ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM491 ARAŞTIRMA TEKNİKLERİ I RAPORU
ZOR STATİK ORTAMLARDA YOL PLANLAMA

ONUR ÖÇALAN
08290449

Aralık,2012
Ankara

Özet

BLM491 Araştırma Techniques I Report

ZOR STATİK ORTAMLARDA YOL PLANLAMA

Onur Öçalan

Ankara Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği

Danışman: Öğretim Görevlisi Kurtuluş Küllü

Bu raporda statik ortamlarda yol planlama problemlerini çözmesi için geliştirilen bir PRM uygulamasının yazılım süreçleri anlatılmaktadır. Giriş bölümünde yol planlama problemlerinin tanımı ve kullanım alanları ile ilgili genel bilgiler verilmektedir. İkinci bölümde yol planlama ile ilgili bazı terimlerin tanımlamaları yapılmıştır. Yapılandırma uzayı, yol haritaları ve serbestlik derecesi kavramları incelenmiştir. Bu bölümde ayrıca statik ortamlarda yol planlamanın tanımı yapılmış ve örnekleme tabanlı yöntemlerin genel özelliklerine değinilmiştir. Örnekleme tabanlı bir yöntem olan PRM'in ayrılmaz parçası olasılıksal yol haritasının tanımı ve oluşturulmasıyla ilgili bir algoritma açıklanmıştır.

Üçüncü bölüm, simülasyon ortamı hazırlanırken gerek duyulan pozisyonların tutulması, çakışma algılama yöntemleri ve bağlanabilirlik testi gibi özelleştirilmiş bazı görevlerin anlatımından oluşmuştur. Bu bölümde simülasyon hazırlanırken karşılaşılan bazı sorunlara da değinilmiştir.

Bulgular bölümünde statik ortamlarda yol planlama problemlerinde kullanılan örnekleme, komşu yapılandırma ve engel sayılarının sonuçları nasıl değiştirdiğinden bahsedilmiştir. Bu etkenlerin ortaya çıkardığı sonuçlar simülasyon görselleriyle açıklanmaya çalışılmıştır. Son bölümde ise ulaşılmış sonuçlar değerlendirilmiş ve getirilebilecek yeniklerden bahsedilmiştir.

Abstract

BLM491 Research Techniques I Report

PATH PLANNING IN STATIC COMPLEX ENVIRONMENT

Onur Öçalan

Ankara University

Faculty of Engineering

Department of Computer Engineering

Supervisor: Instructor Kurtuluş Küllü

This report describe software processes of a PRM application which was developed to solve problems of path planning in static environments. Introduction provides general information about the definition and usage of path planning. There are descriptions of some terms related to path planning in the second chapter. In this chapter, configuration space, roadmaps and degrees of freedom are investigated, also has been defined path planning in static environment and discussed the general properties of the sampling-based methods. An algorithm is described for creation of probabilistic roadmap that is an integral part of PRM's.

The third section is composed some special tasks; keeping positions, collision detection methods and connectivity test that needed on preparation of the simulation environment.

In conclusion, how the variables that sampling, neighbors and the number of obstacles, effect the result in path planning in static environment. The outcomes of these factors are explained with visuals of the simulation and improvements are discussed.

İçindekiler

Bölüm 1

Giriş

Yol planlama; bilgisayar oyunları, CAD tasarımları, moleküler biyoloji, robotik, sanal ortamlar gibi birçok uygulama ve araştırma alanlarında önemli bir yer tutmaktadır. En genel haliyle yol planlama; bir başlangıç ve bitiş noktası arasında, hareketli birimler için yol bulma problemini çözmektedir. Bu problemi daha net açıklamak için iyi bilinen bir bilgisayar oyununu göz önüne alalım: Age of Empires (Şekil 1.1a). Bu strateji oyununda ordular birbirleri ile savaşmaktadır. Oyuncu, ordusundan herhangi bir birimi (asker veya işçi) seçtikten sonra harita üzerinde bir noktayı işaretleyerek hareket ettirebilmektedir. Daha sonra seçilen birim hedef belirlenen noktaya otomatik olarak gidecektir.[3]

(a) Age of Empires Ekran Alıntısı

(b) Endüstriyel Robot Kolu

Şekil 1.1

Tam olarak bu nokta, bilgisayarın çözmek zorunda olduğu bir yol planlama problemini ortaya çıkarır. Bu problemin çözümü; geçilmez olarak belirlenen engellerden (nehir,orman,binalar gibi) kaçınarak, tercihen en kısa yolu, en kısa zamanda bulmaktır. Dikkatli incelendiğinde oyun haritası ulaşılabilen veya ulaşılamayan kare şeklinde hücrelerden oluşmaktadır. Herbir hücre yalnızca oyun içerisinde bir birimi

taşıyabilmekte ve komşuluğundaki 8 hücreye ulaşabilmektedir. Burada bulunacak yol; komşu ve ulaşılabilir hücrelerin birbirlerine bağlanması ile oluşacaktır.

Bir başka zorlu yol planlama problemi, olası tüm durumların hücre yapısı gibi ayrık değilde sürekli olduğu endüstriyel robot kollarının (Şekil 1.1b) hareketlerinde ortaya çıkar. Bu robot kolları da, çevrelerindeki herhangi bir engel veya diğer robot kolları ile çarpışmalardan kaçınarak üç boyutlu bir ortamda hareket etmektedirler. Bu gibi durumlarda çözüm, problemi mantıklı bir şekilde kolay işlenebilir hale getirmektir. Hücre yöntemi kullanılan birçok yöntemden yalnızca birisidir. İki boyut üzerinde bir bilgisayar oyunu ortamı ile üç boyut üzerinde hareket eden endüstriyel robot kolu ortamı, farklı görünseler bile yapılandırma alanları (configuration space) açısından aynı genel formüle sahiptirler. Rapor içeriğinde fabrika yüzeyinde gezinen bir robotun hareketleri modellenirken uygulanan yöntemlerden bahsedilecektir.

Bölüm 2

Kuramsal Temeller

Yol planlama problemleri, geometrileri ve kordinatları değişken hareketli engellerin bulunduğu *Dinamik* ortamlarda ve geometrileri ve kordinatları sabit hareketsiz engellerin bulunduğu *Statik* ortamlarda olmak üzere iki farklı durumda ele alınabilir. Örneğin bir fabrika içerisinde gezinen robot kolu, hareketli engel olarak nitelendirebileceğimiz çalışan işçiler ve diğer robot kolları ile çarpışmadan işlemini gerçekleştirmelidir. Aynı şekilde bilgisayar oyunu Age of Empires yol planlayıcısı da dinamik bir çevre üzerinde çalışmaktadır. Diğer bir yandan karmaşık bir labirent içerisinde çıkışı bulmak isteyen bir robot ise statik ortamlarda yol planlama problemini çözmek zorundadır. Her iki durum için de uygulanabilir yöntemler mevcuttur. Bu raporda statik ortamlar üzerinde geliştirilen yöntemlerden bahsedilecektir.

2.1 Yapılandırma Uzayı (Configuration Space)

Yol planlama probleminin oluşabilmesi için aşağıda bulunan 4 maddeye ihtiyaç vardır.

- Hareket edecek olan nesnenin geometrik şekli
- Nesnenin hareket edeceği çevrenin (workspace[1]) geometrik tanımlaması
- Nesne hareketinin serbestlik derecesi tanımlaması
- Planlanacak olan yolun başlangıç ve bitiş yapılandırmaları

[1]Workspace nesnenin erişebildiği veya erişemediği tüm yapılandırmaları kapsar. Rapor içeriğinde bu yapı kısaca çevre olarak isimlendirilecektir.

Bu maddeler kullanılarak bir robotun yapılandırma uzayı oluşturulabilir. Örneğin çalışma alanındaki robotun yapılandırması, birkaç parametre kullanılarak tanımlanabilir. 2 boyutlu bir çevre üzerinde çalışan bir robot düşünelim. Bu robotun yapılandırması, bulunduğu yüzeyin x ve y kordinatları olarak belirtilebilir. Diğer bir örnek ile; eklemli robot kollarının yapılandırmasını belirtmek için de ardışık bağlantılar arasında yapılan açılar gösterilebilir. Bir robot yapılandırmasını özgün bir şekilde tanımlayabilecek, en düşük parametre sayısına *serbestlik derecesi* (degrees of freedom) denir. İşte bu robot yapılandırmalarının oluşturduğu kümeye *yapılandırma uzayı* adı verilir ve $\mathcal{Q}$ ile sembollendirilir. Kısaca robot kolunun erişebileceği tüm pozisyonların kümesine yapılandırma uzayı denir. Robotun serbestlik derecesindeki artış, yapılandırma uzayında da bir artışa neden olur.

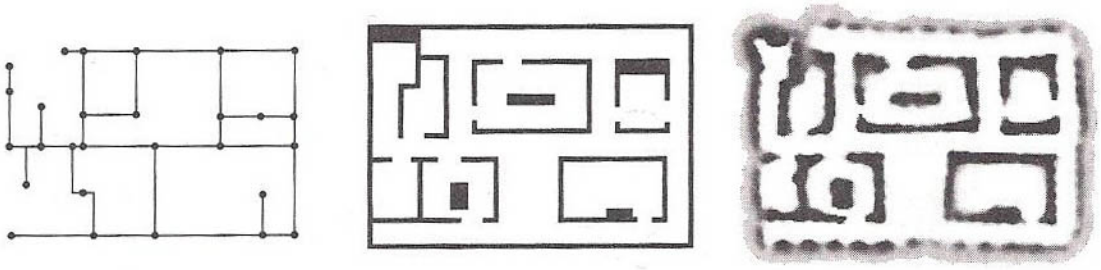
Bir robotun bulunduğu çevrede bazı engeller olabilir. Bu engelleri belirleyen yapılandırmalar yasaklı (*forbidden*) olarak isimlendirilir. Bir q yapılandırması yasaklı olarak belirlenmiş olsun. Robotun bu yapılandırmaya gelmesi çevrede bulunan başka bir engel ile çakışma durumunu ortaya çıkarır. Verilen bu c yasaklı yapılandırma, tüm yasaklı yapılandırmaları içeren $\mathcal{Q}_{forb}$ kümesinin bir elemanıdır. Robotun serbestçe gezinebileceği yapılandırmaların kümesi de $\mathcal{Q}_{free}$ ile gösterilir. Yapılandırma uzayı, serbest ve yasaklı yapılandırma kümelerinin birleşiminden oluşmaktadır.

Bir yol fonksiyonu, $\pi : [0, L] \rightarrow \mathcal{Q}$ şeklinde sürekli fonksiyon olarak tanımlanır. Burada L yolun uzunluğunu belirtmektedir. Oluşturulacak yolda çevrede bulunan engellerin hiçbiri ile çakışma olmamalıdır. Matematiksel bir şekilde ifade etmek istersek; başlangıç (start) yapılandırması $q_{start} \in \mathcal{Q}$ ve hedef (goal) yapılandırması $q_{goal} \in \mathcal{Q}$ olan bir çevrede bulunacak yolun özelliği: $\pi(0) = q_{start}$, $\pi(L) = q_{goal}$ olmak üzere $\forall(t \in [0, L] :: \pi(t) \in \mathcal{Q}_{free})$ olmalıdır. Kısaca başlangıç ve hedef yapılandırmalar arasında ilerlerken geçilecek tüm yapılandırmalar, $\mathcal{Q}_{free}$ kümesinden seçilmelidir. Bazı çevrelerde bu özelliklere sahip birden fazla yol bulunabilir. Bulunan yolların birbirinden üstünlüğü mesafelerinin kısalığıyla ölçülebilir.

2.2 Yol Haritaları (Roadmaps)

Bölüm 2.1 içerisinde bahsedildiği gibi yol planlayıcısı, başlangıç ve bitiş yapılandırmaları arasında bir yol planlar. Eğer aynı çevre içerisinde birden fazla yol bulunduğunu biliyorsak, bu yolları içerisinde barındıran bir veri yapısı oluşturmak mantıklı olacaktır. Böylece bu veri yapısı içerisinden yol daha hızlı bulunabilir. Oluşturulacak veri yapısına *harita* denir. *Haritalandırma* ise robotun çevresinde olup biteni

sensörleri ile modellemesidir. Robot bulunduğu çevre ile ilgili herhangi bir önbilgiye sahip olmadığı zamanlarda haritalandırma önem kazanır. Kapalı ortamlarda haritaların üç değişik gösterimleri bulunur: topolojik, geometrik ve hücresel (bakınız Şekil 2.1).



Şekil 2.1: Çevrelerin farklı gösterimleri: topolojik, geometrik ve hücresel

Bir *yol haritası* topolojik harita sınıfındandır. Yol haritasında robotun gidebileceği seçilmiş yapılandırmalar birbiri ile bağlantılıdır ve tamamı $\mathcal{Q}_{free}$ kümesinden seçilmiştir. Rastgele seçilen bu yapılandırmaların sayısının artması, yol haritasının kapsama alanını arttıracak ve bulunabilecek yollar arasında en kısa olana yaklaşımı sağlayacaktır. Topolojik bir yol haritasının üzerinde bulunan kenarların ve düğümlerin fiziksel anlamları vardır. Örneğin, bir yol haritası düğümü erişilebilir bir konumu belirler veya iki komşu düğüm arasında bulunan şerit bu ikisi arasındaki yolu belirler.

Robotların kullandığı yol haritaları günlük hayatta kullandığımız otoyol haritalarına benzemektedir. Gideceğimiz yere varmak için tüm sokaklara veya arayollara bakmak yerine hangi anayolları kullanacağımızı belirleriz. Daha sonra başlangıç noktamızdan belirlediğimiz anayola en kısa yoldan ve aynı şekilde hedef noktasına en yakın anayoldan çıkarak güzergahımızı oluştururuz. Burada yol haritası robot için anayolların bilgisini tutmaktadır. Bir başlangıç yapılandırması $q_{start} \in \mathcal{Q}_{free}$ konumundan başlayan robot ilk önce kendine en yakın yol haritası düğümüne gitmeli ve daha sonra $q_{goal} \in \mathcal{Q}_{free}$ hedep yapılandırmasına en yakın düğümden çıkıp sonuca ulaşmalıdır. Harita üzerinde ise gidilebilecek tüm komşu yapılandırmalar önceden belirlendiği için en kısa yol algoritmaları uygulanarak çıkışa en yakın düğüme nerelerden gidileceği bulunabilir.

> **Yol Haritası (Roadmap) :** Serbest yapılandırma uzayı $\mathcal{Q}_{free}$ içerisinde bir yol ile bağlanabilen bütün q_{start} ve q_{goal} yapılandırmaları için aşağıdaki özellikleri sağlayan tek boyutlu eğrilerin oluşturduğu yapıya *yol haritası* (roadmap) denir ve RM ile gösterilir.

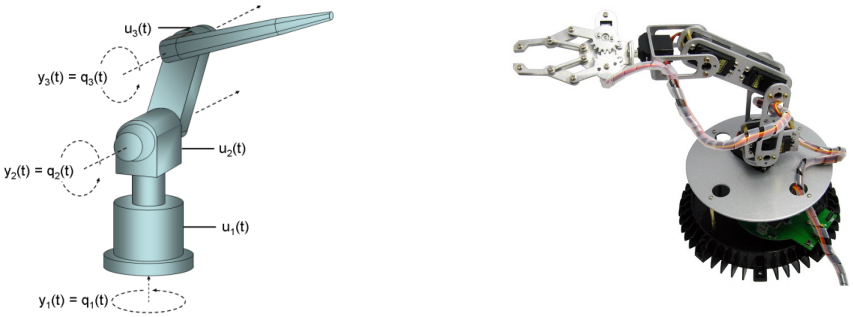
- Erişilebilirlik(Accessibility) : $q_{start} \in Q_{free}$ koşulunu sağlayan bir başlangıç yapılandırmasından, $q'_{start} \in RM$ koşulunu sağlayan bazı yapılandırmalara giden bir yol bulunmalıdır.
- Ayrılabilirlik(Departability) : $q'_{goal} \in RM$ koşulunu sağlayan bazı yapılandırmalardan, $q_{goal} \in Q_{free}$ koşulunu sağlayan yapılandırmaya giden bir yol bulunmalıdır.
- Bağlanabilirlik : RM içerisinde q'_{start} ve q'_{goal} yapılandırmaları arasında bir yol bulunmalıdır.

Bu özelliklerden tekinin bile sağlanmaması, başlangıç ve hedef yapılandırmaları arasında bir yolun bulunamayacağını gösterir. Bu nedenle yol planlamada doğru bir yol haritası oluşturma önemli yer tutmaktadır.

2.3 Serbestlik Derecesi (Degrees of Freedom)

Mekanik bir sistemde, yapılandırmayı oluşturan birbirinden bağımsız değerler özgürlük derecesini (dof [2]) oluşturur. Bu değerlerin tamamı bir sistemin durumu hakkında bilgiler içerir. Aynı zamanda yapılandırma uzayının boyutunu da belirler. Örneğin fabrikanın zemininde temizlik yaparak gezinen eklemsiz bir robotu ele alalım. Bu robotun zemin üzerindeki hareketleri, bulunduğu konumun (x,y) kordinatları ile gösterilebilir.

Özgürlük derecesinin boyutu her zaman tek değişkenden oluşmayabilir. Aynı fabrika içerisinde araç montajında çalışan 3 eklemli bir robot sisteminin durumu birkaç farklı parametrenin birlikteliğinden oluşur (Şekil2.2). Robotun kendi ekseni etrafında dönüşü, eklemlerin birbirleri ile yaptıkları açılar birleşerek sistem yapılandırması hakkında bilgi verir.



Şekil 2.2: Degrees of Freedom

[2]dof = degrees of freedom

Robotun kendi ekseni etrafında dönüşü, tüm kolların birbirleri ile yaptıkları açılar birleşerek sistemin yapılandırmasını oluşturur. Özgürlük derecesinin artması $\mathcal{Q}$ yapılandırma uzayının boyutunu arttırarak hesaplamaların karmaşıklaşmasına neden olmaktadır. Bölüm 2.4 içerisinde düzlem üzerinde gezinen bir robot için yol planlama aşamalarından ve kullanılan algoritmalardan söz edilecektir.

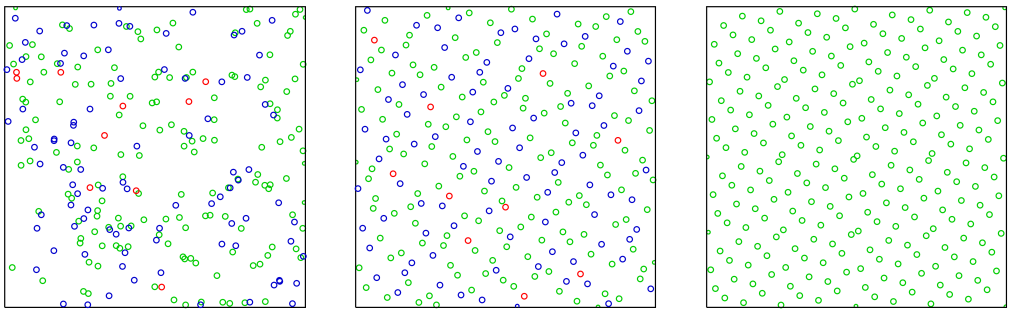
2.4 Statik Ortamlarda Yol Planlama

Sadece hareketsiz engellerin bulunduğu ortamlara statik ortamlar denir. Örneğin karmaşık labirent içinde ilerleyen bir robot statik bir ortamda yol planlama problemini çözmek zorundadır. Burada bulunulan ortamdaki engellerin boyutları, konumları, şekilleri önceden bilinebilir ya da sensörler yardımı ile ilerleme sırasında algılanabilir. Bu ayrımda kullanılacak metodlarda birbirinden farklılık göstermektedir. Bu raporda ortamda bulunan engellerin özelliklerinin önceden bilindiği durumlar için uygulanan bazı yöntemler incelenecektir.

2.4.1 Örnekleme Tabanlı Yöntemler

Q_{free} serbest yapılandırma uzayı içerisinde yol haritaları oluşturarak problemleri çözen bir çok yol planlayıcısı bulunmaktadır. Visibility graph, Cell Decompositions gibi yöntemlerde bu şekilde çalışmaktadır. Yapılandırma uzaylarının boyutları arttıkça bu yöntemlerin çözümleri yetersiz kalmaktadır. Örnekleme tabanlı yöntemlerin *(Sample Based Methods)* kullanımı diğer yöntemlerin verimli olarak çözemediği, yapılandırma uzayı boyutunun fazla olduğu bu tip problemleri çözmede devreye girmektedir.

Örnekleme tabanlı yöntemler ilk olarak ortamdan Q_{free} serbest yapılandırma uzayı içerisinde olmak üzere yapılandırmalar toplayarak ön bilgiye sahip olurlar. Daha sonra topladıkları bu veriler ile yol planlama problemlerini çözerler. Yapılandırma seçimleri rastgele yapılabileceği gibi *Halton Sequence* veya *Hammersley Sequence* yöntemleri kullanılarak da yapılabilir. Şekil 2.3 2 boyutlu bir uzay içerisinde bu yöntemler kullanılarak elde edilmiş yapılandırmaları göstermektedir.



Şekil 2.3: Rastgele, Halton ve Hammersley Yöntemleri

2.4.2 Olasılıksal Yol Haritaları

Olasılıksal yol haritası yöntemi (Probabilistic Roadmap Method[3]) de örnekleme tabanlı bir yöntemdir ve iki safhadan oluşur. Öğrenme safhasında[4] robot için erişilebilir örnek yapılandırmalar üretilmektedir. Bu yapılandırmaların tamamı Q_{free} serbest yapılandırma uzayı içerisinden seçilir. Yapılandırma seçiminin rastgele yapılması birçok problem üzerinde etkili olarak çalışmaktadır. Yapılandırma seçiminin rastgele yapıldığı bu yöntemlere temel *(basic)* PRM'ler denir. PRM'in başarısı, seçilen bu yapılandırmaların Q_{free} uzayı içerisinde olup olmadığının kontrolünün hızına da bağlıdır. Robotun özgürlük derecesinin artması bu kontrolleri karmaşık ve zor hale getirecektir.

PRM, örnek yapılandırmaları elde edildikten sonra bir yol haritası oluşturacak şekilde aralarında bağlantı kurmaktadır. Sorgu safhası[5] olarak adlandırılan bu safhada, bir yapılandırmadan diğer bir yapılandırmaya bağlantı kurulurken arada geçilen tüm yapılandırmaların da Q_{free} kümesinde bulunma zorunluluğu vardır. Bu koşulları sağlayan bir yapı kurulduğu zaman robotun yapacağı tek işlem oluşturulan yol haritası üzerinde başlangıç ve hedef yapılandırmaları arasında ilerlemek olacaktır. Oluşturulan bu yol haritası yönlendirilmemiş bir çizge $G = (V, E)$ olarak ifade edilir. Burada V Q_{free} kümesinden seçilmiş robot yapılandırmalarını göstermektedir. E ise eğer varsa iki yapılandırma (q_1,q_2) arasında serbest bir yol belirtir.

Algoritma 1 bir yol haritasının oluşturulmasındaki tüm aşamaları göstermektedir. Başlangıçta G = (V,E) çizgesi boş kümeden oluşmaktadır. Daha sonra Q uzayından bir örnek yapılandırma alınır. Eğer bu yapılandırma herhangi bir engel ile çakışma yapmıyor ise yol haritasına eklenir. Bu işlem Q_{free} kümesinden n sayıda yapılandırma elde edilinceye kadar devam eder. Daha sonra tüm $q \in V$ yapılandır-

[3]Rapor içeriğinde Probabilistic Roadmap Methods yerine PRM kısaltması kullanılacaktır.
[4]Öğrenme Safhası(Learning Phase)
[5]Sorgu Safhası (Query Phase)

Algorithm 1 Yol Haritası Oluşturma Algoritması

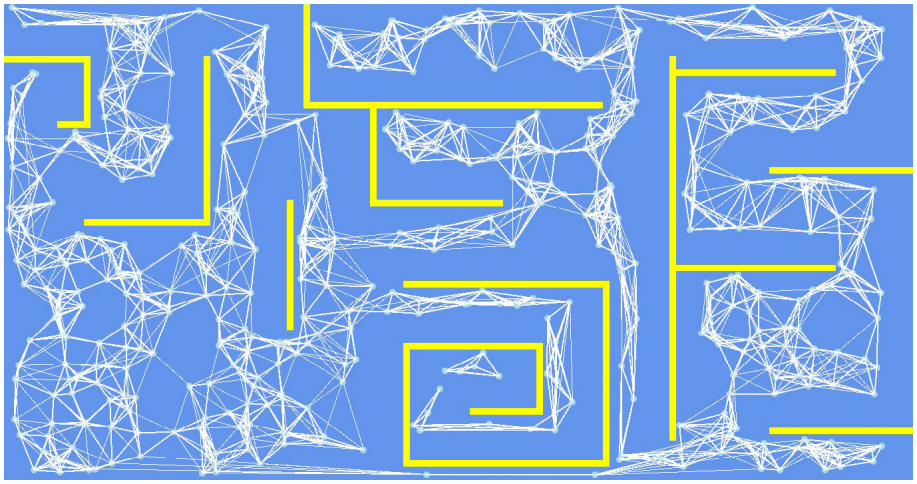
[V,E]

1: $V \leftarrow \emptyset$
2: $E \leftarrow \emptyset$
3: **while** $\mid V \mid < n$ **do**
4: **repeat**
5: $q \leftarrow$Q uzayından rastgele bir yapılandırma
6: **until** $q \in Q_{free}$
7: $V \leftarrow V \cup q$
8: **end while**
9: **for all** $q \in V$ **do**
10: $N_q \leftarrow$V içinde uzaklığı göre seçilmiş k. komşu
11: **for all** $q' \in N_q$ **do**
12: **if** (q,q') $\notin$ E **and**Δ(q,q') $\neq$ NIL **then**
13: $E \leftarrow E \cup$ (q,q')
14: **end if**
15: **end for**
16: **end for**

malarından N_q k elemanlı komşu kümelerindeki her bir yapılandırmaya bir yol olup olmadığına bakılır. Eğer (q,q') arasında bir yol bulunabiliyorsa bu yol E kümesine eklenir. Engellerin çok sık bulunduğu ortamlarda Q uzayı içerisinde n tane çakışma olmayan bir yapılandırma bulma işlemi zorlaşacaktır. Bu yüzden bu değerin fazla seçilmesi bu kısımdaki işlem süresini artıracaktır. Az seçilmesi durumunda ise yapılandırmaların birbirleri ile bağlantıları sağlanamayacak ve sonuçta bir yol bulunamayacaktır.

Bunların yanında bir yapılandırmanın bağlanacağı komşu sayısını tanımlayan k, verimliliği etkileyen önemli bir değişkendir. Bağlanacak komşu sayısının artması bağlanabilirlik testi yapılacak olan (q,q') çiftlerini artıracağından ek işlem yükü getirecektir. Az seçilmesi ise tüm serbest yapılandırmaların üzerinde bir yol oluşturamayacak kadar düşük elemanlı bir çizge oluşmasına neden olacaktır. Bu nedenlerden dolayı n yapılandırma sayısı ve k komşu sayısı değerleri arasındaki hassas denge korunmalıdır. Bağlanabilirlik testine Bölüm 3.2 içerisinde değinilecektir.

Şekil 2.4, XNA Framework[2] ile yapılmış bir yol haritası uygulamasını göstermektedir. Sarı bölgeler ortamda bulunan engelleri ifade etmektedir. Örnek yapılandırmaların sayısı n = 500 kabul edilmiş ve ona göre toplanmıştır. Her bir yapılandırma etrafında en fazla, bağlanabilir k = 8 komşu yapılandırmaya bağlanmıştır. Görüldüğü gibi neredeyse çoğu dar bölgelerde bağlantı sağlanmış, bazı yapılandırmalar dışarda kalmıştır. Bağlantıların sürekli olduğu yapılandırmalardan kopuk bağlantıların olduğu yapılandırmalara erişim olmayacaktır. Örnekleme sayısı veya



Şekil 2.4: 2 boyutlu Euclidean uzayındaki bir kare için yol haritası uygulaması

bağlanabilir komşu yapılandırma sayısı artırılarak bu istenmeyen durum ortadan kaldırabilir.

Bölüm 3

Materyal ve Yöntem

Bölüm 2 içerisinde anlatılan bilgiler doğrultusunda, 2 boyutlu düzlem üzerinde gezinen bir robot simülasyonu hazırlanmıştır. Düzlem üzerinde statik engeller bulunmaktadır. Bu simülasyon çalıştırıldığında ekranda, pozisyonları ve boyutları daha önceden belirtilmiş olan engellerin bulunduğu çevre gösterilmektedir. Başlangıç ve hedef yapılandırmaları simülasyon çalışma zamanında kullanıcıdan alınan robot, belirtilen yapılandırmalar arasında engelleri de hesaba katarak en kısa yolu bulmaya çalışmaktadır. Uygulama Microsoft firmasının sunduğu XNA çatısı altında C# programlama dili kullanılarak hazırlanmıştır. Bu bölümde simülasyon ortamının hazırlanması sırasında kullanılan bazı veri yapılarından ve yöntemlerden bahsedilecektir.

3.1 Pozisyonların Tutulması

Simülasyon ortamı 2 boyutlu olduğundan dolayı her cismin x ve y düzleminde bir pozisyon bilgisi tutulmaktadır. Aynı şekilde en ve boy bilgileri de tutulmalıdır. Bu nedenle *Position* isiminde sınıf tanımlanmıştır. Position sınıfı kendi içinde *leftUpCorner* ve *size* iki değişken bulundurmaktadır. XNA ortamının sağladığı metodları da kullanabilmek adına bu iki değişken *Vector2* tipinde tutulmaktadır. Böylece robotun ekran üzerinde belirlenen bir yerden başka bir yere gitmesi vektör işlemleri ile sağlanacaktır.

leftUpCorner; ekranda görüntülenecek herhangi bir cismin sol üst köşe piksel bilgisini tutmaktadır. Vector2 veri tipinin kendi içinde barındırdığı X ve Y float değişkenleri ile cismin ekran üzerinde yerleşimi sağlanmaktadır. Benzer şekilde size değişkeni de

cismin x ve y eksenlerindeki boyutlarını saklayacaktır. Bu bilgilerin saklanmasında Vector2 kullanımı barındırdığı metodlar ile oldukça kolaylık sağlamıştır.

Robotun bulunduğu çevrede engelleri ifade edebilmek için *Obstacle* sınıfı yaratılmıştır. Bu sınıf bir pozisyon bilgisi ile engelin konumunu ve boyutunu tutmaktadır. Bunların dışında ekranda gösterilecek diğer özellikleri (renk, isim) de tutulmaktadır. Engellerin tamamı daha önceden *Map* sınıfı altında tanımlanmaktadır. Bu sınıf ortamdaki tüm engelleri *obstacleList* isimli bir Obstacle listesinde tutmaktadır. Farklı çevreler yaratabilmek için bir değişken yardımı ile değişik engeller ekranda görüntülenmektedir.

Simülasyonun çalışması başladığı anda robot öğrenme safhasına geçmektedir. Bu safhasında gidebileceği yapılandırmalar hakkında bilgiler toplamaktadır. Topladığı bilgilerin saklanması için *Roadmap* sınıf tanımlaması yapılmıştır. Roadmap sınıfı aktif çevre ve engellerin pozisyonlarını tutmaktadır. Bu sınıf robot yapılandırmasına bağlı olarak örnekleme, örnekleri birbirine bağlama gibi metodlar çalıştıracaktır. Bu yüzden *Agent* isimli sınıfı da tutmaktadır. Ayrıca çalışması sonucu bir yol haritası oluşturacağından daha önceden tanımlanan Position sınıfından bir liste tanımlanmıştır.

3.2 Çakışma Algılama

Algoritma 1 bir yol haritası oluşturmak için gerekli adımları göstermektedir. Yol haritasını oluşturmaya başlamadan önce Q_{free} serbest yapılandırma uzayından n sayıda örnekleme alınmalıdır. Simülasyon bu örnekleri alırken rastgele seçimler yapmaktadır. Her örnek alındığında, V kümesine eklemeden önce seçilen yapılandırmanın Q_{free} içerisinde olup olmadığı kontrol edilmeli eğer bu kümenin elamanı değilse yeni bir yapılandırma seçilmelidir. Eğer bir yapılandırma hiçbir engel ile çakışma yapmıyor ise Q_{free} içerisindedir. Bu kontrol iki aşamadan oluşmaktadır.

İlk aşamada iki yapılandırma arasında bir çakışma olup olmadığını tespit eden bir metod yazılmıştır. Bu metod kendisine parametre olarak gelen iki yapılandırmanın merkezlerini bulmaktadır. Merkez pikseli bulunurken sol üst köşenin x ve y düzlemdeki değerlerine sırasıyla en ve boy değerlerinin yarısı eklenmektedir. Daha sonra x ekseni üzerinde kesişme durumu ve y ekseni üzerinde kesişme durumaları incelenmektedir (Algoritma 2). Eğer her iki eksende birden kesişme durumu sağlanıyorsa sonuç doğru olarak döner. Eksenlerden en az birinde kesişme koşulunun sağlanmaması durumunda ise kesişme yoktur denilebilir.

Algorithm 2 İki Yapılandırma Arasında Çakışma Algılama

```
procedure INTERSECT(conf1, conf2)
    c1 ← conf1 yapılandırmasının merkez noktası
    c2 ← conf2 yapılandırmasının merkez noktası
    s1 ← conf1 yapılandırmasının boyutları
    s2 ← conf2 yapılandırmasının boyutları
    if |c1.X - c2.X|<|(s1.X + s2.X)/2|  &  |c1.Y - c2.Y|<|(s1.Y + s2.Y)/2| then
        return true
    else
        return false
    end if
end procedure
```

Bu yöntem ile Q_{free} içerisinden alınması gereken tüm örnek yapılandırmalar toplanmaktadır. Algoritma 2 konumları sabit iki yapılandırmanın çakışma yapıp yapmadığını kontrol ettiğinden dolayı rastgele seçilmiş iki yapılandırma arasında ilerleyen bir Agent nesnesinin, arada kalan bölgelerde de engellerle karşılaşma olasılığı vardır. Bu yüzden bu ara kısımlarda da çakışma kontrolü yapılmalıdır. Bağlanabilirlik testi ile bu işlem gerçekleştirilmektedir.

Algorithm 3 Bağlanabilirlik Testi

```
procedure isCONNECTABLE(startConf, goalConf, obstacleList)
    curConf ← startConf yapılandırması ile aynı, sanal bir yapılandırma
    repeat
        direction ← curConf - startConf          ▷ gidilecek yönü belirle
        curConf ← curConf + direction                          ▷ ilerlet
        for all obs ∈ obstacleList do
            if Intersect(curConf,obs.conf) then
                return false
            end if
        end for
    until curConf == goalConf
    return true
end procedure
```

Algoritma 3, Agent nesnesinin bir yapılandırmadan diğer yapılandırmaya giderken herhangi bir engel ile çakışıp çakışmadığını kontrol etmektedir. İlk başta bulunulan yapılandırmanın bir kopya yapılandırması oluşturulmaktadır (curConf). Hedef yapılandırmadan bu yapılandırma çıkarılıp normalizasyon işlemi gerçekleştirildiğinde bir yön değeri bulunur (direction). Daha sonra curConf yapılandırması goalConf yapılandırmasına eşit olana kadar yön değeri curConf değerine eklenerek ilerleme sağlanır. İlerlemenin her adımında curConf yapılandırması çevrede bulunan tüm en-

gellerin yapılandırmaları sırayla Intercest (Algoritma 2) metoduna gönderilir. Herhangi bir engel ile çakışma durumunda Intersect metodundan *true* değeri dönecektir. Böylece bu iki yapılandırmanın bağlanabilirlik testi başarız olacaktır.

3.3 En Kısa Yolu Bulma

Yol haritası oluşturulması tamamlandıktan sonra robotun gideceği iki yapılandırma arasında en kısa yol bulunmalıdır. Bir çizge üzerinde gezinerek iki düğüm arasında en kısa yolu bulan algoritmalardan herhangi biri seçilebilir. Dijkstra, Bellman-Ford, A* gibi birçok algoritma çizgeler üzerinde gezinerek en kısa yol problemleri için çözümler üretmektedir. Simülasyon uygulaması hazırlanırken $O(|E| \log |V|)$ çalışma zamanına sahip Dijkstra algoritması seçilmiştir.

Algorithm 4 Dijkstra

```
function DIJKSTRA(Çizge, başlangıç)
    for all düğüm ∈ Çizge do
        uzaklık[düğüm] ← ∞
        kaynak[düğüm] ← NULL
    end for
    uzaklık[başlangıç] ← 0
    Q ← Çizge içindeki tüm düğümler
    while Q != NULL do
        u ← Q içindeki en kısa uzaklığa sahip düğüm
        Q ← Q - u
        if uzaklık[u] == ∞ then          ▷ Kalan diğer düğümlere erişilemez
            break
        end if
        for all kDüğüm ∈ u.komşular do
            kisaYol ← uzaklık[u] + (u, kDüğüm) arası uzaklık
            if kisaYol < uzaklık[kDüğüm] then
                uzaklık[kDüğüm] ← kisaYol
                kaynak[kDüğüm] ← u
            end if
        end for
    end while
end function
```

Bu algoritma ile başlangıç düğümünden diğer tüm düğümlere olan uzaklık belirlenmiştir. Bu uzaklıklara göre her düğüme en kısa nerelerden gelindiği bilgileri de tutulmaktadır. Böylece rastgele seçilen bir düğüme giden en kısa yol bulunabilmektedir.

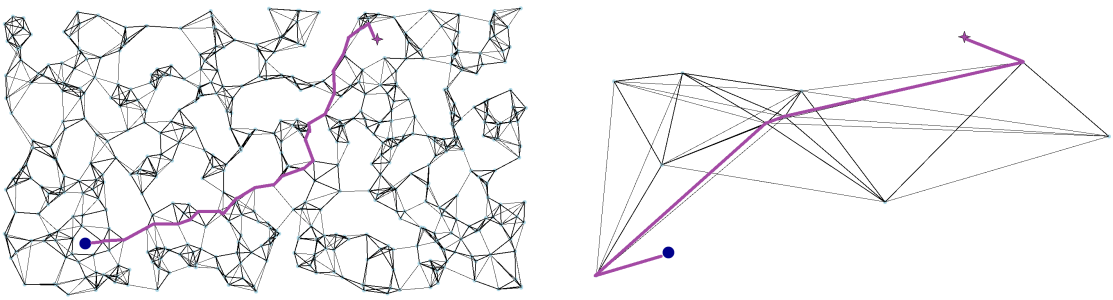
Bölüm 4

Bulgular

Bölüm 3 içerisinden anlatılan algoritmalara uygun olarak geliştirilen 2 boyutlu simülasyon ortamında, PRM yaklaşımlarının çalışmaları ile ilgili bazı çıkarımlar elde edilmiştir. Bu bölümde; simülasyon içerisinde bazı koşullar ve durumlar değiştirildiğinde elde edilen çıktılardaki iyi yönler ile ortaya çıkan bazı sorunlar ele alınacaktır.

4.1 Örnekleme Sayısının Etkisi

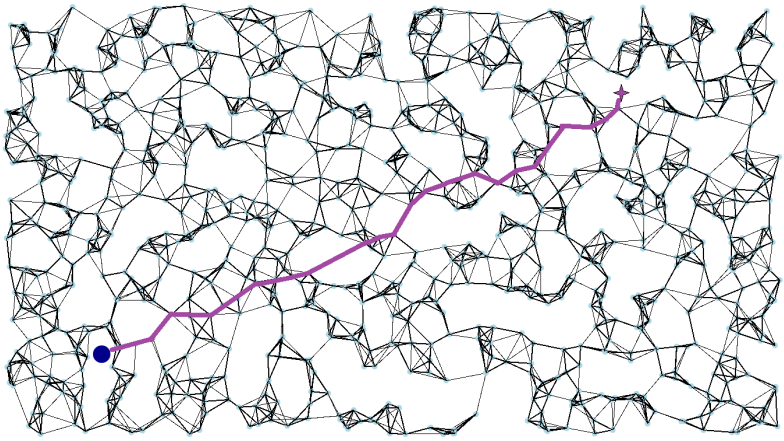
Olasılıksal yol haritaları düzenlenirken alınan örnek yapılandırma sayısı en kısa yol probleminin çözüme ulaşmasında önemli bir etkendir. Örneğin çevrede hiç engel bulunmayan iki simülasyon ortamını ele alalım (Şekil 4.1). Bu simülasyon ortamlarının ilki 500 örnek yapılandırma alınarak oluşturulmuş bir yol haritasını göstermektedir. İkincisi ise 10 örnek alınarak oluşturulmuş bir yol haritasını göstermektedir.



Şekil 4.1: Örnekleme Sayısının Etkisi

Alınan herbir yapılandırma çevresinde bulunan en fazla 5 komşu yapılandırmasına bağlanmıştır. Görüldüğü üzere yol planlayıcısı ilk durumda, ikinci duruma göre daha

iyi bir yol bulabilmiştir. 10 örnek alınan simülasyon ortamında hareket eden cisim başlangıç pozisyonundan çıkarak yol haritasına en yakın düğüme gitmeye çalışmıştır. Bu da yolun uzamasına neden olmuştur. Bu örnek, alınan örnek yapılandırma sayısı ile en iyi yola yaklaşımın doğru orantılı olduğunu göstermektedir.

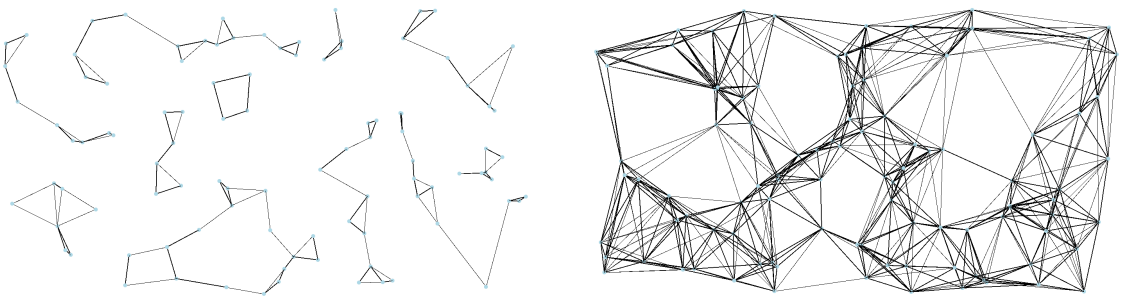


Şekil 4.2: 1000 örnek yapılandırmalı PRM

Şekil 4.2 diğer değişkenler sabit tutulup 1000 örnek alınarak oluşturulan bir yol haritasını göstermektedir. Burada hareketli cisim başlangıç ve hedef yapılandırması arasıda neredeyse düz bir hat üzerinde ilerleyerek en kısa yoldan gitmiştir. Fakat bu simülasyon örneğinde yol haritasının oluşturulması daha uzun bir zaman almıştır. Örnekleme sayısının artırılması çalışma zamanının uzamasına da neden olmuştur.

4.2 Komşu Sayısının Etkisi

Bir yol haritası oluşturulurken herhangi bir düğümden ulaşılabilecek komşu düğümlerin sayısının artması hem bağlanabilirlik testinde, hemde Dijkstra algoritmasında ek işlem yapılmasına neden olacaktır. Bu değişkenin etkilerini görebilmek için örnekleme sayısının sabit tutulduğu iki simülasyon ortamını ele alalım (Şekil 4.3).

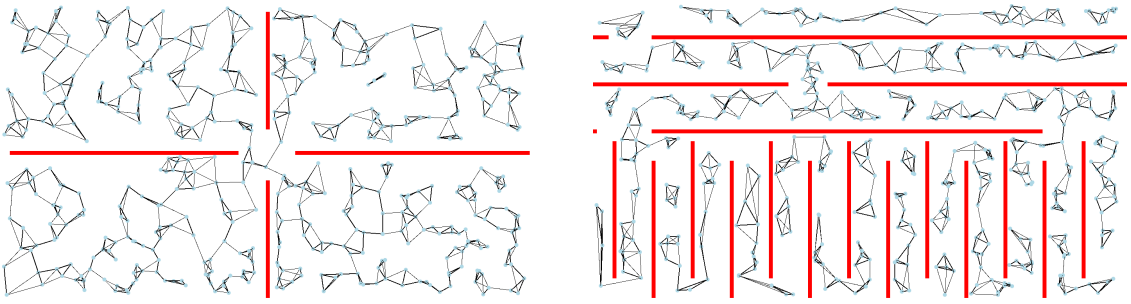


Şekil 4.3: Örnekleme Sayısının Etkisi

İlk örnekte bağlanabilir komşu yapılandırma sayısı 2 olarak ayarlanmıştır. Böylece her yapılandırma kendine yakın en fazla 2 yapılandırmaya bağlanmıştır. Bu yüzden çizge çok sayıda alt çizgelere ayrılmış ve bütünlük sağlanamamıştır. Bu durumda herhangi iki yapılandırma arasında erişim sağlanamayacaktır. İkinci örnekte örnekleme sayısı sabit tutularak bağlanabilir komşu sayısı 10'a çıkarılmıştır. Bu durum her bir düğüm için uygulanacak bağlanabilirlik testi sayısını 5 kat daha arttırmıştır. Bu yüzden yol haritasının oluşturulması daha uzun sürmüştür.

4.3 Engellerin Etkisi

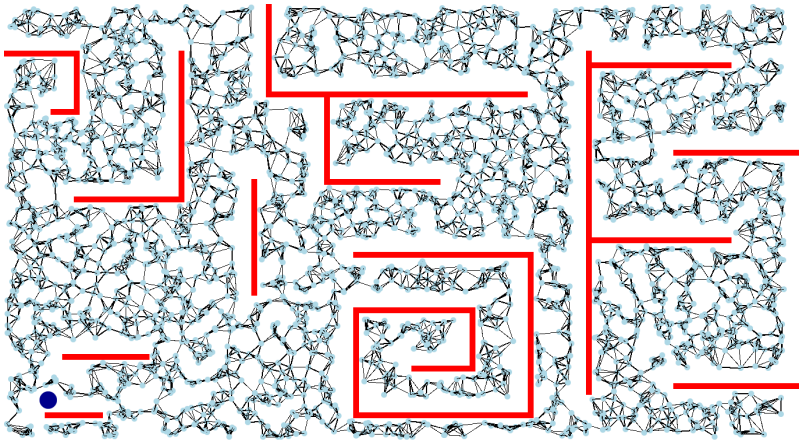
Ortamda bulunan engellerin artması Q_{free} yapılandırma kümesinin eleman sayısını azaltmaktadır. Bu yüzden robotun gidebileceği yapılandırmaların sayısı azalmakta ve hareketleri kısıtlanmaktadır. Örnekleme ve komşu yapılandırma sayısının aynı olduğu biri engel bulundurmayan ve diğeri bazı engellerden oluşan iki simülasyon ortamını ele alalım.



Şekil 4.4: Engellerin Etkisi

Şekil 4.4 içinde bulunan iki simülasyon örneği 500 örnek yapılandırma alınarak ve her yapılandırma 3 komşusuna bağlanarak oluşturulmuştur. Görüldüğü üzere ilk örnekte engellerin bulunma sıklığı az olduğundan, oluşturulan yol haritası üzerinde kopukluklar nadir görülmüştür. Q_{free} yapılandırma kümesinin neredeyse tamamına erişim sağlanmaktadır. İkinci örnekte ise engeller sık görüldüğünden alınan örneklerin birbirine bağlanmasında sıkıntılar yaşanmıştır. Engeller arasında bulunan dar boğaz geçişlerde kopukluklar yaşanmış ve bütün bir yol haritası oluşturulamamıştır.

Engellerin sık olduğu bu tür ortamlarda örnekleme sayısını artırmak tam bir yol haritası oluşturmak için iyi bir çözümdür fakat bağlanabilir komşu yapılandırma sayısı da doğru şekilde ayarlanmalıdır. Şekil 4.5 engellerin bulunduğu bir ortam için 3000 örnek yapılandırma alınarak oluşturulmuş bir yol haritasını göstermektedir. Her yapılandırma çevresinde bulunan 3 komşu yapılandırmasına bağlanmıştır.



Şekil 4.5: 3000 örnek yapılandırma alınarak oluşturulmuş yol haritası örneği

Yol haritasının oluşturulması sırasında alınan örneklerden bazıları $Q_{forbidden}$ yasaklı yapılandırmalar kümesinin elemanıdır. Bu elemanların seçiminde çakışma algılama testi negatif sonuç döndürmekte ve yerine yeni bir yapılandırma seçimi daha yapılmaktadır. Her yapılandırma seçim işlemi programa ek işlem yükü getirmekte ve çalışma zamanını kötü yönde etkilemektedir. Ayrıca 3000 örnek yapılandırmanın tüm komşularına bağlanabilirlik testi uygulanmaktadır. Bu test de örnekleme sayısına bağlı olarak çalışma zamanını etkilemektedir.

Bölüm 5

Tartışma ve Sonuç

Raporda 2 boyutlu yapılandırma uzaylarında olasılıksal yol haritalarının oluşturulmasıyla ilgili yöntemlerden ve algoritmalardan bahsedilmiştir. Bir yol haritasının bilgisayardaki tanımı çizgeyi oluştururken geliştirilen çakışma algılama ve bağlanabilirlik testlerinin algoritmalarına değinilmiş ve çalışma zamanları incelenmiştir. Ayrıca XNA framework ile geliştirilen bir simülasyon uygulamasından ekran alıntıları yapılarak anlatımlar görselleştirilmeye çalışılmıştır.

Yapılan simülasyon ortamında robot sistemi düz bir zemin üzerinde gezinmektedir ve birbirinden bağımsız hareket edebilen herhangi bir parçası bulunmamaktadır. Robotun dof sayısının az olduğu bu gibi durumlarda, PRM'den daha iyi çözüm üreten yöntemler[1] bulunmaktadır fakat bu özellikte meydana gelen artış ile bu yöntemlerin çözümleri yetersiz kalmaktadır.

Raporda verilen örnekler statik ortamlarda yol planlamaya giriş niteliğinde olup bundan sonra yapılması planlanan ve bu projenin devamı niteliğinde olan kısım hem yapılandırma uzayının boyutunu artırmak hem de robota bağımsız parçalar ekleyerek özgürlük derecesini arttırdıktan sonra bir yol planlayıcısı oluşturmaktır. Bu sayede PRM kullanımının amacı daha net biçimde açıklığa kavuşacaktır.

[1] Visibility Graph, GVD, GVG

Kaynakça

[1] H.M. Choset. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. Intelligent Robotics and Autonomous Agents Series. Mit Press, 2005.

[2] Microsoft. Xna framework class library, Aralık 2012.

[3] Jur Pieter van den Berg. Path planning in dynamic environments. Master's thesis, Utrecht University, Utrecht, The Netherlands, 2007.