İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

EŞ ZAMANLI KONUMLANDIRMA VE HARİTALAMA TEKNİKLERİNİN HIZ PERFORMANSININ GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Ziya Uygar YENGİN

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

HAZİRAN 2017

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

EŞ ZAMANLI KONUMLANDIRMA VE HARİTALAMA TEKNİKLERİNİN HIZ PERFORMANSININ GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Ziya Uygar YENGİN
(518141013)

Mekatronik Mühendisliği Anabilim Dalı

Mekatronik Mühendisliği Programı

Tez Danışmanı: Yrd. Doç. Dr. Volkan SEZER

HAZİRAN 2017

İTÜ, Fen Bilimleri Enstitüsü'nün 518141013 numaralı Yüksek Lisans Öğrencisi Ziya Uygar YENGİN, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "EŞ ZAMANLI KONUMLANDIRMA VE HARİTALAMA TEKNİKLERİNİN HIZ PERFORMANSININ GELİŞTİRİLMESİ" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Yrd. Doç. Dr. Volkan SEZER** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Yrd. Doç. Dr. Tufan KUMBASAR** İstanbul Teknik Üniversitesi	.....
	Yrd. Doç. Dr. M. Selçuk Arslan Yıldız Teknik Üniversitesi	.....

Teslim Tarihi : 5 Mayıs 2017
Savunma Tarihi : 6 Haziran 2017

Aileme,

ÖNSÖZ

Tez çalışmamın gerçekleşmesi sürecinde her zaman değerli fikir ve deneyimleriyle bana yol gösteren danışmanım Yrd. Doç. Dr. Volkan SEZER'e, yaptığım simülasyonlarda ve deneysel çalışmalarda büyük yardımları dokunan Aykut ÖZDEMİR, Mustafa DEMİR ve Emrah ABTİOĞLU'na teşekkürü bir borç bilirim.

Ayrıca bugüne kadar bana maddi ve manevi desteklerini hiçbir zaman esirgemeyen aileme şükranlarımı sunarım.

Mayıs 2017

Ziya Uygar YENGİN
Araştırma Görevlisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

SLAM **:** Simultaneous Localization And Mapping
EKF **:** Extended Kalman Filter
ROS **:** Robot Operating System
RANSAC **:** Random Sample Consensus
IMU **:** Inertial Measurement Unit
EZKH **:** Eş Zamanlı Konumlandırma ve Haritalama
GKF **:** Genişletilmiş Kalman Filtresi

SEMBOLLER

N	**:** İşaretçi Sayısı
$\sum$	**:** İşaretçi kovaryansı
$\boldsymbol{\mu}$	**:** Ortalama
$\boldsymbol{N}$	**:** İşaretçi sayısı
$\boldsymbol{z^t}$	**:** t anındaki ölçüm
$\boldsymbol{w_t^{[m]}}$	**:** t anındaki m'inci parçacığın önem ağırlığı
$\boldsymbol{u^t}$	**:** Kontrol girişi
$\boldsymbol{n^t}$	**:** Veri ilişkilendirme

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

EŞ ZAMANLI KONUMLANDIRMA VE HARİTALAMA TEKNİKLERİNİN HIZ PERFORMANSNIN GELİŞTİRİLMESİ

ÖZET

Günümüzde endüstriyel çalışmalardan uzay araştırmalarına, eğitimden tıbbi uygulamalara kadar çok çeşitli alanlarda kullanılan robotlar görevlerini yerine getirirken bunu nasıl yapacaklarına kısmen veya tamamen kendileri karar verirken, bu görevleri insanlar tarafından önceden tanımlanmış olarak sabit bir şekilde de yerine getirebilirler. Görevlerini nasıl yapacaklarına kendileri karar veren robotlara otonom robotlar denir. Otonom robotların büyük bir çoğunluğu yer değiştirme yeteneğine sahiptirler ve genellikle hareket edecekleri ortamla ilgili önceden bir bilgileri yoktur. Çalışmaya başladıkları andan itibaren üzerlerindeki sensörler yardımıyla çevrelerini tanımlamaya ve aynı zamanda bu çevre içerisinde nerede olduklarını anlamaya başlarlar. Eş zamanlı konumlandırma ve haritalandırma diye adlandırılan bu işlem otonom ve mobil robotlar için çok büyük önem taşımaktadır. Çünkü, düzgün bir harita oluşturabilmek için konum bilgisinin doğru olması gerekir. Aynı şekilde konum bilgisini doğru bir şekilde hesaplayabilmek için de düzgün bir haritaya ihtiyaç vardır. Birbirine bağlı olan bu iki durum, sensörlerin bozunumlu veri elde etmesi ve çevre koşullarından kaynaklı sorunlardan dolayı EZKH mobil robotlar için zor bir görevdir.

EZKH alanında çalışan araştırmacıların bir kısmı doğru harita çıkarımı ve konum belirlenmesi üzerine yoğunlaşmışken, bir kısmı da hız performansını geliştirmeye yönelik çalışmalar yapmışlardır. Bu tez çalışmasında, mevcut EZKH yöntemlerinin hız performansını geliştirmek için yeni bir yöntem önerilmiştir.

Robot, oluşturduğu haritayı yeni aldığı ölçümlerle güncellerken veri ilişkilendirme prosedürüne başvurur. Oluşturduğu bu haritadaki eleman sayısı çok büyük boyutlara ulaştığında veri ilişkilendirme, robot için büyük bir zaman kaybına neden olabilir. Özellikle parçacık filtresi tabanlı yaklaşımlarda bu işlemin parçacık sayısı ve haritadaki eleman sayısı kadar tekrar edildiği düşünüldüğünde aşırı miktarda hesaplama yükünün oluştuğu görülmektedir.

Önerilen yöntemde EZKH algoritmalarındaki veri ilişkilendirme adımının daha etkili bir şekilde kullanımının sağlanmasıyla bu aşamadaki çalışma zamanının düşürülmesi amaçlanmıştır. Yöntemin başarısını test etmek için simülasyon ortamında uygulamalar yapılmış ve ne kadar başarıya ulaştığının anlaşılması için farklı EZKH yaklaşımları da bu simülasyon ortamında uygulanıp sonuçlar karşılaştırılmıştır.

Simülasyonlardan elde edilen görsel ve sayısal sonuçlar incelendiğinde, yeni EZKH yönteminin harita ve konum hesaplamadaki yeteneğinden bir şey kaybetmediği; bunun yanında çalışma hızının mevcut yaklaşımlardan açık bir şekilde daha iyi olduğu görülmüştür. Ayrıca EZKH yöntemi uygulanmadan önce, sensörlerden alınan gürültülü verinin düzeltilmesinin harita oluştururken hesapsal olarak büyük kolaylık sağladığı anlaşılmıştır.

IMPROVING RUNTIME EFFICIENCY OF SIMULTANEOUS LOCALIZATION AND MAPPING TECHNIQUES

SUMMARY

When the word "robot" is enounced, it is commonly understood that it is a humanlike machine with its arms, legs, head and body. But, there are lots of different types of robot mechanisms used in industrial sectors, planetary explorations, education and military. For example; a robot in an automobile factory looks like a human arm which is attached to a fixed joint. These machines are designed to carry out the tasks like painting and welding. They do their job recursively in a predefined manner by humans. On the other hand, a machine discovering a planet may looks like a vehicle. Because of unpredictable environments and conditions, these robots should decide how they carry out their tasks on their own. These tasks may be going to somewhere with specific coordinates from current location or . It is hard even to go from one place to another without getting lost and hitting the objects.

In order to manage this change of location mission, a robot should first understand how the environment looks like and where it is in this environment. Without foreknowledge of the surroundings and coordinates of current position, estimating the robot pose and generating the map of environment is called as simultaneous localization and mapping (SLAM).

SLAM also known as concurrent mapping and localization (CML) is a very hard problem for robots due to the noisy measurements, unpredictable conditions, wheel or foot slippage, presence of moving objects around and estimation errors. Additionally, the amount of the data collected by measurement devices may cause a huge computational cost for processors of the robot. For example; a laser range finder with 360 degree field of view can collect a great number of data. This huge data causes the methods which constructs grid map to slow down. Likewise, a system using camera should process lots of data. In order to reduce the computational cost of SLAM algorithms, one can either increase the number of collected data or improving the process of SLAM techniques.

Most of the researhers are focused on solving the problems above using probabilistic methods. In literature, there are two common SLAM technique called as EKF-SLAM and FastSLAM. The EKF-SLAM is easily applicable approach while the FastSLAM is computationally efficient technique. These two methods are landmark based and use kalman filter. Landmarks are distinct points of some objects such as doors, walls or several geometric objects. Using landmarks decrease the operation time when compared to the SLAM methods which construct grid maps. EKF-SLAM has a significant drawback by comparison with FastSLAM. When the number of landmarks reaches to a high value, EKF-SLAM gets slow drastically. This is caused by a procedure in EKF-SLAM which is called as data association. Data association is a process of matching currently sensed landmark with the one in the estimated map. All the elements in the map are associated with each other. Therefore, when a new landmark is detected, the map grows exponentially and this structure gets the algorithm slower.

As a solution for the exponential growth in EKF-SLAM, FastSLAM offers a very innovative approach. It combines the particle filter and extended Kalman filter by separating pose estimation and landmark estimation. While EKF-SLAM is interested in calculating only current pose of the robot which is called as online SLAM, FastSLAM is interested in calculating the path of the robot which is called as full SLAM. With the knowledge of the whole poses of the robot, FastSLAM estimates the landmarks individually. This means that all landmarks are associated with only robot pose. When the robot detects a new landmark, mean and covariance of the landmark are added for all particles that causes a linear growth in the map. Considering the exponential growth in EKF-SLAM, FastSLAM brings an important innovation to map management. However FastSLAM is very efficient approach, it also gets slower if the number of estimated landmarks are gigantic. Because, in such a case, FastSLAM faces the numerous operation in data association step.

In order to decrease the computational cost of the data association, several approaches are proposed. The most effective method limits the number of landmarks to ones only in the range of the measurement device. This method provides a very good computational efficiency for SLAM techniques, especially for the particle filter based ones. However, if the scan range of the measurement device attached to the robot is extensive, the number of landmarks that are considered in data association process is high again. Also in some situations such as a landmark is out of the range of the sensor, this approach may fail.

In this thesis, a novel technique that eliminates unnecessary operations in data association process is proposed. Briefly; when a new landmark is detected by measurement device, it is compared with the landmarks only in a small circular area. Other landmarks in the map and outside the circular area are automatically skipped. Center of this circular area is estimated by using the sensor measurement. Therefore, no matter how big the number of landmarks, the computational cost in data association step remains almost same as the beginning.

In order to see the success of the proposed method, simulations are performed in different environments that were created by a simulation program called as Gazebo. Different SLAM algoritms are also tested in simulation environments in order to understand how much the new approach is faster than other aproaches. Additionally, these tests are done in real laboratory environments.

Before testing the new SLAM technique and others, some arrangements were made. The sensor which is used for the collection of 2D data from environment, produces noisy data. This noisy data makes the landmark extraction very difficult. For this reason, a smoothing filter was implemented on the noisy data. For the sake of simplicity, the mean filter was prefered. After smoothing the laser data, landmark extraction became easier.

Another arrangement was made for the landmark extraction and identification. The curvature function method which simplifies the identification of the landmarks was implemented on the sensor data.

After tuning all the parameters of the SLAM algorithms, the robot operated 30 times for two different environments and three different SLAM methods.

When the simulation results were analized, it was understood that proposed method estimates the robot pose and landmarks map almost at the same accuracy compared with other methods. Afterwards, the runtime values of the three methods were compared. It was clearly seen that the proposed method improves the runtime efficiency of actual SLAM methods. The results of the real world experiments also reveal the improvement of the new approach.

In recent years, the sensor technology has improved a lot. They can scan very wide range of field and collect a huge data. Therefore, it is contemplated to implement the new SLAM method on the systems with sensors of different quality so that how the approach is effective. Similarly, the new technique can be implemented to the different SLAM algorithms to see its suitability.

1. GİRİŞ

İlk defa Çekoslavak yazar Karel Čapek'in 1920 yılında yazdığı tiyatro oyununda kullanılan robot kelimesi insan benzeri bir yapısı olan ve insan davranışlarını taklit eden bir cihazı akla getirse de bugün; endüstriden uzay araştırmalarına, tıbbi uygulamalardan eğlence sektörüne kadar çok farklı alanlarda kullanılan farklı yapıdaki birçok cihazı tanımlamaktadır [1, 2]. Örneğin; otomotiv sektöründe yaygın olarak kullanılan robotlar kol şeklinde bir mekanizmaya sahiptirler ve sabit bir alanda çalışılar. Bu robotlar, boyama ve birleştirme gibi görevleri insanlar tarafından önceden tanımlanmış bir şekilde tekrar tekrar yerine getirmek üzere tasarlanırlar. Diğer yandan, uzay araştırmalarında kullanılan robotlar ise bilinmeyen ortamlarda ve öngörülemeyen durumlarda çalışacağı için, bir görevi yerine getirirken bunun nasıl yapılacağına kısmen veya tamamen kendisi karar verebilecek şekilde tasarlanırlar. Aynı şekilde mağara araştırmaları veya sualtı araştırmaları için üretilen ve kendi kendilerine karar verebilen bu tarz robotlara otonom robotlar denir. Teknik olarak açıklanacak olursa; üzerindeki sensörler aracılığıyla çevresinden veri toplayıp, bu veriyi kendi mikroişlemcisinde anlamlı hale getirerek görevini nasıl gerçekleştireceğine karar veren cihazlara otonom robotlar denir.

Otonom robotların çok büyük bir kısmı yer değiştirmeye ihtiyaç duyduğu için aynı zamanda mobil robotlar olarak da bilinirler. Mobil robotların üstesinden gelmesi gereken en önemli sorunlardan birisi daha önce hiç bilmediği bir ortamda çevresinde nelerin olduğunu ve hareket ettikçe bu ortam içinde nerede olduğunu anlamasıdır. Literatürde eş zamanlı konumlama ve haritalama(Simultaneous Localization and Mapping, SLAM) olarak bilinen bu konu, çok sayıda araştırmacının üzerinde uğraştığı bir konudur. EZKH'yi bu kadar zor hale getiren etmenlerin başında sensörler aracılığıyla toplanan verinin gürültülü olması, çevre koşullarının beklenmedik şekilde değişmesi, tahrik elemanlarına uygulanan komutların kusursuz olarak gerçekleşmemesi ve robotun hesaplama hataları gelmektedir. Örneğin; bir fabrika içinde çalışan bir robot hareket eden çok fazla insan ve yer değiştiren nesnelerle karşılacaktır. Bu durum harita oluştururken büyük bir karmaşaya sebep olabilir. Sualtı

araştırması yapan bir robot için de en büyük sorunlardan birisi suyun akıntısından kaynaklanan konum değişikliğidir. Tüm bu etmenlerin yanında robotun hareket ettiği alan büyüdükçe hesaplamalarda oluşan belirsizlikler de aynı şekilde artmaya başladığı için EZKH uygulamalarının çoğu olasılıksal yöntemler kullanılarak çözülmeye çalışılmaktadır. Bu yöntemlerin bir kısmı daha doğru sonuçlar elde edebilmek için sensörlerden alınan verinin tamamını kullanırken, bir kısmı ise hesaplama yükünü azaltmak ve hafıza gereksinimini düşürmek için verinin içinden sadece anlamlandırması kolay olan kısımları kullanmaktadır [2].

EZKH uygulamalarının performansını etkileyen en önemli unsurlardan biri robot üzerindeki sensörlerin çeşidi ve kalitesidir. Örneğin; ucuz olmaları ve sağladığı verinin kolay işlenebilir olması bakımından tercih edilen sonar sensörlerin, veri yoğunluğunun az ve gürültülü olması düzgün bir harita oluşturmada ve doğru konum belirlemede zorluk çıkarmaktadır. Diğer yandan lazer sensörler, yüksek doğrulukta ve çok fazla veri elde ettiği için EZKH performansı açısından büyük avantaj sağlamaktadırlar. Bir başka sensör çeşidi olan kameralar ise son zamanlarda giderek daha çok ilgi görmeye başlamıştır. Görüntüden elde edilen veri, 2 boyutlu şekil bilgisine ek olarak renk bilgisi de içerdiği için çevrenin anlamlandırılmasında diğer sensör çeşitlerine göre çok daha iyidir. Ancak çok fazla veri toplaması ve belirsizliğin yüksek olması, hız performansı ve doğru konumlandırma için bir dezavantaj olmaktadır [2].

Dışarıdan bilgi toplayan sensörlerin haricinde robotun kendi hareketinden kaynaklanan değişimleri algılayan sensörler de vardır. Tekerleklere bağlı enkoderler, jiroskop ve ivmeölçer gibi sensörleri içinde barındıran Atalet Ölçüm Ünitesi (Inertial Measurement Unit, IMU) bunlara birer örnektir. Enkoderlerden alınan veri IMU'dan alınan veriye göre çok daha düşük hataya sahiptir. Ancak, enkoderler tek boyuttaki değişimi algılayabilirken, IMU üç boyuttaki yer değişimi ve dönme, yuvarlanma ve yunuslama hareketlerini de algılayabilmektedir. Bu da üç boyutlu ortamlardaki EZKH uygulamaları için büyük kolaylık sağlamaktadır [2].

EZKH performansının artırılmasında sensör seçiminin önemi yüksek olsa da, uygulanacak yöntem de bir o kadar önemlidir. Örneğin; konum belirlemenin daha ön planda olduğu durumlar için işaretçi (landmark) tabanlı yöntemler hızlı olması açısından daha çok tercih edilirken, yol planlamanın önemsendiği durumlarda ise daha ayrıntılı harita oluşturan grid tabanlı yöntemler tercih edilir. Başka bir örnek olarak;

uzay araştırmalarında kullanılan robotlarda çevrenin neye benzediği ve hangi cisimlerin olduğu önemsendiği için görüntü tabanlı algoritmalar tercih edilir [3].

Yukarıda bahsedilen sebepler gözönüne alındığında eş zamanlı konumlandırma ve haritalama probleminin optimal çözümü için uygulanacak yöntem belirlenirken robotun hangi ortamda, hangi amaçla kullanılacağı ve üzerindeki sensörler çok etkilidir. Bu tez çalışmasında; geniş, kapalı alanlarda kolayca uygulanabilecek olan FastSLAM yöntemi kullanılmıştır.

1.1 SLAM Tarihçesi

Eş zamanlı konumlandırma ve haritalama üzerine yapılan ilk ciddi çalışmalar 1980'li yıllardan itibaren başlamıştır. [4] ve [5]'te ortaya konan çalışmanın, EZKH yöntemlerinin temellerini oluşturduğu söylenebilir. Çünkü; konumlandırma ve sensör ölçümü hatalarını olasılıksal olarak ifade etmeleri bakımından çalışmaları, günümüzdeki EZKH çalışmalarıyla benzerlik göstermektedir. EZKH problemi ilk defa [6]'da eş zamanlı haritalama ve konumlandırma olarak adlandırılmıştır. Önerilen yöntemde, günümüzde bu alanda çok yaygın olarak kullanılan Genişletilmiş Kalman Filtresi kullanıldı ve 2000'li yılların başlarında artık EZKH olarak adlandırılan çalışma daha da ileriye götürülerek günümüzde EKF-SLAM diye bilinen yöntem geliştirildi [7]. Bu geliştirilmiş yöntemde robot konumu ve işaretçi konumlarının belirsizlikleri olasılıksal olarak hesaplanmaktadır. Basitçe açıklanacak olursa; tahmin ve düzeltme olarak iki temel adımdan oluşan yöntemde, robotun konum bilgisinin ve bu konuma ait belirsizliği belirten bir kovaryans matrisinin olduğu bir matris işlem görmektedir. Bu matrisin içinde ayrıca her bir işaretçinin konum bilgileri ve bu işaretçilerin birbiriyle ilişkili olan kovaryansları vardır ve ölçüm alındıkça güncellenmektedir. Robot yeni bir işaretçi tespit ettiğinde bu işaretçiyi her bir işaretçiyle ilişkilendireceği için işlem yaptığı matriste üstel bir şekilde bilgi artışına sebep olur ki bu, algoritmanın $O(M^2)$ kadar eleman üzerinde işlem yapmasını gerektirir. Burada M işaretçi sayısını belirtir. M değeri yükseldikçe algoritmanın işlem hızı da büyük oranda azalmaktadır. Bu sorunu aşmak için bazı araştırmacılar farklı çözüm yolları önermişlerse de, yaptıkları çalışmalar en fazla birkaç yüz işaretçi söz konusu olduğunda işe yaramaktadır [8, 9, 10]. EKF-SLAM'deki üstel bir şekilde artan işlem karmaşasını çok daha küçük ölçekli hale getirmek ve bellek gereksinimini azaltmak için [11]'de FastSLAM olarak adlandırılan parçacık filtresi temelli bir yaklaşım önerilmiştir.

Genişletilmiş kalman filtresi ve parçacık filtresinin bir kombinasyonu olarak geliştirilen yöntem aslında Rao-Blackwellized parçacık filtresinin bir çeşididir [12]. FastSLAM'de robotun durumu parçacıklar üzerinden ifade edildiği için her bir parçacığın kendine ait bir konum ve yönelim bilgisi bulunmaktadır. Aynı şekilde, her bir parçacık için işaretçilerin konumları ve belirsizlikleri de ayrı ayrıdır. EKF-SLAM'den farklı olarak her bir işaretçi birbirinden bağımsız olduğu için kovaryansları da birbiriyle ilişkilendirilmemektedir. Yani, bir parçacık için yeni bir işaretçi eklendiği zaman bir konum ve kovaryans elemanı eklenmiş olur. Bunun N sayıdaki parçacık için yapıldığı düşünülürse FastSLAM'de $O(MN)$ kadar bir işlem karmaşası oluşmaktadır. EKF-SLAM ile karşılaştırıldığı zaman bunun çok daha düşük hesap yükü anlamına geldiği görülmektedir. Bununla birlikte işaretçi sayısının devasa boyutlara ulaşması FastSLAM algoritmasının da doğru orantılı olarak yavaşlamasına neden olmaktadır [11, 19].

FastSLAM'in hesapsal yükünün azaltılması için birçok çalışma yapılmıştır. Bunların en etkili olanları parçacık sayısının azaltılması, daha az bilgi içeren haritaların oluşturulması veya haritanın daha verimli bir şekilde kullanılmasına yönelik çalışmalardır. Bu konuda yapılan çalışmalardan biri olan Unscented FastSLAM yöntemi [13]'de ortaya konmuştur. Geleneksel FastSLAM yönteminde kullanılan doğrusallaştırma işlemlerini kullanmayan bu yaklaşım, işlem yükünü azaltırken aynı zamanda doğrusallaştırmadan kaynaklanan bilgi kayıplarını da ortadan kaldırdığı için daha az sayıda parçacığa ihtiyaç duymaktadır. Hesap yükünü azaltmak ve daha doğru sonuçlar elde etmek için geliştirilen bir diğer yaklaşım da parçacık sayısının, robotun çalışması sırasında çevrimiçi olarak değiştirilmesine yöneliktir [14, 15]. Bu yaklaşımda parçacık sayısı; sonsal dağılımın belirsizliği büyüdüğü zaman artırılıp aynı şekilde belirsizlik küçüldüğünde de azaltılmaktadır. Bunu yapmaktaki amaç, robotun bütün görevi boyunca aynı sayıda parçacık üzerinde çalışırken belirsizliğin az olduğu durumlarda gereksiz yere yapılan işlemleri ortadan kaldırmaktır.

Hesaplama hızını artırmaya yönelik diğer çalışmalar da haritaların oluşturulması üzerine yapılmıştır. [16] ve [17]'te önerildiği üzere, tek bir harita oluşturmak yerine bu haritayı daha küçük alt haritalar şeklinde oluşturmak bu çalışmalara örnektir. Benzer şekilde hibrit topolojik/metrik haritalar da işlem yükünü azaltması bakımından etkili sonuçlar vermektedir [18].

EZKH yöntemlerinin birçoğu işaretçilerin kimliğinin anlaşılmasını sağlayan ve veri ilişkilendirme adı verilen bir adımı da içermektedir. Bu adımda, sensör tarafından tespit edilen bir işaretçinin haritadaki bir işaretçi mi yoksa yeni bir işaretçi mi olduğuna karar verilir. Bunun için, tespit edilen işaretçi daha önce tespit edilenlerin hepsiyle tek tek karşılaştırılmaktadır. Haritadaki eleman sayısının büyük boyutlara ulaşmasıyla birlikte, veri ilişkilendirme aşamasının tekrar sayısını da bir o kadar artıracağı için, bu durum büyük bir hesapsal yükü de beraberinde getirmektedir. [19]'da önerilen çalışmaya göre, tespit edilen işaretçi haritadaki bütün işaretçilerle değil sadece sensörün taradığı alandakilerle karşılaştırılmaktadır. Böylece veri ilişkilendirme adımının tekrar sayısı oldukça düşmektedir. Günümüzde sensör teknolojisinin geliştiğini ve çok geniş alanları tarayabilen lazer mesafe ölçücülerin olduğunu gözönünde bulundurursak, bu yöntemde bazen işlem yükünün artabileceği anlaşılmaktadır. Bu tez çalışmasında, mevcut yöntemden esinlenerek daha gelişmiş bir yöntem üzerinde durulmuştur.

1.2 Tezin Amacı

İlk geliştirilmeye başlandığı yıllardan itibaren EZKH yöntemleri giderek artan bir şekilde ilgi görmeye başlamış ve çok çeşitli yaklaşımlar ortaya çıkmıştır. EZKH'yi bu kadar ilgi çekici yapan etkenlerin başında robot durum hesaplamalarındaki güçlükler ve işlem hızının, veri miktarı arttıkça yavaşlamasıdır. Hız performansını geliştirmek üzerine farklı bakış açılarıyla yapılan çalışmalar bulunmaktadır. Giriş bölümünde bahsedildiği üzere parçacık filtresi gibi yaklaşımlar için parçacık sayısının düşürülmesi, harita oluşturulurken daha az veri toplama veya oluşturulan haritanın daha akıllıca bir şekilde kullanılmasına yönelik çalışmalar bulunmaktadır.

Bu tez çalışmasında, EZKH ile oluşturulan harita bilgisinin tamamının kullanması yerine sadece ilgilenilen alandaki harita bilgisinin kullanılarak hız performansının artırılması amaçlanmıştır. Önerilen yöntemin performansını görebilmek için simülasyon ortamında farklı uygulama ortamları oluşturularak farklı EZKH yöntemleri ile önerilen yöntem karşılaştırılmıştır.

2. DONANIM VE YAZILIM ALTYAPISI

2.1 Turtlebot 2 Uygulama Platformu ve Kinect Sensörü

Turtlebot 2; robotik alandaki eğitim ve araştırmalar için tasarlanmış, düşük maliyetli bir uygulama platformudur. Robot Operating System(ROS) adı verilen yazılımsal çalışma alanı ile uyumlu olduğu, üzerinde kinect ve jiroskop gibi sensörleri barındırdığı için çoğu öğrencinin ve araştırmacının, çalışmalarını geliştirmek için tercih ettiği bir araçtır. Şekil 2.1'de görüldüğü üzere Turtlebot 2, üç ana kısımdan oluşmaktadır: Diğer bütün bileşenlerin bağlı olduğu ve hareketi sağlayan mobil gövde, üç boyutlu ve renkli görüntü sağlayan kinect sensörü, dizüstü bilgisayar [20].

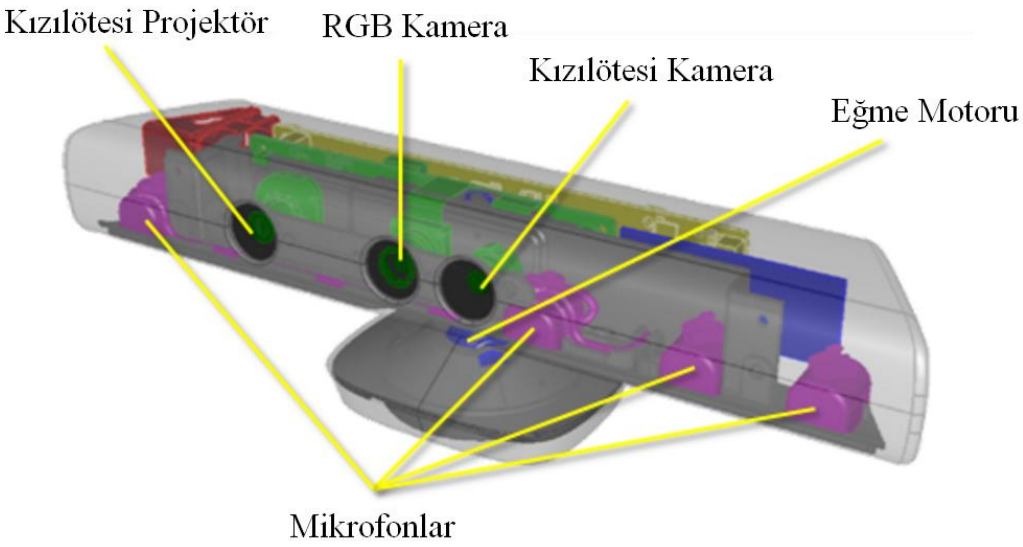
Şekil 2.1 : Turtlebot 2 [20].

Mobil gövde içerisinde, odometri verisi sağlayan ve tekerleklere bağlı olan enkoderler, üç eksende yönelim bilgisi veren jiroskop ve mobilite yeteneğini artıran çarpışma sensörü, uçurum sensörü gibi elemanlar bulunmaktadır [20].

Bilgisayar; algoritmaların geliştirildiği, mobil gövdeye komutların gönderildiği, ve sensörlerden gelen bilgilerin işlendiği kısım olması nedeniyle en önemli bileşendir. Bu işlemlerin gerçekleştirilmesi ROS ile sağlanmaktadır [21].

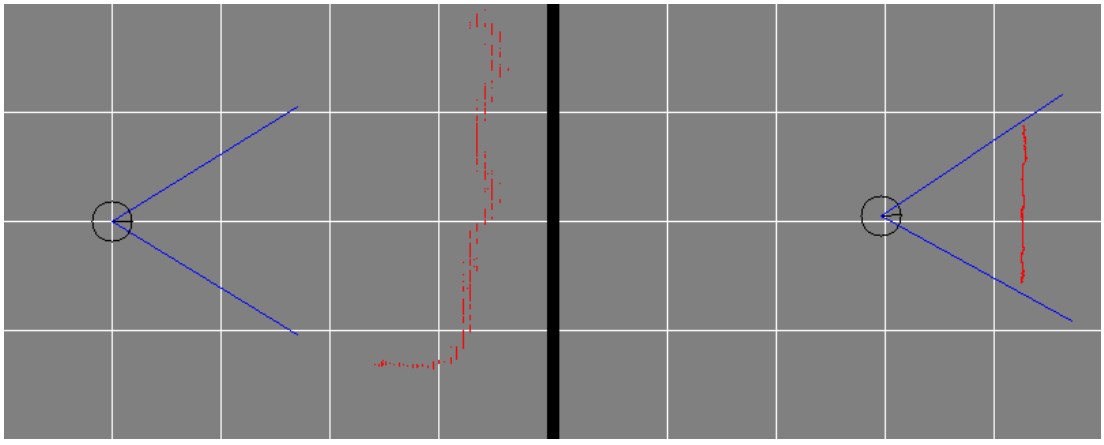
Üçüncü bileşen olan Kinect sensörü ile, farklı formatlarda sağladığı görüntüler sayesinde Turtlebot 2'yle çalışabilecek alanlar çok çeşitli olabilmektedir. Örneğin; 3 boyutlu derinlikli görüntü vermesi, 3 boyutlu EZKH uygulamaları için kolaylık sağlamaktadır [22].

Şekil 2.2'de kinect sensörünün üzerindeki RGB ve kızılötesi kameralar ve kızılötesi reflektör görülmektedir. Bunlara ek olarak dahili bir mikrofon ve ivmeölçer de bulunmaktadır [23].



Şekil 2.2 : Kinect sensörü [23].

640x320 piksel çözünürlüğünde derinlikli görüntü sağlayan kızılötesi kamera yatayda 57 derece ve dikeyde de 43 derecelik bir görüş açısına sahiptir. Ayrıca sensör, 40 santimetre yakına ve 8 metre uzaklığa kadar olan ölçümleri alabilmektedir. Bu tez çalışmasında, derinlikli görüntünün bir satırındaki veri hesaba alınarak kinect sensörü 2 boyutlu lazer mesafe ölçücü gibi kullanılmıştır. Şekil 2.3'te kinect ile elde edilmiş düz bir duvara ait tarama görüntüleri görülmektedir. 7 metre uzaklıktan alınan ölçümler oldukça gürültülü iken, EZKH uygulamasında kullanılabilecek doğrulukta ölçümler alınabilmesi için uzaklığın 3 metreye kadar düşürülmesi gerekmektedir [23].



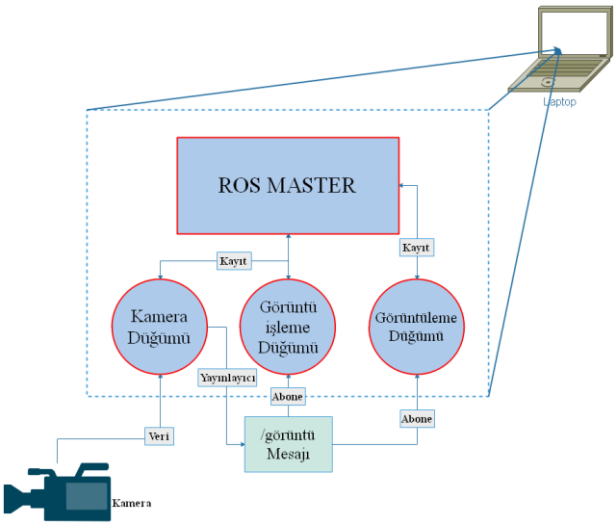
Şekil 2.3 : Kinect ile 7 metre uzaklıktaki (solda) ve 3 metre uzaklıktaki (sağda) düz bir duvardan alınan ölçümler.

2.2 Robot İşletim Sistemi (ROS)

Robotik çalışmaların akademik alanda ve eğitim alanında iyice yaygınlaşmasıyla birlikte robot, bilgisayar ve sensörler gibi bileşenler arasında iletişimin kolaylaştırılmasını ve bu alandaki çalışmaların kolay bir şekilde geliştirilmesini sağlayacak tek bir sistemin oluşturulması gereksinimi doğmuştur. Bu durum düşünülerek 2007 yılında Stanford Üniversitesi'ndeki bir grup araştımacı tarafından geliştirilmeye başlanan ROS, açık kaynak kodlu ve ücretsiz olması nedeniyle bugün çok yaygın bir şekilde kullanılmaktadır [24].

ROS'un işleyişi temel olarak şu şekildedir: Belli görevleri olan düğümler arasında yayınlayıcı ve abone tipindeki mesajlar ile veri iletimi sağlanır. Bu iletimin sağlanabilmesi ve düğümlerin birbirine bağlanabilmesi için ROS Master adında bir yönetim yapısı vardır. Düğümler tek başlarına da olabilirken, bir paket içerisinde de olabilir. Farklı robotlar üzerinde de uygulanabilen bu paketler; içlerinde düğüm, ROS'a bağlı kütüphaneler, veri kümeleri, yapılandırma dosyaları ve paketin işlevselliğiyle ilgili başka dosyalar da barındırır [25].

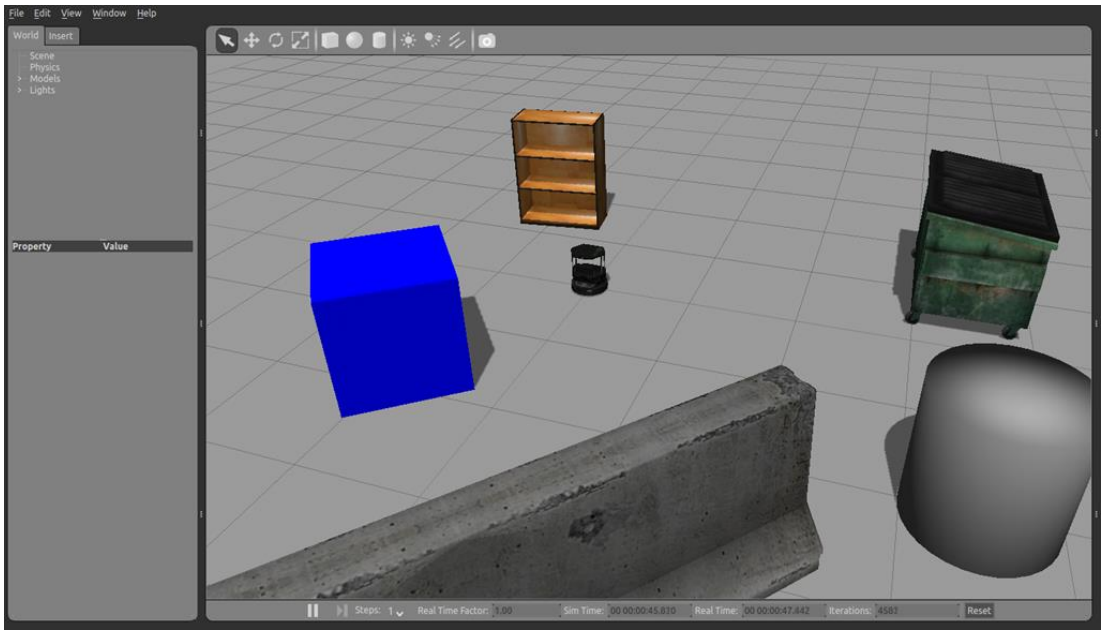
Şekil 2.4'te ROS'un işleyişini bir örnek üzerinden anlatan bir şema görülmektedir. İlk olarak bütün düğümlerin ROS Master yönetiminde birbirlerinden haberdar olması sağlanmaktadır. Kameradan gelen görüntü verisi, görevi kamera ile iletişim kurmak olan görüntü düğümüne aktarılmaktadır. Bu düğüm ROS Master'a kayıt olurken /görüntü isminde bir konu yayınlayacağını bildirirken görüntü işleme düğümü ve görüntüleme düğümü de yayınlanan bu konuya abone olduklarını bildirir.



Şekil 2.4 : ROS işleyişinin örneklendirilmesi.

2.3 Gazebo

Robotik alanındaki uygulamalarda sensörler ve eyleyiciler gibi bileşenlerin genellikle pahalı olması, özel çalışma ortamlarına ihtiyaç duyulması ve uygulama sırasında insanların veya çevredeki eşyaların zarar görmesi olasılığı bu tür çalışmaların öncelikle simülasyon ortamında denenmesini gerektirmektedir. Gazebo; bu gereksinime cevap veren, iç ve dış ortamlarda kullanılan robotların üç boyutlu simülasyonunu gerçekleştirebilmek için geliştirilmiştir. ROS ile uyumlu olması; IMU, lazer, sonar ve kinect gibi sensörleri yapısında hazır olarak bulundurması gibi özelliklerinden dolayı çok kullanışlı bir yazılımdır. Bunların haricinde Gazebo; Turtlebot ve Husky gibi robotları da hazır olarak sunduğu gibi araştırmacıların kendi tasarladıkları robot, eyleyici veya kontrolörleri de Gazebo içerisinde kolaylıkla oluşturmaya olanak sağlamaktadır. Özgün çalışma ortamları oluşturabilmek için kütüphanesinde bulundurduğu çok çeşitli şekiller ve hazır cisimler kullanılabilir. Şekil 2.5'te Gazebo'da oluşturulmuş bir ortam görülmektedir [26].



Şekil 2.5 : Gazebo simülatöründe hazırlanan örnek bir ortam.

2.4 OpenCV

OpenCV; C/C++ programlama dillerinde yazılmış ve Linux, Windows ve Mac OS X işletim sistemleri altında çalışabilen açık kaynak kodlu bir görüntü işleme

kütüphanesidir. Görüntü işleme çalışmalarında işlemsel verimlilik ve gerçek zamanlı uygulamaların kullanımı için tasarlanmıştır [27]. Başlıca özellikleri şunlardır:

- Resim ve video görüntüleme
- Görüntü üzerinde matris ve operatör işlemleri uygulama
- Görüntü yumuşatma ve keskinleştirme
- Kamera kalibrasyonu
- Özellik çıkarımı
- Nesne tespit etme
- Görüntü bölme ve birleştirme [27]

Bu tez çalışmasında OpenCV kütüphanesinden, geliştirilen yeni EZKH uygulamasının sonuçlarını görselleştirmek için yararlanılmıştır. Gazebo simülatöründe hazırlanan ortamda hareket ettirilen robotun izlediği gerçek yol, hesaplanan yol, işaretçilerin gerçek ve hesaplanan konumları ve bu konumlara ait belirsizliği gösteren hata elipsleri çizdirilmiştir.

3. EŞ ZAMANLI KONUMLANDIRMA VE HARİTALAMA

Birinci bölümde bahsedildiği gibi; mobil bir robotun, bilmediği bir ortamda ve bilmediği bir konumda hareket etmeye başladığında o ortamın haritasını çıkarırken aynı zamanda o haritaya göre konumunu belirlemeye çalışması işlemine eş zamanlı konumlandırma ve haritalama (EZKH) denir. Çoğu EZKH algoritmalarının olasılıksal yöntemler kullandığı gözönünde bulundurulursa çevrimiçi (online) EZKH ve tam (full) EZKH olmak üzere iki ayrı yaklaşım olduğu söylenebilir. Çevrimiçi EZKH için sonsal durum berlirten denklem 3.1'de robot durumunun sadece t anındaki bilgisi hesaplanırken, denklem 3.2'de tam EZKH için robot durumunun başlangıçtan t anına kadar olan bütün bilgileri hesaplanır.

$$p(s_t, m|z_{1:t}, u_{1:t}) \tag{3.1}$$

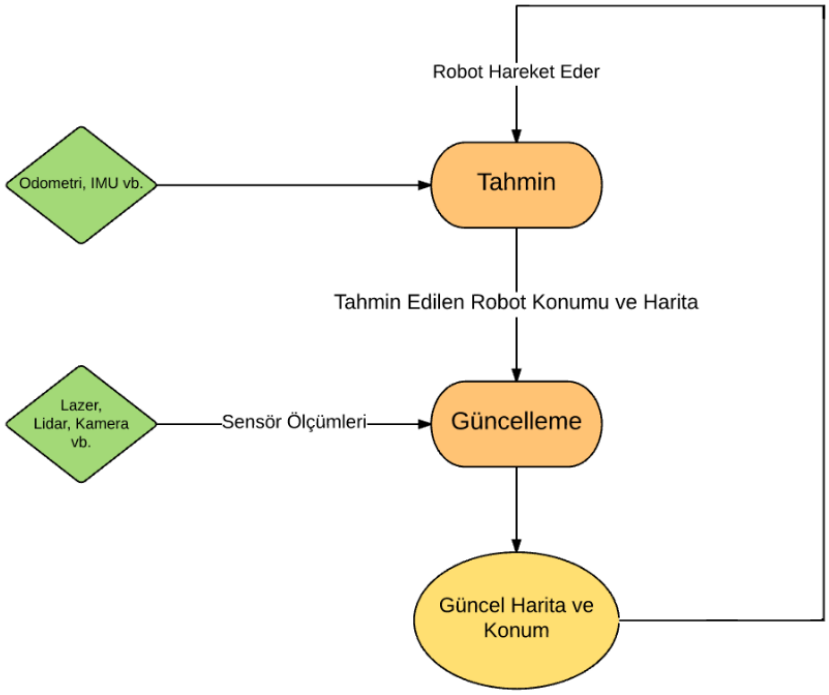
$$p(s_{t:1}, m|z_{1:t}, u_{1:t}) \tag{3.2}$$

Mevcut EZKH yöntemleri işleyiş olarak birbirlerinden farklı olsalar da şekil 3.1'deki akış şemasındaki gibi temel bir prosedür vardır. Öngörü (predict) ve güncelleme (update) olmak üzere iki adım vardır. Öngörü aşamasında; tekerleklere bağlı bir enkoder, IMU (Inertial Measurement Unit) ve GPS (Global Positioning System) gibi hareket algılayıcılarından alınan veriler kullanılarak robotun $t-1$ anına ait konumuna göre t anındaki konumu hesaplanır. Güncelleme aşamasında ise lazer ve kamera gibi algılayıcılarla çevreden alınan ölçümlere göre tahmin aşamasında elde edilen konum bilgisi güncellenerek daha doğru hale getirilmeye çalışılır. Bu aşamada aynı zamanda harita da güncellenir [3, 28, 29].

3.1 EZKH'de Kullanılan Temel Kavramlar

3.1.1 İşaretçi (Landmark)

Mobil bir robotun konumunu güncellerken referans olarak yararlandığı çevresindeki belirgin noktaların veya cisimlerin ortak adı olan işaretçiler, EZKH tekniklerinde yaygın olarak kullanılan bir kavramdır. Bu noktalar kapı dikmesi veya odaların köşeleri gibi noktasal koordinatlarla belirtilebilecek yerler olabilir [2].

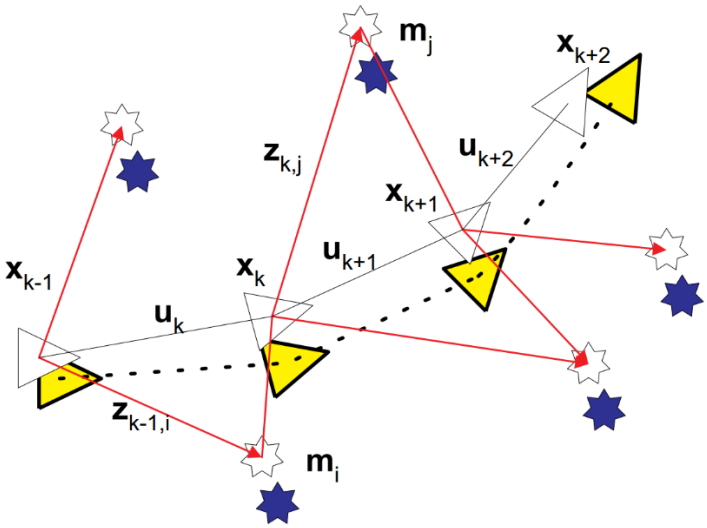


Şekil 3.1 : Çoğu EZKH yöntemleri için temel akış şeması.

Otonom bir robot hareket ederken, yer göstericilerden şekil 3.2'de gösterildiği gibi yararlanır. x_k robotun k anındaki konumunu ve yönelimini, u_k $k-1$ anında robota uygulanan kontrol girişini, m_i zamana bağlı olarak değişmeyen i'nci işaretçinin konumunu ve $z_{i,k}$ da k anında tespit edilen i'nci işaretçi için ölçüm uzaklığını belirten vektörlerdir. Bu örnekte robot, $k-1$ anında tespit ettiği yeni işaretçiye o anki hesaplamış olduğu kendi konumuna ve ölçüm bilgisine göre bir konum ataması yapar. k anında aynı işaretçiyi yeniden gördüğü zaman işaretçiye $k-1$ anında atamış olduğu konum bilgisini ve k anında aldığı ölçüm bilgisini kullanarak hem kendi konumunu hem de işaretçinin konumunu günceller.

İşaretçilerden en iyi şekilde yararlanılabilmesi için bazı özelliklere sahip olması gerekir. İlk olarak bir işaretçi, robot ilerlediği zaman yeniden tespit edilebilmelidir. Çünkü robot ne kadar ilerlediğini bulabilmek için k anında bulduğu ve $k-1$ anında bulduğu işaretçiler arasında bağ kurar. Ayrıca her bir işaretçinin birbirinden kolayca ayırt edilebilir olması gerekir. İki farklı işaretçi aynı işaretçi gibi algılanırsa hatalı güncelleme yapılır.

Aranan bir başka özellik de sayılarının yeteri kadar çok olmasıdır. Çünkü az sayıda işaretçi demek robot konumunun ve haritanın daha az güncellenmesi demektir.



Şekil 3.2 : EZKH'de yer göstericilerin kullanımı [30].

Son olarak da işaretçiler sabit cisimlerden seçilmelidir. Örneğin; hareket halindeki bir insan, robotun kendi konumu hakkında hatalı bir sonuca ulaşmasına neden olabilir [31].

İşaretçi çıkarımı için literatürde bir kaç farklı yöntem önerilmiştir. Bu yöntemlerden hangisinin uygulanacağının seçimi aslında kullanılan sensörler ve çıkarılmak istenen işaretçilerin çeşidine bağlıdır. Örneğin, lazer tarayıcı ya da lidar gibi sensörler için keskin kenarlı işaretçi çıkarımı (Spike Landmarks Extraction), RANSAC (Random Sample Consensus), tarama eşleştirme (Scan Matching) ve geometrik şekilli işaretçi çıkarımı gibi yöntemler geliştirilmiştir. İşaretçinin konumunun kesinliğinin yüksek olması istendiği durumlarda bu yöntemler ve sensörler kullanılmaktadır. Diğer yandan, tespit edilen işaretçilerin sınıflandırılmasının çok önemli olduğu durumlar da söz konusu olabilir. Dış ortamlarda araç, insan ve hayvan gibi hareketli nesnenin sayısının fazlalığı bunların tespit edilmesini önemli hale getirir. EZKH algoritmalarında hareketli nesnelerin algılanması işi kamera yardımıyla çok daha kolaydır. Bu gibi görüntü tabanlı yer gösterici çıkarımlarının zayıf yanı ise uzaklık bilgisinin lazer sensörlere göre daha belirsiz olmasıdır [28, 31, 32, 33].

3.1.1.1 Keskin kenarlı işaretçi çıkarımı

Bu yöntemde, sensörden elde edilen veri taranırken büyük miktardaki değişimler dikkate alınır. Yani ölçüm yapılan alandaki duvar köşeleri, sandalye ve masa bacakları gibi belirgin ve keskin şekilli cisimler tespit edilir. Bir veri dizisinde bir eleman, kendisinden önceki ve sonraki elemanlarla karşılaştırılır. Bu elemanlar arasındaki fark

belli bir değerden fazla ise işaretçi olarak tanımlanır. Keskin şekilli eşyaların, duvar köşelerinin sıkça rastlandığı kapalı alan uygulamaları için oldukça elverişli olan bu yaklaşımda genellikle yüzeyinde girinti ve çıkıntı olmayan cisimler içeren açık alanlarda yeterli sayıda işaretçi tespit edilmesi zordur [28, 31].

3.1.1.2 RANSAC

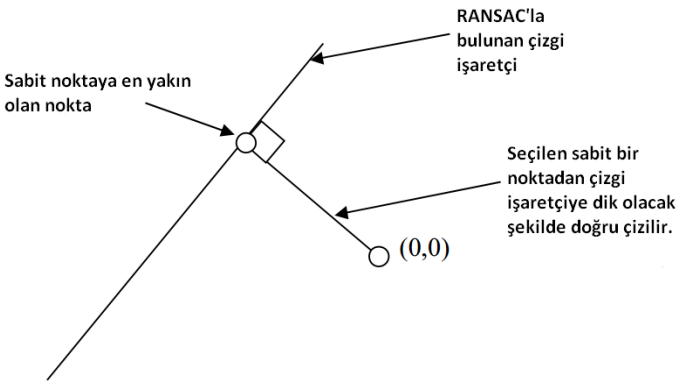
RANSAC; belirlenen bir matematiksel model için bir veri kümesi içinde o modeli, verileri belli bir eşik değerini geçecek sayıda kapsayacak şekilde örneklendiren tekrarlı olarak uygulanan bir tekniktir [3]. İşaretçi tabanlı EZKH uygulamalarında RANSAC, rastgele seçilmiş lazer taramalarından alınan verileri kullanarak bu verilere en uygun olan doğruyu bulmak için kullanılır. Daha çok, robotun bulunduğu ortamdaki duvarların algılanmasında kullanılan bu teknik, robotun konum ve yönelim bilgilerinin güncellenmesinde oldukça faydalıdır. Bir diğer faydası da insan gibi hareketli nesnelerin işaretçi olarak algılanmasını önlemesidir. Çünkü EZKH'de doğru bir konumlandırma için sabit olan nesneler tercih edilir. Bir RANSAC algoritması genel olarak şu şekildedir:

- Model parametrelerini belirlemek için gerekli en az sayıda veriyi gelişigüzel seç.
- Bu verileri kullanarak modeli örneklendir.
- Bütün veri kümesi içinde, örneklendirilen bu modele daha önce belirlenmiş olan hata payını geçmeyen yani modele uygun verileri (inliers) bul.
- Uygun verilerin sayısının bütün verilerin sayısına oranı belli bir eşik değerini geçene kadar algoritmayı baştan itibaren tekrarla.
- Eşik değeri aşıldığında uygun verilerin tamamını kullanarak model parametrelerini yeniden hesapla ve algoritmayı sonlandır.

Bu algoritma eşik değere ulaşmak için sonsuz defa döndürülmez. Onun yerine en çok belli bir sayıda dönecek şekilde ayarlanır. Bu sayı az olduğu zaman işlem hızı yüksek olur ancak doğruluk payı düşük olur. EZKH'de bu algoritma büyük ölçüde aynı şekilde uygulanırken çizgi modeli oluşturulmasında en küçük kareler yöntemi kullanılır.

Şekil 3.3'te görüldüğü gibi bir EZKH uygulamasında robotun konum ve yönelim bilgisi, çizgi olarak tespit edilen işaretçiler nokta işaretçiymiş gibi varsayılarak güncellenebilir. Robotun bulunduğu ortamdan sabit bir nokta seçilir ve şekilde olduğu gibi bu noktaya çizgi üzerindeki en yakın nokta bulunur. Daha sonra çizgi üzerindeki

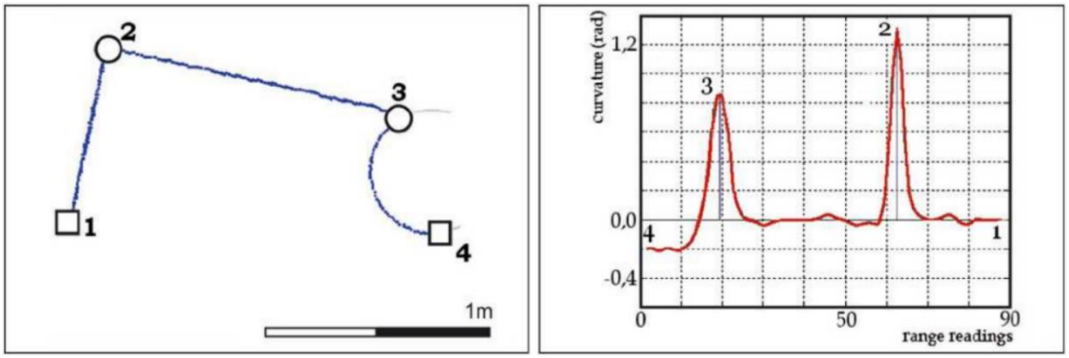
bu sabit nokta ve robotun konumu kullanılarak uzaklık ve açı bilgileri basit trigonometrik hesaplarla bulunur [28, 31, 34].



Şekil 3.3 : RANSAC yöntemi ile tespit edilen duvarın işaretçi olarak kullanılması [31].

3.1.1.3 Geometrik şekilli işaretçi çıkarımı

İki boyutlu ölçüm alan lazer sensör kullanan robotlar için çok kullanışlı olan bu yöntemde lazerden elde edilen veri kullanılarak bir eğrilik fonksiyonu (curvature function) elde edilir. Bu fonksiyonun iki boyutlu görselleştirilmesiyle ortaya çıkan şekilde ortamdaki yuvarlak, düz veya köşeli cisimlere karşılık gelen yerler oluşur. Şekil 3.4'te yuvarlak ve köşeli şekillerin olduğu bir ortama ait lazer ölçümü ve bu ölçümden elde edilen eğrilik fonksiyonun çizimi görülmektedir. Lazer görüntüsünde 2 ve 3 ile işaretlenmiş köşe noktaları eğrilik fonksiyonundaki eğrilerin zirve noktalarına karşılık gelmektedir. Bu köşe noktaları içe doğru değil de dışa doğru olsalardı eğri kısımlar negatif değerler alarak aşağıya doğru olacaklardı. Lazer ölçümündeki yuvarlak şeklin karşılığı eğrilik fonksiyonunda x eksenin altında düz bir çizgi olmuştur. Duvar gibi düz şekiller ise bu fonksiyonda sıfıra yakın değerler alarak x ekseninin üzerinde düz bir çizgi olarak karşılık bulmuştur [35].



Şekil 3.4 : Lazer ile ölçümü yapılan bir ortamın 2 boyutlu görüntüsü (solda) ve bu ölçümden elde edilen eğrilik fonksiyonu (sağda) [35].

Lazer verisinden eğrilik fonksiyonunu elde etmek için ölçüm verisi, (x, y) koordinatlarına dönüştürüldükten sonra x ve y'deki veriler bir boyutlu diziler haline getirilir. Bu dizilerin birinci dereceden ve ikinci dereceden türevleri alınarak denklem 3.3'teki gibi kullanılır ve eğrilik fonksiyonu elde edilir [35].

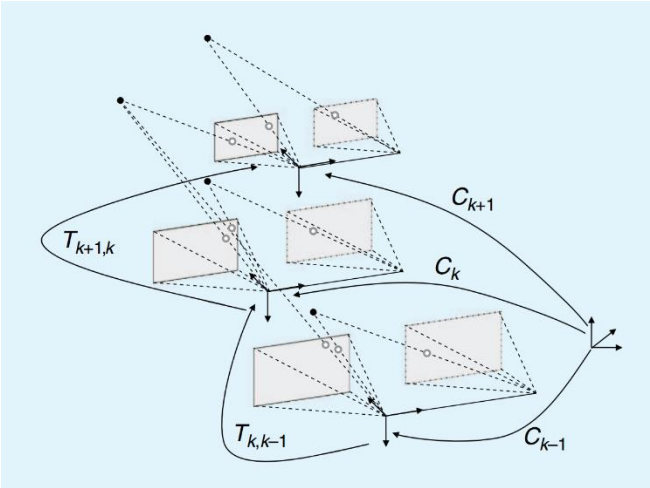
$$K(t) = \frac{\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}(t)^2 + \dot{y}(t)^2)^{2/3}} \tag{3.3}$$

3.1.2 Odometri ve konum tahmini (dead reckoning)

Denizcilik ve havacılıkta yaygın olarak kullanılan dead reckoning robotik uygulamalarda da kullanılan bir yöntemdir. GPS gibi referans kullanan sistemler olmadığı zaman konum tahmin etme ve belirleme için bu yönteme başvurulur. Dead reckoning, robotun hız ve yönelim gibi bilgilerini kullanarak şimdiki konumundan periyodik bir zaman aralığına göre bir sonraki konumunun hesaplanması işlemidir. Teoride, diferansiyel tahrikli mobil robotlar için bu teknik doğru bir konum hesaplamada yeterlidir. Ancak modelleme hataları, tahrik elemanlarının komut farkı ve tekerleklerin kayması gibi gerçek hayatta karşılaşılan sıkıntılardan dolayı olasılıksal hesaplamalardan yararlanılır [36]. Odometri ve dead reckoning birbirine çok benzer kavramlar olmakla birlikte verilerin elde edilmesi biraz farklıdır. Odometri'de veriler sadece hareket sensörlerinden alınırken dead reckoning'de ek olarak pusula, jiroskop ve imu (Inertial Measurement Unit) gibi dünyanın manyetik alanını kullanan sensörler de kullanılır [37].

Görüntü işleme tekniklerinin gelişmesiyle birlikte kameradan alınan görüntülerle yer değiştirme ve yönelimin belirlenmesi üzerine çalışmalar yapılmaya başlamıştır. Hareket halindeki bir robot üzerindeki sabit bir kameradan farklı zamanlarda alınan görüntüler arasındaki farklardan yola çıkarak odometri hesabı yapılmasına görsel odometri (visual odometry) denir. Tek bir kamerayla yapılabildiği gibi iki kamerayla stereo görüntüler elde ederek de yapılabilir. Genellikle bir görüntü üzerindeki öznitelikler (features) tespit edilip bir sonraki görüntüde aynı özniteliklerin karşılıkları bulunur ve özniteliğin görüntü üzerindeki koordinat değişimine göre hesaplama yapılır. Şekil 3.5'te iki kamera kullanılarak odometri hesabının nasıl yapıldığı görülmektedir. $k-1$ anında iki kameradan alınan görüntüler üzerindeki aynı öznitelik tespit edilir. Bu özniteliğin, üçgenleme yöntemiyle kameraya olan uzaklığı hesaplanır. k anında tekrar görüntü alınır ve $k-1$ anında tespit edilen öznitelik tekrar tespit edilir

ve kameraya olan uzaklığı hesaplanır. İki farklı zamanda elde edilen uzaklık bilgisinin farkı robotun ne kadar yer değiştirdiğini belirtir [38].



Şekil 3.5 : İki kamera ile görsel odometrinin hesaplanması [38].

Görsel odometrinin tekerleklerden elde edilen odometriye göre bazı üstünlükleri vardır. Tekerlekler patinaj yaptığı zaman geleneksel odometride hatalar oluşurken bu tarz bir hata görsel odometride meydana gelmez. Kameralar aynı zamanda haritalama amaçlı da kullanılabileceği için hem konumlandırma hem de haritalama işini tek bir sistemle çözerek fazla sensör kullanımını ortadan kaldırmış olurlar. Küçük ve hafif olmaları ve düşük enerjiyle çalışabilmeleri, kameralı odometri sistemlerini bir adım öne çıkarmaktadır. Bunların yanında görsel odometrinin güçlük yaşadığı bazı durumlar da vardır. Yetersiz aydınlatma, ışık yoğunluğunun sürekli değişmesi ya da rüzgar gibi etkenlerden dolayı görüntü alınan ortamın dinamik hale gelmesi gibi sebepler hesaplama güçlüğü oluşturur. Tek kameralı görsel odometride uzaklık bilgisi doğrudan alınamadığı için ek bir sensöre gereksinim duyulması da bir başka zayıflıktır [39].

3.1.3 Veri ilişkilendirme (data association)

Veri ilişkilendirme, robotun oluşturduğu haritadaki bilgiler ile o anda elde edilen ölçüm arasında bağlantının kurulması ile ilgilidir. Bir başka deyişle; gerçek dünyada tek veya aynı nesneye karşılık gelen ve farklı noktalardan ölçüm yapılarak elde edilen iki verinin birbiriyle ilişkilendirilmesidir [40]. Veri ilişkilendirme hem harita çıkarmada hem de konum belirlemede çok büyük bir öneme sahiptir. Çünkü elde edilen veriler arasında doğru bir ilişki kurulamazsa her adımda oluşan hatalar birikerek harita ve konumda büyük yanlışlıklara sebep olabilir. Veri ilişkilendirme için

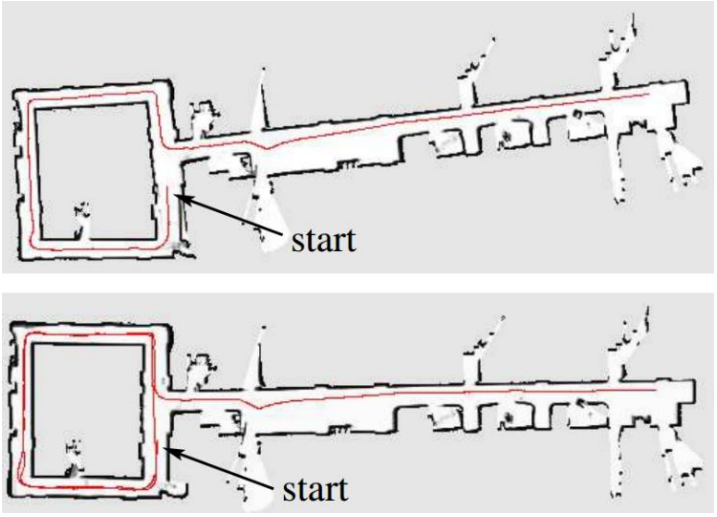
literatürde birden fazla yöntem bulunmaktadır. Bunlardan bazıları aşağıdaki gibi sıralanabilir [41]:

- En büyük olabilirlik (maximum likelihood)
- Joint compatibility branch and bound
- En yakın komşuluk(nearest neighbor)
- Combined constraint data association
- Random sample consensus (RANSAC)

Bu yöntemler içerisinde uygulanabilirliği en kolay olan en büyük olabilirlik yöntemi, bu tez çalışmasında ele alınan EZKH yönteminde kullanılmıştır.

3.1.4 Döngü kapama

EZKH algoritmalarında ortamın haritasının çıkarılması ve konum belirlenmesinin en hızlı şekilde olması istenir. Buna bir çözüm olarak mobil robotlar keşif yaptığı bir yerden tekrar geçmez. Ancak bu durumda daha önce keşfedilen yerlerle veri ilişkilendirilmesi yapılmadığından dolayı haritada ve konumda birikmiş hataların azaltılması yapılamaz. Döngü kapama, daha önce geçilen yerden tekrar geçilerek haritada ve konumda düzeltme işlemi uygulanmasına izin veren bir işlemdir. Şekil 3.6'da döngü kapamaya bir örnek verilmiştir. Üstte robotun döngü kapama yapmadan hemen önceki oluşturduğu harita görülmektedir. Bu haritada koridor ve kare alan arasında açısal bir hata meydana gelmiştir. Altta ise döngü kapama yapıldıktan sonra haritanın düzeltilmiş olduğu görülmektedir [24].



Şekil 3.6 : Döngü kapamanın uygulanmasıyla ilgili karşılaştırmalı bir çalışma [42].

3.1.5 Hareket modeli

Konum belirlemenin en doğru şekilde yapılabilmesi için öncelikle robotun mekanizması da dikkate alınarak uygun bir hareket modelinin belirlenmesi gerekir. Robotik uygulamalarda yaygın olarak hız komutlarına dayalı veya odometri verisini giriş komutu olarak kullanan iki farklı model ve bunların türevleri kullanılır. Hıza dayalı modeller, belli bir örnekleme aralığında motorlara sabit bir hız komutu uygulandığını kabul eder. Böylece mevcut durum bilgisi ve giriş komutları kullanılarak sonraki durum tahmin edilebilir. Bunun yanında odometri verisi, giriş komutları uygulandıktan sonra okunabilir. Bu yüzden, hız tabanlı modeller genellikle yol planlama uygulamaları için daha uygunken odometri tabanlı modeller ise EZKH uygulamaları için daha uygundur. Ayrıca, tahrik elemanlarına uygulanan komutun gerçekleştirilmesi sırasında oluşan belirsizlik tekerleklere bağlı enkoderlerden alınan verinin belirsizliğinden daha büyük olduğu için odometri temelli modeller konum tahmin etmede daha başarılı sonuçlar vermektedir [3].

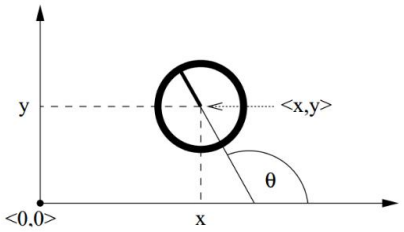
3.1.5.1 Odometri modeli

Bu tez çalışmasında kullanılan diferansiyel tahrikli Turtlebot platformunun yapısı göz önüne alınarak ve birinci bölümde anlatıldığı üzere belirsizliğe sebep olan etmenler de düşünülerek olasılıksal bir yöntem olan odometri hareket modelinin uygulanmasına karar verilmiştir. Bu modelde robot, kontrol girişi olarak periyodik zaman aralıklarında okunan odometri verisini kullanır. Diğer bir deyişle, t anında okunan değer ile $t-1$ anında okunan değer arasındaki fark kontrol komutu gibi kabul edilir. Odometri hareket modeli, kullanılan koordinat sistemine ve EZKH algoritmasına göre farklı şekillerde uygulanabilir. Bu yüzden öncelikle bazı kinematik kavramların açıklanmasında fayda vardır.

Üç boyutlu uzayda bir mobil robotun durum bilgisi altı farklı değişkenle ifade edilir. Bunlar; robotun konumunu belirten üç boyutlu kartezyen koordinatları (x, y, z) ve yönelimini belirten euler açılarıdır (Yuvarlanma (Roll), Yunuslama (Pitch), Sapma (Yaw)). Düzlemsel bir ortamda ise konum bilgisi iki boyutta ve yönelim bilgisi de tek bir açıyla ifade edilebilir. Böylece, denklem 3.4'teki durum matrisi kullanılarak diferansiyel tahrikli bir robotun hareketi modellenebilir. Burada x ve y iki boyutlu koordinat sistemindeki konumu belirtirken θ yönelim açısını belirtir.

$$X = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{3.4}$$

Şekil 3.7'de görüldüğü gibi konum, ortamdaki belli bir noktanın orijin olarak kabul edildiği bir koordinat sistemine göre hesaplanır. Öyle ki; bu orijin noktası genellikle robotun harekete ilk başladığı nokta olarak kabul edilir ve bu koordinat sisteminin x ekseni robotun bu noktadaki yönelim açısına göre belirlenir. Çünkü SLAM uygulamalarına göre robotun, başlangıç anında ortam hakkında bir bilgisi ve referans alabileceği bir koordinat sistemi yoktur.



Şekil 3.7 : Robot konumunun global koordinat sistemine göre gösterilmesi [3].

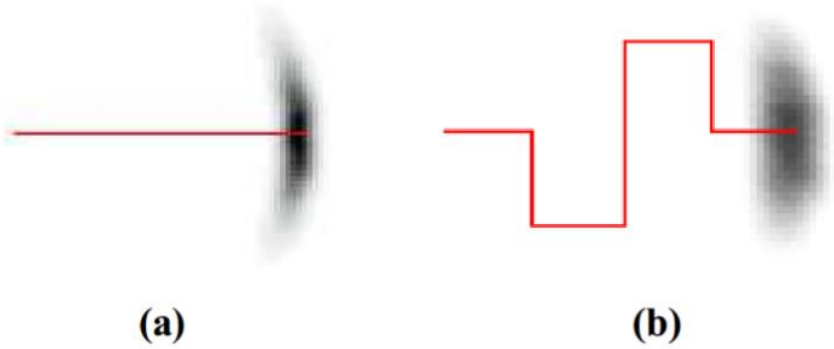
Olasılığın büyük öneme sahip olduğu mobil robot çalışmalarında sıkça karşılaşılan bir kavram da sonsal durumdur. Robotun o anki olası durumu hakkında bilgi veren sonsal durum, denklem 3.5'teki gibi gösterilmektedir. Burada s_t şu anki durumu, s_{t-1} kontrol girişi uygulanmadan önceki yani bir önceki durumu ve u_t de kontrol girişini ifade etmektedir.

$$p(s_t|u_t, s_{t-1}) \tag{3.5}$$

Şekil 3.8'deki çizimler bir robota ait sonsal durumun iki boyutlu olarak görselleştirilmiş halidir. Düzlemsel bir ortamda hareket komutu uygulanmış robotun konumunun belirsizliğini gösteren sonsal durumda koyu renkli alanlardan açık renkli alanlara doğru gidildikçe robotun tahmin edilen konumunun olasılığı azalmaktadır. (a)'da düz bir hareket sonucunda ortaya çıkan dağılımın kısmen küçük ve hilal şeklinde olduğu görülürken (b)'de ise hem düz hem de dönel hareketlerin bileşiminin sonucunda dağılımın daha geniş ve elips şekline benzer olduğu görülmektedir.

$$prob(a, b) = \frac{1}{\sqrt{2\pi \cdot b^2}} e^{-\frac{1}{2}\frac{a^2}{b^2}} \tag{3.6}$$

Odometri hareket modelinde denklem 3.7'deki sonsal durum hesaplanırken kontrol girişleri olarak t ve $t-1$ anındaki sensör okumaları kullanılır.

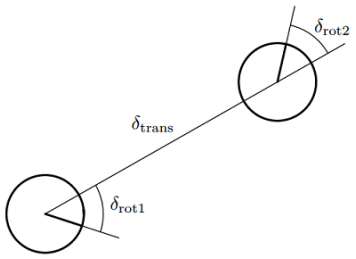


Şekil 3.8 : Hareket komutu uygulanan bir robotun durumunun sonsal dağılım şeklinde gösterimi [3].

Denklem 4'teki kontrol girişleri $s_{t-1} = (\bar{x}\ \bar{y}\ \bar{\theta})$ ve $s_t = (\bar{x}\ \bar{y}\ \bar{\theta})$; $[t-1, t]$ zaman aralığında uygulanan hız komutları sonucunda odometri verisindeki değişimi belirtir.

$$u_t = \begin{pmatrix} \bar{s}_t \\ \bar{s}_{t-1} \end{pmatrix} \tag{3.7}$$

Sensörlerden alınan bu iki andaki durum verisi kullanılarak üç değişim elde edilir. İlk olarak şekil 3.9'daki robotun ilk konumu ile ikinci konumu arasındaki doğru parçasının yukarıda bahsedilen global koordinat sistemine göre açısı ile birinci konumdaki yönelim açısının farkı (δ_{rot1}), ardından iki konum arasındaki uzaklık (δ_{trans}) ve son olarak da robotun son konumundaki yönelim açısı ile iki konum arasındaki doğru parçasının açısının farkı (δ_{rot2}) hesaplanır.



Şekil 3.9 : Odometri hareket modeliyle hesaplanan bağıl yer değişimi ve açı değişimi [3].

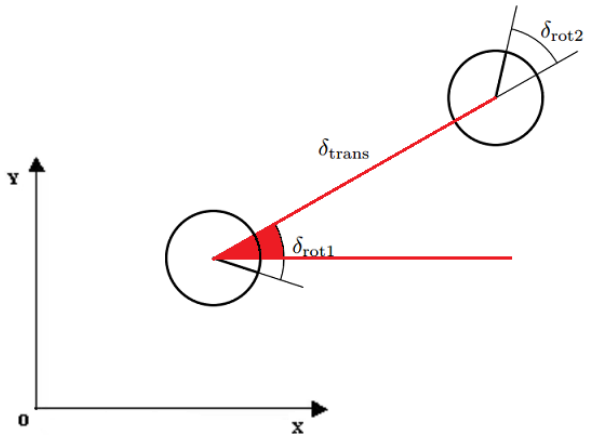
Hesaplanan bu üç değişim $t-1$ anındaki s_{t-1} durumuna eklenerek son durum yani s_t bulunur. Böylece robot konum tahmini her adımda odometri okumalarına göre güncellenmiş olur.

Çizelge 3.1, $p(x_t|u_t, x_{t-1})$ sonsalının nasıl hesaplandığını göstermektedir. Kontrol girişleri olarak odometri okumalarını kullanan algoritmada yukarıda bahsedilen δ_{rot1}, δ_{trans} ve δ_{rot2} 2. satırdan 4. satıra kadar olan denklemlerde hesaplanmaktadır.

Çizelge 3.1 : Odometri hareket modeli algoritması

1:	odometri-hareket-modeli(x_t, u_t, x_{t-1}):
2:	$\delta_{rot1} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x},) - \bar{\theta}$
3:	$\delta_{trans} = \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2}$
4:	$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$
5:	$\hat{\delta}_{rot1} = \text{atan2}(y' - y, x' - x) - \theta$
6:	$\hat{\delta}_{trans} = \sqrt{(x' - x)^2 + (y' - y)^2}$
7:	$\hat{\delta}_{rot2} = \theta' - \theta - \delta_{rot1}$
8:	$p_1 = prob(\delta_{rot1} - \hat{\delta}_{rot1}, \alpha_1\hat{\delta}_{rot1} + \alpha_2\hat{\delta}_{trans})$
9:	$p_2 = prob(\delta_{trans} - \hat{\delta}_{trans}, \alpha_3\hat{\delta}_{trans} + \alpha_4(\hat{\delta}_{rot1} - \hat{\delta}_{rot2}))$
10:	$p_1 = prob(\delta_{rot2} - \hat{\delta}_{rot2}, \alpha_1\hat{\delta}_{rot2} + \alpha_2\hat{\delta}_{trans})$
11:	$return\ p_1 \cdot p_2 \cdot p_3$

2. satırdaki $atan2()$ fonksiyonunda s_{t-1} ve s_t'deki x ve y konumlarının farkı alınarak şekil 4'te görüldüğü gibi doğru parçasının global koordinat sistemine göre olan açısı bulunur. Robotun yönelimi hesaplanan bu değerden çıkarıldığı zaman ilk dönme açısı δ_{rot1} elde edilmiş olur. $[t-1, t]$ zaman aralığında robotun yer değiştirmesi $\|.\|_2$'ye göre 3. satırda hesaplanır. 4. Satırda, t anındaki odometri ölçümünden elde edilen yönelim açısından $t-1$ anındaki yönelim açısı ve δ_{rot1} açısı çıkarılarak ikinci açı değişimi olan δ_{rot2} bulunur.



Şekil 3.10 : Global koordinat sistemine göre yönelim açısının hesaplanması [3].

Buraya kadar yapılan hesaplamalar odometri verisinin gürültüsüz olduğu düşünülerek yapıldı. Bundan sonraki kısımda (5., 6. ve 7. satır) odometri ölçümlerinin hatalı olduğu da hesaba katılarak t anına ait rastgele bir şekilde bir durum seçilir. Burada rastgele olarak seçilen durum aslında şekil 3.8'de görülen koyu ve açık renkli alanları, yani

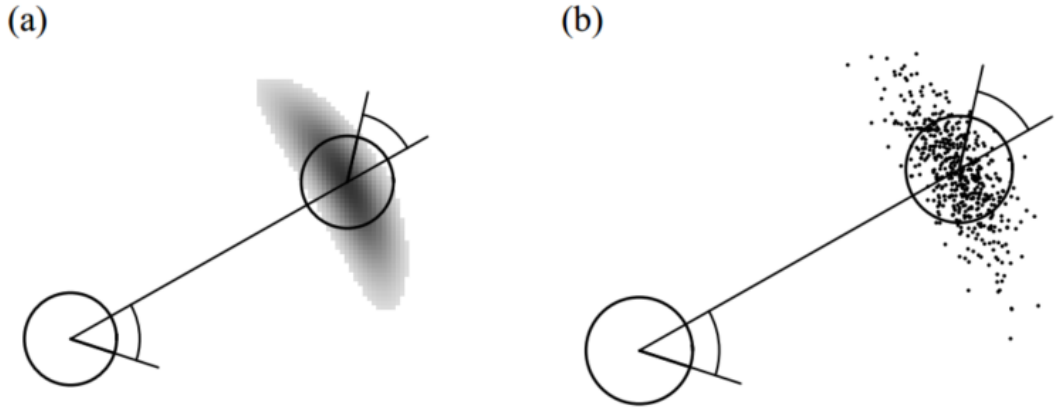
olası durumları belirtmektedir. Seçilen bu durum ve $t-1$ anında hesaplanan durum arasındaki değişimler de 2., 3. ve 4. satırda olduğu gibi hesaplanır.

Son kısımda; odometri ile hesaplanan durum ve rastgele seçilen durum arasındaki hata olasığılı hesaplanır. Gauss dağılımını kullanan olasılık denklemlerindeki varyansı belirleyen α parametreleri robota özgü değerlerdir. Bu değerler hareketi etkileyen gürültünün belirlenmesini sağlamaktadır ve belirsizliğin karakterini şu şekilde etkilemektedir:

- α_1, dönme hareketinin açı değişiminde oluşturduğu gürültüyü
- α_2, yer değiştirme hareketinin açı değişiminde oluşturduğu gürültüyü
- α_3, yer değiştirme hareketinin yer değişiminde oluşturduğu gürültüyü
- α_4, dönme hareketinin yer değişiminde oluşturduğu gürültüyü

temsil etmektedir.

Bu çalışmada parçacık filtresi temelli bir algoritma olan FastSLAM yöntemi kullanıldığı için odometri hareket modelinin de parçacık filtresine uygulanacak şekilde uyarlanmış hali kullanılmıştır. Parçacık filtresindeki örneklendirme bir diğer adıyla tahmin aşamasında kullanılanılan hareket modeli, odometri hareket örneklendirme modeli olarak adlandırılmıştır. Bu modelde, gauss dağılımı şeklinde bir sonsal durum hesaplanması yerine normal gauss dağılımına uygun olarak gelişigüzel örneklendirilmiş parçacıklar söz konusudur. Bundan dolayı, uygulanması daha kolaydır. Şekil 3.11'de aynı gürültü parametleri ile oluşturulmuş sonsal dağılım (a) ve parçacıklar (b) görülmektedir.



Şekil 3.11 : Aynı α parametreleri ile oluşturulmuş sonsal dağılım(a) ve örneklendirilmiş parçacıklar (b) [3].

Örneklendirme modeli; kontrol girişi olarak odometri okumalarını kullanması, hareketin bağıl değişimini hesaplaması ve hatayı gauss dağılımına göre modellemesi bakımından sonsal dağılımın hesaplandığı odometri hareket modeline benzemektedir. Çizelge 3.2'deki örneklendirme algoritmasında 2., 3. ve 4. satırda yönelim açısının değişiminin ve yer değiştirme denklemlerinin odometri hareket modelindekiyle aynı olduğu görülmektedir.

5., 6. ve 7. satırlarda, odometri okumaları baz alınarak hesaplanan değişimlere ortalama değeri sıfır olan ve varyansı a olan gauss dağılımına göre gürültü eklenmiştir. Bu gürültü değerleri denklem 3.8'de görülen fonksiyona göre gelişigüzel üretilir.

$$sample(a) = \frac{a}{6}\sum_{i=1}^{12} rand(-1,1) \tag{3.8}$$

Son olarak, gürültü eklenmiş bağıl hareket değişimleri bir önceki durum olan $x_{t-1} = (x\ y\ \theta)^T$'ye eklenerek son durum x_t bulunur. Gürültü ekleme ve örneklendirme işlemi parçacık filtresindeki her bir parçacığa ayrı ayrı uygulanır [3].

Çizelge 3.2 : Odometri hareket örneklendirme modeli algoritması

1:	odometri-hareket-örneklendirme-modeli(u_t, x_{t-1})
2:	$\delta_{rot1} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x},) - \bar{\theta}$
3:	$\delta_{trans} = \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2}$
4:	$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$
5:	$\hat{\delta}_{rot1} = \delta_{rot1} - sample(\alpha_1\delta_{rot1} + \alpha_2\delta_{trans})$
6:	$\hat{\delta}_{trans} = \delta_{trans} - sample(\alpha_3\delta_{trans} + \alpha_4(\delta_{rot1} + \delta_{rot2}))$
7:	$\hat{\delta}_{rot2} = \delta_{rot2} - sample(\alpha_1\delta_{rot2} + \alpha_2\delta_{trans})$
8:	$x' = x + \hat{\delta}_{trans}\cos(\theta + \hat{\delta}_{rot1})$
9:	$y' = x + \hat{\delta}_{trans}\sin(\theta + \hat{\delta}_{rot1})$
10:	$\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2})$
11:	$return\ x_t = (x', y', \theta')^T$

3.1.5.2 Hız modeli

Hız modelinde kontrol girişleri olarak doğrusal hız ve dönel hız uygulanır ve sırasıyla v ve w ile gösterilir. t anına ait kontrol girişi aşağıdaki denklem 3.9'daki gibi gösterilir.

$$u_t = \begin{pmatrix} v_t \\ w_t \end{pmatrix} \tag{3.9}$$

Doğrusal hızın pozitif bir değer olması ileriye doğru bir hareket komutu, dönel hızın pozitif bir değer olması da saat yönünün tersine doğru bir dönme komutu demektir. Odometri modelinde olduğu gibi hız modelinde de kontrol girişleri periyodik zaman aralıklarında uygulanır.

Çizelge 3.3'te $p(s_t|u_t, s_{t-1})$ olasılığının nasıl hesaplandığı gösterilmektedir. Bu algoritmaya uygulanmadan önce hız komutlarının herhangi bir gürültüsü olmadığı varsayılarak yeni konum ve yönelim açısı denklem 3.10'daki gibi hesaplanır.

$$\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} -\frac{v}{w}\sin\theta + \frac{v}{w}\sin(\theta + w\Delta t) \\ \frac{v}{w}\cos\theta - \frac{v}{w}\cos(\theta + w\Delta t) \\ w\Delta t \end{pmatrix} \tag{3.10}$$

Kontrol girişi olarak $t-1$ anından t anına kadar sabit bir şekilde hem doğrusal hem de dönel hız verildiği kabul edildiği için, robotun sanal bir çember üzerinde yol aldığı varsayılır. 2-5. satırlarda bu sanal çemberin merkezi(x^*, y^*) ve yarıçapı(r^*) hesaplanır.

x' ve y' gürültülü hareket sonucunda elde edilen yeni konumu belirtir. 6. satırda robotun bu gürültülü hareket sonucundaki yönelim açısının değişimi hesaplanır. 7-9. satırlarda gürültülü hızlar ve son yönelim açısı bulunur. Bütün bu hesaplamaların sonunda gerçek hız ve gürültülü hızların farklarını kullanarak sonsal durum olasılığı elde edilir.

Çizelge 3.3 : Hız hareket modeli algoritması

1:	hız-hareket-modeli(x_t, u_t, x_{t-1})
2:	$\mu = \frac{1}{2}\frac{(x-x')\cos\theta + (y-y')\sin\theta}{(y-y')\cos\theta - (x-x')\sin\theta}$
3:	$x^* = \frac{x+x'}{2} + \mu(y-y')$
4:	$y^* = \frac{y+y'}{2} + \mu(x-x')$
5:	$r^* = \sqrt{(x-x^*)^2 + (y-y^*)^2}$
6:	$\Delta\theta = atan2(y'-y^*, x'-x^*) - atan2(y-y^*, x-x^*)$
7:	$\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$
8:	$\hat{w} = \frac{\Delta\theta}{\Delta t}$
9:	$\gamma = \frac{\theta'-\theta}{\Delta t} - \hat{w}$
10:	$return\ prob(v-\hat{v}, \alpha_1\|v\| + \alpha_2\|w\|) \cdot (w-\hat{w}, \alpha_3\|v\| + \alpha_4\|w\|) \cdot (\gamma, \alpha_5\|v\| + \alpha_6\|w\|)$

Odometri modelinde olduğu gibi hız modeli de parçacık filtresine çizelge 3.4'teki gibi uyarlanabilir. Her parçacık için ayrı ayrı uygulanan aşağıdaki algoritmanın 2-4. satırlarında gürültülü hızlar ve son yönelim gürültüsü hesaplanır. Bu gürültülü değerlere göre 5-7. satırlarda da ilgili parçacığın yeni konumu ve yönelim açısı elde edilir [3].

Çizelge 3.4 : Hız hareket örneklendirme modeli algoritması

1:	hız-hareket-örneklendirme-modeli(u_t, x_{t-1})
2:	$\hat{v} = v + sample(\alpha_1\|v\| + \alpha_2\|w\|)$
3:	$\hat{w} = w + sample(\alpha_1\|v\| + \alpha_2\|w\|)$
4:	$\hat{\gamma} = sample(\alpha_1\|v\| + \alpha_2\|w\|)$
5:	$x' = x - \frac{\hat{v}}{\hat{w}}\sin\theta + \frac{\hat{v}}{\hat{w}}\sin(\theta + \hat{w}\Delta t)$
6:	$y' = y + \frac{\hat{v}}{\hat{w}}\cos\theta + \frac{\hat{v}}{\hat{w}}\cos(\theta + \hat{w}\Delta t)$
7:	$\theta' = \theta + \hat{w}\Delta t + \hat{\gamma}\Delta t$
8:	$return\ x_t = (x', y', \theta')^T$

3.1.6 Bayes filtresi

Otonom robotlarda eş zamanlı konumlandırma ve haritalama için birden fazla çözüm önerilmiştir. En yaygın kullanılan teknikler arasında Bayes kuralı uygulayan Kalman Filtresi, Genişletilmiş Kalman Filtresi ve Parçacık Filtresi bulunmaktadır.

EZKH uygulamalarında sıkça kullanılan kavramlardan birisi olan inanç (belief), robotun çevresi hakkındaki ve konumu hakkındaki öngörüsünü belirtir. Çizelge 3.5'te verilen Bayes Filtresi, bu inancın hesaplanmasında kullanılan en önemli algoritmalardan birisidir. Robot ölçümleri ve kontrol verileri kullanarak hesaplanır. Bu tablo Bayes filtresinin güncelleme kuralı olarak da bilinen tekrarlı yapısını göstermektedir.

Çizelge 3.5 : Bayes filtresi algoritması

1:	Bayes-filtresi(bel(x_{t-1}), u_t, z_t)
2:	for all x_t do
3:	$\overline{bel}(x_t) = \int p(x_t\|u_t, x_{t-1})bel(x_{t-1})dx$
4:	$bel(x_t) = \eta\, p(z_t\|x_t)\overline{bel}(x_t)$
5:	end for
6:	return $\overline{bel}(x_t)$

Bayes filtresi yinelemeli bir işlemdir. Yani t anındaki $bel(x_t)$, bir önceki inanç olan $t-1$ anındaki $bel(x_{t-1})$'den hesaplanır. Algoritma görüldüğü üzere iki aşamadan oluşmaktadır. Birinci aşamada x_{t-1}'den x_t'ye geçişi sağlayan u_t kontrol işaretinin

olasılığı ile $bel(x_{t-1})$'nin çarpımının integrali alınır. Bu aşama kontrol güncellemesi ya da tahmin olarak adlandırılır. İkinci aşamada ise kontrol güncellemesinde elde edilen $\overline{bel}(x_t)$ ile z_t ölçüm olasılığı çarpılır. Burada çarpım genellikle bir olasılığı ifade etmediği için sonuç bir düzeltme sabiti (η) ile çarpılır [3].

3.1.7 Kalman filtresi

İlk olarak 1960 yılında Rudolf Emil Kalman tarafından öne sürülen Kalman filtresi, Bayes kuralına dayanan olasılıksal bir tahmin algoritmasıdır. Kalman Filtresi EZKH için basitçe iki aşamada şu şekilde anlatılabilir: İlk olarak önceki durum ile kontrol girişlerinden elde edilen veriyi birleştirerek inanç için bir tahminde bulunur. İkinci aşamada, çevreyi algılayan sensörlerle bir ölçüm yapıp bu inancı güncelleyerek sonraki durum için bir sonuç üretir [3, 43].

Kalman filtresinin üç ana bileşeni vardır: Birincisi; filtrede hesaplanmak istenen konum, hız veya yönelim açısı gibi değişkenlerin olduğu durum vektörüdür. İki boyutlu bir uzayda konum ve yönelim bilgisi içeren bir durum vektörü denklem 3.11'deki gibi gösterilebilir.

$$v = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{3.11}$$

İkinci bileşen dinamik model olarak adlandırılır ve durum vektörünün zaman içindeki dönüşümünü tanımlar. Denklem 3.12'de doğrusal bir durum için bir dinamik model belirtilmiştir. Buradaki F dinamik matristir ve sabittir, ε de gürültüyü simgeler.

$$\dot{v}(t) = Fv(t) + \varepsilon(t) \tag{3.12}$$

Ölçüm modeli olarak bilinen üçüncü bileşen de durum ve hesaplamalar arasındaki ilişkiyi gösterir ve dinamik model denklemiyle benzer yapıdadır [43].

İki aşamalı Kalman filtresinin algoritması denklem 3.12'den denklem 3.16'ya kadar olan denklem grubundaki gibidir. Burada t anındaki inanç $bel(x_t)$, x_t ortalama değeri ve Σ_t kovaryansı ile belirtilir. Tahmin aşamasında herhangi bir ölçüm yapılmadan sadece $t-1$ anındaki durum x_{t-1} ile kontrol girişi u_t toplanıp önsel durum bulunur. A ve B matrisleri x_t ve u_t ile çarpılarak durum geçiş fonksiyonu doğrusallaştırılmış olur. Bu yüzden Kalman Filtresi doğrusal sistemler için uygulanabilirdir. Düzeltme aşamasında; birinci aşamada hesaplanmış olan önsel durum yeni ölçümler kullanılarak yeniden hesaplanır ve buna da sonsal durum denir. 3. adımdaki K_t Kalman kazancı,

düzeltme aşamasındaki kalman hesaplamalarının ne kadar kesin olduğunu anlatır. Yani K_t arttıkça düzeltme hesaplamalarının olasılıksal ağırlığı da artar. Tam tersine K_t azaldıkça da tahmin hesaplamalarının olasılıksal ağırlığı artar [3, 44].

$$\bar{x}_t = A_t x_{t-1} + B_t u_t \tag{3.12}$$

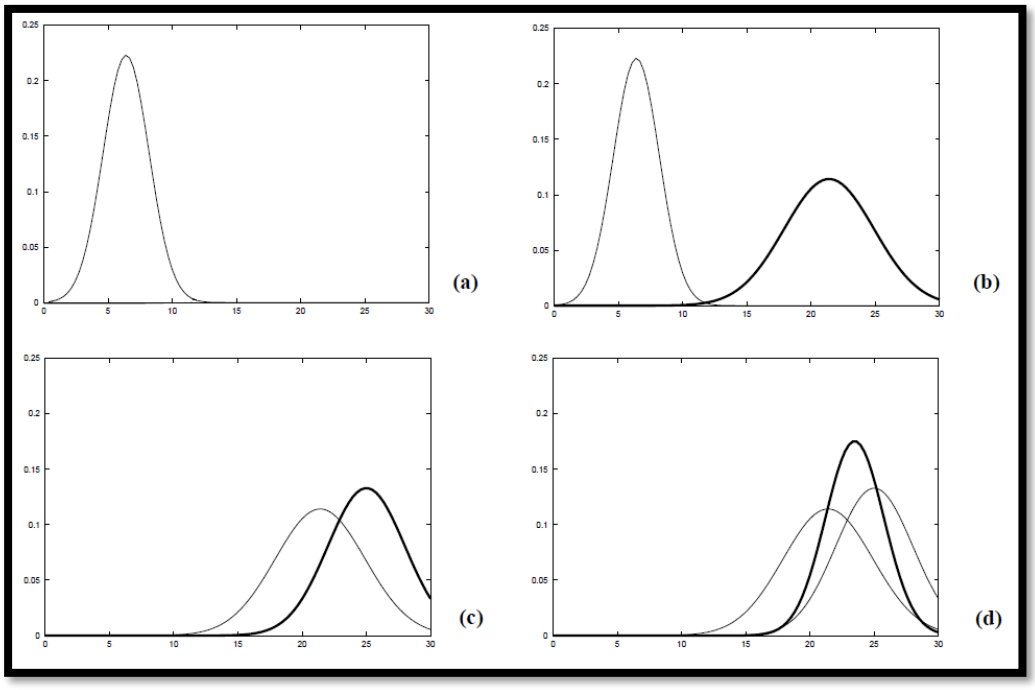
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \tag{3.13}$$

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \tag{3.14}$$

$$x_t = \bar{x}_t + \Sigma_t = (I - K_t H_t)\bar{\Sigma}_t \tag{3.15}$$

$$K_t\big(z_t - h(\bar{x}_t)\big) \tag{3.16}$$

Kalman algoritmasının Gauss dağılımı ile gösterimi şekil 3.12'deki gibidir. a'da x-1 anındaki durumun gauss gösterimi görülmektedir. Kontrol girişi uygullandıktan sonra yeni durumun tahmini şekil b'deki gibidir. Tahmine ait ortalama (mean) değer daha düşük ve kovaryansı daha fazladır. Çünkü belirsizlik önceki inanca göre daha fazladır. c'de görüldüğü üzere bir sonraki aşamada yeni ölçümler alınır. Son olarak d'de bu ölçüm ve tahmin kullanılarak t anına ait yeni bir inanç hesaplanır [3].



Şekil 3.12 : Kalman filtresi algoritmasının Gauss dağılımı ile gösterimi [3].

Kalman Filtresinin yaygın olarak kullanılmasının en önemli sebepleri fazla işlemsel yük gerektirmemesi ve kolay uygulanabilir olmasıdır. Bununla birlikte etkin bir hesaplama yeteneği de olmasına rağmen çoğu robot hareket modelinin doğrusal

olmaması Kalman filtresine seçenek olarak genişletilmiş Kalman filtresi ve parçacık filtresi gibi yöntemler geliştirilmesine sebep olmuştur [3, 45].

3.1.8 Genişletilmiş Kalman Filtresi

Doğrusal sistemler için Kalman filtresi çok kullanışlı olsa da gerçek hayatta robot uygulamalarında çoğu zaman doğrusal olmayan modellerle karşılaşıldığı için onun yerine temel yapısı Kalman filtresine çok benzer olan genişletilmiş Kalman filtresi (GKF) geliştirilmiştir. Aslında GKF, doğrusal olmayan sistemler için doğrudan hesaplama yapmak yerine bunu, sürekli türevlenebilir fonksiyonları kullanarak doğrusallaştırılmış formda yapar. Bu da Kalman filtresinde olduğu gibi gerçek bir inanç hesaplaması yerine benzetim yoluyla gerçek inanca yakın bir değer elde edilmesine sebep olur [3, 43].

GKF'de durum geçişi ve ölçümün, denklem 3.17 ve 3.18'de görüldüğü gibi doğrusal olmayan fonksiyonlarla hesaplandığı kabul edilir. Burada g fonksiyonu Kalman filtresindeki A ve B matrislerine karşılık gelirken h fonksiyonu da C matrisine karşılık gelmektedir.

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \tag{3.17}$$

$$z_t = h(x_t) + \delta_t \tag{3.18}$$

g ve h fonksiyonları ile hesaplanan inancın gauss dağılımı olarak ifade edilebilmesi için bu fonksiyonların doğrusallaştırılması gerekir. Bu da Taylor serisi açılımı kullanılarak yapılır. Denklem 3.19'dan 3.23'e kadar olan ifadeler GKF'nin algoritmasını oluşturmaktadır.

$$\bar{x}_t = g(u_t, x_{t-1}) \tag{3.19}$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t \tag{3.20}$$

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \tag{3.21}$$

$$x_t = \bar{x}_t + K_t\big(z_t - h(\bar{x}_t)\big) \tag{3.22}$$

$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t \tag{3.23}$$

Bu algoritmadan da anlaşılacağı üzere GKF ile Kalman filtresinin yapısı hemen hemen aynıdır. Tek fark, doğrusal olmayan g ve h fonksiyonlarının doğrusallaştırılmasıdır. Bu işlem g fonksiyonu ve onun kısmi türevi kullanılarak denklem 3.24'teki gibi yapılır.

$$g'(u_t, x_{t-1}) := \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}} \tag{3.24}$$

g fonksiyonun yaklaşık değeri denklem 3.25'teki gibi elde edilir.

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + g'(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1}) \tag{3.25}$$

Böylece;

$$g(u_t, x_{t-1}) = g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}) \tag{3.26}$$

elde edilir. h fonksiyonunun elde edilmesi de aynı işlem basamakları uygulanarak gerçekleşir ve 3.27'deki gibi gösterilir.

$$h(x_t) = h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t) \tag{3.27}$$

Jakobiyen matrisi olarak adlandırılan G_t ve H_t matrisleri Kalman filtresindeki A_t, B_t ve $C_t$matrislerine karşılık gelmektedir.

Doğrusal sistemlerde durum geçiş matrisi, dinamik matris ve gözlem matrisinin bir kere hesaplanması yeterliyken doğrusal olmayan sistemlerde bu matrisler her çevrimde yeniden hesaplanmalıdır. Bu prosedür, ek bir işlemsel yük getirmesine rağmen GKF, EZKH uygulamalarında Kalman filtresinden daha çok kullanım alanı bulmaktadır. Diğer yandan, GKF doğrusal olmayan sistemler için bir çözüm getirse de ortalama (μ) ve kovaryans (Σ)'ın kesinliği Kalman filtresindekine göre daha düşüktür [3, 43].

3.1.9 GKF'nin EZKH'ye uygulanması

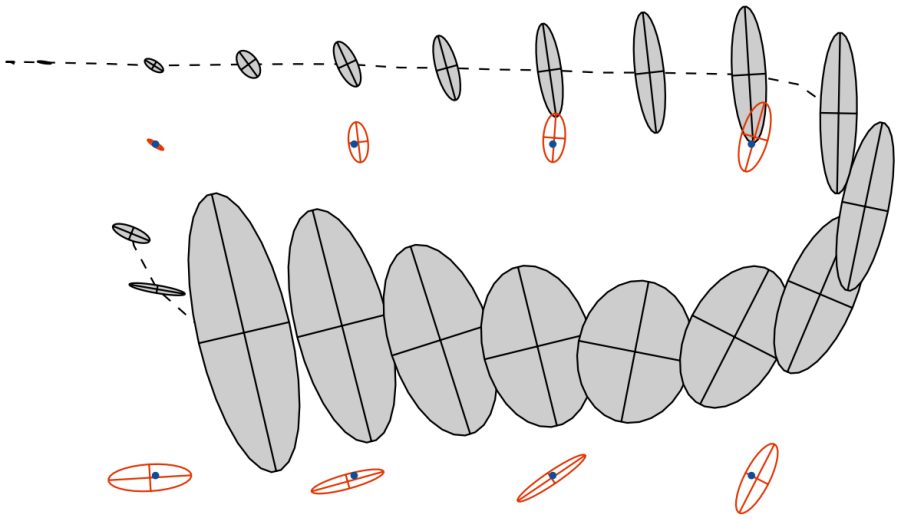
İlk EZKH uygulamalarından itibaren kullanılmaya başlayan GKF, gerek tek olarak gerekse birkaç farklı filtre ile birlikte, bugün birçok EZKH yönteminde hala etkin bir şekilde kullanılmaktadır. GKF'nin en başarılı kullanım şekillerinden birisi olan EKF-SLAM, işaretçi tabanlı bir yaklaşımdır. Haritayı oluşturan matris yapısından dolayı çok sayıda işaretçinin olduğu ortamlarda EKF-SLAM tercih edilmez, çünkü işlem hızı çok yavaşlar.

Diğer birçok EZKH'de olduğu gibi EKF-SLAM'de de ölçüm ve hareket gürültüsü Gauss Dağılımına göre hesaplanır. Monte Carlo konumlandırması ile büyük benzerlik gösteren bu yaklaşımın tek farkı, GKF kullanarak robot konumunu ve belirsizliğini hesaplamasının yanında işaretçilerin konumlarını ve belirsizliklerini de GKF ile

hesaplamasıdır. Bu durumda robot durumu (s_t) ve haritayı (m) içeren vektör, denklem 3.28'deki gibidir [3].

$$y_t = \begin{pmatrix} s_t \\ m \end{pmatrix} \tag{3.28}$$

Şekil 3.13'te EKF-SLAM'ın uygulandığı bir örnek görülmektedir. Robot, 8 tane işaretçinin olduğu bir ortamda dikdörtgen benzeri bir rotayı takip etmektedir. Başlangıçta konum belirsizliği sıfır olan robot ilerledikçe bu belirsizlik de artmaya başlamaktadır. Bu sırada tespit ettiği işaretçilerin de konumlarını ve belirsizliklerini EKF ile hesaplayıp haritasına eklemektedir. Mavi nokta ile gösterilenler işaretçilerin gerçek konumları olup, kırmızı elipslerin merkezi de işaretçilerin hesaplanan konumlarıdır. Robot, harekete başladığı konuma doğru yaklaştığında daha önce tespit etttiği işaretçilerden birini yeniden tespit eder ve bunun sonucunda hem işaretçinin konumunun belirsizliği hem de robotun konumunun belirsizliği azalır [9].



Şekil 3.13 : EKF-SLAM'de işaretçi ve robot ilişkisi [9].

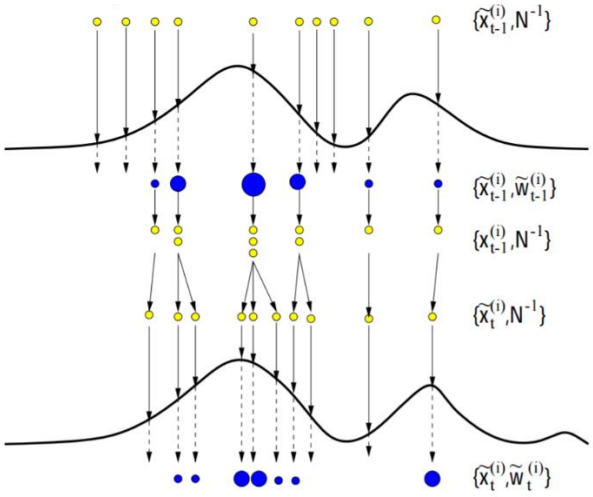
EKF-SLAM, öngörü ve düzeltme olmak üzere iki ana adımdan oluşur. Öngörüde ölçüm alınmadan sadece t anındaki kontrol girişleri ve $t-1$ anındaki robot durumu kullanılarak yeni konum için öngörüde bulunulur. Bu aşamada aynı zamanda robotun konum belirsizliğini belirten kovaryan matrisi de güncellenir ve belirsizlik artar. Bu adım aslında GKF algoritmasında belirtilen öngörü adımıyla aynıdır. Denklemdeki G_t, EKF-SLAM'de kullanılan hareket modelinin jakobiyan matirisidir.

Düzeltme aşaması, robot konumu ve işaretçi konumu belirsizliğin güncellendiği aşamadır. Bu iki unsur için de ortalama ve kovaryans güncellemesi yapıldığı düşünülürse aslında düzeltme aşaması iki kere uygulanıyor denilebilir. Ölçüm

alındığında ilk olarak tespit edilen işaretçi veri ilişkilendirmeye tabi tutularak bu işaretçinin kimliği belirlenir. Yani yeni bir işaretçi mi yoksa haritadaki bir işaretçi mi olduğu tespit edilir. Eğer yeni bir işaretçiyse o işaretçi için ortalama ve kovaryans hesaplanır. Eğer haritadaki bir işaretçiyse harita güncellenir. Düzeltme aşamasının son kısmında da ölçüme göre robot durumu ve belirsizliği güncellenir. Bu aşamada eğer tespit edilen işaretçi yeni bir işaretçiyse robotun belirsizliği artar. Eğer eski bir işaretçiyse belirsizlik azalır ve konum düzeltilir [3].

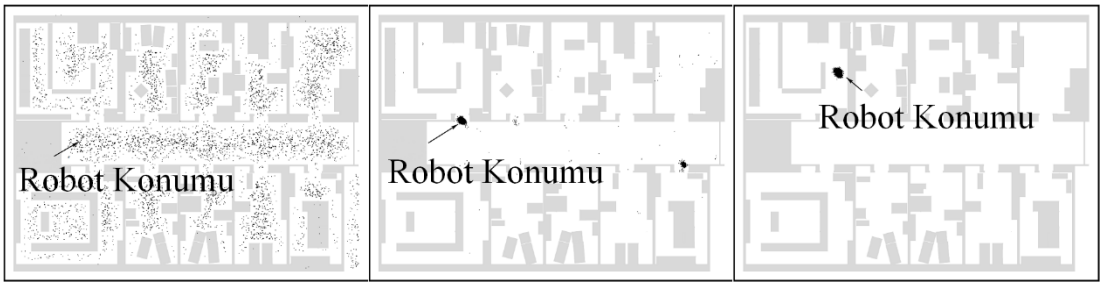
3.1.10 Parçacık filtresi

Parçacık filtresi; Bayes kuralına dayanan ve olasılık yoğunluk dağılımlarını birbirinden bağımsız parçacıklarla hesaplayan bir tahmin algoritmasıdır. Parçacık filtresi terimi ilk defa [46]'da kullanılmıştır. Ekonomi tahminleri [47], hedef izleme, hava trafik kontrolü, robot ve araç konumlandırması gibi çeşitli alanlarda kullanılan parçacık filtresi, doğrusal olmayan sistemlere kolayca uygulanabildiğinden Kalman filtresi ve türevlerine göre üstündür [48]. Bu filtrede ana fikir; sonsal durumdan gelişigüzel bir şekilde örneklendirilen parçacıkların, ölçümler doğrultusunda önem ağırlıklarının hesaplanarak bu ağırlıklar oranında çoğaltılması şeklindedir. Şekil 3.14, parçacık filtresinin nasıl işlediğine dair bir fikir vermektedir. Yukarıda sarı renkli olan parçacıklar için önem ağırlığı hesaplaması yapılmış ve Gauss şeklinde olmayan bir olasılık dağılım fonksiyonuna benzediği görselleştirilmiştir. Önem ağırlıkları oranında çoğaltılan parçacıklardan çok düşük olasılıklı olanlarının bu aşamada elendiği görülmektedir. Son olarak, çoğaltılarak üretilen yeni parçacıklar gelişigüzel olarak yeniden örneklendirilmiş ve yeni ölçüm alınarak önem ağırlıkları hesaplanmıştır.



Şekil 3.14 : Parçacık filtresi şeması.

Monte Carlo konumlandırmasında da kullanılan parçacık filtresinde parçacıklar ilk başta harita içerisinde gelişigüzel dağılmış bir şekilde bulunur. Robot ilerleyip sensör ölçümlerini aldıkça büyük olasılıklı parçacıkların daha fazla çoğalması ve düşük olasılıklı parçacıkların da yok olmasından dolayı harita içerisinde belli bölgelerde yoğunlaşmalar başlar [49]. Şekil 3.15'te, simetrik şekillerin olduğu bir kapalı alanda uygulanan Monte Carlo konumlandırmasına bir örnek verilmiştir.



Şekil 3.15 : Monte Carlo Konumlandırması: Başta parçacıklar dağılmış haldedir (solda), ölçüm alındıkça parçacıklar kümelenmeye başlar (ortada), yeterince ölçüm alındıktan sonra parçacıklar tek bir küme oluşturur (sağda) [49].

FastSLAM'de ise parçacık filtresi biraz daha farklı şekilde uygulanır. Başlangıçta hem konum hem de harita bilgisi olmadığı için bütün parçacıklar robot ile aynı konumda kabul edilirler. Robot ilerledikçe gelişigüzel olarak örneklendirilen parçacıklar dağılmaya başlar. Robot, sensör ölçümleri alındıkça oluşturulmaya başlanan haritada bilinen yerlerden geçildikçe dağılmış durumdaki parçacıklar da tekrar yoğunlaşmaya başlar .

Kalman filtresinde olduğu gibi parçacık filtresinde de durum hesaplaması, $t-1$ anındaki inancı kullanarak t anındaki inancın hesaplanmasıyla yapılır. Ancak parçacık filtresinde inanç gösterimi parçacıklar kümesi olarak belirtildiği için $bel(x_t)$ ile değil denklem 3.29'daki gibi gösterilir. Çünkü Kalman filtresinde durum, tek bir normal gauss dağılımı şeklinde ifade edilirken parçacık filtresinde ise M tane olasılık değerleri olarak belirtilir.

$$X_t := x_t^{[1]}, x_t^{[2]}, \dots, x_t^{[M]} \tag{3.29}$$

Çizelge 3.6'daki parçacık filtresi için verilen algoritmada tahmin, düzeltme ve yeniden örnekleme (resampling) olmak üzere 3 adım vardır. 4. satırda M sayıda parçacığın örneklenmesi için $t-1$ anına ait parçacıkların durumlarına, t anında uygulanan kontrol girişi eklenerek bir tahmin yapılır. Burada üretilen parçacık kümesi Bayes filtresi'ndeki önsel (prior) duruma karşılık gelmektedir. Sonsal (posterior) durum

hesaplaması ise 5. ve 6. satırda yapılmaktadır. Düzeltme aşaması olarak bilinen bu kısım aynı zamanda önem örneklemesi (importance sampling) olarak da anılmaktadır. 4. satırda üretilen her bir parçacık için z_t ölçümleri ile önem ağırlığı w_t hesaplanır. 6. satırda parçacıklar ve bu parçacıklara ait önem ağırlığı $\bar{X}_t$ parçacık kümesine eklenir [3].

Parçacık filtresi'nin en önemli kısmı olan yeniden örnekleme, 8. satırda başlar. Her bir parçacığının önem ağırlığı hesaplanmış olan $\bar{X}_t$ parçacık kümesinden, onunla aynı boyutta yeni bir parçacık kümesi üretilir. $\bar{X}_t$ içindeki önemi çok düşük olan örnekler yeni kümede yer almazlar ve parçacık sayısı azalmış olur. Bu durum bir süre sonra parçacıkların tükenmesine yol açabilir. Tekrar aynı sayıya ulaşabilmek için, sayıları $\bar{X}_t$ içindeki diğer parçacıkların önemi ile orantılı olacak şekilde yeni örnekler üretilir. Yani büyük ağırlığa sahip olan parçacıklardan daha fazla sayıda örnek üretilirken düşük ağırlıklı olanlardan daha az sayıda üretilir. Üretilen bu yeni örneklerin her birinin önem ağırlığı birbirine eşittir ve ağırlıkları toplamı 1'e eşittir [3, 50].

Çizelge 3.6 : Parçacık filtresi algoritması

1:	Parçacık-filtresi(X_{t-1}, u_t, z_t)
2:	$\bar{X}_t = X_t = \emptyset$
3:	for $m = 1$ to M do
4:	$sample\ x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5:	$w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6:	$\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:	end for
8:	for $m = 1$ to M do
9:	draw i with probability $\propto w_t^{[i]}$
10:	add $x_t^{[i]}$ to X_t
11:	end for
12:	return X_t

Doğrusal sistemlere kolayca uygulanabilirliğinin yanında parçacık filtresi, inanç olarak birden fazla seçenek sunabildiği için Kalman filtresinden bir adım daha öne geçmektedir. Bir dezavantaj olarak parçacık filtresinde uygulamada çok sayıda parçacık kullanılıyor olması ve her bir parçanın ayrı bir işlem gerektirmesi hesapsal yükün fazla olmasına neden olsa da günümüzde üretilen hızlı işlemcilerin kullanımıyla bu durumun üstesinden gelmek mümkündür.

3.2 FastSLAM

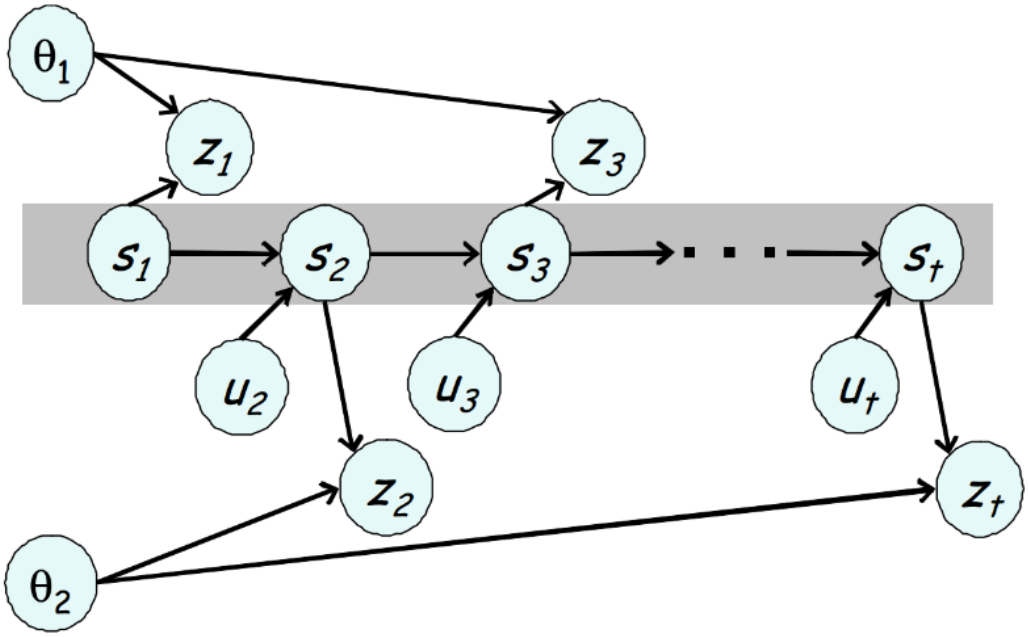
Birinci bölümde bahsedildiği gibi EZKH uygulamaları için önerilen EKF-SLAM'de işaretçi sayısı arttıkça işlem yükü üstel bir şekilde artmaktadır. Bu sorun EKF-SLAM'in harita oluşturma yönteminden kaynaklanmaktadır. Aşağıdaki denklem 3.30 ve 3.31 işaretçi haritasının nasıl oluşturulduğunu göstermektedir. Denklem 3.30'da μ_{s_t}, robotun t anındaki durumunu, $n = 1, \dots, N$ olmak üzere $\mu_{\theta_n,t}$ n'inci işaretçinin konumunu belirtir. Bu vektör her bir yeni işaretçi için doğrusal olarak büyümektedir. Denklem 3.31'de $\Sigma_{s_t,t}$, robotun t anındaki varyansını yani belirsizliğini, $\Sigma_{\theta_n,t}$ n'inci işaretçinin varyansını ve $m = 1, \dots, M$ olmak üzere $\Sigma_{\theta_n\theta_m,t}$ n'inci ve m'inci işaretçinin kovaryansını belirtir. Bu matris her bir yeni işaretçi için üstel bir şekilde büyümektedir [51].

$$\mu_t = \{\mu_{s_t}, \mu_{\theta_1,t}, \dots, \mu_{\theta_N,t}\} \tag{3.30}$$

$$\Sigma_t = \begin{bmatrix} \Sigma_{s_t,t} & \Sigma_{s_t\theta_1,t} & \cdots & \Sigma_{s_t\theta_N,t} \\ \Sigma_{s_t,t} & \Sigma_{\theta_1,t} & \Sigma_{\theta_1\theta_2,t} & \\ \vdots & \Sigma_{\theta_2\theta_1,t} & \ddots & \\ \Sigma_{s_t,t} & & & \Sigma_{\theta_N,t} \end{bmatrix} \tag{3.31}$$

FastSLAM, GKF ve parçacık filtresini bir arada kullanarak EKF-SLAM'deki bu üstel büyüme sorununa çözüm getirmektedir. Öyle ki; işaretçiler GKF'de olduğu gibi birbiriyle ilişkilendirilmek yerine parçacık filtresindeki her bir parçacığın konumu ile ilişkilendirilmektedir. Çünkü robotun gittiği yol tam olarak bilinirse işaretçileri birbirinden bağımsız bir şekilde hesaplamak mümkündür. Şekil 3.16'da dinamik Bayes ağı gösterimi ile işaretçilerin nasıl birbirinden bağımsız olabileceği belirtilmiştir. s_t, z_t ve u_t t anında sırasıyla robotun durumunu, sensör ölçümünü ve kontrol girişini simgelerken θ_{n_t} de n'inci işaretçiyi belirtir. Robot; 1. işaretçiyi $t = 1$ ve $t = 3$ anlarında, 2. işaretçiyi de $t = 2$ anında tespit etmektedir. s_1 konumundayken ilk defa tespit ettiği 1. işaretçinin konumunu s_3 anında yeniden tespit ettiğinde, o ana kadar izlediği yolu bildiği için 2. işaretçiden bağımsız olarak hesaplayabilmektedir.

Denklem 3.32, FastSLAM'deki her bir parçacık için durum bilgisinin ve haritanın nasıl oluşturulduğunu göstermektedir. M parçacık sayısını belirtmek üzere ve $m = 1, \dots, M$ iken $S_t^{[m]}$ m'inci parçacığın durum bilgisini ve işaretçi haritasının bilgisini barındıran vektörü ifade eder.



Şekil 3.16 : EZKH'nin dinamik Bayes ağı ile gösterimi [49].

Bu vektörde $s^{t,[m]}$, m'inci parçacığın başlangıçtan t anına kadar olan durumlarını yani izlediği yolu; $\mu_{\mathrm{n},t}^{[m]}$ ve $\Sigma_{\mathrm{n},t}^{[m]}$ sırasıyla n'inci işaretçinin konumunu ve belirsizliğini (varyansını) belirtir. Denklem 3.32'den anlaşılacağı üzere haritaya eklenen her bir yeni işaretçi, toplam eleman sayısını doğrusal bir şekilde artırır. Bu durumda işlem yoğunluğu $(2N+1) * M$ olarak ifade edilebilir [49].

$$S_t^{[m]} = \langle s^{t,[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \dots, \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]} \rangle \tag{3.32}$$

3. bölümde de anlatıldığı üzere EZKH yöntemlerinin büyük bir kısmı sonsal hesaplamasını, harita bilgisi ve robotun son andaki durum bilgisi üzerinden yaparken FastSLAM bu hesaplamayı harita bilgisi ve robotun başlangıçtan t anına kadar olan bütün durumları üzerinden yapar. Dinamik bayes ağı örneği ile açıklanan koşullu bağımsızlıktan yararlanarak EZKH sonsalı çarpanlara ayrılmış bir şekilde aşağıdaki denklemdeki gibi ifade edilebilir [49].

$$p(s^t, \Theta \mid z^t, u^t, n^t) = p(s^t \mid z^t, u^t, n^t) \prod_{n=1}^{N} p(\theta_n \mid s^t, z^t, u^t, n^t) \tag{3.33}$$

3.2.1 FastSLAM İşlem Basamakları

FastSLAM algoritması dört adımdan oluşmaktadır:

- Kontrol girişleri uygulayarak bir önceki parçacık kümesinden yeni bir küme örneklendirme

- Tespit edilen işaretçinin her bir parçacık için güncelleme
- Sensör ölçümüne göre her parçacık için önem ağırlığı hesaplama
- Parçacıkların önem ağırlıkları kullanılarak yeni bir parçacık kümesi oluşturma

3.2.1.1 Yeni konum örnekleme

FastSLAM'de de diğer Kalman tabanlı EZKH yöntemlerinde olduğu gibi tahmin ve güncellleme olarak iki temel aşama vardır. Tahmin aşamasında $t-1$ anındaki parçacık kümesinden sadece kontrol girişleri uygulayarak yeni bir parçacık kümesi oluşturulur. Öneri dağılımı (proposal distribution) adı verilen bu yeni küme olasılıksal bir hareket modelinin her bir parçacığa tek tek uygulanmasıyla elde edilir. Bir parçacık için yeni bir konum elde edilmesi denklem 3.34'teki gibi gösterilir.

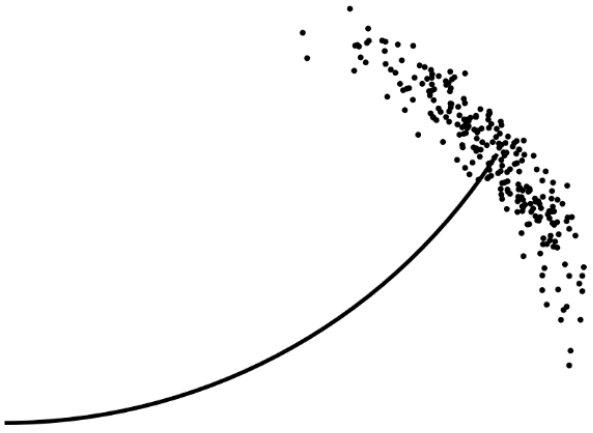
$$s_t^{[m]} \sim p(s_t \mid u_t, s_{t-1}^{[m]}) \tag{3.34}$$

Bütün parçacıklar örneklendirildikten sonra elde edilen öneri dağılımı ise denklem 3.35'teki gibi ifade edilir.

$$p(s^t \mid z^{t-1}, u^t, n^{t-1}) \tag{3.35}$$

Bu aşamadaki işlem zamanı parçacık sayısıyla doğru orantılıyken işaretçi haritasının büyüklüğü bu işlem zamanına herhangi bir etkide bulunmaz.

Şekil 3.17'de, 250 tane parçacık için önceki bölümde anlatılan odometri hareket modeline göre örneklendirilen parçacıkların dağılımı görülmektedir. Yay şeklindeki çizgi kontrol girişleri uygulanarak oluşan gerçek yolu göstermektedir. Her bir parçacığa normal Gauss gürültüsünün gelişigüzel bir şekilde uygulanmasıyla parametrik olmayan bir Gauss dağılımı şeklinde bir parçacık kümesi oluşmuştur [49].



Şekil 3.17 : Olasılıksal hareket modeli ile örneklendirilmiş parçacıklar [49].

3.2.1.2 İşaretçilerin güncellenmesi

Yeni konum örnekleme aşamasında sonsal için bir öneri dağılımı elde edilmesi FastSLAM'in tahmin kısmı iken bundan sonraki adımlar güncelleme kısmıdır. Aslında bu aşamaların gerçekleşebilmesi için sensör tarafından bir işaretçinin algılanması gerekmektedir. Yoksa, sensör herhangi bir işaretçi algılayana kadar yeni konum örnekleme adımı tekrar edilir.

İşaretçilerin hesaplanması birbirlerinden bağımsız olarak robotun izlediği yola göre yapıldığından, parçacık kümesindeki her bir parçacık için N tane GKF vardır. Tespit edilen işaretçinin haritadaki mevcut bir işaretçi mi yoksa ilk defa görülen bir işaretçi mi olduğunun belirlenmesi için bu işaretçi ilk olarak veri ilişkilendirmeye tabi tutulur. Veri ilişkilendirmenin nasıl yapıldığı bu bölümün sonunda anlatıldığı için şimdilik, tespit edilen işaretçinin haritadaki hangi işaretçi olduğunun bilindiği varsayılmaktadır.

t anında tespit edilen bir işaretçi n'inci işaretçi (θ_{n_t}) değilse, θ_{n_t}'ye ait GKF değiştirilmez ve denklem 3.36'daki gibi gösterilir.

$$p(\theta_{n\neq n_t} \mid s^t, z^t, u^t, n^t) = p(\theta_{n\neq n_t} \mid s^{t-1}, z^{t-1}, u^{t-1}, n^{t-1}) \tag{3.36}$$

Eğer tespit edilen işaretçi n'inci işaretçi ise θ_{n_t}'ye ait sonsal, Bayes ve Markov kuralı kullanılarak denklem 3.37 ve 3.38'de görüldüğü gibi güncellenir. Bayes kuralına göre;

$$p(\theta_{n_t} \mid s^t, z^t, u^t, n^t) = \eta\, p(z_t \mid \theta_{n_t}, s^t, z^{t-1}, u^t, n^t)\, p(\theta_{n_t} \mid s^t, z^{t-1}, u^t, n^t) \tag{3.37}$$

elde edilir ve sonra Markov özelliği ile denklem 3.38'deki gibi sadeleştirilir. t anındaki ölçüm z_t sadece θ_{n_t}, s_t ve n_t'ye bağlıdır. Benzer şekilde θ_{n_t} de; s^t, u^t ve n^t'den bağımsızdır.

$$p(\theta_{n_t} \mid s^t, z^t, u^t, n^t) = \eta\, p(z_t \mid \theta_{n_t}, s_t, n_t)\, p(\theta_{n_t} \mid s^{t-1}, z^{t-1}, u^{t-1}, n^{t-1}) \tag{3.38}$$

GKF kullanan diğer EZKH yöntemlerinde olduğu gibi FastSLAM de ölçüm modeli için doğrusal Gauss yaklaşımını uygular. Buna göre; doğrusal olmayan ölçüm modeli $g(s_t, \theta_{n_t})$, birinci dereceden bir Taylor açılımı ile doğrusal bir model hale getirilir. İşaretçi hesaplaması robotun gittiği yola göre koşullandırıldığı için Taylor açılımı işaretçilerin konumları üzerinden yapılır. Açılım; 3.40, 3.41 ve 3.42'de görüldüğü gibi yapılır.

$$\hat{z}_t = g\left(s_t^{[m]}, \mu_{n_t,t-1}\right) \tag{3.40}$$

$$G_{\theta_{n_t}} = \nabla_{\theta_{n_t}} g(s_t, \theta_{n_t})|_{s_t = s_t^{[m]}; \theta_{n_t} = \mu_{n_t,t-1}^{[m]}} \tag{3.41}$$

$$g(s_t, \theta_{n_t}) \approx \hat{z}_t + G_\theta(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}) \tag{3.42}$$

Bu koşullar altında landmark güncelleme denklemindeki çarpımın birinci terimi Gauss dağılımı olarak aşağıdaki denklem 3.43'teki gibi gösterilir. Burada R_t, ölçüm modelinin gürültüsünü belirten kovaryans matrisidir.

$$p(z_t \mid \theta_{n_t}, s_t, n_t) \sim \mathcal{N}(z_t; \hat{z}_t + G_\theta \left(\theta_{n_t} - \mu_{n_t,t-1}^{[m]}\right), R_t) \tag{3.43}$$

Güncelleme denklemindeki çarpımın ikinci terimi de yine Gauss dağılımı olarak denklem 3.44'teki gibi ifade edilir.

$$p(\theta_{n_t} \mid s^{t-1}, z^{t-1}, u^{t-1}, n^{t-1}) \sim \mathcal{N}(\theta_{n_t}; \mu_{n_t,t-1}^{[m]}, \Sigma_{n_t,t-1}^{[m]}) \tag{3.44}$$

İşaretçinin ortalama ve kovaryansı, 3.45'ten 3.50'ye kadar olan geleneksel GKF güncelleme denklemleriyle elde edilir.

$$\hat{z}_t = g\left(s_t^{[m]}, \mu_{n_t,t-1}\right) \tag{3.45}$$

$$G_{\theta_{n_t}} = \nabla_{\theta_{n_t}} g(s_t, \theta_{n_t})|_{s_t = s_t^{[m]}; \theta_{n_t} = \mu_{n_t,t-1}^{[m]}} \tag{3.46}$$

$$Z_{n,t} = G_{\theta_{n_t}} \Sigma_{n_t,t-1}^{[m]} G_{\theta_{n_t}}^T + R_t \tag{3.47}$$

$$K_t = \Sigma_{n_t,t-1}^{[m]} G_{\theta_{n_t}}^T Z_{n,t}^{-1} \tag{3.48}$$

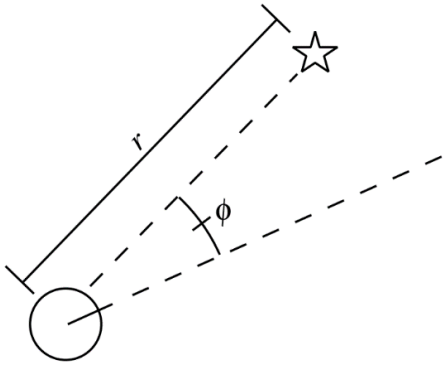
$$\mu_{n_t,t}^{[m]} = \mu_{n_t,t-1}^{[m]} + K_t(z_t - \hat{z}_t) \tag{3.49}$$

$$\Sigma_{n_t,t}^{[m]} = (I - K_t G_{\theta_{n_t}}) \Sigma_{n_t,t-1}^{[m]} \tag{3.50}$$

Düzlemsel bir alanda uygulanan EZKH yöntemlerindeki çoğu ölçüm modeli, şekil 3.18'de görüldüğü gibi tespit edilen işaretçinin robota olan uzaklığını ve robota göre olan açısını hesaplar.

Robotun t anındaki durumunun $\langle s_{t,x}, s_{t,y}, s_{t,\theta} \rangle$ ve işaretçinin konumunun $\langle \theta_{n_t,x}, \theta_{n_t,y} \rangle$ olarak ifade edildiğini varsayarak ölçüm fonksiyonu $g(s_t, \theta_{n_t})$ 3.51'deki gibi yazılır.

$$g(s_t, \theta_{n_t}) = \begin{bmatrix} r(s_t, \theta_{n_t}) \\ \phi(s_t, \theta_{n_t}) \end{bmatrix} = \begin{bmatrix} \sqrt{(\theta_{n_t,x} - s_{t,x})^2 + (\theta_{n_t,y} - s_{t,y})^2} \\ \tan^{-1}\left(\frac{\theta_{n_t,y} - s_{t,y}}{\theta_{n_t,x} - s_{t,x}}\right) - s_{t,\theta} \end{bmatrix} \tag{3.51}$$



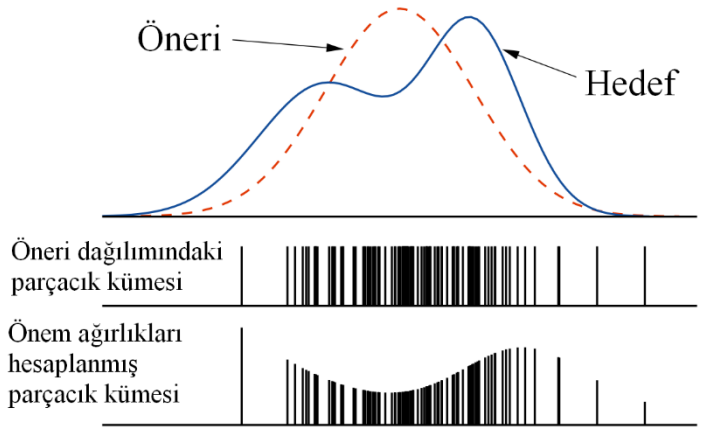
Şekil 3.18 : Robotun aldığı ölçümün açısı ve uzaklığı.

Bu ölçüm modeline göre Jakobiyan $G_{\theta_{n_t}}$ de denklem 3.52'deki gibi bulunur [49].

$$G_{\theta_{n_t}} = \begin{bmatrix} \frac{\theta_{n_t,x} - s_{t,x}}{\sqrt{(\theta_{n_t,x} - s_{t,x})^2 + (\theta_{n_t,y} - s_{t,y})^2}} & \frac{\theta_{n_t,y} - s_{t,y}}{\sqrt{(\theta_{n_t,x} - s_{t,x})^2 + (\theta_{n_t,y} - s_{t,y})^2}} \\ -\frac{\theta_{n_t,y} - s_{t,y}}{(\theta_{n_t,x} - s_{t,x})^2 + (\theta_{n_t,y} - s_{t,y})^2} & \frac{\theta_{n_t,x} - s_{t,x}}{(\theta_{n_t,x} - s_{t,x})^2 + (\theta_{n_t,y} - s_{t,y})^2} \end{bmatrix} \quad (3.52)$$

3.2.1.3 Önem ağırlıklarının hesaplanması

Öneri dağılımındaki parçacık kümesi t anındaki ölçümü ve veri ilişkilendirmeyi değil sadece kontrol girişini kullandığı için istenilen sonsal $p(s^t, \Theta \mid z^t, u^t, n^t)$ ile eşleşmez. Bu farkı gidermek üzere yapılan önem örneklendirmesi için şekil 3.19'da bir örnek verilmiştir. Parçacıklar doğrudan hedef dağılımı kullanılarak örneklendirilmek yerine öneri dağılımına göre örneklendirilirler. Hedef dağılımının öneri dağılımından daha büyük olduğu yerlerdeki parçacıkların ağırlıkları diğerlerinden daha fazladır. Öneri dağılımı hedef dağılımının altında kaldıkça parçacıkların da önem ağırlıklar o oranda azalır.



Şekil 3.19 : Önem ağırlıklarının hesaplanmasının bir örneği [39].

Her bir parçacığın önem ağırlığı, denklem 3.53'te olduğu gibi hedef dağılımın öneri dağılımına oranlanmasıyla elde edilir.

$$w_t^{[m]} = \frac{hedef\ dağılımı}{öneri\ dağılımı} = \frac{p(s^{t,[m]}|z^t, u^t, n^t)}{p(s^{t,[m]}|z^{t-1}, u^t, n^{t-1})} \tag{3.53}$$

Denklem 3.53'teki hedef dağılımı ifadesi Bayes kuralı uygulanarak aşağıdaki 3.54'teki gibi genişletilebilir.

$$w_t^{[m]} \propto \frac{p(z^t|s^{t,[m]}, z^{t-1}, u^t, n^t)p(s^{t,[m]}|z^{t-1}, u^t, n^t)}{p(s^{t,[m]}|z^{t-1}, u^t, n^{t-1})} \tag{3.54}$$

Denklem 3.54'ün pay kısmındaki ikinci terim t anındaki ölçümü kullanmadığı için t anındaki veri ilişkilendirmeye de gereksinim yoktur. Markov özelliği kullanılarak 3.55 elde edilir.

$$w_t^{[m]} = \frac{p(z^t|s^{t,[m]}, z^{t-1}, u^t, n^t)p(s^{t,[m]}|z^{t-1}, u^t, n^{t-1})}{p(s^{t,[m]}|z^{t-1}, u^t, n^{t-1})} \tag{3.55}$$

Böylece önem ağırlığı denklem 3.56'daki gibi ifade edilir.

$$w_t^{[m]} = p(z^t|s^{t,[m]}, z^{t-1}, u^t, n^t) \tag{3.56}$$

Önem ağırlığını hesaplamak için gerçek sensör ölçümü ile tahmin edilen ölçüm arasındaki fark kullanılır. Denklem 3.57, bir parçacığın önem ağırlığının nasıl hesaplandığını bu ölçüm farkları cinsinden göstermektedir. $Z_{n_t,t}$, GKF'deki yenilenme matrisine karşılık gelmektedir [49].

$$w_t^{[m]} = \frac{1}{\sqrt{|2\pi Z_{n_t,t}|}} \exp\{-\frac{1}{2}\left(z_t - \hat{z}_{n_t,t}\right)^T \left[Z_{n_t,t}\right]^{-1} (z_t - \hat{z}_{n_t,t})\} \tag{3.57}$$

3.2.1.4 Yeniden Örneklendirme

Bütün parçacıklar için önem ağırlığı hesaplaması yapıldıktan sonra mevcut parçacık kümesinden yeni bir parçacık kümesi elde edilir. Çoğu yeniden örnekleme yönteminde belli bir eşik değerinin altındaki parçacıklar yok edilirken diğer parçacıklar ise önem ağırlıklarıyla doğru orantılı olarak çoğaltılır. Literatürde çeşitli yeniden örnekleme algoritmaları bulunmaktadır. En çok kullanılan yöntemler; çokterimli (multinomial), katmanlı (stratified), sistematik (systematic) ve kalıntı (residual) yeniden örnekleme yöntemleridir.

Parçacıklara yeniden örnekleme uygulanmadan önce bütün parçacıkların önem ağırlıklarının normalize edilmesi gerekmektedir. Diğer bir deyişle, parçacık kümesindeki önem ağırlıklarının toplamının 1'e eşit olabilmesi için her bir parçacığın önem ağırlığının yeniden ölçeklendirilmesi gerekir. Bir parçacık kümesinde M tane parçacık olduğu varsayılarak normalizasyon denklem 3.58'deki gibi yapılır.

$$w_i = \frac{w_i}{\sum_{i=1}^{M} w_i} \tag{3.58}$$

Bundan sonra yeniden örnekleme işlemi önceden belirlenen eşik değerine göre yapılır.

FastSLAM'in en önemli aşamalarından biri olan yeniden örnekleme aşamasında yaşanan en büyük sıkıntı parçacık bozulması (degeneracy) veya yoksullaşması (impoverishment) adı verilen durumdur. Eğer, kontrol komutları uygulandıktan sonra elde edilen parçacık kümesindeki parçacıkların çoğu sensör ölçümü ile uyumlu ise yeniden örnekleme yapılırken büyük önem ağırlığına sahip yani kaliteli parçacıklar yok edilebilir. Bu da sonsal dağılımın hatalı oluşması demektir [49].

Parçacık bozulmasını ortadan kaldırmak için yeniden örneklemenin her zaman değil, kaliteli parçacık sayısının az olduğu zamanlarda yapılması gerekir. Bu sayıyı tespit edebilmek için [52]'de Efektif Örnek Sayısı (Effective Sample Size) hesaplaması önerilmiştir. Denklem 3.59'da bu hesaplama görülmektedir.

$$N_{eff} = \frac{1}{\sum_{i=1}^{M} {w_i}^2} \tag{3.59}$$

Efektif örnek sayısı belli bir değerin altında olduğu durumlarda yeniden örnekleme yapılır. Parçacık kümesindeki parçacıkları bir sonraki döngüye hazır hale getirmek için bütün parçacıkların önem ağırlıkları denklem 3.60 kullanılarak eşit hale getirilir [53].

$$w_i = \frac{1}{M} \tag{3.60}$$

3.2.2 FastSLAM'deki diğer kavramlar

3.2.2.1 Veri ilişkilendirme (en çok benzerlik yöntemi)

Gerçek ortam uygulamalarında ölçümle tespit edilen işaretçinin haritadaki işaretçilerle olan ilgisi kesin bir şekilde bilinmez. Bu da EZKH yöntemleri için belli başlı zorluklardan birisidir. Çözüm olarak; tespit edilen işaretçinin haritadaki diğer işaretçiler içindeki benzerliği kontrol edilir ve en çok benzerliği olan işaretçi ile

ilişkilendirilir. Denklem 3.61'deki $p(z^t|n_t, \hat{n}^{t-1}s^t, z^{t-1}, u^t)$ ifadesi en çok benzerlik hesaplayıcısının bir örneğidir.

$$\hat{n}_t = \underset{n_t}{\operatorname{argmax}}\, p(z^t|n_t, \hat{n}^{t-1}s^t, z^{t-1}, u^t) \tag{3.61}$$

FastSLAM'de ölçüm alındıktan sonra veri ilişkilendirme için denklem 3.62'den 3.65'e kadar olan işlemler sırasıyla uygulanır. İlk olarak ölçüm modelinin doğrusallaştırılıp Jakobiyan matrisi haline getirilmesi gerekir. m'inci parçacık için ölçüm tahmini yapılır. Jakobiyan matrisi ve n'inci işaretçinin kovaryansı kullanılarak inovasyon matrisi oluşturulur. Son olarak benzerlik hesaplaması yapılır. En büyük benzerliği olan işaretçi için hesaplanan değer aynı zamanda $m'inci$ parçacık için önem ağırlığı olarak atanır. Bu prosedür bütün parçacıkları için tekrar edilir.

$$G_{\theta_{n_t}} = \nabla_{\theta_{n_t}} g(s_t, \theta_{n_t})|_{s_t = s_t^{[m]}; \theta_{n_t} = \mu_{n_t,t-1}^{[m]}} \tag{3.62}$$

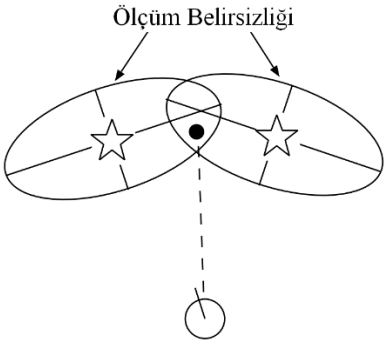
$$\hat{z}_t = g\left(s_t^{[m]}, \mu_{n_t,t-1}\right) \tag{3.63}$$

$$Z_{n,t} = G_{\theta_{n_t}} \Sigma_{n_t,t-1}^{[m]} G_{\theta_{n_t}}^T + R_t \tag{3.64}$$

$$w_t^{[m]} = \frac{1}{\sqrt{|2\pi Z_{n_t,t}|}} \exp\{-\frac{1}{2}\left(z_t - \hat{z}_{n_t,t}\right)^T \left[Z_{n_t,t}\right]^{-1} (z_t - \hat{z}_{n_t,t})\} \tag{3.65}$$

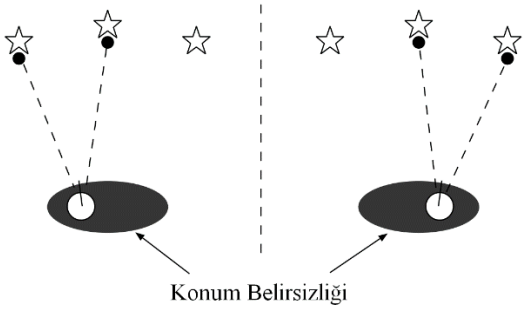
GKF tabanlı EZKH uygulamalarında genellikle tekil veri ilişkilendirme tercih edilir. Bu yöntemler hata yapmaya çok müsaittir ve veri ilişkilendirmedeki bir hata harita hesaplanmasında çok büyük hataların oluşmasına sebep olabilir.

Veri ilişkilendirmedeki hatalar birkaç farklı durumdan dolay ortaya çıkabilir. EZKH'de ölçüm gürültüsü ve hareket gürültüsü olmak üzere, belirsizliğe sebep olan iki temel etken vardır. Ölçüm belirsizliğinin artması haritadaki işaretçilerin belirsizliğinin artmasına sebep olur. Eğer bu belirsizlik çok fazla olursa birbirine yakın olan iki işaretçinin belirsizlikleri çakışabilir. Şekil 3.20'de ölçüm belirsizliği birbiriyle çakışan iki işaretçinin veri ilişkilendirme için oluşturabileceği sorun görselleştirilmiştir. Böyle bir durumda robot hangi işaretçinin, tespit edilen işaretçi olduğuna karar verirken hataya düşebilir. Eğer birden fazla ölçüm kullanılarak veri ilişkilendirilmesi yapılırsa ölçüm belirsizliklerinin çakışmasından kaynaklı bu sorun da çözülebilir.



Şekil 3.20 : EZKH'de ölçüm belirsizliği [49].

Hareket gürültüsü de veri ilişkilendirme açısından sorun oluşturabilecek bir durumdur. Çok büyük hareket gürültüsü söz konusu olduğunda parçacıkların dağılımı da bir o kadar büyüyecek ve konum belirsizliği artacaktır. Şekil 3.21'de konum belirsizliğinden kaynaklanabilecek veri ilişkilendirme karmaşası örneklendirilmiştir. Sensörün algıladığı iki işaretçi birbirlerine olan konumları bakımından, başka iki işaretçi ile benzerlik gösterebilir. Farklı konumlarda ve yönelim açısındaki parçacıklar benzer işaretçi çiftlerini yanlış tespit edebilir [49].



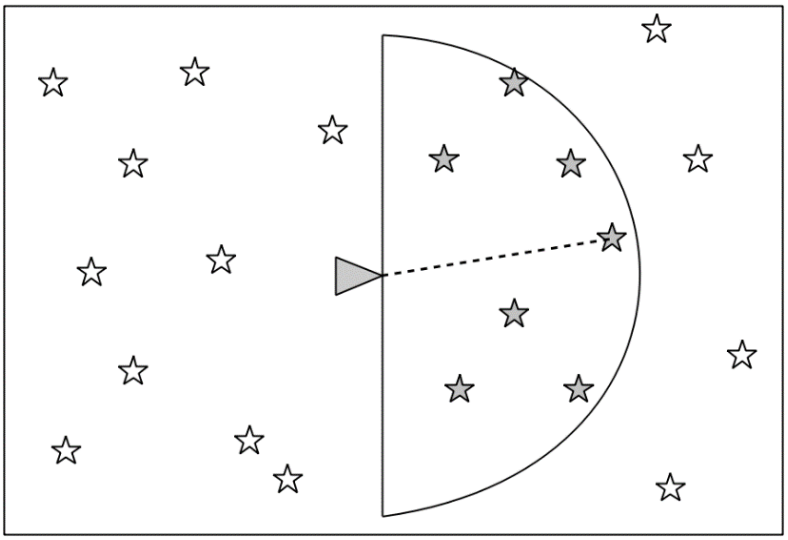
Şekil 3.21 : EZKH'de konum belirsizliği [49].

4. GELİŞTİRİLEN YÖNTEM

4.1 Giriş

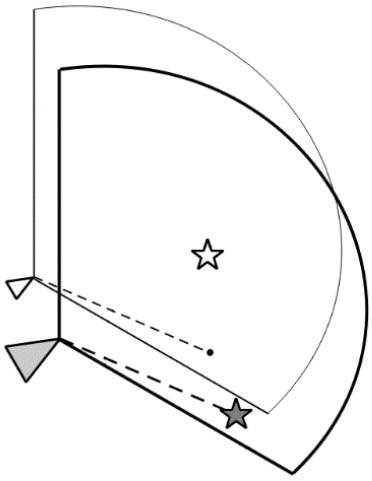
EZKH için önerilen yöntemde, işaretçi tabanlı algoritmaların veri ilişkilendirme aşaması için geliştirilen yeni bir yaklaşımla hız performansının artırılması amaçlanmıştır. Özellikle parçacık filtresi tabanlı EZKH yöntemlerinde işaretçi sayısının devasa boyutlara ulaşmasıyla birlikte veri ilişkilendirme işlemi çok büyük ölçüde zaman kaybına neden olmaktadır. Çünkü sensör ölçümüyle tespit edilen işaretçi, haritadaki bütün işaretçilerle karşılaştırılarak o işaretçinin kimliği belirlenmeye çalışılmaktadır. Bu tez çalışmasında kullanılan FastSLAM'de veri ilişkilendirme adımında her bir parçacık ve işaretçi için bölüm 3'teki denklem grubunun tekrar tekrar hesaplanması gerekmektedir. Bu da bütün algoritma için büyük bir işlem zamanının oluşması demektir. İşaretçi kimliğinin belirlenmesi için haritanın tamamının incelenmesi yerine, aşağıda ayrıntılı olarak anlatılan bazı ölçütleri sağlayan küçük bir alandaki işaretçilerin incelenmesi, gereksiz işlemlerin yapılmasını engeller. Yani binlerce elemanın bulunduğu bir haritada belki de en fazla birkaç tane işaretçinin veri ilişkilendirme prosedürüne girmesi söz konusu olabilir.

Birinci bölümde anlatıldığı üzere bu konuyla ilgili daha önce yapılmış olan çalışmalar bulunmaktadır. [19]'daki çalışmaya göre bir işaretçi tespit edildiğinde haritadaki bütün işaretçiler veri ilişkilendirme işlemine tabi tutulmazlar. Bunun yerine, sadece robot üzerindeki sensörün algılama uzaklığı ve açısı içerisinde kalan işaretçiler bu adıma sokulur. CESLAM olarak adlandırılan bu yaklaşım şekil 4.1'de görsel olarak ifade edilmiştir. 21 tane işaretçi olan haritada robot sadece sensör algılama alanı içinde kalan 7 tane işaretçiyi veri ilişkilendirmeye sokmaktadır. Buradan kolaylıkla anlaşılacağı üzere CESLAM yöntemi büyük bir hesap yükünü çok aza indirmiştir. Bu yaklaşımın başarısı azımsanmayacak kadar büyük olmasına karşın, kullanılan sensör ve ortamdaki işaretçi sayısına göre algoritmanın etkisinin azalma olasılığı vardır. Örneğin; üç boyutlu tarama yapabilen ve algılama uzaklığı 10 metre olan bir lazer sensör kullanan robot, tek bir ölçümde çok sayıda işaretçi tespit edebilir.



Şekil 4.1 : CESLAM yöntemi: sensör algılama alanı içinde kalan gri renkli işaretçiler veri ilişkilendirmeye tabi tutulur.

Ayrıca parçacık filtresi temelli EZKH yaklaşımlarında bu teknik bazen, veri ilişkilendirmeye alınması gereken işaretçileri atlayabilir. Gri renkli büyük üçgenin robotu, beyaz renkli küçük üçgenin de bir parçacığı simgelediği şekil 4.2'de görüldüğü üzere bu durum; tespit edilen bir işaretçinin, ilgili parçacığın sensör algılama alanı dışında kaldığı zaman ortaya çıkar.



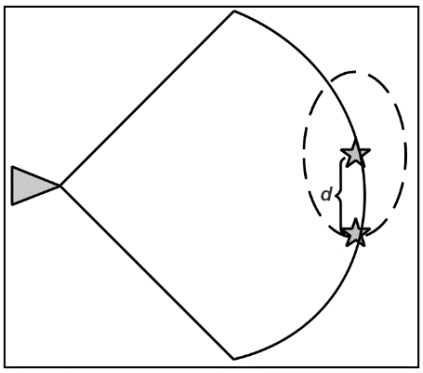
Şekil 4.2 : CESLAM'de oluşabilecek yanlış eşleştirme durumu.

Bu tez çalışmasında önerilen yöntemde, CESLAM'de olduğu gibi sensör algılama alanı içinde kalan işaretçiler değil bu alandan daha küçük dairesel bir alandaki işaretçiler ele alınır. Sensör algılama alanı ne kadar geniş olursa olsun bu dairenin alanı değişmediği için veri ilişkilendirme aşamasında her zaman çok az sayıda işaretçi hesaba katılır. Böylece, bu adımdaki işlem yükü hemen hemen aynı kalmaktadır.

4.2 Önerilen Yöntemin Teknik Ayrıntıları

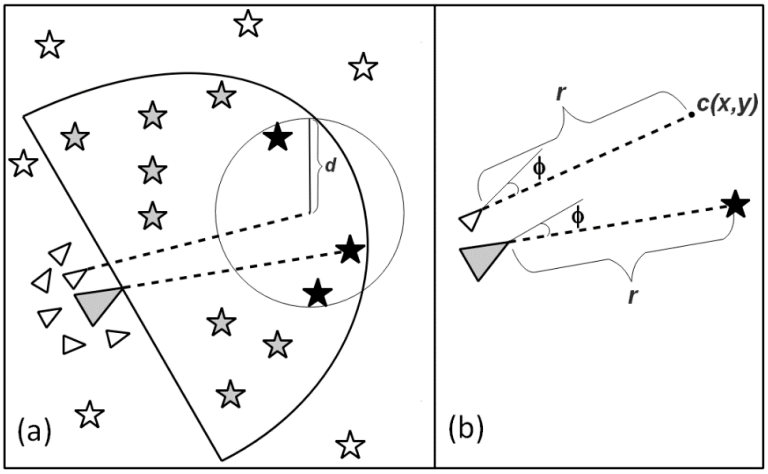
Önerilen yöntem uygulamada oldukça basit olmasına karşın hız performansını artırmadaki başarısı çok yüksektir. Çünkü; haritada çok büyük miktarda işaretçi olsa bile veri ilişkilendirme prosedürü uygulanan işaretçi sayısı çok düşük kalmaktadır.

Yöntemin ana fikri şu şekildedir: Robot, sensörden aldığı ölçüme göre parçacıklara ölçüm tahmini uygular. Merkezi bu tahmin sonucunda bulunan ve d yarıçapı genişliğinde bir dairesel alan belirlenir. Bu dairenin içinde kalan işaretçiler veri ilişkilendirmeye tabi tutulurken dairenin dışında kalan işaretçiler atlanır. Çünkü standart veri ilişkilendirme prosedürü uygulanırken karşılaştırılan bütün işaretçilerin benzerlik olasılıkları hesaplanır ve en büyük olasılıklı olan işaretçi ele alınır. Eğer bu işaretçi belli bir eşik değerinin altındaysa haritada olmayan yeni bir işaretçi olarak kaydedilir. Dairenin yarıçapı da bu eşik değeri gözönünde bulundurularak belirlenir. Bir işaretçinin konum belirsizliği ne kadar büyükse yakınında tespit edilen başka bir işaretçinin ona olan benzerlik olasılığı da o kadar fazladır. Bir başka deyişle, robotun bu iki işaretçiyi birbirinden ayırt etmesi için aralarındaki uzaklığın belli bir değerden fazla olması gerekir ve belirsizliğin artmasıyla bu uzaklık da artar. Dairenin dışında kalmasına rağmen bu benzerlik olasılığı eşik değerin üstünde olan bir işaretçi olabilir. Bu hatanın yaşanmaması için bir işaretçinin belirsizliğinin en fazla olabileceği durum gözönüne alınır. Bir lazer sensörünün işaretçi tespit edebileceği en uzak mesafede o işaretçinin belirsizliği en büyüktür. Bu bilgiler ışığında, dairenin yarıçapı şekil 4.3'te görüldüğü gibi deneysel ölçümlerle belirlenebilir. Belirsizliği gösteren elipsin iki yarıçapı vardır. Bunlardan biri diğerinden daha kısa olabilir. Bu kısa yarıçap, dairenin yarıçapı olarak seçilirse yukarıda bahsedildiği gibi benzerlik olasılığı eşik değerin üstünde olan bir işaretçi dairenin dışında kalabilir. O yüzden uzun olan yarıçap seçilir.



Şekil 4.3 : Dairenin yarıçapının ölçümle belirlenmesi.

Şekil 4.4a'da gri renkli büyük üçgen robotu, beyaz renkli küçük üçgenler parçacıkları, yıldızlar da işaretçileri simgelemektedir. Robot, aldığı ölçümü kullanarak parçacık için ölçüm tahmini yapar ve siyah işaretçileri kapsayan daireyi belirler. Şekil 4.4b'de görüldüğü gibi ölçümün açısı ve uzaklığı, parçacığın konumu ve yönelimine eklendiğinde dairenin merkezi hesaplanır. Bu dairenin içinde kalan siyah işaretçiler veri ilişkilendirmeye sokulurken dışında kalanlar da atlanır.



Şekil 4.4 : Önerilen yöntemde bir parçacık için dairenin belirlenmesi.

Her bir parçacık için farklı olan dairenin merkezi denklem 4.1 ve 4.2'deki gibi hesaplanır. burada ϕ ölçümün sensöre göre açısını, r ölçümün uzaklığını belirtirken c_x ve c_y de sırasıyla dairenin merkezinin x ve y koordinatlarını belirtir.

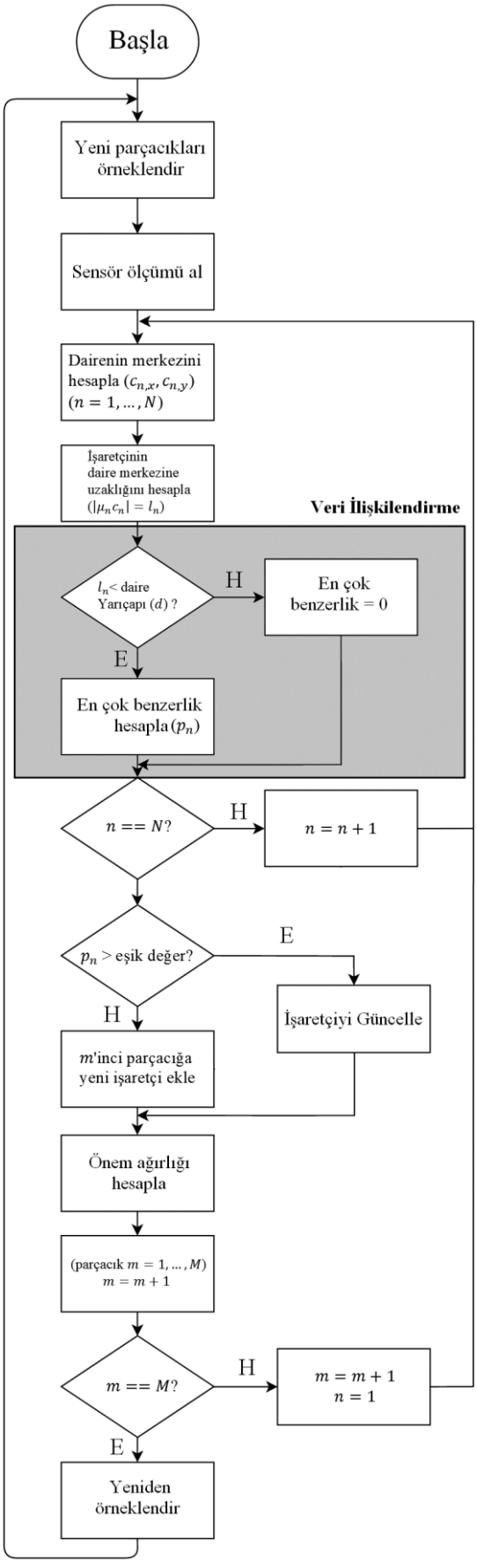
$$c_x = s_{t,x}^{[m]} + r\,cos(s_{t,theta}^{[m]} + \phi) \tag{4.1}$$

$$c_y = s_{t,y}^{[m]} + r\,sin(s_{t,theta}^{[m]} + \phi) \tag{4.2}$$

$\mu_{\mathrm{n,x}}^{[m]}$ ve $\mu_{\mathrm{n,y}}^{[m]}$ m'inci parçacığa ait n'inci işaretçinin sırasıyla x ve y koordinatlarını belirtmek üzere, haritadaki bir işaretçinin dairenin merkezine olan uzaklığı 4.3'teki gibi bulunur.

$$l_n = \sqrt{(c_x - \mu_{\mathrm{n,x}}^{[m]})^2 + (c_y - \mu_{\mathrm{n,y}}^{[m]})^2} \tag{4.3}$$

Eğer l_n, dairenin yarıçapı d'den büyükse bu işaretçi veri ilişkilendirmeye alınmaz. Eğer küçükse, benzerlik olasılığı hesaplanır. Dairenin içinde kalan bütün işaretçiler için benzerlik olasılığı hesaplandıktan sonra en büyük benzerliği olan işaretçi ölçümdeki işaretçidir denir. Şekil 4.5'te, geliştirilen yöntemin akış şeması görülmektedir.



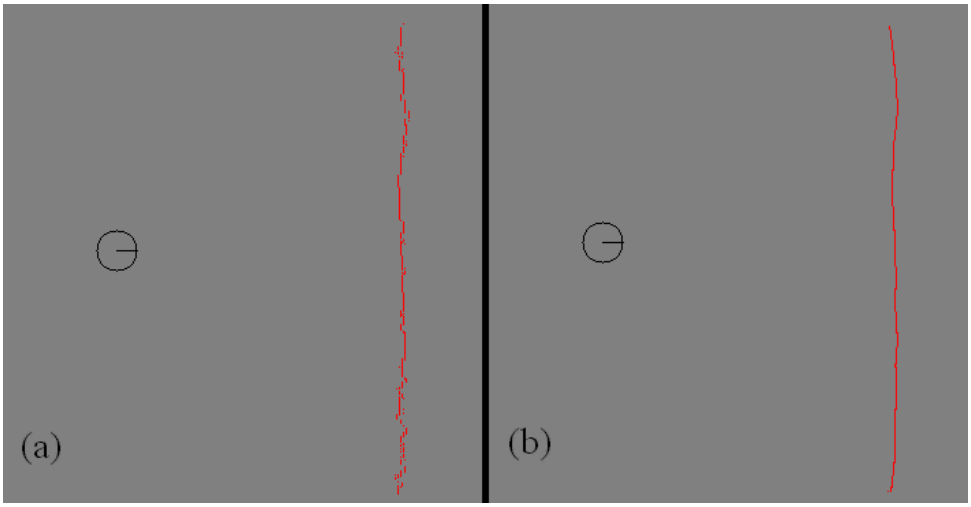
Şekil 4.5 : Geliştirilen yöntemin akış şeması.

5. UYGULAMA

Mobil robotlarda EZKH yöntemlerinin hız performansını artırmak için geliştirilen yöntemin başarısını görmek için Gazebo'da hazırlanan iki farklı haritada simülasyonlar gerçekleştirilmiştir. Üç farklı EZKH yaklaşımı her bir haritada Turtlebot platformu kullanılarak 30'ar defa çalıştırılmıştır. Elde edilen sonuçlar hız performansı, robotun gittiği yolun hesaplaması ve haritanın doğruluğu bakımından incelenmiştir.

5.1 Düzleştirme Filtresi

Yeni EZKH yönteminin uygulanması sırasında Kinect sensörü ile ilgili bazı zorluklar yaşandı. Çünkü Kinect daha çok, eğlence amaçlı olarak tasarlandığı için bilimsel ve endüstriyel çalışmalarda kullanılanılan diğer lazer sensörleri kadar doğrulukta ölçüm yapamamaktadır. Bu tez çalışmasında Kinect, 2-boyutlu lazer sensörü gibi kullanılmıştır. Şekil 5.1'de, robota 2.5 metre uzaklıktaki bir duvardan alınan iki ölçüm görülmektedir. Şekil 5.1a'da görülen çizim sensörden alınan verinin işlenmemiş halidir. Ölçüm uzaklığı az olmasına karşın elde edilen veri oldukça gürültülüdür ve işaretçi tespitini güçleştirmektedir. Şekil 5.1b'de ise düzleştirme filtresi uygulanmış bir ölçüm görülmektedir. Bu ölçüm, EZKH'de kullanmak için daha uygundur.

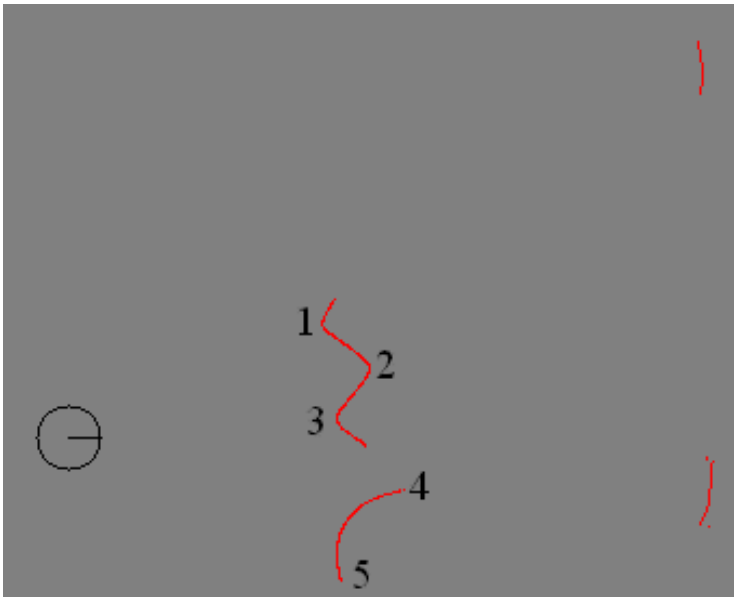


Şekil 5.1 : Kinect sensörü ile 2.5 metre uzaklıktaki düz bir duvardan alınan işlenmemiş ölçüm.

Sinyal işleme ve görüntü işleme gibi alanlarda genellikle ortalama filtresi, ortanca filtresi ve Gauss filtresi gibi düzleştirme filtreleri kullanılır. Uygulama kolaylığı ve iyi sonuç vermesi nedeniyle Kinect sensöründen alınan veri, ortalama filtresi uygulanarak kullanılmıştır. Bu filtrenin işleyişi şu şekildedir: sensörden gelen veri bir dizi olarak olarak düşünüldüğünde filtre edilecek elemanın kendisinin ve kendisinden önce ve sonra gelen n tane elemanın toplamının aritmetik ortalaması alınır. Elde edilen değer filtre edilen eleman ile değiştirilir. Bu işlem bütün bir dizi boyunca her bir elemena sırasıyla uygulanır. Ancak baştaki ve sondaki n tane eleman bu filtreleme işlemine tabi tutulamaz [54].

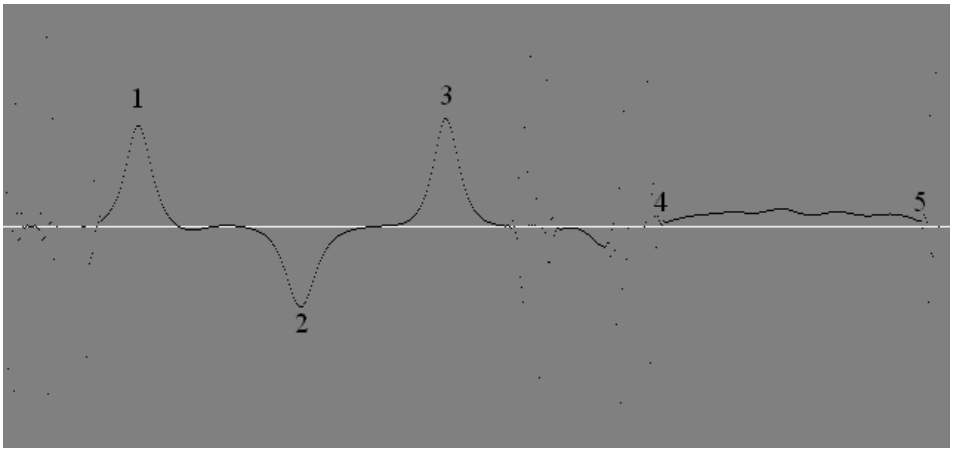
5.2 İşaretçi Çıkarımı

İşaretçi çıkarımı için bölüm 3'te anlatılan eğrilik fonksiyonu yöntemi kullanılmıştır. şekilde köşelerin, düzlüklerin ve yuvarlak bir cismin olduğu bir ortamdan alınan ölçüme ait çizim görülmektedir. Şekil 5.2'de de bu ölçümden hesaplanan eğrilik fonksiyonunun çizimi vardır. Şekil 5.2'de, 1'den 5'e kadar numaralandırılmış olan noktaların şekil 5.3'teki eğrilik fonksiyonunda karşılığı olan yerler de aynı sırayla numaralandırılmıştır. 1 ve 3 ile gösterilen köşeler, eğrilik fonksiyonunda yukarıya doğru tepe oluştururken 2 ile gösterilen köşe, eğrilik fonksiyonunda aşağıya doğru tepe oluşturmuştur. Bu özellik, işaretçilere kimlik ataması yapılırken büyük avantaj sağlar.



Şekil 5.2 : Köşeli, düz ve yuvarlak cisimlerin olduğu bir ortamdan alınan sensör ölçümü.

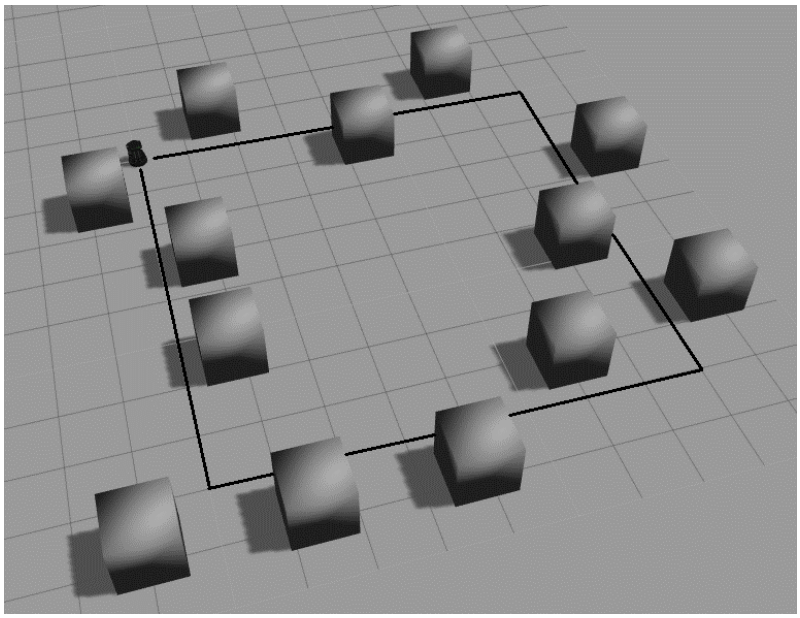
Ölçümde 4 ve 5 noktaları arasında kalan yuvarlak cisim eğrilik fonksiyonunda sıfırın üstünde düz bir çizgi olarak karşılık bulmuştur. Bunun gibi şekiller işaretçi olarak kullanılırken şeklin merkezini işaretçinin konumu gibi kabul etmek mümkündür.



Şekil 5.3 : Köşeli, düz ve yuvarlak cisimlerin olduğu bir ortamdan alınan sensör ölçümünün eğrilik fonksiyonu.

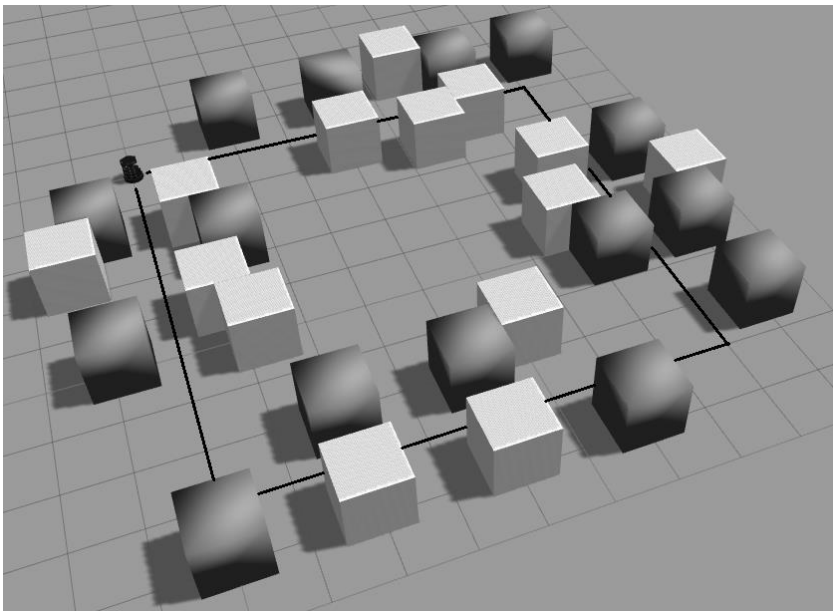
5.3 Gazebo Simülatöründe Oluşturulan Ortamlar ve Gerçek Uygulama Ortamları

Şekil 5.4'te görülen 1. ortamda robot, siyah çizgi ile gösterilen rotayı takip etmektedir ve bu rotada bir tur attığında toplamda 13 tane işaretçi tespit edebilmektedir. İşaretçiler küp şeklindeki cisimlerin, sensörün gördüğü köşe kısımlarıdır. Bu ortam, robotun bir ölçümde en fazla bir tane işaretçi tespit edebileceği şekilde hazırlanmıştır.



Şekil 5.4 : Gazebo simülatöründe hazırlanan 1. ortam.

Şekil 5.5'teki ortamda robot yine siyah çizgi ile gösterilen rotayı takip etmektedir ve bir tur sonunda toplamda 21 tane işaretçi tespit edebilmektedir. Robot bu ortamda ilerlerken bazen bir ölçümde sadece bir işaretçi bazen de iki işaretçi tespit edebilmektedir. Böylece ortamın karmaşıklığı arttıkça, önerilen yöntemin ne kadar başarılı olduğu kolayca gözlemlenebilmektedir.



Şekil 5.5 : Gazebo simülatöründe hazırlanan 2. ortam.

Şekil 5.6 ve şekil 5.7'de görülen laboratuvar ortamlarında robot, simülasyon ortamlarında olduğu gibi kare şeklinde bir rotayı takip etmektedir. Yine simülasyon ortamlarında olduğu gibi iki ortam, işaretçilerin sayısı ve konumu açısından farklı karmaşıklıkta olacak şekildedir. Böylece farklı EZKH yöntemlerinin başarılarının daha net anlaşılması sağlanmaktadır.

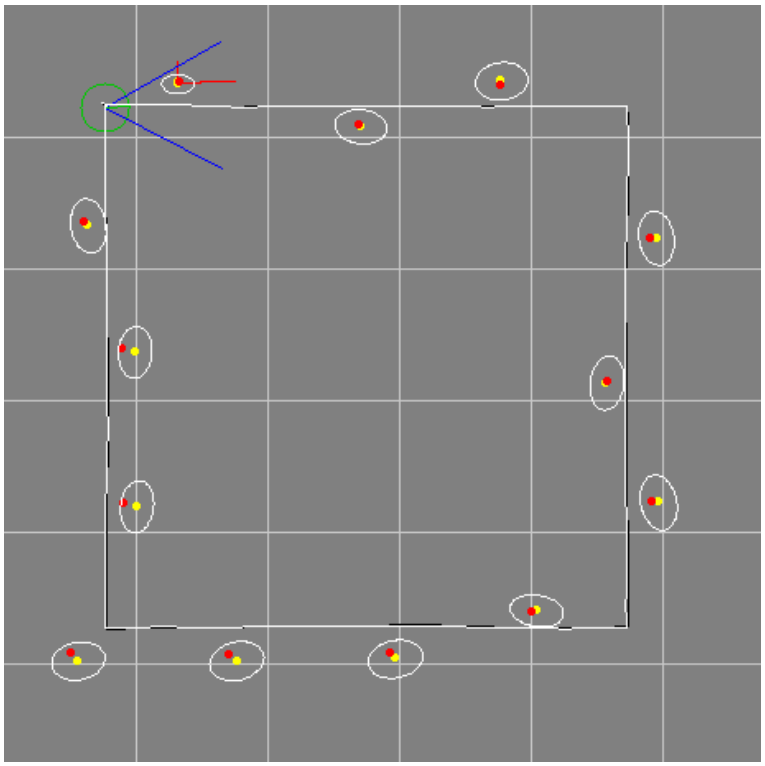
Şekil 5.6 : Gerçek uygulama ortamı-1.

Şekil 5.7 : Gerçek uygulama ortamı-2.

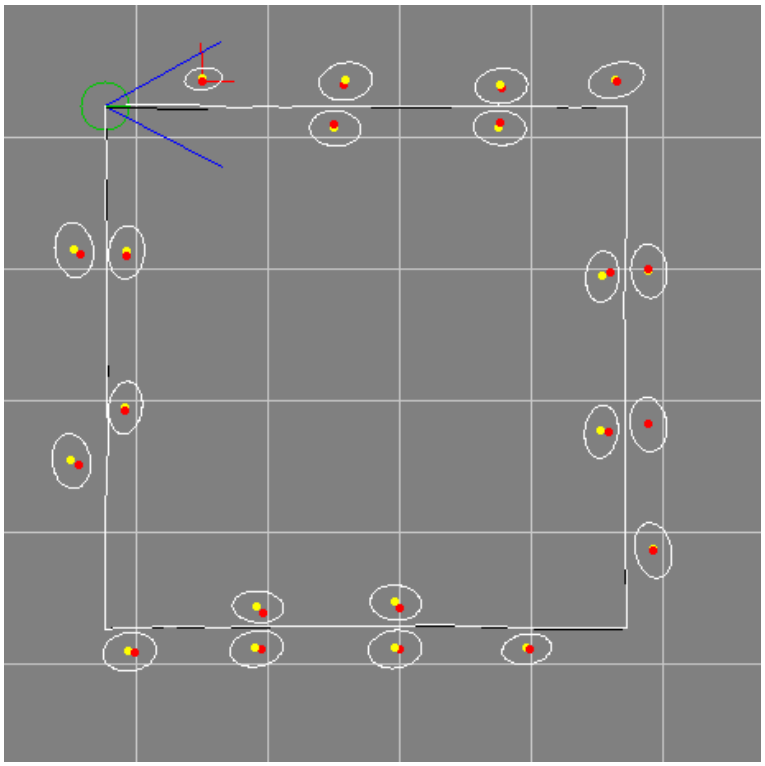
5.4 Geliştirilen Yöntem İle Oluşturulan Haritalar

Şekil 5.5 ve şekil 5.6'daki ortamlarda çalıştırılan robotun, önerilen yöntemle oluşturuduğu işaretçi haritaları şekil 5.7 ve şekil 5.8'de görülmektedir. Çizimlerde robot yeşil renkte belirtilmiş ve sensörün algılama alanının sınırları da mavi çizgi ile gösterilmiştir. Bu haritalar çizdirilirken parçacık filtresindeki önem ağırlığı en yüksek olan parçacığın oluşturduğu harita gözönüne alınmıştır. Kırmızı nokta ile işaretlenmiş yerler robotun gördüğü işaretçilerin gerçek konumları, sarı nokta ile işaretlenmiş yerler de bu işaretçilerin hesaplanan konumlarıdır. Merkezi sarı noktalar olan beyaz renkli elipsler işaretçilerin belirsizliklerini gösterir. Bu belirsizlikler, robot işaretçiyi ilk gördüğünde daha büyükken daha sonra robot tekrar aynı işaretçiyi gördüğünde güncellenerek azalmıştır.

Haritalarda ayrıca robotun gittiği gerçek yol ve EZKH ile hesaplanan yol da çizdirilmiştir. Beyaz renkli kare çizgi gerçek yolu, siyah renkli kare çizgi de parçacık filtresi ile hesaplanan yolu gösterir.



Şekil 5.8 : Önerilen yöntemle birinci simülasyon ortamında yürütülen robotun oluşturduğu harita.



Şekil 5.9 : Önerilen yöntemle ikinci simülasyon ortamında yürütülen robotun oluşturduğu harita.

5.5 Ölçüm Sonuçları

Üç farklı yöntem her bir simülasyon ortamında 30'ar defa çalıştırılarak harita hesaplama hataları, güzergah hataları ve çalışma süreleri ile ilgili sonuçlar elde edilmiştir. Çizelge 5.1'de 1. ve 2. ortamda tespit edilen işaretçilerin gerçek konumlarına göre santimetre cinsinden hataları verilmiştir. Buradaki değerler şu şekilde elde edilmiştir: Her bir hesaplanan işaretçinin gerçek konumuna olan uzaklığı 2-normuna göre hesaplandıktan sonra bu uzaklıklar toplanıp işaretçi sayısına bölünerek hata bulunmuştur.

Çizelge 5.1 : EZKH yöntemlerine ait işaretçi konumu hesaplama hataları.

EZKH Yöntemleri	İşaretçi Konum Hataları(cm) (1.Ortam)	İşaretçi Konum Hataları(cm) (2.Ortam)
FastSLAM1.0	6.8	9.7
CESLAM	6.9	9.6
Önerilen Yöntem	6.8	9.5

Çizelge 5.2'de üç farklı EZKH yönteminin güzergah hesaplama hataları görülmektedir. Bu hataları elde etmek için robotun her adımdaki hesaplanan konumu ile gerçek konumu arasındaki uzaklık yine 2-normuna göre hesaplandıktan sonra bütün adımlardaki uzaklıklar toplanmıştır.

Çizelge 5.2 : EZKH yöntemlerine ait işaretçi güzergah hesaplama hataları.

EZKH Yöntemleri	Güzergah Hataları(m) (1.Ortam)	Güzergah Hataları(m) (2.Ortam)
FastSLAM1.0	10.14	9.71
CESLAM	9.92	9.59
Önerilen Yöntem	9.83	9.72

Çizelge 5.3 ve çizelge 5.4'te üç farklı EZKH yönteminin simülasyon ortamları ve gerçek uygulama ortamlarındaki çalışma zamanları verilmiştir. Bu değerler, robotun harekete başladığı andan itibaren başladığı noktaya tekrar gelene kadar izlediği yol boyunca her bir döngüde harcadığı süreler toplanarak elde edilmiştir. Simülasyon ortamlarında elde edilen değerlerin gerçek uygulama ortamlarında elde edilen değerlerden çok daha büyük olmasının sebebi, simülasyon ortamlarının daha büyük olması ve bu ortamlardaki nesne sayısının daha fazla olmasıdır.

Çizelge 5.3 : EZKH yöntemlerinin simülasyon ortamlarındaki çalışma zamanları.

EZKH Yöntemleri	Çalışma Zamanı(sn) (1. Ortam)	Çalışma Zamanı(sn) (2.Ortam)
FastSLAM1.0	9.27	19.91
CESLAM	4.41	6.66
Önerilen Yöntem	4.19	5.96

Çizelge 5.4 : EZKH yöntemlerinin gerçek ortamlardaki çalışma zamanları.

EZKH Yöntemleri	Çalışma Zamanı(sn) (1.Ortam)	Çalışma Zamanı(sn) (2.Ortam)
FastSLAM1.0	0.81	1.22
CESLAM	0.52	0.69
Önerilen Yöntem	0.46	0.58

6. SONUÇ VE ÖNERİLER

Bu tez çalışmasında, günümüze kadar geliştirilen EZKH yöntemlerinin hız performanslarının artırılması için yeni bir yöntem geliştirilmiş ve simülasyon ortamında uygulanarak farklı EZKH algoritmalarıyla karşılaştırması yapılmıştır. Ancak bu karşılaştırmalar yapılmadan önce uygulamanın kolaylaşması için bir takım işlemler gerçekleştirilmiştir. İlk olarak, sensörden alınan gürültülü veri üzerinde düzleştirme filtresi uygulanmış ve daha tutarlı veriler elde edilmiştir. Böylece, ölçüm verisinin anlamlandırılmasının çok daha kolay olduğu anlaşılmıştır.

Ek olarak; işaretçi çıkarımı yapılırken, eğrilik fonksiyonu yönteminden yararlanılmıştır. Bu yöntem kullanılarak, geometrik şekilli cisimlerin tek bir fonksiyonla tespit edilebildiği görülmüş ve işaretçi kimliklendirmedeki başarısı ortaya konmuştur.

Bölüm 5'te elde edilen sonuçlar incelendiğinde, yeni yöntemin robot konumu hesaplama ve harita oluşturmadaki başarısının hemen hemen hiç değişmediği görülmüştür. Bunun yanında çalışma süreleri incelendiğinde, önerilen yöntemin parçacık filtresi tabanlı ilk yaklaşımlardan biri olan FastSLAM 1.0'a göre büyük oranda ilerleme kaydettiği anlaşılmıştır. FastSLAM 1.0'dan sonra geliştirilen CESLAM yöntemiyle bu çalışmada önerilen yöntem karşılaştırıldığında her ne kadar çok büyük ölçekli bir hız geliştirilmesi görülmese de açık bir şekilde yeni yöntemin hız performansının daha iyi olduğu anlaşılmıştır. Bu sonuçlardan anlaşılmaktadır ki ortamdaki işaretçiler arttıkça, geliştirilen EZKH algoritmasının başarısı daha belirgin hale gelmektedir.

Düzlemsel yörüngede ilerleyen robotlar ve iki boyutlu ölçüm yapan sensörler için başarısının ortaya çıktığı bu yöntemin üç boyutlu ölçüm alan sensörler için daha da başarılı sonuçlar vereceği düşünülmektedir. Çünkü, bu tarz sensörler çok daha fazla bilgi toplarlar ve bunun sonucunda tek bir ölçümde alınabilecek işaretçi sayısı da çok fazla olabilmektedir. Geliştirilen yeni yöntemin, farklı sensörler kullanarak farklı EZKH yöntemleri üzerinde uygulanıp sonuçlarının incelenmesi planlanmaktadır.

7. KAYNAKLAR

[1] **Ort, T.** (2013). *Art and Life in Modernist Prague: Karel Čapek and His Generation, 1911-1938*. Springer.

[2] **Siciliano, B., & Khatib, O**. (2016). *Handbook of robotics.* Springer.

[3] **Thrun, S., Burgard, W., & Fox, D.** (2005). *Probabilistic robotics*. Cambride, MA: MIT Press.

[4] **Smith, R. C., & Cheeseman, P.** (1986). On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, 5(4), 56–68. doi:10.1177/027836498600500404

[5] **Smith, R., Self, M., & Cheeseman, P.** (1990). Estimating Uncertain Spatial Relationships in Robotics. *Autonomous Robot Vehicles*, 4(April), 167–193. doi:10.1109/ROBOT.1987.1087846

[6] **J. J. Leonard and H. F. Durrant-Whyte.** (1991). Simultaneous map building and localization for an autonomous mobile robot, *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91*, (pp. 1442-1447). IEEE/RSJ International Workshop on, Osaka, 1991, vol.3.

[7] **Dissanayake, M. W. M. G., Newman, P., Durrant-Whyte, H. F., Clark, S., & Csorba, M.** (2000). An experimental and theoretical investigation into simultaneous localisation and map building. *In Experimental Robotics VI* (pp. 265–274). London: Springer London. doi:10.1007/BFb0119405

[8] **J. Guivant & E. Nebot.** (2001). Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transaction of Robotic and Automation*, 17(3), 242-257. doi:10.1109/70.938382

[9] **Leonard, J. J., & Feder, H. J. S.** (2000). A computationally efficient method for large-scale concurrent mapping and localization. In D. Koditschek & J. Hollerbach (Eds.), *Robotics Research: The Ninth International Symposium*. Snowbird, Utah: Springer Verlag.

[10] **Lu, F., & Milios, E.** (1997). Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, 4(4), 333–349. doi:10.1023/A:1008854305733

[11] **Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B.** (2002). FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. *In Eighteenth National Conference on Artificial Intelligence* (pp. 593–598). Menlo Park, CA, USA: American Association for Artificial Intelligence.

[12] **Murphy, K. P.** (2000). Bayesian Map Learning in Dynamic Environments. In S. A. Solla, T. K. Leen, & K. Müller (Eds.), *Advances in Neural Information Processing Systems* 12 (pp. 1015–1021). MIT Press. Retrieved from http://papers.nips.cc/paper/1716-bayesian-map-learning-in-dynamic-environments.pdf

[13] **C. Kim, R. Sakthivel and W. K. Chung**, (2008). Unscented FastSLAM: A Robust and Efficient Solution to the SLAM Problem. *IEEE Transactions on Robotics*, 24(4), 808-820. doi:10.1109/TRO.2008.924946

[14] **Kwak, N., Kim, I. K., Lee, H. C., & Lee, B. H.** (2007). Adaptive prior boosting technique for the efficient sample size in FastSLAM. *IEEE International Conference on Intelligent Robots and Systems* (pp. 630–635). doi:10.1109/IROS.2007.4399039

[15] **Xu, W., Jiang, R., Xie, L., Tian, X., & Chen, Y.** (2017). Adaptive Square-root Transformed Unscented FastSLAM with KLD-resampling. *International Journal of Systems Science*, 48(6), 1322–1330. doi:10.1080/00207721.2016.1256449

[16] **Chang, H. J., Lee, C. S. G., & Hu, Y. C.** (2004). A computational efficient SLAM algorithm based on logarithmic-map partitioning. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*

(IROS) (IEEE Cat. No.04CH37566), 2, 1041–1046. doi:10.1109/IROS.2004.1389534

[17] **M. Yokozuka and O. Matsumoto,** (2012). Sub-map dividing and re-alignment FastSLAM with scalable voxel map system. *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Kachsiung, 2012, pp. 180-185. doi: 10.1109/AIM.2012.6265914

[18] **K. Kouzoubov and D. Austin,** (2004). Hybrid topological/metric approach to SLAM. Proceedings. *ICRA '04. 2004 IEEE International Conference on Robotics and Automation*, pp. 872-877 Vol.1. doi:10.1109/ROBOT.2004.1307259

[19] **C. K. Yang, C. C. Hsu and Y. T. Wang,** (2013). Computationally efficient algorithm for simultaneous localization and mapping (SLAM). *10th IEEE International Conference on Networking, Sensing and Control (ICNSC)*, Evry, 2013, pp. 328-332. doi: 10.1109/ICNSC.2013.6548759

[20] **Url-2** *<http://www.turtlebot.com/turtlebot2/>* alındığı tarih: 10.04.2017

[21] **Url-1** *<http://wiki.ros.org/Robots/TurtleBot>* alındığı tarih: 10.04.2017

[22] **Oliver, A., Kang, S., Wünsche, B. C., & MacDonald, B.** (2012). Using the Kinect As a Navigation Sensor for Mobile Robotics. *27th Conference on Image and Vision Computing*, (pp. 509–514). New York, NY, USA: ACM. doi:10.1145/2425836.2425932

[23] **Url-3** *<https://msdn.microsoft.com/en-us/library/jj131033.aspx>* alındığı tarih: 10.04.2017

[24] **Quigley, M., Berger, E., & Ng, A. Y.** (2007). STAIR : Hardware and Software Architecture. *AAAI 2007 Robotics Workshop*, Vancouver, BC, 31–37.

[25] **Boucher, S.** (2012). Obstacle Detection and Avoidance Using TurtleBot Platform and XBox Kinect. New York, NY, USA.

[26] **Koenig, N. & Howard, A.** (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* (Vol. 3, pp. 2149-2154). IEEE.

[27] **Url-4** <*http://opencv.org/about.html/*> alındığı tarih: 14.04.2017.

[28] **M. R. Naminski.** (2013). *An Analysis of Simultaneous Localization and Mapping Algorithms*. Macalester Math, Statistics and Computer Science Department.

[29] **Leung, K. Y. K.** (2010). *An Introduction to Multi-robot Simultaneous Localization and Mapping*. Middlebury College. Retrieved from http://middarchive.middlebury.edu/cdm/ref/collection/scholarship/id/159.

[30] **Durrant-Whyte, H. & Bailey, T.** (2006). Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE robotics & automation magazine*, *13* (2), 99-110.

[31] **Riisgaard, S., & Blas, M. R**. (2004). *SLAM for Dummies: A Tutorial Approach to Simultaneous Localization and Mapping.* Retrieved from http://ocw.num.edu.mn/NR/rdonlyres/Aeronautics-and-Astronautics/16-412JSpring-2005/9D8DB59F-24EC-4B75-BA7A-F0916BAB2440/0/1aslam_blas_repo.pdf.

[32] **Srinivasan, N.** (2010). Feature based landmark extraction for real time visual SLAM. *2nd International Conference on Advances in Recent Technologies in Communication and Computing*, (pp. 390–394). doi:10.1109/ARTCom.2010.10.

[33] **F. Bonaccorso, G. Muscato and S. Baglio,** (2012). Laser range data scan-matching algorithm for mobile robot indoor self-localization. *World Automation Congress*, Puerto Vallarta, Mexico, pp. 1-5.

[34] **Fischler, M. A., & Bolles, R. C.** (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Communications of the ACM, 24(6), 381–395. doi:10.1145/358669.358692.

[35] **Núñez, P., Vázquez-Martín, R., Del Toro, J. C., Bandera, A., & Sandoval, F.** (2008). Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation. *Robotics and Autonomous Systems, 56* (3), 247-264.

[36] **Allawi, Z. T., & Abdalla, T. Y.** (2014). Article: An Accurate Dead Reckoning Method based on Geometry Principles for Mobile Robot Localization. *International Journal of Computer Applications*, 95(13), 21–25. doi:10.5120/16654-6632.

[37] **R. Siegwart, I. R. Nourbakhsh.** (2004). *Introduction to Autonomous Mobile Robots*. The MIT Press.

[38] **D. Scaramuzza and F. Fraundorfer,** (2011). Visual Odometry [Tutorial]. *IEEE Robotics & Automation Magazine*, 18(4), pp. 80-92. doi:10.1109/MRA.2011.943233.

[39] **Aqel, M. O. A., Marhaban, M. H., Saripan, M. I., & Ismail, N. B.** (2016). Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, *5*(1), 1897. doi:10.1186/s40064-016-3573-7.

[40] **Hähnel, D., Burgard, W., Wegbreit, B., & Thrun, S.** (2003). Towards Lazy Data Association in {SLAM}. *International Symposium of Robotics Research (ISRR'03)*. Sienna, Italy: Springer.

[41] **Cooper, A.** (2005). *A comparison of data association techniques for simultaneous localization and mapping*. Massachusetts Instıtute of Technology. Retrieved from http://dspace.mit.edu/handle/1721.1/32438.

[42] **C. Stachniss, D. Hahnel & W. Burgard,** (2004). Exploration with active loop-closing for FastSLAM. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 2004, pp. 1505-1510 vol.2.

[43] **Kleinbauer, R.** (2004). *Kalman Filtering Implementation with Matlab Study Report in the Field of Study. Technology.* Retrieved from https://pdfs.semanticscholar.org/cb43/08212420e350701233495490c2a5c8905cd4.pdf.

[44] **Welch, G., & Bishop, G.** (1995). An Introduction to the Kalman Filter. Chapel Hill, NC, USA: University of North Carolina at Chapel Hill.

[45] **E. A. Rückert.** (2009). *Simultaneous Localisation and Mapping for Mobile Robots with Recent Sensor Technologies*. Graz University of

Technology, Institute for Computer Graphics and Vision, Master Thesis. Austria, December, 2009.

[46] **Moral, P. Del.** (1997). Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de l'Académie Des Sciences - Series I - Mathematics*, 325(6), 653–658. doi:10.1016/S0764-4442(97)84778-7.

[47] **Kim, S., Shephard, N., & Chib, S.** (1998). Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models. *The Review of Economic Studies*, 65(3), 361. doi:10.1111/1467-937X.00050.

[48] **Gustafsson, F., Gunnarsson, F., Bergman, N., Forssell, U., Jansson, J., Karlsson, R., & Nordlund, P.-J.** (2002). Particle Filters for Positioning, Navigation, and Tracking. *IEEE Transactions on Signal Processing*, 50(2), 425–437. doi:10.1109/78.978396.

[49] **Fox, D., Burgard, W., Dellaert, F., & Thrun, S.** (1999). Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. *16th National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, (pp. 343–349). Menlo Park, CA, USA.

[50] **E. Orhan.** (2012). *Bayesian Inference: Particle Filtering*. Department of Brain & Cognitive Sciences, University of Rochester. USA.

[51] **Montemerlo, M., & Thrun, S.** (2010). *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics (1st ed.)*. Springer Publishing Company, Incorporated.

[52] **Kong, A., Liu, J. S., & Wong, W. H.** (1994). Sequential Imputations and Bayesian Missing Data Problems. *Journal of the American Statistical Association*, 89(425), 278–288. doi:10.1080/01621459.1994.10476469

[53] **Havangi, R., Nekoui, M. A., & Teshnehlab, M.,** (2012). An improved FastSLAM framework using soft computing." *Turkish Journal of Electrical Engineering & Computer Sciences* 20.1 (2012): 25-46. doi:10.3906/elk-1004-504.

[54] **Url-5** <*http://homepages.inf.ed.ac.uk/rbf/HIPR2/filtops.htm*> alındığı tarih: 26.04.2017.

ÖZGEÇMİŞ

Ad-Soyad **:** Ziya Uygar YENGİN

Doğum Tarihi ve Yeri **:** 12.04.1989/Ağlasun

E-posta **:** yengin@itu.edu.tr

ÖĞRENİM DURUMU:

- **Lisans** **:** Kocaeli Üniversitesi Mekatronik Mühendisliği (2013)
- **Yüksek Lisans** **:** İstanbul Teknik Üniversitesi (halen)