BAŞKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ENGELLERİN BULUNDUĞU ORTAMDA GEZGİN ROBOTUN EN İYİ YOLU BULMASI VE İZLEMESİ

VOLKAN ARICI

YÜKSEK LİSANS TEZİ
ANKARA
2008

ENGELLERİN BULUNDUĞU ORTAMDA GEZGİN ROBOTUN EN İYİ YOLU BULMASI VE İZLEMESİ

TWO-STAGE SHORTEST PATH ALGORITHM WITH OBSTACLE AVOIDANCE FOR MOBILE ROBOTS

VOLKAN ARICI

Başkent Üniversitesi

Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliğinin

ELEKTRİK-ELEKTRONİK Mühendisliği Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2008

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma, jürimiz tarafından **ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan :

Doç. Dr. Nizami Gasilov

Üye :

Doç. Dr. İsmail Avcıbaş

Danışman :

Yrd. Doç. Dr. Mustafa Doğan

ONAY

Bu tez 11/06/2008 tarihinde Enstitü Yönetim Kurulunca belirlenen yukarıdaki jüri üyeleri tarafından kabul edilmiştir.

/06/2008

Prof.Dr. Emin AKATA

FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRÜ

TEŞEKKÜR

Çalışmalarımı yönlendiren, araştırmalarımın her aşamasında bilgi, öneri ve yardımlarını esirgemeyerek akademik ortamda olduğu kadar beşeri ilişkilerde de engin fikirleriyle yetişme ve gelişmeme katkıda bulunan danışman hocam sayın Yrd. Doç. Dr. Mustafa Doğan'a, çalışmalarım sırasında önemli katkılarda bulunan ve yönlendiren saygıdeğer hocam Doç. Dr. Nizami Gasilov'a, tez çalışmasında değerli görüş ve düşüncelerini belirten sayın hocam Doç. Dr. İsmail Avcıbaş'a, çalışmalarım süresince maddi manevi desteklerini esirgemeyen değerli enstitü müdürümüz sayın Prof. Dr. Emin AKATA' ya en içten teşekkürlerimi sunarım.

Son olarak beni bugünlere getiren, manevi ve maddi destekleriyle daima yanımda olan ve beni her zaman destekleyen çok değerli aileme en içten teşekkürlerimi sunarım.

ÖZ

ENGELLERİN BULUNDUĞU ORTAMDA GEZGİN ROBOTUN EN İYİ YOLU BULMASI VE İZLEMESİ

Volkan ARICI

Başkent Üniversitesi Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Birçok uygulamalarda, hareketi denetlenen nesnenin (robotun) engellere çarpmadan bir başlangıç konumdan bir hedef konuma en kısa yolla gitmesinin sağlanması gerekir. Söz konusu problem, engellerden sakınma optimizasyon problemi olarak da adlandırılır. Bu çalışmada engellerin, farklı yarıçaplı daireler şeklinde ve hareketsiz oldukları varsayılır. Nesnenin noktasal boyutta olduğu kabul edilmiştir.

Problemin sayısal çözümü için iki aşamalı algoritma önerilir. Birinci aşamada, bir adım için optimal yöntem uygulanmıştır. Her adımda nesnenin şu andaki konumu ile hedef konum arasında düz yol üzerindeki ilk engel, tek engel gibi düşünülmüştür. Yöntem, geometrik gösterimlere dayanarak gerçekleştirilmiştir. Birinci aşamadan elde edilen yol optimal olmayabilir, ama bu yolun uzunluğu esas alınarak optimal yolun yer aldığı bölge, bir elipsle sınırlandırılarak küçültülebilir. Elde edilen bölge bir sonraki aşamada işlem tasarrufu yapılmasını sağlamıştır.

Genel algoritmanın ikinci aşamasında engeller arasındaki minimal mesafe dikkate alınarak, bölge karelere bölünmüştür. Engellerle kesişimi olan karelere geçişler yasaklanarak ayrıklaştırma yapılmıştır. Bu yolla elde edilen problem, çizge kuramında en kısa yolun bulunması problemi olarak yorumlanır ve Dijkstra algoritmasının uygulanması ile çözülmüştür. Problemin özelliklerine dayanarak

Dijkstra algoritmasının daha verimli kullanımını sağlayan bazı değiştirmeler yapılmıştır.

Önerilen iki aşamalı algoritmayı sınamak için sayısal benzetimler yapılmıştır. Benzetimlerde rasgele engeller oluşturulur. Bir hedef konum seçilir. Farklı başlangıç konumlar alınarak önerilen algoritma çalıştırılır. Sonuçlar, önerilen algoritmanın engellerden sakınma optimizasyon probleminin çözümü için kullanılabilir olduğunu göstermektedir.

Deneysel çalışmada, deney düzeneğinden tek kamera ile alınan sayısal görüntülere, görüntü işleme teknikleri uygulanarak, dairesel kesitli engellerin konumlarının bulunması ve başlangıç konumundan bitiş konumuna yol optimizasyonu gerçekleştirilmiştir. Bulunan bu yol gezgin robot kullanılarak izlenmiştir. Deneysel olarak; bu çalışmada gezgin robotun en iyi yolu bulması ve izlemesi başarı ile gerçekleştirilmiş ve farklı boyut ve konumlardaki dairesel engeller için test edilmiştir.

ANAHTAR SÖZCÜKLER: Engellerden sakınma, çizge kuramı, en kısa yol problemi, Dijkstra algoritması, Gezgin robotlar.

Danışman: Yrd. Doç. Dr. Mustafa DOĞAN, Başkent Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü.

ABSTRACT

TWO-STAGE SHORTEST PATH ALGORITHM WITH OBSTACLE AVOIDANCE FOR MOBILE ROBOTS

Volkan ARICI

Başkent University Institute of Science
The Department of Electrical and Electronics Engineering

In most of the path-planning applications, the controlled object (mobile robot) is expected to reach its predetermined target by following the shortest path and avoiding the obstacles. This navigation problem is also called optimal obstacle avoidance. In this work, obstacles are assumed to be motionless circles in different sizes. The object is supposed to be a point robot.

The two-stage algorithm is proposed to find a numerical solution to the problem. At first stage, the method, which is optimal for one step, is applied iteratively. In every step of the method the first obstacle on the straight line between the current position of the object and the target is assumed to be a single obstacle. The proposed method is realized with using geometric representations. Some evaluations are made to prove that the method is convergent. The path obtained at the first stage might not be optimum. However, its length can be used to limit the feasible region through an ellipse, which contains the shortest path. Thus, the reduced search space makes the next stage more efficient and endurable for real-time applications.

In the second stage of the algorithm, the elliptic region is meshed with squares the side length of which is set in agreement with the minimum distance between obstacles. It is prohibited to pass through the squares that intersect obstacles. Thus, by discretization the problem becomes the shortest path problem in a graph, and is solved by applying the Dijkstra's algorithm.

The proposed two-stage algorithm is verified with numerical simulations. Obstacles are chosen randomly. A target position is selected and fixed. For different starting points, the algorithm is tested repeatedly. The results show that the proposed algorithm can be applied to find an optimal solution for the obstacle avoidance problem.

In experimental work, images were taken from an experimental set-up with a single camera. Identification of circular objects was realized by using image processing techniques. Shortest path optimization was performed by defining starting and target points. Experimentally, shortest path algorithm with obstacle avoidance for mobile robot have been designed, tested and applied successfully.

KEYWORDS: Obstacle avoidance, graph theory, shortest path problem, Dijkstra algorithm, Mobile Robots.

Advisor: Asst. Prof. Dr. Mustafa DOĞAN, Başkent University Electrical and Electronics Engineering Department.

İÇİNDEKİLER LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

ÇİZELGELER LİSTESİ

Sayfa

SİMGELER LİSTESİ

u_i	:	1.düğümden i.düğüme en kısa uzaklık
d_{ij}	:	(i,j) kenar uzunluğu
$f(x)$	:	A* algoritması için Maliyet işlevi
$g(x)$	:	A* algoritması için Uzaklık işlevi
$f(x)$	:	A* algoritması için Buluşsal (heuristic) işlev
L	:	Başlangıç ve bitiş noktası arasındaki uzaklık
δ	:	Engeller arası minimum uzaklık

1. GİRİŞ

Engellerden sakınma (obstacle avoidance) probleminin çözümü için çeşitli yöntemler uygulanır [3, 7]. Bir gezgin robotun engellerden kaçınabilmesi için geliştirilen gerçek zamanlı yöntemlerden birisi, potansiyel alanlar yaklaşımıdır [3, 27]. Bu yaklaşımın en önemli avantajı, hareket planlaması ile robotun denetiminin bütünselleştirmesi ve bundan dolayı gerçek zamanda verimli olmasıdır. Bununla birlikte bazı durumlarda yerel minimumlardan kurtulamamasından dolayı hedefe varamadan yolu sonlandırması, en önemli dezavantajıdır. Bu sorunu aşmak için [3]'de harmonik potansiyeller, [27]'de ise, gezinim işlevleri kullanılmıştır. Yine de belirtmek gerekir ki, alınan bu önlemler, engellerden kaçınmakta yararlıdırlar, ancak optimal yolun bulunmasına önemli bir katkı sağlamıyorlar. Buna ek olarak, robotun çalışma uzayının boyutu arttığında gezinim işlevinin hesaplanması zorlaşır ve çevrimiçi yapılması imkânsız hale gelir [17]. Ayrıca, gezinim işlevları, tanıma göre türevlenebilir işlevlerdir ve bu yüzden parçalı sürekli denetim gerektiren uygulamalarda sorun çıkartabilirler [27].

Optimal yolun bulunmasında olasılıksal haritalandırma gibi rastsal yöntemler de kullanılmıştır [16, 17]. Bu yöntemlerin, yukarıda anlatılanlara göre üstünlüğü daha genel durumlarda ve daha güvenli olarak uygulanabilmeleridir. Teorik analizlerin karmaşıklaşması ise bu yöntemlerin dezavantajıdır.

Gezgin robotlar için en kısa yolun bulunması konusunda verimli yöntemler [11] ve [19]'da önerilmiştir. Her iki makale de çizge kuramından yola çıkarak en kısa yolu bulmaya adanmıştır. Bunun dışında dinamik programlama (Hamilton-Jacobi-Bellman) [1, 6, 30] yöntemleri de kullanılır. Örneğin, [1]'da en kısa yolun bulunması için azalan eğimi izleyerek optimale yakın (near-optimal) çözümler bulunmuştur. Bu yöntemin dezavantajı, global anlamda en kısa olan yolu bulmamasıdır.

Sunulan çalışmada [1, 11, 20] makalelerindeki yaklaşımlardan yararlanılarak yola çıkılmıştır. Bu yaklaşımlar üzerine önce optimale yakın çözüm elde edilir ve onu kullanarak optimal çözümün yer aldığı bölge bir elipsle sınırlandırılır. Ancak bundan sonra global anlamda optimal çözümün bulunması aşamasına geçilir. Optimal çözümün, başlangıçtaki bölgeye göre çok daha dar olan bir bölge üzerinde aranması, bu çalışmada getirilen en önemli yeniliktir.

1.1. Problemin Tanımı

Dikdörtgen şeklinde bir bölge ve bu bölgede yer alan hareketsiz dairesel engeller verilsin. Engeller arasındaki mesafelerin verilen bir eşik değerin üstünde olduğu kabul edilir. Araştırılan problem, aşağıdaki sorunun yanıtlanmasından ibarettir: Gezgin robot, engellere çarpmadan, verilen S başlangıç konumundan verilen F hedef konumuna en kısa yolla nasıl gidebilir?

Problemin sayısal çözümü için iki aşamalı algoritma önerilir. Bu aşamaların ayrıntıları ilerideki bölümlerde verilecektir.

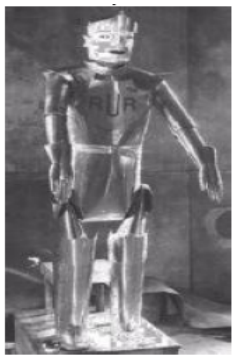
1.2. Gezgin Robotun İzleyeceği Yolu Bulunmasında Uygulanan Yöntem

Gezgin robotun izleyeceği yolu bulunmasında uygulanan yöntem iki ana kısımdan oluşmaktadır. Birinci kısımda görüntü işleme teknikleri kullanılarak engellerin bulunduğu ortamın görüntüsünü almak ve engellerin tespiti (merkez koordinatlarının bulunması), ikinci kısımda ise iki aşamalı algoritma kullanarak gezgin robotun engellere çarpmadan izleyeceği en kısa yolun bulunması anlatılmaktadır.

2. GEZGİN ROBOTLAR VE ENGELLERDEN SAKINMA

Günümüzde robotlar birçok alanda kullanılmaktadır. Artan kullanım alanı robotların çok farklı ortamlarda çalışabilmeleri ihtiyacını doğurmuştur. Bu nedenle, robotların farklı ortamlarda çalışabilmeleri için geliştirme çalışmaları hız kazanmıştır. Robotların gelişimi açısından birçok çalışma yürütülmektedir.

Robot kelimesi ilk olarak 1921 yılında Çek oyun yazarı Karel Capek'in "Rossum'un evrensel robotları" (R.U.R) adlı oyununda kullanılmıştır. Yazar zorunlu iş anlamındaki "robota" kelimesiyle işçi anlamına gelen "robotnik" kelimelerini birleştirerek "robotic" kelimesini türetmiştir. Ancak, genel olarak bilinenin aksine robot kelimesini icat eden kişi Karel Capek'in abisi Joseph Capek'tir [12]. Carel Kapek R.U.R. adlı oyununda robot kelimesine karşılık olarak, kendi kendilerine çalışabilen işçiler tanımlamasını yapmıştır.



Şekil 2.1. Capek'in oyunundaki robot

Robot kavram olarak ise duyargaları (sensör) ile çevresini algılayan, algıladıklarını yorumlayan, bunun sonucunda karar alan (yapay zeka), karar sonucuna göre davranan, eylem olarak hareket organlarını çalıştıran veya durduran bir aygıttır. Robotların birçoğu bilgisayar ya da işlemciler tarafından denetlenen kollardan oluşmakta, bazıları da tekerlekler, bacaklar veya paletler üzerinde hareket eden yapılardan oluşmaktadır. Bunun yanında robotlar,

başlangıçtaki durağan yapılarına ve işlevselliklerine kıyasla daha gezgin ve daha çok kendi kendilerine hareket edebilen yapılara kavuşmuşlardır.

Gezginlik, robotların yerlerini değiştirebilme yeteneklerinin olup olmaması olarak tanımlanabilir. Robot evriminin başlangıcındaki robotlar endüstriyel amaçlı geliştirildikleri ve büyük ölçekli yapılara sahip olduklarından dolayı gezginlik özelliğine sahip değillerdi. Günümüzde ise robotlar, endüstriyel üretim haricinde günlük hayatımızda ve uzay araştırmalarında kullanılmaya başlanmıştır. Örneğin, yer süpürgesi robotu, posta robotu, çöp toplama robotu veya Mars'a gönderilen robotlar gezginlik özelliğine sahiplerdir.

Robotların gezginlik özelliğini yerine getirebilmeleri için değişen durumlara göre hareketlerini değiştirme yetisine sahip olmaları gerekir. Kısacası otonom özellikleri olmalıdır. Otonom olma özelliği, robot üzerinde bulunan işlemci sayesinde robotun daha önceden belirlenmiş veya belirlenmemiş bir durum karşısında karar verebilme yeteneğine sahip olmasıdır. Robot hareketine karar vermeden önce çevresinden veri toplamalıdır. Robotun algılayıcıları aracılığı ile çevreden elde ettiği veriler bütünü sayesinde robot, sonraki hareketine karar verebilir. Örnek olarak Mars'a, Ay'a, insan için tehlikeli olabilecek ortamlara veya insanın giremeyeceği bir mağaraya gönderilecek robotun otonom olması gerekmektedir.

Gezgin robotların görevlerini yerine getirebilmeleri için bulunduğu ortamı bilmesi veya öğrenmesi, kendi konumunu bilmesi gerekmektedir. Robotun ortamdaki gezinimi sırasında öncelikle engellerden sakınması gerekmektedir.

Deneysel çalışmalar için öğretim ve uygulama amaçlı olarak üretilmiş Pioneer 3-DX robotu kullanılmıştır [13]. Kullandığımız robot 2 tekerlekli gezgin bir robottur. İki tekerlekli robotlar, manevra kabiliyeti en yüksek robotlardır. Kendi etraflarında 360 derece dönebilirler. Ayrıca, robot üzerinde 14 adet ses ötesi algılayıcı (sonar) ve yüksek çözünürlükte kodlayıcılar bulunmaktadır.

3. EN KISA YOL BULMA ALGORİTMALARI

Bu bölümde en çok kullanılan en kısa yol bulma algoritmalarından bahsedilecektir. Bunlar Dijkstra algoritması ve A* (A-yıldız) algoritmasıdır. İki algoritmanın en önemli farkı en kısa yol bulunurken çizge üzerinde gezilen düğüm sayısıdır.

3.1. Dijkstra Algoritması

En kısa yolu bulma probleminde kullanılan en eski çözümlerden biri 1959 yılında yayınlanan Dijkstra Algoritmasıdır [5]. Dijkstra algoritması bir başlangıç düğümünden ortamdaki diğer bütün düğümlere olan en kısa yolu bulmada kullanılır.

d_{ij} (≥ 0) → (i,j) i. Düğüm ile j.düğüm arası uzunluk olmak üzere;

Dijkstra algoritması, başlangıç düğümüyle ağdaki başka bir düğüm arasındaki en kısa yolu belirlemek üzere tasarlanmıştır. Algoritma bir etiketleme prosedürü kullanır. Etiketleme şu şekilde yapılmaktadır:

u_i → Başlangıç düğümden i.düğüme en kısa uzaklık,

k → Başlangıç düğümden i.düğüme kadar olan yolda i.den önceki tepenin numarası,

j düğümü için etiket:

$[u_j, i] = [u_i + d_{ij}, i]$, $d_{ij} \geq 0$ düzenlenir.

Düğüm etiketleri geçici ve kalıcı olarak işaretlenirler. Geçici etiket, aynı düğüme daha kısa bir yol bulunursa başka bir etiketle değiştirilir. Daha iyi bir yol bulunamayacaksa etiket kalıcı olarak işaretlenir. Algoritma adım adım şu şekilde açıklanabilir:

Çizelge 3.1 Dijkstra algoritması

0.adım :

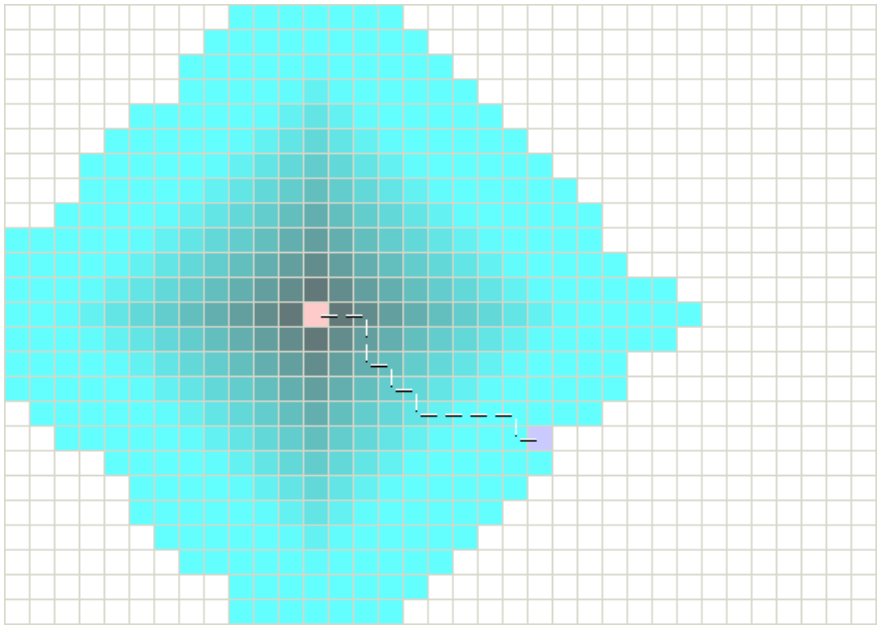
1.düğüm(başlangıç düğümü) kalıcı etiketle [0,-] şeklinde işaretlenir.i = 1 dir.

i.adım :

j'nin kalıcı etiketlenmemiş olması koşuluyla, i. düğümden ulaşılabilen her j düğümü için geçici $[u_i + d_{ij}, i]$ etiketleri hesaplanır. j düğümü başka bir k düğümü için de $[u_j, k]$ ile zaten etiketli ise ve $u_i + d_{ij} < u_j$ ise $[u_j, k]$, $[u_i + d_{ij}, i]$ ile değiştirilir.

Tüm etiketler kalıcı ise işlem durdurulur. Aksi halde tüm geçici etiketler $[u_r, s]$ arasından en kısa mesafeli olanı seçilir (eşitlik varsa herhangi biri seçilebilir.) i=r olarak atanır ve i. adım tekrarlanır.

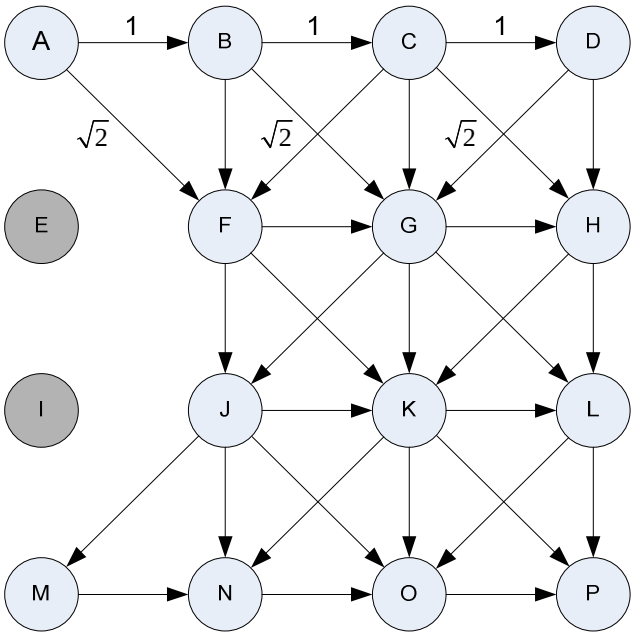
Dijkstra algoritmasında, başlangıç noktasından başlayarak bitiş noktasına ulaşıncaya kadar tüm düğümlerin gezilmesiyle en kısa yol bulunur. Düğümler gezilirken en yakın ve hiç uğranmamış düğüm, erişilme maliyeti hesaplanarak, gezilen düğümler kümesine eklenir. Dijkstra algoritması ile en kısa yolun bulunması garantidir. Aşağıdaki şekilde başlangıç noktası pembe kare ile bitiş noktası mor kare ile ve Dijkstra algoritması tarafından gezilen düğümler de mavi karelerle gösterilmiştir.



Şekil 3.1. Dijkstra algoritması tarafından gezilen düğümler

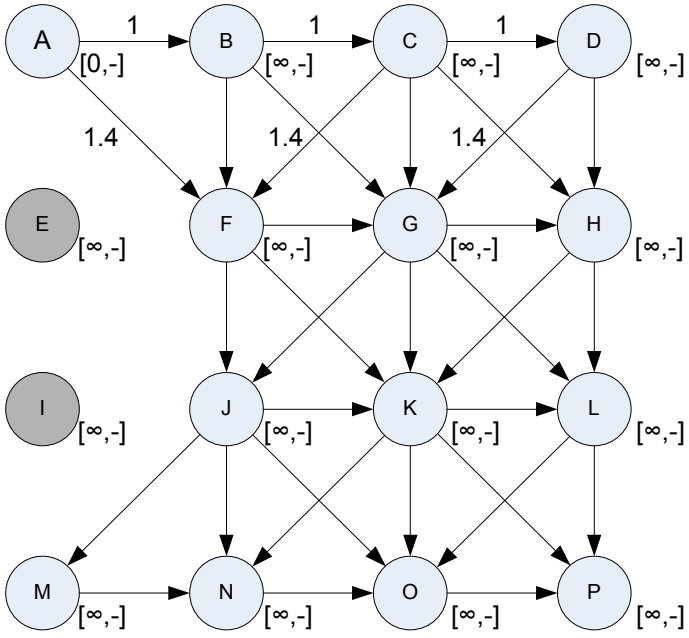
Dijkstra algoritması tarafından gezilen düğümlerdeki mavi renk dışa gidildikçe açıklaşmıştır, ayni renk tonundaki kareler başlangıç noktasına eşit uzaklıktadır. Dijkstra algoritması bir örnek üzerinde ilerleyen kısımlarda detaylı bir şekilde açıklanacaktır

Örnek: Aşağıdaki verilen ağı kullanarak Dijkstra algoritması ile kısa yolları bulalım. Bir düğümden komşu düğüme bir adım için yatay ve dikey uzaklıklar 1 birim, çapraz uzaklıklar $\sqrt{2}$ birim olarak kabul edilmiştir. Bu örnekte amaç A düğümünden P düğümüne en kısa yolu bulmaktır.



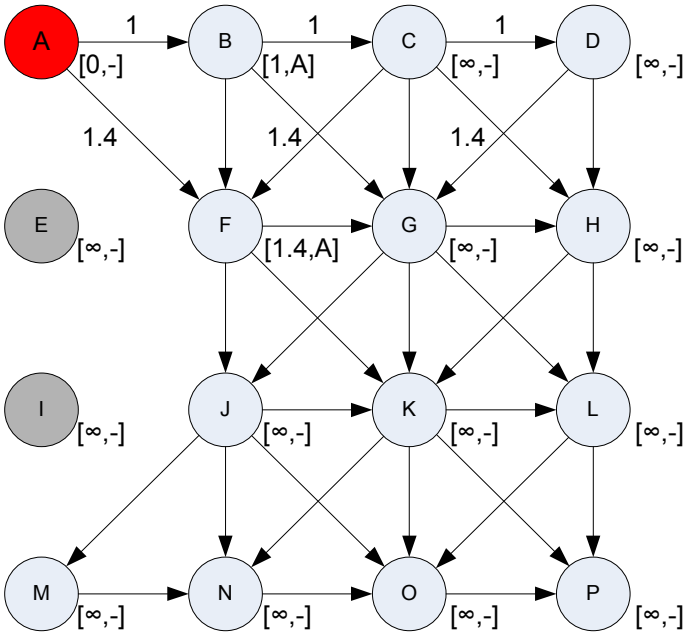
Şekil 3.2. Dijkstra algoritmasının anlatımı için kullanılan çizge

0. adım: A düğümüne [0,-] kalıcı etiketi atanır



Şekil 3.3. Dijkstra algoritmasının 0. adımında çizge

1. adım: B ve F düğümlerine A düğümünden (en son kalıcı etiketlenen) ulaşılır ve düğümler aşağıdaki gibi etiketlenir.



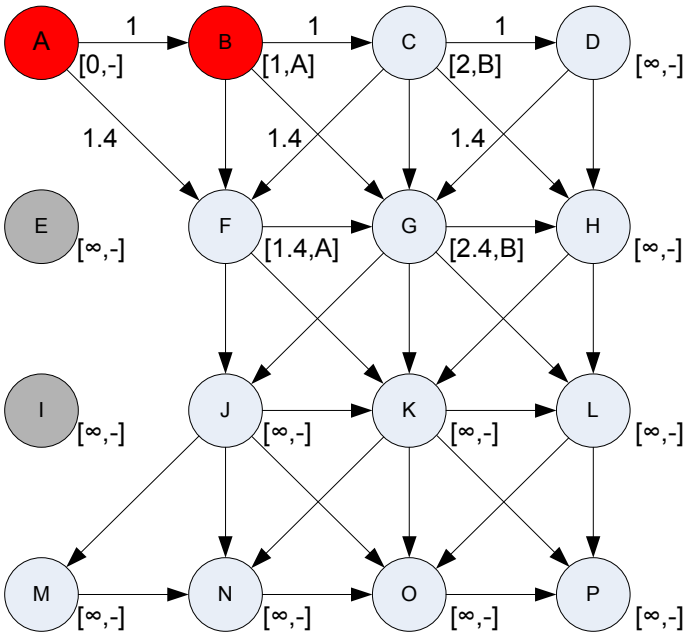
Şekil 3.4. Dijkstra algoritmasının 1. adımında çizge

Çizelge 3.2 Dijkstra algoritması örneği 1. adım

Düğüm	Etiket	Durum
A	[0, -]	kalıcı
B	[0+1,A]=[1,A]	geçici
F	[0+1.4,A]=[1.4,A]	geçici

Burada B düğümüne olan yol en kısa olduğu için bir sonraki adımda durumu kalıcı olarak değiştirilir.

2. adım: C, F ve G düğümlerine B düğümünden ulaşılmaktadır ve yeni etiketleme aşağıdaki gibidir.



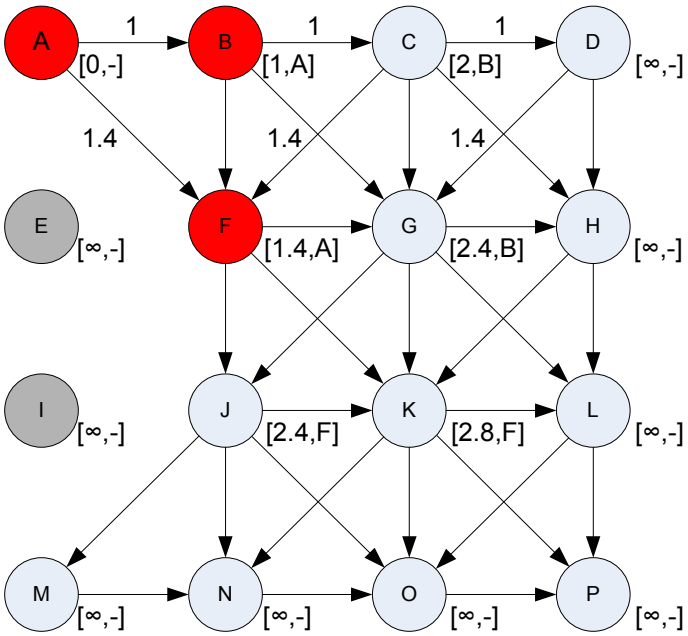
Şekil 3.5. Dijkstra algoritmasının 2. adımında çizge

Çizelge 3.3 Dijkstra algoritması örneği 2. adım

Düğüm	Etiket	Durum
A	[0, -]	kalıcı
B	[1,A]	kalıcı
F	[1.4,A]	geçici
C	[1+1,B]=[2,B]	geçici
G	[1+1.4,B]=[2.4,B]	geçici

Burada F düğümüne olan yol en kısa olduğu için bir sonraki adımda durumu kalıcı olarak değiştirilir.

3. adım: G, J ve K düğümlerine F düğümünden ulaşılmaktadır ve yeni etiketleme aşağıdaki gibidir.



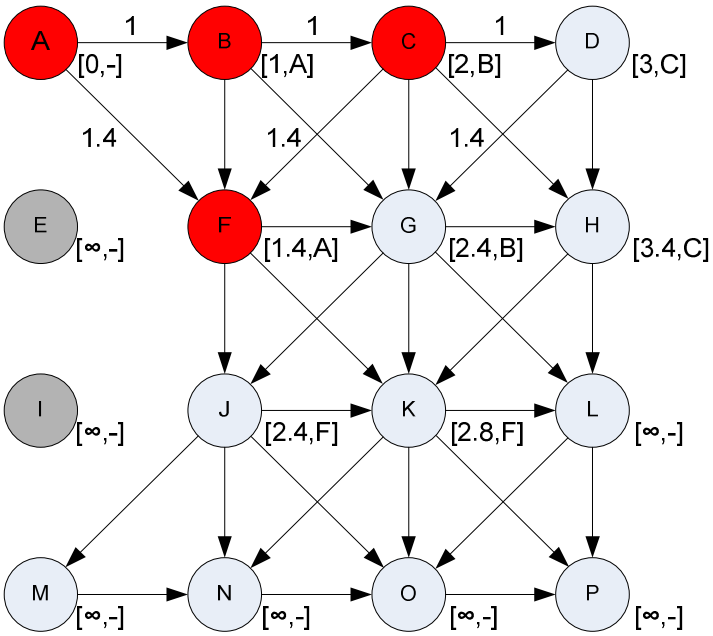
Şekil 3.6. Dijkstra algoritmasının 3. adımında çizge

Çizelge 3.4 Dijkstra algoritması örneği 3. adım

Düğüm	Etiket	Durum
A	[0, -]	kalıcı
B	[1,A]	kalıcı
F	[1.4,A]	kalıcı
C	[2,B]	geçici
G	[2.4,B]	geçici
J	[1.4+1,F]=[2.4,F]	geçici
K	[1+1.4,F]=[2.4,F]	geçici

Burada C düğümüne olan yol en kısa olduğu için bir sonraki adımda durumu kalıcı olarak değiştirilir.

4. adım: D, G ve H düğümlerine C düğümünden ulaşılmaktadır ve yeni etiketleme aşağıdaki gibidir.



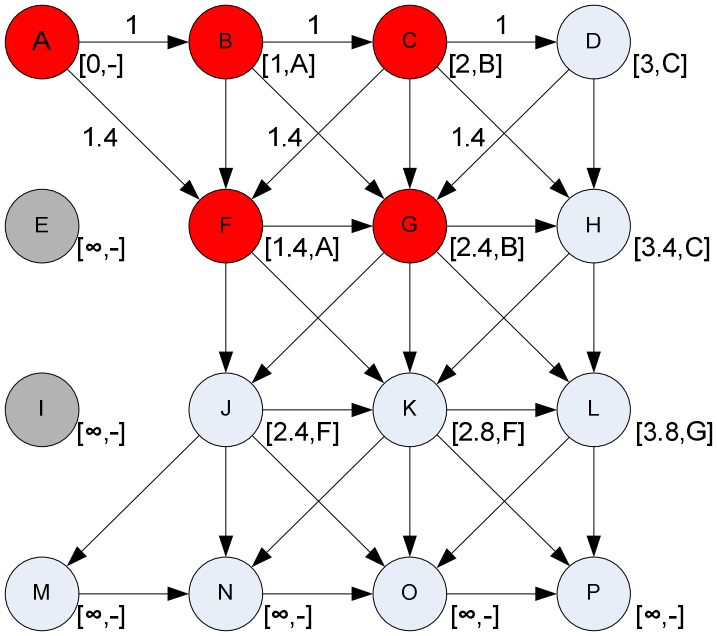
Şekil 3.7. Dijkstra algoritmasının 4. adımında çizge

Çizelge 3.5 Dijkstra algoritması örneği 4. adım

Düğüm	Etiket	Durum
A	[0, -]	kalıcı
B	[1,A]	kalıcı
F	[1.4,A]	kalıcı
C	[2,B]	kalıcı
G	[2.4,B]	geçici
J	[2.4,F]	geçici
K	[2.8,F]	geçici
D	[2+1,C]=[3,C]	geçici
H	[2+1.4,C]=[3.4,C]	geçici

Burada G ve J düğümlerine olan yollar en kısa olduğu için bir sonraki adımda herhangi birinin durumu kalıcı olarak değiştirilir. Bu adımda ise G düğümü seçilmiştir.

5. adım: H, J, K ve L düğümlerine G düğümünden ulaşılmaktadır ve yeni etiketleme aşağıdaki gibidir.



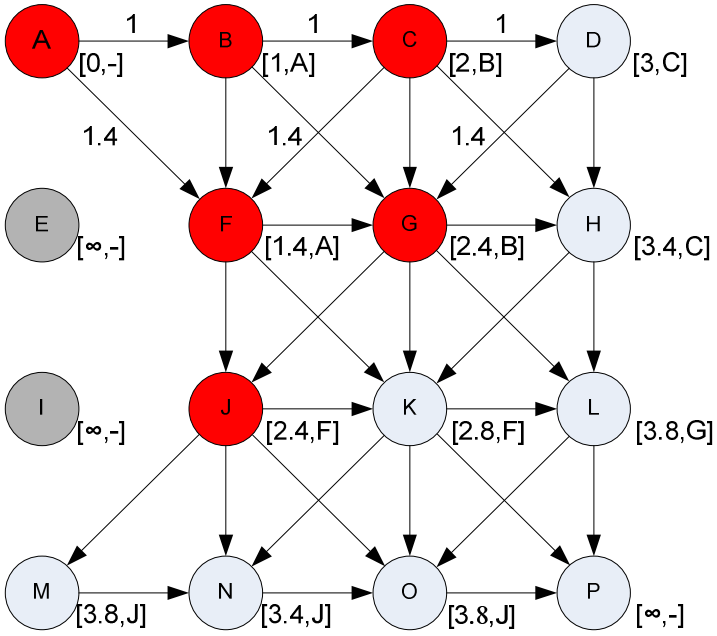
Şekil 3.8. Dijkstra algoritmasının 5. adımında çizge

Çizelge 3.6 Dijkstra algoritması örneği 5. adım

Düğüm	Etiket	Durum
A	[0, -]	kalıcı
B	[1,A]	kalıcı
F	[1.4,A]	kalıcı
C	[2,B]	kalıcı
G	[2.4,B]	kalıcı
J	[2.4,F]	geçici
K	[2.8,F]	geçici
D	[3,C]	geçici
H	[3.4,C]	geçici
L	[2+1.4,G]=[3.8,G]	geçici

Burada J düğümüne olan yol en kısa olduğu için bir sonraki adımda durumu kalıcı olarak değiştirilir.

6. adım: K, M, N ve O düğümlerine J düğümünden ulaşılmaktadır ve yeni etiketleme aşağıdaki gibidir.



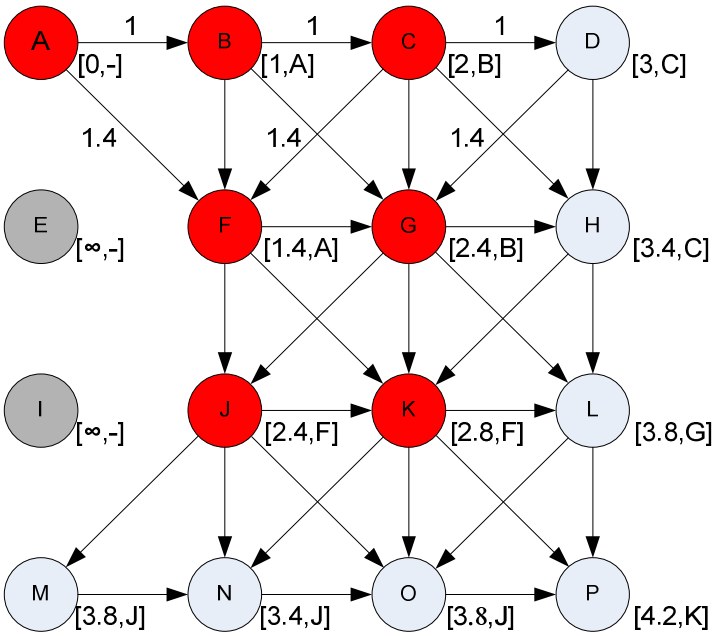
Şekil 3.9. Dijkstra algoritmasının 6. adımında çizge

Çizelge 3.7 Dijkstra algoritması örneği 6. adım

Düğüm	Etiket	Durum
A	[0, -]	kalıcı
B	[1,A]	kalıcı
F	[1.4,A]	kalıcı
C	[2,B]	kalıcı
G	[2.4,B]	kalıcı
J	[2.4,F]	kalıcı
K	[2.8,F]	geçici
D	[3,C]	geçici
H	[3.4,C]	geçici
L	[3.8,G]	geçici
M	[2.4+1.4,J]=[3.8,J]	geçici
N	[2.4+1,J]=[3.4,J]	geçici
O	[2.4+1.4,J]=[3.8,J]	geçici

Burada K düğümüne olan yol en kısa olduğu için bir sonraki adımda durumu kalıcı olarak değiştirilir.

7. adım: L, N, O ve P düğümlerine K düğümünden ulaşılmaktadır ve yeni etiketleme aşağıdaki gibidir.



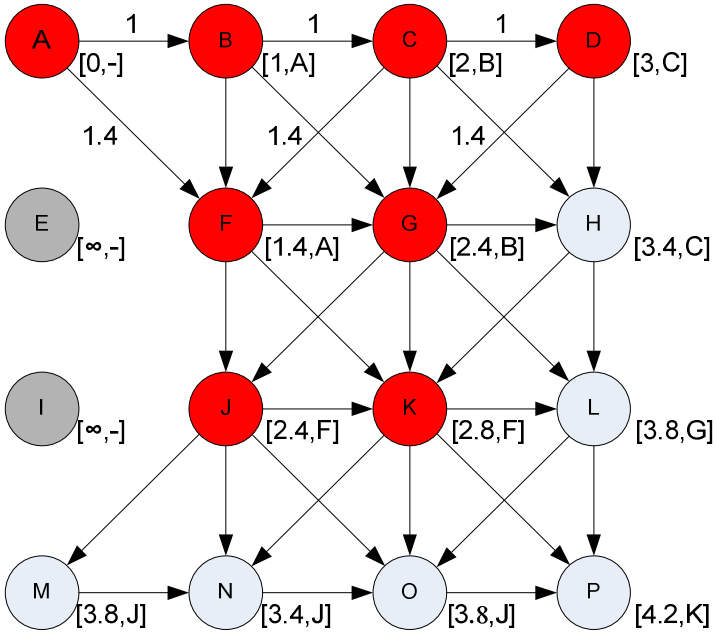
Şekil 3.10. Dijkstra algoritmasının 7. adımında çizge

Çizelge 3.8 Dijkstra algoritması örneği 7. adım

Düğüm	Etiket	Durum
A	[0, -]	kalıcı
B	[1,A]	kalıcı
F	[1.4,A]	kalıcı
C	[2,B]	kalıcı
G	[2.4,B]	kalıcı
J	[2.4,F]	kalıcı
K	[2.8,F]	kalıcı
D	[3,C]	geçici
H	[3.4,C]	geçici
L	[3.8,G]	geçici
M	[3.8,J]	geçici
N	[3.4,J]	geçici
O	[3.8,J]	geçici
P	[2.8+1.4,K]=[4.2,K]	geçici

Burada ise D düğümüne olan yol en kısa olduğu için bir sonraki adımda durumu kalıcı olarak değiştirilir.

8. adım: G ve H düğümlerine D düğümünden ulaşılmaktadır, etiketleme aşağıdaki gibidir.



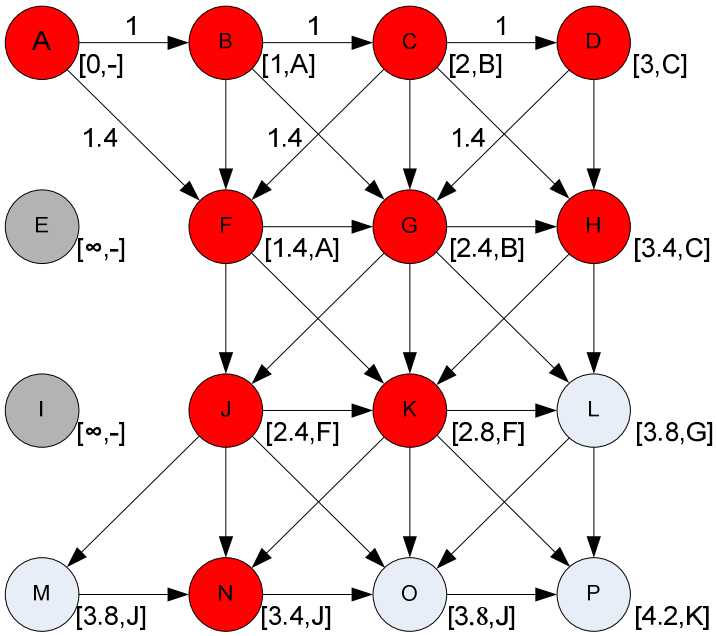
Şekil 3.11. Dijkstra algoritmasının 8. adımında çizge

Çizelge 3.9 Dijkstra algoritması örneği 8. adım

Düğüm	Etiket	Durum
A	[0, -]	kalıcı
B	[1,A]	kalıcı
F	[1.4,A]	kalıcı
C	[2,B]	kalıcı
G	[2.4,B]	kalıcı
J	[2.4,F]	kalıcı
K	[2.8,F]	kalıcı
D	[3,C]	kalıcı
H	[3.4,C]	geçici
L	[3.8,G]	geçici
M	[3.8,J]	geçici
N	[3.4,J]	geçici
O	[3.8,J]	geçici
P	[4.2,K]	geçici

Burada H ve N düğümlerine olan yollar en kısa olduğu için bir sonraki adımda durumları kalıcı olarak değiştirilir.

9-10. adım: Bu adımda aynı etikete sahip H ve N düğümleri de kalıcı düğüm olarak değiştirilir. Bu düğümlerden diğer düğümlere daha kısa bir yol olmadığı için komşu düğümlerin içerikleri de değişmeyecektir.

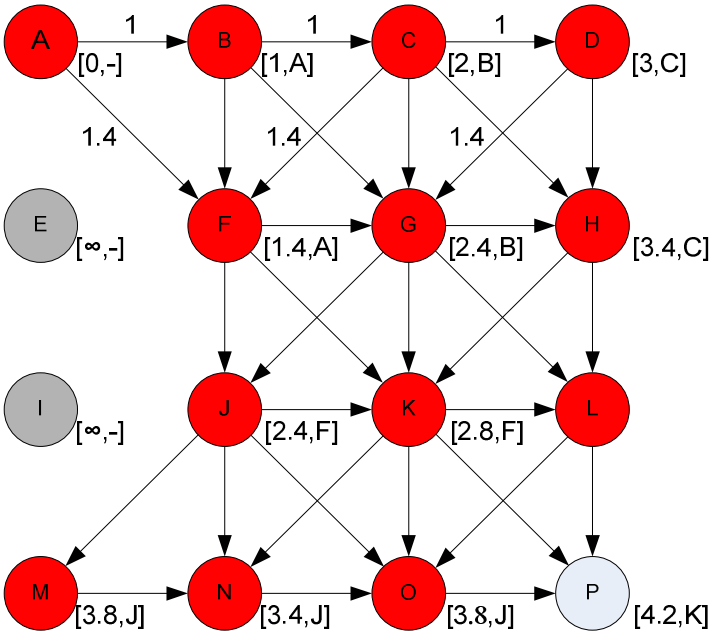


Şekil 3.12. Dijkstra algoritmasının 9. ve 10. adımında çizge

Çizelge 3.10 Dijkstra algoritması örneği 9. ve 10. adım

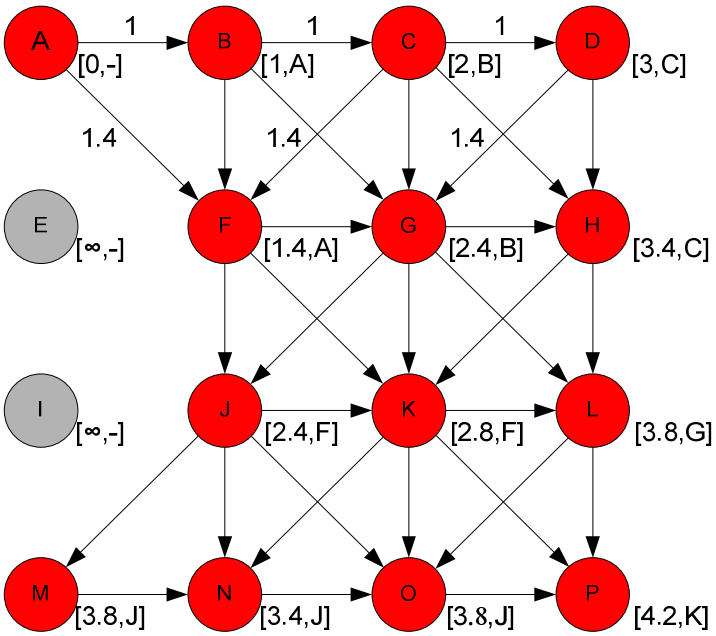
Düğüm	Etiket	Durum
A	[0, -]	kalıcı
B	[1,A]	kalıcı
F	[1.4,A]	kalıcı
C	[2,B]	kalıcı
G	[2.4,B]	kalıcı
J	[2.4,F]	kalıcı
K	[2.8,F]	kalıcı
D	[3,C]	kalıcı
H	[3.4,C]	kalıcı
N	[3.4,J]	kalıcı
L	[3.8,G]	geçici
M	[3.8,J]	geçici
O	[3.8,J]	geçici
P	[4.2,K]	geçici

11-12-13. adım: Bu adımda aynı etikete sahip L, M ve O düğümleri de kalıcı düğüm olarak değiştirilir. Bu düğümlerden diğer düğümlere daha kısa bir yol olmadığı için komşu düğümlerin içerikleri de değişmeyecektir.



Şekil 3.13. Dijkstra algoritmasının 11.-12.-13. adımında çizge

14. adım: P düğümünden diğer düğümlere gidiş olmadığı için etiketleme işlemi sonlandırılır. Son olarak P düğümündeki etiket de kalıcı olarak işaretlenir.



Şekil 3.14. Dijkstra algoritmasının 14. adımında çizge

Çizelge 3.11 Dijkstra algoritması örneği 14. adım

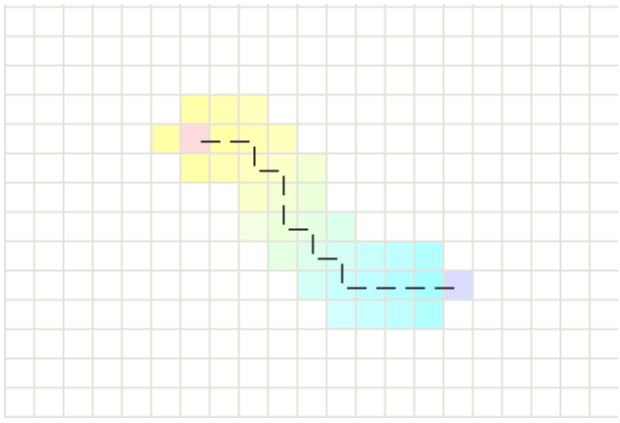
Düğüm	Etiket	Durum
A	[0, -]	kalıcı
B	[1,A]	kalıcı
F	[1.4,A]	kalıcı
C	[2,B]	kalıcı
G	[2.4,B]	kalıcı
J	[2.4,F]	kalıcı
K	[2.8,F]	kalıcı
D	[3,C]	kalıcı
H	[3.4,C]	kalıcı
N	[3.4,J]	kalıcı
L	[3.8,G]	kalıcı
M	[3.8,J]	kalıcı
O	[3.8,J]	kalıcı
P	[4.2,K]	kalıcı

A düğümü ile ağdaki başka bir düğüm arasındaki en kısa yolu bulmak için, istenen varış düğümünden başlanır ve kalıcı etiketler kullanılarak geriye doğru gidilir. Örneğin A düğümden L düğüme giden en kısa yol;

(L) → [3.8,G]→ (G) → [2.4,B]→ (B) → [1,A]→ (A)

Aranan yol A→B→G→L şeklinde olup toplam uzaklık 3.8 birimdir.

3.2. A* (A-yıldız) Algoritması



Şekil 3.15. A* algoritması tarafından gezilen düğümler

Bu algoritma ilk olarak 1968 yılında Peter Hart, Nils Nilsson, ve Bertram Raphael tarafından açıklanmıştır [4, 9]. İlk olarak A algoritması olarak adlandırılsa da, daha sonraları buluşsal (heuristic) optimal yaklaşımdan dolayı A* (A yıldız) olarak adlandırılmıştır.

A* algoritması başlangıç düğümünden bitiş düğümüne olan en az maliyetli yolu bulmada kullanılır [10]. Çizge kuramında bir arama algoritmasıdır.

En kısa yol hesaplanırken; toplam maliyet işlevi $f(x)$, uzaklık işlevi $g(x)$ ile buluşsal (heuristic) işlev $h(x)$ toplanarak elde edilir.

$$f(x)= g(x)+ h(x) \quad (3.1)$$

Yol maliyet işlevi $g(x)$, başlangıç düğümünden üzerinde bulunulan düğüm arasındaki yol uzunluğu ile hesaplanır.

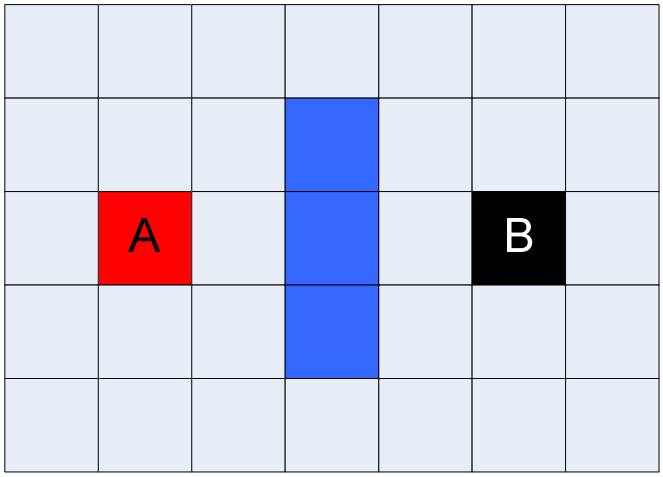
$h(x)$ işlevi buluşsal işlev olarak adlandırılır, bu işlev ise üzerinde bulunulan düğüm ile bitiş düğümüne bakılarak bulunur. $h(x)$ bitiş düğümüne kadar olan düz bir çizgiyi temsil edebilir, fiziksel olarak ise iki nokta arasındaki mümkün olan en kısa mesafedir [23].

Algoritma başlangıç düğümünden başlayarak en düşük f(x) değerli düğüme doğru yayılır. Önce gidilecek düğüm belirlenirken toplam maliyet işlevi

$$f(x) = g(x) + h(x) \quad (3.2)$$

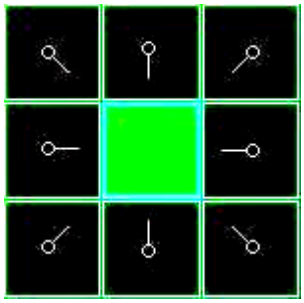
kullanılır [24].

Bir örnek ile açıklamak gerekirse, A noktasından B noktasına gidilmek istenmektedir ve duvar iki noktayı birbirinden ayırmaktadır. Bu durum aşağıda gösterilmektedir ve A noktası kırmızı kare ile B noktası siyah kare ile aradaki duvar ise mavi karelerle gösterilmiştir.



Şekil 3.16. A* (A yıldız) algoritması örnek

Karelerin ortasında düğümler yer almaktadır. En kısa yol bulunduğunda karelerin ortalarındaki bir düğümden diğer düğüme geçilerek en kısa yol takip edilebilir.



Şekil 3.17. Karelerin ortasındaki düğüm

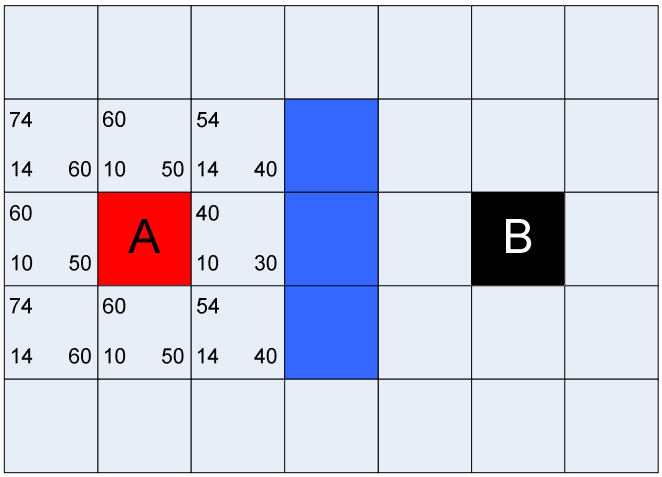
İlk olarak başlangıç düğümünün çevresindeki tüm düğümlere bakılır ve *f*(*x*) işlevi hesaplanır. Burada *g*(*x*) başlangıç noktası olan A dan, üzerinde bulunulan noktaya kadar olan hareket maliyetidir. Sezgisel işlev *h*(*x*) ise daha öncede belirtildiği gibi üzerinde bulunulan düğümden bitiş düğümüne kadar olan hareket maliyetidir.

Bu örnekte bir düğümden diğer düğüme yatay ve dikey geçişlerin maliyeti 10 birim, çapraz geçişlerin ise maliyeti 14 birim olarak atanmıştır. Bu maliyetleri kullanmamızın sebebi çapraz harekette gerçek maliyetin $\sqrt{2}$ kat daha fazla olmasıdır, bu da yatay ve dikey geçiş maliyetlerinin yaklaşık 1.414 katıdır, kolaylık sağlaması açısından *g*(*x*) işlevi hesaplanırken 10 ve 14 maliyetleri kullanılmıştır.

Buluşsal işlev *h*(*x*) ise birçok değişik yol ile hesaplanabilir, bunlardan bir tanesi Manhattan yöntemidir. Bu yöntemde toplam yatay ve dikey düğüm sayıları hesaplanarak uygulanır, çapraz ve engel bulunan düğümler hesaba katılmaz. Bizim örneğimiz için bulunan sonuç 10 ile çarpılır.

$$h(x) = 10(\text{abs}(\text{currentX}-\text{targetX}) + \text{abs}(\text{currentY}-\text{targetY})) \qquad (3.3)$$

Toplam maliyet işlevi *h*(*x*) ile *g*(*x*) i toplayarak bulduktan sonra aşağıda verilen Şekil 3.20 elde edilir. Toplam maliyet işlevi *f(x)* karenin sol üst köşesinde, *g(x)* karenin sol alt köşesinde ve *h(x)* ise karenin sağ alt köşesinde yer almaktadır.

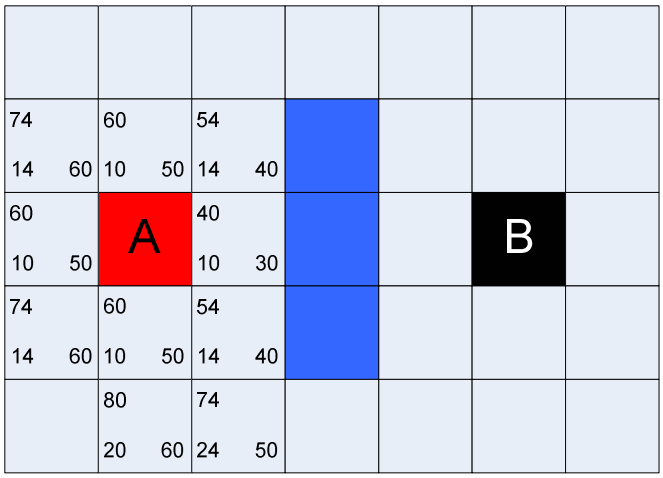


Şekil 3.18. Düğümlerin *g(x)*, *h(x)*, *f(x)* değerleri

Görüldüğü gibi başlangıç düğümüne komşu olan yatay ve dikey düğümlerin *g(x)* maliyet değerleri 10 birim, çapraz komşuların maliyet değerleri ise 14 birimdir.

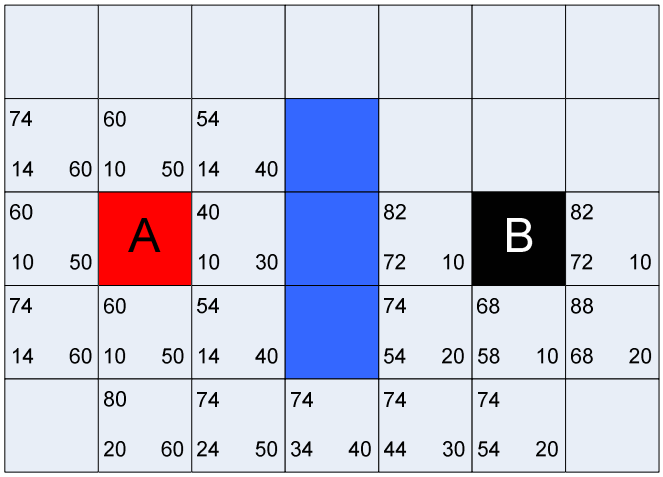
Buluşsal işlev *h(x)*'in değerlerine bakıldığında başlangıç düğümünün hemen sağındaki düğümün bitiş düğümüne 3 birim uzaklıkta olduğu görülmektedir, bu yüzden *h(x)* değeri 30 dur. Aynı şekilde başlangıç düğümünün hemen sağ alt çaprazındaki düğümün bitiş düğümüne 4 birim uzaklıkta olduğu görülmektedir ve *h(x)* değeri 40 dır.

Bu iki işlev toplandığında *f(x)*, toplam maliyet değerimiz bulunmuş olur. *f(x)* değeri en küçük olan düğüm işaretlenir ve işaretlenen düğümün çevresindeki düğümlere bakılır.



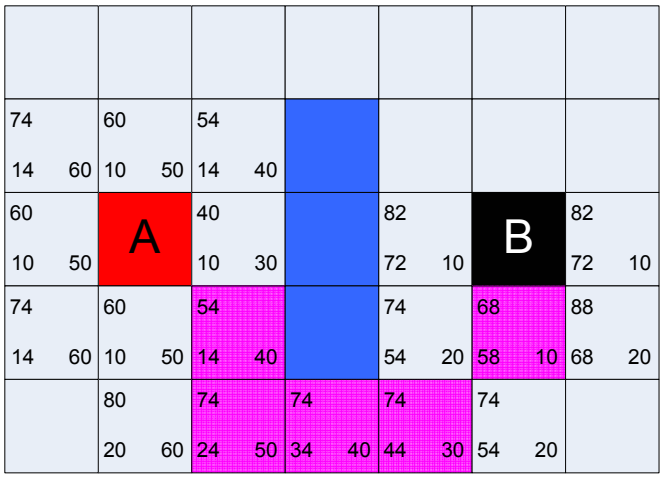
Şekil 3.19. Çevredeki düğümlerin *g(x)*, *h(x)*, *f(x)* değerleri

Eğer *f(x)* değerleri eşit ise *h(x)* değerlerine bakılır ve bunların içerisinden en küçük olanı seçilir.



Şekil 3.20. Başlangıç-hedef arası işaretlenen düğümler

Başlangıç noktasından hedefe doğru, yol üzerindeki düğümler işaretlenerek en kısa yol çıkartılmış olur.



Şekil 3.23. Şekil 3.21. Başlangıç-hedef arası izlenecek yol

3.2.1. A* Algoritmasının Özeti

A* Algoritmasının uygulanışı aşağıda verilmiştir.

1) Başlangıç karesi (ya da düğümü) açık listeye eklenir.

2) Aşağıdaki adımlar yinelenir:

a) Açık listedeki en düşük F maliyetli kareye bakılır. Bu şimdiki (current) kare olarak adlandırılır.

b) Bu kare kapalı listeye geçirilir.

c) Bu karenin komşusu olan 8 kare aşağıdaki durumlara göre değer alır:

- Yasaklı ya da kapalı listedeyse ihmal edilir. Aksi takdirde aşağıdakiler uygulanır.
- Eğer açık listede değilse bu listeye eklenir. Şimdiki kare bu karenin öncüsü kare yapılır. Karenin F, G ve H maliyeti kaydedilir.
- Zaten açık listedeyse, G maliyeti ölçü olarak kullanılarak, bu yolun daha iyi olup olmadığı kontrol edilir. Düşük G maliyeti daha iyi yol demektir. Eğer öyleyse; karenin öncüsü şimdiki kare olarak değiştirilir ve karenin G, F maliyetleri tekrar hesaplanır. Açık liste daha önce F

maliyetine göre sıralı ise, liste sezgisel maliyetlere göre nihai olarak tekrar sıralanır.

d) Aşağıdaki durumlarda durulur:

- Hedef kare kapalı listeye eklendiğinde ve yol bulunduğunda
- Hedef kare bulunamadığında ve açık liste bos kaldığında, bu durumda yol yok demektir.

3) Yol kaydedilir. Hedef kareden, başlangıç karesine tersten gidildiğinde ve her karenin bağlı olduğu kareler sıralandığında yol bulunmuş olur.

4. İKİ AŞAMALI EN KISA YOL BULMA ALGORİTMASI

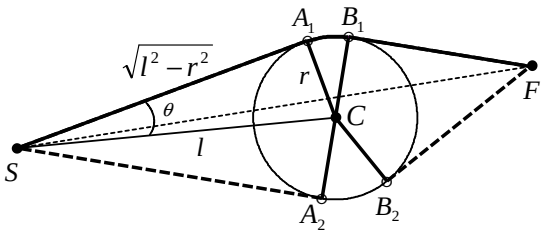
4.1. Birinci Aşama: Geometrik Çözüm Yöntemi

Genel algoritmanın bu aşamasında uygulanan yöntem, bir adım için optimal yöntem olarak nitelendirilebilir. Yöntemin her bir adımı geometrik gösterimlere dayanarak gerçekleştirilir. Her adımda nesnenin şu andaki konumu ile hedef konum arasındaki düz yol üzerindeki ilk engel, tek engel gibi düşünülür. Bu düşünceye uygun olarak önce başlangıç konumdan engele kadar götüren teğet yol belirlenir. Sonra bu yolda ek engellerin bulunup-bulunmadığına bakılır. Eğer bulunmuyorsa (bu durum altbölüm 4.1.1'de incelenir), belirlenen yol üzerinden engele ulaşılır. Bundan sonra engelin çevresi boyunca gidilerek hedef konumdan çizilen teğetin engele dokunduğu noktaya varılır. Son nokta taban alınarak yöntemin yeni adımı başlatılır.

Eğer belirlenen teğet yol üzerinde ek engeller varsa (bu durum altbölüm 4.1.2'de ele alınır), onlardan taban konuma en yakın olanı bulunur. Yukarıda belirlenmiş teğet yol referans alınarak, taban konumdan teğet boyunca bu ek engele ulaşılır ve yay boyunca gidilerek engelden sakınılır. Varılan konum, bir sonraki adım için taban olarak atanır. Bu süreç, hedef konuma götüren düz yolda engel kalmayana kadar devam ettirilir.

4.1.1. Tek Engelden Sakınma

SF yolu üzerinde tek engelin (C merkezli, r yarıçaplı daire) bulunduğunu düşünelim. Bu durum şematik olarak Şekil 4.1.'de gösterilmiştir. S ve F noktalarından çembere ikişer teğet çizilebilir. Bunlardan SF doğrusu ile daha küçük açı oluşturan birer teğet alalım: SA_1 ve FB_1 olsun. Geometrik olarak, SA_1 doğru parçasından, A_1B_1 yayından ve B_1F doğru parçasından oluşan yol, en kısa yoldur.



Şekil 4.1.Tek engelden optimal sakınma

Hesaplamalarda A_1 ve A_2 noktalarının bulunması için aşağıdaki formül kullanılmıştır:

$$\mathbf{SA} = \frac{\sqrt{l^2 - r^2}}{l} P_{\pm\theta}\, \mathbf{SC} \tag{4.1}$$

$$\theta = \arcsin(r / l) \tag{4.2}$$

Burada SA ve SC vektörler; P_θ ise, S noktası etrafında θ radyan döndürme operatörüdür. θ'nın değeri, "+" işaretle alındığında A_1 ve A_2 noktalarından birisi, "-" işaretle alındığında ise, diğeri elde edilir. Bu noktalardan SF doğrusuna daha yakın olanı seçilir.

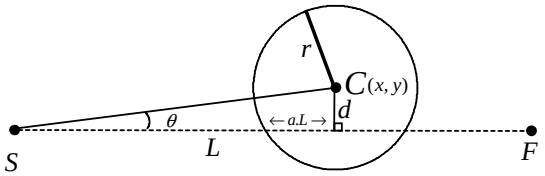
Başlangıç olarak fiziksel ortamı temsil eden dosya okunur. Bu dosyanın içerisinde ortam boyutu (en ve boy), engel sayısı, başlangıç noktasının koordinatları, ulaşılacak noktanın koordinatları, ortamdaki dairesel engellerin merkez koordinatları, dairesel engellerin yarıçapları, engeller arası minimum uzaklık tutulur.

Bir sonraki adımda başlangıç noktası ile ulaşılacak nokta arasında engel olup olmadığına bakılır. Engel var ise program en yakın engeli bulur. En yakın engeli bulmak için tüm engel koordinatları, engel yarıçapları, engel sayıları, başlangıç ve ulaşılacak nokta koordinatları bir alt programa gönderilir. Bu program başlangıç ile ulaşılacak nokta arasına çizilecek doğrunun bir veya birkaç engeli kesip kesmediğine bakar. Bir kaç engeli kesiyorsa başlangıç noktasına en yakın engeli bulur.

Bu aşamadan sonra başlangıç noktasına en yakın dairesel engel başlangıç noktası ile ulaşılacak nokta arasındaki tek engel gibi düşünülür ve tek adım için en kısa yol algoritması uygulanır.

Tek adım için en kısa yolu bulmak için ilk olarak başlangıç noktasından engellere teğetler çizilir ve bu teğetlerin dairesel engeli kestiği noktalar bulunur. Aynı işlem hedef ya da bitiş noktası için de yapılır ve bu noktanın engele teğet olduğu noktalar bulunur. Elimizde altı nokta bulunmaktadır. Başlangıç, bitiş ve teğet noktaları. Bu noktaları birleştirerek engele iki doğru ve bir daire parçasından oluşan iki yol çizilebilir. En kısa yolu bulmak için başlangıç noktasından çekilen teğetlerin engeli kesen iki noktasından başlangıç ve bitiş noktasını birleştiren doğruya olan uzaklıklarına bakılır. Bu uzaklıklardan en küçüğü bize izlememiz gereken yolu verir. Bu adımda amaç yeni başlangıç noktasını belirlemektir. Yeni başlangıç noktası en kısa yolun üzerindeki, ulaşılacak noktadan çekilen teğetlerin engeli kestiği noktadır.

Bu aşamadan sonra algoritma kendini yineleyerek ulaşılması gereken noktaya varıncaya kadar devam eder.



Şekil 4.2. Doğru ve çemberin karşılıklı konumları

Şekil 4.2 de başlangıç noktasından ulaşılacak noktaya çekilen doğrunun ortamdaki bir engeli kestiği görülmektedir. Bu durumu göstermek için ilk olarak başlangıç (x_s, y_s) ve bitiş noktası (x_f, y_f) arasındaki uzaklık bulunur.

$$L = \sqrt{(x_f - x_s)^2 + (y_f - y_s)^2} \tag{4.3}$$

Bir sonraki adımda iki vektör $(\vec{u}, \vec{v})$ arasındaki açı formülü (4.4) kullanılarak başlangıç noktasından, çemberin merkezine (x, y) ve ulaşılacak noktaya çekilen iki doğru arasındaki θ açısı ve daha sonra $\cos\theta$ ile orantılı α değeri (4.6) bulunur.

$$\cos\theta = \frac{\vec{u}.\vec{v}}{\left|\vec{u}\right|\left|\vec{v}\right|} \tag{4.4}$$

$$0 \le \theta \le \frac{\pi}{2} \tag{4.5}$$

$$\alpha = \frac{(x - x_s)(x_f - x_s) + (y - y_s)(y_f - y_s)}{L^2} \tag{4.6}$$

Bu bölümdeki en önemli adımlardan birisi noktanın doğruya olan uzaklık formülünü kullanarak, çemberin merkez noktasının SF doğrusuna olan uzaklığı d (4.8) sayısını bulmaktır. Dairesel engelin merkez noktasının koordinatlarını SF doğru denklemine koyduğumuzda elde edilen sayıyı L uzunluğu ile normalize edersek d değerini elde etmiş oluruz.

$$d = \frac{\left|(x - x_s)(y_f - y_s) - (y - y_s)(x_f - x_s)\right|}{L} \tag{4.7}$$

Elde edilen α değeri 0 ile 1 arasında olacağından, doğrunun engeli kesip kesmediğini bu değeri, d sayısını, L uzunluğunu ve dairesel engelin yarıçapını (r) aşağıdaki denklemlere koyduğumuzda elde edilen sonuçlara bakılarak karar

verilebilir. Sonuçlar δ (4.9) ve β (4.10) olarak adlandırılmıştır. β değeri ile engelin, başlangıç ve bitiş noktalarından geçen dikey doğruların sınırladığı bölgede olup olmadığı kontrol edilir. δ değeri ile engellerin büyüklüğü kontrol edilir. δ ve β'nın değerlerine göre doğrunun engeli kesip kesmediğini anlayabiliriz. Kesim durumunda $\beta \geq 0$, $\delta \leq 1$ ve $\delta > \beta$ koşulları sağlanmalıdır. Aksi taktirde doğru engeli kesmez.

$$a = \frac{\sqrt{r^2 - d^2}}{L} \tag{4.8}$$

$$\delta = \alpha + a \tag{4.9}$$

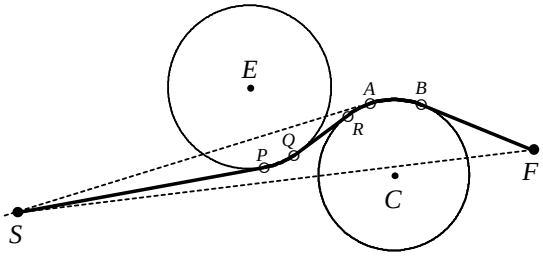
$$\beta = \alpha - a \tag{4.10}$$

4.1.2. Ekstra Engellerden Sakınma

Altbölüm 4.1 de anlatılan SA teğet yolu üzerinde ekstra engeller olabilir. Böyle bir durum ve bu durumda nasıl bir yol izlenebileceği, Şekil 4.2'de şematik olarak gösterilmiştir. Burada C merkezli engel, SF yolu üzerinde bulunan normal engel, E merkezli engel ise, ekstra engeldir. Ekstra engeller üzerine 2 farklı durum olabileceğinin altını çizelim:

1) Başlangıç konum, ekstra engelden belli bir uzaklıktadır,
2) Başlangıç konum, ekstra engel üzerindedir.

Ekstra engellerden sakınma üzerine bu çalışmada gerçekleştirilen algoritmanın açıklaması aşağıda verilmiştir.



Şekil 4.3. Ekstra engelden sakınma

SA teğetinin kestiği ekstra engeller içerisinden S tabanına en yakın olanı (şekilde E engelidir) belirlenir. (Burada ekstra engeller üzerine yukarıda belirtilen 1.durum söz konusudur). SA yönü referans alınarak bu engelden sakınmak düşünülür. Buna göre SP teğeti belirlenir ve onun üzerine P noktasına gidilir. P noktası, yeni başlangıç konum olarak atanır ve yöntemin yeni adımı başlatılır. Yeni adımda PF yolu üzerindeki C engelinden sakınmak istendiğinde, E engeli, ekstra engel olarak tekrar karşıya çıkıyor. (Burada engeller üzerine 2.durum gerçekleşmiş olur). Engellerin ortak teğeti QR belirlenir. PQ çember yayı üzerine Q noktasına varılır. Böylelikle, önce SP teğeti, sonra ise, PQ çember yayı üzerinden gidilerek, E ekstra engelinden sakınılmış olur. Q, yeni başlangıç konum atanarak yöntemin yeni adımı başlatılır.

Not: Bu bölümde tartışılan yöntem üzerine elde edilen yol, doğru$_S$, yay$_1$, doğru$_{12}$, yay$_2$, ... , doğru$_F$ şeklinde olacaktır. Ortalarda yer alan doğrular yerine ilgili çemberlerin uygun ortak teğetleri alınırsa (eğer bu teğet hiç bir ekstra engeli kesmiyorsa), daha kısa bir yol elde edilebilir.

4.2. İkinci Aşama: Dijkstra Algoritması ile Optimal Yolun Bulunması

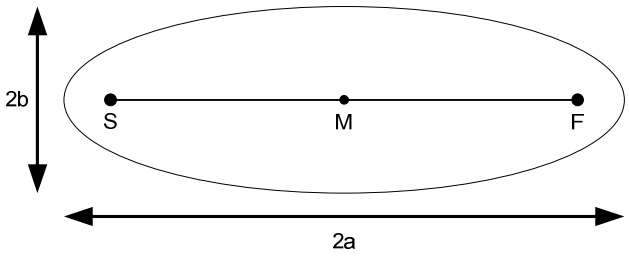
Genel algoritmanın birinci aşamasından elde edilen yol optimal olmayabilir ama bu yolun uzunluğu L_1^* , optimal yolun uzunluğu L^* için bir üst sınır oluşturmaktadır: $L^* \le L_1^*$. Bu eşitsizliğe dayanarak optimal yolun yer aldığı bölge (ileride “olurlu bölge” (feasible region) olarak adlandırılacaktır) daraltılabilir.

Optimal yol üzerindeki bir X noktasını düşünelim. $L^* = L^*_{SF} = L^*_{SX} + L^*_{XF}$ dir. Doğru parçası boyunca yol en kısa olduğundan $|SX| \le L^*_{SX}$ ve $|XF| \le L^*_{XF}$ dir. Buradan ve yukarıda anlatılanlardan $|SX| + |XF| \le L^*_1$ olduğu görülebilir. Son eşitsizliğe göre, olurlu bölgeye giren bir X noktasının, S ve F noktalarından uzaklıkları toplamı, L^*_1 değerini aşmamalıdır. Bu durum, olurlu bölgenin, odakları S ve F noktalarında olan elipsin içerisinde yer aldığını göstermektedir. Böylece, L^*_1 değeri esas alınarak olurlu bölge bir elipsle sınırlandırılarak küçültülebilir. Elde edilen bölge, ilerideki işlemlerin tasarruflu yapılmasını sağlıyor.

Elips, verilen iki noktaya (S, F) uzaklıkları toplamı sabit olan noktaların geometrik yeridir. Verilen bu iki noktaya elipsin odakları denir. Elips, şekli ve geometrik özellikleriyle çok eski yıllarda bile bilinmekteydi. Denklemi $x^2+y^2=a^2$ olarak verilen bir dairenin koordinatlarının b/a oranında büyütülmesiyle veya küçültülmesiyle elipsin denklemi,

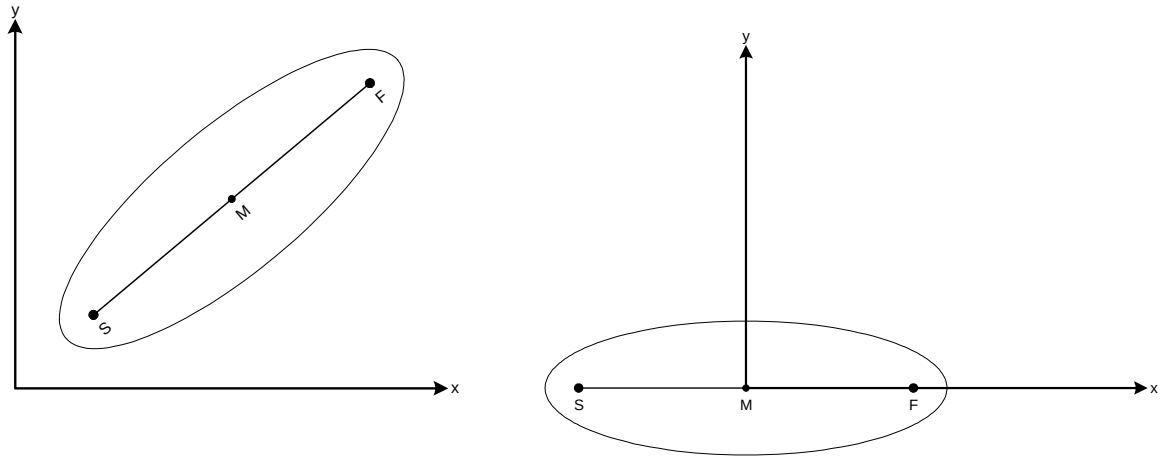
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \qquad (4.11)$$

olarak bulunur.



Şekil 4.4. Elipsin eksenleri

Elipsin 2a büyüklüğünde büyük (büyük ekseni) ve 2b büyüklüğünde küçük ekseni vardır bkz Şekil 4.4.



Şekil 4.5. Yeni koordinat sistemine geçiş

Genel algoritmanın ikinci aşamasında önce başlangıcı SF doğru parçasının M orta noktasında, x ekseni ise MF doğrultusunda olan koordinat sistemine geçiş yapılır. Bu koordinat sisteminde olurlu bölge sade bir şekilde ifade edilir:

$$\left(\frac{x}{a}\right)^2+\left(\frac{y}{b}\right)^2 \le 1 \tag{4.12}$$

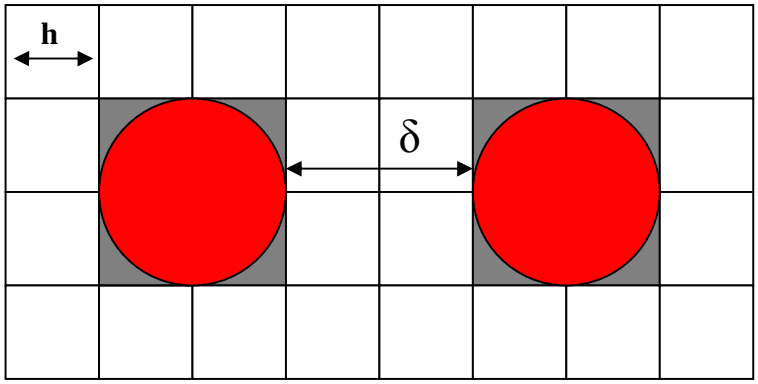
Burada

$$a = L_1^* / 2 \tag{4.13}$$

$$b = \sqrt{L_1^{*2} - \left|SF\right|^2} / 2 \tag{4.14}$$

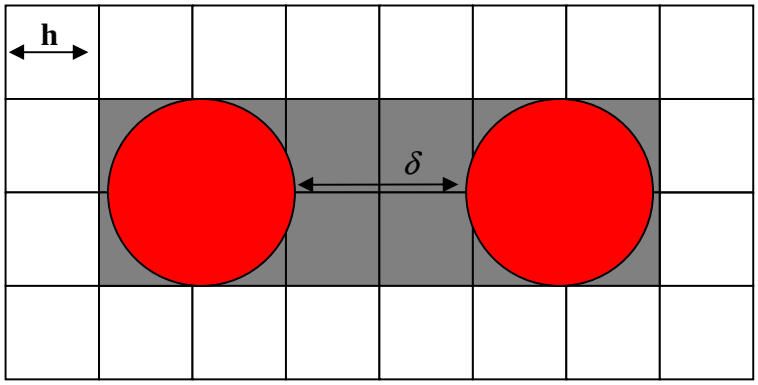
Yeni koordinat sistemi bir sonraki algoritmanın daha verimli şekilde gerçekleştirilmesinde yararlı olmaktadır, bkz Şekil 4.5.

Bir sonraki adımda Dijkstra algoritmasının uygulanabilmesi için problemin ayrıklaştırılması (discretization) yapılır. Bunun için bölge üzerinde eşit karelerden oluşan bir ızgara (grid) oluşturulur. Karelerin kenar uzunluğu *h*, engeller arasındaki minimal mesafe δ ile uyumlu şekilde alınır: $h \le \delta / 3$.



Şekil 4.6. δ mesafesinin iki kare uzunluğuna eşit olması ve yasaklı bölgeler

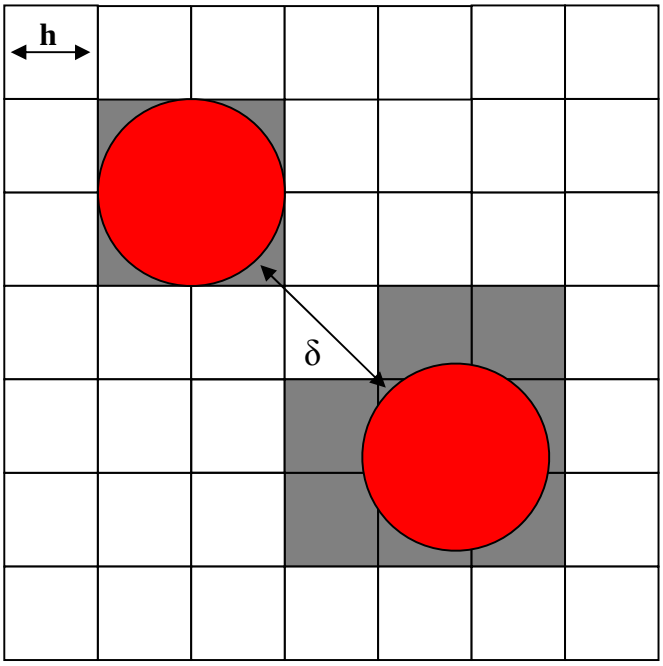
Engeller arası minimum uzaklık olan δ mesafesinin en az iki kare ile bölünmesi ve yasaklı bölgelerin oluşturulması yukarıdaki şekilde gösterilmiştir, engellerin yer değiştirmesi ve aradaki mesafenin δ<2h durumunda yasaklı bölgelerin konumu da değişecektir ve bu durum aşağıdaki şekilde gösterilmiştir. Aşağıdaki şekle bakılarak iki engel arasında bir geçiş yapılamayacağı görülmektedir ve buradan yola çıkılarak δ>2h sonucuna varılır. Bununla birlikte δ≥3h olması gerektiğinin geometrik açıklaması aşağıda yapılmıştır.



Şekil 4.7. δ mesafesinin iki kare uzunluğundan az olması

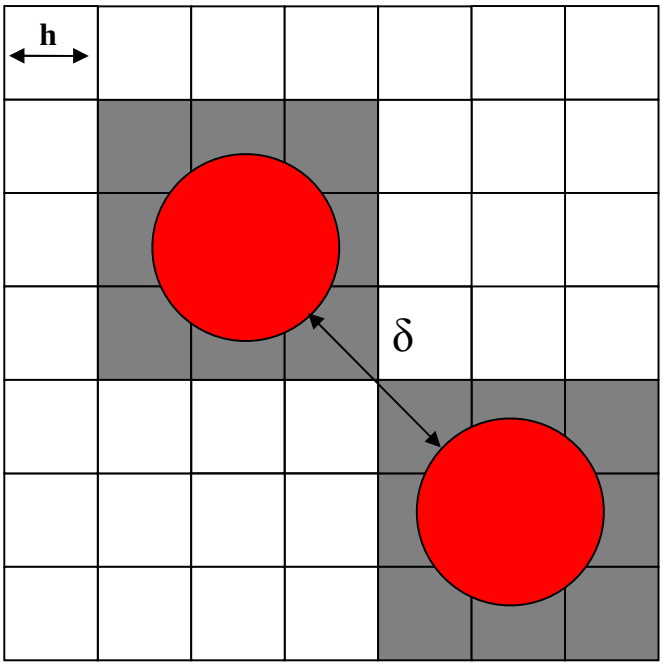
Fakat iki engel 45 derecelik açıyla yerleştirildiğinde ve aralarındaki δ uzaklığı iki kare ile bölündüğünde Şekil 4.8 de bir geçişin olduğu fakat Şekil 4.9 da bir geçişin olmadığı görülmüştür. Bu durumda engeller arası minimum mesafe

$\delta > 2\sqrt{2}$ h olmalıdır. Bu mesafe çizgedeki karelerin kenar uzunluğu cinsinden ifade edilmek istenirse $\delta \geq 3h$ olmalı yani engeller arası minimum uzaklık δ en az 3 kare olmalıdır.



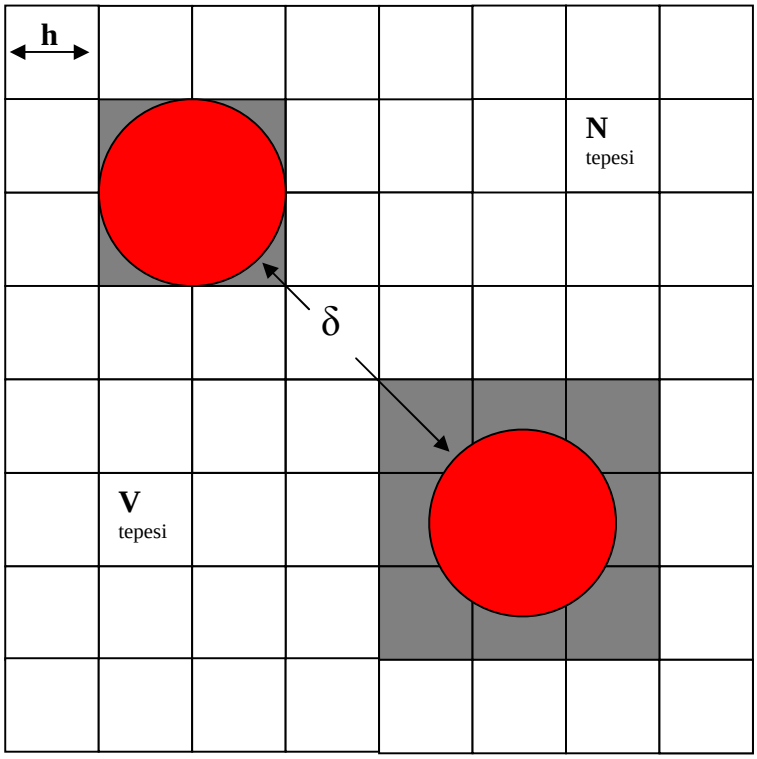
Şekil 4.8. δ mesafesinin iki kare olması ve Engellerin çapraz konumu

Aşağıdaki şekilde görüldüğü gibi herhangi bir engelin yer değiştirmesi durumunda engeller arasından geçmek mümkün değildir.



Şekil 4.9. δ mesafesinin iki kare olması ve Engellerin çapraz konum değiştirmesi

Herhangi bir engelin yer değiştirmesi halinde, δ mesafesinin 3h olduğu durum aşağıdaki şekilde gösterilmiştir. Engeller birbirlerine göre 45 derece açı ile yerleştirilmelerine rağmen hala arada bir geçişin mümkün olduğu gözlenmektedir.



Şekil 4.10. δ mesafesinin Üç kare olması ve Engellerin çapraz konumu

Çizgenin düğüm noktaları, tepe noktaları olarak adlandırılarak, incelenen problem, bir çizge (graph) problemine dönüştürülür. a) Eğer tepe noktası N olurlu bölgenin dışarısında yer alıyorsa yada b) eğer merkezi N'de ve kenar uzunluğu h olan karenin, bir engel ile kesişimi varsa, bu tepeye geçişler yasaklanır.

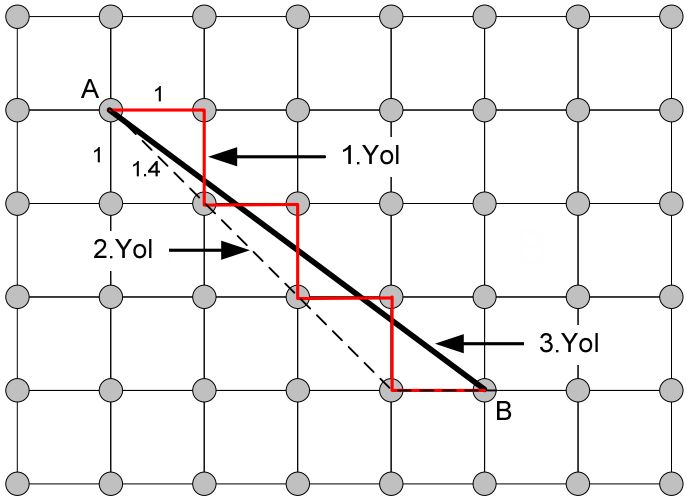
1) Herhangi bir yasaklanmamış V tepesi için onun yakın etrafında bulunan 8 tepe teker-teker alınarak incelenir. Eğer incelenen tepeye geçişler yasak değilse, V tepesini bu tepeyle bağlayan kenar eklenir.

2) Tüm (U, V) tepe çiftleri teker-teker alınarak incelenir. Eğer U ve V, yasak olmayan tepeler ise ve UV doğru parçası hiç bir engeli kesmiyorsa, bu tepeler arasında uzunluğu |UV| olan kenar doğrudan oluşturulur.

Kenarların oluşturulması için 2 farklı yöntem izlenebilir.

Yöntem 1: Kenarların oluşturulması sırasında da ayrık yaklaşım kullanılmaktadır. Bundan dolayı kenar yapısı sadedir.

Yöntem 2: Açgözlü (greedy) bir yaklaşımı yansıtmaktadır, kenar yapısının zorlaşmasına neden olmaktadır, ancak optimal çözüme daha iyi yaklaşım elde edilmesini sağlamaktadır. Benzetim çalışmalarında 2. yöntem izlenmiştir.



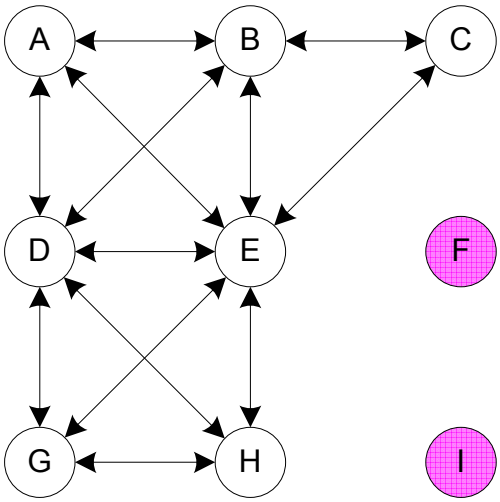
Şekil 4.11. Ayrık yaklaşım - Açgözlü (greedy) bir yaklaşım

Şekil 4.11' de A düğümünden B düğümüne gidilmek istenmektedir. Bir düğümden komşu düğüme olan kenarlar, yatay, düşey ve çapraz olmak üzere sekiz adettir. Aşağıdaki şekilde A noktasından B noktasına gitmek için üç yol izlenmiştir. Birinci yolda çapraz kenarlar kullanılmamış sadece yatay ve düşey kenarlar kullanılmış ve yolun uzunluğu 7 birim tutmuştur. İkinci yolda, yukarıda bahsi geçen ayrık yaklaşım kullanılmıştır ve yol maliyeti 5.2 birimdir. Üçüncü yolda ise açgözlü (greedy) bir yaklaşım uygulanmış ve çizgeye yeni bir kenar eklenmiştir. Yeni kenarın uzunluğu 5 birimdir.

Şekil 4.8' de görüldüğü gibi çizgeye Açgözlü (greedy) yaklaşım uygulandığında yol kısalmakta fakat yapıya yeni kenarlar eklendiği için çizge yapısı karmaşıklaşmaktadır.

Böylece, araştırılan problemin çözümü, elde edilen çizgede S tepesinden F tepesine giden en kısa yolun bulunmasına getirilir. Yeni problem, Dijkstra algoritmasının [4, 5] uygulanması ile çözülür. Bu sırada problemin özelliklerine dayanarak Dijkstra algoritmasının daha verimli kullanımını sağlayan bazı değiştirmeler yapılır. Örneğin, S'den yasak tepelere kadar olan uzaklıklar $-\infty$, diğer tepelere kadar uzaklıklar ise $+\infty$ olarak alınırsa (burada "-" işareti uzaklık değerinin kalıcı olduğunu, "+" işareti ise uzaklık değerinin geçici olduğunu göstermek için kullanılmıştır), çizgenin komşuluk matrisine gerek kalmıyor.

Şekil 4.12'deki örnek çizge için komşuluk matrisi aşağıda verilmiştir. Şekilde 9 düğüm vardır, F ve I yasaklı düğümler olduğu için bu düğümlere ulaşım yoktur fakat diğer düğümlerin birbirleri arasındaki çift yönlü ulaşımlar Şekilde gösterilmiştir. Yukarıda da belirtildiği gibi, 9 düğümlü bir çizgenin komşuluk matrisi 9x9 büyüklüğünde olduğu düşünülürse, komşuluk matrisinin kullanılmaması büyük derecede işlem kolaylığı getirmektedir.



Şekil 4.12. Örnek Çizge

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
A	0	1	0	1	1	0	0	0	0
B	1	0	1	1	1	0	0	0	0
C	0	1	0	0	1	0	0	0	0
D	1	1	0	0	1	0	1	1	0
E	1	1	1	1	0	0	1	1	0
F	0	0	0	0	0	0	0	0	0
G	0	0	0	1	1	0	0	1	0
H	0	0	0	1	1	0	1	0	0
I	0	0	0	0	0	0	0	0	0

Şekil 4.13. Komşuluk Matrisi

5. ROBOT SİSTEMİ VE MODELLER

5.1. ActivMedia Robotlar

Deneysel çalışmalarda kullanılan ActivMedia robotun önemli elemanları sırasıyla; dengeli sürüş sisteminden(ör: diferansiyel iki tekerlek ve bir mobilya tekerleği), tersine dönebilen DA motorlar, motor sürücü ve denetim elemanları, yüksek çözünürlüklü hareket kodlayıcılardan oluşmaktadır. Bütün bu elemanlar, robotların üzerlerinde bulunan işlemci ve gezgin robot gömülü yazılımlarıyla yönetilirler. Bu robotlar iki tekerli ve dört tekerli yapılara sahiptirler.

Şekil 5.1 1995 yılında üretilen Pioneer1 robotu

ActivMedia robotların çeşitli ortamlarda özerk hareketlerinin denetlenebilmesi için geliştirilmiş ARIA (Advanced Robotics Interface for Applications) adlı gömülü yazılıma sahiptir. Bu yazılım C++ tabanlı açık-kaynak geliştirilmeli bir yazılımdır. ARIA yazılımı sayesinde ActivMedia robotların denetimi ve yedek parça sistemlerindeki değişimlere göre uyum sağlayacak bir iletişim arayüzü sağlanmıştır.

Tez çalışmasında kullanılan Pioneer 3-DX modelinin kullanılan özellikleri:

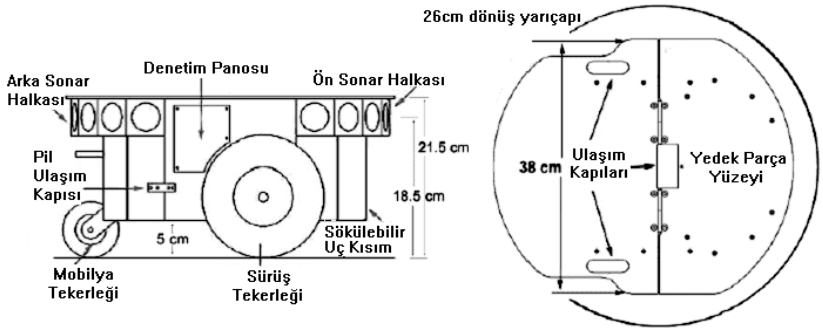
- 4 tane RS232 seri port, 9.6 ile 115.2 kilobaud arasında ayarlanabilir,
- 4 tane Sonar dizisi, her birinde en fazla 8 sonar
- Yüksek çözünürlüklü hareket kodlayıcıları
- Kullanıcı denetim paneli

- Ana güç ve güç seviyesi gösterici iki renkli LED
- RESET ve MOTORS basma düğmeleri
- Piezo buzzer

Pioneer 3-DX robotunun daha ayrıntılı özellikleri EK 1'de verilmiştir.

5.1.1. Dış Yapısı

Daha yaygın olan robotlara oranla Pioneer 3-DX'in makul boyutu, dar ve karmaşık alanlarda gezinim yapmasına imkan sağlamaktadır.



Şekil 5.2 Pioneer 3-DX'in fiziksel boyutu ve dönüş yarıçapı

Şekil 5.2'de gösterildiği gibi, Pioneer 3-DX'in dönüş yarıçapı, yani kendi etrafında döndüğü sırada kullandığı serbest uzayın iki boyuttaki yarıçapı 26 cm'dir. Sonar algılayıcıların merkezlerinin yerden yüksekliği 18.5 cm robotun üzerinde yük olmadığında yüksekliği de 21.5 cm'dir.

5.1.2. Motorlar, Tekerlekler ve Pozisyon Kodlayıcıları

Pioneer 3-DX yüksek hızlı, yüksek torklu, çift yönlü DA motorlara sahiptir. Pioneer 3-DX robotlar üzerinde dolma teker kullanılmaktadır. Bu tekerleklerin çapları 19.53 cm'dir. Tekerleklerin aşınması sonucu bu değerde değişiklik olabilir. Motorların şaftı üzerine yerleştirilmiş yüksek çözünürlüklü kodlayıcılar sayesinde

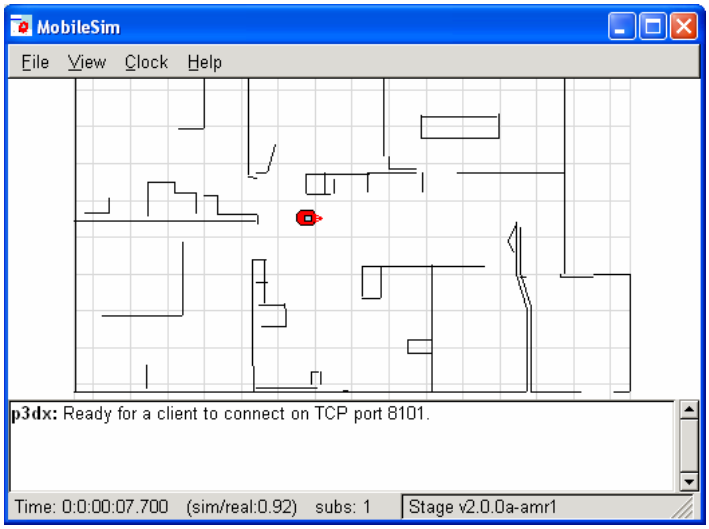
robotun hız, yön ve aldığı mesafe bilgileri elde edilebilmektedir. Kullanılan kodlayıcılar bir dönüşte 500 adet vurum üretebilme özelliğine sahiptirler. Pioneer-3DX robotunda kullanılan kodlayıcı ve hareket elemanları için bazı değerler çizelge 5.1'de verilmiştir.

Çizelge 5.1 Robot yapısına bağlı parametreler

Değerler	
PARAMETRELER	**P3-DX**
kodlayıcı vurum/dönüm	500
dişli oranı	38.3:1
tekerlek çapı (mm)	195.3
kodlayıcı vurum/mm	128

5.2. Robot Benzetim Ortamı

Robot hareketini sağlayan programın testi öncelikle benzetim ortamında yapılmıştır. Robotun benzetimi, ActivMedia robotlar için tasarlanan, *MobileSim* adlı program aracılığı ile yapılmıştır. Ayrıca robotun konumunun izlenmesi işlemlerinde de bu benzetimciden yararlanılmıştır. Benzetim programın bir görüntüsü şekilde gösterilmiştir.



Şekil 5.3 MobileSim programının görünümü

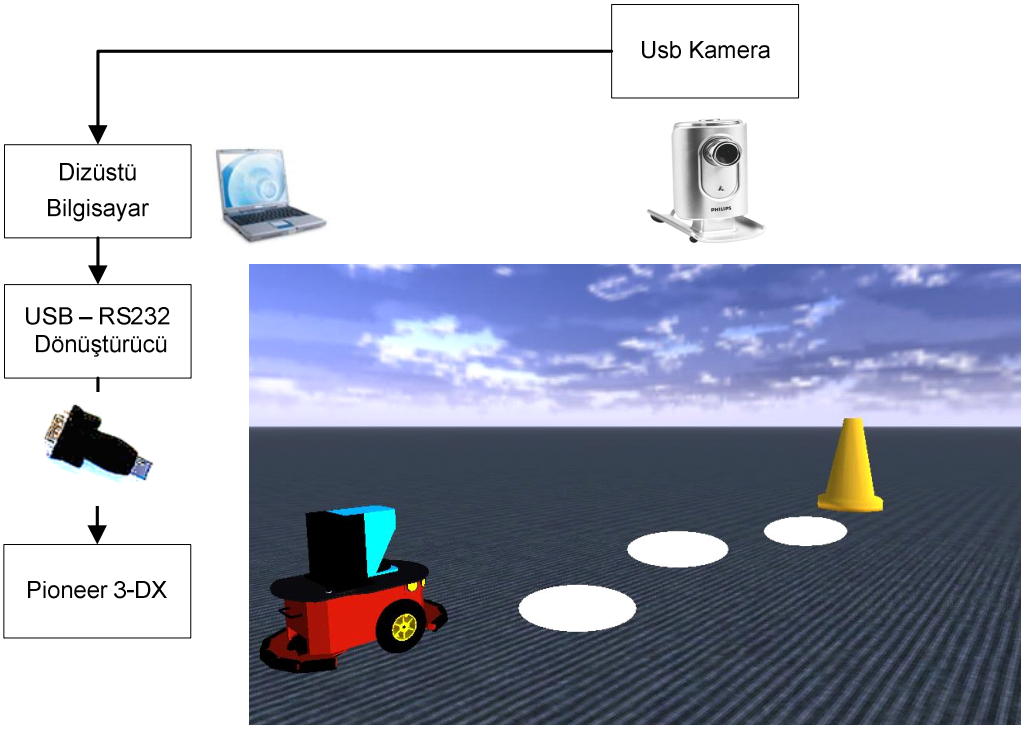
Bu program sayesinde, robot hareketi için yazılan programların robota aktarılmadan sanal ortamda çalışırlılığının testi gerçekleştirilebilmektedir. Program, ortamların 2 boyutlu olarak tanımlanmasına olanak sağlamaktadır. Dolayısıyla ortamda belirtilen her engelin yüksekliğinin robotun yüksekliğinden büyük olduğu kabul edilmektedir.

6. OPTİMAL EN KISA YOLUN BULUNMASI

6.1. Sistem Tasarımı

Tez çalışmasında *ActivMedia Robotics* firması tarafından üretilmiş Pioneer 3-DX robotu kullanılmıştır. Robotun en kısa yolu bulması işlemi tez çalışmasında iki şekilde gerçekleştirilmiştir:

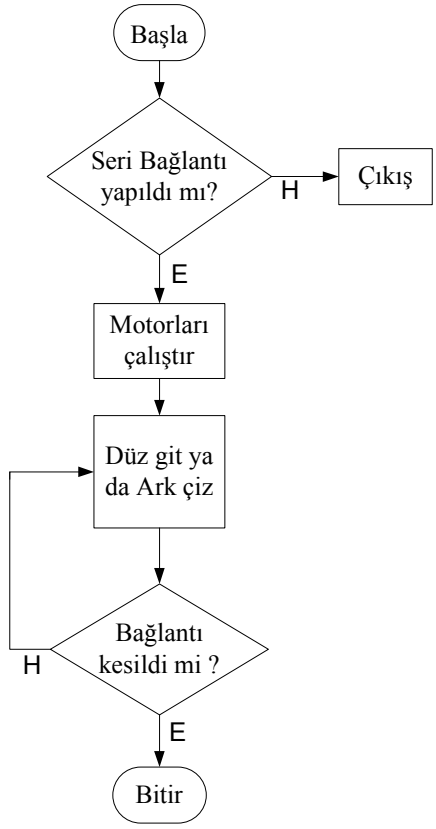
- Benzetim ortamında
- Gerçek ortamda



Şekil 6.1 Sistemin ana şeması

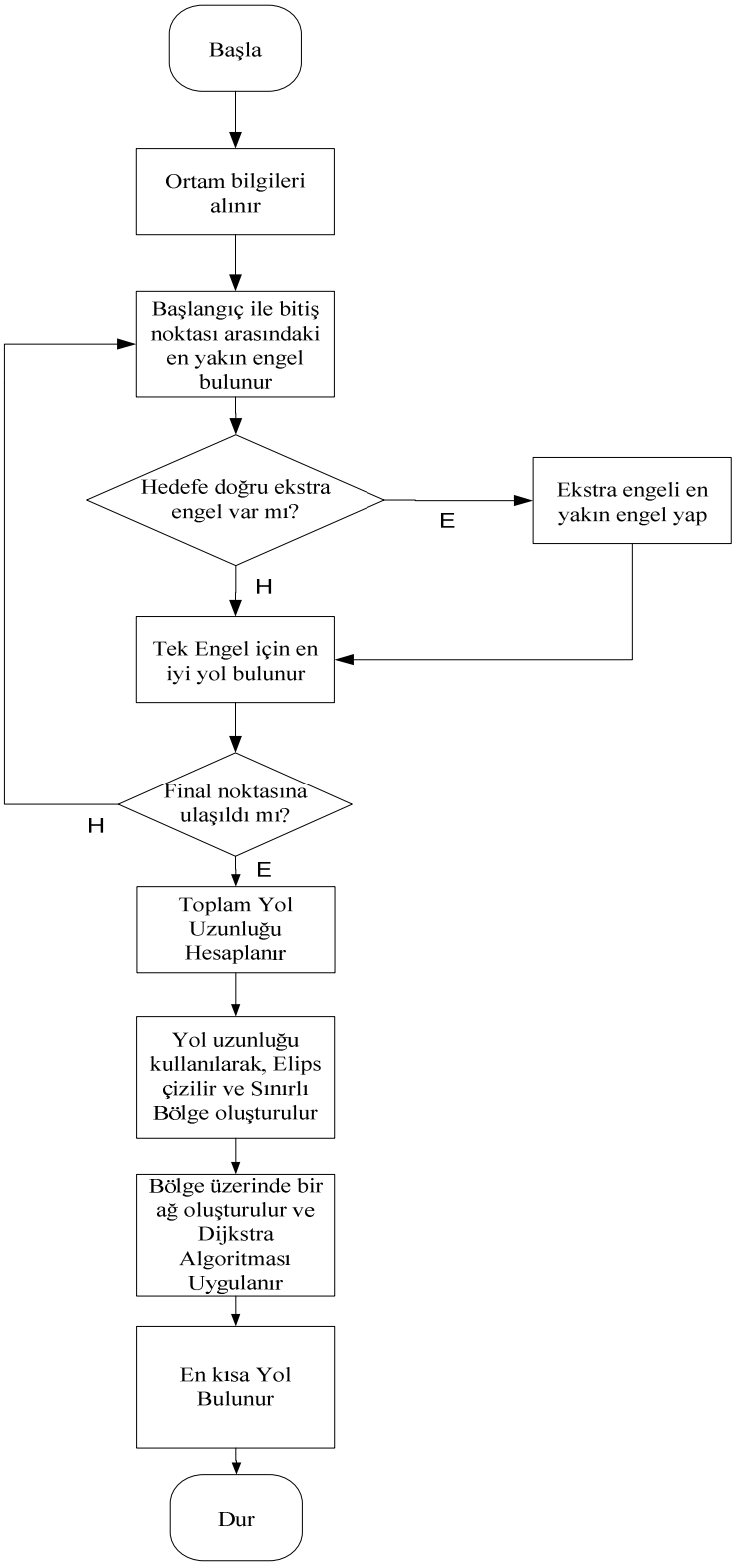
Fiziksel ortamda robot hareketlerinin gerçekleştirilebilmesi için robot üzerine yerleştirilmiş bir dizüstü bilgisayar kullanılmıştır. Robotun hareketleri ve algılayıcılarla elde edilecek veriler için C++ tabanlı program yazılmıştır. Yazılan program sayesinde robotun ortam içerisinde engellere çarpmadan hareketi

sağlanmıştır. Aynı zamanda bu program benzetim ortamında da kullanılmıştır. Bu programın basit akış diyagramı şekil 6.1'deki gibidir.



Şekil 6.2 C++ tabanlı programın akış diyagramı

İki aşamalı algoritmanın uygulanması için MATLAB programından yararlanılmıştır ve akış diyagramı şekil 6.3'deki gibidir.

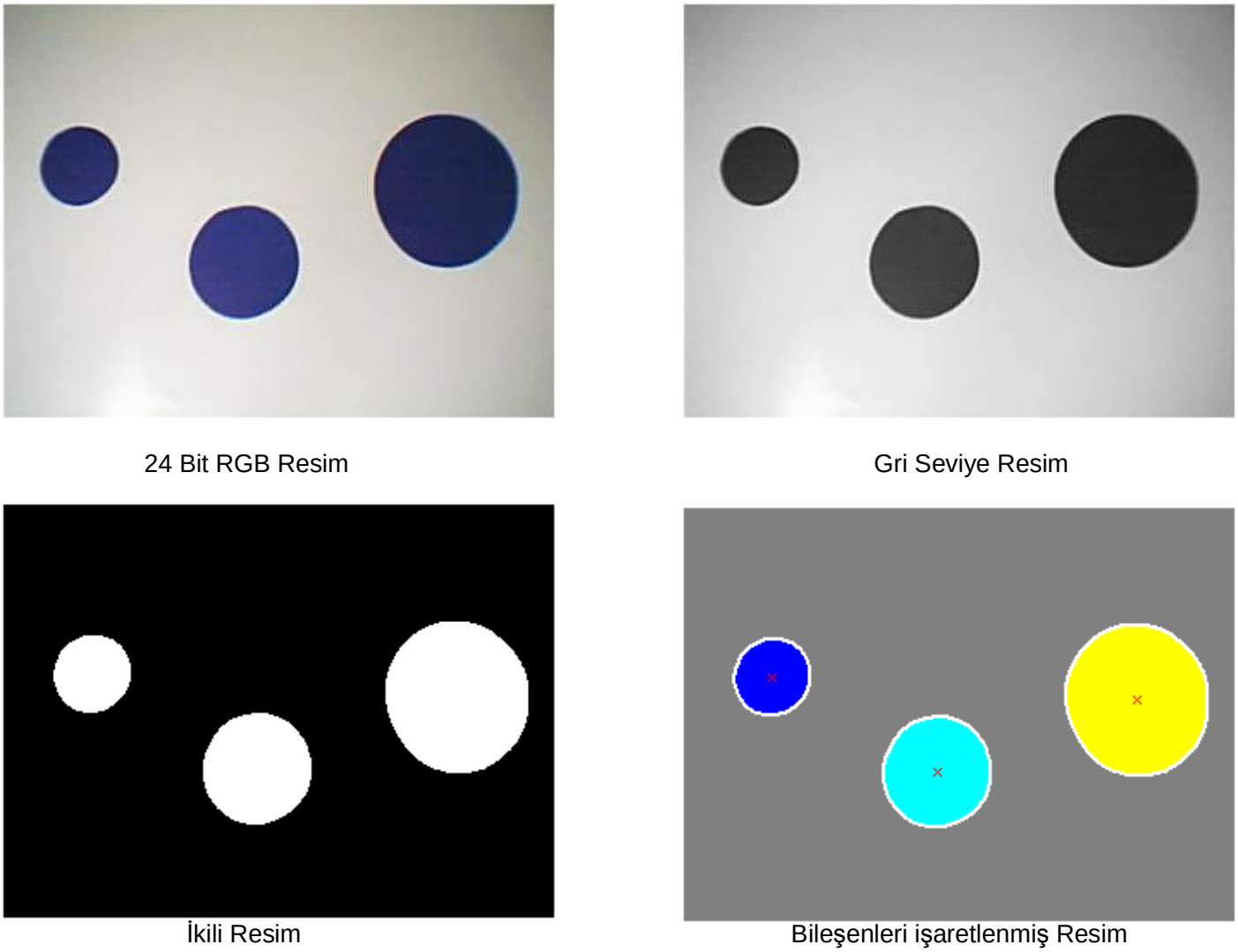


Şekil 6.3 İki aşamalı algoritmanın akış diyagramı

6.2 Görüntü İşleme

Bu çalışmada görüntü işleme teknikleri görüntünün gri seviye resme dönüştürülmesi, ikili resmin elde edilmesi, dairesel engellerin sayılarının, merkez koordinatlarının (x,y), yarıçaplarının belirlenmesi, görüntüdeki nesnelerin ve zeminin birbirinden ayrıştırılması için kullanılmıştır [14].

İmgeye görüntü işleme teknikleri sonucunda elde edilen imgeler aşağıda Şekil 6.3 ’de gösterilmiştir.



24 Bit RGB Resim

Gri Seviye Resim

İkili Resim

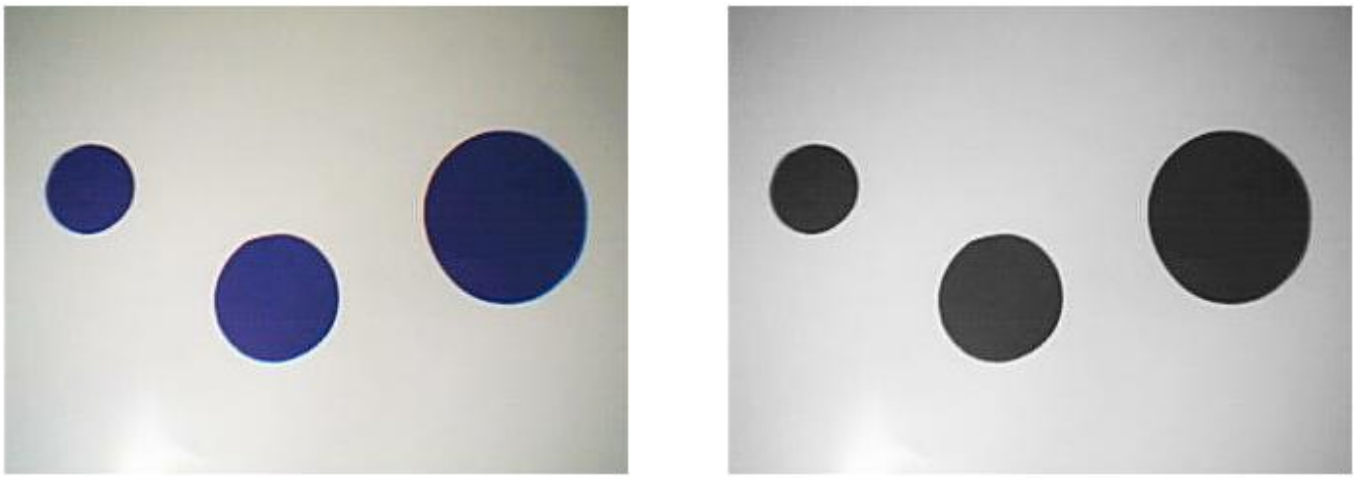
Bileşenleri işaretlenmiş Resim

Şekil 6.4. Uygulanan görüntü işleme teknikleri ile elde edilen imgeler

6.1.1 Gri Seviye Resim

Resim piksellerindeki RGB (Red-Green-Blue) ile gösterilen Kırmızı, Yeşil, Mavi renk kanallarının Eşitlik (6.1) verilen katsayılarla çarpımının toplamı o pikselin gri-seviye karşılığını verir. Bu işlem resimdeki tüm piksellere uygulanarak renkli resim gri seviyeye dönüştürülür [15]. Şekil 6.4'de gerçek resime gri seviye dönüşüm uygulanarak gri seviye resim elde edilmiştir.

$$Y=0.299R+0.587G+0.114B \qquad (6.1)$$



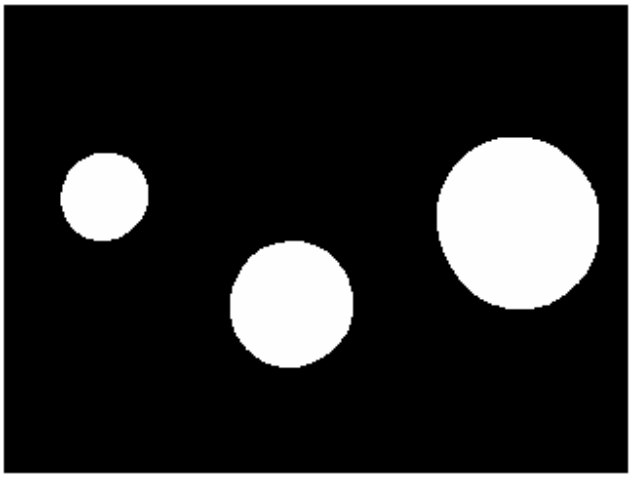
(a) (b)

Şekil 6.5 (a) Gerçek resim- (b) Gri seviye resim

6.1.2. Eşik Seçimi

Gri seviyeye dönüştürülen resimdeki renk değerleri 0 ile 255 arasında değişir. Eşik değeri için 0 ile 255 arasında bir değer seçilir. Resimdeki piksellerin gri seviye değeri eşik değerinden küçük veya eşitse pikselin yeni değeri 0, büyükse 1 (255) olacak şekilde değiştirilir ve iki renkli (siyah-beyaz) bir resim elde edilir bu ifade Eşitlik (6.2)' de belirtilmiştir. Bu işlem ile resim ikili resme dönüştürülmüş olur. Dönüştürülen resimde beyaz olarak görülen bölgeler nesneyi, siyah olarak görülen bölgeler ise zemini temsil eder (Sekil 6.6).

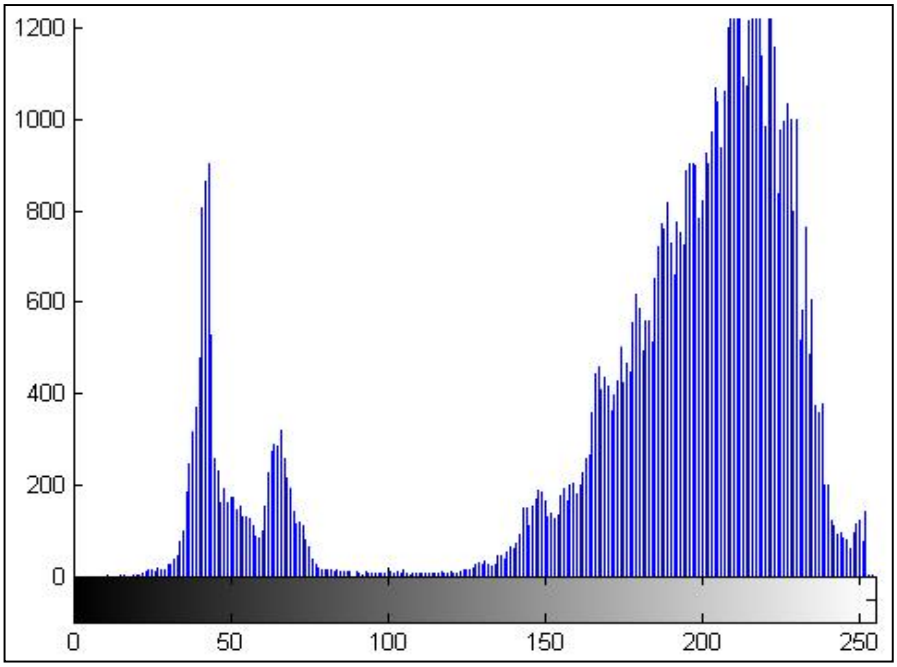
$$g(x,y)=\begin{cases}1 & f(x,y)\geq T\\ 0 & \text{aksi takdirde}\end{cases} \tag{6.2}$$



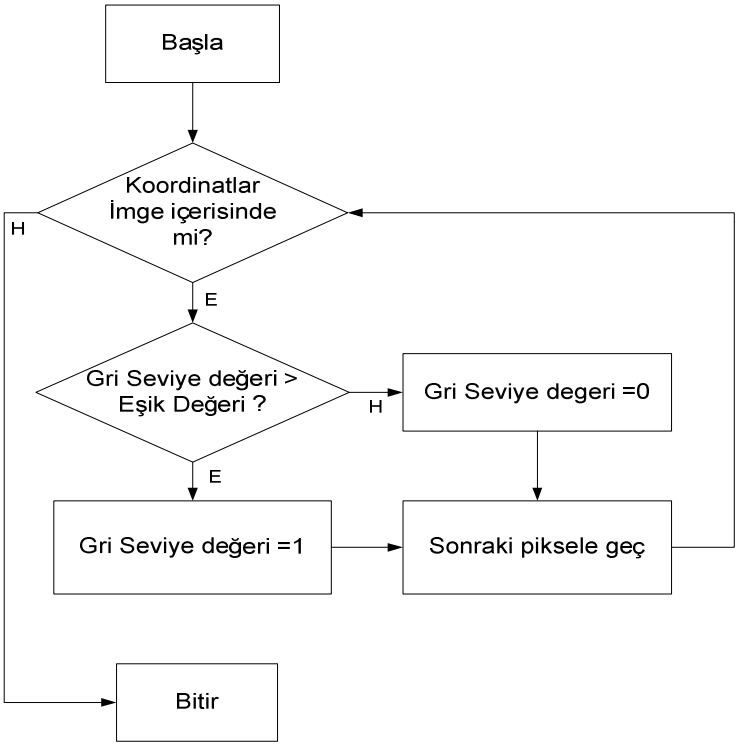
Şekil 6.6. İkili resim

g(x,y)= ikili görüntü değeri, f(x,y)= gri seviye, T = Eşik değeri (threshold)

Şekil 6.6'deki gri seviye resimde koyu bir zemin üzerinde parlak bir nesnenin histogramı görülmektedir. Bu grafik incelendiğinde piksellerin dağılımı iki gruba ayrılmıştır. Gri seviye bir resimde piksellerin renk değerleri (0-255) aralığında değiştiği için, histogramdaki pikseller (0-255) arasında dağılmıştır. Eğer eşik değeri olarak T değerini seçersek ve bu eşik değerinden küçük gri seviye değerine sahip pikselleri 0, T eşik değerinden büyük ve eşit olanları ise 1 (255) ile değiştirilerek iki renkli (siyah- beyaz) bir resim elde edilir. Bu resimde beyaz olarak görülen bölgeler nesneyi, siyah olarak görülen bölgeler ise zemini temsil eder (şekil 6.5).



Şekil 6.7 Gri seviye resmin histogramı



Şekil 6.8 Eşikleme algoritmasının akış diyagramı

Basit eşikleme algoritmasının sözde kodu aşağıdadır:

1) Eşik değerini seç, x;
2) Görüntünün başlangıç ve bitiş koordinatlarını bul, $(x_0, y_0), (x_n, y_n)$;
3) Başlangıç koordinatından bitiş koordinatına kadar piksellerin gri seviye değerine bak;
4) Gri seviye değeri eşik değerinden küçükse yeni değeri 0, büyükse 1 yap;

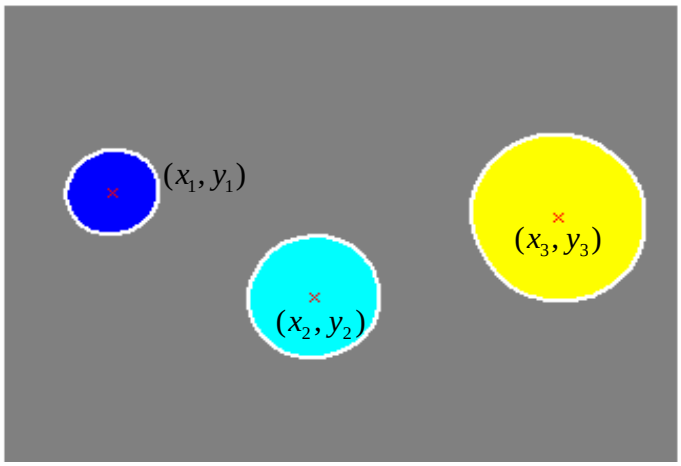
Eşik değeri seçimi Otsu yöntemi ile yapılmıştır. En eski eşikleme yöntemlerinden biri olan bu eşikleme yönteminde optimum eşik değerinin belirlenmesi nesne ve arka plan piksellerine ilişkin ağırlıklandırılmış toplam sınıf içi degişintilerinin minimize edilmesi ile gerçekleştirilmektedir. Bu eşikleme yönteminin sınıflardaki piksel sayıları birbirine yakın olduğu sürece iyi sonuçlar verdiği gözlemlenmektedir.

6.1.3. Etiketleme

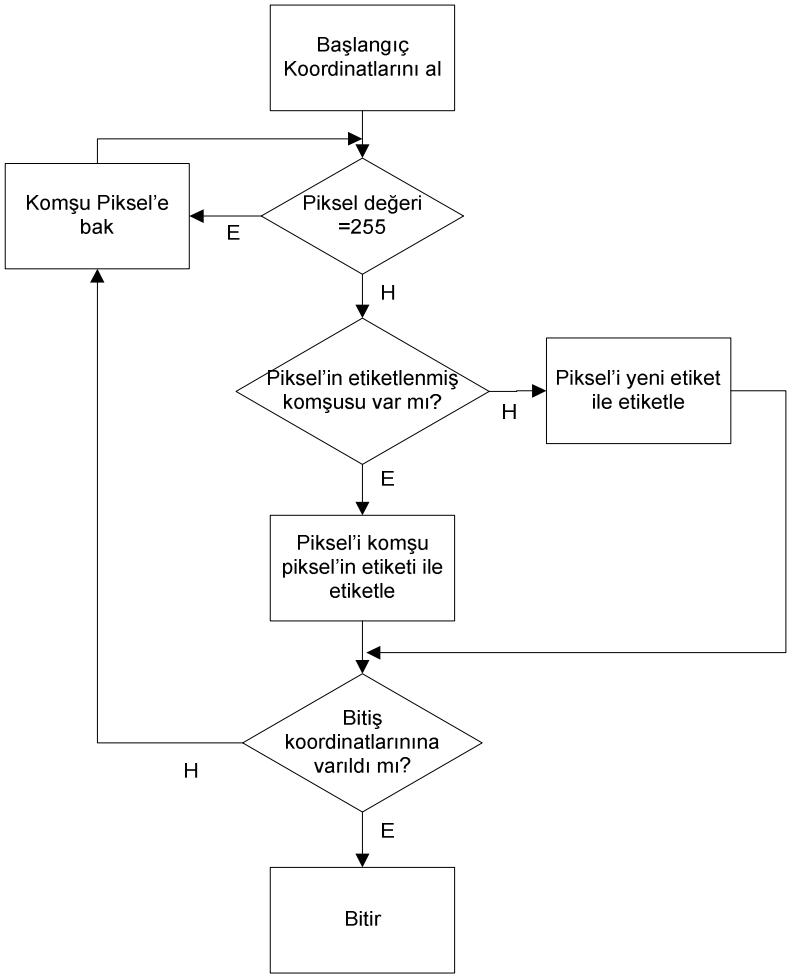
İkili bir resim içerisinde birbiriyle hiçbir komşuluğu olmayan nesnelerin değişik renklere boyanarak birbirlerinden ayrılmasına etiketleme denir [16].
Etiketleme işlemi, ikili resmin (0,0) noktasından başlayarak piksel piksel aranması ile yapılır. İkili resim içinde nesneler beyaz, zemin siyah renktedir. Tarama sırasında beyaz piksele (nesneye) rastlandığında bu pikselin 3x3 matris şeklindeki piksel komşularına bakılır. Bu piksel komşuları arasında önceden etiketlenmiş bir piksele rastlanırsa beyaz piksele de aynı etiket değeri verilir. Beyaz pikselin komşulukları arasında önceden etiketlenmiş bir piksel değeri yoksa son etiketleme değeri bir arttırılarak beyaz piksele verilir ve etiketleme tüm resme uygulanır (Şekil 6.8). Etiketleme işlemi sonucunda resimdeki nesneler farklı renge boyanır ve resimdeki renkler sayıldığında nesne sayısı bulunur. Bir pikselde Kırmızı-Yeşil-Mavi üç renk katmanı olduğu ve her renk katmanı 0–255 arasında değer aldığına göre bir resimde toplam olarak,

$$255 \times 255 \times 255 = 16.581.375 \qquad (6.3)$$

adet nesne sayılabilir.



Şekil 6.9. Etiketlenmiş resim



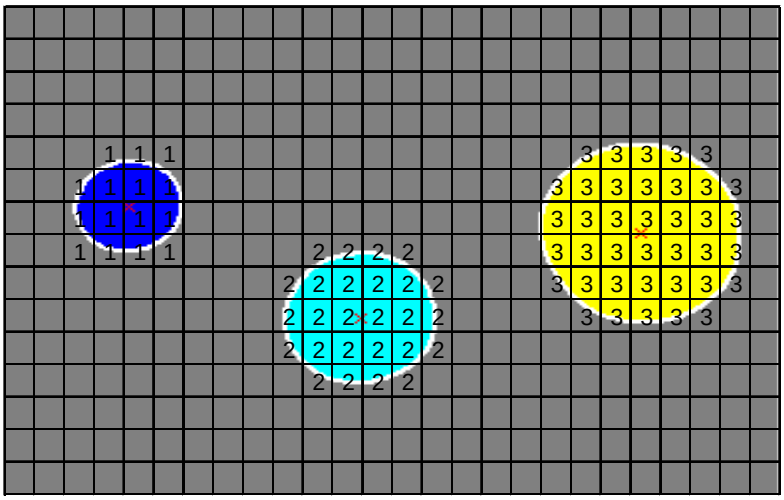
Şekil 6.10 Etiketleme algoritmasının akış diyagramı

Basit etiketleme algoritmasının sözde kodu aşağıdadır;

1) Görüntünün başlangıç ve bitiş koordinatını bul, $(x_0, y_0),(x_n, y_n)$;
2) Başlangıç koordinatından bitiş koordinatına kadar pikselleri tara;
3) Eğer piksel değeri 255 ise, komşu piksel değerlerine bak;
4) Komşu pikseller arasında etiketlenen varsa, pikseli de aynı değer ile etiketle, komşu pikseller arasında etiketlenen yoksa en son etiket değerini piksele ver;
5) Etiket değerini bir arttır;

Etiketleme yapılan resimdeki renkler sayılarak birbirine piksel komşuluğu olmayan nesnelerin sayısı bulunur.

Dairesel kesitli nesneleri etiketlemek ve iki boyutlu koordinat bulmak için, çevre-alan oranı ile başlangıç ve bitiş koordinatları bulunan engeller, ikili resim içinde, zemin rengine boyanır ve tekrar yapılan etiketleme ile resim içindeki dairesel kesitli nesnelerin sayısı bulunur. Şekil 6.11'de görüldüğü gibi nesneler, farklı renkler kullanılarak etiketleme yapılmıştır. Etiketleme işlemi üst satırdan alt satıra doğru yapıldığından nesnelerin numaralandırılması satır sırasına göre belirlenmiştir.



Şekil 6.11 Dairesel kesitli nesnelerin etiketlenmesi

Etiketlemeleri yapılan dairesel kesitli nesnelerin iki boyutlu koordinatlarının bulunması (6.4) numaralı eşitlikte gösterildiği gibi nesnenin x ve y koordinatların toplamının, nesnenin toplam piksel sayısına (kapladığı alana) bölünerek ağırlık merkezi koordinatları hesaplanır [14].

$$x = \frac{\sum_{i}^{n}\sum_{j}^{m} jB[i,j]}{\sum_{i}^{n}\sum_{j}^{m} B[i,j]} \qquad y = \frac{\sum_{i}^{n}\sum_{j}^{m} iB[i,j]}{\sum_{i}^{n}\sum_{j}^{m} B[i,j]} \tag{6.4}$$

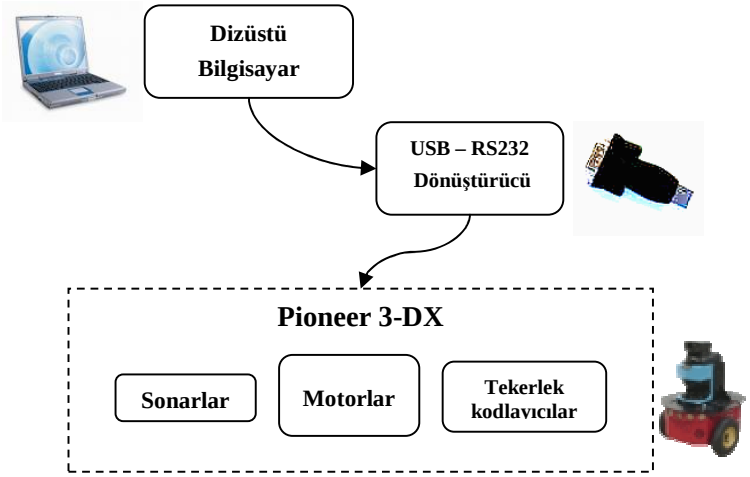
i,j = Nesne başlangıç koordinatları, B = Etiketlenen resim, n,m = Nesne bitiş koordinatları

6.3. İki Aşamalı Algoritma

İki aşamalı algoritmanın uygulanması için MATLAB programından yararlanılmıştır. Bu algoritmanın uygulama aşamasında ilk olarak en yakın engel bulunur. Başlangıç noktasından bitiş noktasına çekilen doğru parçasının kestiği engeller bir dizi içerisine kaydedilir ve başlangıç noktasına koordinatları en yakın engel en yakın engel olarak atanır. İlk engel tek engelmiş gibi düşünülüp başlangıç ve bitiş noktasından dairesel engele teğetler çekilir. Başlangıç ve bitiş noktasından çekilen teğetlerden en kısa yol uzunluğunu sağlayan teğetler seçilir ve tek engel için gidilecek yol belirlenmiş olur. Bitiş noktasından çekilen teğet noktası bir sonraki adımın başlangıç noktası olarak atanır ve işlem bitiş noktasına varana kadar devam ettirilir. Bu işlemler sırasında başlangıç noktası ile çemberin teğet noktası arasına ekstra engeller girebilir. Böyle bir durum olursa ekstra engel tek engelmiş gibi düşünülüp program devam ettirilir. İki aşamalı algoritmanın akış diyagramı ve işleyiş mantığı üst bölümlerde detaylı olarak anlatılmıştır.

6.4. Veri İletişimi

Pioneer 3-DX, üzerinde bulunan kişisel bilgisayar (PC) ile birlikte, tam olarak otonom, akıllı gezgin robot özelliğine sahiptir.. Pioneer 3-DX'in belirtilen bağlantı şekillerinden Şekil 6.12'de gösterilmiş olan donanım bağlantı yapısı kullanılmıştır.

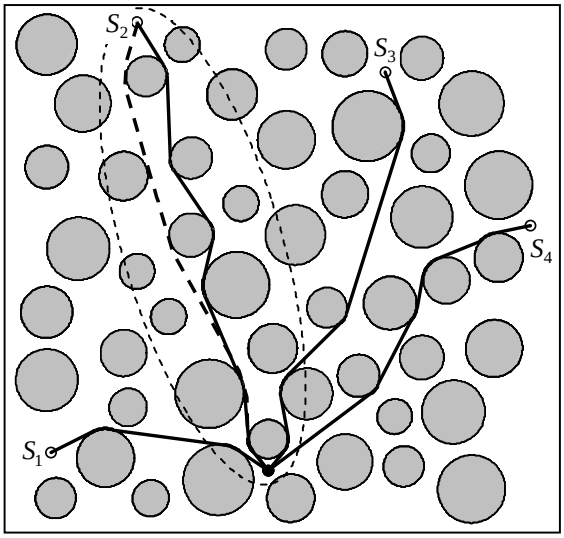


Şekil 6.12 Pioneer 3-DX donanım bağlantıları

6.5. Benzetim Sonuçları

Önerilen iki aşamalı algoritmayı sınamak amacıyla sayısal benzetimler yapılmıştır. Benzetimlerde dikdörtgensel bölge içerisinde rasgele engeller oluşturulur. Bir hedef konum seçilir. Farklı başlangıç konumlar alınarak önerilen algoritma çalıştırılır.

Gerçekleştirilmiş benzetimlerden birisinin sonuçları Şekil 3'te gösterilmiştir. Genel algoritmanın 1.aşamasından elde edilen yollar sürekli çizgilerle gösterilmiştir. Başlangıç konumlardan birisi (S_2) için optimal yol (kesikli çizgi) ile 1.aşamadan elde edilen yol arasında ciddi farklılıklar olduğu tespit edilmiştir. Diğer durumlarda yollar yakın olduklarından, şekli fazla karmaşıklaştırmamak için, optimal yollar çizilmemiştir. Başlangıç konumun S_2'de olduğu durum için 2.aşamada kullanılan bölgeyi sınırlandıran elips, şekilde ince kesikli çizgi ile gösterilmiştir.



Şekil 6.13 Engelin bulunduğu ortamda hesaplamalardan elde edilen optimale yakın (sürekli çizgiler) ve optimal (kesikli çizgi) yollar

Ağırlıklı bir çizgede iki düğüm arasındaki en kısa yolun uzunluğunu bulmak için Dijkstra'nın en kısa yol algoritması $O(n^2)$ işlem (toplama ve karşılaştırma) gerçekleştirir [18].

Benzetimlerde, 2.aşamada kullanılan bölgeyi sınırlandıran elips, dikdörtgensel bölgenin yaklaşık 1/5'ini kapsamaktadır. Dijkstra algoritmasında kullanılan düğümlerin sayısı n olduğundan ve algoritmanın işlem karmaşıklığı $O(n^2)$ derecesinde olduğu için, elips ile çevrili bölgenin kullanılması yaklaşık 1/25 oranında daha az işlem yapılması demektir. Bu benzetim için, elips ile sınırlandırılmış olurlu bölge uygulaması yaklaşık 25 kat daha az işlem yapılmasını sağlar.

Benzetimler, önerilen algoritmanın, engellerden sakınma optimizasyon probleminin çözümü için kullanılabilir olduğunu gösteriyor. Elde edilen sonuçlara göre, robot uygulamalarında, önemli olan zaman ve bellek kısıtlamaları da göz önünde bulundurulursa, algoritmanın yalnız birinci aşaması ile yetinilebilir.

6.6 Deney Sonuçları

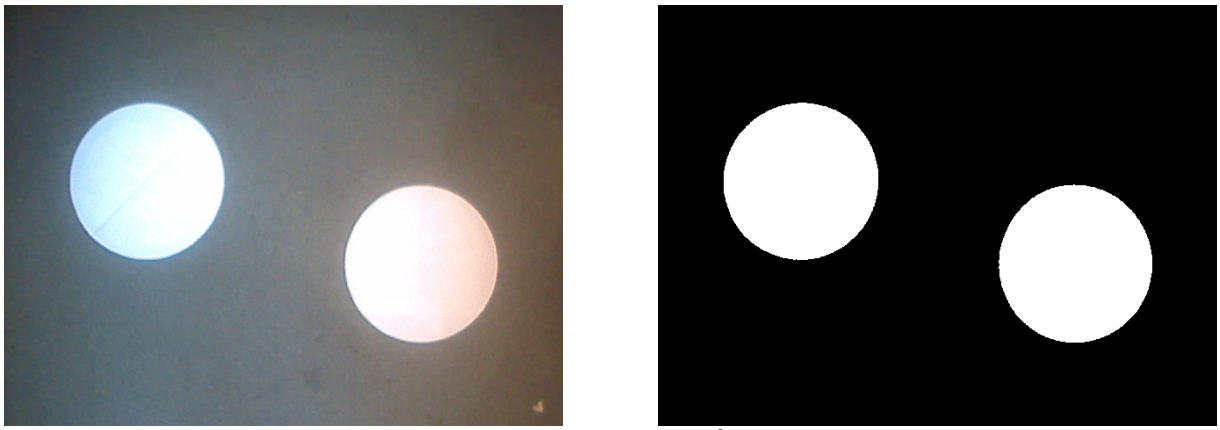
Deneysel çalışmada, deney düzeneğinden tek kamera ile alınan sayısal görüntülere, görüntü işleme teknikleri uygulanarak, dairesel kesitli engellerin konumlarının bulunması ve başlangıç konumundan bitiş konumuna en kısa yol optimizasyonu gerçekleştirilmiştir. Bulunan bu yol gezgin robot kullanılarak izlenmiştir.

Deney düzeneğinden sabit odak uzaklığında çekilen fotoğraflar önce gri seviye resimlere dönüştürülüp ardından bir eşik değeri ile eşiklenerek ikili (binary) resimlere dönüştürülmüştür bkz. Şekil 6.14. İkili resimlerdeki dairesel engeller etiketlenerek sayıları bulunmuş ve merkezlerinin (x,y) koordinatları tespit edilmiştir bkz. Şekil 6.15.

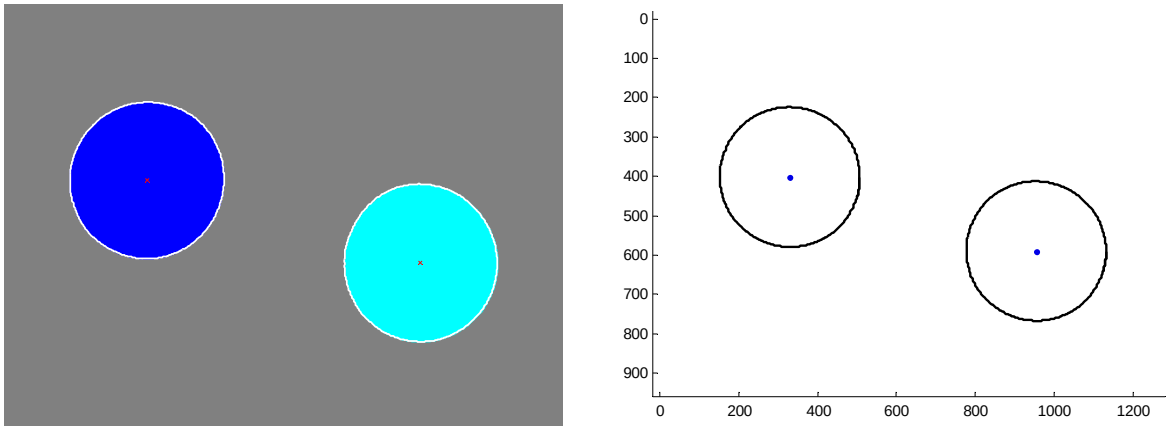
Bu örnek için başlangıç ve bitiş noktası, engellerin merkez koordinatları ve yarıçapları aşağıdaki çizelgedeki gibi bulunmuştur.

Çizelge 6.1 Şekil 6.14 deki örneğin başlangıç ve bitiş noktası, engellerin merkez koordinatları ve yarıçapları.

Koordinatlar		
X ekseni	Y ekseni	
-180.00	430.00	Başlangıç
1300.00	473.00	Bitiş
Engeller		Yarıçap
329.19	401.90	177.66
954.15	589.86	177.09



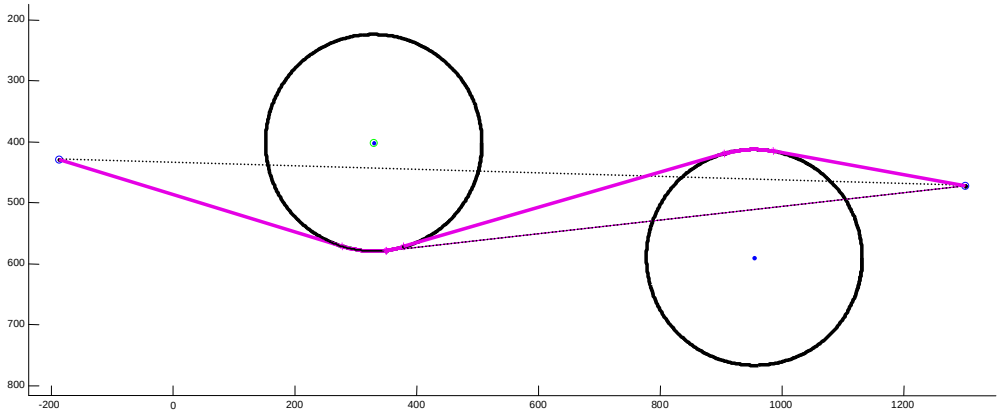
Şekil 6.14 Ortamın Görüntüsü ve İkili resim



Şekil 6.15 Etiketlenmiş resim ve sanal ortamda yeniden oluşturulmuş ortam

Engellerin koordinatları ve yarıçapları dairesel engel sayısı, başlangıç ve bitiş noktasının koordinatları ve engeller arası minimum mesafe bilgileri ortam.txt dosyasına (EK2) kaydedilir.

Ortam.txt doyası kullanılarak engeller MATLAB programı ile sanal ortamda yeniden oluşturulur ve iki aşamalı algoritmanın ilk kısmı olan geometrik yaklaşım kullanarak optimale yakın yol ve uzunluğu bulur bkz. Şekil 6.16.

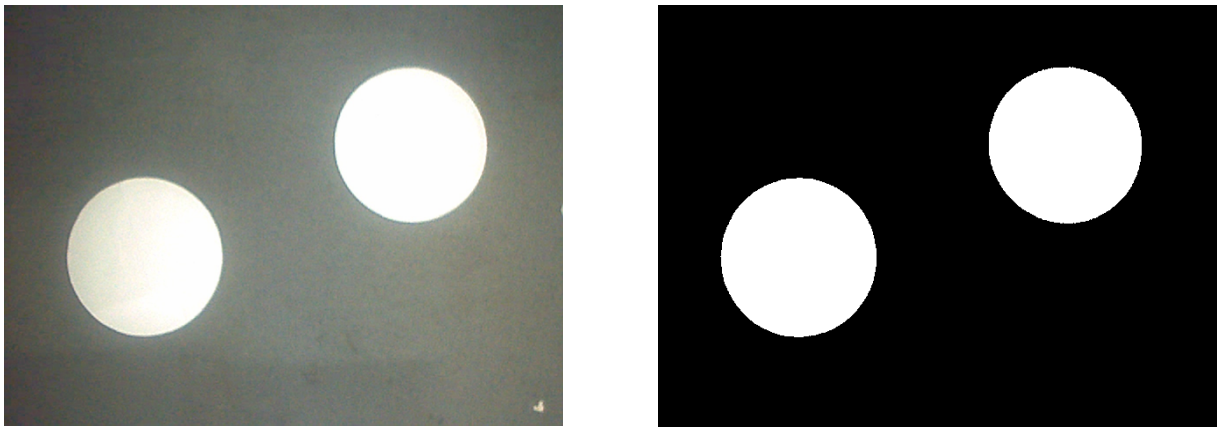


Şekil 6.16 Geometrik yaklaşım kullanarak bulunan optimale yakın yol

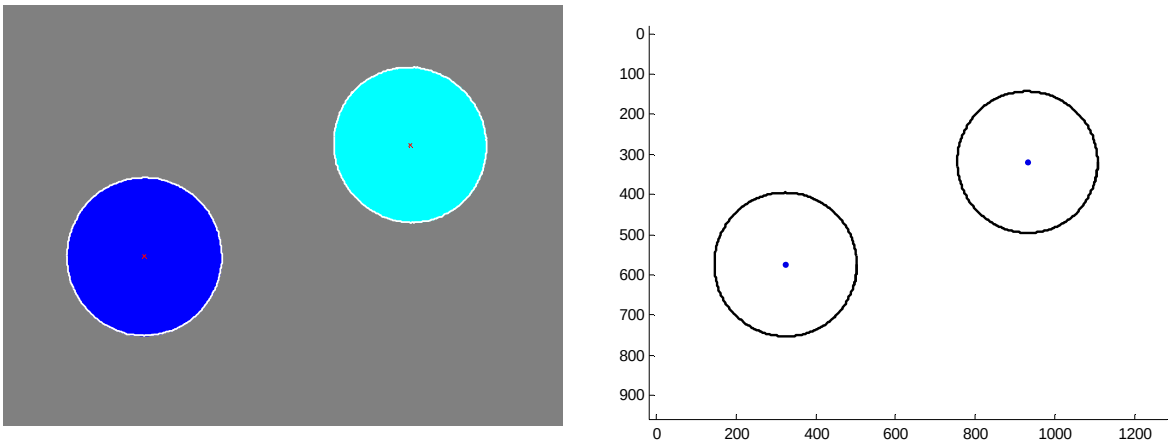
Deneysel çalışmada ikinci olarak engellerin yerleri değiştirilmiş, ortamın görüntüsü alınıp, gri seviye resimlere dönüştürülüp ardından bir eşik değeri ile eşiklenerek ikili (binary) resimlere dönüştürülmüştür bkz. Şekil 6.17. İkili resimlerdeki dairesel engeller etiketlenerek sayıları bulunmuş ve merkezlerinin (x,y) koordinatları tespit edilmiştir bkz. Şekil 6.18.

Çizelge 6.2 Şekil 17 deki örneğin başlangıç ve bitiş noktası, engellerin merkez koordinatları ve yarıçapları.

Koordinatlar		
X ekseni	Y ekseni	
-180.00	430.00	Başlangıç
1300.00	473.00	Bitiş
Engeller		Yarıçap
323.90	574.06	178.96
931.07	319.16	175.09



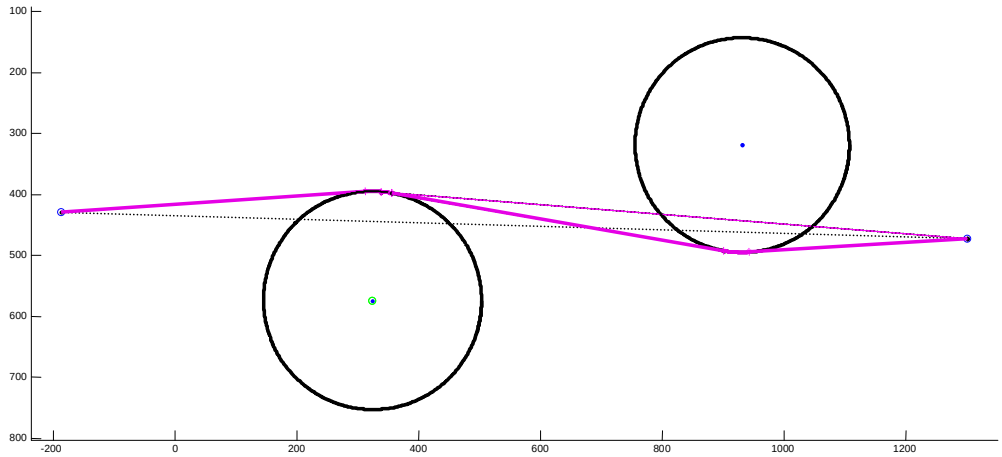
Şekil 6.17 Ortamın Görüntüsü ve İkili resim



Şekil 6.18 İkinci örnek için etiketlenmiş resim ve sanal ortamda yeniden oluşturulmuş ortam

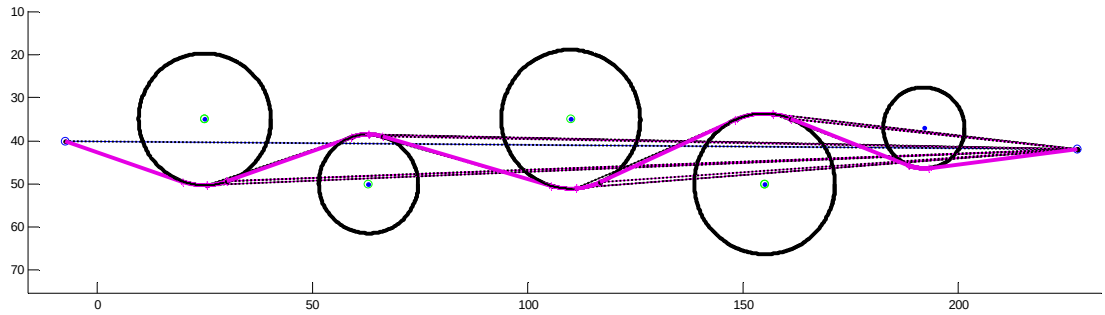
Bu örnek içinde bir önceki basamaklar takip edilir. Engellerin koordinatları ve yarıçapları dairesel engel sayısı, başlangıç ve bitiş noktasının koordinatları ve engeller arası minimum mesafe bilgileri ortam.txt dosyasına (EK2) kaydedilir.

Ortam.txt doyası kullanılarak engeller MATLAB programı ile sanal ortamda yeniden oluşturulur ve iki aşamalı algoritmanın ilk kısmı olan geometrik yaklaşım kullanarak optimale yakın yol ve uzunluğu bulur bkz. Şekil 6.19.



Şekil 6.19 İkinci örnek için geometrik yaklaşım kullanarak bulunan optimale yakın yol

Son örnek olarak farklı yarıçaplardan oluşan beş tane engel başlangıç ve bitiş noktası, engellerin merkez koordinatları ve yarıçapları, engeller arası minimum uzaklık değerleri verilerek sanal ortamda oluşturulur. Bu değerler kullanılarak bir gerçek ortam hazırlanır. Gerçek ortamda robotun engellere çarpmadan gidişi test edilmiştir.



Şekil 6.20 Beş engel için Geometrik yaklaşım kullanarak bulunan optimale yakın yol

Çizelge 6.3 Şekil 6.20 deki örneğin başlangıç ve bitiş noktası, engellerin merkez koordinatları ve yarıçapları.

Koordinatlar		
X ekseni	Y ekseni	
-7.45	40	Başlangıç
227.45	42	Bitiş
Engeller		Yarıçap
155.00	50.00	16.33
63.00	50.00	11.51
110.00	35.00	16.13
25.00	35.00	15.35
192.00	37.00	9.42

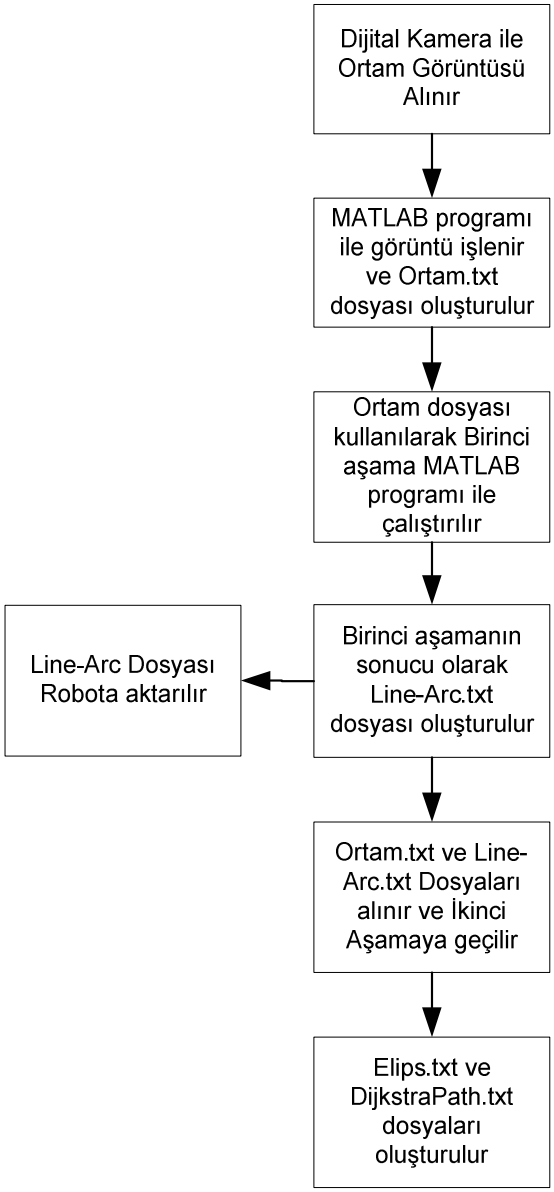
Algoritmanın ilk aşamasında, ikinci aşamada kullanılmak üzere linearc.txt dosyası (EK3) daha oluşturulur. Bu dosya içerisinde doğruların başlangıç ve bitiş koordinatları, daire parçalarının başlangıç ve bitiş koordinatları ve daire parçalarının ait olduğu çemberlerin merkez koordinatları ve çemberlerin yarıçapları tutulur.

Bulunan toplam yol uzunluğu, başlangıç ve bitiş noktasının koordinatları kullanılarak bir elips oluşturulur ve engellerin bulunduğu ortam, bulunan elips kullanılarak sınırlandırılır.

Elips ile sınırlandırılan bölge üzerinde bir ağ oluşturulur, engellerin bulunduğu düğümler yasaklı bölge sayılır ve bu düğümlere ulaşım sağlanmaz. Düğümlere ayrılan bölge üzerinde Dijkstra algoritması uygulanarak iki metin dosyası daha oluşturulur. Bu dosyalar; elipsin noktalarının koordinatlarını tutan elips.txt (EK4) ve Dijkstra yolunun noktalarının koordinatlarını tutan dijkstrapath.txt' dir (EK5) .

Robot uygulaması için EK3'deki metin dosyasındaki bilgiler C++ programı aracılığı ile robota aktarılmış ve robotun algoritmanın ilk aşamasında elde edilen yolu izlemesi sağlanmıştır.

Deneysel olarak; bu çalışmada gezgin robotun en iyi yolu bulması ve izlemesi başarı ile tasarlanmış, gerçekleştirilmiş ve farklı boyut ve konumlardaki dairesel engeller için test edilmiştir.



Şekil 6.21 Metin dosyalarının kullanıldığı yerleri gösteren akış diyagramı

7. SONUÇ

Düzlemde engellerden sakınma optimizasyon problemi araştırılmıştır. Problemin çözümü için iki aşamalı algoritma önerilmiştir. Algoritmanın birinci aşamasında geometrik gösterimlere dayanan yöntem üzerine optimale yakın çözüm elde edilir. Bu çözüm üzerine optimal yolun yer aldığı bölge bir elipsle sınırlandırılır. İkinci aşamada problem, çizgelerde en kısa yol problemine getirilir ve Dijkstra algoritmasının uygulanmasıyla optimal yol bulunur.

Önerilen iki aşamalı algoritmayı sınamak için sayısal benzetimler yapılmıştır. Benzetim sonuçlarının teoriye uygun ve doğru sonuçlar vermektedir.

Robot uygulamasında dairesel engeller arasından geçen robotun geçişi test edilmiştir. Bunun için, engellerin pozisyon bilgileri sayısal kameradan alınan görüntülerden tespit edilmekte ve bir noktadan istenilen diğer bir noktaya hareket gerçekleştirilmektedir. Zemine yerleştirilen cisimlerin yerleri değiştirilse bile, mobil robot bu cisimlere temas etmeden istenilen noktaya gidebilmektedir.

Deneysel çalışmada tek kameradan alınan dijital resim ile nesnenin konumları ve bu konumlara erişim için gerekli sıra bulunmuştur. Gezgin Robotun hareket optimizasyonunu gerçekleştirmek için önce dijital fotoğraflar görüntü işleme teknikleri ile işlenmiş ve dairesel kesitli nesnelerin merkezlerinin x, y koordinatları ile başlangıç ve bitiş x, y koordinatları bulunmuştur.

Sabit odak uzaklıklarında çekilmiş resimlerdeki dairesel engellerin, her biri için ayrı ayrı nesne sayımı yapılmıştır. Odak mesafe değerine göre dairesel kesitli nesnelerin yarıçapları bulunmuştur. Nesnelerin koordinat verilerinin iki aşamalı algoritmaya aktarılması ile en kısa yol problemi çözülmüştür.

İleriki çalışmalarda kameradan alınan görüntüde bulunan bozulmalar (lens bozulması gibi) ikili doğrusal (bilinear) denklemler seklinde modellenebilir ve daha düşük bir hata oranı ile problem çözülebilecektir. Ayrıca kullanılan kamera ve

platformda bazı değişiklikler yapılarak deneysel çalışma alanı genişletilebilir. Bu sayede de değişik algılayıcılarla donatılan mobil robota, sahip olduğu bütün algılayıcıları kullanmak suretiyle ortama yerleştirilen üç boyutlu engelleri de kolaylıkla aşma kabiliyeti kazandırılabilir. İleri aşamalarda renk algılayıcısı (colour sensor), kızıl ötesi algılayıcı (infra-red sensor), mesafe algılayıcısı (distance sensor) gibi bileşenler kullanılarak ortam içerisinde bulunan üç boyutlu engeller arasında hareket etme ve en kısa yolu bulma işlemleri de gerçekleştirilebilir.

Deneysel çalışmada öne çıkan bir diğer özellik ise uygulamanın tüm işlemlerinin gerçek zamanlı olarak yapılmamasıdır. İleriki çalışmalarda gerçek zamanlı bir uygulama düşünülmektedir. Son olarak gezgin robotun hızında değişiklikler yapılarak zaman optimizasyonun gerçekleştirilmesi düşünüllebilir.

Bu robot sisteminin gerçekleştirilmesinde karşılaşılan güçlükler çok çeşitli donanımların, bileşenlerin ve yazılımların bir tasarımda bir araya getirilmesidir.

Sonuç olarak; bu çalışmada gezgin robotun en iyi yolu bulması ve izlemesi başarı ile tasarlanmış, gerçekleştirilmiş ve farklı boyut ve konumlardaki dairesel engeller için deneysel olarak test edilmiştir.

KAYNAKLAR LİSTESİ

[1] Anderson J.A., “Discrete Mathematics with Combinatorics”, 2nd Ed., Prentice Hall, (2004).

[2] Bhattacharya, S.; Murrieta-Cid, R.; Hutchinson, S.; “Path planning for a differential drive robot: minimal length paths - a geometric approach, Intelligent Robots and Systems”, (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference, (2004).

[3] Connoly C.I., Burns J.B., Weiss R., “Path Planning Using Laplace’s Equation”, Proc. IEEE Int. Conf. Robot. Automat., (1991), 2102–2106.

[4] Dechter, Rina; Judea Pearl "Generalized best-first search strategies and the optimality of A*". Journal of the ACM**,** (1985), 505 - 536.

[5] Dijkstra E., “Two Problems in Connexion with Graphs”, Numerische Mathematik, (1959), 269-271.

[6] Dreyfus S., “Dynamic Programming and the Calculus of Variations”. New York, London: Academic, (1965).

[7] Enxiu Shi; Tao Cai; Changlin He; Junjie Guo; “Study of the New Method for Improving Artifical Potential Field in Mobile Robot Obstacle Avoidance”, Automation and Logistics, (2007).

[8] Gonzales R. C. and Woods R. E. Dijital Image Processing. Addison-Wesley Publishing Company Inc. UK, (1992).

[9] Hart, P. E.; Nilsson, N. J.; Raphael, B. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Transactions on Systems Science and Cybernetics SSC4, (1968), 100–107.

[10] Hart, P. E.; Nilsson, N. J.; Raphael, B. "Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". SIGART Newsletter, (1972), 28-29.

[11] Helgason R.V., Kennington J.L., Lewis K.R., “Cruise Missile Mission Planning: A Heuristic Algorithm for Automatic Path Generation”, Journal of Heuristics, Kluwer Academic Publishers, (2001), 473–494.

[12] http://capek.misto.cz/english/robot.html (18 Haziran 2006,WEB)

[13] http://www.mobilerobots.com/

[14] Jain, R.C, Kasturi, R. Schunck B.G. Machine Vision, ISBN 007-0320-187, Mc Graw-Hill, (1995).

[15] J. Borenstein, H. R. Everett, ve L. Feng, “Where am I? Sensors and Methods for Mobile Robot Positioning”, University of Michigan, (1996), 48 109.

[16] Kavraki L.E., Kolountzakış M.L., Latombe J.C., “Analysis of Probabilistic Roadmaps for Path Planning”, IEEE Transactions on Robotics and Automation, (1998), 166-171.

[17] Kavraki L.E., Svestka P., Latombe J.C., Overmars M.H., “Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces”, IEEE Transactions on Robotics and Automation, (1996), 566-580.

[18] Kenneth H. Rosen. Discrete Mathematics and Its Applications, (1999), 492-496.

[19] Kim, J.-O.; Khosla, P.K.; “Real-time obstacle avoidance using harmonic potential functions.”, IEEE Transactions on Robotics and Automation, (1992), 338-349.

[20] Liu Y.H., Arimoto S., “Path Planning using a Tangent Graph for Mobile Robots Among Polygonal and Curved Obstacles”, Int. J. Robot. Res. , (1992), 376–382.

[21] Leonard, J. ve Durrant-Whyte, H. Directed Sonar Sensing for Mobile Robot Navigation. Boston: Kluwert Accademic Publisher, (1992).

[22] N. Gasilov, M. Dogan, V. Arıcı, “Engellerin bulunduğu ortamda en kısa yol probleminin çözümü için iki aşamalı algoritma”, UMES'07, Kocaeli, Turkey, (2007), 408-411.

[23] Nguyen Hoang Viet; Ngo Anh Vien; SeungGwan Lee; TaeChoong Chung; Obstacle Avoidance Path Planning for Mobile Robot Based on Multi Colony Ant Algorithm Advances in Computer-Human Interaction, Automation and Systems, (2008).

[24] Nilsson, N. J. Principles of Artificial Intelligence. Palo Alto, California: Tioga Publishing Company, (1980).

[25] Pearl, Judea. Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, (1984).

[26] PITAS I. Digital Image Processing Algorithms. Prentice Hall, Europe. (1995).

[27] Rimon E. ve Koditschek D.E., “Exact Robot Navigation using Artificial Potential Functions”, IEEE Transactions on Robotics and Automation, (1992), 501–518.

[28] Sarmiento, A.; Murrieta-Cid, R.; Hutchinson, S.; “Planning expected-time optimal paths for searching known environments, Intelligent Robots and Systems”.(IROS 2004).Proceedings, (2004).

[29] Sundar, S.; Shiller, Z, ”Optimal obstacle avoidance based on the Hamilton-Jacobi Bellman equation”,1994. Proceedings. 1994 IEEE International Conference on Robotics and Automation, (1994).

[30] Sundar S., Shiller Z., “Optimal Obstacle Avoidance Based on the Hamilton-Jacobi-Bellman Equation”, IEEE Transactions on Robotics and Automation, (1997), 305-310.

[31] Xuan-Thu Le; Eun-Zu Hong; Han-Sung Kim; Young-Rok Cheon; Se-Han Lee; Sung Hyun Han; Yong-Guen An; “Real-time obstacle avoidance of mobile robots Control”, Automation and Systems, (2007).

[32] Xuesong Yan; Qinghua Wu; Jia Yan; Lishan Kang;”A Fast Evolutionary Algorithm for Robot Path Planning”, IEEE International Conference on Control and Automation, (2007).

EKLER LİSTESİ

EK 1 Pioneer 3-DX Robotunun Özellikleri

FEATURE	**ROBOT BASE = Pioneer 3-DX**
Length	44.5cm (44)
Width	40.0cm (38)
Height (body)	24.5cm (22)
Body clearance	6.5cm (6)
Weight (with min. battery capacity)	9kg
Payload of base platform with included battery *(other drive ratios available on large orders)*	23kg flat 14kg @ 13% grade
Body	1.6mm CNC fabricated, painted aluminum
Battery Access	Hinged, latched access door
Assembly	Allen hex screws
Battery	12V sealed, lead-acid
Charge	252 watt-hr
Run time, base platform	18-24 hrs
Recharge time, standard charger	12 hrs
Recharge time, 1 high-capacity charger	2.4 hrs
Docking station available	Yes
Drive	2-wheel drive, plus rear balancing caster
Wheel composition	Foam filled nylon, hard casters
Dr. Wheel diam.	19cm
Dr. Wheel width	5cm
Steering	Differential
Gear ratio	38.3:1
Pushing force	6kg
Swing radius	32cm
Turn radius	0cm
Translate speed max	1.2 m/sec
Traversable step max	2.5cm
Traversable gap max	8.9cm
Traversable slope max	25% grade
Traversable terrains	Wheelchair accessible
ARIA Robotics API	Yes
ACTS Color-Track	Requires onbd PC
Festival/Sphinx	Requires onbd PC
Laser Mapping & Nav.	Requires laser
Front sonar ring	8 included 1 each side 6 forward @ 20° intervals
Rear sonar ring	8 optional 1 each side 6 rear @ 20° intervals
Sonar range	15cm - 5m
Std. Position encoders	500 tick encoders
Surveillance option	Yes
IR Table Sensors	No
Compass option	Yes
Arm	Yes
Gripper option	Yes
Bumper option	Yes
Wireless Ethernet Operation option	Yes
Optional onboard computer	Embedded size
Max. no. Cards & ports	3 PC104+ 2 USB, 4 serial
Speaker	Piezo std., opt. high decibel
Laser option	Yes
Gyro option	Yes
GPS option	Yes
Vision/ACTS Color- Tracking options	Yes
Speech/Voice option	Yes
StereoCam Rangefinder option	Yes
Processor	Hitachi H8S
Sonar inputs	16 max
Custom I/O connections	8-bit external I/O bus w/ up to 16 devices + PC104 I/O boards
Analog/Digital	5 @ 0-5 VDC
Communications ports	3 RS-232 serial ports on microcontroller, 4 RS-232 and 1 Ethernet on optional embedded computer
Wireless Communications options	Radio modem pair without embedded computer; Ethernet station adapter & access point with
Flash Memory	1 mB
Power switches	1 main; 2 auxiliary
LCD display	-na-
Reset pushbutton	Warm reboot
Function pushbutton	Self-test
Main power switch	Robot power; 12VDC; red LED indicator
Radio power switch	Radio modem or other 12VDC option
Aux power switch	12VDC
Motors pushbuttons	Single enable/disable
Serial comm ports	9-pin RS232 with Rcv and Xmt LED indicators
Joy drive port	Off & opt. onbd.
Charging	12VDC charge port & Docking

EK 2 Ortam Dosyası

```
%Ortam.txt
%Bu dosya ortam bilgilerini tutmaktadır. İçerisinde ortam boyutları,
%engel sayısı, başlangıç ve bitiş noktasının koordinatları, en küçük ve
%en büyük yarıçap, engeller arası uzaklık, engellerin merkezlerinin
%koordinatları ve yarıçaplarının bilgileri vardır.

   746            a ortam boyutu
   534            b ortam boyutu
     4            engel sayisi
 50.00 120.00     start
320.00 130.00     finish
 83.76            min radius
126.50            max radius
  5.00            d(engeller arasi minimum uzaklik)



          obstacles
x_centr   y_centr   radius
136.72    292.43     83.76
306.75    136.00    106.84
343.15    405.21    106.98
585.74    232.64    126.50
```

EK 3 Line ve Ark Dosyası

```
%Line_Arc.txt
%Bu dosya ortam dosyası kullanılarak ve İlk aşama (geometrik yaklaşım)
%algoritmasının sonucunda üretilmiştir. İçerisinde doğru ve arkların
%başlangıç ve bitiş noktaları, arkların üzerinde bulunduğu dairenin
%merkez koordinatları ve yarıçapları tutulmaktadır. Ayrıca elips
%çizmek için kullanılmak üzere ilk aşama sonucu bulunan yolun uzunluğu
%da dosyanın sonunda verilmektedir.

                                        Circle          type
beginning_point   ending_point   xcenter ycenter radius kind-of-curve
 23.54  80.13     32.41 79.66                           line
 32.41  79.66     37.54 77.31    32.00  72.00  7.67     ark
 37.54  77.31     49.10 66.32                           line
 49.10  66.32     49.39 66.03    45.00  62.00  5.96     ark
 49.39  66.03     67.38 51.37                           line
 67.38  51.37     69.27 48.95    63.00  46.00  6.93     ark
 69.27  48.95     73.67 39.58                           line

 67.67  yolun uzunlugu
```

EK 4 Elips Dosyası

```
%Elips.txt
%Bu dosya Line-Arc dosyasının işlenmesi sonucunda oluşturulmuştur.
%İçerisinde Bölgeyi sınırlandırmak için oluşturulan Elips üzerindeki
%noktaların x ve y koordinatları tutulmaktadır.
      x           y
  0.75883E+02  0.37927E+02              0.77044E+02  0.43431E+02
  0.75948E+02  0.38012E+02              0.77025E+02  0.43573E+02
  0.76011E+02  0.38099E+02              0.77004E+02  0.43717E+02
  0.76073E+02  0.38187E+02              0.76982E+02  0.43861E+02
  0.76132E+02  0.38276E+02              0.76958E+02  0.44007E+02
  0.76190E+02  0.38367E+02              0.23272E+02  0.83360E+02
  0.76247E+02  0.38459E+02              0.23169E+02  0.83307E+02
  0.76301E+02  0.38552E+02              0.23069E+02  0.83253E+02
  0.76354E+02  0.38647E+02              0.20079E+02  0.77908E+02
  0.76405E+02  0.38742E+02              0.20076E+02  0.77778E+02
  0.76455E+02  0.38840E+02              0.20076E+02  0.77648E+02
  0.76503E+02  0.38938E+02              0.20077E+02  0.77516E+02
  0.76549E+02  0.39038E+02              0.20080E+02  0.77382E+02
  0.76593E+02  0.39139E+02              0.20084E+02  0.77248E+02
  0.76636E+02  0.39241E+02              0.20091E+02  0.77113E+02
  0.76676E+02  0.39345E+02              0.20099E+02  0.76977E+02
  0.76716E+02  0.39450E+02              0.20109E+02  0.76839E+02
  0.76753E+02  0.39556E+02              0.20121E+02  0.76701E+02
  0.76789E+02  0.39663E+02              0.20134E+02  0.76561E+02
  0.76822E+02  0.39772E+02              0.20149E+02  0.76421E+02
  0.76855E+02  0.39881E+02              0.20166E+02  0.76279E+02
  0.76885E+02  0.39993E+02              0.20185E+02  0.76137E+02
  0.76914E+02  0.40105E+02              0.20206E+02  0.75993E+02
  0.76941E+02  0.40218E+02              0.20228E+02  0.75849E+02
  0.76966E+02  0.40333E+02              0.20252E+02  0.75703E+02
  0.76989E+02  0.40449E+02              0.20278E+02  0.75557E+02
  0.77011E+02  0.40566E+02              0.20305E+02  0.75409E+02
  0.77031E+02  0.40685E+02              0.20334E+02  0.75261E+02
  0.77049E+02  0.40804E+02              0.20365E+02  0.75112E+02
  0.77066E+02  0.40925E+02              0.20398E+02  0.74961E+02
  0.77080E+02  0.41047E+02              0.20432E+02  0.74810E+02
  0.77093E+02  0.41170E+02              0.20469E+02  0.74658E+02
  0.77104E+02  0.41294E+02              0.20506E+02  0.74505E+02
  0.77114E+02  0.41419E+02              0.20546E+02  0.74351E+02
  0.77121E+02  0.41545E+02              0.20588E+02  0.74196E+02
  0.77127E+02  0.41673E+02              0.20631E+02  0.74040E+02
  0.77131E+02  0.41802E+02              0.20675E+02  0.73884E+02
  0.77134E+02  0.41932E+02              0.20722E+02  0.73726E+02
  0.77134E+02  0.42062E+02              0.20770E+02  0.73568E+02
  0.77133E+02  0.42194E+02              0.20820E+02  0.73409E+02
  0.77130E+02  0.42328E+02              0.20872E+02  0.73248E+02
  0.77126E+02  0.42462E+02              0.20925E+02  0.73088E+02
  0.77119E+02  0.42597E+02              0.20981E+02  0.72926E+02
  0.77111E+02  0.42733E+02              0.21037E+02  0.72763E+02
  0.77101E+02  0.42871E+02              0.21096E+02  0.72600E+02
  0.77089E+02  0.43009E+02              0.21156E+02  0.72436E+02
  0.77076E+02  0.43149E+02              0.21218E+02  0.72271E+02
  0.77061E+02  0.43289E+02              0.21282E+02  0.72106E+02
```

0.21347E+02	0.71939E+02	0.60784E+02	0.37134E+02
0.21414E+02	0.71772E+02	0.60986E+02	0.37064E+02
0.21483E+02	0.71604E+02	0.61187E+02	0.36996E+02
0.21553E+02	0.71436E+02	0.61388E+02	0.36929E+02
0.21625E+02	0.71266E+02	0.61588E+02	0.36863E+02
0.21699E+02	0.71096E+02	0.61787E+02	0.36799E+02
0.21774E+02	0.70926E+02	0.61985E+02	0.36737E+02
0.21851E+02	0.70754E+02	0.62183E+02	0.36675E+02
0.21930E+02	0.70582E+02	0.62380E+02	0.36615E+02
0.22010E+02	0.70410E+02	0.62575E+02	0.36557E+02
0.22092E+02	0.70236E+02	0.62770E+02	0.36500E+02
0.40036E+02	0.49261E+02	0.62964E+02	0.36445E+02
0.40249E+02	0.49088E+02	0.63158E+02	0.36390E+02
0.40464E+02	0.48917E+02	0.63350E+02	0.36338E+02
0.40679E+02	0.48745E+02	0.63541E+02	0.36287E+02
0.40894E+02	0.48575E+02	0.63732E+02	0.36237E+02
0.53544E+02	0.40307E+02	0.63921E+02	0.36189E+02
0.53765E+02	0.40192E+02	0.64110E+02	0.36142E+02
0.53985E+02	0.40079E+02	0.64297E+02	0.36096E+02
0.54205E+02	0.39967E+02	0.64484E+02	0.36053E+02
0.54424E+02	0.39856E+02	0.64670E+02	0.36010E+02
0.54644E+02	0.39747E+02	0.73292E+02	0.36062E+02
0.54862E+02	0.39638E+02	0.73404E+02	0.36107E+02
0.55081E+02	0.39531E+02	0.73514E+02	0.36152E+02
0.55299E+02	0.39426E+02	0.73622E+02	0.36200E+02
0.55516E+02	0.39321E+02	0.73729E+02	0.36248E+02
0.55734E+02	0.39218E+02	0.73835E+02	0.36298E+02
0.55950E+02	0.39116E+02	0.73938E+02	0.36350E+02
0.56167E+02	0.39015E+02	0.74041E+02	0.36403E+02
0.56383E+02	0.38916E+02	0.74141E+02	0.36457E+02
0.56598E+02	0.38817E+02	0.74240E+02	0.36513E+02
0.56813E+02	0.38720E+02	0.74338E+02	0.36570E+02
0.57027E+02	0.38625E+02	0.74434E+02	0.36629E+02
0.57241E+02	0.38531E+02	0.74528E+02	0.36689E+02
0.57454E+02	0.38438E+02	0.74621E+02	0.36751E+02
0.57667E+02	0.38346E+02	0.74712E+02	0.36814E+02
0.57879E+02	0.38256E+02	0.74802E+02	0.36878E+02
0.58091E+02	0.38167E+02	0.74890E+02	0.36944E+02
0.58302E+02	0.38079E+02	0.74976E+02	0.37011E+02
0.58512E+02	0.37993E+02	0.75061E+02	0.37080E+02
0.58722E+02	0.37908E+02	0.75144E+02	0.37150E+02
0.58931E+02	0.37824E+02	0.75225E+02	0.37222E+02
0.59140E+02	0.37742E+02	0.75305E+02	0.37294E+02
0.59348E+02	0.37661E+02	0.75383E+02	0.37369E+02
0.59555E+02	0.37582E+02	0.75459E+02	0.37444E+02
0.59761E+02	0.37504E+02	0.75534E+02	0.37521E+02
0.59967E+02	0.37427E+02	0.75607E+02	0.37600E+02
0.60172E+02	0.37352E+02	0.75679E+02	0.37680E+02
0.60377E+02	0.37278E+02	0.75749E+02	0.37761E+02
0.60581E+02	0.37205E+02	0.75817E+02	0.37843E+02

EK 5 Optimal Yol Dosyası

```
%DijksraPath.txt
%Bu dosya Line-Arc dosyasının işlenmesi sonucunda oluşturulmuştur.
%İçerisinde Başlangıç noktasından bitiş noktasına gidilecek en kısa
%yolun x ve y koordinatları tutulur.

   x           y
  0.23709E+02  0.79993E+02
  0.30253E+02  0.79958E+02
  0.34666E+02  0.79455E+02
  0.37793E+02  0.77364E+02
  0.63951E+02  0.54452E+02
  0.69034E+02  0.49902E+02
  0.73501E+02  0.39717E+02
```