RP-LIDAR KULLANILARAK MOBİL ROBOTLAR İÇİN EŞ ZAMANLI KONUM BELİRLEME VE HARİTALAMA

Selman AKYOL

Yüksek Lisans Tezi

Mekatronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Ayşegül UÇAR

NİSAN-2017

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

RP-LIDAR KULLANILARAK MOBİL ROBOTLAR İÇİN EŞ ZAMANLI KONUM BELİRLEME VE HARİTALAMA

YÜKSEK LİSANS TEZİ

Selman AKYOL
(141134108)

Tezin Enstitüye Verildiği Tarih : 21 Mart 2017
Tezin Savunulduğu Tarih : 11 Nisan 2017

Tez Danışmanı : Doç. Dr. Ayşegül UÇAR (F.Ü.)
Diğer Jüri Üyeleri : Yrd. Doç. Dr. Mehmet POLAT (F.Ü.)
Yrd. Doç. Dr. Özal YILDIRIM (M.Ü.)

NİSAN-2017

ÖNSÖZ

Bu tez çalışmasında ve daha önceki lisans eğitimimde yanımda olan, her konuda desteğini esirgemeyen, birlikte çalışmaktan dolayı çok memnun olduğum ve her zaman olacağım değerli hocam sayın Doç. Dr. Ayşegül UÇAR'a bana göstermiş oluğu ilgi, hoşgörü ve sabrın yanı sıra duymuş olduğu güven için de çok teşekkür ederim.

Tez çalışmamızın uygulamaları için gerekli laboratuvar şartlarının sağlanması ve gerçekleştirilmesi aşamasında beni yalnız bırakmayan sevgili dostlarım ve kıymetli hocalarıma ayrıca teşekkür ederim.

Hayatım boyunca hiçbir zaman maddi ve manevi eksikliklerini hissetmediğim ailemin tüm bireylerine bu tez çalışmasında de bana gösterdikleri ilgi alaka ve sevgi için sonsuz teşekkürler.

Selman AKYOL

ELAZIĞ - 2017

İÇİNDEKİLER

ÖZET

RP-LIDAR KULLANILARAK MOBİL ROBOTLAR İÇİN EŞ ZAMANLI KONUM BELİRLEME VE HARİTALAMA

Otonom robotlar, çevreden aldıkları verileri kendi kendine yorumlayıp karar verebilen mekanizmalardır. Tehlikeli olan veya insanlar tarafından ulaşılması zor ya da ulaşılamayan yerlerde gerekli işlemleri yapabilmek için otonom robotlar kullanılır. Böyle bir durumda otonom robotun gerekli işi yapabilmesi için öncelikle çevresini ve kendi konumunu bilmesi gereklidir. Bilinen bir ortam ise, önceden hazırlanan harita otonom robota yüklenir ancak bilinmeyen bir ortam ise, eş zamanlı olarak hem ortamın haritasını hem de kendi konumunu belirlemesi gerekir. Bu nedenle bilinmeyen bir ortama girdiği zaman hem ortamın haritasını çıkaracak hem de kendi konumunu belirleyecek bir otonom mobil robota ihtiyaç duyulmaktadır.

Bu çalışmada ilk olarak, otonom mobil robotların modellenmesi için gerekli adımlar genel hatları ile verilmiştir. İkinci olarak, Windows 7'de MATLAB ortamında sanal bir ortam oluşturularak Eş Zamanlı Konum Belirleme ve Haritalama (SLAM) işleminin benzetimi yapılmıştır. Üçüncü olarak, Linux'da Robot İşletim Sistemi (ROS) ortamında konum belirleme ve haritalama eş zamanlı olarak gerçekleştirilmiştir. Bu amaç için, Kobuki firması tarafından üretilen Turtlebot mobil robotu kullanılmıştır. Bu uygulamada, Turtlebot bilgisayar USB arabirimi ile manuel olarak kontrol edilmiştir. Ortam haritasının çıkarılması ve robotun konumunu belirlemesi için lazer mesafe ölçümü yapan LIDAR sensörü kullanılmıştır. SLAM için istatistiksel kestirim yöntemlerinden biri olan Kalman Filtresi tabanlı Parçacık Filtresi uygulanmıştır. Bu yöntem için ROS içerisine Gmapping algoritması yüklenmiştir. Son olarak, ROS kullanılarak Turtlebot'un kontrolü otonom bir şekilde gerçekleştirilmiş, deney ortamın haritası aynı zamanda konumunu da belirlenerek çıkarılmıştır.

Tüm uygulamalar başarılı bir şekilde gerçekleştirilmiştir. Elde edilen sonuçlar şekillerle verilmiştir.

Anahtar Kelimeler: Eş zamanlı konum belirleme ve haritalama algoritmaları, LIDAR, Kalman Filtresi, Parçacık Filtresi, Robot işletim sistemi

SUMMARY

SIMULTANEOUS LOCALIZATION AND MAPPING FOR MOBILE ROBOTS USING RP-LIDAR

Autonomous robots are the mechanisms by which they can interpret and decide on the data they receive. Autonomous robots are used to perform necessary operations in areas that are dangerous or difficult to reach or accessible by people. In such a case, the autonomous robot must first know its surroundings and its position in order to be able to do the necessary work. If it is a known environment, the previously prepared map is loaded to autonomous robot. On the other hand, if it is an unknown one, it must simultaneously determine both the map of the environment and its own location. For this reason, when entering an unknown environment, an autonomous mobile robotic is needed that will both map out the environment and determine its own location.

In this thesis, firstly, the necessary steps for the modeling of autonomous mobile robots were given. Secondly, the process of Simultaneous Localization And Mapping (SLAM) was realized for a virtual environment in MATLAB at Windows 7. Thirdly, it was performed SLAM in the Robot Operating System (ROS) at Linux. For this purpose, Turtlebot mobile robot, manufactured by Kobuki Company, was used. In this application, Turtlebot computer was manually controlled via USB interface. The LIDAR sensor, which measures the laser distance, was used to extract the environment map and determine the position of the robot. Particle Filter based on Kalman Filter that is one of the statistical estimation methods was used for SLAM. For this method, the Gmapping algorithm was loaded into the ROS. Finally, the control of Turtlebot was performed autonomously using ROS and the map of the experimental environment was also determined by determining its position at the same time.

All applications were successfully carried out. The results obtained were illustrated in figures.

Keywords: Simultaneous Localization And Mapping algorithm, LIDAR, Kalman Filter, Particle fitler, Robot Operating System

ŞEKİLLER LİSTESİ

Sayfa No

TABLOLAR LİSTESİ

Sayfa No

KISALTMALAR

SLAM **:** Eş Zamanlı Konum Belirleme ve Haritalama - Simultaneous Localization and Mapping

LIDAR **:** Işık Algılama ve Ölçme - Light Detection And Ranging

ROS **:** Robot İşletim Sistemi - Robot Operating System

IMU **:** Atalet Ölçüm Birimi - Inertial Measurement Unit

API **:** Uygulama Programlama Arayüzü - Application Programming Interface

DNS **:** Alan Adı Sistemi - Domain Name System

GKF **:** Geliştirilmiş Kalman Filtresi

EM **:** Beklenti Enbüyültme - Expectation Maximization

GPS : Küresel Konumlandırma Sistemi - Global Positioning System

1. GİRİŞ

Günümüz teknolojisinin büyüyüp ilerlemesinde otonom sistemler büyük öneme sahiptir. Otonom sistemlerin gelişmesi için insanlardan daha hızlı ve doğru kararlar verebilen insansı özelikleri taşıyan algılayıcılar ve algoritmalar geliştirilmesi gerekmektedir. Bunun dışında bir otonom sistemin sadece teorik olarak değil aynı zamanda gerçek hayatta üretilebilmesi mümkün olmalıdır.

Otonom robotların, robotlara ilişkin özelikleri taşıması gerekmektedir. Robotlar, gerçek dünyadan yapılacak işe bağlı olarak bazı ölçümler alırlar. Bu ölçümler ısı, mesafe, ses, şekil ve koku gibi farklı niteliklerde olabilir. Otonom robotlar aldıkları bu ölçümleri otonom olarak yorumlamalı ve en uygun kararı verebilmelidir. En önemlisi de verdikleri bu kararı uygulayabilmelidir.

Otonom robotlar en fazla endüstride kullanılmaktadır. Bunun nedeni daha hassas ve daha çok güç gerektiren işleri hem daha kısa zamanda hem de daha kaliteli bir şekilde yapabilmeleridir. Otonom robotlar endüstri dışında insanların giremeyeceği yerlere veya çok tehlikeli olduğu düşünülen yerlere gönderilerek araştırmalar yapılmaktadır. Örneğin deprem sonrası hasarlı binalarda, karanlık ve keşfedilmemiş mağaralarda, okyanus derinliklerinde veya aktif volkanların çevresinde gerekli araştırmalar yapmak için otonom robotlar kullanılabilmektedir. Bütün bunlar göz önünde bulundurduğunda otonom robotların insan hayatında ne kadar büyük bir öneme sahip olduğu görülmektedir. Özellikle gerekli araştırmaların yapılabilmesi ve uygulamaların gerçekleştirilebilmesi için, ortamın robot tarafından tanınması ve robotun ortam içinde nerede olduğunu bilmesi önemlidir.

Bilimsel yazımda yukarıda bahsedilen bu durumlara Eş Zamanlı Konum Belirleme ve Haritalama (SLAM) problemi denilmektedir. SLAM problemi kısmen bilinen yada hiç bilinmeyen bir ortamda eş zamanlı olarak haritalama ve konum belirlemenin yapılması olarak tanımlanır.

Literatürde SLAM hakkındaki ilk çalışma 1987 yılında Smith ve Cheesman tarafından yapmıştır [1].

Bu alandaki diğer öncü çalışmalar 1990'ların başında Leonard ve Durrant-Whyte [2] araştırma grubu tarafından gerçekleştirilmiştir. Bu grup çalışmalarında, SLAM çözümlerinin sonsuz sayıda olduğunu göstermiştir. Bu bulgu, hesaplama açısından çekiciliğe ve çözüme yaklaşık olan algoritmaları araştırmaya yönlendirmiştir.

2000 yılında Dissanayake ve arkadaşları SLAM için teorik temel ve hesaplamalı bir pratik çözüm sunmuşlardır [3]. Bu çalışmalarla SLAM problemi geliştirmiştir.

2006 yılında da Bailey ve Durrant-Whyte aynı dergide iki farklı çalışma yayınlamışlardır. İlk çalışmalarında, SLAM sorununun olasılıksal biçimi, temel çözüm yöntemleri ve önemli uygulamalar anlatılmıştır [4]. İkincisinde, hesaplama yöntemlerinde son gelişmeler ile büyük ölçekli ve karmaşık ortamlar için SLAM sorununun yeni formülasyonlarını ele alarak, SLAM'ı genel bir kavram şeklinde sunmuşlardır [5].

Haritalama yapılırken çevre algılamasında Işık Algılama ve Ölçme Sistemi (LIDAR) kullanılmıştır. LIDAR, çevresindeki nesnelerin konumlarını algılamak için kullanılan radar benzeri bir sistemdir. LIDAR sistemleri radar sistemlerinden farklı olarak, radyo dalgaları yerine lazer darbeleri kullanır. Lazer ışınları ile ölçülebilecek en büyük uzaklık LIDAR'ın menzili olarak adlandırılmaktadır. LIDAR sistemlerinin temel çalışma ilkesi şu şekildedir: Kaynaktan lazer ışını gönderildikten sonra ışının yayılma ve yansıma zamanları hassas olarak kaydedilir. Işının kaynaktan çıkıp geri gelme süresi ile ışığın sabit hızı kullanılarak LIDAR'ın etrafındaki cisimlerin uzaklıkları hesaplanır [6].

Başlangıçta sadece cisimlerin konumlarını belirlemek için kullanılan LIDAR sistemleri, gelişen teknoloji ile beraber birçok farklı sektörde kullanılmaya başlanmıştır. Bu sektörlerin başında geomatik sektörü gelmektedir. Geomatik sektöründe LIDAR, hem havadan profil oluşturma sistemleri için hem de karadan haritalama ve konum belirleme için kullanılabilmektedir. LIDAR sistemi ile yüzeylerin iki ve üç boyutlu modellerinin çıkarılması sağlanmaktadır [7].

Nguyen ve arkadaşları 2005 yılında LIDAR kullanarak kapalı ortamda bir mobil robotla 2 boyutlu olarak çizgi çıkarma yöntemlerini karşılaştırmışlardır [8].

Chong ve arkadaşları 2015 yılında sensör teknolojileri ve eş zamanlı konum belirleme ve haritalama başlıklı çalışmalarında SLAM problemi için en çok tercih edilen sensörün LIDAR olduğunu söylemişlerdir [9].

SLAM problemiyle ilgili çalışmalar araştırılırken çoğunda istatistiksel çözüm yöntemleri kullanılmıştır. Bunların başında Kalman Filtresi gelmektedir [10]. Ancak Kalman Filtresi algılayıcının aldığı verileri belirli işlemlerden geçirdiği için ve gauss gürültüsüne sahip olması şartlarını koştuğu için farklı yöntemlere yönelim olmuştur.

Calleja, Cetto ve Sanfeliu 2004 yılında yaptıkları çalışmalarında Kalman Filtresinin kararlılık problemini incelenmiş ve bu probleme yönelik çözüm önerileri getirmeye çalışmışlardır [11].

Çayıroğlu 2012 yılında Kalman Filtresinin genel tanımını yapıp uygulamalı bir örnekle göstermiştir [12]. Kalman Filtresinin gürültülü ölçüm yapan bir sistemde kullanıldığı zaman bile güçlü ve başarılı sonuçlar alınabileceğini belirtmiştir.

SLAM problemi sadece karada değil aynı şekilde havadaki mobil robotlar için de temel problem teşkil etmektedir. Oğuz ve Temeltaş 2013 yılında Genişletilmiş Kalman Filtresini (GKF) kullanarak uçak üzerinde SLAM uygulamasını MATLAB Simulink yardımıyla yapmışlardır [13].

Dempster, Laird ve Rubin 1977 yılında yeni bir yöntem olan Beklenti Enbüyültme (BE) yöntemini geliştirmişlerdir [14]. Bu yöntem, Kalman Filtresinde olduğu gibi istatistiksel bir çözüm yöntemidir. Kalman Filtresinden farklı olarak veri gürültüsü problemini ele alarak başarılı bir sonuç ortaya koymuştur.

Thrun, Fox ve Burgard 1998 yılında, algılayıcının aldığı dataları hiçbir işleme tabi tutmadan ve veri gürültüsünün ne olduğuna bakmaksızın farklı bir istatistiksel yöntem olan BE yöntemini kullanmışlardır [15]. Çok başarılı olan bu yöntemin eksikliği gerçek zamanlı olarak uygulanamaması olmuştur. Bu yüzden gerçek zamanlı uygulamalarda kullanılmamıştır.

Yuen ve MacDonald 2002 yılında, Parçacık Filtreleme yöntemini kullanmışlardır [16]. Bu yöntem, Kalman Filtresi gibi aktif ve hızlı olmamasına rağmen gerçek zamanlı ve çoğu probleme çözümler getirdiği için kullanılmıştır.

Bu tez çalışmasında, diferansiyel tekerlek modeline sahip Turtlebot mobil robotu kullanılarak üzerine yerleştirilen Robo Peak firmasının LIDAR (RP-LIDAR) sensörü sayesinde hem otonom olarak hem de manüel olarak kontrolünü sağlayıp SLAM uygulamasının yapılması amaçlanmıştır. Bu amaca uygun deneysel düzenek oluşturulmuştur. Çalışmada hem SLAM hem de mobil robot kontrolü Linux tabanlı işletim sistemine sahip bir bilgisayarda gerçekleştirilmiştir. Bilgisayara Linux tabanlı işletim sistemine uyumlu Robot İşletim Sistemi (ROS) kurulmuştur. SLAM uygulaması ve manüel olarak mobil robot kontrolü ROS üzerinden yapılmıştır. ROS üzerinde Parçacık Filtresi tabanlı Gmapping paketi ile SLAM uygulaması gerçekleştirilmiştir.

Sonuç olarak, bu tez çalışması için yapılan deney düzeneği ile hem otonom hem de manüel olarak mobil robot kontrolü ve SLAM algoritmaları başarılı bir şekilde gerçekleştirilmiştir.

1.1. Tez Düzeni

Tezin geri kalan bölümleri şu şekilde organize edilmiştir.

İkinci bölümde, mobil robotlarla ilgili genel bilgiler verilmiştir. Ackermann tekerlek modeli ve tez çalışmasında kullanılan Turtlebot mobil robotun modellenmesi için gerekli hesaplanmalar yapılmıştır.

Üçüncü bölümde, uygulama platformu olan ROS tanıtılmış ve ROS'un alt birimleri ile ilgili geniş bilgiler verilmiştir.

Dördüncü bölümde, kullanılan istatistiksel kestirim yöntemlerinden Kalman Filtresi ve Parçacık Filtresi detaylı bir şekilde ele alınarak SLAM algoritmaları incelenmiştir.

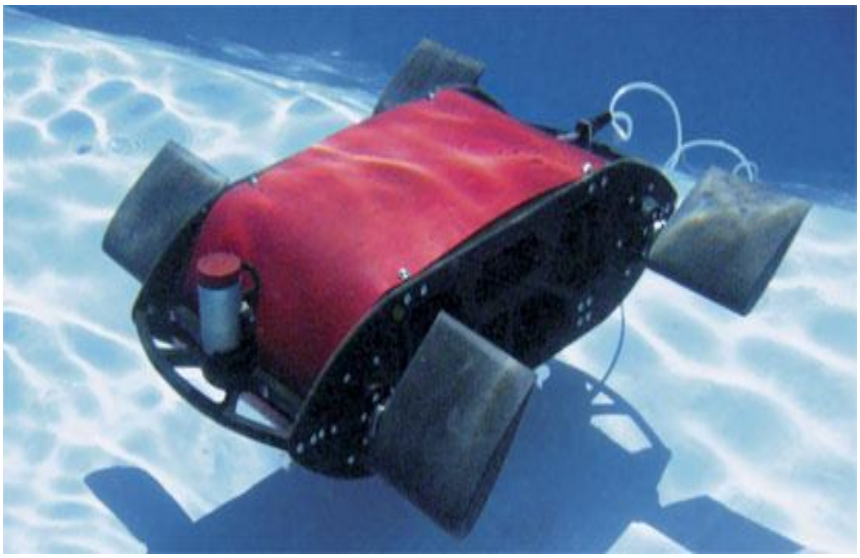
Beşinci bölümde, RP-LIDAR kullanılarak eş zamanlı haritalama yapan ve konum belirleyen mobil robot uygulamalarının gerçekleştirme aşamaları ayrıntılı bir şekilde anlatılmıştır. Uygulama sonuçları şekiller ile verilmiştir.

Altıncı bölümde, tezde elde edilen sonuçlar ve öneriler sunulmuştur.

2. MOBİL ROBOTLARIN KİNEMATİK MODELLEMESİ

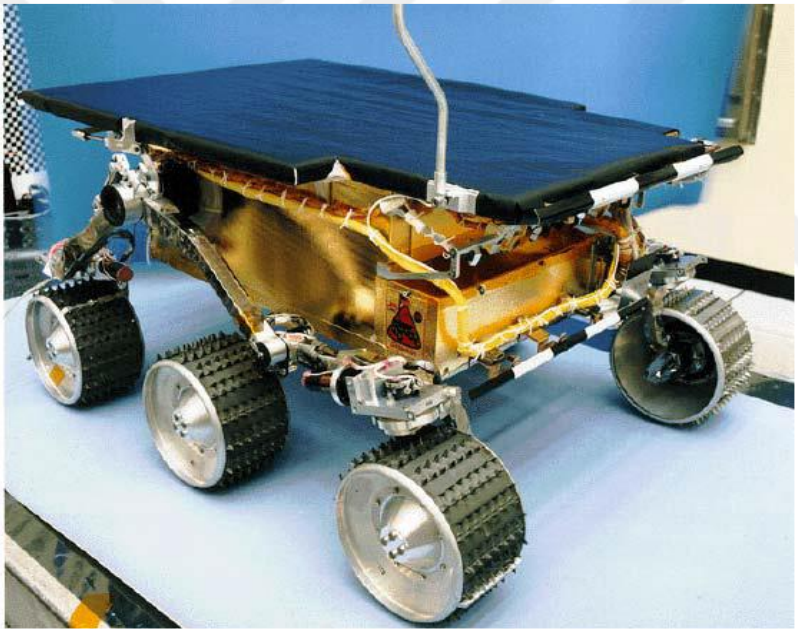
2.1. Mobil Robot Tanımı

Mobil robotlar endüstriyel robotlardan farklı olarak belirli bir noktada sabitlenmemiş, karada, su altında ya da havada hareketini sağlayabilen ve istenilen görevleri uygun bir şekilde uygulayabilen robotlardır. Mobil robotlar, gelişen teknolojiyle birlikte insan hayatında çok fazla yer almaya başlamışlardır. Mobil robotların kullanım alanlarına bakıldığında karada, havada, su altında ve uzayda olmak üzere insanların hizmetine sunularak farklı görevlerde ve farklı çalışmalarda kullanıldığı görülmektedir. Su altındaki mobil robotlar genellikle güvenlik veya keşif amacıyla kullanılan mobil robotlardır. Hava mobil robotları genellikle insansız hava aracı olarak isimlendirilir ve genellikle savunma sanayisinde kullanılmıştır. İnsansız hava araçları orduda; bombalama, keşif, savaş ve lojistik gibi değişik görevlerde kullanılmıştır. Aşağıda farklı görev ve çalışmalar için üretilmiş mobil robotların fotoğrafları gösterilmiştir. Şekil 2.1'de denizaltında araştırmalar ve gerekli görevler için kullanılan insansız denizaltı robotu, Şekil 2.2'de ise hava gözlemleri yapan bir insansız hava aracı gösterilmiştir. Mobil robotlar uzay araştırmalarında da kullanılmaktadır. 1996 yılında NASA tarafından Marsa fırlatılan ve 4 Temmuz 1997'de Mars yüzeyine inen Sojourner robotu. Şekil 2.3'de gösterilmiştir.



Şekil 2.1. Denizaltı mobil robotu

Şekil 2.2. Hava mobil robotu



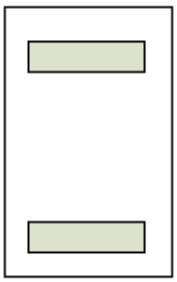
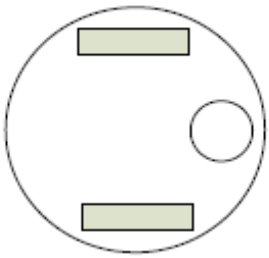
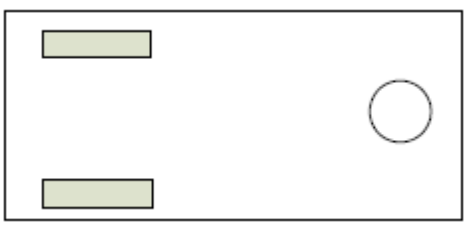
Şekil 2.3. Marsa gönderilen ilk mobil robot

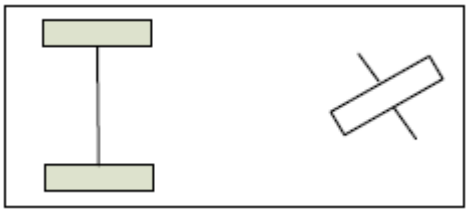
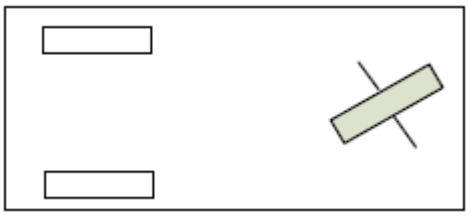
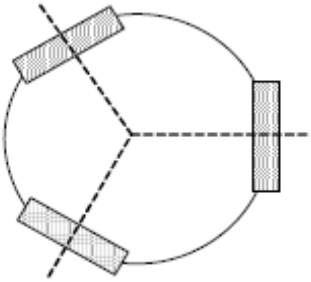
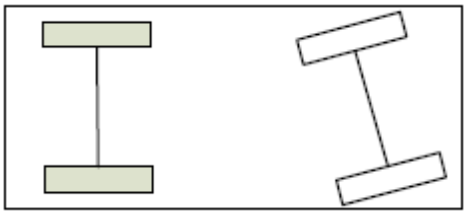
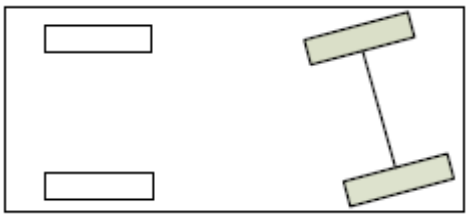
2.2. Tahrik Sistemleri

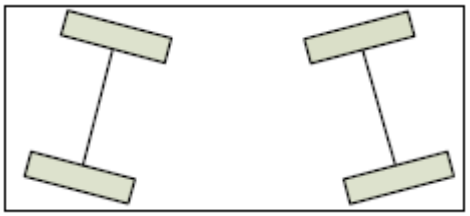
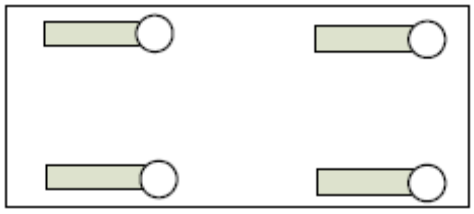
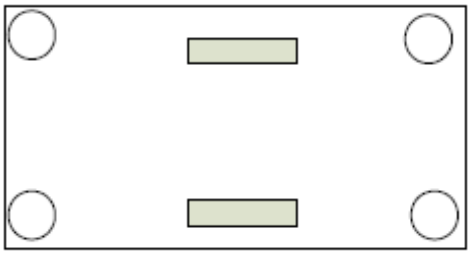
İnsansız kara araçlarının hareketlerini düzgün bir şekilde yapabilmeleri için farklı tahrik sistemlerine ihtiyaç vardır. Bu sistemler tekerlekli olabileceği gibi, paletli, ayaklı da olabilmektedirler. Bu yüzden robotun yapacağı çalışmayı göz önünde bulundurarak ne tür bir mekanizma kullanılacaksa ona uygun tahrik sistemini seçmekte çok önemlidir. En fazla tercih

edilen tahrik sistemi tekerlekli mekanizmalardır. Mobil robotun hareket kabiliyeti ve yapması gereken manevraları rahatlıkla yapabilmesi açısından tekerlek yapıları fazla tercih edilmektedir. İnsansız kara araçlarında kullanılan tekerleklere bakıldığında standart, İsveç, küresel ve nakliye olmak üzere dört temel tekerlek sınıfı bulunmaktadır [17]. Tablo 2.1'de bu tekerleklerle birlikte farklı insansız kara araç yapılanışları gösterilmiştir.

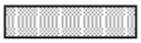
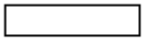
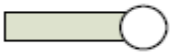
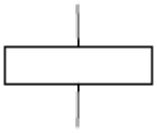
Tablo 2.1. Tekerlekli insansız kara araçlarının bazı yapılanışları [18]

Teker sayısı	**Teker düzeni**	**Açıklama**	**Örnek**
2		Yön belirleyen bir ön teker, gövdeyi ön tekerin çektiği yere sürükleyen bir arka teker vardır.	Bisiklet, Motorsiklet
		İki teker bir mille bağlıdır, ağırlık merkezi milin orta noktasıdır.	
3		Önde bir teker, arkada farklı sürüş merkezleri olan iki teker yerleştirilmiştir.	
		Önde çok yönlü bir teker, arkada iki bağımsız teker yerleştirilmiştir.	Genellikle kapalı ortam robotları (Plgmalion ve Alice)

		Önde yön veren bir teker, arkada bir mille birbirine bağlı iki ayrı teker yerleştirilmiştir.	
		Önde yön veren bir teker, arkada iki teker yerleştirilmiştir.	Neptune (Carnegie Mellon Üniversitesi)
		Üçgen şeklinde üç adet çok yönlü İsveç tekeri veya küresel teker yerleştirilmiştir.	Tribolo EPFL, Palm Pilot Robot Kit (Carneige Mellon Üniversitesi)
4		Önde yön veren iki teker, arkada motorlu iki teker yerleştirilmiştir.	Arkadan sürüşlü arabalar
		Önde iki motorlu ve yön veren teker, arkada iki serbest teker yerleştirilmiştir.	Önden sürüşlü arabalar

		Dört adet yön belirleyen ve motorlu teker yerleştirilmiştir.	Dört tekerli sürüşe sahip Hyperion (Carnegie Mellon Üniversitesi)
		Dört adet motorlu ve yön belirleyen taşıyıcı tekerler yerleştirilmiştir.	Nomad XR4000
6		Merkezde iki adet çekici (sürükleyici) teker, her köşede birer tane çok yönlü teker yerleştirilmiştir.	Terragator (Carnegie Mellon Üniversitesi)

Yukarıda gösterilen teker çeşitlerine ait sembollerin anlamı

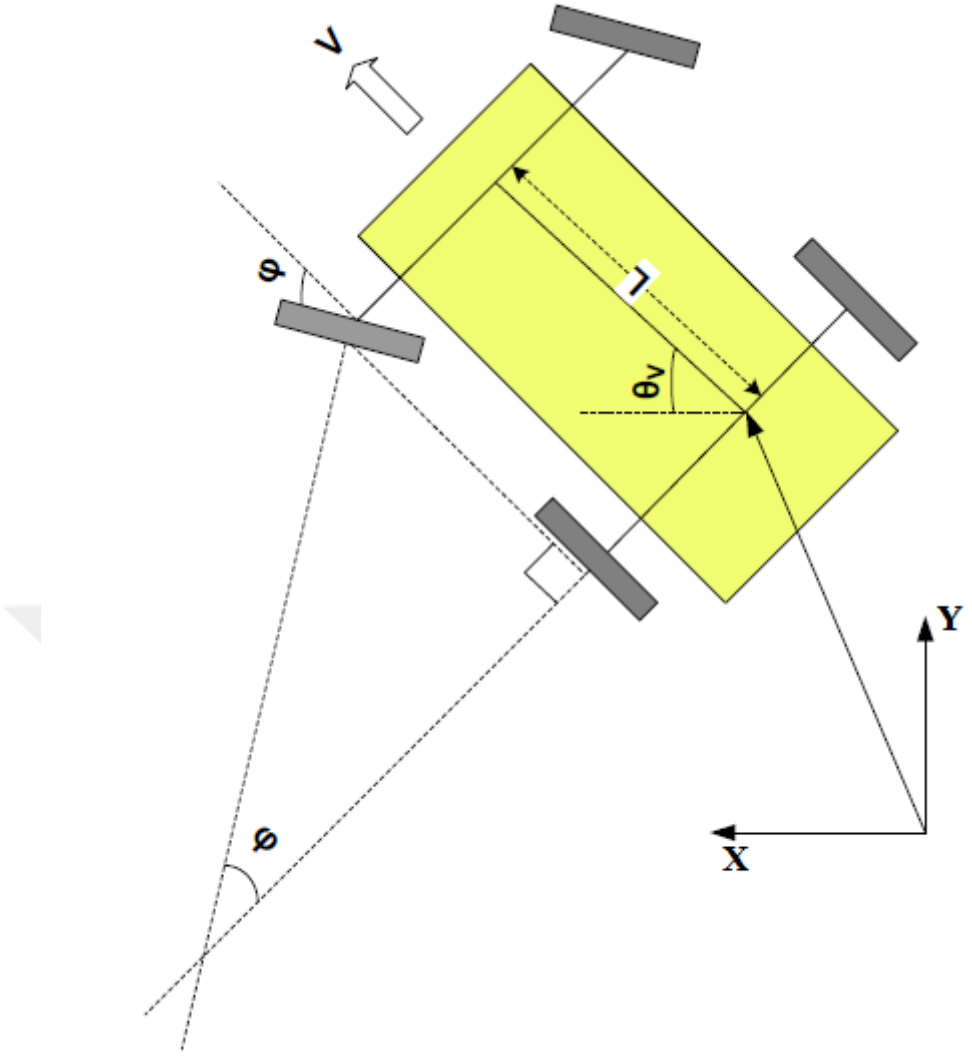
	Güç verilmemiş çok yönlü teker (küresel, nakliye tekeri, İsveç tekeri)
	Motorlu İsveç tekeri (Stanford tekeri)
	Güç verilmemiş standart teker
	Motorlu standart teker
	Motorlu ve yön belirleyen nakliye tekeri
	Yön belirleyen standart teker

	Bağlı tekerler

Bu yapılanışlardan bazıları mobil robot uygulamalarında fazla tercih edilmemektedir. İki tekerlekli olan mekanizmaların ilk sırasındaki yapılanışa bakıldığında hem hareket kabiliyeti sınırlıdır hem de kontrol edilip ayakta durdurulması zordur. En fazla tercih edilen tekerlek yapıların başında Ackermann tekerlek yapısı olarak adlandırılan ve Tablo 2.1'deki 4 tekerlek yapısının ilk sırasında verilen yapılanış gelmektedir. En fazla tercih edilen bir diğer tekerlek yapısı diferansiyel sürücü olarak adlandırılan ve Tablo 2.1'de 3 tekerlekli, 4 tekerlekli ve 6 tekerlekli olarak verilen yapılanışlardır.

2.2.1. Ackermann Tekerlek Modeli

Ackermann modeli 1817 yılında Rudolph Ackermann tarafından önerilmiştir [19]. Günümüzdeki araçlarda kullanılan modellerin çoğunda, Ackermann modelini göz önünde bulundurarak geliştirmeler yapılmıştır. Ackermann modeli hem öndeki iki tekerleklerle hem de arkadaki iki tekerlekle tahrik edilebilmektedir.



Şekil 2.4. Ackermann tekerlek modeli

V hızıyla hareket eden aracın kinematik denklemeleri

$$\dot{x} = V.\cos\theta \quad \textbf{(2.1)}$$

$$\dot{y} = V.\sin\theta \quad \textbf{(2.2)}$$

$$\dot{\theta} = V.\frac{\tan\varphi}{L} \quad \textbf{(2.3)}$$

$$u(k) = \begin{bmatrix} V(k) \\ \varphi(k) \end{bmatrix} \quad \textbf{(2.4)}$$

olarak elde edilir. Burada; (x,y) düzlemdeki konum ve $\dot{\theta}$ aracın referans alınan düzlemle yaptığı açı değişimini, u(k) aracın kontrol girişini, $V(k)$, k anındaki aracın hızını ve $\varphi(k)$'da k anındaki aracın tekerleklerinin yaptığı açıyı ve L aracın tekerlekler arasındaki mesafesini göstermektedir.

Aracın k anındaki pozisyonu

$$x_v(k) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \quad \textbf{(2.5)}$$

ile gösterilir.

Aracın kinematik modeli

$$x_v(k+1) = f(x_v(k), u(k)) \quad \textbf{(2.6)}$$

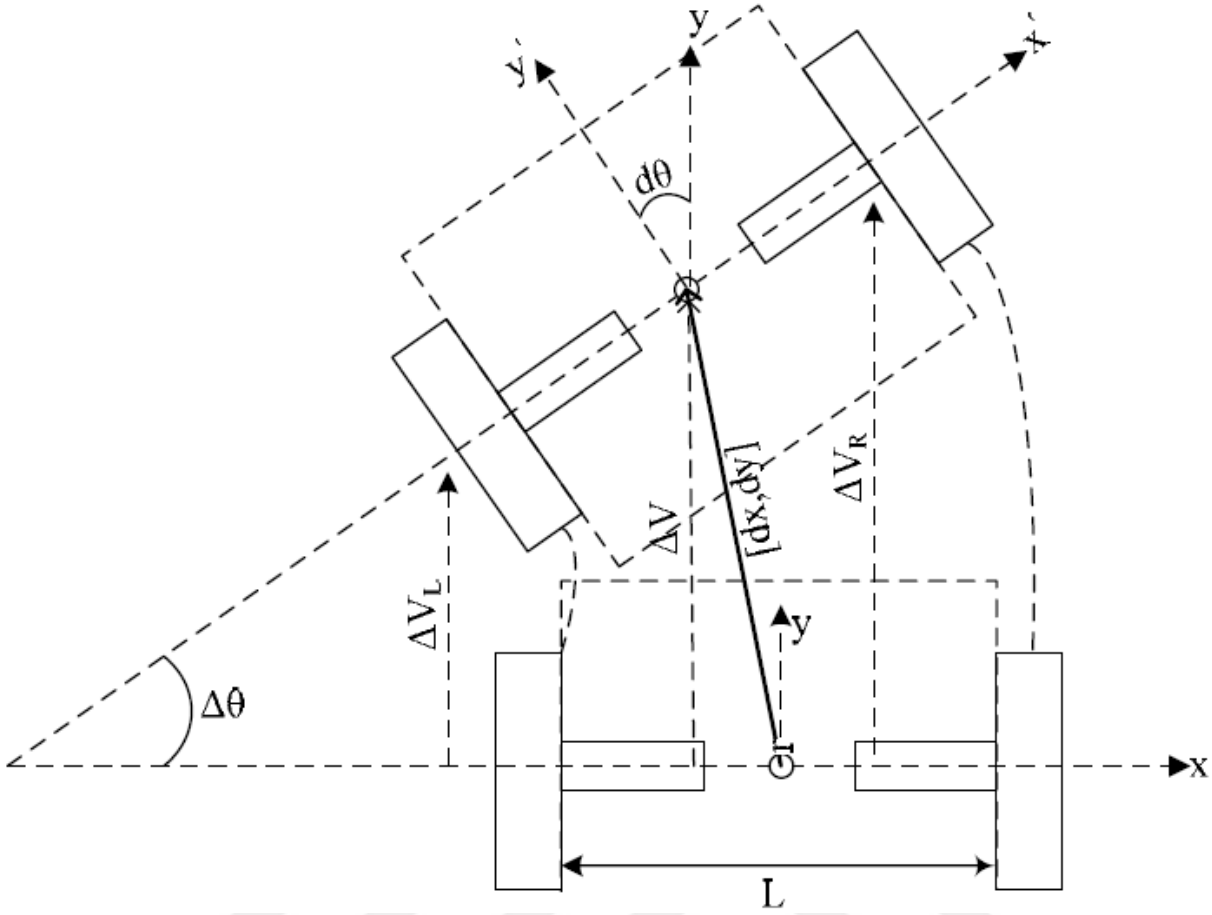
olarak tanımlanır. Araca (2.4)'deki u(k) kontrol işareti uygulandığında aracın k+1 anındaki pozisyonu

$$\begin{bmatrix} x_v(k+1) \\ y_v(k+1) \\ \theta_v(k+1) \end{bmatrix} = \begin{bmatrix} x_v(k) + \Delta T.V(k).\cos(\theta_v(k)) \\ y_v(k) + \Delta T.V(k).\sin(\theta_v(k)) \\ \theta_v(k) + \frac{\Delta T.V(k)\tan(\varphi_v(k))}{L} \end{bmatrix} \quad \textbf{(2.7)}$$

olarak bulunur [19].

2.2.2. Diferansiyel Tekerlek Modeli

Ackermann tekerlek modeline göre diferansiyel tekerlek modelinin kontrolü daha karmaşık olmasına rağmen uygulamada kolaylık sağladığından dolayı tercih edilmektedir. Bu kolaylıklar, çok esnek olması, dönüşlerini daha iyi yapması, ileri geri hareketlerini daha hızlı ve kolay yapması, engelleri fark ederek hızlı bir şekilde manevra yapması olarak sıralanılabilir. Ayrıca diferansiyel tekerlek modelindeki çok yönlü tekerlekler, mobil robotun üzerindeki fazla yükü alarak robotun daha kolay hareket etmesini sağlamaktadır.



Şekil 2.5. Diferansiyel tekerlek modelinde mobil robotun bir yerden başka bir yere hareket işlemi

Diferansiyel tekerlek modeline sahip araç k anından k+1 anına kadar ötelenirse kinematik denklemleri aşağıdaki gibi çıkartılır [20].

Sağ ve sol tekerlek hız değişimleri kullanılarak aracın hız ve açı değişimleri

$$\left.\begin{aligned}\Delta V_L &= C_m.N_L\\ \Delta V_R &= C_m.N_R\end{aligned}\right\} \quad \textbf{(2.9)}$$

olarak hesaplanır. Burada; N_L sol tekerlek encoder farkı ve N_R sağ tekerlek encoder farkını göstermektedir.

$$C_m = \frac{\pi.D_n}{n.C_e} \quad \textbf{(2.8)}$$

Burada; D_n nominal tekerlek çapını, C_e encoder çözünürlüğünü ve n de dişli oranını göstermektedir.

Sağ ve sol tekerlek hız değişimlerini kullanarak aracın hız ve açı değişim değerleri

$$\Delta V = \frac{\Delta V_R + \Delta V_L}{2} \; ; \; \Delta\theta = \frac{\Delta V_R - \Delta V_L}{L} \tag{2.10}$$

olarak tanımlanır.

$$\begin{bmatrix} d_x \\ d_y \\ d_\theta \end{bmatrix} = \begin{bmatrix} \Delta V . \cos(\theta(k+1)) \\ \Delta V . \sin(\theta(k+1)) \\ \Delta\theta \end{bmatrix} \tag{2.11}$$

$$\left.\begin{aligned} \Delta V &= \mathrm{r}.\frac{\Delta\omega_R + \Delta\omega_L}{2} \\ \Delta\omega &= \mathrm{r}.\frac{\Delta\omega_R - \Delta\omega_L}{L} \\ \Delta\theta &= \Delta\omega \end{aligned}\right\} \tag{2.12}$$

Burada; *r* tekerlek yarıçapını göstermektedir.

$$\left.\begin{aligned} x_v(k+1) &= x_v(k) + \cos\big(\theta(k) + \Delta(\theta)\big) \\ y_v(k+1) &= y_v(k) + \sin\big(\theta(k) + \Delta(\theta)\big) \\ \theta(k+1) &= \theta(k) + \Delta(\theta) \end{aligned}\right\} \tag{2.13}$$

2.3. Turtlebot

Bu tezde uygulamalar Kobuki firması tarafından üretilmiş olan Turtlebot mobil robotu üzerinde yapılmıştır. Turtlebot mobil robotu birçok akademik çalışmada kullanılmıştır [21]. Turtlebot'u tercih etme sebebi SLAM uygulamaları için uygun özeliklere sahip olabilmesidir. Turtlebot diferansiyel tekerlek modeli ile oluşturulmuştur. Diferansiyel tekerlek modeli SLAM uygulamalarını yapabilmek için en önemli özeliktir. Bu özelik sayesinde mobil robot olduğu yerde dönebilmektedir. Robotun genel özelikleri Tablo 2.2'de verilmiştir [22].

Tablo 2.2. Turtlebot mobil robotun genel özelikleri

Özelikler	**Açıklama**
Çap	351.5 mm
Yükseklik	124.8 mm
Ağırlık	2.35 kg

Maksimum öteleme hızı	70 cm / s
Maksimum dönme hızı	180 derece / s
Taşıma kapasitesi	5 kg (sert zemin), 4 kg (halı)
Çarpma sensörleri	Sağ, orta, sol
Uçurum sensörü	Sağ, orta, sol
Uçurum	5cm daha büyük bir derinliğe sahip bir uçurumdan aktifleşir
Normal Çalışma Süresi	3/7 saat (küçük / büyük pil)
Motor	2 tane fırçalı DC motor

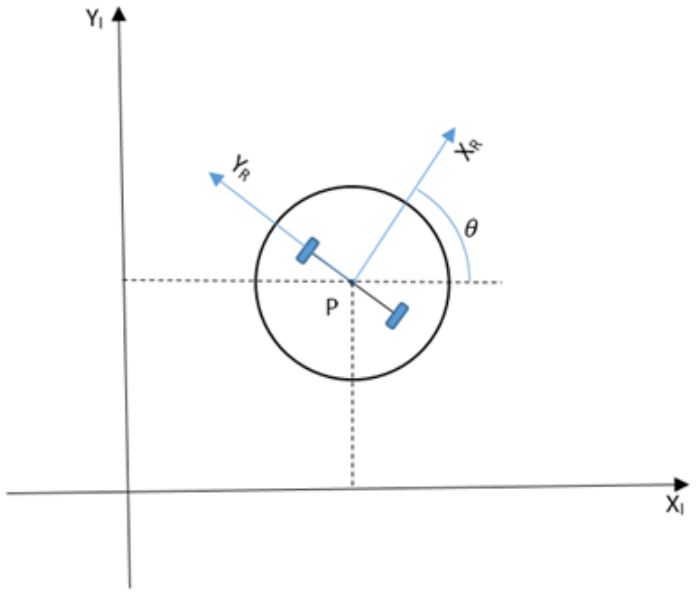
Şekil 2.6. Uygulamada kullanılan Turtlebot mobil robotu

Turtlebotun içinde Linux tabanlı bir işletim kartı mevcuttur. Bu kart kullanılarak Turtlebot otonom olarak hareket edebilir. Ayrıca Linux tabanlı bir bilgisayara ROS işletim sistemi kurularak gerekli işlemler yapıldıktan sonra manuel kontrolü sağlanabilir.

2.3.1. Turtlebot Kinematik Modeli

Turtlebot'ta kullanılan modelleme Şekil 2.7'de gösterilmiştir [23]. Tablo 2.1'de gösterilen 3 tekerlekli modellere benzemektedir. Tek farkı sadece önde değil aynı şekilde arkada da bir

tane çok yönlü tekerin bulunmasıdır. Bu özelik, Turtlebot'un yükünün bir kısmını kendi üzerine alarak hareketin kolaylaştırmasını sağlamaktadır.



Şekil 2.7. Turtlebot tekerlek modeli

X_I ve Y_I evrensel referans çatısı X_R ve Y_R bölgesel referans çatısı için Turtlebot mobil robotun dönüş hızı

$$\dot{x}_R = \frac{r(\dot{\varphi}_1+\dot{\varphi}_2)}{2} \quad \textbf{(2.14)}$$

olarak tanımlanır. Burada; r parametresi tekerlek yarıçapını göstermektedir. P noktasının açısal hızı

$$\dot{\theta}_1 = \frac{r\dot{\varphi}_1}{2l} \quad \textbf{(2.15)}$$

ile bulunur. Burada; $\dot{\varphi}_i$ her bir tekerleğin hızını tanımlamaktadır. Dolayısıyla, P noktasının toplam açısal hızı

$$\dot{\theta} = \frac{r}{2l}(\dot{\varphi}_1 - \dot{\varphi}_2) \quad \textbf{(2.16)}$$

olarak elde edilir. Robotun pozisyonu

$$\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(\mathrm{l}, \mathrm{r}, \theta, \dot{\varphi}_1, \dot{\varphi}_2) \quad \textbf{(2.17)}$$

ile tanımlanır. Referans çerçevesinin açısal dönüşü için, döndürme matrisi aşağıdaki şekilde verilir:

$$\mathrm{R}(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(2.18)}$$

Bu nedenle, çerçeveler arasındaki robot hareketi

$$\dot{\xi}_R = \mathrm{R}(\theta)\, \dot{\xi}_I \quad \textbf{(2.19)}$$

$$\dot{\xi}_I = R(\theta)^{-1} \dot{\xi}_R \quad \textbf{(2.20)}$$

$$R(\theta)^{-1} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(2.21)}$$

$$\dot{\xi}_I = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \frac{r}{2} \begin{bmatrix} \dot{\varphi}_1 + \dot{\varphi}_2 \\ 0 \\ \frac{\dot{\varphi}_1 + \dot{\varphi}_2}{l} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \quad \textbf{(2.22)}$$

olarak elde edilir.

3. ROBOT İŞLETİM SİSTEMİ

2000'li yılların başından itibaren teknoloji ve robotik alanlarında hızlı bir ilerleme olduğu görülmektedir. Bununla beraber karmaşıklığı artmakta olan robotik uygulamalar da çoğalmıştır. Robotlarda tıpkı bilgisayarlardaki gibi yazılım ve donanım arasındaki ilişkiyi sağlamak için bir işletim sistemine ihtiyaç duyulmaktadır. Çoğu araştırmacı yaptıkları robotik uygulamalarda kendi işletim sistemlerini yaparak gerekli çalışmaları yapmışlardır. Ancak bu birden fazla robotik uygulamada ya da bir uygulamada birden fazla işletim sistemi kurulan uygulamalarda hem yapılması hem de böyle bir çalışmanın anlaşılması çok karmaşıklaşmıştır.

Bu karmaşıklığın ortadan kaldırılması için 2007 yılında ROS yani robot işletim sistemi çalışmaları yapılmıştır. ROS platformunun ana hedeflerinden biri, kodun tekrar kullanılmasıdır. Bunun içinde açık kaynak olarak herkesin kolaylıkla erişebileceği şekilde sunulmuştur [24]. Bu özelik sayesinde belirli bir amaç için yazılan bir kod, başkaları tarafından kullanılabilir ve paylaşılabilir. Bu sayede her geçen gün ROS kütüphaneleri gelişmektedir.

ROS'un en büyük rahatlıklarından biri de bağımsız bir dil mimarisine sahip olmasıdır. Aynı uygulama için C++, python, lisp ya da java gibi farklı diller kullanılabilir. Hatta aynı robotik uygulamada birden fazla farklı dil kullanılarak uygulama gerçekleştirilebilir.

3.1. Ana Platformlar

ROS, kavramsal bir işletim sistemidir. Dolayısıyla gerçek bir işletim sisteminde çalışmalıdır. Şu an için yalnızca UNIX tabanlı sistemler üzerinde çalışılmaktadır. ROS'un ana desteği Ubuntu ve Mac OSX içindir. Fakat topluluğun katkılarıyla Fedora, Gentoo, Arch Linux, Raspbian ve daha pek çok diğer Linux dağıtımlarına aktarılmıştır. Windows için çalışan bir sürüm gerçekleştirilmemiştir.

3.2. ROS'un Sürümleri ve Dağıtımlar

ROS'un farklı sürümleri, farklı zamanlarda dağıtılmıştır. Bu sürümler ve zamanları aşağıdaki gibidir.

- ROS Box Turtle (March 2, 2010),
- ROS C Turtle (August 2, 2010),
- ROS Diamondback (March 2, 2011),

• ROS Electric Emys (August 30, 2011),
• ROS Fuerte Turtle (April 23, 2012),
• ROS Groovy Galapagos (December 31, 2012),
• ROS Hydro Medusa (September 4th, 2013),
• ROS Indigo Igloo (July 22nd, 2014),
• ROS Kinetic Kame (May 23nd, 2016),

3.3. ROS Kavramları

ROS'un üç seviyeli kavramı vardır: dosya sistemi seviyesi, hesaplama grafiği seviyesi ve isimlendirme seviyesidir. Bu seviyeler ve kavramlar ayrıntılı olarak incelenip alt başlıklar da açıklanmıştır.

3.3.1. ROS Dosya Sistemi Seviyesi

ROS üzerinde geliştirilen tüm kavramlar ve uygulamalar, verileri işlemek, taşımak ve saklamak için kaynaklara ihtiyaç duyar. Bu ihtiyaç için eşsiz bir dosya sistemi oluşturulmuştur. Bu dosya sistemi uygun veri yönetimi, uygulamalar ve paketleme kısımlarının yapılmasını sağlar.

3.3.1.1. Paketler

ROS'ta geliştirilen bir uygulama, bileşenler birleştirilir ve paket halinde düzenlenir. Böylece yazılımın ROS üzerinden oluşturulması ve yayın izni verilmesi sağlanır. Bir paket düğümler, ROS bağımlı-bağımsız kütüphaneler, yapılandırmalar, veri kümeleri ve uygulama ile ilgili her şeyi içerebilir.

3.3.1.2. Metapaketler

Metapaketler ROS'ta uzmanlaşmış temelde sanal paketlerdir. Dosyaları yüklemezler ve genellikle paketlerde bulunan testler, kodlar, dosyalar veya başka öğeler içermezler. Yalnızca bir veya daha fazla pakete başvurmaya yarar.

3.3.1.3. Paket Manifestleri

Paket manifestleri, herhangi bir catkin uyumlu paketin kök dizininde bulunması gereken package.xml adlı bir XML dosyasıdır. Bu dosya, paketin adı, sürüm numarası, yazarını, bakımcı ve diğer catkin paketleri üzerindeki bağımlılıklar gibi paketle ilgili özellikleri tanımlar. Ayrıca paket oluşturma sistemi paket manifestoları tarafından sağlanan bu bilgileri kullanır.

3.3.1.4. Mesaj Türleri

Mesaj türü, temel formunda, ".msg" uzantılı bir dosya ve dâhil edilen veri yapıları için önceden tanımlanmış bir yazı gösterimidir. ROS, ROS düğümlerinin yayınladığı veri değerlerini tanımlamak için basitleştirilmiş mesaj açıklama dili kullanır. Bu açıklama, ROS araçlarının mesaj türü için birkaç hedef dilde otomatik olarak kaynak kodu oluşturmasını kolaylaştırır. Mesaj açıklamaları bir ROS paketinin msg / alt dizinindeki. msg dosyalarında saklanır. Bir .msg dosyasında iki bölüm vardır: alanlar ve sabitler. Alanlar, mesajın içinde gönderilen veridir. Sabitler, bu alanları yorumlamak için kullanılabilecek yararlı değerleri tanımlar.

3.3.1.5. Hizmet Türleri

Hizmet türü, dosya biçimi olarak ileti türüne benzer. Bunun dışında bir ".srv" şeklinde bir uzantısı bulunur ve veri yapısı gösterimi istek ve yanıt olarak ayrılmış iki bölüm içerir. Sistem üzerinden hizmet iletişimi sağlamak için isim hizmetinde kullanılır.

3.3.2. ROS Hesaplama Grafik Seviyesi

ROS'un en heyecan verici yönlerinden biri, yalnızca bir ana veri denetleyicisi tarafından denetlenmeyen, dağıtılmış bir sistemdir ve sistemi büyük ağlar vasıtasıyla genişletir ve uygulamaları ölçeklenebilir yapar. Ağın her yerinden iletişim, meslektaşları ile akran arasındadır. Bu eşler arası ağ, veriyi birlikte işlemek için bir ağ zinciri olan "Hesaplama Grafiği" ni ortaya çıkarır. Temel hesaplama grafiği kavramı master, düğümler, parametre sunucuları, mesajlar, servisler, konular ve veri çantaları gibi süreçlerden oluşur.

3.3.2.1. Düğümler

Düğüm, hesaplamayı ROS'da yapan en temel birimdir. ROS'taki her düğüm bir işlemde bulunur. Örnek bir robot kolu uygulaması düşünüldüğünde, bir düğüm eksenler arasındaki dönüşümleri kontrol edebilir. Bir tanesi hareket komutlarını kontrol eder ve son düğüm sistemin grafik görüntüsünü kontrol eder. Düğüm birimleri, ROS'un dağıtılmış ve ölçeklenebilir yapısında önemli rol oynamaktadır.

3.3.2.2. Master

Ana kontrol sistemi karşılaştırıldığında, ROS'daki "Master" herhangi bir hesaplama yapmaz. Bunun yerine hesaplama grafiğinin geri kalanı için isim kaydı ve arama sağlar. Tüm düğüm iletişimi, ileti değişimi ve hizmet çağrılarını denetler. Başka bir açıdan bakıldığında, internetin ünlü DNS sistemine benzemektedir.

3.3.2.3. Parametre Sunucusu

Parametre Sunucusu, verilerin merkezi bir konumda anahtar tarafından saklanmasını sağlar. Bir parametre sunucusu, ağ API'ları aracılığıyla erişilebilen, paylaşılan, çok değişkenli bir sözlüktür. Düğümler bu sunucuyu, çalışma zamanında parametreleri depolamak ve almak için kullanır. Yüksek performans için tasarlanmamış olduğundan, yapılandırma parametreleri gibi statik, ikili olmayan veriler için en iyi şekilde kullanılır.

3.3.2.4. Mesajlar

Düğümler arasındaki iletişim mesajlar üzerinden sağlanır. ROS içinde hazır olarak bulunan birden fazla mesaj standardı vardır aynı şekilde özel olarak oluşturulup buraya yenileri de eklenebilmektedir.

3.3.2.5. Konular

Konu, ROS ağındaki bir veri kaynağı sistemidir. Birisi, konuyla ilgili verileri abone edebilir veya yayınlayabilir ve önceden tanımlanmış mesajı ağ üzerinden değiş tokuş edebilir. Birden çok düğüm aynı konuyu aynı anda yayınlayabilir veya abone olabilir (Tabii ki, bu

gerçek zamanlı eşzamanlılık değildir, ancak yeteneği tanımlamak için kullanılır). Örneğin, x, y ve z eksenlerinde açısal konumlar içeren bir başlık adı IMU (Inertial Measurement Unit) olarak adlandırılan atalet ölçüm birimi düşünelim. Donanımdan veri toplayan bir düğüm, alınan veriyi konuya yayınlayabilir. Bundan sonra birçok başka düğüm abone olabilir ve bu konudaki IMU verilerini alabilir. Aynı zamanda ters yönde aynıdır. Farklı düğümler IMU konu yayınlama verilerini besleyebilir. Bir düğüm bu konuya abone olabilir. Konular, düğümlerin mesaj alışverişinde bulunduğu otobüsler olarak adlandırılır. Herkes otobüsten veri gönderebilir veya alabilir.

3.3.2.6. Hizmetler

Hizmet, ROS'da bire bir iletişim uygulamasıdır. Konular birden çok iletişim için iyidir ancak dağıtılmış sistemlerde bire bir iletişimin daha uygun olarak gerçekleştirmek için hizmet servisi kullanılmıştır. Hizmetler istek bildirme mekanizması ile çalışır. Bu istek ve cevap arasındaki veri tipi servis türüne kaydedilir. Bir hizmet talebi yapmak için, bir düğümün ona bir isim vererek hizmet vermesi gerekir. Ardından müşteri düğümü servis türünün istek bölümünü doldurabilir ve çağrı yapar. Servis sağlayıcı düğümü talebi alırken gerekli işlemleri yapar ve yanıtını servis türünün yanıt kısmını doldurarak geri döndürür. Hizmetler, bir ROS istemci kütüphanesi tarafından kaynak kodu derlenen srv dosyaları kullanılarak tanımlanır. Servisler genellikle bir motor sürmek, kolu hareket ettirmek vb. gibi uzaktan kumandalar için kullanışlıdır.

3.3.2.7. Torbalar

Bir torba, ROS'ta mesaj verilerini depolamak için kullanılan bir dosya biçimidir. Düğümler arasında paylaşılan önemli veriler bir mesaj olarak torbalarda saklanabilir. Torbalar, mesajları gerçek eylemler halinde depolamak ve daha sonra benzetimler da ve incelemelerde kullanmak için kullanışlıdır. Örnek olarak, bir lazer mesafe bulucu kullanan bir uygulama verileri gerçek ortamda toplayabilir ve bir torbaya koyabilir. Daha sonra algoritma geliştirme evrelerinde tekrar kullanılabilir.

3.3.3. İsimler

ROS hesaplama grafiğinin düzgün çalışması için, grafikteki her kaynakta uygun bir adın olması gerekir. "Grafik Kaynak Adları" bu hiyerarşik adlandırma yapısını sağlar. Her düğüm, parametre, konu veya hizmetin bu hiyerarşi altında adlandırılması gerekir. Aslında bu isimlendirme kuralları, nesne yönelimli programlama dillerinde ad alanı sözleşmelerine benzer. Örnek olarak, genel ad alanı aşağıdaki gibi tanımlanabilir:

• /

Bu genel ad alanındaki herhangi bir kaynak adı aşağıdaki örnek olabilir:

• /örnekparametre

Bu parametre altında başka kaynaklar da ekleyebiliriz. Bu durumda:

• /örnekparametre/diğerparametreler

tanımlanabilir. Bu örnekte " diğerparametreler", " örnekparametre" ad alanının altında tanımlanmıştır.

Hatırlanması gereken önemli bir nokta, bu kaynakların adlandırma kurallarıdır. "diğerparametre" bir düğüm adı olabilir veya bir parametre adı olabilir değeri tutar.

Ad alanı mantığıyla kaynaklar ad alanlarının altında kendi kaynaklarını oluşturur. Farklı ad alanları arasındaki iletişim mümkündür, ancak genellikle her iki ad alanının da üstünde ve uygun semantiye göre mümkündür. Aksi takdirde isim-alanı mantığının kapsülleme getirisi anlamsız olur.

Kaynaklar kendi ad alanlarının farkında değildir. Böylece hepsi aynı seviyede çalışıyormuş gibi yazılabilirler. Başka bir grafik altında kullanıldıkları takdirde, uygun bir alt grafiğe koymak yeterlidir. Örneğin LazerKütüphaneleri adlı bir uygulamanın Tarayıcı adlı bir düğüme sahip olduğu varsayılsın. Başka biri kendi uygulamasında bunu kullanmak istiyorsa, o zaman LazerKütüphaneleri adlı alt grafiğe eklenmesi yeterlidir. Üst düzey uygulama bir Tarayıcı düğüm adı yazarsa, diğer Tarayıcı düğümüyle çakışmaz. Çünkü adlandırma

• /Tarayıcı

• /LazerKütüphaneleri/Tarayıcı

şeklindedir. Bu isimlendirme sayesinde, kod tekrar kullanımı için bir araştırmacı, kendi uygulamasını etkilemeden kendi uygulamasında Lazerkütüphaneleri paketini kullanabilir ve kodu yeniden kullanarak zamandan tasarruf edebilir.

3.3.3.1. İsimlendirme Uyumu

İsimlerin bu kurallara göre geçerli olması gerekir:

• İlk karakter, alfabe karakteri "[a-z | A-Z]", tilde "~" veya eğik çizgi olmalıdır "/"

• Sonraki karakterler bir rakam ya da alfabe karakteri "[0-9 | a-z | A-Z]", alt çizgi "_" veya Eğik çizgi "/" olabilir

3.4. Bir ROS Paketi Oluşturma

Tanım gereği, ROS'daki paket basit XML dosyasından alınan bir klasördür. ROS'ta paket oluşturmak için kullanılacak araç

• $ catkin_create_pkg

olarak çalıştırılır. Bu komut, klasör ve xml yapısını oluşturur. Ancak yapıyı anlamak için paketi manuel olarak da oluşturulabilir.

3.4.1. Klasör Yapısı

Herhangi bir eylemden önce, ortam ayarlarının yapılması gereklidir. Örneğin "LazerSürücü" isimli bu paket için aşağıda komutlar kullanılarak cetkin çalışma alanında "LazerSürücü" adlı klasörü oluşturulması ve klasör dizisine girmesi gerekir.

• mkdir -p src / LazerSürücü (Klasör oluşturma)

• cd src / LazerSürücü (Oluşturulan klasöre girmek)

Daha sonra klasörün içinde "package.xml" ve txt dosya adlarını "CMakeLists.txt" olarak adlandırılan bir XML dosyasının oluşturması gereklidir.

3.4.2. Paket Manifest

Paket manifest, paket adı, sürüm, yazar, tarih, iç ve dış bağımlılıklar gibi bilgileri içeren basit bir XML dosyasıdır. Örnek bir XML dosyası içeriği Şekil 3.1'de gösterilmiştir.

Eğer doğru paket oluşturulmuşsa ve ortam düzgün bir şekilde kurulursa, ROS yeni oluşturulan paketi bulacaktır. "Rospack" komutu "LazerSürücü" paketi örneğini bulacaktır. Çalıştırma aşağıdaki gibi gerçekleştirilebilir.

• $ rospack find LazerSürücü

Şekil 3.1'deki örnekte XML yapısı LazerSürücü paketi roscpp ve std_msgs paketlerine bağlıdır. Bunlar paketi oluşturmak için sisteme kurulmalıdır.

```
>> <package>
    <name>LazerSurucu</name>
    <version>1.0.0</version>
    <description>
    Example package for ROS.
    </description>
    <maintainer>Selman Akyol</maintainer>
    <license>BSD</license>

    <buildtool_depend>catkin</buildtool_depend>

    <build_depend>roscpp</build_depend>
    <build_depend>std_msgs</build_depend>

    <run_depend>roscpp</run_depend>
    <run_depend>std_msgs</run_depend>
</package>
```

Şekil 3.1. Paket .xml dosya yapısı

3.4.3. Paket Yapısı Yapılanışı

Düzgün hazırlanmış bir paket oluşturmak için "CMakeLists.txt" dosyası gereklidir. Bu dosya catkin tarafından kullanılır ve CMake yapılandırmasını sağlar. Paketi için örnek CMakeLists.txt dosyası LaserSürücü Şekil 3.2'de gösterilmiştir.

```
>> cmake_minimum_required (VERSION 2.8.3)
project (LazerSurucu)
find_package (catkin REQUIRED roscpp std_msgs)
catkin_package()
```

Şekil 3.2. Derleme işlemi için temel kurulum ile CMakeLists.txt dosyası

Burada "CMakeLists.txt" dosyası ikili derleme işlemi için gereklidir. Bununla birlikte, genellikle paket manifestinde bildirilen sistem bağımlılıkları "find_package ()" işlevinin içindeki CmakeLists.txt dosyasında yer almaktadır.

4. EŞ ZAMANLI KONUM BELİRLEME VE HARİTALAMA

Otonom araçlardan mikro robotlara kadar her hareketli otomatik sistem hem çevresini hem de çevresindeki konumunu bilmelidir. SLAM, bilinmeyen bir ortamı haritalamak ve eş zamanlı olarak oluşturduğu bu haritada bulunduğu konumu belirlemek için kullanılan bir yöntemdir. SLAM, son yıllarda aktif araştırma alanı olmuştur. Sadece araç ya da robotlar için değil, aynı zamanda denizaltı izleme, arkeoloji gibi diğer araştırma alanlarını incelemek için SLAM kullanılmıştır.

SLAM sorununun kendi içinde bir ikiliği vardır. Bilinmeyen bir ortamı bir araç ile haritalamak için aracın konumunu bilmesi gerekir. Aracının konumunu bilmesi için aynı şekilde çevre haritasını görmesi gerekmektedir. SLAM için önerilen bir çözüm, bu ikiliği ortadan kaldırmalı, haritalama ve konum belirleme problemi için bir çözüm getirilmelidir. Aksi takdirde, önerilen yöntem yalnızca haritalama veya yalnızca konum belirleme için bir çözüm olacaktır.

Uzun süren araştırmalar bu yönde devam etmiştir ve araştırmalar yalnızca haritalama veya yalnızca konum belirleme problemlerine odaklanmıştır. Başlangıçta arazi işaretindeki ve araç konumlarındaki tüm hataların bağımsız olduğunu ve ilintinin en aza indirgenmesi gerektiği düşünülmüştür. Fakat daha sonra, araç konumu hataları ve döngüsel tahminlerin yüksek derecede ilintiye sahip olduğu ve bunun da tek çözüme yol açtığı anlaşılmıştır. Böylece, SLAM probleminin tekli haritalama veya konum belirleme probleminden ziyade tekli olasılık probleminde ele alınması gerektiği ortaya çıkmıştır.

4.1. SLAM Problemine İstatistiksel Yaklaşımlar

SLAM probleminin çözümüne istatistiksel yöntemler ile ulaşılabilir. SLAM haritalama yöntemleri, kullanılan harita çeşitlerine göre farklılık gösterebilir. Çoğunlukla kullanılan ölçümlü haritalar için tasarlanmış olan gelişmiş yöntemler robotların istatistiksel yöneliş modellemeleri ve belirli olmayan kavrama modellemelerinin baz alındığı istatistiksel verilerdir [25]. Bu verilerin, Kalman Filtresi, Parçacık Filtresi ve benzeri bir filtre olan Bayes Filtresi ne giriş olarak verildiği zaman, robot kordinat bilgileri oluşturulur. Çoğunluk ile istatiksel haritalama sorununa ek seçenek çözüm olarak kullanılabilir. İstatistiksel yaklaşımlarda ilk yaklaşım önemli bir yere sahiptir. İlk yaklaşım, olan tüm istatiksel yaklaşımların ilk istatiksel yaklaşımın temel alınması manasına gelmektedir. Yaklaşımların sonlandırılması, topolojik olarak oturtulmuş bir modellemenin gerçekleştirilmesi için var olan

haritanın yeniden oluşturulması yöntemi ile oluşan haritaların sınırlandırılmasını sağlar. SLAM haritalandırılmasının sonuca ulaştırılması için birçok yöntem geliştirilmiştir.

SLAM haritalama sorununun çözümünde başvurulan yöntemlerin özellikleri önemli bir yere sahiptir. Yönteme uygun sensör seçimi de yöntemi etkileyen temel özeliklerden biridir. Bir sensörün özeliğini belirleyen üç temel durum aşağıda verilmiştir.

1-Gürültü

2-Giriş verisinin boyutu

i. İki boyutlu lazer tarayıcılarda 2B noktalar

İİ. Üç boyutlu lazer tarayıcılarda 3B nokta kümeleri

İİİ. Kameralarda ise kamera özellikleri

3-Referans çerçevesinin boyutu

i. Lazer tarayıcı ve kamera da robotun kendi referans çerçevesi

ii. GPS'de dünyanın koordinat çerçevesi

iii. İvme ölçer ve jiroskopta ataletsel koordinat çerçevesi

4.2. SLAM Çerçevesinin Uygulamaları

4.2.1. Kalman Filtresi Tabanlı SLAM

Robotik uygulamalarda, Kalman Filtresi tabanlı yöntemlere sıklık ile başvurulmaktadır. Kalman filtresi robot konumunun kestirimi ve ortamın haritasını aynı anda yapabilen bir yöntemdir. Kalman filtresi Gauss dağılımı ile beraber sonsal değerlerin p($r_k, m \backslash z^k, u^k$) temsilidir [26],[27]. Robotik haritalama yöntemlerinde, *x* durum vektörü, *r* robot konumu ve *m* haritasından oluşur.

$$x_k = (r_k, m)^T \quad \textbf{(4.1)}$$

İki boyutlu düzlemde *r* robot konumu üç değişken ile gösterilmektedir. Robot konumu r_x, r_y kutupsal gösteriminde yer alan koordinat değerleri ile r_θ açısından oluşur. Kalman fitresinde haritalar, nesnelerin kutupsal kordinatlardaki konumlarını içerir. Haritada bulunan nesnelere referans olarak ortamda var olan konum işaretçileri, ayırt edici niteliğe sahip olan nesneler ile birlikte şekiller gösterilebilir. Haritada *N* adet nesne olduğu varsayılır ise durum vektörünün büyüklüğü *2N+3* olur.

$$x_k = (r_{x,k}, r_{y,k}, r_{\theta,k}, m_{1,x,k}, m_{1,y,k}, m_{2,x,k}, m_{2,y,k}, \dots, m_{d,x,k}, m_{d,y,k})^T \quad \textbf{(4.2)}$$

Burada; yazılan $m_{d,x,k}$ ve $m_{d,y,k}$ haritadaki d numaralı nesnenin koordinatlarıdır. Dolayısıyla ortalama vektörünün boyutu *2N+3* ve kovaryans matrisinin boyutu da $(2N+3)^2$ olacaktır.

Kalman filtresi tabanlı haritalama algoritması için üç varsayım yapılmıştır [28]. İlk varsayım doğrusal bir model olup Gauss dağılımlı gürültüye sahip olmasıdır. İkinci varsayım aynı özelikler kavramsal model içinde geçerli olmalıdır. Son varsayım başlangıç durum belirsizliği Gauss dağılımına sahip olmalıdır.

Haritalama probleminde r_k robot konumu ve m_k harita doğrusal olarak bir önceki r_{k-1} robot konumu, m_{k-1} harita ve u_k kontrolüne bağlıdır. Geçen zaman içerisinde haritanın değişmediği kabul edilmektedir. Doğrusal olmayan bir davranışta robot pozu r_k bir önceki robot pozu olan r_{k-1} ve u_k kontrolüne bağlıdır.

Bu varsayımlar ve Bayes filtresi göz önünde bulundurularak elde edilen Kalman Filtresi denklemleri

$$x_k = F_k x_{k-1} + B_k u_k + \varepsilon_k \quad \textbf{(4.3)}$$

$$z_k = H_k x_k + v_k \quad \textbf{(4.4)}$$

Burada; B_k kontrol vektörüne uygulanan kontrol giriş modeli, ε_k sıfır ortalamalı normal dağılımlı sistem gürültüsü, F_k bir önceki duruma uygulanan durum geçiş modeli, H_k gözlem modelini ve v_k sıfır ortalamalı Gaussian beyaz gürültülü gözlem gürültüsünü göstermektedir.

$$\hat{\mathrm{x}}_{\mathrm{k|k-1}} = F_k \hat{\mathrm{x}}_{\mathrm{k-1|k-1}} + B_k u_k \quad \textbf{(4.5)}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \quad \textbf{(4.6)}$$

Filtrelerin durumu iki değişkenle gösterilir. Bunlar $\hat{x}_{k|k}$ sonsal durum kestirimi $P_{k|k}$ sonsal hata kovaryans matrisidir.

$$\hat{y}_k = z_k - H_k \hat{\mathrm{x}}_{\mathrm{k|k-1}} \quad \textbf{(4.7)}$$

$$S_k = H_k \mathrm{P}_{\mathrm{k|k-1}} H_k^T + R_k \quad \textbf{(4.8)}$$

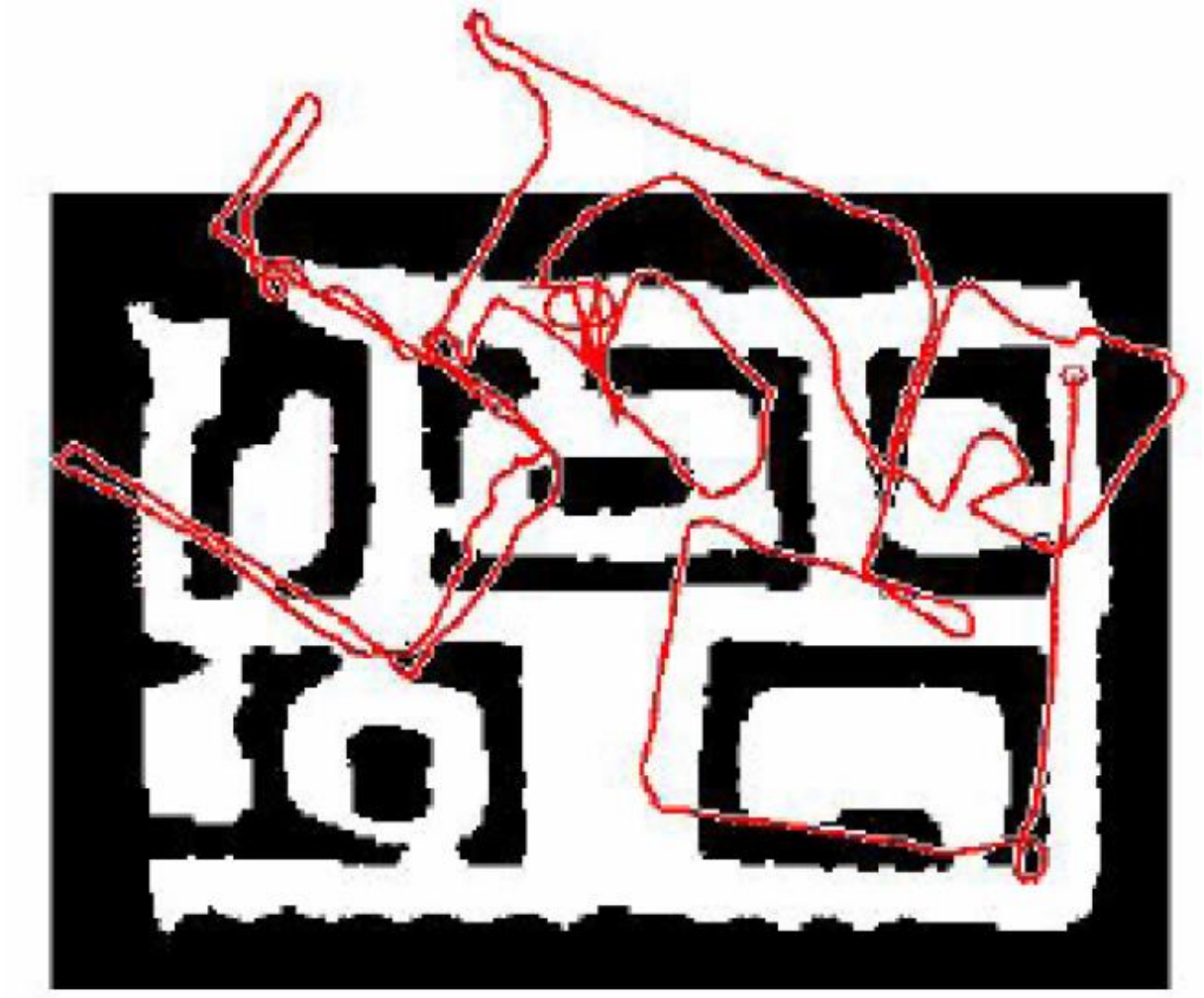
$$K_k = \mathrm{P}_{\mathrm{k|k-1}} S_k^{-1} \quad \textbf{(4.9)}$$

$$\hat{\mathrm{x}}_{\mathrm{k|k}} = \hat{\mathrm{x}}_{\mathrm{k|k-1}} + K_k \hat{y}_k \quad \textbf{(4.10)}$$

$$P_{k|k} = (\mathrm{I} - K_k H_k) P_{k|k-1} \quad \textbf{(4.11)}$$

Burada; K_k Kalman kazancıdır.

Kalman Filtresi çok fazla kullanılmasına rağmen sakıncaları da çoktur. Birincisi; algılayıcı verilerini doğrudan kullanamaz belirli ön işlemlerden geçirmek gerekmektedir. İkincisi; algılayıcı verisinin gürültüsü Gauss şeklinde olmalıdır. Üçüncüsü; başlangıçta robot pozisyonunda en ufak hatadan dolayı zaman geçtikçe hata büyüyecek ve yanlış haritalar çıkacaktır. Şekil 4.1'de görüldüğü gibi başlangıç pozisyonundaki hatadan dolayı kötü bir harita örneği verilmiştir. Son olarak, Kalman Filtresinin doğrusal olan fonksiyonlara uygulandığında iyi sonuçlar verdiği görülmüştür ancak doğrusal olmayan fonksiyonlara uygulanamadığı için en büyük sakıncalardan biri olmuştur.



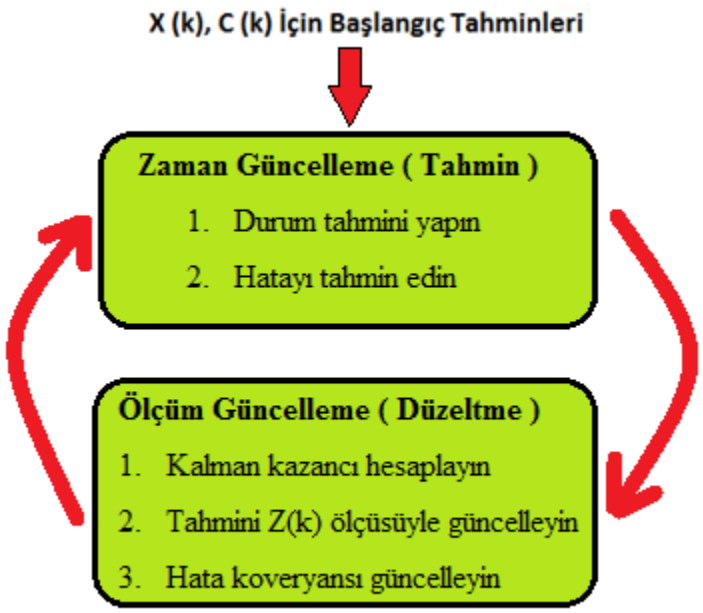
Şekil 4.1. Başlangıç pozisyonundaki hatadan dolayı kötü bir harita örneği [28]

4.2.2. Genişletilmiş Kalman Filtre Tabanlı SLAM

Genişletilmiş Kalman Filtresi (GKF) Kalman Filtresinin özel bir şeklidir. Kalman Filtresindeki doğrusalsızlık probleminin üstesinden gelmek için tasarlanmıştır. Doğrusal olmayan doğasına rağmen, çoğu durumda doğrusal varsayımlar kullanarak Kalman Filtresi ile başarılı sonuçlar alınabilir. Ancak bazen doğrusalsızlıklar ortadan kaldırılamaz. Dairesel yörüngeler gibi doğrusal olmayan durumlar doğrusal Kalman Filtresi ile iyi bir şekilde hesaplanamaz. GKF, doğrusal olmayan durum geçişi ve ölçüm geçiş fonksiyonunu doğrusallaştırarak bu durumların üstesinden gelir.

GKF; genel olarak kestirim, durum tahmini, ölçüm tahminleri, gözlem ve eşleştirme adımlarından oluşur. Şekil 4.3 GKF tabanlı SLAM algoritmasının adımları verilmiştir. GKF için yer belirleyici tabanlı bir harita kullanılır ve yinelemeli olarak iki aşamada gerçekleştirilir. Tahmin aşamasında robot hareket modeli ve u_k kontrolü yapılarak robotun

konum tespiti yapılır. Güncelleme kısmında ise yer işaretçisinin pozisyonunu yenilemek ve robotun konumunu belirlemek için harici bir sensörden gelen z_k gözlemi kullanılır. Şekil 4.2'de bu döngüler gösterilmiştir.



Şekil 4.2. GKF süresi ve durum güncelleme döngüleri

GKF için kullanılan ayrık zamanlı doğrusal olmayan model

$$x_k = f(x_{k-1},\ u_{k-1}) + w_{k-1} \tag{4.12}$$

olarak tanımlanmıştır [38]. Burada; w_{k-1} sistem modelindeki Q_k kovaryanslı Gauss dağılımlı gürültüyü ve f fonksiyonu, doğrusal olmayan durum geçiş fonksiyonunu tanımlamaktadır. Doğrusal olmayan gözlem modeli

$$z_k = \mathrm{h}(x_k) + v_k \tag{4.13}$$

olarak tanımlanmıştır. Burada; v(k) sıfır ortalamalı R_k kovaryanslı Gauss dağılımlı ölçüm gürültüsünü göstermektedir. h fonksiyonu doğrusal olmayan gözlem fonksiyonudur. Durum tahmin denklemi

$$\hat{\mathrm{x}}_{\mathrm{k|k-1}} = f(\hat{\mathrm{x}}_{\mathrm{k-1|k-1}}, u_{k-1}) \quad \textbf{(4.14)}$$

olarak belirlenir. Tahmin edilen kovaryans matrisi

$$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F_{k-1}^{T} + Q_{k-1} \quad \textbf{(4.15)}$$

olarak tanımlanır. Burada; F_{k-1} durum geçiş matrisinin jakobienini, Q_{k-1} modeldeki gürültüyü göstermektedir. Robotik uygulamalarda ve insansız araçlarla gerçekleştirilen SLAM uygulamalarında kaçınılmazdır. Bir gözlem gerçekleştirildiğinde güncellenmiş durum vektörü

$$\hat{\mathrm{x}}_{\mathrm{k|k}} = \hat{\mathrm{x}}_{\mathrm{k|k-1}} + K_k\hat{y}_k \quad \textbf{(4.16)}$$

olarak güncellenir. Burada; y_k yeniliği gösterir. Yenilik denklemi

$$\hat{y}_k = z_k - h(\hat{\mathrm{x}}_{\mathrm{k|k-1}}) \quad \textbf{(4.17)}$$

olarak tanımlanır. Kalman kazancı ile yenilik kovaryans matrisleri

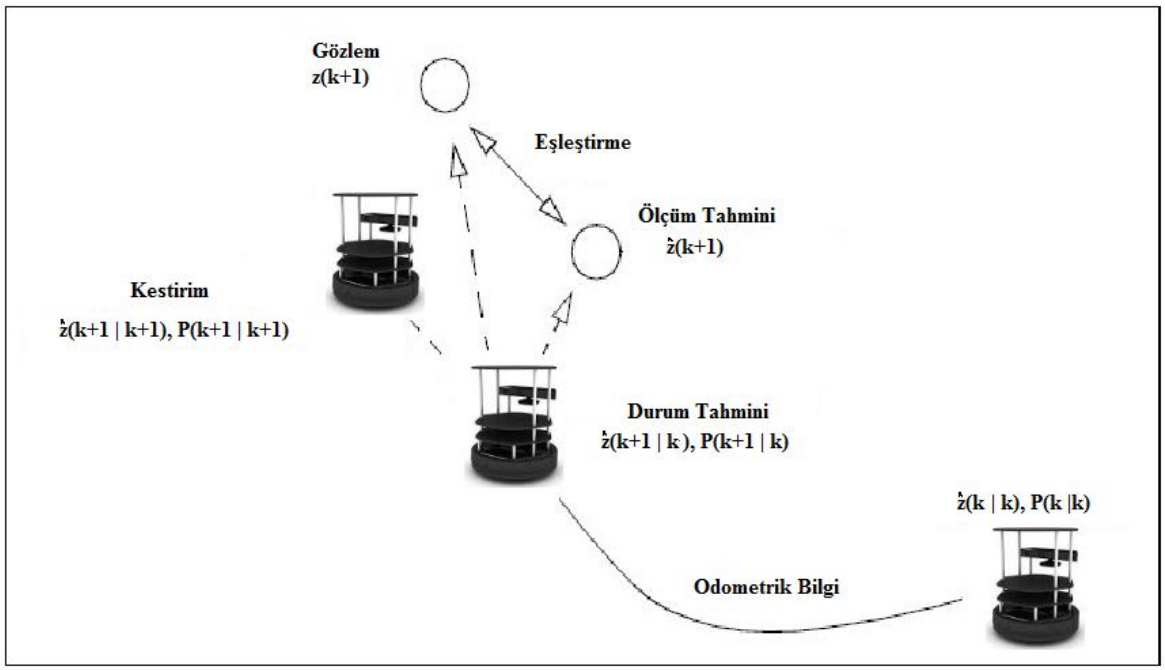
$$K_k = \mathrm{P}_{\mathrm{k|k-1}}H_k^{T}S_k^{-1} \quad \textbf{(4.18)}$$

$$S_k = H_k\mathrm{P}_{\mathrm{k|k-1}}H_k^{T} + R_k \quad \textbf{(4.19)}$$

olarak tanımlanır. Burada; H_k gözlem modelinin jakobienini, R_k da gözlemdeki gürültüyü göstermektedir. Güncellenmiş kovaryans matrisi denklemi

$$P_{k|k} = (\mathrm{I} - K_kH_k)P_{k|k-1} \quad \textbf{(4.20)}$$

olarak tanımlanmıştır.



Şekil 4.3. GKF tabanlı SLAM algoritmasının adımları

Kalman Filtresi tabanlı SLAM algoritmaları iki sakıncası vardır. Birincisi durum vektörüne eklenen yer işaretçilerinden dolayı hesaplama karmaşıklığının artmasıdır. İkinci sakıncası ise yanlış veri eşleştirmelerinin filtrede sapmaya neden olmasıdır [29]. Kalman Filtresi tabanlı SLAM algoritmaları hesaplamada karmaşıklığa neden olduğu için durum vektörüne eklenebilecek yer işaretçiler sınırlı sayıdadır [28],[30].

N parametresi, ortamda bulunan yer işaretçilerinin sayısını gösterirse GKF'nin her bir yenileme adımı O(N^2) seviyesinde karmaşıklık oluşturur. GKF'nin toplam hesaplama karmaşıklığı ise O(N^3) seviyesindedir [31], [32]. Bu nedenle Kalman Filtresi tabanlının dış ortamlarda yapılan uygulamalarda kullanılabilmesi için hesaplama karışıklığı ve bellek ihtiyacını azaltan iyileştirmeler tavsiye edilmiştir [33]. Sıkıştırılmış Genişletilmiş Kalman Filtresi [32], Seyrek Ağırlık Kalman Filtresi [34] ve harita birleştirici SLAM [35] bu onarımlardan bazılarıdır. Bu yöntemler, istatistiksel haritanın yerel kısımlarında çalışarak toplam işlem yükünü düşürmeyi hedefler [36].

4.2.3. Beklenti Enbüyültme Tabanlı SLAM

SLAM problemi uygulamalarında Kalman Filtresine farklı seçenekler sunmak için Beklenti Enbüyültme (BE) algoritması geliştirilmiştir. BE algoritması, en büyük olabilirlik tahmin modelinden ortaya çıkarılmış, gizli değişkenlere sahip olasılıksal bir algoritmadır [14].

BE algoritmasında artımlı bir model kullanılarak veri işlendiği için birkaç defa işlenir. Bir başka deyişle model yalnız bir önceki adımdaki duruma bağlı değil, o ana kadar olan tüm adımlardaki durumlara bağlıdır. BE'nin Kalman filtresinden en önemli ve en belirgin farkı bu özeliktir.

BE algoritmasında yapılan bir harita için robot konumunun sonsal durumunun hesaplandığı bir beklenti B-adımı ve elde edilen robot konumu kullanılarak en olası haritanın hesaplandığı bir en büyültme E-adımı bulunmaktadır. Sonuçta gittikçe kesinleşen ve gerçek ortama benzeyen bir dizi harita elde edilmektedir.

BE algoritmasının diğer yöntemlere göre en büyük getirisi veri ilişkilendirme problemini çözmesidir. BE yöntemiyle sunulan çözüm şimdiye kadar yapılmış en iyi çözüm olarak görülmüştür. BE eş zamanlı olarak performans gösteren bir yöntem değildir. Şekil 4.4'te görülen harita BE ile elde edilmiştir, ancak bu haritanın elde edilmesi için BE algoritması saatlerce çalıştırılmıştır [28].

Haritalar m[0] boş bir harita olmak üzere m[0]; m[1]; m[2]; … ile gösterilmiştir [18].

Şekil 4.4. Beklenti Enbüyültme ile yapılmış harita örneği [28]

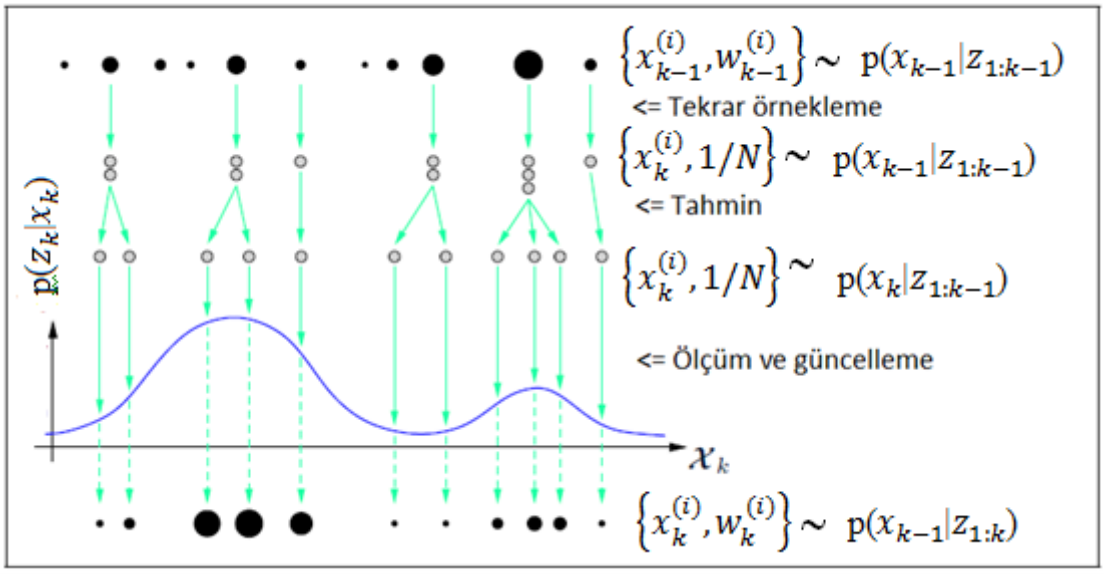
4.2.4. Parçacık Filtre Tabanlı SLAM

Durum gözlemleri ve tahminleri için Kalman Filtre Tabanlı Algoritmaların işe yaradığı gözlenmiştir. Fakat Kalman Filtresi doğası gereği bir doğrusal tahmin edicidir. Bu nedenle doğrusalsızlığın nerede olduğu konusunda başarılı değildir. Doğrusal olmayan durumları doğrusallaştıran GKF olsa da, yine de Gauss varsayımlarına dayanmaktadır. Bu durumun üstesinden gelmek için doğası gereği tamamen doğrusal olmayan bir filtre gereklidir. Parçacık Filtresi bu amaçla önerilmiştir. Bilimsel yazında Parçacık Filtresinin çeşitli türevleri bulunmaktadır. Bu yöntemin temeli tüm olasılık yoğunluğu fonksiyonları örneğe dayalı olarak temsilini oluşturmaktır [37], [38]. Bir modeli baz alarak gerçekleştirilen uygulamalar değişkenin durumunu değiştirmektedir. Farklı aralıklarla elde edilen gözlemlerin ise değişkenin durumunu kısıtladığı görülmüştür. Birçok parçacık kullanılarak değişken temsil edilmiştir. Bu parçacıkların her birini temsil etmek için birer ağırlık değeri verilerek parçacıkların kalitesi belirlenmiştir. Parçacıkların ağırlıklarını toplayarak değişkenin kestirimi yapılmıştır. Parçacık Filtreleme algoritmaları tahmin ve güncelleme adımlarıyla yinelemeli olarak gerçekleştirilmiştir.

Tahmin: Yapılan her hareketin ardından her bir değişken için bir model göz önünde bulundurularak değişkenler değiştirilir. Değişkenler üzerindeki gürültünün benzerini yapmak için rastgele olarak belirlediği gürültü değerini eklemektedir.

Güncelleme: Algılayıcıdan alınan son bilgilere göre tüm ağırlıklar tekrardan değerlendirilir. Bu ağırlıklardan belirli anlarda düşük değere sahip olanları örnekleme olarak bilinen işlemle ortadan kaldırılır.

Şekil 4.5 ile gösterilen yöntemle, N sayıdaki durum örneği ve bunlara karşılık gelen ağırlık dizisi $\left\{x_{k-1}^{(i)}, w_{k-1}^{(i)}, i = 1, \dots, N\right\}$ k-1 anındaki sonsal yoğunluğu p($x_{k-1} \mid z_{1:k-1}$) temsil ettiği varsayımı altında k anı için p($x_k \mid z_{1:k}$) sonsal yoğunluğu temsil edecek parçacık setini güncellemek için izlenmiştir.



Şekil 4.5. Bir parçacık kümesinin doğrusal olmayan dağılımı (Parçacık ağırlıklarına karşılık gelen daire çapları) [38]

4.3. Rao-blackwellized Parçacık Filtresi Kullanarak Harita Oluşturma

Rao-Blackwellized filtre yöntemi SLAM problemlerinde kullanılmasında ana fikir, *m* harita, $x_{1:k}=x_1, \ldots, x_k$ robotun gezinmesi, $z_{1:k}=z_1, \ldots, z_k$ gözlemler, $x_{1:k-1}=x_1, \ldots, x_{k-1}$ mobil robotun odometri ölçümleri $p(x_{1:k}, m \mid z_{1:k}, , u_{1:k-1})$ robotun anlık konumunu tahmini olarak tutmaktır [39]. İlgili denklem sadeleştirilirse

$$p(x_{1:k}, m \mid z_{1:k}, u_{1:k-1}) = p(m \mid x_{1:k}, z_{1:k}) \cdot p(x_{1:k} \mid z_{1:k}, u_{1:k-1}) \quad \textbf{(4.17)}$$

olarak tanımlanır. Böylece robotun gezdiği yerlerin tahminini ve bu tahmin üzerinden hesaplanan haritayı verir. Harita çıkarımı robotun pozisyon tahminine bağlı olduğu için bu yöntemle verimli bir hesaplama yöntemi ortaya konmuş olur. Potansiyel yörünge üzerinden $p(x_{1:k} \mid z_{1:k}, u_{1:k-1})$ pozisyon tahmini için parçacık filtresi uygulanır. Bu filtrelemede her bir parçacık robotun potansiyel yörünge bilgisini içermektedir. Ayrıca her bir yörünge birbirinden bağımsız olarak harita bilgisini de içerir. Harita, parçacıklara karşılık gelen yörünge ve izlenimler sonucu elde edilir. Haritalama için kullanılan Rao-Blackwellized filtre algılayıcılardan okunan sonuçları işler ve mümkün olan odometri taramalarını gerçekleştirir. Bu verileri kullanarak parçacıkların robotun yörüngesine ait istatistiksel tahminlerini yeniler. Aşağıda yapılan adımlarda açıklanmıştır.

-Örnekleme: Bir sonraki parçacıklar $\left\{x_k^{(i)}\right\}$, π tavsiyesi örneklenerek şimdiki kuşaktan $\left\{x_{k-1}^{(i)}\right\}$ elde edilir. Genellikle, istatistiksel odiyometri hareket modeli, dağılım için kullanılır.

-Ağırlıklandırma: (4.18) numaralı denklemde verilen ehemmiyet örnekleme tarzına göre her bir parçacığa ağırlık dağılımı $w_k^{(i)}$ tayin edilir.

$$w_k^{(i)} = \frac{\mathrm{p}(x_{1:k}^{(i)} \mid z_{1:k}, u_{1:k-1})}{\pi(x_{1:k}^{(i)} \mid z_{1:k}, u_{1:k-1})} \quad \textbf{(4.18)}$$

-Yeniden Örnekleme: Her bir parçacık kendi ağırlığına göre çizilir. Sürekliliğe yaklaşabilmek için bu adımda yeterli sayıda sonlu parçacık kullanılır. Yeniden örneklemede hedef dağılım önerisi farklı oluğunda yeniden parçacık filtresi uygulanır. Yeni baştan örneklenen bütün parçacıklar aynı ağırlığa sahip olur.

- Harita Çıkarma: Her parçacık için, o parçacığa denk gelen harita tahmini $\mathrm{p}\left(\mathrm{m}^{(\mathrm{i})} \left| x_{1:k}^{(i)}, z_{1:k}\right.\right)$, $z_{1:k}$ izleme geçmişine ve $x_{1:k}^{(i)}$ yörünge örnekleri baz alınarak hesaplanır.

4.4. Gmapping

Gmapping lazer mesafe sensöründen ızgara haritalama öğrenmek için Reo-Blackwellized Parçacık Filtresini etkili adımlarla uygulayan ROS paketidir. Başlangıçta tercih edilen dağılımda karmaşık işlemleri azaltacak biçimde normalleştirmeler yapılarak dağılım alanı küçültülmüş ve beklenen sonuçlara ulaşılmıştır. Daha sonra, tekrardan örnekleme bölümünde bir adaptasyon yapılarak, tekrar örnekleme esnasında tüm parçacıkların verilerini kaybetmemesi, onun yerine düşük önceliği olan parçacıkların yok edilmesi ve yüksek öncelikli parçacıklardan oluşturulan yeni parçacıkların yerini alması sağlanmıştır. Böylelikle yüksek öncelikli parçacıklar tekrardan örnekleme esnasında yok olmayacağı için hesap karmaşıklığı yeterince azalmış olacaktır [40]. Bu çalışmalar neticesinde Gmapping'in çözüme ilişkin adımları aşağıdaki gibidir.

1) Robot konumunu temsil eden i numaralı parçacık, başlangıç tahmini $x_k^{\prime(i)} = x_{k-1}^{(i)} * u_{k-1}$ olacak biçimde bir önceki parçacığın konumundan $x_{k-1}^{(i)}$ ve en son filtre güncellenmesi esnasında alınan odometri ölçülerinden oluşur. Bu formüllerde kullanılan * operatörü standart konum birleştirme operatörüdür.

2) $m_{k-1}^{(i)}$ haritasında başlangıç konumu, $x_k'^{(i)}$ olacak biçimde tarama eşleme algoritmaları gerçekleştirilir. Tarama başlangıç konumu olan $x_k'^{(i)}$ çevresinde sınırlar olacak biçimde oluşturulur. Eğer tarama eşleme hata verirse konum ve ağırlıklar hareket sistemine göre hesaplanır ve 3. ve 4. adım atlanır.

3) Tarama eşleme verdiği sonuçlar ile robotun konumunun çevresinde bir örnekleme noktalarının kümesi alınır. Buna göre, önerinin ortalama değeri ve kovaryans matrisi, örneklenen x_j konumlarında hedef dağılım p($z_k \mid m_{k-1}^{(i)}, x_j$) p($x_j \mid x_{k-1}^{(i)}, u_{k-1}$) noktasal olarak baz alınarak hesaplanır. Bu sırada ağırlık vektörü $\eta^{(i)}$ de hesaplanır.

4) i parçacığının yeni konumu $x_k^{(i)}$, varsayılan dağılımın iyileştirilmişi olan Gauss yaklaşımı N($\mu_k^{(i)}, \sum_k^{(i)}$) sayesinde çizilir.

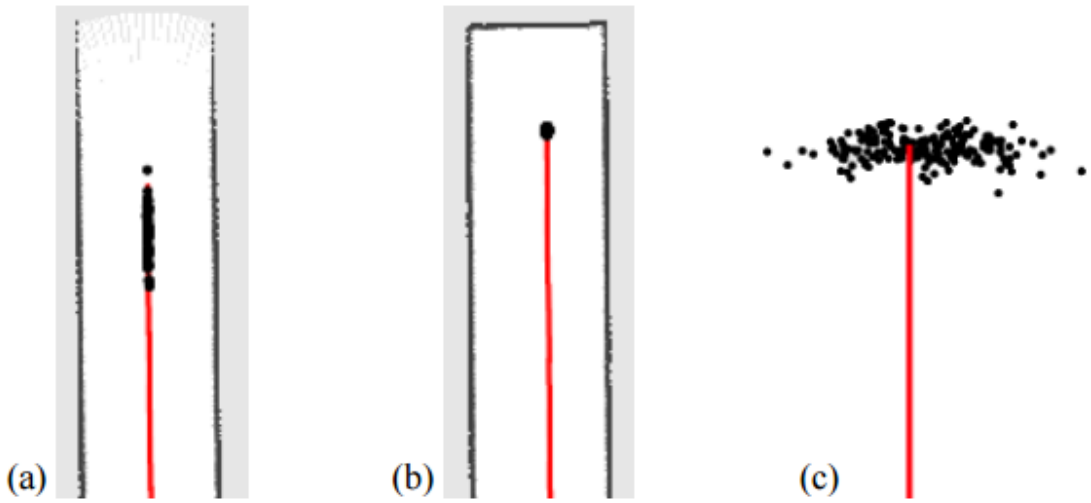
5) Parçacıkların ağırlıkları önemli olma durumuna göre güncelleştirilir.

6) i parçacığının haritası $m^{(i)}$, beklenilen $x_k^{(i)}$ konumu ve z_k gözlemine göre iyileştirilir.

Yeni nesil örneklerin hesap edilmesinden sonra tekrardan örnekleme adımı N_{eff} sonucuna bağlı olarak oluşturulur. N_{eff} değeri; $w'^{(i)}$, i parçacığının normalleştirilmiş ağırlığı

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w'^{(i)})^2} \quad \textbf{(4.19)}$$

olarak hesaplanır. Burada; N parçacık sayısıdır.



Şekil 4.6 a) Açık koridor üzerinde öneri dağıtımı b) Çıkmaz koridor üzerinde öneri dağıtımı c) Odometri modelini varsayarak öneri dağıtımı [40].

5. UYGULAMA

Tezin bu bölümünde yaptığımız uygulamalarda kullanılan malzemeler tanıtılmış ve yapılan tüm çalışmalar ayrıntılı olarak anlatılmıştır.

5.1. RP-LIDAR ile Elde Edilen Verilerin MATLAB ile Görselleştirilmesi

5.1.1. Lazer Mesafe Ölçümü

Etkime ile ışımanın varlığını ilk defa 1917 yılında Albert Einstein tarafından öne sürülmüştür. 1960 yılında Theodore Maiman optik frekansta lazer hareketini gerçekleştirmiş ve yakut lazerinin varlığını kanıtlamıştır. Bu olaydan sonra lazer kullanımında oldukça önemli gelişmeler olmuştur. Günlük hayattaki ilk kullanımı 1974 yılında olmuştur. Süpermarketlerin barkod okuyucuları, daha sonra 1982 yılında tanınan lazer disk okuyucu ve kompakt disk çalarlar ilk lazer donanımlı cihazlardır. Çoklu ortam sunumlarında, reklamcılıkta, açık hava mekanlarının vitrin düzenlemesinde, oyunların özel efektlerinde, müzelerde, kulüplerde, konserlerde, tıpta, iletişim alanlarında lazer kullanılmıştır [41]. Bunların ötesinde lazer ışını endüstriyel süreçlerde, mühendislik alanında, tıpta, bilimsel araştırmalarda, meteorolojide, iletişimde, holografide ve savunma donanımlarında kullanılmıştır.

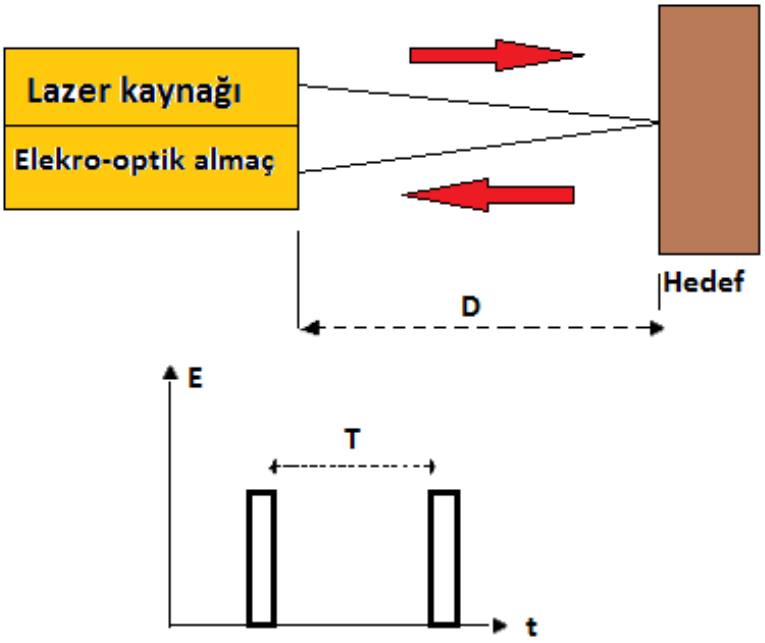
5.1.2. Lazer Sensörün Çalışma İlkesi

Lazer mesafe sensörleri yaydıkları lazer ışınlarının engellere çarpıp geri dönmesine kadar geçen süreyi hesaplayarak aradaki uzaklığı belirleyebilen sensörlerdir. Lazer enerjisini kısa sürede ve belirli bir menzildeki nesnelere doğru yollanır. Sensörün menzili içinde bulunan herhangi bir nesne yollanan bu ışık enerjisinin belirli bir bölümü geri yansıtılır. Geri gelen ışık enerjisi sensörün algılayıcısı tarafında algılanır. Sensörden çıkan ışık enerjisi nesneye çarpıp geri gelmesi için geçen süre hesaplanır. Lazerden çıkan bu ışık demetinin hızı ışık hızı bilindiğinden nesne ile sensör arasındaki mesafeyi kolaylıkla bulunabilir. Nesneden yansıyan ışık demetini sensörde bulunan alıcı sayesinde algılanması sağlanır. Şekil 5.1'de sensörün basit olarak nasıl çalıştığı gösterilmektedir.

Nesne ile sensör arasındaki uzaklık

$$D = \frac{C.T}{2} \quad \textbf{(5.1)}$$

olarak hesaplanır. Burada; *D* nesne ile sensör arasındaki uzaklık, *C* ışık hızı ve *T* peryot olarak göstermektedir.



Şekil 5.1. Lazer mesafe ölçümü

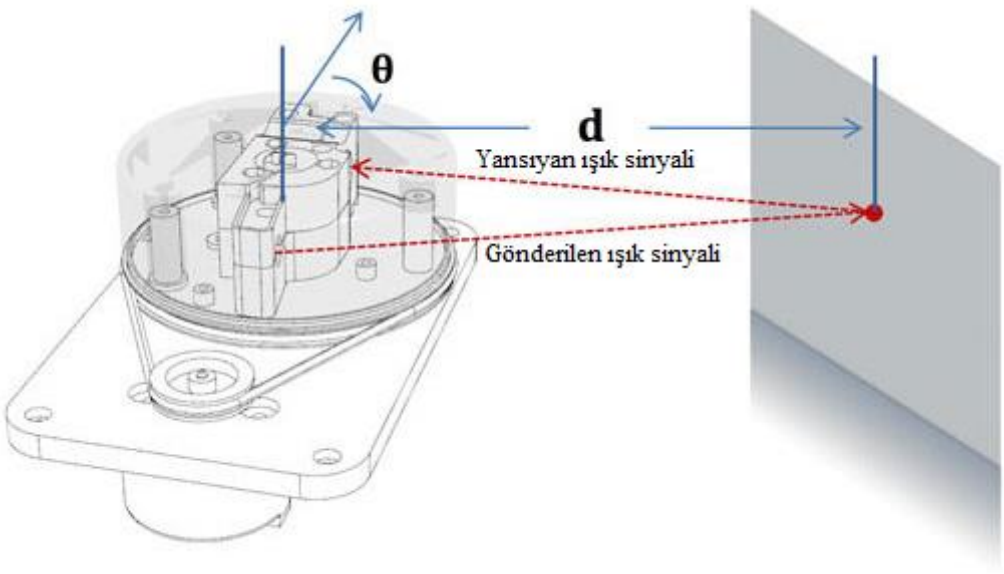
Mobil robotlarda en fazla tercih edilen lazer çeşidi 2B olarak adlandırılan hem nesnenin lazere olan mesafesi hem de lazerle olan açısal konumunu veren çeşitlerdir. Uygulamamızda Robo Peak firması tarafından üretilmiş Şekil 5.2'deki RP-LIDAR [42] lazer sensörü kullanılmıştır. Bu sensörün özelikleri Tablo 5.1'de verilmiştir. 360 derece tarama yaparak 6 metrelik menzilindeki tüm nesnelerin mesafe bilgisini ölçebilen bir lazer sensördür.

Şekil 5.2. RP-LIDAR mesafe ölçüm sensörü

Tablo 5.1. RP-LIDAR sensörünün genel özelikleri

Özelikler	**Açıklama**
Kapsama açısı	360 derece
Hassasiyet	0.5 derece
Tarama hızı	2-10 Hz
Saniyede verilen örnek	2000
Menzil	0.2-6 m

Uygulamada kullanılan RP-LIDAR sensörünün çalışma prensibi Şekil 5.3'te gösterilmiştir. Birden başlayarak 360 dereceye kadar birer derece artırarak çevresine yaklaşık 360 tane ışık sinyali yollanır. 0.2-6 metrelik menzil içinde bulunan tüm nesnelerden yansıyan ışık sinyalleri sensördeki alıcılar sayesinde algılanır. Bu sayede 360 derecelik açı boyunca etrafındaki nesnelerin hem mesafesini hem de hangi açıda durduklarını belirleyerek nesnelerin sensöre göre konumları belirlenmiş olur.

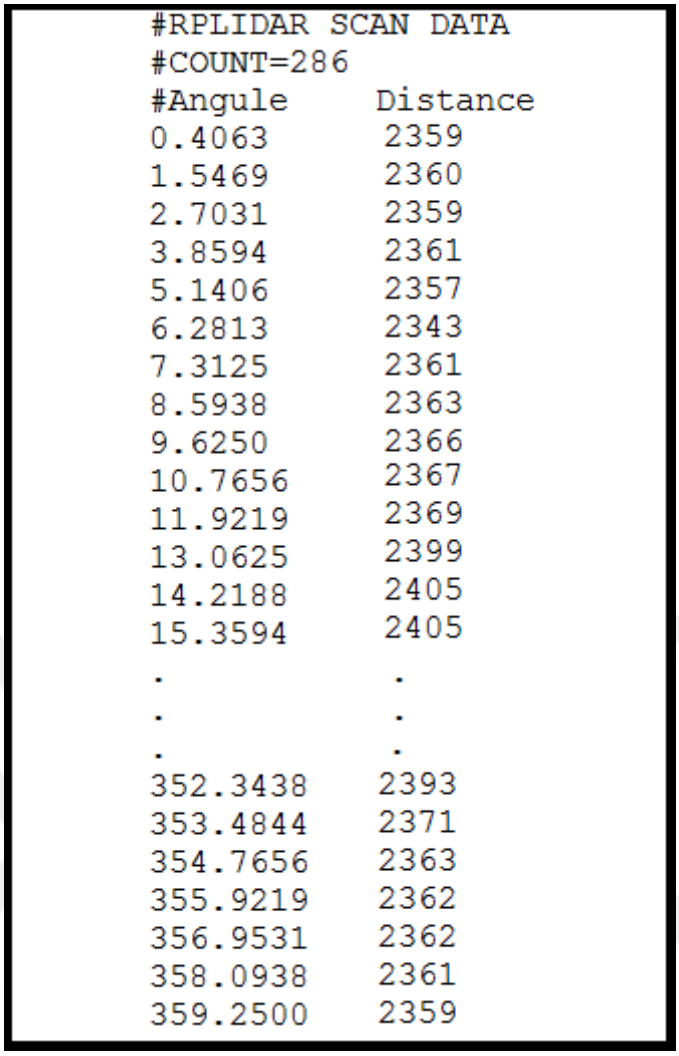


Şekil 5.3. RP-LIDAR çalışma prensibi

5.1.3. RP-LIDAR Verisi

RP-LIDAR'dan veri alabilmek için öncelikle Windows tabanlı bilgisayara RP-LIDAR'ın SDK dosyaları kurulup gerekli çalışmalar yapılmıştır. Daha sonra Şekil 5.4'teki gibi veriler elde edilmiştir. RP-LIDAR 360 derecelik açı boyunca tarama yapmaktadır. Her bir derecede bir ışın demeti gönderilmektedir.

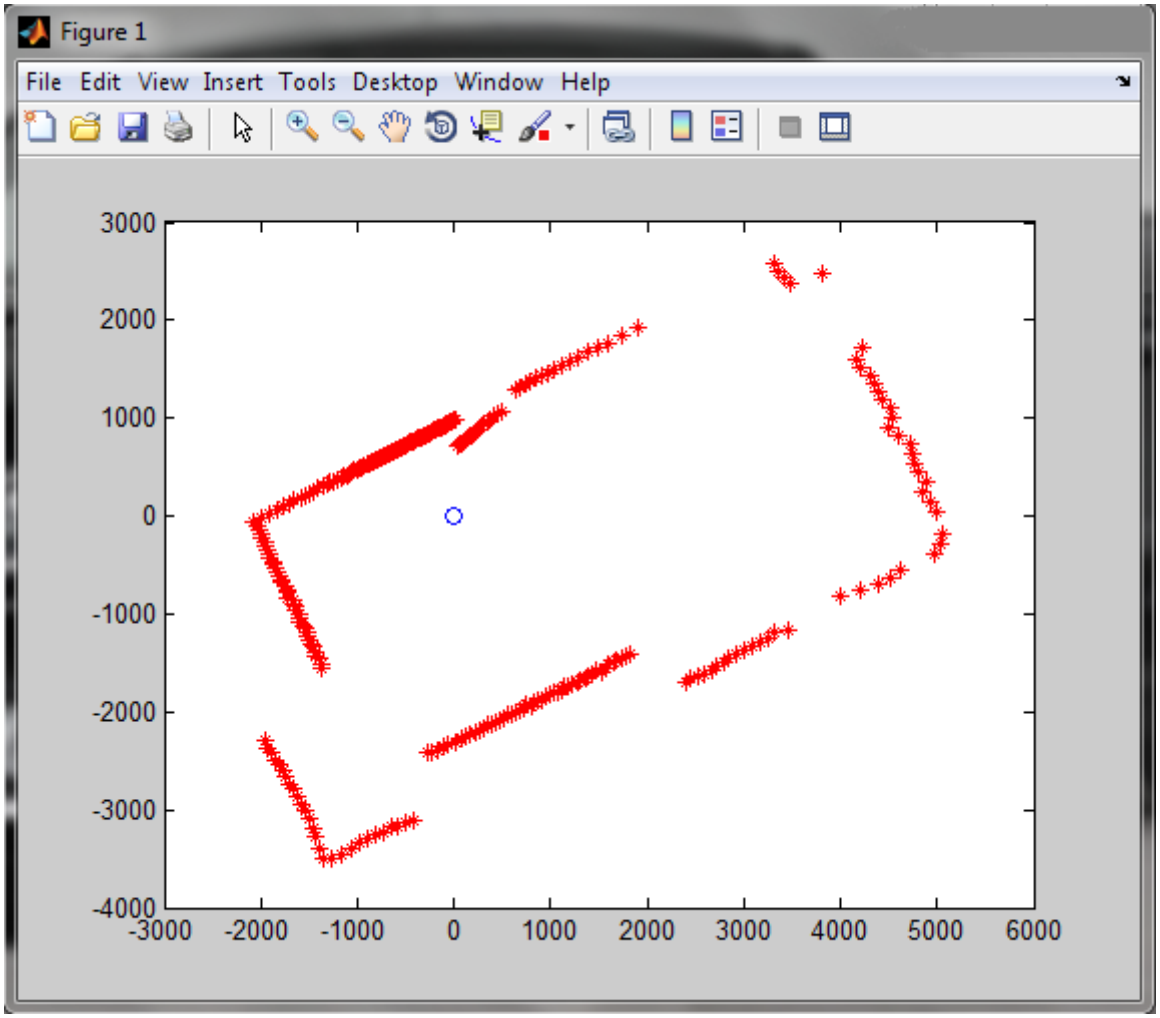
Ancak, kullanılan RP-LIDAR 0.5 derecelik hassasiyete sahip olduğu için 360 tane nokta bilgisi alınamamaktadır. Onun dışında 0.2-6 metre arasındaki menzili dışındaki noktaları da veri setine yüklemediği için Şekil 5.4'te görüldüğü gibi 286 tane nokta hesaplanmıştır.



```
#RPLIDAR SCAN DATA
#COUNT=286
#Angule     Distance
0.4063      2359
1.5469      2360
2.7031      2359
3.8594      2361
5.1406      2357
6.2813      2343
7.3125      2361
8.5938      2363
9.6250      2366
10.7656     2367
11.9219     2369
13.0625     2399
14.2188     2405
15.3594     2405
.           .
.           .
.           .
352.3438    2393
353.4844    2371
354.7656    2363
355.9219    2362
356.9531    2362
358.0938    2361
359.2500    2359
```

Şekil 5.4. RP-LIDAR verisi

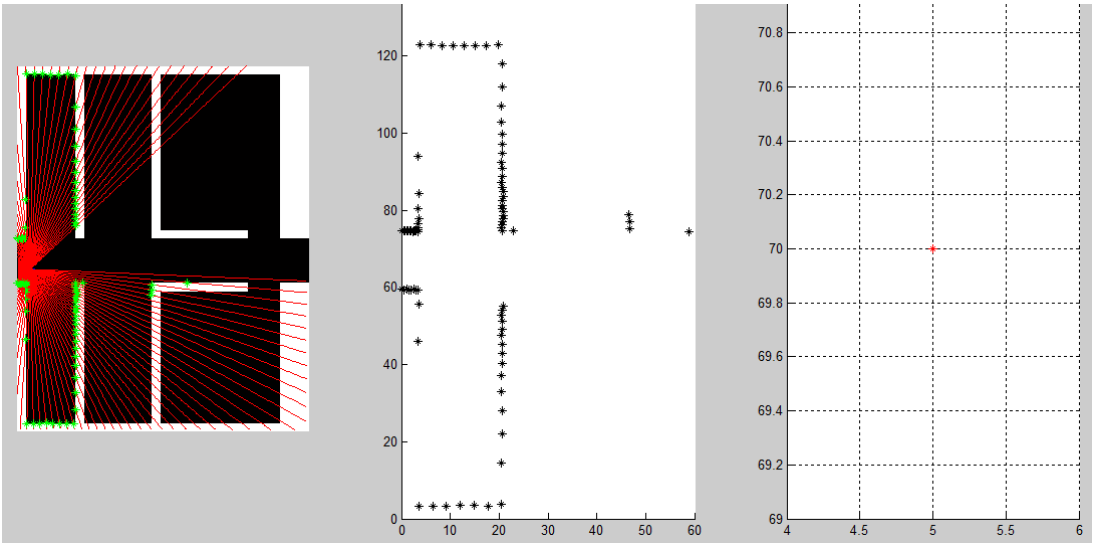
Veriler elde edildikten sonra bunları anlaşılır bir şekilde çizdirmek için MATLAB programında Şekil 5.5 te görüldüğü gibi çizdirme işlemi yapılmıştır. Bunu yapabilmek için öncelikle RP-LIDIR'ın SDK dosyalarından 360 derecelik tek tarama sonucu elde edilen veri kümesini MATLAB'ta hazırlanan programa yüklenir. Veri kümesinde belirlenen her nesne için menzil ve açı değeri okunarak çizdirme işlemi başarılı bir şekilde gerçekleştirilir.



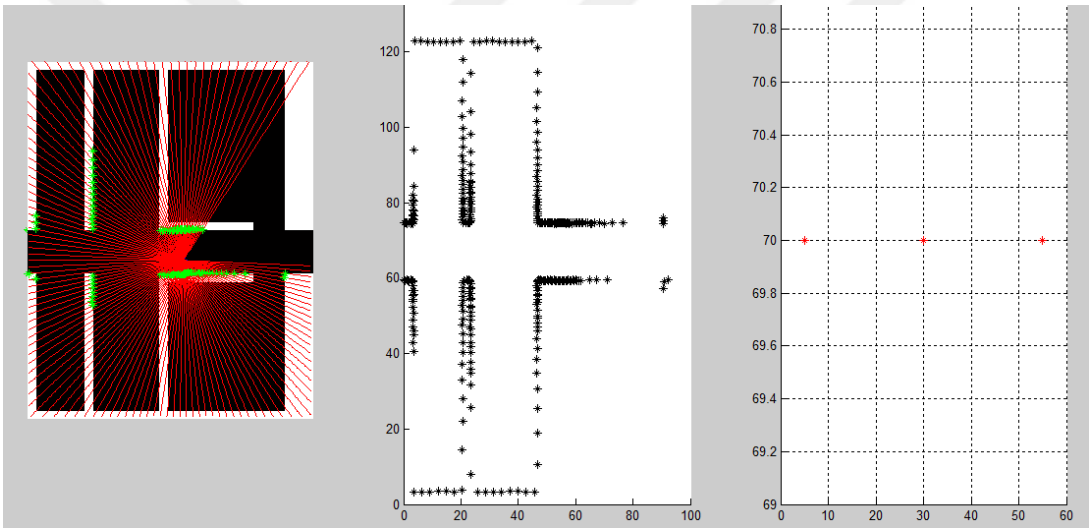
Şekil 5.5. MATLAB'ta okunan verilerin görselleştirilmesi

5.2. MATLAB Kullanarak SLAM Benzetimi Uygulaması

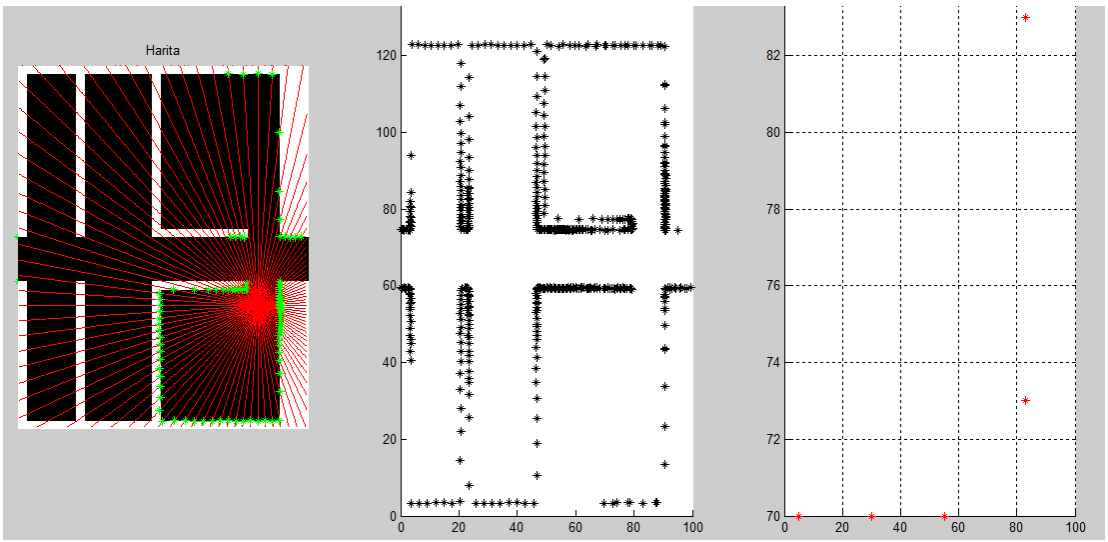
SLAM uygulaması öncelikle MATLAB ile gerçekleştirilmiştir. Uygulamada ilk olarak sanal bir ortam oluşturulmuştur. İkinci olarak RP-LIDAR'a benzer şekilde tarama yapılarak çevre algılanmıştır. Son olarak ortam haritası çıkarılmış ve aynı anda bulunan noktanın konumu belirlenmiştir. Şekil 5.6-5.8'de sanal olarak oluşturulan ortamda RP-LIDAR çalışma ilkesine benzer olarak tarama sonuçlarının sırasıyla 1, 3 ve 5 adımları gösterilmiştir. Bu şekillerin ilk grafiğinde, tarama sonucunda belirlenen noktalar sayesinde çıkarılan ortamın haritası verilmiştir. İkinci grafiğinde belirlenen noktaların x ve y koordinatları göstermiştir. Üçüncü grafiğinde ise, mobil robotun bulunduğu konum verilmiştir. Her adımda bir tane robot konumu belirlendiği için, Şekil 5.6-5.8 arasında robot konumu sırasıyla 1, 3 ve 5 kırmızı nokta ile gösterilmiştir.



Şekil 5.6. MATLAB ile SLAM uygulamasının 1. adımı



Şekil 5.7. MATLAB ile SLAM uygulamasının 3. adımı



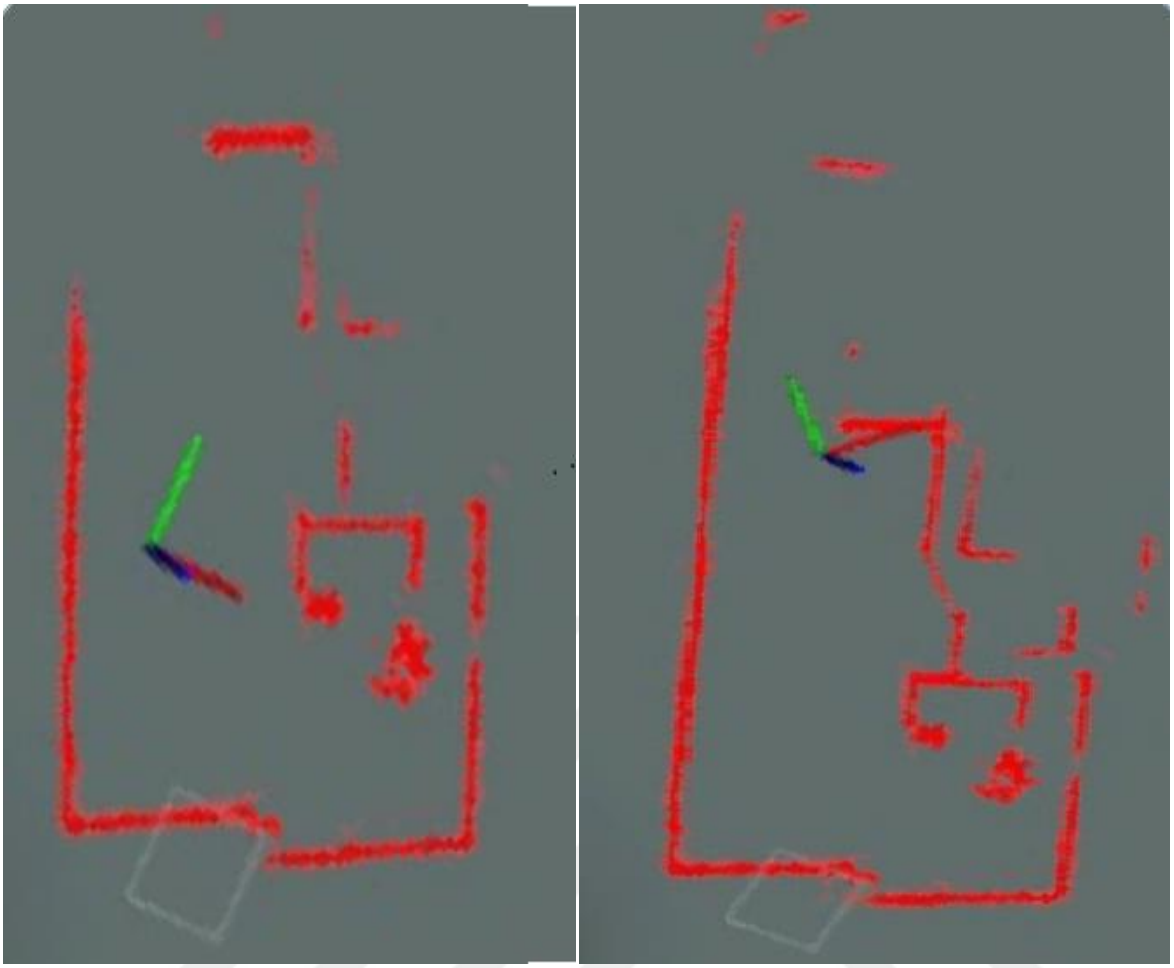
Şekil 5.8. MATLAB ile SLAM uygulamasının 5. adımı

Şekillerde görüldüğü gibi sonuçlar oldukça başarılıdır. Ancak sanal olduğu için ve eş zamanlı çalışmadığından dolayı, SLAM'in tanımı olan eş zamanlı konum belirleme ve haritalama gerçekleştirilmemiştir. Bu nedenle, başka bir yöntem olarak Linux ortamında ROS kullanılmıştır.

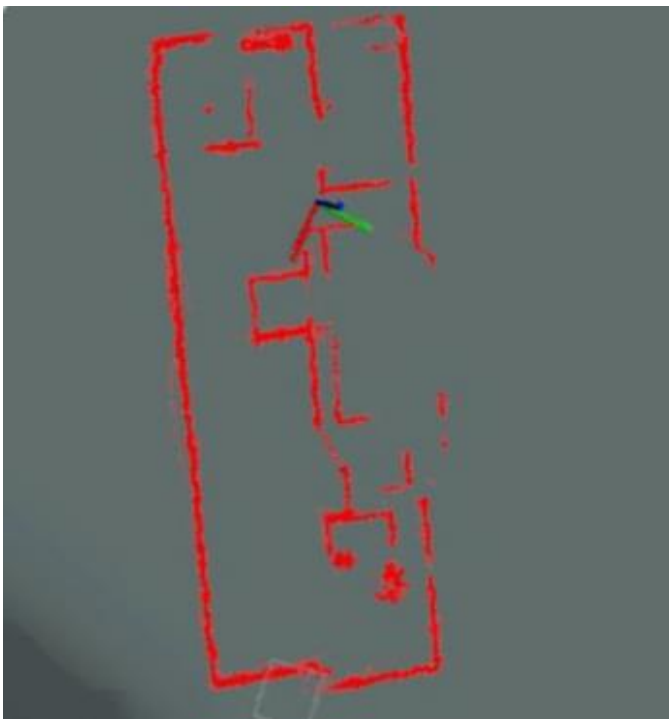
5.3. ROS Kullanarak Gerçek Zamanlı Otonom SLAM Uygulaması

İlk olarak, Linux tabanlı bilgisayara sanal bir işletim sistemi olan ROS'un son versiyonu Kinetic Kame yüklenmiştir. İkinci olarak RP-LIDAR'dan gerçek zamanda verilerini almak için gerekli RP-LIDAR paketler yüklenmiştir. Üçüncü olarak Parçacık Filtre tabanlı Gmapping paketi yüklenmiştir. Son olarak de Turtlebot'a yüklenen programla otonom olarak hareket etmesi sağlanmıştır. Şekil 5.9'da gösterilen ortamda otonom olarak hareket ederek SLAM uygulaması gerçekleştirilmiştir. Şekil 5.10'de otonom olarak hareket eden Turtlebot'un çizdiği harita verilmiştir.

Şekil 5.9. Uygulama ortamı



a.) b.)



c.)

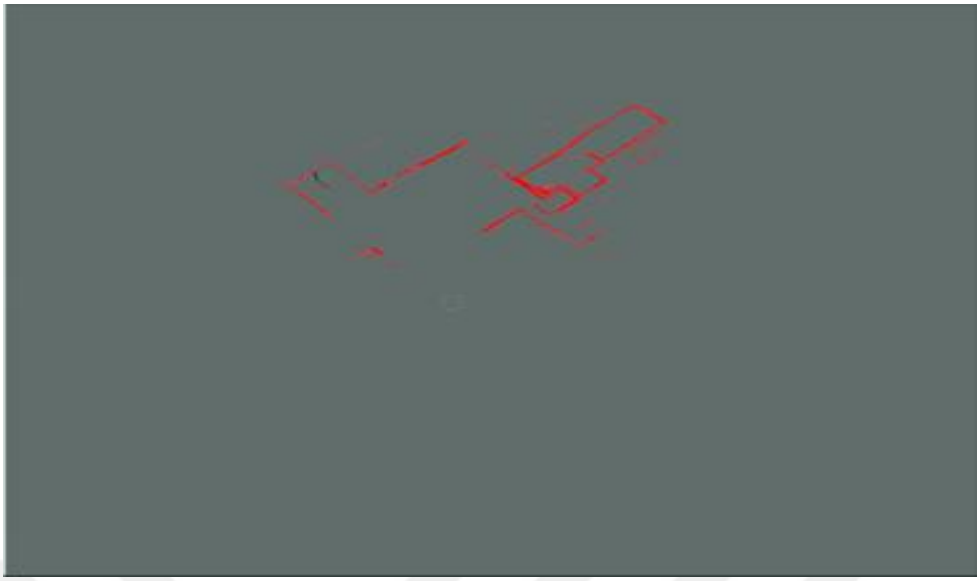
Şekil 5.10. Otonom olarak hareket eden Turtlebot'un SLAM uygulaması a)1. adım, b) 2. adım ve c) 3. adım

5.4. ROS Kullanarak Gerçek Zamanlı Manuel SLAM Uygulaması

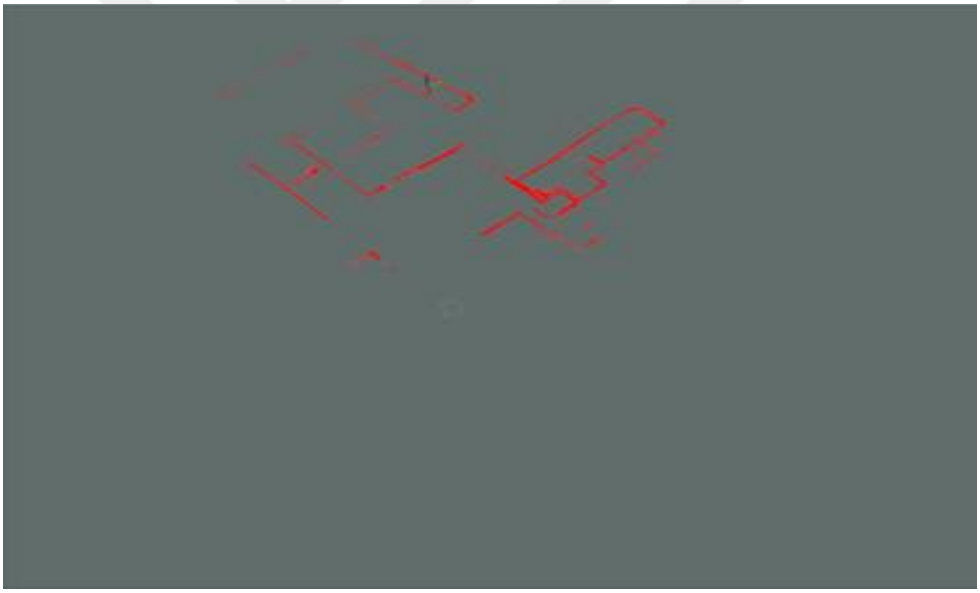
Son olarak yaptığımız uygulama Turtlebot mobil robotu manuel olarak hareket ettirerek SLAM uygulamasını gerçekleştirilmesi olmuştur. Bunun içinde ikinci bir bilgisayara yüklediğimiz ROS işletim sistemini kullanarak USB ara birimi ile Turtlebot'un hareketi manuel olarak kontrol edilmiştir. İkinci bilgisayarın klavyesi kullanılarak mobil robotun sağa-sola, ileri-geri ve bulunduğu noktada sabit kalarak kendi etrafında dönmesi hareketleri gerçekleştirilmiştir. Turtlebot'un üzerindeki bilgisayarda ROS'un Gmapping paketi kurularak SLAM uygulaması gerçekleştirilmiştir. Örnek uygulama adımları Şekil 5.11-17 ile verilmiştir.

Şekil 5.11. Fırat Üniversitesi Mekatronik Mühendisliği kat haritası uygulamasının 1. adımı

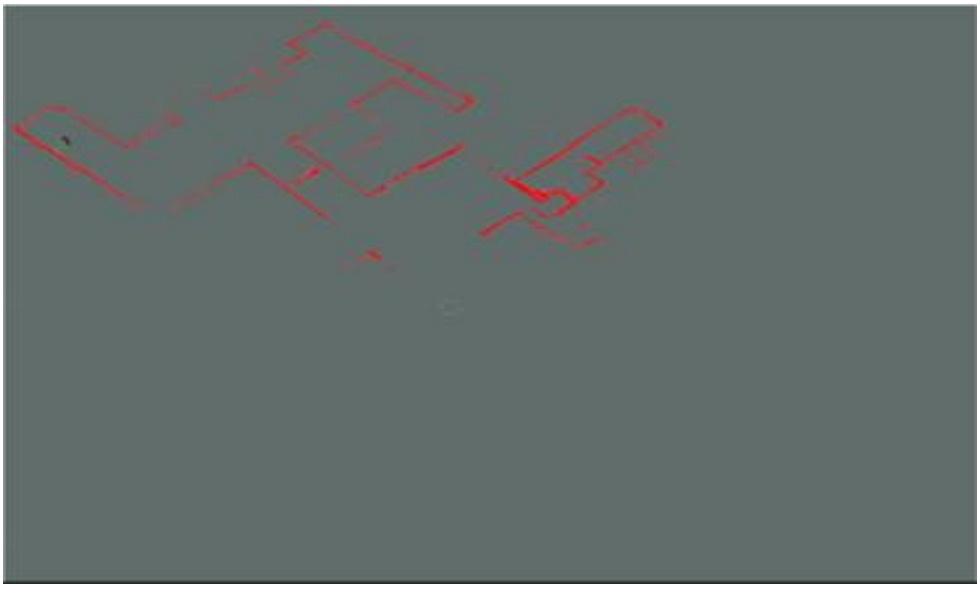
Şekil 5.12. Fırat Üniversitesi Mekatronik Mühendisliği kat haritası uygulamasının 2. adımı



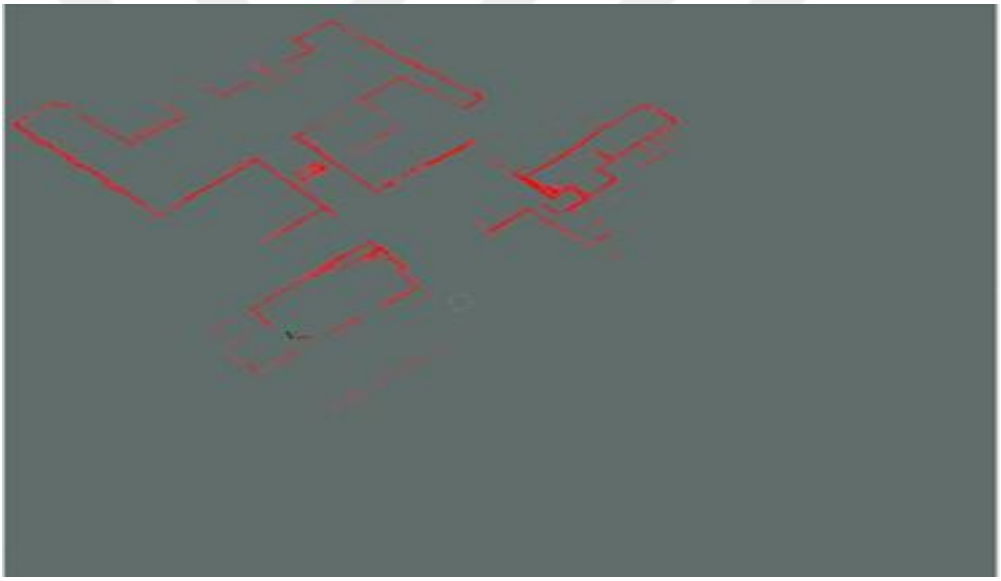
Şekil 5.13. Fırat Üniversitesi Mekatronik Mühendisliği kat haritası uygulamasının 3. adımı



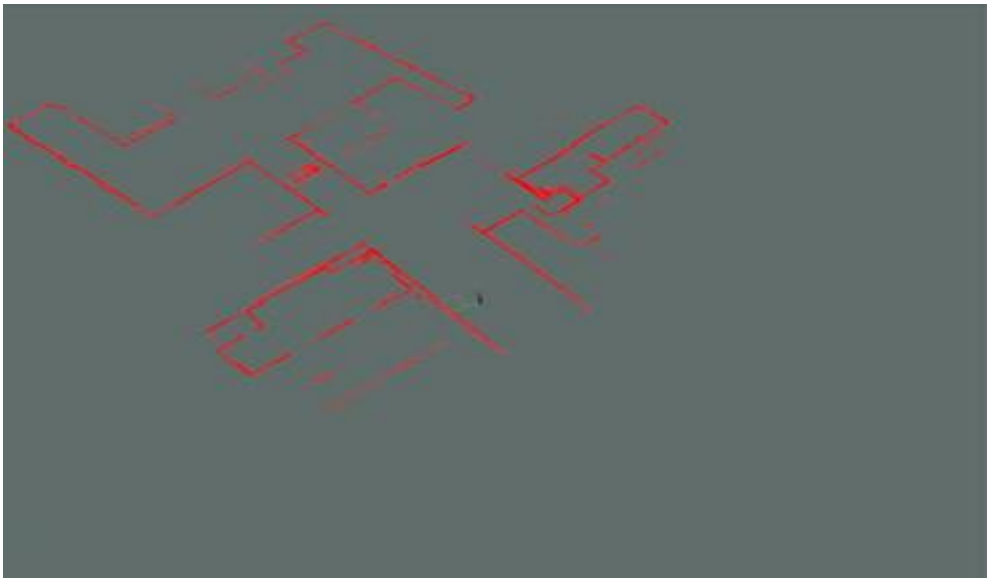
Şekil 5.14. Fırat Üniversitesi Mekatronik Mühendisliği kat haritası uygulamasının 4. adımı



Şekil 5.15. Fırat Üniversitesi Mekatronik Mühendisliği kat haritası uygulamasının 5. adımı



Şekil 5.16. Fırat Üniversitesi Mekatronik Mühendisliği kat haritası uygulamasının 6. adımı



Şekil 5.17. Fırat Üniversitesi Mekatronik Mühendisliği kat haritası uygulamasının 7. adımı

6. SONUÇLAR

Otonom kavramı, son on yılda giderek artan bir eğilimle birçok uygulamada kullanılmış ve bu kavramın önemi her geçen gün artmıştır. Bu kavram sadece araçlar için değil, aynı zamanda diğer robotik uygulamalarda da büyük öneme sahiptir. Belirli hedefler için çevre tanıma ve kendi karar yeteneği, robotik uygulamaları otonom olarak kullanmak için gereklidir. Mikro elektro-mekanik sensör teknolojilerindeki ve gömülü elektronik cihazlardaki gelişmeler, otonom kavramını yalnızca askeri veya uzay teknolojisi için değil, günlük yaşam uygulamaları için de uygular. Araçların ve robotların, bu kendinden karar kabiliyeti, insan hayatını daha kolay ve daha güvenli hale getirecektir.

Bu çalışmada, otonom mobil robot kavramı için en yeni algoritma çerçevesi olan SLAM incelenmiş ve Turtlebot mobil robotu kullanılarak SLAM uygulamaları yapılmıştır. Otonom olarak sensörler yardımı ile çevresini algılayarak hareket edebilen ve bulunduğu ortamın haritasını çıkarıp konumunu belirleyen mobil robot uygulamaları gerçekleştirilmiştir. İlk uygulamada RP-LIDAR ile elde edilen verilerin Windows'da MATLAB ortamında görselleştirilmesi gerçekleştirilmiştir. İkinci uygulamada, MATLAB ortamında sanal olarak hazırlanmış bir ortamın haritalaması ve robotun bulunduğu yerin konumunun belirlenmesi için SLAM benzetimi gerçekleştirilmiştir. Üçüncü uygulamada mobil robotumuzda bulunan çarpma sensörleri yardımı ile tekerleklere bağlı fırçalı DC motorlar kontrol edilerek mobil robotumuzun otonom olarak hareket etmesi sağlanmıştır. Otonom mobil robotun üzerine yerleştirilen RP-LIDAR'dan anlık olarak veri alınarak ROS ortamında Kalman Filtresinin genişletilmiş versiyonu olan parçacık filtreleme yöntemini içeren Gmapping SLAM algoritmasıyla haritalama ve konum belirleme gerçekleştirilmiştir. Son uygulamada USB ara birimi ile manuel olarak mobil robotun kontrolü yapılarak aynı anda konum belirleme ve haritalama işlemi gerçekleştirilmiştir.

Gerçekleştirilen tüm uygulamalar, mobil robotun etrafını başarıyla tanıdığını ve robotun içinde yer aldığı haritaları oluşturduğunu göstermiştir. Ancak SLAM çerçevesinin uygulanmasının kolay olmadığı açıkça görülmüştür. Farklı sensörlerin birlikte kullanımı, ileri yazılım yöntemleri, kontrol algoritmaları ve filtreler gibi konular bu alandaki çalışmalar için gereklidir.

Gelecek çalışma olarak kinect sensörü kullanılarak SLAM gerçekleştirilebilir. Maliyeti düşürerek ve çoklu robotlarla SLAM bilgisini daha hızlı bir şekilde gerçekleştirilebilecek çalışmalar yapılabilir.

KAYNAKLAR

[1] Smith, R. and Cheesman, P., (1987). On the representation of spatial uncertainty, International Journal Robotic Research, 5(4), 56–68.

[2] Leonard, J.J., Durrant-whyte, **H.F.,** (1991). Simultaneous map building and localization for an autonomous mobile robot, IEEE/RSJ International Workshop Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, 1442–1447.

[3] Dissanayake, G., Durrant-Whyte, H. and Bailey, T., (2000). A Computationally efficient solution to the simultaneous localisation and map building (SLAM) problem, International Conference on Robotics and Automation, 109–114.

[4] Bailey, T. and Durrant-Whyte, H., (2006). Simultaneous localization and mapping (SLAM): Part I The essential algorithms, IEEE Robotics and Automation Magazine, 99–108.

[5] Bailey, T. and Durrant-Whyte, H., (2006). Simultaneous localisation and mapping (SLAM): Part II State of the art, IEEE Robotics and Automation Magazine, September, 108–117.

[6] Bellian, J.A., Kerans, C., Jennette, D.C., (2005). Digital out crop models: applications of terrestrial scanning lidar technology in stratigraphic modeling, Journal of Sedimentary Research, 75, 166–176.

[7] Surmann, H., Nüchter, A. and Hertzberg, J., (2003). An autonomous mobile robot with a 3D lazer renge finder for 3D exploration and digitalization of indoor environments, Robotics and Autonomous Systems, 45, 181–198.

[8] Nguyen, V., Martinelli, A., Tomatis, N. and Siegwart, R., (2005). A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics, IEEE/RSJ International Conference on Intelligent Robots and Systems, 1929–1934.

[9] Chong, T.J., Tang, X.J., Leng, C.H., Yogeswaran, M., Ng, O.E. and Chong, Y.Z., (2015). Sensor technologies and simultaneous localization and mapping (SLAM), Robotics and Intelligent Sensors (IRIS 2015),174–179.

[10] Kalman, R.E., (1960). A New approach to linear filtering and prediction problems, trans. ASME, Journal of Basic Engineering, 82, 35–45.

[11] Vidal-Calleja, T., Andrade-Cetto, J. and Sanfeliu, A., (2004). Conditional for suboptimal filter stability in SLAM, IEEE Inernational Conference on Intelligent Robots and Systems, Sendai Japan, 27–32.

[12] Çayıroğlu, İ., (2012). Kalman filtresi ve bir navigasyon uygulaması, Science and Technology Information Sharing, 2012–2.

[13] Oğuz, A.E. and Temeltaş, H., (2013). Genişletilmiş kalman filtresi (GKF) tabanlı uçak üzeri eş zamanlı konumlama ve haritalama(U-EZKH), Havacılık ve Uzay Teknolojileri Dergisi, 6(2), 69–74.

[14] Dempster, A.P., Laird A.N. and Rubin, D.B., (1977). Maximum likelihood from incomplete data via the EM algorith, Journal of the Royal Statistical Society, Series B, 39(1), 1–38.

[15] Thrun, S., Fox, D. and Burgard, W., (1998). A Probabilistic approach to concurrent mapping and localization for mobile robot, Machine Learning, 31(1-3), 29–53.

[16] Yuen, D.C.K. and MacDonald, B.A., (2002). A Comparison between EKF and sequential monte carlo techniques for Simultaneous localisation and map-building, Australian Conference on Robotics and Automation, November 2002, Auckland, New Zealand, 26–28.

[17] Siegwart, R. and Nourbakhsh, I.R., (2004). Introduction to autonomous mobile robots, The MIT Pres, Cambridge, Massachusetts, 102(1), 75–78.

[18] Kurt, Z., (2007). Eş zamanlı konum belirleme ve haritalamaya yönelik akıllı algoritmaların geliştirilmesi, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[19] Kavak, D., (2007). İnsansız kara araçları navigasyonunda genişletilmiş kalman (GKF) ve sıkıştırılmış kalman filtre (CGKF) tabanlı SLAM yöntemlerinin geliştirilmesi ve karşılaştırılması, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[20] Paslıoğlu, K., (2010). Otonom mobil robotlarda dağılımlı kalman filtresi tabanlı eş zamanlı lokalizasyon ve haritalama, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Ensititüsü, İstanbul.

[21] Wang, P., Chen, Z., Zhang, Q. and Sun, J., (2016). A loop closure improvement method of gmapping for low cost and resolution lazer scanner, International Federation of Automatic Control, 49-12, 168–173.

[22] Turtlebot, http://www.turtlebot.com/

[23] Jagtap, V., (2015). Cyber physical system for continuous evaluation of fall risks to enable aging-in-place, Yüksek Lisans Tezi, Worcester Polytechnıc Institute.

[24] ROS, http://www.ros.org/

[25] Ruckert, E. A., (2009). Simultaneous localisation and mapping for mobile robots with recent sensor technologies, Yüksek Lisans Tezi, Graz University of Technology, Graz.

[26] Smith, R.C., Self, M. and Cheeseman, P., (1990). Estimating uncertain spatial relationships in robotics, In I.J. Cox and G.T. Wilfong, editors, Autonomous Robot Vehicles, Springer-Verlag, 167–193.

[27] Smith, R.C. and Cheeseman, P., (1985). On the representation and estimation of spatial uncertainty, Technical Report TR 4760 & 7239, SRI.

[28] Thrun, S., (2002). Robotic mapping: A survey, Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann, 1–35.

[29] Montemerlo, M., Thrun, S., Koller, D. and Wegbreit, B., (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem, Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence, (UAI'02), 1-4 August 2002, Alberta, Canada, 593–598.

[30] Montemerlo, M., Thrun, S., Koller, D. and Wegbreit, B., (2003). FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges, the 18th International Joint Conference on Artificial Intelligence (IJCAI'03), 2003, Acapulco, Mexico, 1151–1156.

[31] Paz, L. M., Tardos, J. D. and Neira, J., (2008). Divide and conquer: EKF SLAM in O(n), IEEE Transactions on Robotics, 24(5), 1107–1120.

[32] Guivant, J. E. and Nebot, E. M., (2001). Optimization of the simultaneous localization and map-building algorithm for real-time implementation, IEEE Transactions on Robotics and Automation, 17 (3), 242–257.

[33] Guivant, J. and Nebot, E., (2002). Improving computational and memory requirements of simultaneous localization and map building algorithms, the IEEE International Conference on Robotics and Automation (ICRA'02), 11-15 May 2002, Washington, USA, 3, 2731–2736.

[34] Julier, S., (2001). A Sparse weight kalman filter approach to simultaneous localisation and map building, the IEEE/RJS International Conference on Intelligent Robots and Systems, October 29-November 2 2001, Hawaii, USA, 1, 1251–1256.

[35] Tardos, J. D., Neira, J., Newman, P. M. and Leonard, J. J., (2002). Robust mapping and localization in indoor environments using sonar data, International Journal of Robotics Research, 21(4), 311–330.

[36] Thrun, S., (2002). Particle filters in robotics, the 18th Conference in Uncertainty in Artificial Intelligence, (UAI'02), 1–4 August 2002, Alberta, Canada, 511–518.

[37] Rekleitis, I. M., (2004). A particle filter tutorial for mobile robot localization, Technical Report TR-CIM-04-02, McGill University, Canada.

[38] Tuna, G., (2012). Çoklu algılayıcı füzyonunun çoklu robot sistemlerinde eş zamanlı konum belirleme ve haritalama problemine uygulanması, Doktora Tezi, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

[39] Murphy, K., (1999). Bayesian map learning in dynamic environments, 1015–1021.

[40] Grisetti, G. Stachniss, C. and Burgard, W., (2007). Improved techniques for grid mapping with rao-blackwellized particle filters, IEEE Transactions on robotics, 23, 34–46.

[41] LAZER, https://tr.wikipedia.org/wiki/Lazer

[42] RP LIDAR, http://www.slamtec.com/en/Lidar/A1

ÖZGEÇMİŞ

Selman AKYOL, 1990 yılında Mardin'in Dargeçit ilçesinde doğdu. İlk ve orta öğrenimini Nusaybin'de tamamladı. 2009 yılında Nusaybin Anadolu Lisesi'nden mezun oldu. 2010 yılında kazandığı Fırat Üniversitesi Mekatronik Mühendisliği Bölümü'nden 2014 yılında mezun oldu. Aynı yıl içinde Fırat Üniversitesi'nin Mekatronik Mühendisliği Ana Bilim Dalı'nda yüksek lisans eğitimine başladı.