İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL ROBOT SİMÜLATÖRLERİ VE İLERİ SEVİYELİ SİMÜLASYONLAR

YÜKSEK LİSANS TEZİ

Ender YOLAL

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

OCAK 2015

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL ROBOT SİMÜLATÖRLERİ VE İLERİ SEVİYELİ SİMÜLASYONLAR

YÜKSEK LİSANS TEZİ

Ender YOLAL
(504111113)

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı

Kontrol ve Otomasyon Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Hakan TEMELTAŞ

OCAK 2015

İTÜ, Fen Bilimleri Enstitüsü'nün 504111113 numaralı Yüksek Lisans Öğrencisi **Ender YOLAL**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"MOBİL ROBOT SİMÜLATÖRLERİ VE İLERİ SEVİYELİ SİMÜLASYONLAR"** başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Prof. Dr. Hakan TEMELTAŞ** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Osman Nuri UÇAN** İstanbul Aydın Üniversitesi	.....
	Yard. Doç. Dr. Murat Yeşiloğlu İstanbul Teknik Üniversitesi	.....

Teslim Tarihi : **15 Aralık 2014**
Savunma Tarihi : **21 Ocak 2015**

Aileme,

ÖNSÖZ

Öncelikle tez çalışmaları sırasında benden desteğini esirgemeyen hocam Sn. Prof. Dr. Hakan TEMELTAŞ'a teşekkürü bir borç bilirim.
Beni bugünlere getiren, manevi ve maddi destekleriyle daima yanımda olan ve her zaman destekleyen çok degerli aileme en içten teşekkürlerimi sunarım.
Son olarak yüksek lisans çalışmaları döneminde gerekli desteği sağlayarak süreci kolaylaştıran kurumum ASELSAN A.Ş.'deki yöneticilerime akademik hayata verdikleri destek için ayrıca teşekkür ederim.

Aralık 2014

Ender Yolal
(Kontrol Mühendisi)

İÇİNDEKİLER

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

MOBİL ROBOT SİMÜLATÖRLERİ VE İLERİ SEVİYELİ SİMÜLASYONLAR

ÖZET

Bu çalışma kapsamında öncelikle robot kavramının tarihsel gelişimi incelenmiştir. Robot kelimesi ilk defa Karel Capek tarafından bir tiyatro oyununda kullanılmış olsa da robot fikrini kökeni Eski Yunan'a kadar uzanmaktadır. Gezgin Robotlar incelenmeden önce gezginlik kavramı tanımlanmıştır. Eğer gezginlik yeteneği yeteneği konum değiştirebilme yeteneği olarak tanımlanırsa; çeşitli eyleyiciler yardımıyla konumunu değiştirme yeteneğini sahip robotlara gezgin robot denir. Gezgin robotun kendisine verilen görevi yerine getirmek için bulunduğu ortamı algılamak zorundadır. Bu amaçla kullandığı birimlere algılayıcı denir. Gezgin robotların kullandığı algılayıcı türleri Kızılötesi Algılayıcılar, Ultrasonik Algılayıcılar, Laser Algılayıcılar, LIDAR'lar ve Kameralar olarak sıralanmıştır. Bu algılayıcıları türlerinin birbirlerine göre avantajları, dezavantajları araştırılmıştır. Robotik Simülatörlerin ne oldukları ve Robotik Simülatör kullanımının bir gezgin robotun gerçeklenmesinin hangi aşamasında ihtiyaç duyulduğu incelenmiştir. Robotik Simülatörü, zamandan ve maliyettten tasarruf edilmesi amacıyla robotlara yüklenecek gömülü yazılım uygulamalarının, fiziksel olarak robottan bağımsız olarak test edilmesi amacıyla kullanılan bilgisayar programlarıdır. Piyasada birden fazla robotik simülatörü bulunmaktadır. Bu robot simülatörlerinden başlıcaları V-Rep, Webots, MobileSim olarak sıralanabilir. Robotik simülatörleri, gerçekçi simülasyon amacıyla fizik motorları içermektedirler. Kullanıcılar robotik simülatörleri, içerisindeki tasarım araçlarını kullanarak kendi robotularını oluşturabilirler. Oluşturdukları robota kendi ihtiyacına uygun olan algılayıcıları ve eyleyicilere entegre edebilirler. Gezgin robotlar kendilerine verilen görevi yerine getirmeleri için etkin yol planlama algoritmalarına ihtiyaç duyarlar. Gezgin robotların gelişimi süresince yol planlama sorununun çözümü için pek çok farklı çözüm önerilmiştir. Bu çözüm önerilerinin kendine özgü artı ve eksileri bulunmaktadır. Yol planlama algoritmalarını çevrimiçi ve çevrimdışı olmak üzere ikiye bölebiliriz. Gezgin robotun izleyeceği yol, gezgin robot hareket halinde iken gerçek zamanlı olarak belirleniyor ise bu yol planlama algoritması çevrimiçi olarak adlandırılır. Gezgin robotun izleyeceği yol, Bu tez kapsamındaki çalışmada robotun önüne çıkacak engelleri başlangıçta bilmediği varsayılmaktadır bu nedenle kullanılacak yol planlama algoritmaları çevrim içi yol planlama algoritmaları arasından seçilmeleri uygun olur. Bu nedenle tez kapsamında programlaması pratik olan Bug algoritmaları ve Potansiyel Alanlar Yaklaşımı incelenmiştir. Simülasyon çalışmaları kapsamında ise daha önce tarışılan yol planlama algoritmaları incelenmiş ve LIDAR simülasyonu ile Benzerlik Alan Metodu uygulaması gerçekleştirilmiştir.

MOBILE ROBOT SIMULATORS AND ADVANCED LEVEL SIMULATIONS

SUMMARY

In this thesis first of all historical background of robots.The word “robot” firstly used by Karel Capek in a theatre. The origin of the idea of robot came from Ancient Greek. Before analysing mobile robots one should analyse concept of mobility. If we define mobility as ability of position change; if a robot can change its position by its actuators, we can call the robot as mobile robot. A mobile robot should change its position with its own actuators in order to be defined as mobile robot. A mobile robot also perceives its environment in order to accomplish the tasks given to the robot. The tools that used by the mobile robot in order to perceive its environment is called sensors. The most used sensors used by mobile robots in order to perceive environment are infrared sensors, ultrasonic sensors, laser sensors, LIDAR and cameras. All this sensors have advantegeous and disadvantageous according to each other. In the next chapter what the robotic simulators are and what is the reason they are used for is investigated. Robotic simulators are computer programs that used for simulation of robots in order to dispose of time and cost. There are more than one robotic simulators on the market some of them are open source and some of them are licensed. ARS, Breve, EZPhysics, Gazebo, Klamp't, LpzRobots, miniBloq, Moby, MORSE, OpenHRP3, Simbad 3d Robot Simulator, SimRobot, Stage, STDR Simulator, UCHILSIM, UWSim are some of the open sourced robotic simulators and Actin, Marilou, Cogmation robotSim, COSIMIR, Microsoft Robotics Developer Studio, RoboLogix, SimplyCube, V-REP PRO, Visual Components, Webots, WorkCellSimulator, Workspace 5 are licensed robotic simulators. Robotic simulators use physics engines in order to simulate more realistic robotic simulations. In robotic simulators users can create their own environments, actuators, sensors and robots by using design tools in robotic simulators. The codes which are compiled in robotic simulators can be used in real robots as aibo, lego midstorm robots, kphera, koala, pionneer and nao. Robotic simulators let users to use different programming languages as C,C++, JAVA, Lua, PHYTON, Octave, Urbi and MATLAB.

In this thesis a robotic simulator in MATLAB environment has been developed. Main target of this simulator is simulation of path planning algorithms. In this simulator user can develop path planning algorithms with a differential wheeled robot based on the real robot developed in İTÜ Robotics Laboratuary. This robot is equipped with SICK LIDAR. Simulator is also simulates this SICK LIDAR. The Objectives of mobile robots mostly involve position change. Mobile robot shoulb change its position to complete its objective and mobile robots need path planning algoritms in order to achieve their objective. Different solutions for mobile robot path planning problem has been suggested till the present day. These suggestions have their positive and negative points. We can divide path planning algorithms in mobile robotics into two categories as online path planning algorithms and offline path planning algorithms.In online path planning algorithms it is assumed that the mobile robot is not aware of its environment before the mobile robot starts its movement. The path which is followed by the mobile robot, is planned at the same time as the mobile robot moves. In offline path planning algorithms it is assumed that the mobile robot is aware of its environment before the mobile robot starts its movement so the mobile robot plans all of the path from starting point to end point before any movement. Online path planning algorithms are more appropriate solutions than offline path planning algorithms in dynamic environments or environment of the robot is unknown before th mobile robot moves. In this thesis The environment of the mobile robot is not known before the mobile robot starts to move. That's why in this thesis using online path planning algorithms are more appropriate than using offline path planning algorithms. There are many different online path planning algorithms in mobile robot path planning literature. In this thesis Bug algorithms and potential field method are analysed. Bu algorithms are simply inspired from real life bugs. There are many differen bug algorithms which is divided from each other with small differences. Bug1, Bug2, Alg1, Alg2, Distbug, TangentBug, REV1, REV2, Angulus, CautiousBug algorithms are some of these bug algorithms. The Bug Algorithms makes three assumptions about the mobile robot. The mobile robot is a point object. the robot has perfect localization ability, the mobile robot has perfect sensors. First assumption is not possible when it comes to real life implementations but we can change this assumption as the mobile robot can move through every direction. Second assumption means that the mobile robot

knows its true position and orientation at any time of action. This assumption allows the mobile robot to determine termination of movement when arriving at the target if the target is reached. About third assumption it is known that there is no sensor can be called as perfect but we can change this assumption as the sensor which is used in the mobile robot is trustworthy. In the next chapter Bug1, Bug2, Alg1, Alg2 algorithms are analysed. The Bug1 algorithm is the first algorithm in the Bug algorithms. The Bug1 algorithm searches each obstacle which comes across the mobile robot for the point which is closest to the final point. When the point closest to the final point is determined. If the mobile robot can drive towards the final point. It drives towards the final point. If it cannot, the final point is said to be unreachable. A mobile robot uses Bug2 algorithm can leave encountered obstacle earlier due to the M-line. M-line is a hypothetical line between start point of the mobile robot anf-d final point. If the mobile robot encounters the M-line while following the obstacle boundary, the mobile robot ends following the obstacle and starts following the M-line. Bug2's faulty side is it can trace the same path twice. To correct this Alg1 Algorithm remembers previous points and uses these points to generate shorter paths. Alg2 Algorithm is an improved version of Alg1 algorithm without M-line concept. Another path planning algorithm which has been analysed in this thesis is Potentiel Field Method. Potentiel Field Method has both online and offline versions. The basic idea of Potentiel Field Method is to guide the robot by defining hypothetical attractive and hypothetical repulsive forces representing goal and obstacles respectively. It is assumed that the resultant of these attractive forces and repulsive forces guides the mobile robot to the final point while avoiding obstacles. Repulsive force vectors usually computed by distances and directions to abstacles around the mobile robot if this obstacles close to the mobile robot less than a predefined distance. In order to determine the attractive vector Potentiel Field Method need to have the true position of the mobile robot an the final point. There is known failure in Potential Field Method if attractive forces an repulsive forces are equal at a point the mobile robot can not pass this point and this point is called as Local Minimum Point. At the first part of the simulation chapter. Bug1, Alg1, Alg2 algorithms simulations made in the same map and compared. According to simulation results Alg2 algorithm finds the closest path. Second part of the simulation chapter Potential Field Method simulation made at four different maps. At two of them the mobile robot reached the final point and other two simulations robot

could not reach the final point because it stuck at the local minimum points at these maps. A at the last part of simulation chapter Likelihood Field Simulation is made using MobileSim robotic simulator and MATLAB.

1. GİRİŞ

Gezginlik yeteneği konum değiştirebilme yeteneği olarak tanımlanırsa; çeşitli eyleyiciler yardımıyla konumunu değiştirme yeteneğini sahip robotlara gezgin robot denir. Gezgin bir robot kendisine verilen görevi yerine getirmek için bulunduğu konum ile hedef konum arasında belirlediği yolda ilerlerken kendisini engelleyebilecek cisimleri farketmeli ve bu cisimlere çarpmadan yoluna devam edebilmelidir. Bu nedenle robotun bulunduğu ortam hakkında bilgi almasını sağlayan mesafe ve yakınlık algılayıcıları ile bu algılayıcılardan aldığı verileri işleyerek robotun görevini yerine getirebilmesini sağlayan yol planlama algortimaları gezgin robot uygulamaları için anahtar rol oynamaktadır. Çalışmanın ilk bölümünde gezgin robotlarda en sık kullanılan algılayıcılar; Kızılötesi algılayıcılar, Ultrasonik algılayıcılar, Laser Algılayıcılar, LIDAR'lar ve Kameralar anlatılmıştır. Çalışmanın ikinci bölümünde gezgin robot uygulamalarında zamandan ve maliyetten tasarruf edilmesi amacıyla robotları ve çalışma uzaylarını modelleyebilen; WEBOTS, V-Rep, MobileSim gibi robotik simülatörleri tanıtılmıştır. Çalışmanın üçüncü bölümünde gezgin robotun gezginlik görevini yerine getirebilmesi için ihtiyaç duyduğu yol planlama algoritmalarının farklı türü tanıtılmıştır. bunlar arasında çevrim içi ve çevrim dışı olarak [11] bir ayrıma gidilebilir. Pek çok farklı yol planlama algoritması vardır bu tez kapsamında incelenen Bug algoritmaları ilk defa Lumelsky ve Stepanov tarafından önerilmiştir [12]. Sankaranarayanan ve Vidyasagar tarafından tarafından önerilen [13] Alg algoritmaları ile Bug algoritmaları genişletilmiştir. Bu çalışma kapsamında incelenen diğer bir yol planlama metodu olan Potansiyel Alan Yaklaşımı ise Oussama Khatib tarafından önerilmiştir [15]. Boresntein ve Koren'in bu yöntemin yerel minimum sorununa değinmiştir [16]. Son bölümde ise incelenen yol planlama algoritmalarının MATLAB ortamında geliştirilen simülatör programı ile simülasyonu yapılmıştır.

2. ROBOT KAVRAMI VE ROBOTLARIN TARİHSEL GELİŞİMİ

2.1 Robot Kavramının Kökenleri

Robot kelimesi her ne kadar 1921 yılında Karel Capek'in RUR (Rossums Universal Robots, Rossum'un Evrensel Robotları) isimli tiyatro eserinde ilk kez kullanılmıştır. Robot kelimesi köken olarak Çekçe'de "robota" kelimesinden gelir. Robota, hizmet eden, mecburi hizmet, angarya,ve köle emeği anlamına gelir.

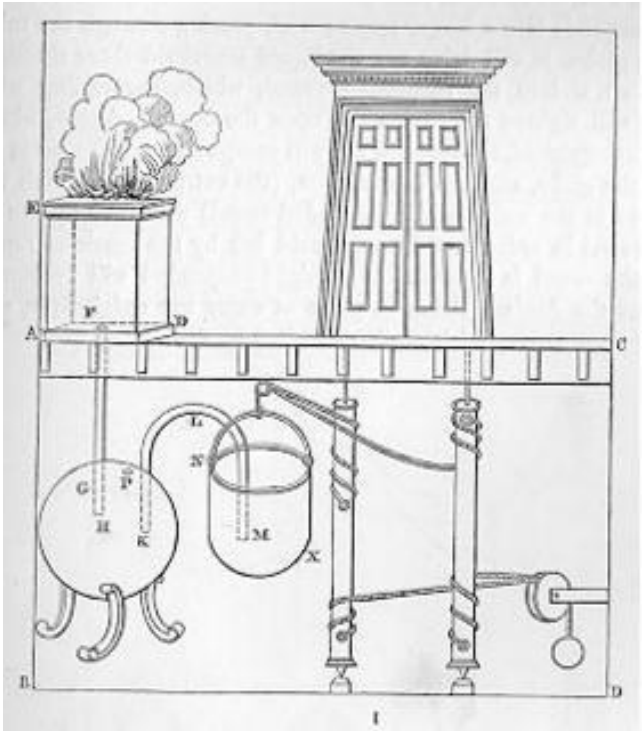
Birebir robot olarak tanımlanmamış olsada robot fikrinin kökleri 3000 yıl öncesine Eski Yunan'a kadar uzanır. Homeros'un eseri olan 'İlyada' da Demircilik Tanrısı Hephaistos tarafından yapılmış tekerlekleri altından olan ve kendiliğinden hareket eden üç ayaklı masalardan bahsedilmektedir. Bu üç ayaklı masalar Hephaistos'tan aldıkları emirleri yerine getiriyorlardı. Benzer şekilde eski yunan efsanesi Jason ve Argonotlar'da Talos adlı dev nöbetçinin Hephaistos tarafından Girit adasını korumak üzere programlandığı belirtilmiştir. [20, 21].

2.2 Robotların Tarihsel Gelişimi

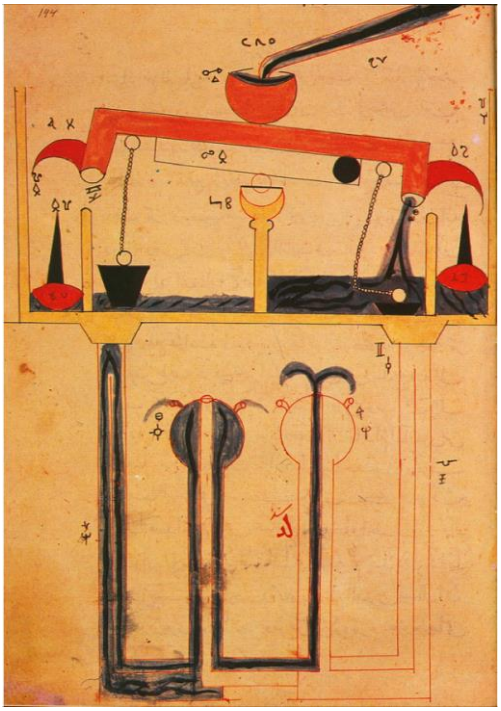
M.Ö. 350'de Aristo'nun "Eğer her araç kendi işini görebilseydi, insan eline ihtiyaç duymadan mekik kendi dokuyabilse, lir kendi çalabilseydi, yöneticilerin elemanlara ihtiyacı kalmazdı." sözü ile adeta otomasyon kavramını o tarihlerde tanımlamıştır. M.Ö. 100 civarında yaşadığı varsayılan iskenderiyeli Heron'un yaptığı otomatlar oldukça ünlüdür. İskenderiyeli Heron tapınak kapıları için otomatik açma düzenekleri, yine tapınaklar için kutsal su otomatı, tiyatrolarda kullanılmak üzere hareketli düzenekler icat etmiştir [1].

Artuklu Türklerinin Diyarbakır'da hüküm sürdüğü yıllarda yasayan El-Cezerî'nin (Ebü'l İz İbni İsmail İbni Rezzaz El Cezerî, M.S. 12 yy, tahmini 1136-1206) otomatlar üzerine yazdığı kitabı robotik konusunda çok sayıda ve zamanına göre ileri önerileri ve uygulamaları barındırmaktadır. El-Cezerî 32 yıl Artuklu sarayında mühendislik yapmış ve bu süre içerisinde otomatik olarak çalışan çok sayıda düzenek kurmuştur. Şekil 2.1'de El Cezeri tarafından tasarlanmış bir fıskiye düzeneği

verilmiştir. Havuz içinde iki fıskiye bulunmaktadır. Fıskiyelerden biri suyu inci çiçeği şeklinde, diğer fıskiye yay gibi fışkırtır. 15 dakika sonra suyu yay biçiminde fışkırtan inci çiçeği, inci çiçeği şeklinde fışkırtan yay gibi fışkırtır.



Şekil 2.1 : İskenderiyeli Heron'un otomatik açılan kapı düzeneği [22].



Şekil 2.2 : El Cezeri'nin zaman ayarlı fıskiye düzeneği [22].

Robot alanında kilometre taşları olarak nitelendirilebilecek gelişmeler aşağıda verilmiştir.

MÖ 270: Ctesibus isimli bir eski Yunan bilgini hareketli parçalardan oluşan organ ve su saatleri üretti.

MÖ 100: Otomatik açılan tapınak kapıları (İskenderiye).

1136 – 1206: El Cezeri' ye ait çesitli otomatik makinalar.

1800: Jacques de Vaucanson, Pierre & Henri-Louis Jacquet-Droz, Henri Maillerdet otomatik yazı yazan ve müzik enstrümanı çalan makinalar gelistirdiler.

1801: Joseph Jacquard ilk kez delikli kart kullanarak çalıştırılan otomatik dokuma makinası geliştirdi.

1818: Mary Shelley "Frankenstein" isimli hikayesinde yapay bir yasam seklini kullandı.

1830: Christopher Spencer mekanik kam denetimli otomatik bir torna tezgahı geliştirdi.

1892: Seward Babbitt sıcak metal parçaları fırından almak üzere motorlu tutucuya sahip robot düzenek tasarladı.

1920 – 1921: Çekoslovak Karel Capek' in yazdığı bir tiyatro oyununda ilk kez robot kelimesi kullanıldı. Yazar bu kelimeyi Çek lisanında "hizmet eden" anlamında kullanılan "robota" dan türetmiştir.

1938: DeVillbis firması için Willard Pollard ve Harold Roselund programlanabilir püskürtme boyama makinası geliştirdiler.

1940: MIT' de radar teknolojisinin geliştirilmesi, cisimleri insan etmeni olmadan algılanması konusunda en önemli adımlardan birisi oldu.

1940: Grey Walter ısıga yönelen ilk gezer robotları (machina speculatrix) üretti.

1941: Isaac Asimov "Robot" kelimesinden "Robotik" kelimesini türeterek ilk kez kullandı. Robotik, robot teknolojisi ile ilgili tüm alanları kapsayan bir tanım olarak kabul edilmektedir.

1942: Isaac Asimov "Runaround" isimli hikayesinde robotların üç yasasını yazdı.

1946: George Devol, genel amaçlı, manyetik kayıt yapabilen, ve tekrar çalıştırılabilen bir cihaz geliştirdi ve çeşitli makinalarda kullandı.

1946: J. Presper Eckert ve John Mauchly, Pennsyivania Üniversitesi' nde ilk elektronik bilgisayar olarak bilinen ENIAC isimli bilgisayarı gelistirdi. Whirlwind isimli bir başka bilgisayar M.I.T.'de ilk olarak bir bilimsel problemi çözdü.

1948: M.I.T.'den Norbert Wiener elektronik, mekanik ve biyolojik sistemlerin denetim ve iletişimini inceleyen, "Sibernetik" baslıklı kitabı yayınladı.

1951: Raymond Goertz, ABD Atom Enerjisi Komisyonu için uzaktan isletilen (teleoperated) bir kol tasarladı.

1954: George Devol programlanabilir genel amaçlı robotu tasarladı ve patent başvurusunu yaptı.

1956: G. Devol ve Joseph F. Engelberger "Unimation Inc." isimli dünyanın ilk robot firmasını kurmuşlardır.

1958: Satıs amaçlı ilk ticari robot üretildi.

1959: MIT' de servomekanizma laboratuarında robot kullanılarak bilgisayar destekli üretim amaçlı bir gösteri yapıldı.

1959: Planet firması ilk genel amaçlı ticari robotu pazarlamaya başladı. Atom Enerjisi Komisyonu için uzaktan isletilen (teleoperated) bir kol tasarladı.

1960: Harry Johnson ve Veljko Milenkovic tarafından tasarlanan Versatran isimli robot pazarlanmaya başladı. Unimation robotlarının adı Unimate Robot sistemleri olarak değiştirildi.

1962: General Motors ilk kez bir endüstriyel robotu (Unimate), sıcak parçalan kalıp döküm makinasından alarak istiflemek amacıyla üretim hattında kullanmaya başladı.
1963: Bilgisayar denetimli, altı eklemli ilk yapay kol (Rancho arm) geliştirildi.

1964: Dünyanın önde gelen bazı üniversite ve araştırma merkezlerinde (M.I.T. Stanford Araştırma Enstitüsü, Stanford Üniversitesi, Edinburgh Üniversitesi) ilk kez Yapay Zeka araştırmaları başladı ve laboratuarları açıldı.

1965: Dendral isimli ilk uzman sistem yazılımı geliştirildi.

1966: Nokta kaynağı yapan ilk robot üretildi

1967: Japonya ilk kez robot ithal ederek robot teknolojisini kullanmaya başladı.

1968: Stanford Araştırma Enstitüsü tarafından Shakey isimli ve görme yeteneği olan ilk gezer robot üretildi.

1968: Marvin Minsky tarafından on ayaklı ahtapot benzeri robot geliştirildi.

1970: Stanford Üniversitesi tarafından bir robot kol geliştirilmiş ve bu robot kol Stanford kolu adı ile araştırma projelerinde bir standart olarak yerleşti.

1973: Richard Hohn tarafından Cincinnati Milacron Corporation adına ilk mini bilgisayar denetimli robot geliştirildi. Gelistirilen robot T3 (The Tomorrow Tool) olarak adlandırıldı.

1974: Stanford kolunu geliştiren Prof. Scheinman, Vicarm Inc. isimli bir firma kurarak mini-bilgisayar kullanan robot kollarının pazarlamasına başladı.

1974: Dokunma ve basınç duyucuları kullanarak küçük parçaların montajını yapabilen ilk robot, üretim hattında kullanılmaya başladı.

1976: Viking 1 ve 2 uzay araçlarında robot kollar kullanıldı.

1977: ASEA isimli Avrupalı bir robot firması iki ayrı boyutta robot üretimine başladı.

1978: Puma isimli robot üretildi ve pazarlanmaya başladı.

1979: Stanford Cart isimli gezer robot, üzerine monte edilmiş bir kameradan alınan görüntüleri kullanarak engellerle dolu bir odayı engelleri asarak boydan boya geçti.

1990: ABD'de 12 dolaylarında, Japonya'daysa 40'dan fazla robot firması kuruldu.

1993: MIT'den Rodney A. Brooks bir insan gibi yetiştirilen ve eğitilen robot Cob'u yapmaya başladı.

1994: Dante II, Carnigie Mellon Üniversitesi'nde geliştirilen yürüyen robot Alaska'da aktif bir volkana kesif gezisi yaparak ve volkanik gaz örnekleri topladı.

1996: Honda, P-2 (prototype 2) yürüyen insansı robot dünyaya tanıtıldı.

1997: İlk yıllık robotlar arası futbol turnuvası "Robocup" Japonya'da düzenlendi.
1997: NASA'nın Pathfinder uzay aracı Mars'a indi ve "Sojourner" robotu Mars yüzeyinde kesif gezisi yaptı.

2000: RoboCup 2000'de üç insansı robot ilk defa karsılaştılar. Bu robotlar Batı Avustralya Üniversitesi'nden Johnny Walker, Japonya Aoyama Gakuin Üniversitesi'nden Mk-II ve Pino adlı insansı robotlardır.

2003: NASA'nın Mars'a robot gönderme çalışmaları [1].

3. GEZGİN ROBOTLAR

3.1 Gezgin Robot Nedir?

Gezginlik yeteneği konum değiştirebilme yeteneği olarak tanımlanırsa; çeşitli eyleyiciler yardımıyla konumunu değiştirme yeteneğini sahip robotlara gezgin robot denir.

Gezgin robotların gezginlik özelliklerini yerine getirebilmeleri için değişen durumlara göre hareketlerini değiştirme yeteneğine sahip olmaları gerekmektedir. Bunun için de öncelikle bulundukları ortamdan veri toplamalı ve bu verileri işlemeleri gerekmektedir. Robot bulunduğu ortamdan verileri algılayıcıları yardımıyla toplar sonra topladığı verileri işleyerek sonraki hareketine karar verir [2].

Gezgin robotlara verilecek bir görevi daha kolay gerçeklenmesi amacıyla alt görevlere ayırırsak bu alt görevlerden biri belirli bir noktadan belirli bir noktaya gitme alt görevi olacaktır. Gezgin robotlar yerine getirebilmeleri için bulundugu konumu bilmesi, bulunduğu ortamdaki engellerden sakınarak gitmesi gereken nokataya kendisi götürecek şekilde hareket etmesi gerekmektedir.

3.2 Gezgin Robotlarda Kullanılan Algılayıcı Türleri

Gezgin robotlara verilecek temel görevin gezgin olarak adlandırılmalarından yola çıkılarak belirli bir konumdan belirli bir konum gitmeleri olacağı açıktır. Gezgin bir robot kendisine verilen görevi yerine getirmek için bulunduğu konum ile hedef konum arasında belirlediği yolda ilerlerken kendisini engelleyebilecek cisimleri farketmeli ve bu cisimlere çarpmadan yoluna devam edebilmelidir. Bu nedenle robotun bulunduğu ortam hakkında bilgi almasını sağlayan mesafe ve yakınlık sensörleri gerçek zamanlı engelden sakınma özelliğine sahip gezgin robot uygulamaları için anahtar rol oynamaktadır.

Gezgin robotlar uygulamalarında konum ve cisim algılama problemine çözüm amacıyla aşağıdaki sensör tiplerinden biri veya birkaçı kullanılabilmektedir.

1) Kızıl Ötesi Algılayıcılar
2) Ultrasonik Algılayıcılar
3) Laser Algılayıcılar
4) LIDAR
5) Kamera

3.2.1 Kızılötesi algılayıcılar

Kızılötesi algılayıcılar özellikle endüstriyel uygulamalarda cisim algılama gibi işlemler için sıklıkla kullanılan maliyet etkin çözümlerdir. Kızılötesi algılayıcılar ile 10cm ila 80cm arası mesafeler ölçülebilir. Kızılötesi algılayıcılar kaynaktan gönderilen belirli bir frekanstaki ışığın, yansıtıcı aynadan geri yansıtılarak alıcı tarafından algılanması ile; ışık kaynağı ile yansıtıcı ayna arasındaki uzaklığı ölçerler.

Şekil 3.1 : SHARP-GP2Y0A21YK0F kızılötesi algılayıcı.

Kızılötesi algılayıcılarda kızılötesi ışığın yansımasından kaynaklı kısıtları vardır. Işık kaynağı ile yansıtıcı ayna arasındaki uzaklık kızılötesi algılayıcının performansını doğrudan etkiler. Işık kaynağı ile yansıtıcı ayna arasındaki uzaklık arttıkça kaynaktan yollanan ışığın dağılımı veya geri yansımaması nedeniyle sağlıklı mesafe ölçümü yapılamaz. Ayrıca yansıtıcı yüzey de kızılötesi algılayıcının performansını etkiler. Örneğin yansıtıcı yüzeyin parçacıklı olması durumunda kızılötesi ışık beklenenden fazla bir dağılım gösterebilir ve bu durumda algılayıcı sağlıklı ölçüm yapamaz veya yansıtıcı yüzeyin koyu renkli olması durumunda yansıtıcı yüzeyin ışığı emmesi nedeniyle ışık yansımayabilir [1].

3.2.2 Ultrasonik algılayıcılar

Ultrasonik algılayıcıların çalışma prensibinin doğada birebir karşılığı vardır. Yarasalar karmaşık ultrasonik algıma sistemleri sayesinde yaşamlarını geceleri avlanarak sürdürürler. Ultrasonik algılayıcılarda algılayıcının etrafındaki engellerin algılayıcıya olan uzaklığı algılayıcıdan gönderilen ses dalgasının, algılayıcıdan gönderildiği zaman ile engelden yansıyıp yeniden algılayıcıya dönmesi arasında geçen sürenin ölçülmesi ile bulunur. Bu süreye uçuş zamanı da denir. Bu işlem yapılırken ses hızının bildiğimiz değerinin değişmediği veya ihmal edilebilir bir seviyede değiştiği varsayılır.

Şekil 3.2 : HC-SR04 ultrasonik algılayıcı.

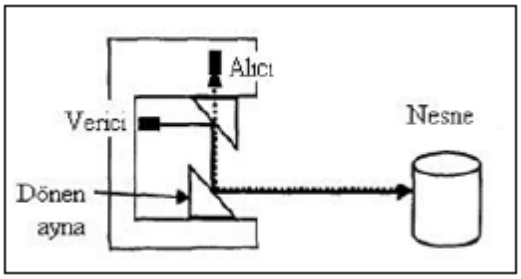
Ultrasonik algılayıcıların en büyük dezavantajı nesnelerin yüzeylerinden gerçekleşen yansımalar kaynaklıdır. Yansıma yönü, gelen ses dalgasının yüzeyle yaptığı açıya ve yansıma yapılan yüzeyin şekline bağlıdır. Geliş açısı küçüldükçe, ses dalgasının yansıma yapmadan yüzeyi sıyırıp geçmesi ihtimali ortaya çıkar ve bu şekilde yapılan mesafe ölçümü hatalı olur. Kaygan yüzeylerin yansıtıcı özellikleri sebebiyle bu duruma yol açmaları daha yüksek olasılıklıdır. Kaba yüzeylerde ise düzensiz yansımaların da algılayıcıya geri dönmesi yanlış mesafe ölçümlerine neden olur. Uzak mesafelerde ise yanlı yansımaların dönüş olasılığının artması sebebiyle ölçüm kesinliği daha düşük olur. Sayılan dezavantajlarına rağmen ultrasonik algılayıcılar lazer algılayıcılara göre daha ekonmik olmaları ve güneş ışığından etkilenmemeleri nedeniyle hareketli robot uygulamalarında iç mekan ve dış mekan uygulamalarında sıklıkla kullanılmaktadırlar [3].

3.2.3 Lazer algılayıcılar

Lazer algılayıcılar temelde optik algılayıcılar ile aynı şekilde çalışırlar. Aradaki en büyük fark ise lazer algılayıcıları optik algılayıcılardan çok daha büyük bir dalga boyundaki ışık ile çalışmalarıdır. Lazer algılayıcıdan çıkan ışın bir engele çarptıktan sonra algılayıcıya geri döner. Bu gidiş-geliş süresi ikiye bölünüp ışık hızı ile çarpılarak algılayıcı ile engel arası mesafe bulunur. Mesafe algılama konusunda lazer algılayıcıları optik algılayıcılardan çok daha etkindir. Lazer algılayıcılar ultrasonik algılayıcıların aksine kötü hava koşullarında (yağmur, sis vb) bile yüksek doğrulukla çalışırlar. Dezavantajlarını saymak gerekirse lazer algılayıcılar sadece belirli bir düzlemdeki cisimleri algılayabilirler. Bu düzlemin altındaki veya üstündeki cisimleri algılayamazlar. Güneş ışığı lazer algılayıcılar için bozucu etki yapar. Dış ortam uygulamalarında bu durumdan kaynaklı sorunlar oluşabilir. Ayrıca cam, şeffaf plastik gibi ışığın geçişine izin veren saydam cisimlerde lazer algılayıcı tarafından algılanmayabilir. Bu dezavantajları yanında aynı kullanım alanına hitap ettikleri optik algılayıcılar ve ultrasonik algılayıcılara göre daha pahalı olmaları en büyük dezavantajlarıdır.

3.2.4 LIDAR

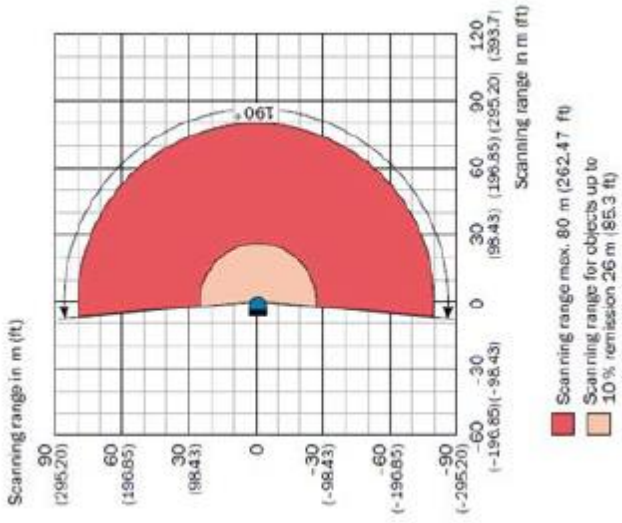
Light Detection and Ranging (LIDAR) sensörler lazer algılayıcılar ile benzer yapıda bir teknolojidir. LIDAR algılayıcının iç yapısı şöyle anlatılabilir; lazer kaynağından çıkan ışın, döner ayna yardımıyla algılayıcının etrafını 180-190 derece tarayacak şekilde gönderilir yansıyan ışın yine döner ayna yardımıyla algalanarak algılayıcı etrafındaki nesnelerin LIDAR'a uzaklıkları belirlenir.



Şekil 3.3 : LIDAR çalışma düzeneği.

LIDAR sensörler çok farklı amaçlar için kullanılabilir. Bunlar arasında havadaki partiküllerinin yoğunluğunun ölçülmesi, cisimlerin üç boyutlu modellerinin çıkarılarak bilgisayar ortamına aktarılması, üç boyutlu topografik haritaların elde

edilmesinde, otonom gezgin araçlarda engel tanıma ve ortam haritası oluşturma uygulamalarında kullanılırlar. Lazer algılayıcalr ile aynı prensipte çalıştıkları içinKötü hava koşullarından etkilenmezler. LIDAR sensörlerin çözünürlükleri kullanılan lazer kaynağı ve algılayıcıya bağlı olarak oldukça yüksektir. Örneğin SICK firmasına ait LMS 511 modeli LIDAR'ın çözünürlüğü 0.167 derece ve 80 metrede 30-40 mm civarındadır.Lazer algılayıcılar ile aynı prensipte çalıştıkları için lazer algılayıcıların dezavantajlarına da sahiptirler. Yolladıkları ışının yansıdığı yüzeyin ışığı soğurması LIDAR'ın performansını etkileyecektir. Örneğin LMS 511 modeli LIDAR'ın normal menzili 80m iken ışığı soğurma oranı yüksek olan siyah renkli cisimler için 26m'dir. Şekil 6'da LMS 511 modeli LIDAR'ın maksimum menzili ve %10'un altında yansıtma olan ortamdaki menzili görülmektedir. LIDAR algılayıcıların Kameralara göre avantajı LIDAR çıkış verisinin kamera çıkş verisinin aksine derinlik bilgisini de vermesidir [4].



Şekil 3.4 : LIDAR menzili.

3.2.5 Kamera

Gezgin robotlarda kullanılan diğer bir algılayıcıda kameralardır. Kameralar için gezgin robotlarda kullanlan en karmaşık algılayıcılardır diyebiliriz. Gezgin robotlarda çok farklı kamera sistemleri kullanılır. Kameradan gelen görüntünün işlenmesini içeren görüntü işleme algoritmalarının gerektirdiği yüksek işlem yükü gezgin robotun gerçek zamanlı olması isteniyorsa yüksek işlem hacmine sahip hızlı ve pahalı veri işleme sistemlerine ihtiyaç duyulmasına neden olur aksi takdirde gezgin robot gerçek zamanlı işlem yapma yeteneğinden feragat etmiş olacaktır.

Kameranın ortam algılayıcı olarak kullanılması durumunda gezgin robot bulunduğu ortam hakkında derinlik bilgisine sahip olmayacaktır. Bunun önüne geçmek için doğayı taklit edecek biçimde gezgin robot üzerine iki adet kamera yerleştirilerek derinlik algısı oluşturulan uygulamalar bulunmaktadır. Ancak bu durum görüntü işlemeden kaynaklı işlem ihtiyacını artırarak farklı bir sıkıntı oluşturmaktadır. Algılayıcı olarak kamera kullanımının bir diğer dezavantajı kameraların ortam ışığından ve kötü hava koşullarından olumsuz olarak etkilenmesidir.

4. ROBOTİK SİMÜLATÖRLERİ

Robotik Simülatörü, robotlara yüklenecek gömülü yazılım uygulamalarının, fiziksel olarak robottan bağımsız olarak test edilmesi amacıyla kullanılır. Bu sayede zamandan ve maliyettten tasarruf edilir.

Robotik simülatörlerinde uygulama geliştirilen robota bağlı olarak robotik simülatöründe geliştirilen uygulama değişikliğe ihtiyaç duyulmadan gerçek robota yüklenebilir. Örnek vermek gerekirse WEBOTS simülatöründe NAO robotlarına, MobileSim simülatöründe Pionneer robotlarına simülasyonda geliştirilen yazılım aktarılıp robot çalıştırılabilir.

Bazı robotik simülatörleri kullnıcının robotun rijid objeler ve ışık kaynakları ile çalışma uzayının benzerini oluşturmasına izin verirler. Daha sonra robot üzerindeki algılayıcılar yardımıyla(LIDAR, kamera vb.) bu çalışma uzayı ile etkileşime geçerek kendisine verilen görevi yerine getirmesi amacıyla programlanabilir. Robotik Simülatörlerin en popüler özelliği, robotları ve çalışma uzaylarını üç boyutlu olarak modelleyebilmeleridir. Bazı Robotik Simülatörleri robotun bu üç boyutlu hareketini daha gerçekçi hale getirebilmek için fizik motorlarına sahiptirler.

Robotik simülatörü kullanımı, simülatör içinde programlanması istenen robotun olup olmamasından bağımsız düşünülmelidir. Programlama denemelerinin Robotik Simülatör üzerinde yapılması robot üzerinde koşacak programın gerçek robotta sıkıntı yaratmayacak şekilde yazılmasını ve hatalardan arındırılmasını sağlar. Bu aşamanın atlanması robotun kendine veya çalışma uzayndaki nesnelere zarar vermesine yol açabilir.

Robotik simülatörleri, program içi uygulamaları (V-Rep, Webots, R-Station, Marilou) ile veya harici uygulamalar yardımıyla kullanıcının robot prototipi oluşturmasına izin verirler.Robotik simülatörleri, gerçekçi simülasyon amacıyla fizik motorları içermektedirler. Örneğin Gazebo, LpzRobots, Marilou, Webots; ODE fizik motorunu, Microsoft Robotics Studio; PhysX, fizik motorunu

kullanmaktadırlar.Robotik simülatörleri, kullanıcıya kolaylık olması amacıyla birden fazla programlama dili ile program geliştirilmesine izin verirler.

Piyasada birden fazla robotik simülatörü bulunmaktadır. Bu robot simülatörlerinden başlıcaları aşağıdaki gibi sıralanabilir.

Çizelge 4.1 : Robotik Simülatörleri.

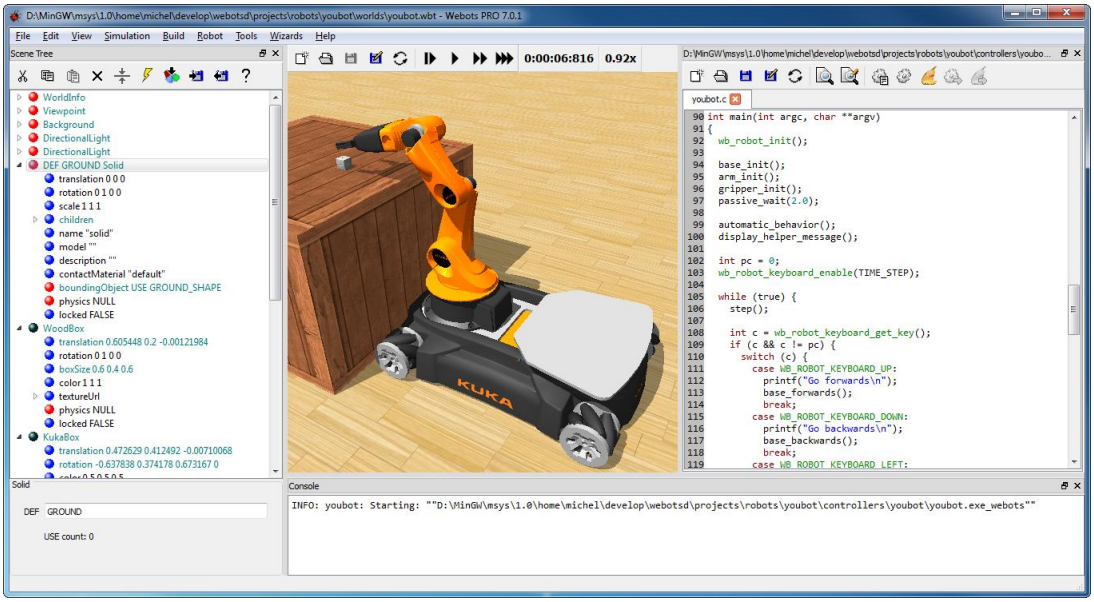
Açık Kaynak Kodlu Robotik Simülatörleri	Lisanslı Robotik Simülatörleri
ARS Breve EZPhysics Gazebo Klamp't: LpzRobots miniBloq Moby MORSE OpenHRP3 Simbad 3d Robot Simulator SimRobot Stage STDR Simulator . UCHILSIM V-Rep UWSim	Actin anyKode Marilou Cogmation robotSim COSIMIR Microsoft Robotics Developer Studio RoboLogix SimplyCube V-REP PRO Visual Components Webots WorkCellSimulator Workspace 5

4.1 Robotik Simülatör Örnekleri

4.1.1 Webots

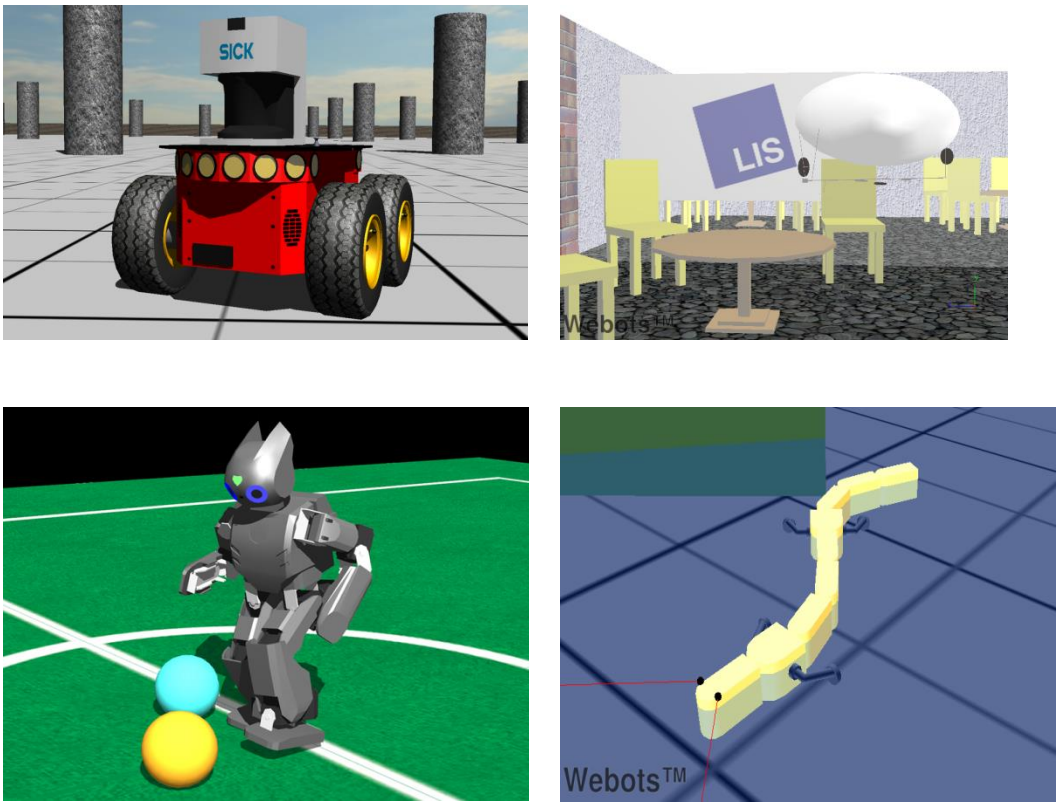
Webots, Cyberbotics şirketinin ürettiği lisanslı bir robotik simülatör programıdır. Khepera Simulator isimli açık kaynak kodlu programın geliştiricileri tarafından Khepera Simulator temel alınarak yazılmıştır. Webots'un özellikleri aşağıdaki gibi sıralanabilir.

- Webots yazılım kütüphanesi içerisinde gezgin robotların kullanacağı algılayıcıları ve eyleyicileri içermektedir. Webots kütüphanesinde uzaklık algılayıcılar, ışık sensörleri, kameralar, ivme ölçerler, dokunma/basınç sensörleri, GPS bulunmakta bu algılayıcılardan gelen verileri kullanıcı işleyebilmektedir.



Şekil 4.1 : Webots programlama ekranı ekran görüntüsü.

- Webots kullanılarak, webots kütüphanesinde bulunan Aibo, Lego Mindstorms, Kpehera, Koala ve Hemission gibi robotların gerçek versiyonları programlanabilir. Bu işlem sırasında webots ortamında çalıştırılan kodun değiştirilmesi gerekmemektedir.
- Webots kullanılarak her tipte gezgin robotun simülasyonu yapılabilir. Kütüphanesi içerisinde tekerlekli, iki bacaklı yürüyen, dört bacaklı yürüyen, yüzen ve uçan gezgin robot tiplerini içermektedir.
- Webots ,uygun fiziksel simülasyon için ODE(Açık Dinamik Motoru) kütüphanesini kullanmaktadır.
- Kullanıcının C, C++, JAVA, MATLAB veya TCP/IP üzerinden üçüncü parti bir yazılım ile robot programlamasına izin vermektedir.
- Webots, kullanıcın çalışmalarını görsel olarak sergileme isteğini düşünerek yapılan simülasyonun AVI ve MPEG formatında videolarını oluşturabilir.
- Webots kullanılarak çoklu robot sistemlerini programlanabilir.
- Kullanıcı Webots içerisindeki tasarım araçlarını kullanarak kendi robotunu oluşturabilir. Bu robota kendi ihtiyacına uygun olan algılayıcıları ve eyleyicilere entegre edebilir [23].



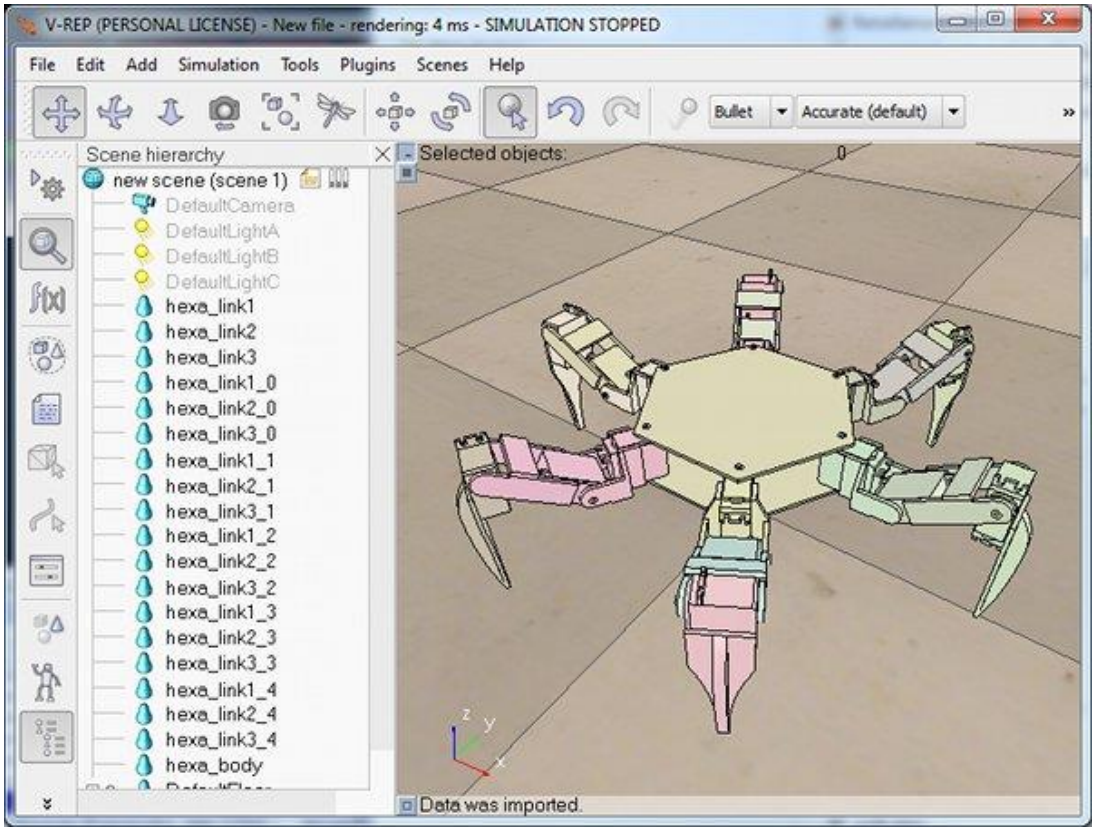
Şekil 4.2 : Webots simülasyon ekranı ekran görüntüleri.

4.1.2 V-Rep

V-Rep, Coppelia Robotics tarafından geliştirilen lisanslı bir robotik simülatör programıdır. V-Rep'in özellikleri aşağıdaki gibi sıralanabilir.

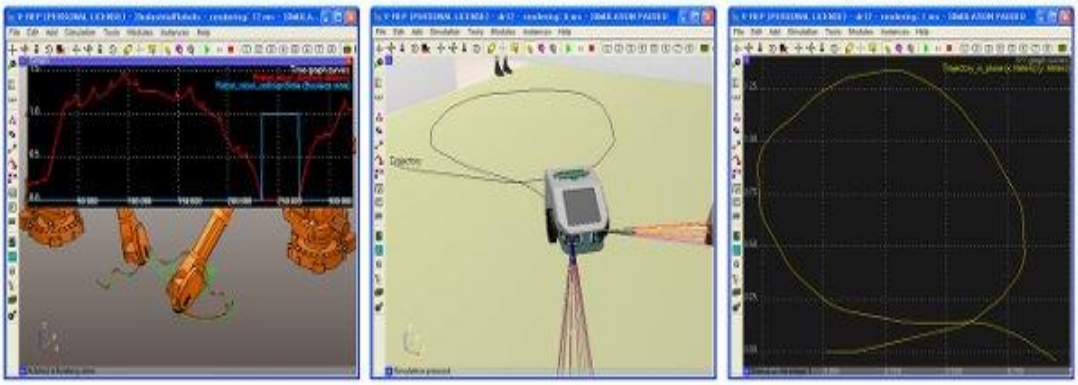
- V-Rep kullanıcının C, C++, Lua, PHYTON, JAVA, MATLAB, Octave ve Urbi ile robot programlamasına izin vermektedir.
- V-Rep kütüphanesinde uzaklık algılayıcılar, , kameralar, ivme ölçerler, dokunma/basınç sensörleri, bulunmakta bu algılayıcılardan gelen verileri kullanıcı işleyebilmektedir.
- V-Rep, kullanıcın çalışmalarını görsel olarak sergileme isteğini düşünerek yapılan simülasyonun videolarını oluşturabilir. Bu videolar V-Rep üzerinden oynatılabilir.
- Kullanıcı V-Rep içerisindeki tasarım araçlarını kullanarak kendi robotunu oluşturabilir. Bu robota kendi ihtiyacına uygun olan algılayıcıları ve eyleyicilere entegre edebilir.

- Kullanıcı V-Rep içerisindeki tasarım araçlarını kullanarak kendi algılayıcısını oluşturabilir. Bu algılayıcıyı istediği görevi yerine getirmesi için programlayıp diğer projelerinde kullanabilir.



Şekil 4.3 : V-Rep tasarım ekranı ekran görüntüsü.

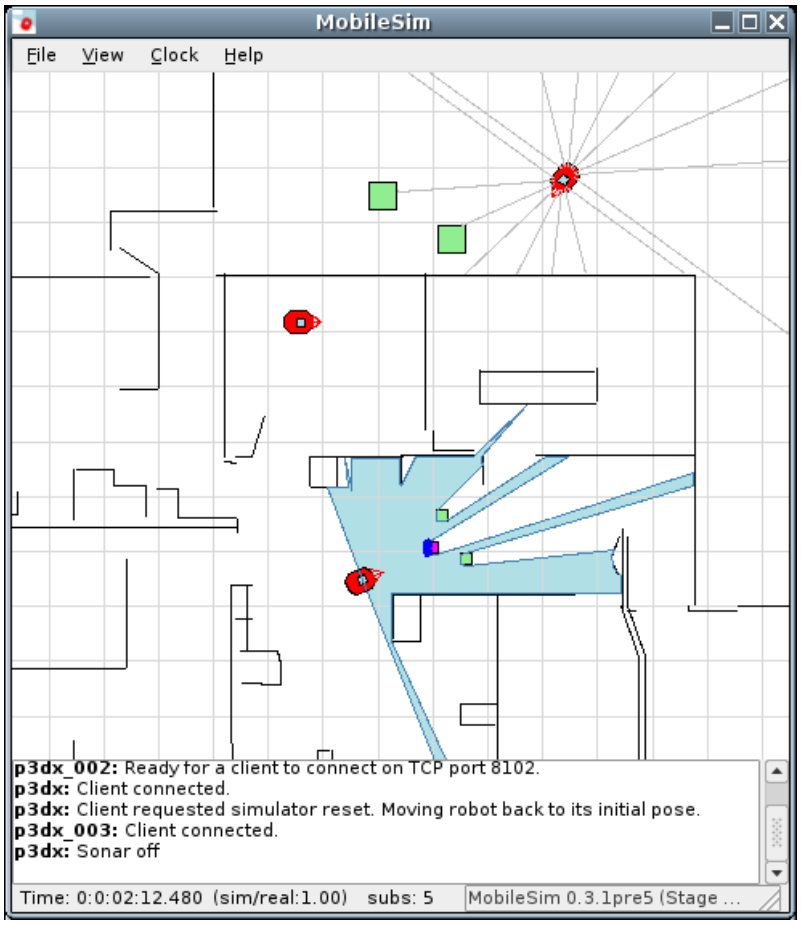
- V-Rep ,uygun fiziksel simülasyon için ODE, Bullet Physics ve Vortex Dynamics kütüphanelerini kullanabilmektedir. Kullanıcı yapacağı simülasyon için uygun gördüğü fizik motorunu kendi seçebilir.
- V-Rep hava veya su jetleri, jet motorları ve pervanelerin simülasyonunu destekleyebilir.
- Her türden robot mekanizması için ileri/geri kinematik hesaplamaları yapabilir. V-Rep içerisinde hazır olana ileri/geri kinematik algoritmaları kullanıcının kendi oluşturacağı robotların kontrolünde de kullanılabilir.
- V-Rep Simülasyon sırasında kullanıcının istediği veriakışını dışarıya verebilir. Kullanıcı isteği bağlı olarak bu veriler grafik olarak da simülasyon sırasında sergilenebilmektedir [24].



Şekil 4.4 : V-Rep simülasyon ekranı ekran görüntüsü.

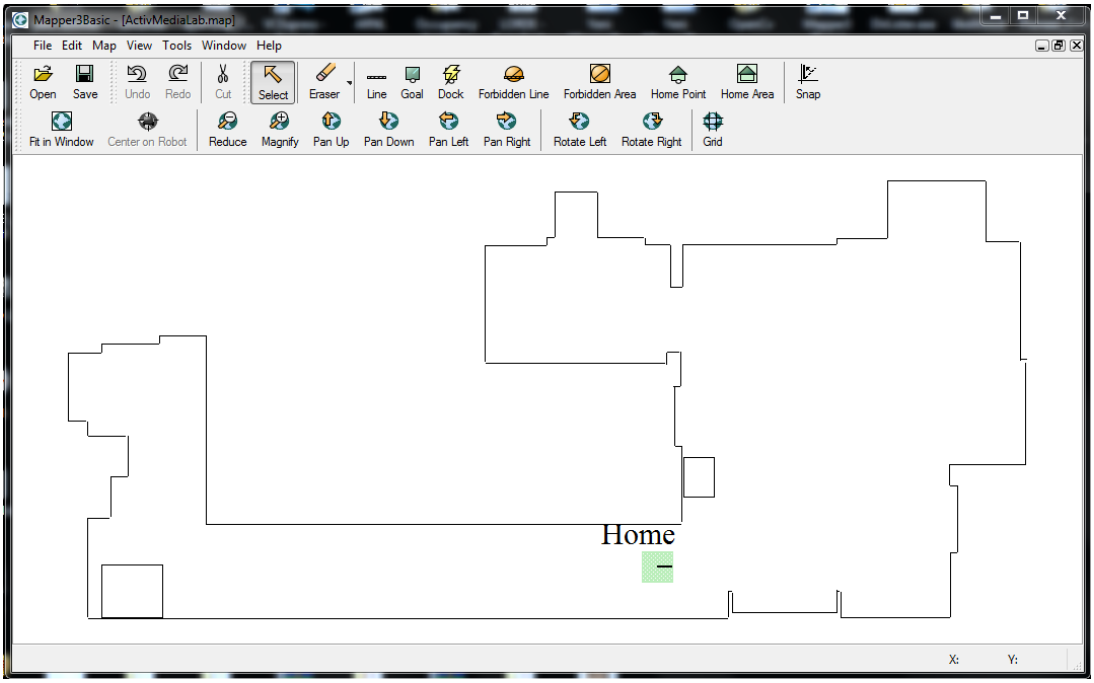
4.1.3 MobileSim

MobileSim, Pionneer robotlar için geliştirilen Adept Mobile Robots firmasına ait bir simülatördür. Pioneer robotlarda koşacak yazılımların robotlara yüklenmeden önce denenmesi amacıyla geliştirilmiştir. Bu sayede kullanıcı, robot çalışırken oluşması muhtemel hataları farkedip düzeltebilir.



Şekil 4.5 : MobileSim simülasyon ekranı ekran görüntüsü.

MobileSim, simülasyonun yapılacağı ortamı oluşturabilmek için .map uzantılı harita dosyasına ihtiyaç duyar. Ortam haritası Mapper3Basic ile oluşturulabilir. Kullanıcı, robotun kendisine verilen görevi yerine getireceği ortamın benzerini Mapper3Basic ile oluşturabilir. Mapper3Basic ile oluşturulan .map uzantılı dosya MobileSim ile açılır. Böylece robotun kullanıcının çaluşmasını istediği ortamda nasıl çalışacağı gözlemlenmiş olur.



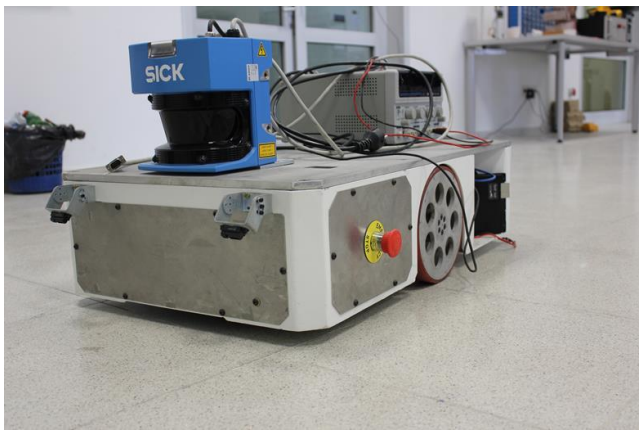
Şekil 4.6 : Mapper3Basic ekran görüntüsü.

MobileSim içerisindeki robotlarda ultrasonic algılayıcı, GPS, Cayro, LIDAR bulunmaktadır. MobileSim ortamında çoklu robot uygulamaları gerçekleştirilebilmektedir.

MobileSim Ortamında kod geliştirilirken ARIA(Gelişmiş Robotik ArayüzUygulamaları) Kütüphanesi kullanılır. ARIA, Windows ve Lınux işletim sistemlerinde çalışabilen, C++ tabanlı, açık kaynak kodlu bir kütüphanedir. ARIA kütüphanesi kullanılarak programlama yapmak için Windows işletim sistemleri kullanılıyorsa Ms Visual C++ .NET, Linux tabanlı işletim sistemleri kullanılıyorsa G++ 3.x derleyicileri kullanılır. Ayrıca kullanıcının kütaphaneye daha çabuk hakim olabilmesi amacıyla ARIA kütüphanesi içerisinde hazır uygulamalar ile gelmektedir [25].

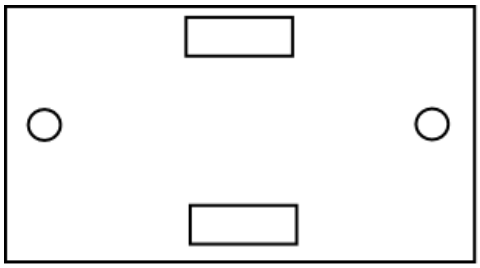
4.1.4 MATLAB ortamında hazırlanan simulator

Daha önce anlatılan simülatörlere ek olarak tez çalışmaları kapsamında MATLAB ortamında bir simülatör geliştirilmiştir. Geliştirilen simülatör yol planlama odaklıdır. Kullanıcı robotunun yerini belirleyebilmekte, robotunun yönlenmesi için hedef noktalar seçebilmekte ve robotun içerisinde koşmasını planladığı kodunu bu ortamda test edebilmektedir. Simülatörde kullanılan robot olarak Şekil 4.7'de verilen robotik laboratuarında bulunan diferansiyel sürüşlü robot temel alınmıştır.



Şekil 4.7 : LIDAR diferansiyel sürüşlü robot.

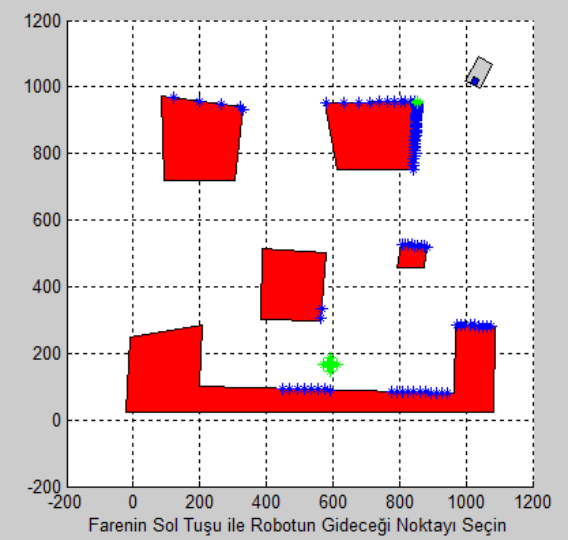
Şekilde de görüldü gibi robot üzerinde yol planlama algoritmalarında kullanılması amacıyla bir SICK LIDAR bulunmaktadır. Robotun eyleyicileri robotun her iki yanında bulunan Şekil4.8'de dikdörtgen ile gösterilmiş tekerleklere bağlıdır. Robotun dengesinin sağlanması için ön ve arkasında birer adet olmak üzere iki tane de sarhoş tekerlek bulunmaktadır.



Şekil 4.8 : Robotun tekerleklerinin temsili çizimi.

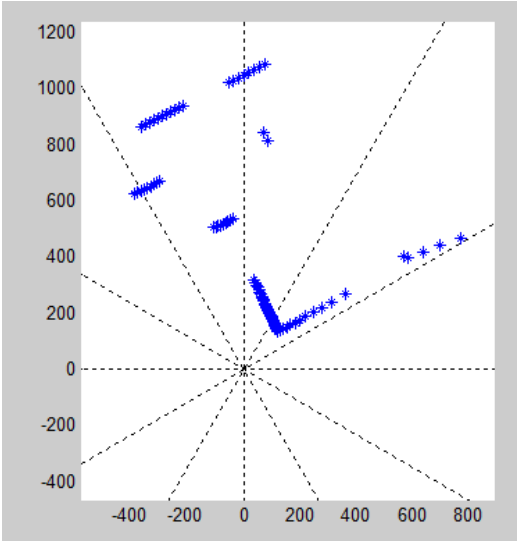
Simülatör oratamındaki robotta da temel alındığı robota benzer şekilde LIDAR simülatörü ile bulunduğu ortamdan veri toplayabilmektedir. Şekil 4.9'da MATLAB ortamında hazırlanan simülasyonun ekran görüntüsü görülmektedir. Kırmızı ile gösterilen bölgeler ortamdaki engeller, kırmızı bölgelerin üzerindeki mavi noktalar robot üzerinde bulunan LIDAR'ın algıladığı engel noktaları, yine kırmızı bölgelerin

üzerindeki yeşil noktalar LIDAR tarafından algılanan robota en yakın iki noktadır. Büyük yeşil çapraz işareti ile de kullanıcı tarafında seçilmiş robotun gitmesi istenen nokta gösterilmektedir.



Şekil 4.9 : MATLAB simülasyon ortamı.

Şekil 4.10 ise robot aynı pozisyonda iken robotun üzerinde bulunan LIDAR merkeze alındığı, LIDAR'ın gördüğü engellerin polar çizimidir.



Şekil 4.10 : LIDAR merkezli polar çizim.

5. YOL PLANLAMA ALGORİTMALARI

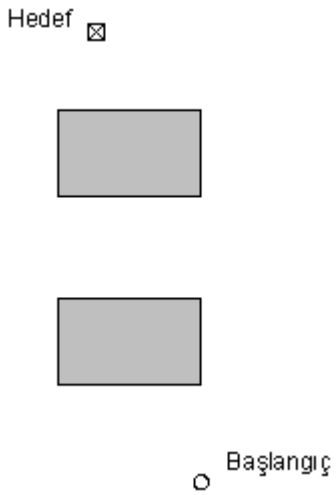
Gezgin robotların kullanımı birçok farklı alanda her geçen gün artmaktadır. Gezgin robotlar için etkin yol planlama robotik uygulamaların başarısı için zorunludur. Gezgin robotların gelişimi süresince yol planlama sorununun çözümü için pek çok farklı çözüm önerilmiştir. Bu çözüm önerilerinin kendine özgü artı ve eksileri bulunmaktadır.

Yol planlama algoritmalarını çevrimiçi ve çevrimdışı olmak üzere ikiye bölebiliriz. [5]. Gezgin robotun izleyeceği yol, gezgin robot hareket halinde iken gerçek zamanlı olarak belirleniyor ise bu yol planlama algoritması çevrimiçi olarak adlandırılır. Çevrim içi yol planlama algoritmalarında robot bulunduğu tüm haritasını bilmeye ihtiyaç duymaz. Sensörleri ile aldığı anlık ortam haritasına göre yol planlaması yapar.

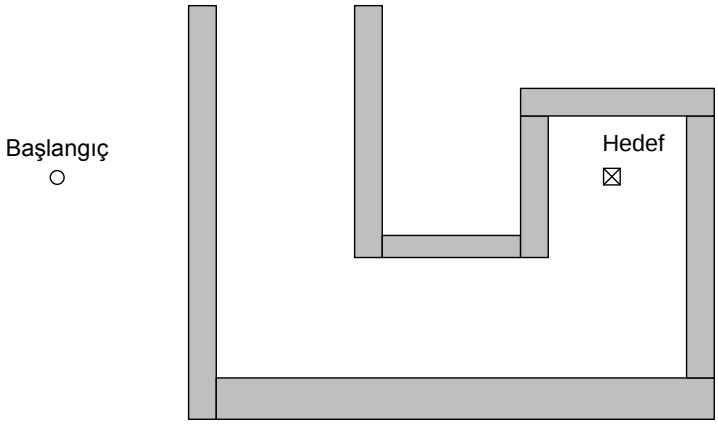
Çevrim dışı programlama ise, gezgin robot harekete başlamadan önce izleyeceği yolun belirlenmesinde kullanılır. Çevrim dışı yol planlama algoritmaları gezgin robotun bulunduğu ortamın tüm haritasına ihtiyaç duyarlar. Çevrim dışı yol planlama algoritmalarının planladıkları yollar için zaman değişkenini de içermeleri nedeniyle yörünge denmesi daha uygundur. Gezgin robotun belirlenen yörüngeyi takip ederek hedef noktaya ulaşması için ayrıca bir yörünge takip algoritmasına ihtiyacı vardır.

Yol planlama algortimasının seçimi robotun yol planlaması yapacağı ortam değerlendirilerek verilmelidir. Bu tez kapsamındaki çalışmada haritası daha önceden bilinmeyen bir ortamda bulunan bir gezgin robotun kartezyen koordinatları bilinen bir başlangıç noktasından yine kartezyen koordinatları bilinen bir hedef noktasına en kısa yoldan ve önüne çıkan engellere çarpmadan gidebilmesi için uygun olan yöntemler incelenecektir. Robotun önüne çıkacak engelleri başlangıçta bilmediği varsayılmaktadır bu nedenle kullanılacak yol planlama algoritmaları çevrim içi yol planlama algoritmarı arasından seçilmeleri uygun olur.Gezgin robotların hareketinde algılama ve karar verme adımı olarak iki temel adımdan söz edilebilir. Bu iki adım hereket esnasında arka arkaya sürekli olarak çalıştırılır. Algılama süreci, gezgin robotun algılayıcıların verdiği bilgilere dayanarak bulunduğu konumu, engellerin

bulunduğu konumları ve ulaşmak istendiği konumu karar verme algoritmasının kullanabileceği formata çevirmesi olarak açıklanabilir. Karar verme süreci ise, gezgin robotun hareketini gerçekleştirirken ortamdaki engellerden kaçınarak hedef noktaya ulaşma planıdır. İlerleyen bölümde yol planlama algoritmalarının Şekil 5.1'de verilen A ortamında ve Şekil 5.2'de verilen B ortamındaki davranışları incelenecektir.



Şekil 5.1 : A ortamı.



Şekil 5.2 : B ortamı.

5.1 Bug Algoritmaları

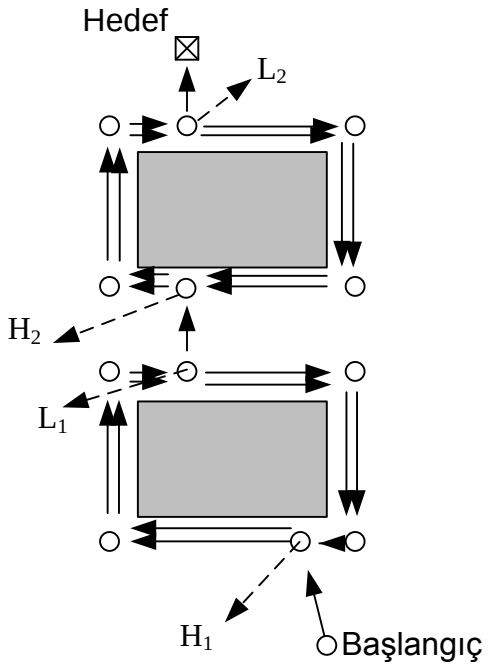
Çevrimiçi yol planlama algoritmalarından olan bug(böcek) algortimaları dokunma sensörüne sahip noktasal robotların yol planlama problemlerinin çözümü amacıyla böceklerden ilham alınarak geliştirilmişlerdir. Bug algoritmaları robotun bulunduğu ortamın tüm bilgisine ihtiyaç duymazlar. Hedef noktanın bulundukları konuma göre konumu bilgisi ve yerel ortam bilgisi ile hedef noktaya ulaşabilirler. Bug algoritması

ile çalışan bir robotun iki temel durum arasında geçiş yaparak hareket eder. Bu iki durum; Herhangi bir engel önlerine çıkmadığı sürece bir doğru üzerinde hedef noktaya doğru ilerlemek veya önlerine çıkan engeli, engelden ayrılma durumu oluşana kadar takip etmek olarak tanımlanabilir.

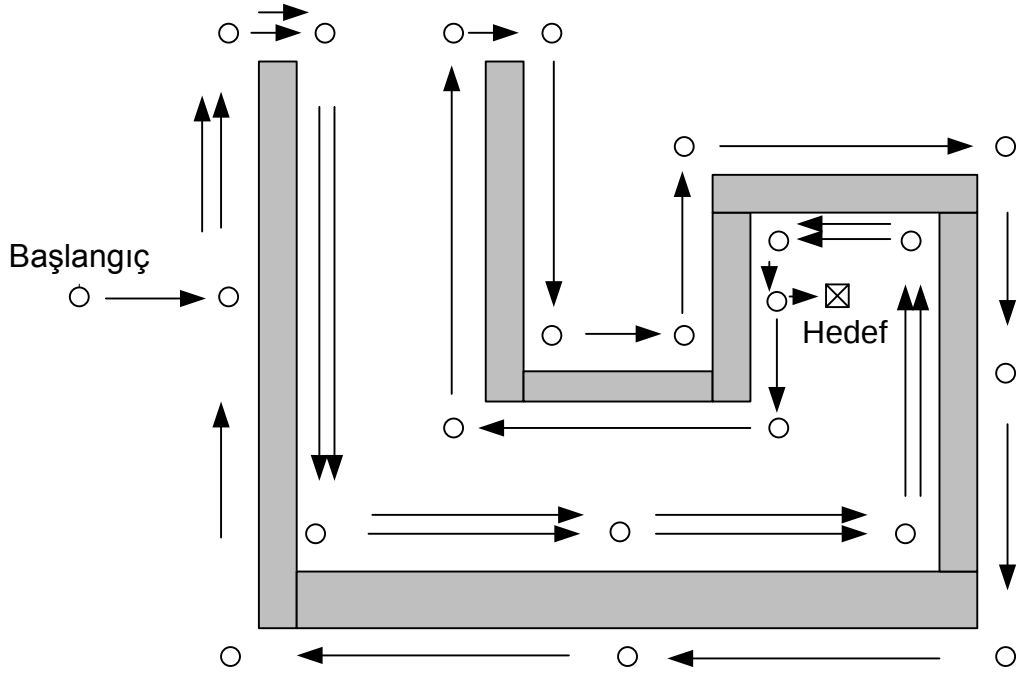
5.1.1 Bug1 algoritması

Bug1 algoritması ilk defa Lumelsky ve Stepanov tarafından önerilmiştir[6]. Bug 1 algoritması adımları aşağıdaki gibi tanımlanabilir. Şekil 5.3'te ve Şekil 5.4'te anlatılan Bug1 algoritmasının adımları aşağıda açıklanmıştır.

1) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar hedefe doğru bir doğru üzerinde ilerler.
   - Robot hedef ulaşırsa durur.
   - Robot ilerlediği doğru üzerinde bir engelle karşılaşır. Engelle karşılaştığı nokta H olarak adlandırılıp 2. Adıma geçilir.
2) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar, engel robota göre sağda kalacak şekilde engel takip eder. Bu adım devam ederken robotun hedef noktaya en yakın olduğu nokta L olarak adlandırılır.
   - Robot hedef ulaşırsa durur.
   - Robot H noktasına ulaşırsa 3. Adıma geçilir.
3) Öncelikle hedefin ulaşılabilir olup olmadığı kontrol edilir. Bunun için H noktası ile L noktasının aynı olup olmadığı kontrol edilir.
   - Eğer H ve L noktası aynı ise hedef ulaşılabilir değildir. Robot durur.
   - Eğer H ve L noktası aynı değilse robot yine engeli takip ederek L noktasına gider. L noktasına ulaştığında 1. Adıma geçilir.



Şekil 5.3 : A ortamında Bug1 algoritması.



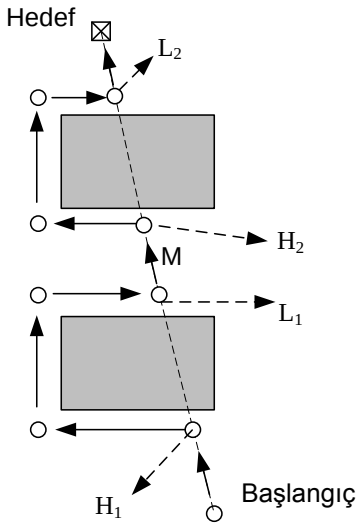
Şekil 5.4 : B ortamında Bug1 algoritması.

5.1.2 Bug2 algoritması

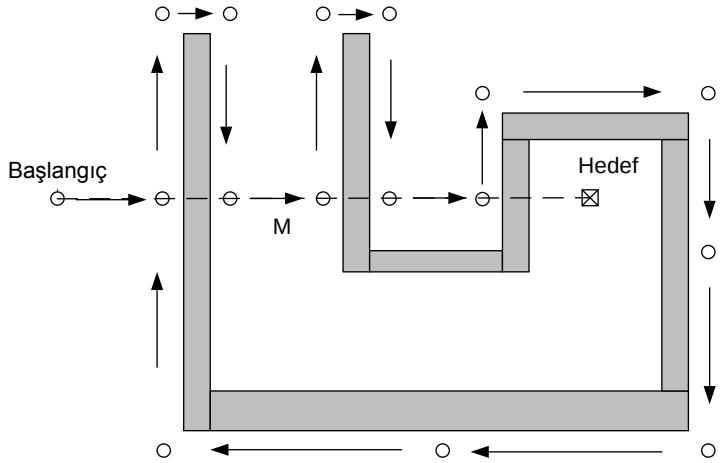
Bug2 algoritması Lumelsky ve Stepanov tarafından önerilmiştir[6]. Bug1 algoritması ile Bug2 algoritması arasındaki en büyük fark engel takibi durumundan ayrılma koşulu olarak getirilen M doğrusu kavramıdır. adımları aşağıdaki gibi tanımlanabilir.

Şekil 5.5'de anlatılan Bug2 algoritmasının adımları aşağıda açıklanmıştır. Şekil 5.6'da Hedefe ulaşamayan bir Bug2 algoritması süreci görülmektedir.

1) Başlangıç olarak robot başlangıç konumu ile hedef nokta arasında hayali bir M doğrusu tanımlanır. 2. Adıma geçilir
2) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar M doğrusu üzerinde hedefe doğru ilerler.
   - Robot hedef ulaşırsa durur.
   - Robot ilerlediği bir engelle karşılaşır. Engelle karşılaştığı nokta H olarak adlandırılıp 3. Adıma geçilir.
3) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar, engel robota göre sağda kalacak şekilde engeli takip eder.
   - Robot hedef ulaşırsa durur.
   - Robot engel takip ederken M doğrusu üzerindeki bir noktaya ulaşırsa bu nokta L noktası olarak adlandırılır. Robot engel takibini sonlandırır. 1. Adıma geçilir.
   - Robot H noktasına ulaşırsa hedef ulaşılabilir değildir Robot durur.



Şekil 5.5 : A ortamında Bug2 algoritması.



Şekil 5.6 : B ortamında Bug2 algoritması.

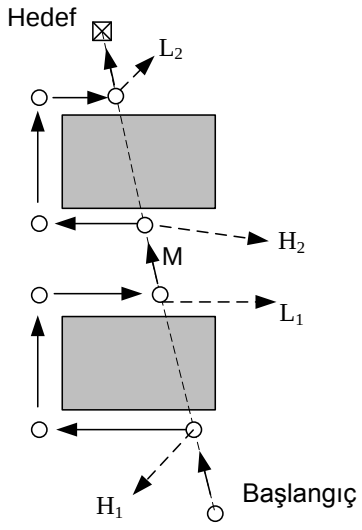
5.1.3 Alg1 algoritması

Alg1 algoritması, Bug2 algoritması geliştirilmiş hali olarak Sankaranarayanan ve Vidyasagar tarafından önerilmiştir[13]. Bug2 algoritmasının Şekil 5.6'da da görüldüğü gibi aynı yolu tekrar tekrar takip etme olasılığı bulunmaktadır. Bunun önüne geçmek için Alg bir algoritması önceki H engelle ile karşılaşma noktaları ve L engel takibinden ayrılma noktalarını hatırlar ve bu noktaları daha kısa bir yol üretmek için algoritması içerisinde kullanır. Alg1 algoritmasının adımları aşağıda açıklanmıştır. Şekil 5.7'de Alg1 algoritmasının A ortamında Bug 2 algoritması ile aynı yolu izlediği görülebilmektedir. Şekil 5.8'de B ortamında Bug2 algoritmasının başarısız olduğu ortamda Alg1 algoritmasının nasıl başarılı olduğu görülebilmektedir. Alg1 algoritması 3. Adımındaki değişiklik ile H noktasına engeli soluna alarak engel takibi yaparak hedef noktaya ulaşabilmiştir.

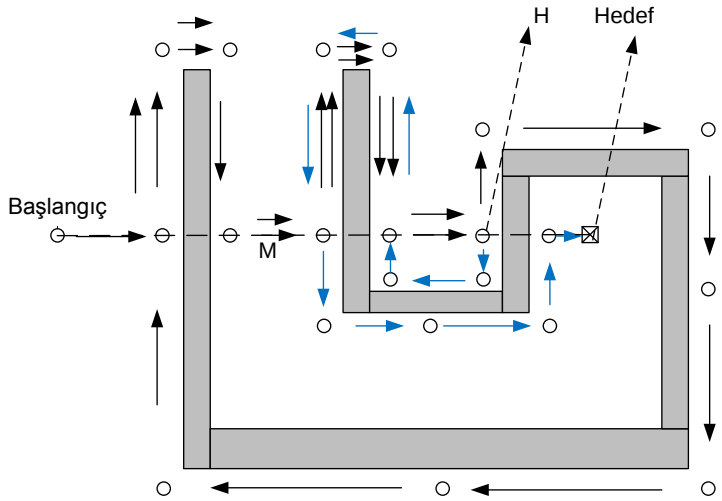
1) Başlangıç olarak robot başlangıç konumu ile hedef nokta arasında hayali bir M doğrusu tanımlanır. 2. Adıma geçilir
2) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar M doğrusu üzerinde hedefe doğru ilerler.
   - Robot hedef ulaşırsa durur.
   - Robot ilerlediği bir engelle karşılaşır. Engelle karşılaştığı nokta H olarak adlandırılıp 3. Adıma geçilir.

3) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar, engel robota göre sağda kalacak şekilde engeli takip eder.

- Robot hedef ulaşırsa durur.
- Aşağıdaki koşulları sağlayan bir Y noktası bulunması durumunda bu Y noktası L olarak tanımlanır. 2. Adıma gidilir.
  - ❖ Y noktası M doğrusu üzerindedir.
  - ❖ Y noktasının hedefe olan uzaklığı M doğrusu üzerindeki daha önce robotun bulunduğu tüm noktalardan daha azdır.
  - ❖ Y noktasında iken robot hedef noktaya doğru doğrusal olarak ilerleyebilmektedir.
- Robotun daha önceden tanımlanmış bir L veya H noktasına ulaşması durumunda robot H noktasına gidip engel robota göre solda kalacak şekilde engeli takip eder. Bu adım yeni bir L noktası bulunana kadar tekrarlanamaz.
- Robot bu adımdaki H noktasına ulaşırsa hedef ulaşılabilir değildir. Robot durur.



Şekil 5.7 : A ortamında Alg1 algoritması.



Şekil 5.8 : B ortamında Alg 1 algoritması.

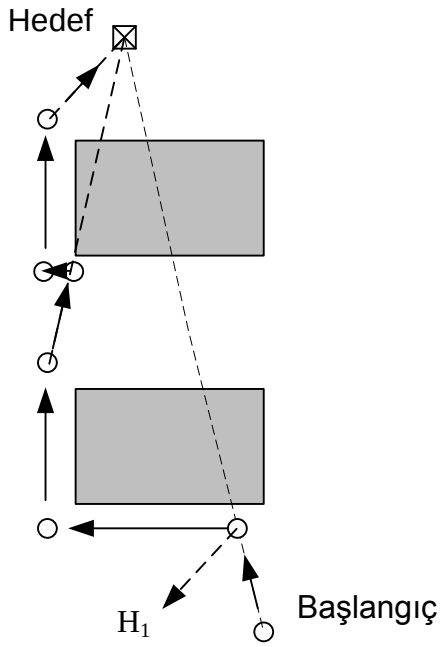
5.1.4 Alg2 algoritması

Alg2 algoritması, Alg1 algoritmasının geliştirilmiş hali olarak Sankaranarayanan ve Vidyasagar tarafından önerilmiştir[14]. Alg2 algoritması M doğrusu kavramı yerine yeni bir engelden ayrılma kavramı getirmiştir. Şekil 5.9'da ve Şekil 5.10'da A ve B ortamlarında Alg2 algoritmasının yeni engelden ayrılma yöntemi sayesinde engeli sola alarak engeli takip etme aşamasına Alg1 algoritmasına kıyasla daha kısa bir yol aldıktan sonra vardığı açıkça görülmektedir.

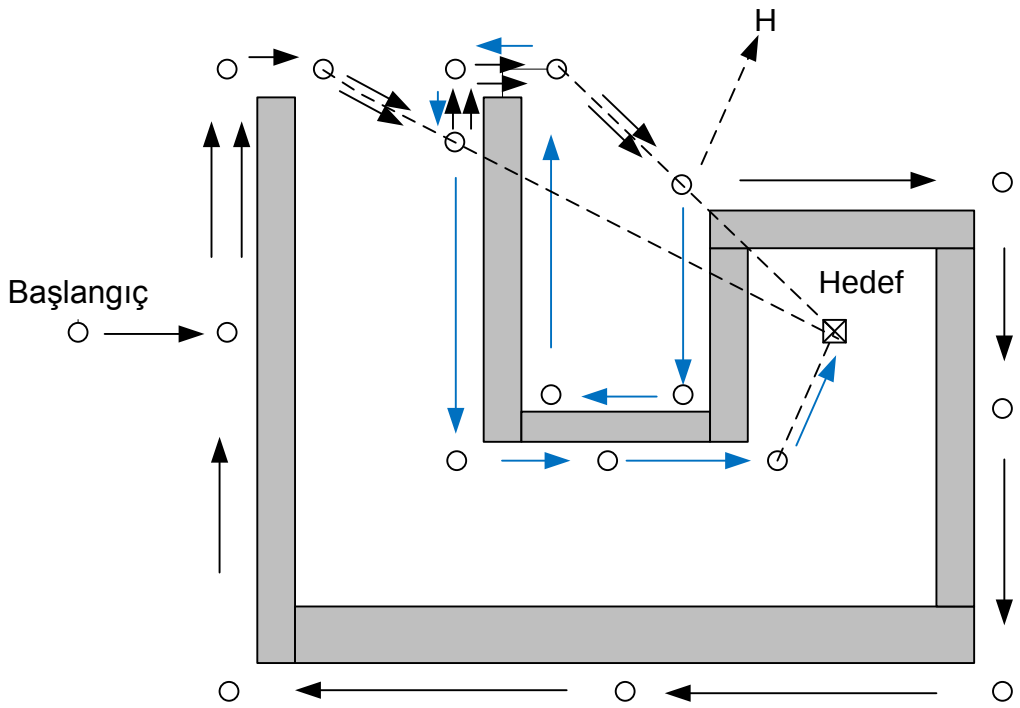
1) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar hedefe doğru bir doğru üzerinde ilerler. Hedefe doğru ilerlerken hedefe en yakın olduğu konumları Q noktası olarak tanımlar. Q robotun izlediği yol üzerinde robota en yakın olduğu noktadır.
   - Robot hedef ulaşırsa durur.
   - Robot ilerlediği bir engelle karşılaşır. Engelle karşılaştığı nokta H olarak adlandırılıp 2. Adıma geçilir.
2) Robot aşağıdaki durumlardan herhangi biri gerçekleşene kadar, engel robota göre sağda kalacak şekilde engeli takip eder. Bu arada Q noktasını hedefe yaklaştıkça yeniden tanımlar.
   - Robot hedef ulaşırsa durur.
   - Aşağıdaki koşulları sağlayan bir Y noktası bulunması durumunda bu Y noktası L olarak tanımlanır. 2. Adıma tekrar gidilir.

- ❖ Y noktası hedefe Q noktasında daha yakındır.
- ❖ Y noktasında iken robot hedef noktaya doğru doğrusal olarak ilerleyebilmektedir.

- Robotun daha önceden tanımlanmış bir L veya H noktasına ulaşması durumunda robot H noktasına gidip engel robota göre solda kalacak şekilde engeli takip eder. Bu adım yeni bir L noktası bulunana kadar tekrarlanamaz.
- Robot bu adımdaki H noktasına ulaşırsa hedef ulaşılabilir değildir. Robot durur.



Şekil 5.9 : A ortamında Alg2 algoritması.



Şekil 5.10 : B ortamında Alg2 algoritması.

5.1.5 Diğer bug algoritmaları

Bug algoritmalarının kolay programlanmaları ve geliştirilmeye açık olmaları nedenle pek çok farklı türü vardır. Buraya kadar gördüğümüz örneklerde olduğu gibi bu algoritmaların arasındaki fark engel takibinin bırakılacağı H noktasının belirlenmesi üzerinedir.

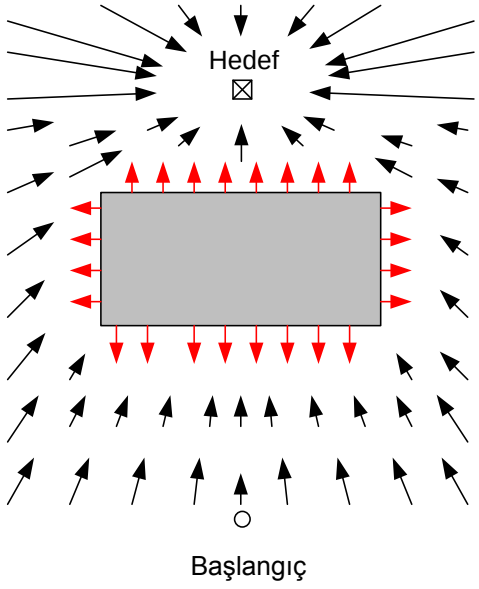
Literatürdeki Bug Algoritmalarının en bilinenlerini sıralamak gerekirse Distbug; Kamon and Rivlin [12] tarafından önerilen Alg2 algoritması içerisine H noktasının bulunmasında robotun aldığı yolun uzunluğunu dahil etmiştir, TangentBug; Kamon, Rivlin ve Rimon tarafında önerilmiştir [13]. Robotun bulunduğu ortamdaki köşe noktaları belirleme ve buna göre yol planlama özelliği ile diğer bug algoritmalarından ayrılır. REV1 algoritması, Alg1 algoritması temel alınarak Horiuchi and Noborio tarafından önerilmiştir [14]. Alg1 algoritmasından engel takip yönünü her seferinde değiştirmesi ile ayrılır. REV2 algoritması, Alg2 algoritması temel alınarak Horiuchi and Noborio tarafından önerilmiştir [15]. Alg2 algoritmasından engel takip yönünü her seferinde değiştirmesi ile ayrılır. Angulus algoritması Lumelsky and Tiwari tarafından önerilmiştir [16]. Engelden ayrılma koşulunu robotun hız vektörüne ve robotun doğrultusunagöre belirler. Magid and Rivlin tarafında geliştirilen CautiousBug [17], engel takibi sırasında alınan yolun

uzunluğunu değerlendirerek engel takibi yönünü değiştirebilir. Optim-Bug Kriechbaum tarafından önerilmiştir [18]. Diğer Bug algoritmalarından farklı olarak robot ilerledikçe içinde bulunduğu ortamın haritasını oluştururlar. Buna rağmen çevrim içi yol planlama yapar. Engel takibi yerine her çevrim sonucu en kısa yol olarak belirledikleri yönde ilerler.

5.2 Potansiyel Alanlar Yaklaşımı

Potansiyel Alan Yaklaşımı gezgin robotların yol bulma problemlerinin çözümünde sıkça kullanılan bir yöntemdir. Potansiyel Alan Yaklaşımı Oussama Khatib tarafından önerilmiştir[9]. Çevrim içi ve çevrim dışı uygulamaları mevcuttur.

Potansiyel Alan Yaklaşımının temel kavramı gezgin robotun yapay bir potansiyel alan ile dolu bir ortamda bulunduğunu kabul etmektir. Bir başlangıç noktası, bir bitiş noktası ve engellerden oluştuğu varsayılan ortamda hedef nokta robot üzerinde çekici bir vektör oluşturarak robotun hedefe çekilmesini, ortamda bulunan engeller ise robot üzerinde itici vektör oluşturarak robotun ilgili engelelrden uzaklaştırılmasına sağlamaktadır. Şekil 5.11'de robota etkiyen bu vektörler gösterilmiştir. Robot bu çekici ve itici vektörlerin bileşke vektörü yönünde ve hızında hareket etmektedir. Potansiyel Alan Yaklaşımının varsayımı bu bileşke vektörün etkisindeki robotun engellere çarpmadan en kısa yoldan hedef noktaya ulaşacağıdır.



Şekil 5.11 : Robota etkiyen potansiyel alan vektörleri.

Potansiyel alan yönteminde çekici potansiyel alan fonksiyonu:

$$U_{ç}(q) = \frac{1}{2}\zeta\rho^{m}(q, q_{hedef}) \tag{4.1}$$

Burada q robotun [X, Y] ^{T} ile tanımlana mevcut pozisyonudur. q_{hedef} hedef noktanın pozisyonudur. ξ ise pozitif bir çarpandır. Kullanıcıya bağlı olarak değiştirilebilir. Potansiyel Alan Yönteminde çekici kuvvet, çekici potansiyel fonksiyonun jakobieni alınarak aşağıdaki gibi bulunur:

$$F_{ç} = \xi(q_{hedef} - q) \tag{4.2}$$

Potansiyel Alan Yönteminde itici potansiyel alan fonksiyonu:

$$U_{i}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{\rho(q, q_{engel})} - \frac{1}{q_0})^2 & eğer\, \rho(q, q_{engel}) < p_0 \\ 0 & eğer\, \rho(q, q_{engel}) > p_0 \end{cases} \tag{4.3}$$

İtici kuvvet, itici potansiyel fonksiyonun jakobieni alınarak aşağıdaki gibi bulunur:

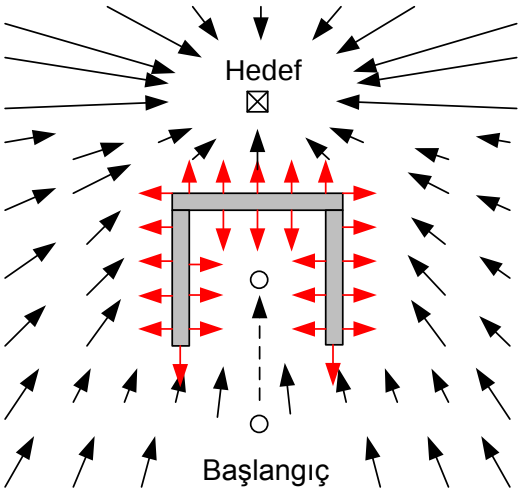
$$F_{i}(q) = \begin{cases} \frac{1}{\rho^2(q, q_{engel})} J(\rho(q, q_{engel})) & \rho(q, q_{engel}) < p_0 \\ 0 & \rho(q, q_{engel}) > p_0 \end{cases} \tag{4.4}$$

Şekil 5.11'den ve denklem 4.4'ten de anlaşılacağı gibi çekici vektör robotun çalışma uzayının her noktasında büyüklüğü değişmekle birlikte etki n iken itici vektör robota engele belirli bir mesafeden daha yakın ise etkin olmaktadır.

5.2.1 Yerel minimum problemi

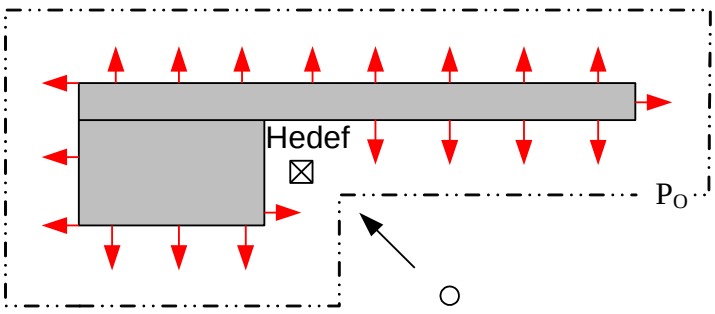
Potansiye Alan Yaklaşımında çekici ve itici vektörlerin bileşke vektörünün robot hedef noktaya yaklaştıkça 0'a yaklaşması robot hedef noktaya ulaştığında 0 olacağını varsaymaktadır. Bu sayede robot hedef noktaya yaklaştıkça hızı azalacak ve hedef noktaya ulaştığında duracaktır. Fakat [10] Borenstein ve Koren'in belirttiği gibi çalışma uzayında birden fazla yerel minimum potansiyel alana sahip nokta bulunabilir ve bu noktalardan herhangi birine takılan bir robot etrafındaki diğer noktaların kendinden daha yüksek potansiyele sahip olması nedeniyle hareket

edemez. Şekil 5.12'de böyle bir yerel minimum noktasına yakalanmış bir robotun hareketi görülebilmektedir.



Şekil 5.12 : Yerel minimum problemi.

Yerel Minimum probleminin yanında Ge S.S. and Cui Y.J.'nin belirttiği gibi ulaşılamayan hedef noktaları problemi bulunmaktadır[11]. Ulaşılamayan hedef noktaları problemi şöyle tanımlanabilir. Eğer bir hedef noktası etrafındaki engellere fazla yakınsa itici potansiyel alanlar nedeniyle robot bu hedef noktaya, Şekil 5.13'de gösterildiği gibi bir tehlike bulunmamasına rağmen ulaşamaz.



Şekil 5.13 : Ulaşılamayan hedef noktası problemi.

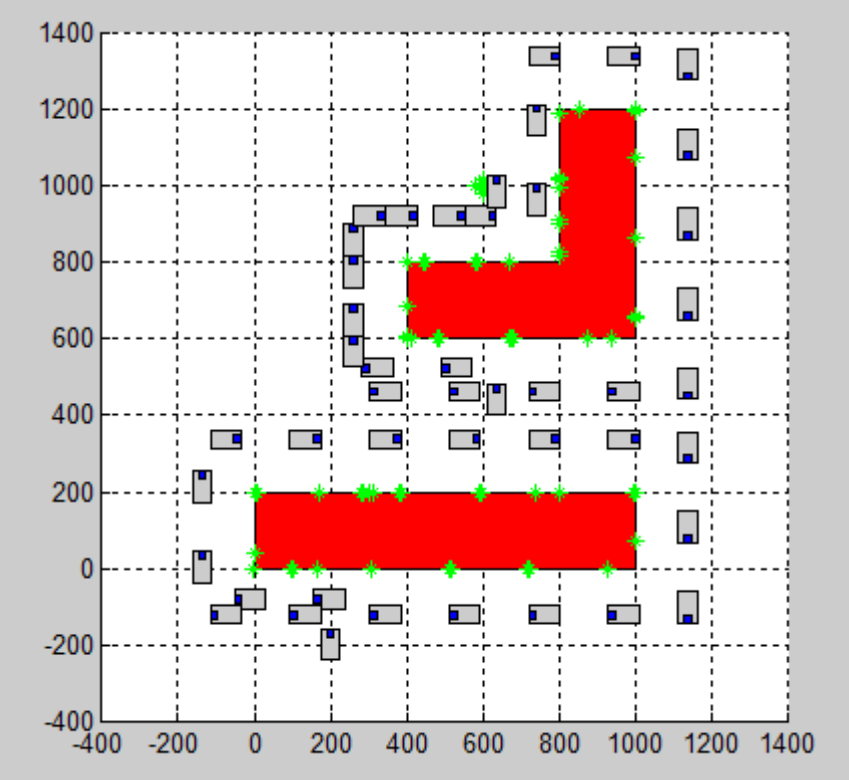
Ulaşılamayan Hedef Noktası Probleminin sebebi itici potansiyel alandan gelen kuvvetin, robot engelin yakınında iken hedefin ürettiği çekici kuvvetten daha fazla olmasıdır, Bu yüzden robot belirtilen hedefe ulaşamamaktadır. Bu problemin çözülebilmesi için, toplam potansiyel alan kuvvetinin hedef noktasında minimum olacak şekilde ayarlanmalıdır.

6. SİMÜLASYON ÇALIŞMALARI

6.1 Bug Algoritmalarının Karşılaştırılması

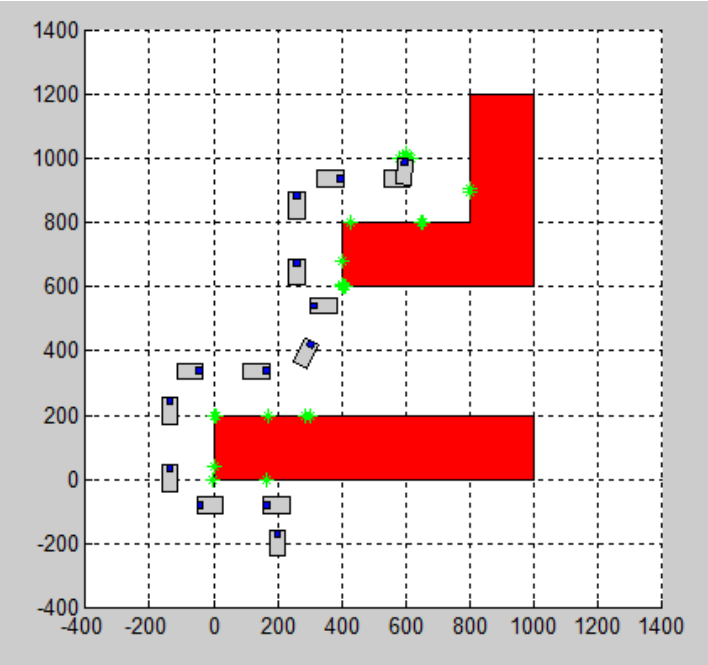
Simülasyon çalışmaları kapsamında yol planlama algoritmaları anlatılırken kullanılan A ve B ortamlarında yol planlama algoritmalarının karşılaştırmaları yapılmıştır. A ortamında Bug1, Bug2 ve Alg2 algoritmaları karşılaştırılmıştır.

Şekil 6.1'de A ortamında Bug1 algoritmasının davranışı görülmektedir.



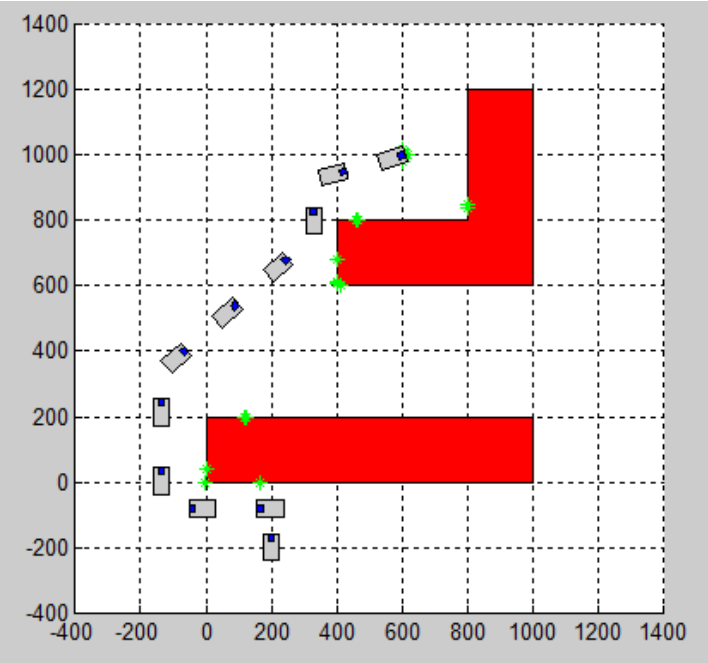
Şekil 6.1 : A ortamında Bug1 algoritması.

Şekil 6.2'de A ortamında Alg1 algoritmasının davranışı görülmektedir.



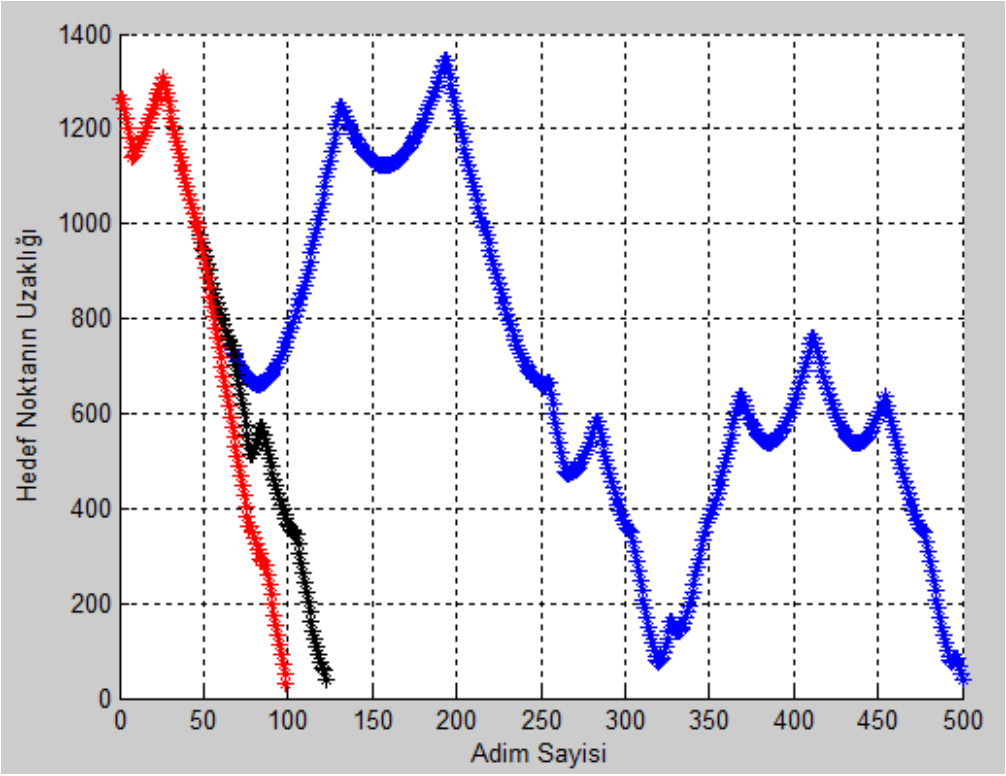
Şekil 6.2 : A ortamında Alg1 algoritması.

Şekil 6.3'te A ortamında Alg2 algoritmasının davranışı görülmektedir.



Şekil 6.3 : A Ortamında Alg2 algoritması.

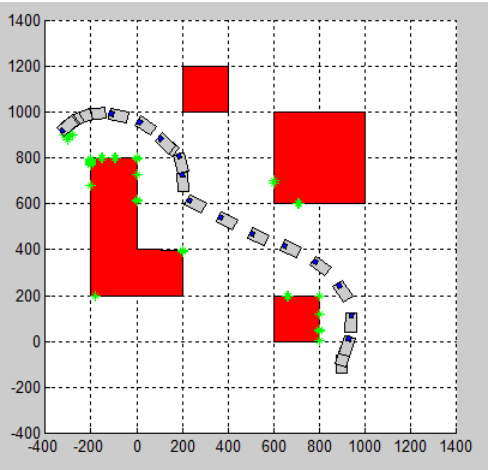
Şekil 6.4'te Bug algoritmalarının hedefe olan uzaklık, adım sayısı grafiği görülebilmektedir. Tüm Bug algoritmalarında robotun hızı sabit alınmıştır. Bug1 algoritması mavi ile, alg1 algoritması siyah ile ve Alg2 algoritması kırmızı ile gösterilmiştir. Alg2 algoritmasının daha kısa sürede çözüme ulaştığı görülebilmektedir.



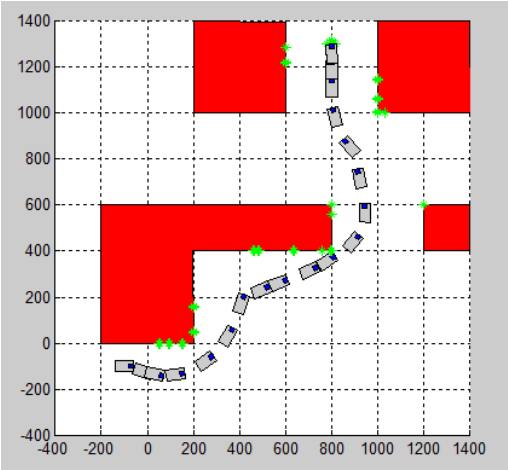
Şekil 6.4 : A ortamında Bug algoritmalarının karşılaştırılması.

6.2 Potansiyel Alanlar Yaklaşımı Simülasyonu

Potansiyel alanlar yaklaşımı ile hareket eden bi robotun iki farklı haritada başlangıç noktasından hedef noktasına olan hareketi Şekil 6.5'te ve Şekil 6.6'da gösterildiği gibidir. Robotun engellere yaklaştıkça yavaşlaması, önü açık olduğu halde itici kuvvetlerin etkisi ile ilerlerken uzun yolu tercih etmesi, engellerden uzak olduğu durumda hızlanması, Şekil 6.6'da ilk engelin 2. köşesine o an üzerine etkiyen itici kuvvetlerin yeterli olmaması nedeniyle tehlikeli şekilde yaklaşması görülmektedir.

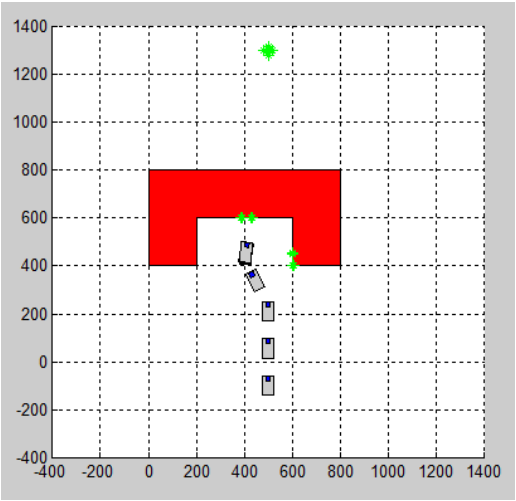


Şekil 6.5 : Potansiyel alan yaklaşımı.



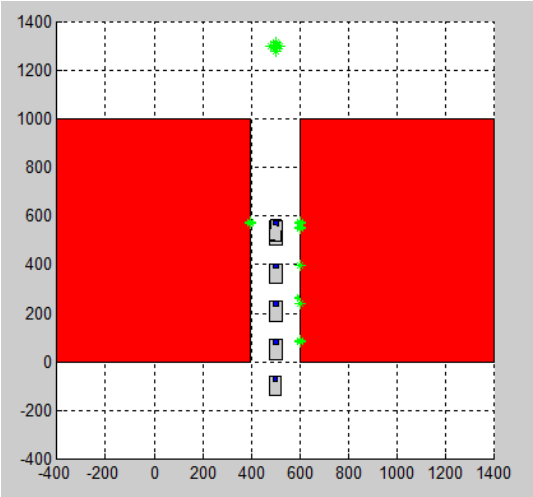
Şekil 6.6 : Potansiyel alan yaklaşımı.

Potansiyel alanlar yaklaşımının başarısız olduğu iki farklı durum aşağıda görülmektedir. Şekil 6.7'de robot yerel minimuma takılmıştır. Kullanıcının itici ve çekici vektörleri ayarlama olasılığına bağlı olarak hedef Yerel minimum probleminin iki farklı türü aşağıda görülmektedir. Buradaki durumda robot hareket edememektedir.



Şekil 6.7 : Yerel minimum 1.

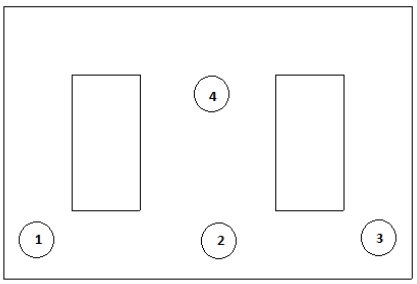
Şekil 6.8'de robot hedef nokta ile arasında herhangi engel olmamasına rağmen içerisinden koridorun darlığı nedeniyle yerel bir minimuma girmiştir. Koridorun başında değilde ortasında bu yerel minimuma girmesinin sebebi robotun çekici kuvvetinin hedefe yaklaştıkça azalması fakat itici kuvvetin sabit kalmasıdır. Koridorun başındaki çekici kuvvet, itici kuvve te üstünken; koridorun ortasında bu durum tersine dönmiş robot yerel minimuma takılmıştır.



Şekil 6.8 : Yerel minimum 2.

6.3 Benzerlik Alanı Metodu Simülasyonu

Benzerlik Alanları Metodu LIDAR verilerinin kullanılarak robotun çalışma uzayının haritasının elde edilmesinde kullanılır. LIDAR verileri işlenirken her bir laser ışını izlediği tüm yol değil bir engele çarpıp geri döndüğü son noktası kullanılır. LIDAR engel olabileceği bilgisini verdiği bu noktanın etrafında LIDAR'ın güvenilirliği ile orantılı olarak engeller olabilir. Benzerlik Alanı Metodunda LIDARın bulduğu son noktalar ve etraflarında engel olma olasılığına dayalı olarak çalışma uzayı haritası oluşturulur [19].



Şekil 6.9 : Mapper3Basic ile oluşturulan ortam.

Benzerlik Alanı Metodu Algoritması aşağıdaki gibidir.

1) q=1
2) for k= 1 : K
3) if LIDAR ölçümü maksimum ölçüm uzaklığı zmax'a eşit ise dikkate alınmaz
4) Harita üzerindeki bir nokta belirlenir.

5) Bu noktanın en yakın engele(LIDAR ölçümüne) olan uzaklığı d bulunur.

6) $$p_r = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{d^2}{2\sigma^2}}$$

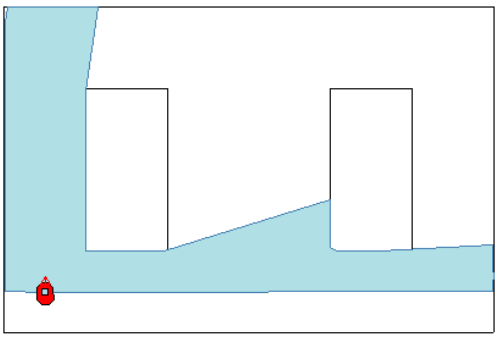
7) $$p = \beta_r p_r + \beta_o \frac{1}{z_{MAX}^2}$$

8) $$q = q\, p_r$$

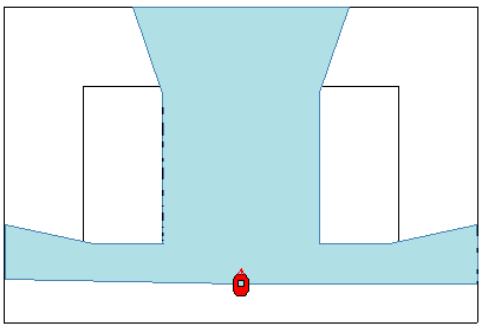
q ilgili noktanın engelli olma olasılığını vermektedir. $\beta_r \beta_0$ sabit katsayılardır.

Benzerlik Alanları Metodunun simülasyonunun yapılması amacıyla Mapper3Basic ortamında Şekil 6.9'da görülen ortam oluşturulmuştur. Bu ortam üzerinde LIDAR ile ölçüm alınabilecek dört farklı ortam belirlenmiştir.

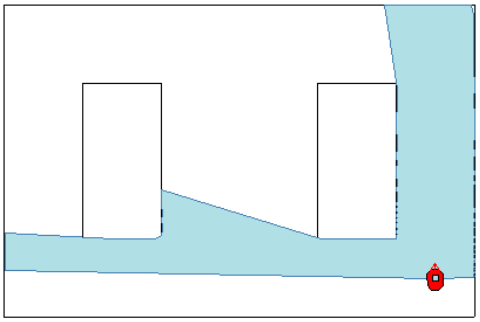
Şekil 6.9'da verilen harita MobileSim ortamına aktarılmıştır. MobileSim ortamında bulunan LIDAR algılayıcıya sahip Pionneer robot belirli haritada belirtilen 4 farklı pozisyona yerleştirilmiştir. Şekil 6.10-6.13'de MobileSim ekran görüntüleri görülebilmektedir. Pionneer robot kırmızı ile görülebilmektedir. Açık mavi renk ile belirtilen bölgeler ise robot üzerinde bulunan LIDAR'ın görüş alanıdır.



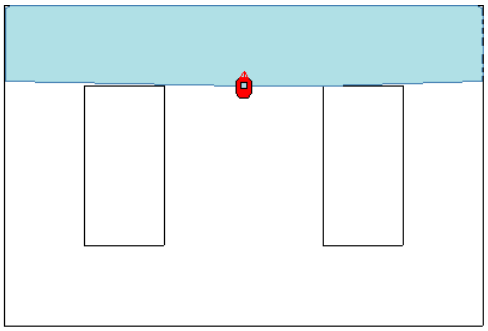
Şekil 6.10 : Birinci pozisyon MobileSim ekran görüntüsü.



Şekil 6.11 : İkinci pozisyon MobileSim ekran görüntüsü.

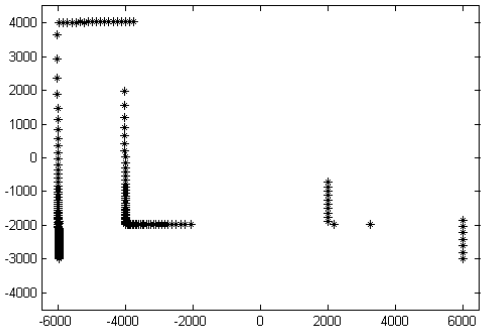


Şekil 6.12 : Üçüncü pozisyon MobileSim ekran görüntüsü.

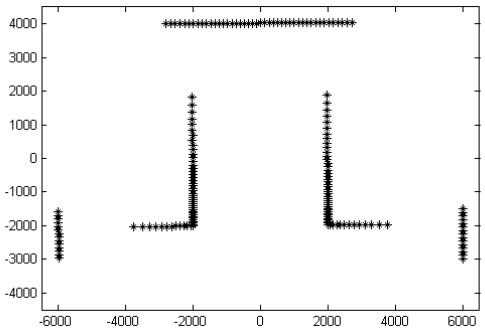


Şekil 6.13 : Dördüncü pozisyon MobileSim ekran görüntüsü.

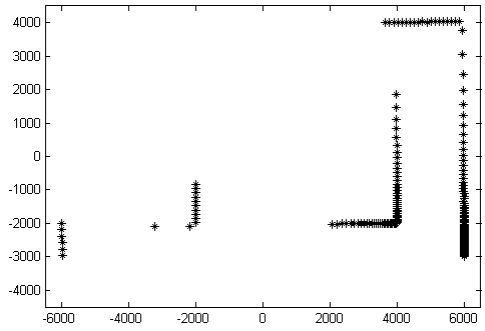
MobileSim ortamından elde edilen LIDAR dataları MATLAB ortamına aktarılıp plot fonksiyonu ile çizdirildiğinde Şekil 6.14-6.17'deki grafikler elde edilmiştir.



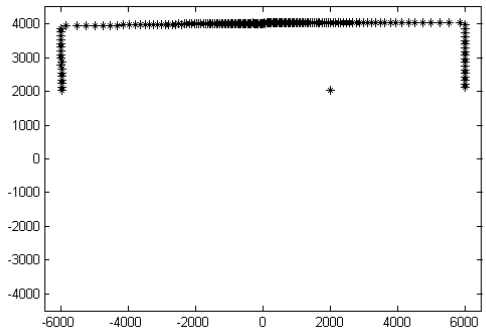
Şekil 6.14 : Birinci pozisyon LIDAR verilerinin MATLAB görüntüsü.



Şekil 6.15 : İkinci pozisyon LIDAR verilerinin MATLAB görüntüsü.



Şekil 6.16 : Üçüncü pozisyon LIDAR verilerinin MATLAB görüntüsü.



Şekil 6.17 : Dördüncü pozisyon LIDAR verilerinin MATLAB görüntüsü.

MATLAB ortamında elde edilen veriler yine MATLAB ortamında Benzerlik Alanı Modeli algoritmasından geçirildiğinde Şekil 46-49'deki sonuçlar elde edilmiştir.

Şekil 6.18 : Birinci pozisyon benzerlik alanı modeli.

Şekil 6.19 : İkinci pozisyon benzerlik alanı modeli.

Şekil 6.20 : Üçüncü pozisyon benzerlik alanı modeli.

Şekil 6.21 : Dördüncü pozisyon benzerlik alanı modeli.

7. SONUÇLAR

Bu çalışmada kapsamında robot kavramının tarihsel gelişimi eincelenmiş. Gezgin robotların tanımı ve amaçları araştırılmıştır. Gezgin robotun kendisine verilen görevi yerine getirmek için niçin algılayıcılara gerek duyduğu ve bu algılayıcıları türleri ile birbirlerine göre avantajları, dezavantajları araştırılmıştır. Robotik simülatör kullanımının bir gezgin robotun gerçeklenmesi için niçin gerekli olduğu açıklanmaya çalışılmıştır. Piyasada bulunan robotik simülatörleri incelenmiş. Aralarında en uygunu olarak seçilen MobileSim üzerinde uygulama geliştirilmiştir. Gezgin robotların görevlerini yerine getirmede kullandıkları yol planlama algortimalarından Bug algoritmaları ve potansiyel alan yaklaşımı incelenmiş bu algoritmalar ile MATLAB ortamında geliştirilen simülatör ile simülasyon yapılmıştır. Benzerlik Alanı Modeli incelenmiş robotik simülatörleri bölümünde incelenen MobileSim simülatörü ile Benzerlik Alanları Metodu simülasyonu yapılmıştır.

KAYNAKLAR

[1] ÖZDEMİR, Durmuş Gezgin Robotların Çiftliklerde Ürün Yeri Belirleme Ve Taşıma İşlemlerinde Kullanımı Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Trabzon, 2009

[2] Arıcı, Volkan Engellerin Bulunduğu Ortamda Gezgin Robotun En İyi Yolu Bulması ve İzlemesi, Başkent Üniversitesi, Fen Bilimleri Enstitüsü, Ankara, 2008

[3] Eroğlu, Elif Gezgin Robotlarda Ultrasonik Mesafe Algılayıcılarla Robot Davranışlarının Kontrolü ve Çevre Haritalama, Osmangazi Üniversitesi, Fen Bilimleri Enstitüsü, Eskişehir, 2006

[4] Yalçın, Özkan İnsansız Kara Araçlarında 2d Lidar Kullanarak Yol Sınırları Tespiti, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü, Kocaeli, 2014

[5] Kızılhan, **Ahmet** Gezgin Robot Tasarımı ve Hareket Planlaması, TOK'07 Bildiriler Kitabı, İstanbul, 5-7 Eylül 2007

[6] V. Lumelsky and P. Stepanov. "Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment". IEEE Transactions on Automatic Control. 1986

[7] A. Sankaranarayanan and M. Vidyasagar, "A New Path Planning Algorithm For Moving A Point Object Amidst Unknown Obstacles In A Plane", IEEE Conference on Robotics and Automation, 1990.

[8] A. Sankaranarayanan and M. Vidyasagar, "Path Planning For Moving A Point Object Amidst Unknown Obstacles In A Plane: A New Algorithm And A General Theory For Algorithm Development", Proc. of the 29th Conference on Decision and Control, 1990

[9] Khatib, **Oussama** Real-time obstacle avoidance for manipulators and mobile robots, Proceedings of the IEEE International Conference on Robotics and Automation, 1985

[10] Borenstein J. and Koren Y. The vector field histogram-fast obstacle avoidance for mobile robots, IEEE Transactions on Robotics and Automation 7, 1991

[11] Ge S.S. and Cui Y.J. , "New potential functions for mobile robot path planning,"*IEEE Transactions on Robotics and Automation*, v.16 (5), pp.615-620, 2000.

[12] I. Kamon and E. Rivlin. Sensory-Based Motion Planning with Global Proofs. IEEE Transaction on Robotics and Automation, 1997.

[13] I. Kamon, E. Rivlin and E. Rimon. TangentBug: A range-sensor based navigation algorithm. Journal of Robotics Research, 1998.

[14] Y. Horiuchi and H. Noborio. Evaluation of Path Length Made in SensorBased Path-Planning with the Alternative Following. Proc. of the IEEE International Conference on Robotics and Automation, 2001.

[15] H. Noborio, Y. Maeda and K. Urakawa. Three or More Dimensional Sensor-Based Path-Planning Algorithm HD-1. Proc. of the 1999 IEEE/RSJ International Conference on Intelligent Robotics and Systems, 1999.

[**16] Lumelsky and S. Tiwari.** An Algorithm for Maze Searching with Azimuth Input. Proc. of the 1994 IEEE Int. Conf. On Robotics and Automation, 1994

[17] E. Magid and E. Rivlin. CautiousBug: A Competitive Algorithm for Sensor-Based Robot Navigation. Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004

[18] K. Kreichbaum. Tools and Algorithms for Mobile Robot Navigation with Uncertain Localization. Ph.D thesis, California Institute of Technology, 2006

[19] TEMELTAŞ. HAKAN. KON 613E Probabilistic Robotics Ders Notları, İstanbul Teknik Üniversitesi, 2012

[20] http://robot.cmpe.boun.edu.tr/593/history/1_Giri_c_s.html, Boğaziçi Üniversitesi Robot Sitesi, 15.12.2014

[21] http://robotics.megagiant.com/history.html, Robotics Laboratory, 15.12.2014

[22] http://el-cezeri1136.blogspot.com.tr/, 15.12.2014

[23] http://www.cyberbotics.com/documentation.php, 15.12.2014

[24] http://www.coppeliarobotics.com/features.html, 15.12.2014

[25] http://robots.mobilerobots.com/wiki/MobileSim#Documentation, 15.12.2014

ÖZGEÇMİŞ

Ad Soyad **:** Ender YOLAL

Doğum Yeri ve Tarihi **:** Üsküdar/İstanbul, 14.06.1988

E-Posta **:** yolalender@gmail.com

ÖĞRENİM DURUMU:

- **Lisans** **:** 2011, İstanbul Teknik Üniversitesi, Elektrik Elektronik Fakültesi, Kontrol Mühendisliği Bölümü

MESLEKİ DENEYİM VE ÖDÜLLER:

2011 Aralık -2012 Mayıs Ar Gör. İTÜ

2012 Mayıs - 2013 Kasım Yazılım Test Mühendisi ASELSAN

2013 Kasım - Sistem Mühendisi ASELSAN