot bir mesafe taraması yapar ve robotların göreceli konumlarını hesaplar. Bu konumlardan genel konumlar hesaplanır ve potansiyel fonksiyonlarda kullanılır. Bu yöntem, algoritmada bulunan gecikmeleri engelleyerek sonuçları çok iyi etkilemiştir. Algoritma geliştirmesinde kullanılan ikinci yöntem ise sürüde bulunan her robota bir kaynak veya girdap noktası eklemektir. Akış çizgileri doğal akışkanların hareketlerini temsil ettiği için birbirini kesmemektedir. Birinci deneyde her robota bir kaynak noktası, ikinci deneyde ise girdap noktası eklenmiştir. Robot sürüsü istenilen şekli koruyarak ve önceki deneylere göre daha az salınımlar ile hedef noktasına ulaşmıştır. Kaynak noktası eklemenin daha etkili olduğu deneysel sonuçları ile gözlemlenmiştir.

6.1. Gelecek Çalışmalar

Her çalışmada olduğu gibi, bu çalışma da geliştirilmeye müsait yönleri ve gelecekte çalışma yapılabilecek konular vardır. Bu konuların çoğu kullanılan malzemeler ve tabana yönelik konulardır. Panel metodun en büyük dezavantajı yüksek işlem gücü gerekmesidir. Panel metodun kısıtlı ya da yavaş işlemciyle kullanılması sistemin yavaşlamasına yol açar. Bu çalışmada geliştirilen algoritmada panel metodu belli zaman aralıklarla uygulanmaktadır. Gezinme algoritması daha gerçekçi olması için ve ortamda oluşabilecek değişiklikleri hemen algılaması için daha hızlı işlemci ile kullanılabilir. Kullanılan KheperaIII robotların işlemcisi değiştirilemediği için, panel metodu ile ilgilenen ikinci işlemci kullanılabilir. Lazer algilayıcısı kontrol eden kart tasarlanır ve robotlara eklenebilir. Ortamı ayıklamak için bir GPU (Graphical Processing Unit) kartı da kullanılabilir. Böyle bir uygulamada lazer kartı ortamı tarar, engelleri ayıklar ve robot işlemcisin isteği üzerine robota engel bilgisini yollar. Robotun işlemcisi sadece itim-çekim kuvvetleri hesaplamaları ve gezinmeyle uğraşırken lazer kartı ortamın keşfiyle uğraşır. Bu tür modüler uygulama algoritmaya çok zaman kazandıracak ve gerçekçi gerçek zamanlı olacaktır. Bu durumda 45^o derece görüş kısıtlamasına gerek kalmaz ve robotlar taramadan alınan tüm bilgileri kullanır.

Gerçek zamanlı algoritma lider tabanlı algoritmadır; takipçi robotlar ortamın bilgisini lider robottan almaktadır. Lider robotta bir problem çıkarsa tüm sistem durabilir. Üstelik engel bilgisi lider robotun perspektifindendir ve takipçilere atarken kablosuz iletişimi kullandığı için yavaşlama sözkonusu. Bu problemlerin önünü kesmek için sürüde bulunan her robot kendi ortam haritasının çıkarması daha uygun olabilir. Robotların kapattığı engelleri çıkarmak için bir yöntem geliştirilmelidir. Sürü robotları ortam bilgilerini paylaşır ortamın tam haritasını çıkarırsa daha gerçekçi bir algoritma olabilir.

EK 1

A Hokuyo Lazer Tarayıcısı

Hokuyo URG-04LX lazer tarayıcısı küçük, hafif, performanslı ve düşük güç tüketen algılayıcıdır. Lazerin boyutları $50\text{mm}\times50\text{mm}\times70\text{mm}$, menzili 4 m ve güç harcaması 2.5 W'lıktır ve küçük robot uygulamalarında ideal boyutlardadır. Lazerin görüş açısı 0.36^o çözünürlük ile 240^o'dir. Lazerin hassasiyeti ise 1 mm'dir.

A1. İletişim

URG-04LX lazer tarayıcısı RS-232C ya da USB vasıtasıyla bağlanmaktadır. lazerin $19.2Kbps$, $57.6Kbps$, $115.2Kbps$, $250Kbps$, $500Kbps$ ya da $750Kbps$ yüksek hızlarla haberleşme yapma imkanı vardır. Lazer tarayıcısı bilgisayardan ya da kontrol eden nesneden gelen komutlar ile kontrol edilmektedir. İletişim formatı ise "Komut, Parametreler, Bitirme Komutu" şeklindedir. Bitirme komutu Hex formatında [0a] (ASCII: LF) ya da [0d] (ASCII: CR)'dır.

A1.1. Açma Kapatma Komutları

Lazer tarayıcısı bilgisayara ya da kontrol eden nesneye bağlandıktan sonra açılması gerekir. Açma/kapatma komutu "L" komutu olarak bilinmekte ve ASCII formatı "[L] [1 byte] [LF] (ya da [CR]) " Hex formatında açma komutu ise "[4c] [01] [0a] (ya da [0d])" şeklindedir. lazeri açmak için parametre kısmında [01] ve kapatmak için parametre kısmında [00] göndermemiz gerekir.

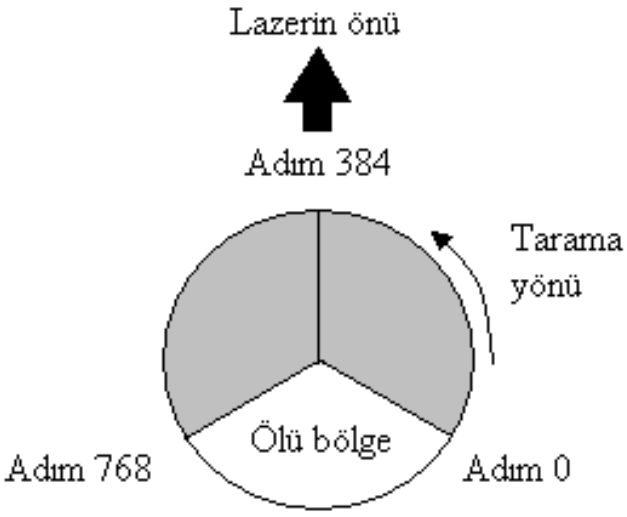
A1.2. İletişimi Düzenleme Komutu

lazeri bağlayıp açtıktan sonra ve iletişime geçmeden önce iletişim bilgilerini düzenlemek gerekir. İletişimi düzenleme komutu: "S" komutu olarak bilinmekte ve ASCII formatı bu şekildedir: "[S] [HIZ (6 bit)] [7 boş bit] [LF]/[CR]". Bu komutun Onaltılık tabanda formatı bu şekildedir "[53] [HIZ (6 bit)] [Boş (7 bit)] [0d]". Tarayıcının bu komuta olan tepkisi bu şekildedir "[S] [Hız (6 bit)] - [Boş (7 bit)] [Durum] [LF]/[CR]". Komutta bulunan iletişim hızı ASCII karşılığı yazılır (örn. $57.6Kbps$ -> ASCII: "057600").

A1.3. Mesafe Ölçme Komutu

lazerin hassasiyeti 1mm iken menzili 4m'dir. Her mesafe bilgisi 12 bit (0 ile 4095 arasında) ile gösterilmektedir. İletişimde mesafe bilgisine ayrılan alanı küçültmek için 6-bitlik iki tabanlı bilgi 1-bayt karaktere çevirilmiştir. Kodlama işi bu şekildedir: 12 bitlik bilgi 6'şar bite bölüp 30Hex eklenmiştir. Örneğin 1234 mm'lik mesafe bilgisi iki tabanda "010011010010" gösterilmektedir. Bu bilgi iki 6'şar bitlik bilgiye ayrıldıktan sonra (ikilik tabanda (010011,010010) ya da Onaltılık tabanda (13,12))

30Hex eklenmektedir. 1234 mm'lik bilginin karşılığı (43,42) ya da (C,D)'dir. Bilgiyi okumak için kodlamanın tersi yapılır. Okunan bilgiden 30Hex çıkartılıp iki bilgi big-endian sistemi kullanarak birleştirilir.



Şekil A1. Lazer tarayıcısının görüş alanı

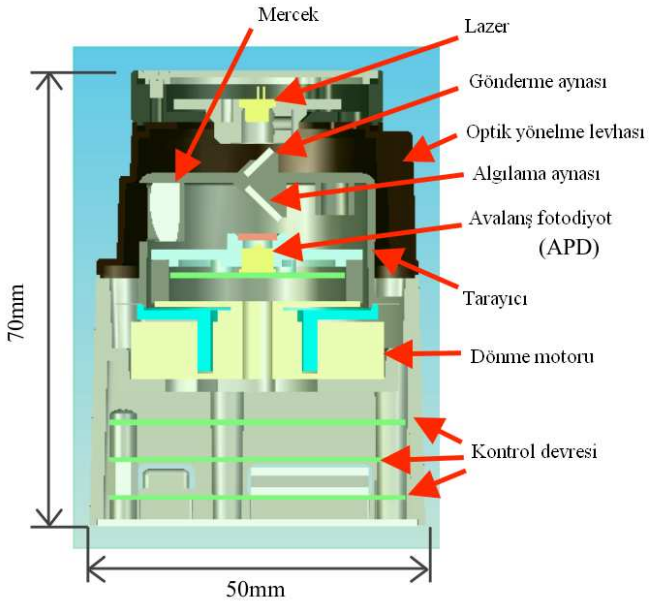
Mesafe bilgisi okumak için lazere bu komut yollanır "[G (47 Hex)] [Başlangıç noktası] [Bitiş noktası] [Demet sayısı] [LF]/[CR] ([0a]/[0d] Hex)". Lazerin tepkisi ise bu komutun aynısını tekrar yollar ardından da durum baytını ve mesafe bilgilerini ardarda yollamaya başlar. Komuttaki başlangıç ve bitiş noktaları taramak istediğimiz alanın başlangıç ve bitiş noktalarıdır. Bu alan Şekil A1.'de gösterilmektedir. Demet sayısı 0 ile 99 arasında değişen bir sayıdır. Demet sayısı 5 ise örneğin, lazer tarayıcısı tüm alanı beşer noktalara ayırır ve her beş nokta içerisinde en yakın noktanın uzaklığı geri gönderir.

Lazerden gelen mesafe bilgisi şu şekildedir. Eğer başlangıç adımı 44, bitiş adımı 725 ve demet sayısı 1 olarak seçilirse, lazer bize 682'lik bir vektör döndürür. Bu vektörün ilk elemanı 44. adımda bulunan noktanın uzaklığıdır. Vektörün ikinci elemanı ise 45. adımda bulunan noktanın uzaklığıdır. Adım 0 -135^o'ye tekabul ederken, adım 384 0^o'ye ve adım 768 $+135^o$'ye tekabul etmektedir. Tarama yaptıktan sonra her noktanın yönü ve uzaklığı elde edilir. Küçük hesaplamalar yaparak etrafta bulunan noktaların konumları bulunabilir.

A1.4. Yoğunluk Ölçme Komutu

Piyasaya sürülen Hokuyo URG-04LX lazer tarayıcıları sadece mesafe algılamak için kullanılabilir. lazeri geliştiren firma yansıyan ışık yoğunluğu algılamak için tarayıcının sürümünü geliştirmiş ama piyasaya sürmeden testler altına almış. Işık şiddetini okuyabilmek için Hokuyo firmasıyla temasa geçildi ve sürümde yapılması gereken

değişiklikleri öğrenildi.



Şekil A2. Lazer tarayıcısının iç yapısı

Lazer tarayıcısı 46.55 ve 53.2 MHz'lik olmak üzere iki farklı ışık dalgası kullanmaktadır. Yansıyan ışığın şiddeti çok yüksek olursa tarayıcının içindeki algılayıcı kısmı doyma noktasına ulaşabilir ve gelen ışık şiddeti hissedilmeyebilir. Doyma durumunu önlemek için ışık algılayan Avalanş Fotodiyod'un kazancı (APD) AGC devresi (Auto-Gain Control) yardımıyla düşürülür. Böylece daha geniş algılama sağlanmış olur. Öte yandan, yansıyan çok düşük şiddetli ışıkların görülebilmesi için AGC devresi APD'nin kazancını düşürür. Başka deyişle, lazer tarayıcısının AGC kazancı yansıyan ışığın şiddetinin yüksek olup olmadığını söyler.

Yansıyan ışık şiddetini okumak için mesafe ölçme komutunda bulunan demet sayısı değiştirilir. Normal şartlarda, mesafeyi okumak için demet sayısı 0 ile 99 arasındadır. Işık şiddetini okumak için daha büyük sayılar kullanılır. Işık şiddetini okumak için demet sayısı Onaltılık tabanında "EF" olarak yollanır. Işık şiddet bilgisi 0 ile 40000 arasında değişmektedir. Bu bilgiyi kodlamak için 16 bitlik yere ihtiyacımız olup mesafe kodlama yöntemiyle kodlanmaz. Bu sorunu çözmek için alınan ışık şiddeti bilgisi 4 bit kaydırılarak 12 bit bilgiye çevirildikten sonra eski kodlama sistemi kullanılabilir.

EK 2

B Programlar

B1. MATLAB programları

B1.1. Ana program

Bu program çevrimdışı yöntemi ile altı tane robot kullanarak önceden bilinen haritada robot seyrüseferi sağlar. Sürünün dizilimi robotların arasında mesafeleri ayarlayarak belirtilmektedir. Benzetimdeki kaynak ve hedef noktaların konumları ve şiddetleri bu kısımda belirtilmektedir.

```
% Robot swarm navigation in unknown map using Panel Method
% Abdel-Razzak MERHEB
% TOBB-ETU  17-05-2009
clear all
close all
global M N flowvect sink gammas source gammasr a b c
n = 2;     % x-y coordinates
% the initial conditions of the leader
z0 = [];
z0 = [z0;10*rand(n,1);2*pi*rand(1,1)]; % x-y coordinates-initial angle
% the initial conditions of the other elements in the swarm
z1 = [];
for i = 1:5
    z1 = [z1;10*rand(n,1);2*pi*rand(1,1)];
end
% the vector containing all the agents
za = [z0;z1];
% Inter-agent distances for formation control
% Equilateral triangle - three possible inter-agent distances
d1 = 1;
d2 = 2*d1;
d3 = d1*sqrt(3);
% Distance matrix
d = [0  d1 d1 d2 d3 d2;
     d1  0 d1 d1 d1 d3;
     d1 d1  0 d3 d1 d1;
     d2 d1 d3  0 d1 d2;
     d3 d1 d1 d1  0 d1;
     d2 d3 d1 d2 d1  0];
% Potential function parameters
% The "b" and "c" parameters are fixed the "a" parameter depends on the
% desired inter-agent distance
b = 10; c = 1;
a = zeros(6,6);
for i = 1:6
    for j = 1:6
        if (j ~= i)
            a(i,j) = b*exp(-(d(i,j)*d(i,j))/c);
```

```
39          end
40      end
41  end
42  % the goal is represented by a sink
43  sink = [50;50];
44  gammas = 5; % the magnitude of the attractive function at the goal
45  % the start point is represented by a source
46  source = [2;2];
47  gammasr = 5; % the magnitude of the repulsive function at the source
48  % using the panel function find the magnitude of the vortex at each panel
49  map_detection_sw
50  t = 0:0.1:10; %simulation time (t=3.5 to show formation)
51  figure(2)
52  % Solving the differential equation
53  %options = odeset('OutputFcn','odeplot');
54  %[t, z] = ode23('robotipswfn', t, za, options);
55  [t, z] = ode23('robotipswfn', t, za);
56  % PLOT
57  % ============================================================================
58  Ni = 6;
59  [rsize csize]=size(z);
60  error=zeros(rsize,Ni);
61  Total_error=zeros(rsize,1);
62  J = zeros(rsize,Ni);
63  %% Draw the paths of the robots on the flow lines graph
64  [rsize csize] = size(z);
65  nn = 3;
66  figure(10)
67  hold on
68  for i = 1:6
69      plot(z(:,nn*(i-1)+1),z(:,nn*(i-1)+2),'-r');
70      plot(z(rsize,nn*(i-1)+1),z(rsize,nn*(i-1)+2),'or');
71  end
72  hold off
73  % ============================================================================
74  % Draw the formation taken by the robots
75  for i = 1:Ni
76      zfinal(i,:) = z(rsize,nn*(i-1)+1:nn*(i-1)+2);
77  end
78  figure(2);
79  plot(zfinal(:,1), zfinal(:,2), 'ro');
80  hold on
81
82  for i = 1:Ni
83      for j = i:Ni
84        if (i ~= j)
85                    x(:,1) = z(:,nn*(i-1)+1)-z(:,nn*(j-1)+1);
86                    y(:,1) = z(:,nn*(i-1)+2)-z(:,nn*(j-1)+2);
87                    dist2(:,1) = sqrt(x(:,1).^2+y(:,1).^2);
88                    error(:,i) = error(:,i)+dist(:,1)-d(i,j);
89        end
90  plot([zfinal(i,1), zfinal(j,1)], [zfinal(i,2), zfinal(j,2)], 'r--');
91      end
92  end
93  title('Final position of the members');
```

```
94  hold off
95  grid
96  for i = 1:Ni
97      for j = 1:Ni
98          if (j ~= i)
99                      x(:,1) = z(:,nn*(i-1)+1)-z(:,nn*(j-1)+1);
100                     y(:,1) = z(:,nn*(i-1)+2)-z(:,nn*(j-1)+2);
101                     dist2(:,i) = sqrt(x(:,1).^2+y(:,1).^2);
102                     J(:,i) = J(:,i)+(dist2(:,i)-d(i,j)).^2;
103         end
104     end
105 end
106 %% Calculate and draw the total error
107 for i = 1:Ni
108 Total_error(:,1)=Total_error(:,1)+error(:,i);
109 end
110 figure(3);
111 plot(t,Total_error(:,1));
112 title('Total error betw the desired and the current inter-agent dist');
113 grid
114 figure(4);
115 plot(t,J(:,1));
116 title('Graph of the Potential Function for 1 Robot');
117 grid
```

B1.2. Harita inceleme fonksiyon

Bu fonksiyon Çevrimdışı yöntemle kullanılmaktadır. Harita resim olarak okunur ve işlenir. Bu programın çıktısı ortamda bulunan engellerin panellere ayrılmasıdır ve bu panellerin bilgileridir (kontrol noktaları, normalleri, açıları vs.).

```
1  % Image processing program
2  % Abdel-Razzak MERHEB
3  % TOBB-ETU  13-05-2009
4  clc
5  clear all
6  close all
7  % Set the free flow data for the panel program
8  Q = 5; % intensity of the flow vector
9  AA = 60;    % angle of attack of the flow vector
10 % Read the map and transform the image from rgb to binary format
11 map = input('please enter the name of the map:','s');
12 image = imread(map);              % read the map image
13 figure(1),imshow(image)             % show the map
14 title('The given map')
15 im = rgb2hsv(image);% Convert red-green-blue colors to hue-saturation-value
16 im = im2bw(im);                   % transform the map to binary image
17 % arrange the map to clear the noise from the map
18 se = strel('rectangle',[3 3]);  % form a rectangle to dilate the image
19 im = imdilate(im,se);            % dilate the map so obstacles are dilated
20 figure,imshow(im)                % show the new map
21 title('Obstacles are dilated')
22 im = ~ im;                       % take the negative
23 se = strel('rectangle',[3 3]);  % form a rectangle to dilate the image
```

```
im = imdilate(im,se);            % dilate the negative map(noise is cleared)
im = imclose(im,se);             % morphological closing on the map
imneg = im;                      % save the negative of the map
figure,imshow(im)                % show the new map
title('noise is cleared')
% make each object a matrix with different value
L = ~ im;
L = bwlabel(L);
figure,imshow(L)
title('Map with labeled objects')
% edge detection program
imed = imdilate(im, ones(3,3)) & ~im;
im = imed;
figure,imshow(imed)
title('Edge detected obstacles')
imneg = rot90(imneg,-1);  % Rotate the negative image by -90 degrees
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Place Panels on the obstacles %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extract points from the edges (place the panel's points)
[rows,cols] = size(im);
for i = 5:1:(rows-1)
    for j = 5:1:(cols-1)
        if ( im(i,j) ~= 0)
            % if we use i+-4 and j+-4 the points will be a bit far from
            % each others, more panels will be used
            im(i-4,j-4) = 0;
            im(i-4,j-3) = 0;
            im(i-4,j-2) = 0;
            im(i-4,j-1) = 0;
            im(i-4,j) = 0;
            im(i-4,j+1) = 0;
            im(i-4,j+2) = 0;
            im(i-4,j+3) = 0;
            im(i-4,j+4) = 0;

            im(i-3,j-4) = 0;
            im(i-3,j-3) = 0;
            im(i-3,j-2) = 0;
            im(i-3,j-1) = 0;
            im(i-3,j) = 0;
            im(i-3,j+1) = 0;
            im(i-3,j+2) = 0;
            im(i-3,j+3) = 0;
            im(i-3,j+4) = 0;

            im(i-2,j-4) = 0;
            im(i-2,j-3) = 0;
            im(i-2,j-2) = 0;
            im(i-2,j-1) = 0;
            im(i-2,j) = 0;
            im(i-2,j+1) = 0;
            im(i-2,j+2) = 0;
            im(i-2,j+3) = 0;
            im(i-2,j+4) = 0;
```

```
79
80                  im(i-1,j-4) = 0;
81                  im(i-1,j-3) = 0;
82                  im(i-1,j-2) = 0;
83                  im(i-1,j-1) = 0;
84                  im(i-1,j) = 0;
85                  im(i-1,j+1) = 0;
86                  im(i-1,j+2) = 0;
87                  im(i-1,j+3) = 0;
88                  im(i-1,j+4) = 0;
89
90                  im(i,j-4) = 0;
91                  im(i,j-3) = 0;
92                  im(i,j-2) = 0;
93                  im(i,j-1) = 0;
94                  im(i,j+1) = 0;
95                  im(i,j+2) = 0;
96                  im(i,j+3) = 0;
97                  im(i,j+4) = 0;
98
99                  im(i+1,j-4) = 0;
100                 im(i+1,j-3) = 0;
101                 im(i+1,j-2) = 0;
102                 im(i+1,j-1) = 0;
103                 im(i+1,j) = 0;
104                 im(i+1,j+1) = 0;
105                 im(i+1,j+2) = 0;
106                 im(i+1,j+3) = 0;
107                 im(i+1,j+4) = 0;
108
109                 im(i+2,j-4) = 0;
110                 im(i+2,j-3) = 0;
111                 im(i+2,j-2) = 0;
112                 im(i+2,j-1) = 0;
113                 im(i+2,j) = 0;
114                 im(i+2,j+1) = 0;
115                 im(i+2,j+2) = 0;
116                 im(i+2,j+3) = 0;
117                 im(i+2,j+4) = 0;
118
119                 im(i+3,j-4) = 0;
120                 im(i+3,j-3) = 0;
121                 im(i+3,j-2) = 0;
122                 im(i+3,j-1) = 0;
123                 im(i+3,j) = 0;
124                 im(i+3,j+1) = 0;
125                 im(i+3,j+2) = 0;
126                 im(i+3,j+3) = 0;
127                 im(i+3,j+4) = 0;
128
129                 im(i+4,j-4) = 0;
130                 im(i+4,j-3) = 0;
131                 im(i+4,j-2) = 0;
132                 im(i+4,j-1) = 0;
133                 im(i+4,j) = 0;
```

```
134                 im(i+4,j+1) = 0;
135                 im(i+4,j+2) = 0;
136                 im(i+4,j+3) = 0;
137                 im(i+4,j+4) = 0;
138             end
139         end
140     end
141     figure,imshow(im)
142     title('Panels points')
143     % the origin of the image processing tool is different from the origin of
144     % MATLAB (plaot command,...), so we have to rotate the image by -90 degrees
145     % before starting putting the panels
146     im = rot90(im,-1);  % Rotate the image by -90 degrees
147     [rows,cols] = size(im); % the new size (cols->rows and rows->cols)
148     % Extract the coordinates of the panel points
149     % the coordinates of the points will be given in xp-yp vectors
150     % we'll have k points
151     k = 1;
152     for i = 1:rows
153         for j = 1:cols
154
155         if (im(i,j) ~= 0)
156             xp(k) = i;
157             yp(k) = j;
158             k = k+1;
159         end
160
161         end
162     end
163     % here find the distance from each point to all other points
164     % if this distance is less than a given threshold, the two
165     % points are neighbors, find the midpoint(the control point)
166     % and the orientation of the panel
167     l = 1;
168     for i = 1:(k-1)
169         for j = 1:(k-1)
170             dist = sqrt(((xp(i) - xp(j))^2)+((yp(i) - yp(j))^2));
171             if (dist < 10)          % the two points are neighbors
172             % ay and jey are the indices of the neighbor points
173             % [xp(ay),yp(ay)] is the neighbor point of [xp(jey),yp(jey)]
174             % but they're repeated indices and zero indices
175             % you have to get rid of them
176             ay(l) = i;
177             jey(l) = j;
178             l = l+1;
179             end
180         end
181     end
182     % Now get rid of the repeated indices
183     % if ay(i) = jey(i) => the same point not neighbors!get rid of these points
184      for i = 1:(l-1)
185             if (ay(i) == jey(i))
186             % [xp(ay),yp(ay)] is the neighbor point of [xp(jey),yp(jey)]
187             ay(i) = 0;
188             jey(i) = 0;
```

```
189           end
190   end
191   % if ay(i)=jey(j) and ay(j)=jey(i)=>repeated neighbor points!get rid...
192   for i = 1:(l-1)
193         for j = 1:(l-1)
194           if (ay(i) == jey(j) && jey(i) == ay(j))
195           % [xp(ay),yp(ay)] is the neighbor point of [xp(jey),yp(jey)]
196           ay(i) = 0;
197           jey(i) = 0;
198           end
199       end
200   end
201   % Now make new ay() and jey() vectors, without the zero indices
202   n = 1;
203   for i = 1:(l-1)
204       if (ay(i) ~= 0 && jey(i) ~= 0)
205            ay2(n) = ay(i);
206            jey2(n) = jey(i);
207            n = n+1;
208       end
209   end
210 % [xp(ay2),yp(ay2)] is the neighbor point of [xp(jey2),yp(jey2)]
211
212 %  Each node has only two neighbors, if it has more than 2
213 %  take only the closest two neighbors
214    f = 1;
215   for i = 1:(n-1)
216           xxc(i) = xp(ay2(i))+((xp(jey2(i))-xp(ay2(i)))/2);
217           yyc(i) = yp(ay2(i))+((yp(jey2(i))-yp(ay2(i)))/2);
218 a(i) = imneg(round(xxc(i)+3),round(yyc(i)));
219 b(i) = imneg(round(xxc(i)),round(yyc(i)-3));
220 c(i) = imneg(round(xxc(i)-3),round(yyc(i)));
221 d(i) = imneg(round(xxc(i)),round(yyc(i)+3));
222
223       if (a(i) == 0 && b(i) == 0 && c(i) == 0 && d(i) == 0)
224
225           ay2(i) = 0;
226           jey2(i) = 0;
227           f = f+1;
228       end
229   end
230    % Now make new ay() and jey() vectors, without the zero indices
231   [mm,nn] = size(ay2);
232   k = 1;
233   for i = 1:(nn)
234       if (ay2(i) ~= 0 && jey2(i) ~= 0)
235            ay3(k) = ay2(i);
236            jey3(k) = jey2(i);
237            k = k+1;
238       end
239   end
240 % Now find the midpoints (the control points) and the orientations
241 % the control points [mm,nn] = size(ay3);
242 nn = k;
243 for m = 1:(nn-1)
```

```
        xc(m) = xp(ay3(m))+((xp(jey3(m))-xp(ay3(m)))/2);
        yc(m) = yp(ay3(m))+((yp(jey3(m))-yp(ay3(m)))/2);
end
% the orientation and the normal of each panel
normalp = zeros(2,m);
dx = 5;
dy = 5;
for m = 1:(nn-1)
    % find the angle of the panel using its two end-points
    op(m) = atan((yp(ay3(m))-yp(jey3(m)))/(xp(ay3(m))-xp(jey3(m))));
    % the normal vector of the panel
    normalp(:,m) = [-sin(op(m)); cos(op(m))];
    % here the ends of the normal vector (raso lal vecteur)
    xc1(m) = xc(m)+dx*normalp(1,m);
    yc1(m) = yc(m)+dy*normalp(2,m);
    % if the end-points of the normal vectors are inside the object, add pi
    if (imneg(floor(xc1(m)),floor(yc1(m))) == 0)
        op(m) = atan((yp(ay3(m))-yp(jey3(m)))/(xp(ay3(m))-xp(jey3(m))))+pi;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sqt = op(1,:);
xc = xc(1,:);
yc = yc(1,:);
N = m;
% make everything smaller
xp = 0.3*xp;
yp = 0.3*yp;
xc = 0.3*xc;
yc = 0.3*yc;
 figure(10)
 hold on
 % plot the panels, their starting and ending points, and their control
 % points
 plot(xp,yp, 'bo');
 plot(xc,yc, 'rx');
for m = 1:(nn-1)
    plot([xp(ay3(m)) xp(jey3(m))],[yp(ay3(m)) yp(jey3(m))]);
end
% Refind the normal vector using the improved orientations
for m = 1:(nn-1)

    normalp(:,m) = [-sin(op(m)); cos(op(m))];
end
% draw the normal
dx = 0.05*50;
dy = 0.05*50;
for i = 1:1:N
   plot([xc(i) xc(i)+dx*normalp(1,i)],[yc(i) yc(i)+dy*normalp(2,i)], 'r');
end
hold off
% now save data into a .dat file
% transform data into one matrix: [xc yc sqt] and save it
[M,N] = size(xc);
xxc = Q*ones(N+1,1);
```

```
299 yyc = AA*ones(N+1,1);
300 ssqt = N*ones(N+1,1);
301 for i = 2:1:N+1
302     xxc(i) = xc(i-1);
303     yyc(i) = yc(i-1);
304     ssqt(i) = sqt(i-1);
305 end
306 data = [xxc yyc ssqt];
307 save data.dat data -ascii
308 % call the panel program
309 panel_IP
```

B1.3. Panel metodu fonksiyonu

Bu fonksiyon, harita inceleme programından elde edilen bilgileri kullanarak panel metodu uygulamaktadır. Programın çıktısı panelleri göstermekte kullanılan girdap noktalarının şiddetleri ve akış çizgileridir.

```
 1 % Applying Panel method
 2 % Abdel-Razzak MERHEB
 3 % TOBB-ETU  13-04-2009
 4 % data in "square.dat" is as follows:
 5 % Q AA N
 6 % x1 y1 teta1 : 1st panel
 7 % x2 y2 teta2 : 2nd panel
 8 % x3 y3 teta3 : 3rd panel
 9 load data.dat;      % read data into the square matrix
10 % loading the free flow data
11 Q = data(1,1);      % load the flow magnitude
12 AA = deg2rad(data(1,2));      % load the angle of attack
13 N = data(1,3);      % load the total number of panels
14 % loading the panels' data: coordinates of starting and ending points of
15 % each panel along with its angle with the horizontal
16 for i = 2:1:(N+1)
17     xc(i-1) = data(i,1);      % load the x-coordinates of the points
18     yc(i-1) = data(i,2);      % load the y-coordinates of the points
19     sqt(i-1) = data(i,3);      % load the panel inclination
20 end
21 % here is the infinity flow vector
22 flowvect = [Q*cos(AA); Q*sin(AA)];
23 % forming the normal vector of each hpanel
24 normalvector = zeros(2,N);
25 for i = 1:1:N
26      normalvector(:,i) = [-sin(sqt(i)); cos(sqt(i))];
27 end
28 % forming the RHS vector
29 for i = 1:1:N
30     RHS = -(normalvector')*flowvect;
31 end
32 % forming the Influence coefficient matrix
33 % the vortex is located at panel i
34 av = -0.5*eye(N,N);
35 uv = zeros(N,N);
36 wv = zeros(N,N);
```

```
for i = 1:1:N
    for j = 1:1:N
        if (j~=i)
            uv(j,i) = ((yc(j)-yc(i))/((xc(j)-xc(i))^2+(yc(j)-yc(i))^2)) -...
                ((xc(j)-sink(1,1))/((xc(j)-sink(1,1))^2+...
                (yc(j)-sink(2,1))^2)) + ((xc(j)-source(1,1))/...
                ((xc(j)-source(1,1))^2+(yc(j)-source(2,1))^2));
            wv(j,i) = -((xc(j)-xc(i))/((xc(j)-xc(i))^2+(yc(j)-yc(i))^2)) -...
                ((yc(j)-sink(2,1))/((xc(j)-sink(1,1))^2+...
                (yc(j)-sink(2,1))^2)) + ((yc(j)-source(2,1))/...
                ((xc(j)-source(1,1))^2+(yc(j)-source(2,1))^2));
            av(j,i) = 0.4.*[uv(j,i) wv(j,i)]*normalvector(:,j);
            % 0.4 is the unit length of each panel
        end
    end
end
% solve the equation "a*gamma = RHS" for gamma
gammav = inv(av)*RHS; % 3*
% the number of points in the mesh
nx = 50;
ny =50;
figure(1)
% finally plot the panels used and their control points
hold on
axis([1, nx, 1, ny])
% title('The panels and their control points');
grid
% plot the panels, their starting - ending and control points
plot(xc,yc, 'rx');
dx = 0.05*nx;
dy = 0.05*ny;
for i = 1:1:N
   plot([xc(i) xc(i)+dx*normalvector(1,i)],...
       [yc(i) yc(i)+dy*normalvector(2,i)], 'r');
end
% now define a mesh and plot the velocity vectors at each point
sx = 0.05;
sy = 0.05;
xp = 1:sx:nx;
yp = 1:sy:ny;
for kk = 1:length(xp),
    for jj = 1:length(yp),
        % the velocity components of the free flow at a given point
        u = flowvect(1,1) - (gammas/2*pi)*((xp(kk)-sink(1,1))/...
                ((xp(kk)-sink(1,1))^2+(yp(jj)-sink(2,1))^2)) + ...
                (gammasr/2*pi)*((xp(kk)-source(1,1))/...
                ((xp(kk)-source(1,1))^2+(yp(jj)-source(2,1))^2));
        w = flowvect(2,1) - (gammas/2*pi)*((yp(jj)-sink(2,1))/...
                ((xp(kk)-sink(1,1))^2+(yp(jj)-sink(2,1))^2)) + ...
                (gammasr/2*pi)*((yp(jj)-source(2,1))/...
                ((xp(kk)-source(1,1))^2+(yp(jj)-source(2,1))^2));
        for i = 1:1:N
            u = u + (gammav(i)/2*pi)*((yp(jj)-yc(i))/((xp(kk)-xc(i))^2+...
                (yp(jj)-yc(i))^2));
            w = w - (gammav(i)/2*pi)*((xp(kk)-xc(i))/((xp(kk)-xc(i))^2+...
```

```
                    (yp(jj)-yc(i))^2));
            end
% the velocity components at the given point
    u1(kk,jj) = u/norm([u w]);
    w1(kk,jj) = w/norm([u w]);

        end
end
% draw the flow vectors/the stream lines
streamslice(xp, yp, u1', w1')   % flow lines are right and at the object :)
hold off
```

B1.4. Robot kontrol fonksiyonu

Bu fonksiyon robot dinamiklerini modelleyerek robotların akış çizgilerine ve potansiyel fonksiyonlara göre hesaplanan hızlarını ve konum değişikliklerini hesaplamaktadır.

```
% Abdel-Razzak MERHEB
% TOBB-ETU   17-05-2009
function zdot = robotipswfn(t,x)
global gammav flowvect xc yc u1 w1 sink gammas N source gammasr a b c
M = 0;
P = 0;
z = zeros(3, 6);
% Seperate the states of the 6 agents
for i = 1:6
    z(:, i) = x(3*(i-1)+1:3*i);
end
% here is the derivatibe gain
K = 100;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%Here find the desired velocity and angle using Panel method%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the velocity components of the free flow at a given point
for j = 1:6
        u(j) = flowvect(1,1) - (gammas/2*pi)*...
                ((z(1,j)-sink(1,1))/((z(1,1)-sink(1,1))^2+...
                (z(2,j)-sink(2,1))^2)) + (gammasr/2*pi)*((z(1,j)-source(1,1))/...
                ((z(1,j)-source(1,1))^2+(z(2,j)-source(2,1))^2));
        w(j) = flowvect(2,1) - (gammas/2*pi)*...
                ((z(2,j)-sink(2,1))/((z(1,1)-sink(1,1))^2...
            +(z(2,j)-sink(2,1))^2)) + (gammasr/2*pi)*((z(2,j)-source(2,1))/...
                ((z(1,j)-source(1,1))^2+(z(2,j)-source(2,1))^2));
end

for i = 1:1:M+N+P
        for j = 1:6
            u(j) = u(j) + (gammav(i)/2*pi)*(((z(2,j)-yc(i))/...
                ((z(1,j)-xc(i))^2+(z(2,j)-yc(i))^2)));
            w(j) = w(j) + (-gammav(i)/2*pi)*(((z(1,j)-xc(i))/...
                ((z(1,j)-xc(i))^2+(z(2,j)-yc(i))^2)));
        end
end
```

```
37
38 for j = 1:6
39     velocity(j) = norm([u(j) w(j)]);
40     % velocity bounding
41     vmax = 10;
42     if (velocity(j) > vmax)
43        velocity(j) = vmax;
44     end
45 end
46 % the real orientation of the robots
47 for j = 1:6
48     theta(j) = mod(z(3,j), 2*pi); % take the smallest angle of the robot
49 end
50 % Calculate the angle
51 for j = 1:6
52     desiredtheta(j) = mod(atan2(w(j),u(j)), 2*pi);
53 end
54 % Find the derivative of the velocity components: u and w, udot and wdot
55 for j = 1:6
56     udot(j) = 0;
57     wdot(j) = 0;
58 end
59
60 for i = 1:1:M+N+P
61     for j = 1:6
62 udot(j) = udot(j) + (gammav(i)/2*pi)*(velocity(j)*sin(theta(j))*...
63     ((((z(1,j)-xc(i))^2+(z(2,j)-yc(i))^2)))-2*(z(2,j)-yc(i))*...
64     ((z(1,j)-xc(i))*velocity(j)*cos(theta(j))+(z(2,j)-yc(i))*velocity(j)*...
65     sin(theta(j))))/(((z(1,j)-xc(i))^2+(z(2,j)-yc(i))^2))^2 + ...
66      (gammas/2*pi)*(velocity(j)*cos(theta(j))*((((z(1,j)-sink(1,1))^2+...
67      (z(2,j)-sink(2,1))^2)))-2*(z(1,j)-sink(1,1))*((z(1,j)-sink(1,1))*...
68      velocity(j)*cos(theta(j))+(z(2,j)-sink(2,1))*velocity(j)*...
69      sin(theta(j))))/(((z(1,j)-sink(1,1))^2+(z(2,j)-sink(2,1))^2))^2 + ...
70      (gammasr/2*pi)*(velocity(j)*cos(theta(j))*((((z(1,j)-source(1,1))^2+...
71      (z(2,j)-source(2,1))^2)))-2*(z(1,j)-source(1,1))*...
72      ((z(1,j)-source(1,1))*velocity(j)*cos(theta(j))+...
73      (z(2,j)-source(2,1))*velocity(j)*sin(theta(j))))/...
74      (((z(1,j)-source(1,1))^2+(z(2,j)-source(2,1))^2))^2;
75 wdot(j) = wdot(j) + (gammav(i)/2*pi)*(velocity(j)*cos(theta(j))*...
76     ((((z(1,j)-xc(i))^2+(z(2,j)-yc(i))^2)))-2*(z(1,j)-xc(i))*...
77     ((z(1,j)-xc(i))*velocity(j)*cos(theta(j))+(z(2,j)-yc(i))*velocity(j)*...
78     sin(theta(j))))/(((z(1,j)-xc(i))^2+(z(2,j)-yc(i))^2))^2 + ...
79     (gammas/2*pi)*(velocity(j)*sin(theta(j))*((((z(1,j)-sink(1,1))^2+...
80     (z(2,j)-sink(2,1))^2)))-2*(z(2,j)-sink(2,1))*((z(1,j)-sink(1,1))*...
81     velocity(j)*cos(theta(j))+(z(2,j)-sink(2,1))*velocity(j)*...
82     sin(theta(j))))/(((z(1,j)-sink(1,1))^2+(z(2,j)-sink(2,1))^2))^2 + ...
83     (gammasr/2*pi)*(velocity(j)*sin(theta(j))*((((z(1,j)-source(1,1))^2+...
84     (z(2,j)-source(2,1))^2)))-2*(z(2,j)-source(2,1))*...
85     ((z(1,j)-source(1,1))*velocity(j)*cos(theta(j))+...
86     (z(2,j)-source(2,1))*velocity(j)*sin(theta(j))))/...
87     (((z(1,j)-source(1,1))^2+(z(2,j)-source(2,1))^2))^2;
88     end
89 end
90 % The derivative of arctan is 1/(1+x^2)=>
91 % desiredthetadot = 1/(1+(w/u)^2)*(w/u)' = (wdot*u - udot*w)/(u^2 + w^2);
```

```
for j = 1:6
    desiredthetadot(j) = (wdot(j)*u(j) - udot(j)*w(j))/(u(j)^2+w(j)^2);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Here find the force between the elements of the swarm (formation control)%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Find the gradient of the potential function at the positions of the agents
for i = 1:6
  grad_g(:, i) = zeros(2,1);
  for j = 1:6
    if (j ~= i),
      grad_g(:, i) = grad_g(:, i) + gradfor(z(1:2, i) - z(1:2, j),a(i,j));
    end
  end
end
% Define the controllers for the elemets' motion (other than the leader)
for i = 1:6
    v(i) = norm(grad_g(:, i)); % The velocity controller
    % P Controller for the orientation dynamics (The desired orientation)
    thetadesired(i) = mod(atan2(grad_g(2, i), grad_g(1, i)), 2*pi);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%% add a repulsive potential function to the obstacles %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% first find the distance from each robot to each panel
for i = 1:6             % for all the robots
    for j = 1:M+N+P     % for all the panels
        dist(i,j) = sqrt((xc(j)-z(1,i))^2+(yc(j)-z(2,i))^2);
    end
end
% now find the minimum distances of each robot to the closest panel
for i = 1:6             % for each robot
        mindist(i) = min(dist(i,:));
end
% now find which panel is closest to which robot
for i = 1:6             % for each robot
    for j = 1:M+N+P     % for all the panels
        if (dist(i,j) == mindist(i))
            jey(i) = j;
        end
    end
end
% form a repulsive potential function for close obstacles to each robot
for i = 1:6             % for each robot
    rpf(i) = 0;
    if (mindist(i)<0.5)
        rpf(i) = (1/(mindist(i))^2) -((1/(mindist(i))^3));
    end
end
% now find the x and y gradient of the potential function (10:gain)
for i = 1:6             % for each robot
        gradx(i) = 10*(xc(jey(i))-z(1,i))*rpf(i);
        grady(i) = 10*(yc(jey(i))-z(2,i))*rpf(i);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% evaluate the derivatives for the leader robot
for i = 1:6
    xdot(1,i) = velocity(i)*cos(theta(i)) + v(i)*cos(thetadesired(i)) + ...
        gradx(i);   % xdot = v.cos(theta)
    xdot(2,i) = velocity(i)*sin(theta(i)) + v(i)*sin(thetadesired(i)) + ...
        grady(i);   % ydot = v.sin(theta)
    xdot(3,i) = - K*(mod(theta(i) - desiredtheta(i) + pi, 2*pi) - pi) + ...
        desiredthetadot(i);   % a kind of P controller with feedforward term "de
end
% put the data again into z vector(Return a column vector)
for i = 1:6
  zdot(3*(i-1)+1:3*i) = xdot(:, i);
end;
zdot = zdot';
%%% gradient of the potential Function g(.) to be used above
function gout = gradfor(y, aa)
global b c;
% Linear attraction and exponential repulsion
gout = - y.*(aa - b*exp(-(norm(y)^2)/c));
```

B2. C programları

B2.1. Lazer okuma fonksiyonu (yoğunluk ve mesafe)

```
/*
    Abdel-Razzak MERHEB, JUNE 2010
    improvement of yunus' program
    this program can read the intensity
    SWARM SYSTEMS LABORATORY
    TOBB UNIVERSITY OF
    ECONOMICS AND TECHNOLOGY
    ANKARA, TURKEY
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <termios.h>
#include <string.h>
#include <errno.h>
#include <math.h>
#include "khepera3.h"
#include "khepera3_urglaser.h"

#define URGSIZE 4096
#define INTSIZE 40000 // intensity size
#define maxtry_com 1250
#define PI 3.14159265358979

struct termios oldtermios;
static int fd;
```

```
31 char sCmd[20]="\0";
32 char status[20]="\0";
33 int datalen=0;
34
35 // Open the laser and set the communication
36 int khepera3_openurgport(char *portname)
37 {
38     char cmd[URGSIZE]="\0";
39     char USBport[20]="/dev/ttyACM0";
40     // Open the URG laser port. If not opened return an error with -1
41     if(OpenUrgPort(portname,115200)<1)
42     {printf("UrgLaser port opening error!\n"); return -1;}
43     // Open the URG laser
44     sprintf(cmd,"L1\n");
45     WriteUrgPort(cmd);
46     usleep(300000);
47     return 0;
48 }
49
50 // Initiate the laser for distance reading
51 int  khepera3_urglaser_no11(char *portname,int start_step,
52     int stop_step,int cluster)
53 {
54     int mod=64;
55     int steplen,modsteplen,modlen;
56     sprintf(sCmd,"G%03d%03d%02d\n",start_step,stop_step,cluster);
57     steplen=ceil((stop_step-start_step+1)/(float)cluster)*2;
58     modsteplen=steplen%mod;
59     modlen=(steplen-modsteplen)/mod;
60     datalen=strlen(status)+modlen*(mod+1)+modsteplen+2;
61     if(modsteplen==0) datalen--;
62     if (WriteUrgPort(sCmd) < 0){printf("WriteUrgPort Error!"); return -1;}
63     return 0;
64 }
65
66 // Initiate the laser for intensity reading
67 int  khepera3_urglaserintensity_no11(char *portname,int start_step,
68     int stop_step,int clusterhex)
69 {
70     char cmd[URGSIZE]="\0";
71     int mod=64;
72     int steplen,modsteplen,modlen;
73     sprintf(sCmd,"G%03d%03d%x\n",start_step,stop_step,clusterhex);
74     steplen=ceil((stop_step-start_step+1)/(float)1)*2;
75     modsteplen=steplen%mod;
76     modlen=(steplen-modsteplen)/mod;
77     datalen=strlen(status)+modlen*(mod+1)+modsteplen+2;
78     if(modsteplen==0) datalen--;
79     if (WriteUrgPort(sCmd) < 0){printf("WriteUrgPort Error!");
80     return -1;}
81     return 0;
82 }
83
84 // Initiate the laser for distance reading
85 int  khepera3_urglaser_init(char *portname,int start_step,
```

```
86      int stop_step,int cluster)
87  {
88      char cmd[URGSIZE]="\0";
89      char buffer[10]="\0";
90      char USBport[20]="/dev/ttyACM0";
91      int mod=64;
92      int steplen,modsteplen,modlen;
93
94      if(strcmp(USBport,portname)!=0)
95      {
96          if(OpenUrgPort(portname,19200)<1) {printf("UrgLaser port error!\n");
97          return -1;}
98          sprintf(cmd,"S115200*******\n");
99          WriteUrgPort(cmd);
100     }
101     if(OpenUrgPort(portname,115200)<1) {printf("UrgLaser port opening error!\n");
102     return -1;}
103     sprintf(cmd,"L1\n");
104     WriteUrgPort(cmd);
105     usleep(300000);
106     ReadUrgPort(buffer,10);
107
108     sprintf(sCmd,"G%03d%03d%02d\n",start_step,stop_step,cluster);
109     sprintf(status,"%s0\n",sCmd);
110     steplen=ceil((stop_step-start_step+1)/(float)cluster)*2;
111     modsteplen=steplen%mod;
112     modlen=(steplen-modsteplen)/mod;
113     datalen=strlen(status)+modlen*(mod+1)+modsteplen+2;
114     if(modsteplen==0) datalen--;
115
116     if (WriteUrgPort(sCmd) < 0){printf("WriteUrgPort Error!"); return -1;}
117
118     return 0;
119 }
120
121 // I added this function to read the intensity of the laser
122 // Read the intensity by reading the AGC voltage
123 int khepera3_urgintensity_read(float* distance)
124 {
125     char sResult[URGSIZE]="\0";
126     char buffer[URGSIZE]="\0";
127     char *dataPtr;
128     int try_com=0,bufferlen=0,i,arraysize=0;
129     unsigned short int buf0=0,buf1=0,buf=0;
130     while((dataPtr==NULL)&&(try_com <maxtry_com))
131     {
132         bufferlen=ReadUrgPort(buffer,datalen);
133
134         if(bufferlen<0) return -1;
135
136         dataPtr=strstr(buffer,status);
137
138         if (dataPtr!=NULL){
139             strcpy(sResult,dataPtr);
140             break;
```

```
141             }
142
143             if(try_com>1){
144                 usleep(100);
145             }
146             try_com++;
147         }
148         if(try_com>=maxtry_com)
149         {
150             printf("Try=%d\nCommunication error!\n",try_com);
151             return -1;
152         }
153
154         try_com=0;
155         while((strlen(sResult)<datalen)&&(try_com <maxtry_com))
156         {
157             bufferlen=ReadUrgPort(buffer,datalen-strlen(sResult));
158             if(bufferlen<0) return -1;
159
160             if(bufferlen>0)
161             {
162                 strncat(sResult,buffer,bufferlen);
163             }
164
165             if(try_com>1){
166                 usleep(100);
167             }
168             try_com++;
169         }
170         if(try_com>=maxtry_com)
171         {
172             printf("Try=%d\nCommunication error!\n",try_com);
173             return -1;
174         }
175
176         if (WriteUrgPort(sCmd) < 0){printf("WriteUrgPort Error!"); return -1;}
177
178         for(i=strlen(status);i<(strlen(sResult)-3);i+=2)
179         {
180
181             if(sResult[i] == '\n') i++;
182
183             buf1=sResult[i]-0x30;
184             buf0=sResult[i+1]-0x30;
185             buf1<<=6;
186             buf=buf1|buf0;
187             distance[arraysize]=buf;
188             arraysize++;
189         }
190         return arraysize;
191     }
192
193     // Read distance
194     int khepera3_urglaser_read(float* distance)
195     {
```

```
196     char sResult[URGSIZE]="\0";
197     char buffer[URGSIZE]="\0";
198     char *dataPtr;
199     //int num0[6]={0},num1[6]={0},buf0={0},buf1={0};
200     int try_com=0,bufferlen=0,i,arraysize=0;
201     unsigned short int buf0=0,buf1=0,buf=0;
202     while((dataPtr==NULL)&&(try_com <maxtry_com))
203     {
204         bufferlen=ReadUrgPort(buffer,datalen);
205
206         if(bufferlen<0) return -1;
207
208         dataPtr=strstr(buffer,status);
209
210         if (dataPtr!=NULL){
211             strcpy(sResult,dataPtr);
212             break;
213         }
214
215         if(try_com>1){
216             usleep(100);
217         }
218         try_com++;
219     }
220     if(try_com>=maxtry_com)
221     {
222         printf("Try=%d\nCommunication␣error!\n",try_com);
223         return -1;
224     }
225
226     try_com=0;
227     while((strlen(sResult)<datalen)&&(try_com <maxtry_com))
228     {
229         bufferlen=ReadUrgPort(buffer,datalen-strlen(sResult));
230         if(bufferlen<0) return -1;
231
232         if(bufferlen>0)
233         {
234             strncat(sResult,buffer,bufferlen);
235         }
236
237         if(try_com>1){
238             usleep(100);
239         }
240         try_com++;
241     }
242     if(try_com>=maxtry_com)
243     {
244         printf("Try=%d\nCommunication␣error!\n",try_com);
245         return -1;
246     }
247
248     if (WriteUrgPort(sCmd) < 0){printf("WriteUrgPort␣Error!"); return -1;}
249
250     for(i=strlen(status);i<(strlen(sResult)-3);i+=2)
```

```
251     {
252         if(sResult[i] == '\n') i++;
253 
254         buf1=sResult[i]-0x30;
255             buf0=sResult[i+1]-0x30;
256         buf1<<=6;
257         buf=buf1|buf0;
258         distance[arraysize]=buf*0.001;
259         arraysize++;
260     }
261     return arraysize;
262 }
263 
264 int binarytodecimal(int* num1,int* num0,int size)
265 {
266     int decimal=0,i;
267     for(i=0;i<size;i++)
268     {
269         decimal+=num1[i]*power(2,size+i)+num0[i]*power(2,i);
270     }
271     return decimal;
272 }
273 
274 int OpenUrgPort(char* sPortName,long unsigned int baudrate)
275 {
276     if(fd>0) close(fd);
277 
278     fd = open(sPortName, O_RDWR | O_NOCTTY);
279     tcgetattr(fd,&oldtermios); /* save current port settings */
280 
281     if (fd < 0)
282     {
283         printf("open error %d %s\n", errno, strerror(errno));
284     }
285     else
286     {
287         struct termios newtermios;
288         tcgetattr(fd, &newtermios);
289     newtermios.c_cflag &= ~PARENB;
290     newtermios.c_cflag &= ~CSTOPB;
291     newtermios.c_cflag &= ~CSIZE;
292     newtermios.c_cflag |= CS8;
293     newtermios.c_cflag |= (CLOCAL | CREAD);
294     newtermios.c_iflag  &= ~(IXON | IXOFF | IXANY);
295     newtermios.c_iflag  &= IGNPAR;
296     newtermios.c_oflag &= ~OPOST;
297     newtermios.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);
298     newtermios.c_cc[VMIN]=0;
299     newtermios.c_cc[VTIME]=1;
300     if(baudrate==115200)
301     {
302         cfsetispeed(&newtermios, B115200);
303         cfsetospeed(&newtermios, B115200);
304     }
305     else
```

```
306     {
307         cfsetispeed(&newtermios, B19200);
308         cfsetospeed(&newtermios, B19200);
309     }
310         tcsetattr(fd, TCSANOW, &newtermios);
311
312     usleep(70000);
313     tcflush(fd, TCIFLUSH);
314     tcflush(fd, TCOFLUSH);
315     }
316
317     return fd;
318 } // end OpenUrgPort
319
320 // writes zero terminated string to the Urg port
321 // return code:
322 //   >= 0 = number of characters written
323 //   -1 = write failed
324 int WriteUrgPort(char* psOutput)
325 {
326     int iOut;
327     if (fd < 1)
328     {
329         printf("␣port␣is␣not␣open\n");
330         return -1;
331     } // end if
332     iOut = write(fd, psOutput, strlen(psOutput));
333     if (iOut < 0)
334     {
335         printf("write␣error␣%d␣%s\n", errno, strerror(errno));
336         return -1;
337     }
338     return iOut;
339 } // end WriteSerialPort
340
341 // read string from the serial port
342 // return code:
343 //   >= 0 = number of characters read
344 //   -1 = read failed
345 int ReadUrgPort(char* psResponse, int iMax)
346 {
347     int iIn;
348
349     if (fd < 1)
350     {
351         printf("␣port␣is␣not␣open\n");
352         return -1;
353     } // end if
354     iIn = read(fd, psResponse, iMax);
355     if (iIn < 0)
356         {
357             if (errno == EAGAIN)
358             {
359             return 0; // assume that command generated no response
360         }
```

```
361             else
362             {
363                 printf("read error %d %s\n", errno, strerror(errno));
364             } // end if
365             }
366         else
367         {
368              psResponse[iIn<iMax?iIn:iMax] = '\0';
369         } // end if
370
371             return iIn;
372     } // end ReadSerialPort
373
374     void CloseUrgPort()
375     {
376         char cmd[10]="\0";
377         if (fd > 0)
378             {
379             sprintf(cmd,"L0\n");
380             WriteUrgPort(cmd);
381             tcsetattr(fd, TCSANOW, &oldtermios);
382             close(fd);
383         }// end if
384     } // end CloseSerialPort
385
386     int UrglaserDataReceived()
387     {
388         int bytes;
389         ioctl(fd, FIONREAD, &bytes);
390         if(bytes>=datalen) return 1;
391         else return 0;
392     }
```

B2.2. Lazer barkod fonksiyonu

Bu fonksiyon vasıtasıyla robot sürü arkadaşlarını barkod kimliklerinden tanır. Robot ilk önce yoğunluk taraması yapar ve barkodları ortamda bulunan başka nesnelerden ayırır. İkinci adımda robot barkodları okur ve hangi barkodun hangi robota ait olduğunu tespit eder. Robot bir mesafe taraması yapar ve bulunan robotların konumları çıkarır. Robotların görünen konumlarının önceki adımdaki konumlarından çok uzak olduğunu tespit edilirse okumada hata oluştuğu varsayarak iletişimden gelen konum bilgileri kullanılır. Aynı şekilde sürüden bir robot algılanmazsa konumu iletişimden alınır.

```
1  // Find the positions of other robots using LASER scanner
2  void las_func(){
3
4  // SCAN for the distance data
5  // initialize lazer for distance reading
6  khepera3_urglaser_no11(URGPORTNAME,44,725,1);
7  Laser_distance_Function(); // scan obstacles
8
9  // SCAN for the intensity data
```

```
10 // initialize lazer for intensity reading
11 khepera3_urglaserintensity_no11(URGPORTNAME,44,725,0xEF);
12 Laser_intensity_Function(); // scan intensities
13
14 // HERE show data seen and seperate those of retro-reflective band
15     num_retpt = 0; // number of points that belong to retro-reflective band
16     // printf("DATA is like this: point-x, point-y, point-intensity \n \n");
17
18     for(i=0;i<682;i++)
19             {
20                 // printf("%f %f %f \n",xD[i],yD[i],intensityD[i]);
21                 // if the AGC voltage is less than 4 => it is a retro-reflective
22                 if ((intensityD[i] > 2) && (intensityD[i] < 3.9)) // was 3.9
23                     {
24                         xret[num_retpt] = xD[i];
25                         yret[num_retpt] = yD[i];
26                         num_retpt = num_retpt + 1;
27                     } // if (intensityD[i] > 250)
28             } // for(i=0;i<682;i++)
29
30     // Now extract Robots from the objects =>
31         all its points are closer to each others more than 0.1
32
33     int num_robots = 1; // number of robots detected
34     int len_robot[80] = {0}; // length of the robots
35
36 // printf("\n \n The detected Robots are \n");
37
38     for (i=0;i<num_retpt;i++)
39         {
40     // distance between a point and its next neighbour
41     dist[i] = sqrt(power(xret[i+1]-xret[i],2)+power(yret[i+1]-yret[i],2));
42
43 // printf("points %f %f %f\n",dist[i],xret[i],yret[i]);
44
45     if((dist[i] < 0.1) && (xret[i]!= 0 && yret[i]!= 0) && (xret[i]!= xret[i+1])
46      && (yret[i]!= yret[i+1]) && (xret[i]!= xret[i+2]) && (yret[i]!= yret[i+2])
47       && (xret[i]!= xret[i+3]) && (yret[i]!= yret[i+3]))
48         {
49         // for the real objects
50         drobotx[num_robots][len_robot[num_robots]] = xret[i];
51         droboty[num_robots][len_robot[num_robots]] = yret[i];
52         len_robot[num_robots] = len_robot[num_robots] + 1;
53
54         } // if(dist[i] < 0.1 && fabs(xret[i]) < 0.6 &&
55
56         if (dist[i] > 0.1) // distance between two robots
57             {
58         // if distance is big shift to another object
59             num_robots = num_robots + 1;
60             } // if (dist[i] > 0.1)
61         } // for (i=0;i<=num_retpt;i++)
62
63 //////////////////////////////////////////////////////////////////////////
64 // FIRST FIND THE ROBOT WITH MAX NUMBER OF POINTS, THIS IS ROBOT-8
```

```
65 // THEN FIND THE ROBOT WITH LESS NUMBER OF POINTS, THIS IS THE OTHER ROBOT
66 // OBSTACLES WITH LESS POINTS THAN THE FOLLOWER ROBOT ARE CONSIDERED AS ERRORS
67 //////////////////////////////////////////////////////////////////////////
68
69 leaderbot = 0;
70 folbot = 0;
71 // Find the leader robot (Robot with maximum number of points)
72     max = 0;
73     indexmax = 0;
74    for (i = 0;i<num_robots;i++)
75    {
76      // the leader is the robot with more retro-points but less than 100 points
77      if ((len_robot[i] > max)&&(len_robot[i] < 100))
78      {
79             max = len_robot[i];
80         indexmax = i;
81      } // ((len_robot[i] > max)&&(len_robot[i] < 100))
82    } // (i = 1;i<=num_robots;i++)
83
84 // The leader robot is the robot with index indexmax
85 leaderbot = indexmax;
86
87 // Find the follower robot(Robot with maximum number of points after the leader)
88     max = 0;
89     indexmax = 0;
90    for (i = 0;i<num_robots;i++)
91    {
92      if ((i != leaderbot) && (len_robot[i] > max))
93      {
94             max = len_robot[i];
95         indexmax = i;
96      } // (len_robot[i] > max)
97    } // (i = 1;i<=num_robots;i++)
98
99 // The follower robot is the robot with index indexmax
100 folbot = indexmax;
101 //////////////////////////////////////////////////////////////////////
102 // HERE FIND THE CLOSEST POINT OF EACH ROBOT TO THE SCANNING ROBOT
103 //////////////////////////////////////////////////////////////////////
104     // Read the current position of the robot and set it
105     odometry_track_step(&s_odometry);
106     robotx = s_odometry.result.x;
107     roboty = s_odometry.result.y;
108     robotyaw = s_odometry.result.theta;
109
110 // For the leader robot
111    mind = 10;
112    for (i=0;i<len_robot[leaderbot];i++)
113    {
114
115     if ((drobotx[leaderbot][i] != 0)&&(droboty[leaderbot][i] != 0))
116     {
117         distrobot = sqrt ( power((robotx - drobotx[leaderbot][i]),2) +
118             power((roboty - droboty[leaderbot][i]),2) );
119         if (distrobot < mind)
```

```
120          {
121              mind = distrobot;
122              closestleaderx = drobotx[leaderbot][i];
123              closestleadery = droboty[leaderbot][i];
124           }
125      }
126     } // (i=0;i<=len_robot[leaderbot];i++)
127
128 ////////////////////////////////////////////////////////////
129      // The position of the leader
130      rx[3] = closestleaderx;
131      ry[3] = closestleadery;
132 ////////////////////////////////////////////////////////////
133 distrobot = sqrt ( power((robotx - rx[3]),2) + power((roboty - ry[3]),2) );
134
135 // if the new position of the leader is very far from its old position => ERROR!!
136 distrobotold = sqrt(power((robotx-oldleaderx),2)+power((roboty-oldleadery),2));
137
138 if (fabs(distrobotold - distrobot) > 0.05)
139 {
140 rx[3] = oldleaderx;
141 ry[3] = oldleadery;
142 printf("Taking␣the␣OLD␣distance␣between␣me␣and␣the␣leader:␣%f␣\n",distrobotold);
143 }
144
145 // If the distances are close, update (refresh) the old coordinates
146 if (fabs(distrobotold - distrobot) < 0.05)
147 {
148 printf("Taking␣the␣NEW␣distance␣between␣me␣and␣the␣leader:␣%f␣\n",distrobot);
149 }
150
151 // For the follower robot
152     mind = 10;
153     for (i=0;i<len_robot[folbot];i++)
154     {
155      if ((drobotx[folbot][i] != 0)&&(droboty[folbot][i] != 0))
156      {
157           distrobot = sqrt(power((robotx - drobotx[folbot][i]),2)+
158               power((roboty - droboty[folbot][i]),2));
159           if (distrobot < mind)
160           {
161               mind = distrobot;
162           closestfolx = drobotx[folbot][i];
163           closestfoly = droboty[folbot][i];
164           }
165      }
166     } // (i=0;i<=len_robot[folbot];i++)
167
168 // Update the other's positions for the fifth Robot
169      if(MY_id == 5)
170           {
171      rx[1] = robotx;
172      ry[1] = roboty;
173      rt[1] = robotyaw;
174      // get the position of the second robot (3rd)
```

```
175      rx[2] = closestfolx;
176      ry[2] = closestfoly;
177
178  // printf("closest point of the follower is: %f,%f \n",closestfolx,closestfoly);
179  distrobot = sqrt(power((robotx - rx[2]),2) + power((roboty - ry[2]),2));
180
181  // if the new position is very far => it is wrong!!!
182  distrobotold = sqrt(power((robotx - oldfbotx),2)+power((roboty - oldfboty),2));
183
184  if (fabs(distrobotold - distrobot) > 0.05)
185  {
186  rx[2] = oldfbotx;
187  ry[2] = oldfboty;
188  printf("Taking␣the␣OLD␣distance␣between␣me␣and␣the␣follower:%f\n",distrobotold);
189  }
190
191  // If the distances are close, update (refresh) the old coordinates
192  if (fabs(distrobotold - distrobot) < 0.05)
193  {
194  printf("Taking␣the␣NEW␣distance␣between␣me␣and␣the␣follower:%f\n",distrobot);
195  }
196          }
197  // Update the other's positions for the third Robot
198      if(MY_id == 3)
199          {
200      rx[2] = robotx;
201      ry[2] = roboty;
202      rt[2] = robotyaw;
203      // get the position of the second robot (5th)
204      rx[1] = closestfolx;
205      ry[1] = closestfoly;
206
207  // printf("closest point of the follower is: %f,%f \n",closestfolx,closestfoly);
208  distrobot = sqrt ( power((robotx - rx[1]),2) + power((roboty - ry[1]),2) );
209
210  // if the new position is very far => it is wrong!!!
211  distrobotold = sqrt(power((robotx - oldfbotx),2)+power((roboty - oldfboty),2));
212
213  if (fabs(distrobotold - distrobot) > 0.05)
214  {
215  rx[1] = oldfbotx;
216  ry[1] = oldfboty;
217  printf("Taking␣the␣OLD␣distance␣between␣me␣and␣the␣follower:%f\n",distrobotold);
218  }
219  // If the distances are close, update (refresh) the old coordinates
220  if (fabs(distrobotold - distrobot) < 0.05)
221  {
222  printf("Taking␣the␣NEW␣distance␣between␣me␣and␣the␣follower:␣%f␣\n",distrobot);
223  }
224          }
225
226          } // las_func()
```

KAYNAKLAR

[1] H.V.D. Parunak, *Making Swarming Happen*, The Conference On Swarming And C4ISR, Tyson Corner, VA, 3 Ocak 2003.

[2] M. Dorigo ve M. Birattari, *Swarm Intelligence*, Scholarpedia, sayfa 22437, 2007.

[3] M.J. Wooldridge, *An Introduction to Multiagent Systems*, John Wiley and Sons Ltd., UK, 2002.

[4] G. Weiss, *Multiagent Systems, A Modern Approach to Distributed Modern Approach to Artificial Intelligence*, The MIT Press, Cambridge, Massachusetts, Londra, İngiltere, 1999.

[5] S. Russell ve P. Norvig, *Artificial Intelligence: A Modern Approach*, Birinci baskı, Prentice Hall, New Jersey, ABD, Ocak 1995.

[6] M.J. Milford, *Springer Tracts In Advanced Robotics*, Springer-Verlag, Cilt 41, Berlin Heidelberg, 2008.

[7] R.R. Murphy, *Introduction To AI Robotics*, The MIT Press, Cambridge, Massachusetts, Londra, İngiltere, 2000.

[8] J. Katz ve A. Plotkin, *Low Speed Aerodynamics*, İkinci baskı, Cambridge University Press, Cambridge, İngiltere, 2001.

[9] R.I. Lewis, *Vortex Element Methods For Fluid Dynamic Analysis Of Engineering Systems*, Cambridge University Press, Cambridge, İngiltere, 1991.

[10] O. Uzol, I. Yavrucuk, ve N. Sezer-Uzol, *Collaborative Target Tracking For Swarming MAVs Using Potential Fields And Panel Methods*, AIAA Paper 2008-7167, AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii, ABD, 18-21 Ağustos 2008.

[11] S. Waydo ve M. Murray, *Vehicle Motion Planning Using Stream Functions*, Proceeding of the IEEE International Conference on Robotics and Automation, sayfa 2484-2491, Taipei, Tayvan, 14-19 Eylül 2003.

[12] J. Sullivan, S. Waydo, ve M. Campbell, *Using Stream Functions For Complex Behavior And Path Generation*, AIAA Paper 2003-5800, AIAA Guidance, Navigation, and Control Conference and Exhibit, Austin, Teksas, ABD, 2003.

[13] G. Ye, H. Wang, ve K. Tanaka, *Coordinated Motion Control Of Swarm With Dynamic Connectivity In Potential Flows*, Proc. 16'th IFAC World Congress, Prague, Czeck Republic, Temmuz 2005.

[14] G. Ye, H. Wang, K. Tanaka, ve Z. Guan, *Managing Group Behaviours In Swarm Systems By Associations*, Proc. of the 2006 American Control Conference, pp. 3537-3544, Minneapolis, Minnesota, ABD, Haziran 2006.

[15] Y. Zhang ve K. Valavanis, *Sensor-Based 2-D Potential Panel Method For Robot Motion Planning*, Robotica, Cilt 14, sayfa 81-89, 1996.

[16] Y. Zhang ve K. Valavanis, *A 3-D Potential Panel Method For Robot Motion Planning*, Robotica, Cilt 15, sayfa 421-434, 1997.

[17] A.T. Şamiloğlu, O. Çayırpunar, V. Gazi, ve A.B. Koku, *An Experimental Set-up For Multi-Robot Applications*, International Workshop on Standarts and Common Platforms for Robotics (SCPR2008), sayfa 539-550, Venedik, İtalya, Kasım 2008.

[18] O. Khatib, *Real-Time Obstacle Avoidance For Manipulators And Mobile Robots*, The International Journal of Robotics Research, Cilt 5, No. 1, sayfa 90-98, 1986.

[19] E. Rimon ve D. E. Koditschek, *Exact Robot Navigation Using Artificial Potential Functions*, IEEE Trans. on Robotics and Automation, Cilt 8, No. 5, sayfa 501-518, Ekim 1986.

[20] V. Gazi ve B. Fidan, *Coordination And Control Of Multi-Agent Dynamic Systems: Models And Approaches*, Proceedings of the SAB06 Workshop on Swarm Robotics, Lecture Notes in Computer Science (LNCS) 4433, pp. 71-102, Berlin Heidelberg, Almanya, 2007.

[21] V. Gazi ve K. M. Passino, *Stability Analysis Of Swarms*, IEEE Trans. on Automatic Control, Cilt 48, No. 4, sayfa 692-697, Nisan 2003.

[22] V. Gazi ve K. M. Passino, *Stability Analysis Of Social Foraging Swarms*, IEEE Trans. on Systems, Man, and Cybernetics, Cilt 34, No. 1, sayfa 539-557, Şubat 2004.

[23] V. Gazi ve K. M. Passino, *A Class Of Attraction/Repulsion Functions For Stable Swarm Aggregations*, International Journal Of Countrol, Man, and Cybernetics, Cilt 77, No. 18, sayfa 1567-1579, Aralık 2004.

[24] V. Gazi, *Swarm Aggregations Using Artificial Potentials And Sliding Mode Control*, IEEE Trans. on Robotics, Cilt 21, No. 6, sayfa 1208-1214, Aralık 2005.

[25] V. Gazi, B. Fidan, Y. Hanay, ve M. Köksal, *Aggregation, Foraging, and Formation Control of Swarms with Non-Holonomic Agents Using Potential Functions and Sliding Mode Techniques*, Turkish Journal of Electrical Engineering and Computer Sciences, Cilt 15, No. 2, sayfa 149-168, 2007.

[26] V. Gazi, M. Köksal, ve B. Fidan, *Aggregation In A Swarm Of Non-Holonomic Agents Using Artificial Potentials And Sliding Mode Control*, European Control Conference, Temmuz 2007.

[27] A.T. Şamiloğlu, V. Gazi, ve A.B. Koku, *Comparison Of Three Orientation Agreement Strategies In Self-Propelled Particle Systems With Turn Angle Restrictions in Synchronous and Asynchronous Settings*, Asian Journal of Control, Cilt 10, No. 2, sayfa 212-232, Mayıs 2008.

[28] H. Kawata, A. Ohya ve S. Yuta, *Development Of Ultra-Small Lightweight Optical Range Sensor System*, Proceedings 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS'05, Edmonton, Kanada, sayfa 3277-3282, Ağustos 2005.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, Adı	: Merheb, Abdel-Razzak
Uyruğu	: Lübnan
Doğum tarihi ve yeri	: 13.03.1983 Lübnan
Medeni hali	: Bekar
Telefon	: 0 (312) 292 42 91
Faks	: 0 (312) 292 40 91
e-mail	: armerheb@etu.edu.tr

Eğitim

Derece	**Eğitim Birimi**	**Mezuniyet Tarihi**
Lisans	Eskişehir Osmangazi Üniversitesi Elektrik ve Elektronik Mühendisliği	2005

İş Deneyimi

Yıl	**Yer**	**Görev**
2008-2010	TOBB ETÜ	Eğitim Asistanı
2006-2008	ODESA Ltd. Şti.	Ar-Ge Mühendisi

Yabancı Dil

Arapça
Fransızca
İngilizce

Yayınlar

A. Merheb, V. Gazi, ve N. Sezer-Uzol, "Experimental Study Of Robot Formation Control And Navigation Using Potential Functions And Panel Method", *The joint 41st International Symposium on Robotics (ISR 2010) and the 6th German Conference on Robotics (ROBOTIK 2010)*, ,s. 586-593, Münih, Almanya, Haziran 2010.

A. Merheb, Y. Atas, V. Gazi, ve N. Sezer-Uzol, "Implementation of Robot Formation Control and Navigation Using Real-Time Panel Method", *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. (IROS 2010)*, Taipei, Tayvan, 18-22 Ekim, 2010.