ÖZ

GEZGİN ROBOTLARDA EŞ ANLI HARİTALAMA VE KONUM BELİRLEME

Orkun ALP

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Otonom gezgin robotlar görevlerini yerine getirmek için gezinim sırasında öncelikli olarak konumlarını belirlemelidir. Robot konumunun belirlenmesi odometrik veriler, algılayıcı ölçümler ve harita bilgilerinin birleştirilmesiyle gerçekleştirilir. Haritalar başlangıçta robota verilebileceği gibi robotun hareketi sırasında da oluşturulabilir. Bu durumda robot konumunun belirlenmesi, eş zamanlı olarak ortam haritasının çıkartılması ve bu haritanın kullanılmasıyla mümkündür. Bu tez çalışmasında ortam haritası oluşturulurken, robot konumunun da eş zamanlı olarak belirlenmesi hedeflenmiştir. Ortamın öznitelik tabanlı haritasını çıkarmak için Geliştirilmiş Üçgenleme Tabanlı Birleşim algoritması, gezgin robotun konumunda ve yöneliminde odometriden kaynaklanan hataların azaltılabilmesi için de genişletilmiş Kalman süzgeci kullanılmıştır. Deneysel çalışmada üzerinde 16 adet ses ötesi algılayıcı ve tekerleri üzerinde 2 adet optik şaft kodlayıcı bulunan Pioneer 3DX otonom gezgin robot kullanılmıştır. Pioneer robotlar için hazırlanmış MobilSim benzetimcisi ve Matlab programı kullanılarak benzetim tabanlı uygulamalar gerçekleştirilmiştir. Üçgenleme Tabanlı Birleşim algoritmasına ilave edilen ek süreçler ile gerçekte harita üzerinde yer almayan sahte kenar noktalar filtrelenerek algoritmanın performansı artırılmıştır. GKS başlangıç koşullarına göre konum belirleme performansı değerlendirilmiştir. İki yöntem bir arada kullanılarak robotun son konumundaki odometriden kaynaklanan hatanın azaldığı gözlenmiştir. Gerçek konum ile kestirilen robot konumu arasındaki hatanın GKS tutarlığı için istenilen sınır değerleri arasında kaldığı görülmüştür.

ANAHTAR SÖZCÜKLER: Eş Anlı, Haritalama, Konum Belirleme, Üçgenleme Tabanlı Birleşim, Öznitelik Tabanlı Haritalama, Genişletilmiş Kalman Süzgeci.

Danışman: Yrd. Doç. Dr. Hamit ERDEM, Başkent Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü.

ABSTRACT

CONCURRENT MAPPING AND LOCALIZATION IN MOBILE ROBOTS

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

First of all autonomous mobile robots have to localize their own positions to perform their tasks. By fusing the odometric data, measurements acquired from sensors and map features, it is possible to localize the robot position. Maps either can be given at the beginning of motion or can be built by a mobile robot while it wanders around the environment. In this situation it is necessary to build the map and use it simultaneously to localize the robot pose. In this research, it is aimed to localize the robot position and to map the environment simultaneously. Advanced Triangulation Based Fusion algorithm was used to build a feature-based map of the environment. Extended Kalman filter was used to reduce the errors based on odometric motion. In experimental work, Pioneer 3DX, an autonomous mobile robot with 16 sonars and 2 optical shaft encoders located on the wheels were used. The simulation-based applications were realized by using the MobileSim simulator which is prepared for Pioneer robots and Matlab program. After the additional two processes have been incorporated into Triangulation Based Fusion algorithm, it is succeed to remove the false edge points from the map and to enhance the performance of the algorithm. According to the initial conditions of EKF, the performance of localization was discussed. It is observed that last position error of the mobile robot is decreased with the use of EKF and ATBF together. The error between the real and the estimated robot positions during the motion was remained into the desired boundary range. This fact guarantees the consistency of EKF-based concurrent mapping and localization.

KEY WORDS: Concurrent, Mapping, Localization, Triangulation Based Fusion, Feature Based Mapping, Extended Kalman Filter
Advisor: Yrd. Doç. Dr. Hamit ERDEM, Başkent University, The Department of Electrical and Electronics Engineering

İÇİNDEKİLER LİSTESİ

ŞEKİLLER LİSTESİ

ÇİZELGELER LİSTESİ

SİMGELER VE KISALTMALAR LİSTESİ

r	:	Algılayıcı mesafe bilgisi
θ	:	Yay açıklığını gösteren açı
β	:	Sonar algılayıcı merkezi ile bütünsel yatay düzlem arasında kalan açı
n_t	:	Üçgenleme sayısı
d_1	:	Mesafe farkı için eşik değeri
d_2	:	Maksimum sapma
d_f	:	Yer gösterici boyutu
x_g	:	Bütünsel yatay düzlem
y_g	:	Bütünsel düşey düzlem
x_r	:	Robotun yatay düzlemi
y_r	:	Robotun düşey düzlemi

EKBH	:	Eş Zamanlı Konum Belirleme ve Haritalama
GKS	:	Genişletilmiş Kalman Süzgeci
TBF	:	Üçgenleme Tabanlı Birleşim
GPS	:	Bütünsel Konumlandırma Sistemi
T.O.F	:	Uçuş süresi (time of flight)
MD	:	Mahalanobis Uzaklığı
YG	:	Yer Gösterici
BUEYK	:	Bireysel Uyumluluk ile En Yakın Komşuluk

SÖZLÜK LİSTESİ

EKBH	: SLAM
Yer gösterici	: Landmark
Öznitelik	: Feature
Izgara	: Grid
Haritalama	: Mapping
Konum Belirleme	: Localization
Ses ötesi	: Ultra Sonic
Üçgenleme Sayısı	: Number of Triangulation
Çevrim dışı	: Off-line
Olasılıksal Haritalama	: Stochastic Mapping
Doğal Yer göstericiler	: Natural Landmarks
Yapay Yer Göstericiler	: Artificial Landmarks
Öznitelik Tabanlı Harita	: Feature-Based Map
Izgara Tabanlı Harita	: Grid-Based Map
En Yakın Komşuluk	: Nearest Neighbor
Veri İlişkilendirme	: Data Association
Hesaplama Karmaşıklığı	: Computational Complexity
Bireysel Uyumluluk ile En Yakın Komşuluk	: Individual Compatibility Nearest Neighbor

1.GİRİŞ

Gelişen teknolojiyle birlikte, günümüzde robotların bir çoğu gezginlik özelliğine sahiptir. Bu gezgin özellik robot üzerine takılan teker, bacak veya palet gibi yapılarla sağlanmış durumdadır. Robotların gezginlik özelliğini yerine getirebilmeleri otonom özelliklerinin olması ile mümkündür. Otonom olma özelliği, robotlar üzerinde bulunan işlemciler sayesinde daha önceden tanımlanmış veya tanımlanmamış bir durum karşısında karar verme yeteneğine sahip olmasıdır. Bu karar verme yeteneği robot algılayıcıları sayesinde çevreden elde edeceği veriler ile olur. Robot teknolojisinin gelmeye çalıştığı nokta, otonom özelliğe sahip olan gezgin robotlara verilecek görevleri, kusursuz şekilde yapmalarını sağlamaktır. Gezgin robotların görevlerini yerine getirebilmeleri için bulunduğu ortamı bilmesi veya öğrenmesi, dolayısıyla kendi konumunu belirleyebilmesi gerekir. Robot ortamı biliyor diğer bir ifade ile ortam haritasına sahipse çevresine bakarak nerede olduğunu belirleyebilir. Ancak robotun ortamı bilmediği durumlarda da görevini yerine getirebilmesi beklenir. Çünkü robot ortam ile ilgili her zaman ön bir bilgiye sahip olmayabilir. Bu durumda robot ortamı öğrenirken nerede olduğunu da tespit etmelidir.

Eş zamanlı konum belirleme ve haritalama (EKBH), (Simultaneous Localization and Mapping veya Concurrent Mapping and Localization) gezgin robot uygulamalarında birbiri ile ilişkilendirilmiş iki probleme hitap eder. Bu problemlerden birincisi konum belirlemedir. Robot ortam üzerinde “Neredeyim?” sorusunu yanıtlamaya çalışır. İkincisi ise ortamın haritalanmasıdır. Burada robot “Ortam neye benziyor?” sorusuna yanıt arar. Herhangi bir ortamı haritalamak için robotun sensör ölçümleri alması gerekir. Robot konumuna göre ölçülmüş olan engel verilerinin birleştirilmesiyle haritalama yapılabilir. Ancak ortam haritalandırılması bu ölçümlerin göreceli koordinat verilerine göre değil bütünsel koordinat verilerine dönüştürülmesiyle yapılır. Özetle konum belirlemek için haritaya, harita çıkarmak için de konum bilgisini ihtiyaç vardır. Dolayısıyla yukarıdaki iki soruyu istatistiksel bir bakış içerisinde tek bir soruda birleştirerek EKBH için çözüm aranması gerekir. “Şu ana kadar elde ettiğim veriler ile ortamın en muhtemel sürümü içinde nerede sayılırım ?” Bu soru robot konum kestirimi yapılırken aynı anda harita kestiriminin de yapılmasıyla cevaplanabilir [1].

Ortam haritasının tam olarak bilindiği durumlar için robot konumunun kestirimi ile ilgili günümüze dek bir çok çalışma yapılmıştır [2, 3]. Konum belirleme problemlerinde ortam haritasının %100 bilindiği varsayılır ve ortamdaki tüm nesnelerin durağan olduğu kabul edilir. Diğer bir ifade ile ortam içersinde tek hareketli nesnenin robot olduğu düşünülür. Genellikle konum belirleme algoritmaları girdi olarak ;

1. Ortamın geometrik haritasını,
2. Robot başlangıç konumunu,
3. Odometrik verileri,
4. Sensör ölçümlerini

alır.

Başarılı bir konum belirleme algoritması bu girdileri kullanarak ortamdaki robot konumuyla ilgili en iyi kestirimi yapar. Ortam birçok farklı biçimde tanımlanabilir. Bu tanımlama ya tamamen geometrik harita gösterimi ya da yer gösterici konum bilgileri ile olabilir.

Robot uygulamalarındaki haritalama ise, robot konumunun doğru olarak bilindiği durumlarda robotun ortam içinde hareket etmesiyle, doğru bir ortam haritası çıkarılmasına dayanır. Haritalama ortam içinde robot konumunun %100 bilindiği varsayılarak yapılır. Elfes ve Moravec tarafından harita çıkarmak için geliştirilen doluluk ızgara metodu ile ilk kez geometriksel ortam haritası çıkartılmıştır [4,5,6].

EKBH ile ilgili çalışmalar ise son on yılda hız kazanmıştır. Leonard ve Durrant-Whyte genişletilmiş Kalman süzgeci (GKS) kullanarak yapmış oldukları robot konum kestirimi ile EKBH probleminin çözümüne öncülük etmişlerdir [7]. Bu çalışmada GKS kullanılarak konumları önceden bilinen yer göstericiler üzerinden alınan ölçümler doğrultusunda robot konum kestirimi yapılmıştır. Cox, ses ötesi algılayıcılardan elde ettiği mesafe verilerini kullanarak, bunların ortamdaki yer göstericiler ile eşleştirilmesini özyinelemeli yöntem kullanarak yapmıştır [8].

EKBH ile ilişkili ilk çalışma ortamın topolojik haritasını oluştururken robot konumunun da beraberinde belirlenmesi prensibini geliştiren araştırmacılar Chatila ve Laumond tarafından yapılmıştır [9]. Bu çözüm olasılıksal bir bakış açısı ile yapılmış olmamasına rağmen, her iki problem bir arada ilk kez ele alınmıştır. Bu prensiplere dayanarak Smith, Self ve Cheeseman, Csorba ile birlikte konum belirleme ve haritalamanın eş zamanlı yapılabilmesi için çözüm önerilerini ortaya koymuşlardır. Böylece bu iki probleme ilk kez olasılıksal bir yöntem ile çözüm önerisi getirilmiştir [10,11]. Bu çalışmada harita bir takım yer göstericilerden oluşturulmuş ve model belirsizliği için kovaryans matris tanımlanmıştır. GKS kullanılarak her bir yer gösterici için ortalama konum değerleri kestirilmeye çalışılmıştır. Bu nedenle robot konum kestirimi yapılırken, eş zamanlı olarak olasılıksal harita kestirimi de yapma yaklaşımı ilk kez bu çalışmalar sonunda ortaya konmuştur. Günümüze kadar bu iki problemin bir arada ele alındığı ve olasılıksal yaklaşım ile çözüm arandığı bir çok başarılı araştırma yapılmıştır [12,13,14]. Kapalı ve açık alandaki uygulamalarının yanında, otonom sualtı robot araştırmalarında da EKBH uygulamalarına yer verilmeye başlanmıştır [15].

EKBH'nin robot uygulamalarındaki yeri ise çok önemlidir. Özellikle daha önceden ortamın keşfedilmediği veya ortam ile ilgili ön bir bilgi sahibi olunmadığı (örneğin Mars gezegeni yüzeyinde yapılacak araştırmalar, mağara keşifleri) durumlarda da tamamen otonom olarak çalışabilen robotlara ihtiyaç duyulmaktadır. Bu tür ortamlarda da görevlerini yerine getirebilecek robot sistemleri tasarlamak robot teknolojisinin ulaşmaya çalıştığı başlıca hedeflerdendir. Ortamın daha önceden tanımlanmamış olması işleri biraz daha karmaşık hale sokmaktadır. Bu durumda gezgin robotların görevlerini yerine getirebilmeleri için bulunduğu ortamı öğrenmesi bunun için de kendi konumunu bilmesi gerekir. Bu problemin çözümü de, gezgin robotun eş zamanlı olarak konum belirleyip çevreyi tanımlayabilmesi ile mümkün olur.

EKBH'nin ilk ele alındığı günden bugüne kadar, olasılıksal yaklaşım metodu ile çözüm aranması standart bir hale gelmiştir. Genel çözüm Kalman algoritması ile ilişkilendirilerek tanımlanmış ve Parçacık süzgeci algoritmalarının (FastSLAM) da bu uygulama içinde kullanılmaya başlamasıyla, algoritma hesaplama

karmaşıklığı $O(N^3) \rightarrow O(logN)$'e düşürülebilmiştir. Dolayısıyla EKBH probleminin çözümünü olasılıksal yaklaşım metotları oluşturmaktadır.

EKBH probleminin çözümü temel birkaç adımdan oluşmaktadır. Bu adımlar sonunda beklenen hedef ortam hakkındaki bilgiler kullanılarak robot konumunun belirlenebilmesidir. Ancak ortam hakkındaki bilgiler daha önceden bilinmediği için gezinim sırasında elde edilmelidir. Elde edilen bu bilgiler ile robot hareketi (odometri) birleştirilerek hem robot konumu hem de çevre tanımlanmalıdır. Robotun yalnızca odometrik verilerine güvenerek, konumunun belirlenmesi hatalı sonuçlar doğuracaktır [3]. EKBH uygulamasının yapılabilmesi için robot gezinimi sırasında ortamın da tanımlanması gerekir. Bu tanımlana ızgara tabanlı veya öznitelik tabanlı haritalama ile yapılır [16]. EKBH uygulamalarında daha çok öznitelik tabanlı haritalama kullanılır. Öznitelik tabanlı haritalar ortam içindeki nesnelerin konumunu gösteren haritalardır. Bu nesneler yer göstericiler olarak adlandırılır. Kapalı alanlarda doğal yer göstericilere (kendiliğinden var olan) kapı dikmeleri, masa ayakları, pencereler, duvar köşeleri ve kenarları örnek olarak verilebilir. Dolayısıyla çeşitli algılayıcılar kullanılarak bu gibi yer göstericilerin konumlarının tespit edilmesiyle öznitelik tabanlı haritalar oluşturulabilir. EKBH için gerekli olan diğer bir konuda veri ilişkilendirilmesidir. Bu ilişkilendirme algılayıcı ölçümleri ile yer göstericilerin doğru şekilde eşleştirilmesi için yapılır [17].

GKS, EKBH problemin çözümü için çok yaygın olarak kullanılan başarılı bir durum uzay kestirim yöntemidir [2].

Bu tez çalışmasında;

1. GKS algoritması kullanılarak robot konumunun ve yer gösterici konumlarının eş zamanlı olarak kestirimi hedeflenmiştir. GKS odometrik hatanın azaltılması ve robot-yer gösterici konumlarının eş zamanlı olarak kestirimi için kullanılmıştır.

2. Sensör olarak ses ötesi algılayıcılar ile çalışılmıştır. Yer gösterici çıkarmak için Geliştirilmiş Üçgenleme Tabanlı Birleşim algoritması kullanılmıştır. Standart ÜTB algoritmasına ilave edilen ek süreçler ile

algoritmada iyileştirmeler yapılmış ve GÜTB algoritması ile daha doğru ve güvenilir sonuçların elde edilmesi hedeflenmiştir [13]. Bu algoritma ile ortamdaki kenar bölgelerin konumları tespit edilmiş ve kenar noktalar birer yer gösterici olarak kullanılarak aynı zamanda ortamının noktasal olarak iki boyutlu haritası oluşturulmuştur.

3. Sonar algılayıcılarından elde edilen ölçümler ile yer göstericiler arasındaki eşleştirmeler Mahalanobis uzaklığı yaklaşımı ile "En Yakın Komşuluk" algoritması kullanılarak bireysel uyumluluk kriterine göre ilişkilendirilmesi hedeflenmiştir.

4. Kalman süzgeci uygulamalarında genellikle sistem ve ölçüm modelindeki gürültü varyansları sabit alınsa da [18], bu çalışmada varyanslar GKS algoritmasının her döngüsünde yeniden hesaplanmıştır. Bunun nedeni gürültü varyanslarının sabit seçilmesinin gerçekçi bir yaklaşım olmamasıdır. Diğer bir ifadeyle zamanla değişen robot konumuna bağlı olarak sistem ve ölçüm gürültü varyanslarının da zamana bağlı olarak değişkenlik göstereceği gerçeğidir.

Deneysel çalışmalar için Pioneer robotlar için hazırlanmış, ARIA arayüz programı ve MobilSim benzetimcisi kullanılmıştır. Bu programlar sayesinde robotun tanımlanan sanal ortam içinde engellere çarpmadan gezinimi sağlanmış ve yukarıda bahsedilen algoritmalar için gerekli ölçümler belirli sıklıkta örneklenerek bir dosyada depolanmıştır. Daha sonra Matlab programı ile yukarıdaki algoritmalar ve benzetimciden elde edilen ölçümler kullanılarak benzetim tabanlı uygulamalar gerçekleştirilmiştir.

Tezin, ikinci bölümünde haritalama tekniklerine açıklanmış ve konum belirleme problemlerinin sınıflanması yapılmıştır. Üçüncü bölümde, ortamın iki boyutlu düzlemde öznitelik tabanlı haritasını oluşturmak için kullanılan tekniklerden ve algoritmalarından bahsedilmiştir. Dördüncü bölümde EKBH probleminin çözümü olasılık modeller ile ilişkilendirilmiş, doğrusal ve GKS algoritmanın matematiksel ifadelerine yer verilmiştir. Beşinci bölümde deneysel çalışmalarda model olarak kullanılan Pioneer-3DX robot tanıtılmış ve robot algılayıcıların

oluşturduğu sistem modellenmiş ve kullanılacak GKS algoritması ile ilişkilendirilmiştir. Altıncı bölümde benzetim ortamında yapılan uygulamalara yer verilmiştir. Bu bölümde ortam içinde robotun engelleri çarpmadan gezinmesi için geliştirilen program, akış diyagramı ile açıklanmıştır. ÜTB algoritması kullanılarak ortamdaki kenar noktaların çıkartılması ile ilgili yapılan çalışmalara ve standart ÜTB algoritmasında tanımlanan iki ek süreç ile elde edilen sonuçlardaki iyileştirmelere değinilmiştir. Daha sonra başlangıç koşulları verilen GKS ve GÜTB algoritması beraber kullanılarak yapılan eş zamanlı konum belirleme ve haritalama uygulaması ile ilgili sonuçlar verilmiştir. Sonuç ve öneriler bölümünde yapılan deneysel çalışmaların sonuçları değerlendirilmiş ve önerilerde bulunulmuştur.

2. HARİTALAMA VE KONUM BELİRLEME TEKNİKLERİ

Robotun kendi konumunu belirleyebilmesi için iki farklı bilgiye ulaşması gerekir. Bunlardan ilki öncelikli veriler olarak adlandırılır. Bu bilgi, robotun kendisinden (ortam haritasına sahip olması gibi) elde ettiği verilerdir. Diğer veriler ise ortam verileridir. Bu bilgiler ise robotun hareketi sırasında elde ettiği sürüş (sürüş sistemini sağlayan motorlar) ve algılama (sensör ölçümleri) verileridir. Robot sürüş ve algılama verilerini bir arada kullanarak konumunu belirleyebilir. Robot konum belirleme problemi üç başlık altında incelenir :

1. Konum izleme
2. Bütünsel konum belirleme
3. Eş zamanlı konum belirleme ve haritalama

Robotun ortam haritasını çıkarması için öncelikle kendi konumu bilmesi gerekir. Bu bilgi bütünsel konumlandırma sisteminden (GPS) harici olarak alınabileceği gibi, robotun tekerleri üzerindeki yüksek çözünürlüklü hareket kodlayıcılarından da alınabilir. Daha sonra robot, ortamı tanımlamak için algılayıcılarından elde edeceği verileri kullanarak çevre haritasını oluşturur.

2.1 Haritalama İçin Kullanılan Algılayıcılar

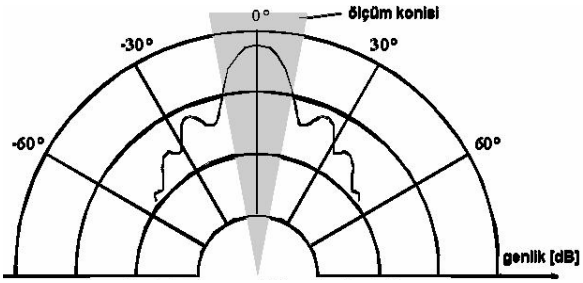
Ortam haritasını çıkarmak için robot uygulamalarında genel olarak kamere, lazer ve sonar algılayıcılar kullanılır.

Kamera ile harita çıkarma uygulamaları lazere ve sonar algılayıcılara göre çok daha ileri düzeydedir. İmge işleme algoritmalarının yüksek işlem yükü gerektirmesi nedeniyle gerçek zamanlı uygulamalarda hızlı ve pahalı sistemlerde kullanılır. Yetersiz ışık ve işlemsel yük en büyük dezavantajıdır.

Lazer algılayıcılar yüksek dalga boyundaki ışık ile çalışır. Mesafe ve nesne algılama sorunu açısından incelediğinde lazer algılayıcıların sonarlara göre çok daha etkin sonuçlar verdiği gözlenmektedir. En önemli dezavantajları pahalı

olmaları ve ışığın yayılma prensibinden dolayı saydam yüzeylerde doğru çalışmamalarıdır.

Sonar algılayıcılar ilk defa 1917 yılında kullanılmaya başlanmıştır. Sonar algılayıcılar sesin uçuş süresi özelliğini kullanırlar. Bu algılayıcılar ses dalgasının belirli bir noktaya gönderilip geri gelme süresine bağlı olarak mesafe değeri üretir. Bu sistemde birden fazla sefer ses paketleri yayımlanır ve yankının alındığı süre ölçülür. Bu süre uçuş zamanı olarak adlandırılır (T.O.F). Bu zamanın mesafelerin ölçümünde kullanılmasında ses hızının (340m/s 20 C^{o}'de) değişmediği yada çevresel sıcaklığa bağlı olarak ihmal edilebilir bir biçimde değiştiği varsayılır. Sonar algılayıcılar, lazer algılayıcılar göre daha ekonomiktir. Sonarla mesafe ölçümündeki en büyük dezavantaj nesnelerin yüzeyinden gerçekleşen yansıma ile ilgili problemdir. Yansıma yönü gelen ses dalgasını yüzeyle yaptığı açı ve yüzeyin şekli ile ilgilidir. Gerçek hayatta sonar algılayıcılar, sesin yayılımı özelliğini kullandıkları için, mesafe bilgisini koni şeklinde bir alandan elde ederler (şekil 2.1). Bu yüzden sonar ölçümleri merkez doğrultusundan alma yaklaşımı gerçekçi olmayacaktır. Bunun yerine yoğunluk dağılımındaki yanal lobların ihmal edilmesiyle 20^{o}-30^{o} açıklık aralığında, sonarın fiziksel yapısına bağlı olarak değişen, bir yay olarak modellenir. Bir çok dezavantajına rağmen ekonomik nedenlerden dolayı sonar algılayıcılar robot uygulamalarında oldukça fazla kullanılır. Bu tez çalışmasında ekonomik olmaları ve güvenilir mesafe ölçme yetenekleri sebebiyle sonar algılayıcılar kullanılmıştır.



Şekil 2.1 Ses ötesi algılayıcısının tipik yoğunluk dağılımı

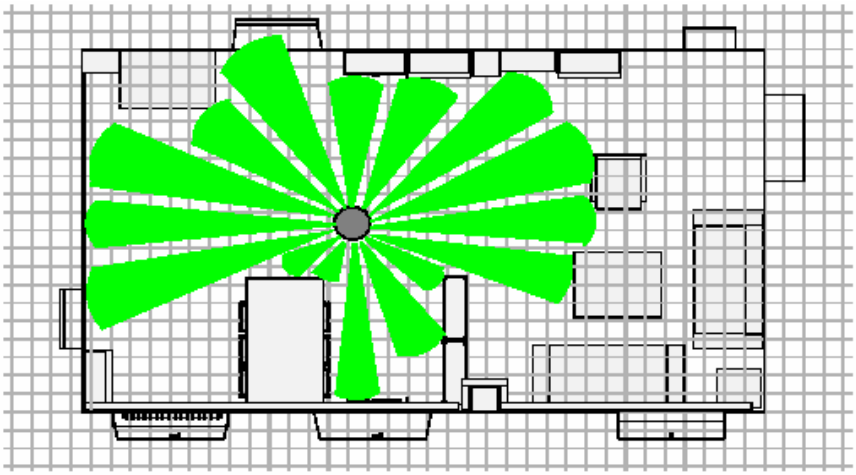
2.2 Haritalama Teknikleri

Haritalar, robot konum belirleme problemlerinin çözümü için çok önemlidir. Robot uygulamalarında en genel harita gösterimi iki şekilde yapılır.

1. Izgara Tabanlı Harita
2. Öznitelik Tabanlı Harita

2.2.1 Izgara Tabanlı Haritalama

Izgara tabanlı haritalar ortamın belirli oranlarda parçalara bölünmesiyle oluşur. Izgara içersindeki her bir hücre çevre hakkındaki bilgileri tutar. Bu bilgiler genellikle olasılıksal yada hücrenin dolu, boş veya keşfedilmemiş olduğunu gösteren karakterlerden oluşur. Izgara tabanlı haritanın oluşturulması ve güncellenmesi, robotun çevreyi gezinmesi sırasında sensörlerden topladığı ölçümlere göre yapılır. (Şekil 2.2)

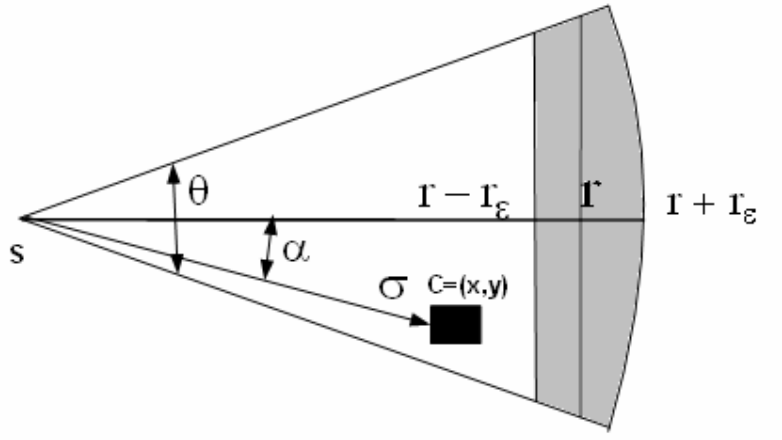


Şekil 2.2 Izgara tabanlı harita

Ortamın ızgara tabanlı haritasını çıkarmak için en çok kullanılan yöntem Izgara Doluluk tekniğidir [5]. Izgara doluluk tekniği ilk olarak Elfes (1987) tarafından Bayesian olasılık teoremi kullanılarak ızgara hücrelerinin dolu veya boş olma olasılıklarının hesaplanması ile ortaya çıkmıştır. Bu çalışmada ses ötesi

algılayıcılar kullanılarak yöntemin doğruluğu ispat edilmiştir. Bu yöntem başlangıçta boş veya belirsiz olarak tanımlanan hücreler ile oluşturulmuş haritanın, sonar ölçümler kullanılarak Bayesian olasılık teorileri ile güncellenmesi prensibine dayanmaktadır. Hücre (C) güncellemelerinin yapılabilmesi için gerekli olan sensör karakteristikleri ve diğer veriler aşağıda belirtilmiş olup sonar ölçümler şekil 2.3'de gösterildiği gibi birer yay olarak modellenmiştir.

- **r** sonar algılayıcıya dönen uzaklık mesafesi
- $\mathbf{r_{min}}$ sonar algılayıcı ile ölçülebilen en kısa mesafe
- $\mathbf{r_\varepsilon}$ ölçüm sapması
- **θ** sonar hüzme açısı
- **α** SC çizgisi ile ışının ana ekseni arasındaki açı
- $\mathbf{S=(x_s,y_s)}$ sonar sensör konumu
- **σ** C=(x,y) ile S=(x_s,y_s) noktaları arasındaki mesafe



Şekil 2.3 Sonar ışınının modellenmesi

Taranan hücrelerin dolu veya boş olma olasılıkları iki farklı ölçüm ile hesaplanır. p_E, hücrenin boş olma olasılığı, p_O ise hücrenin dolu olma olasılığını gösterir. Şekil 2.4'de bu olasılıklara ait dağılım fonksiyonları gösterilmiştir. Aşağıda p_E ve p_O olasılıklarının matematiksel ifadeleri yer almaktadır.

$$p_E(x,y) = p[\text{hücre}(x,y) \quad \text{boş}] = E_r(\sigma).E_a(\alpha) \tag{2.1}$$

$E_r(\sigma), E_a(\alpha)$ sırasıyla, uzaklık ve açısal bağımlılıkların olasılık yoğunluk fonksiyonlarıdır.

$$E_r(\sigma) = \begin{cases} 1-[(\sigma - r_{min})/(r - r_\varepsilon - r_{min})]^2 & \sigma \in [r_{min}, r - r_\varepsilon], \\ 0 \quad \text{Diger Durumlarda} \end{cases} \tag{2.2}$$

$$E_a(\theta) = 1-(2\alpha/\theta)^2, \qquad \alpha \in [-\frac{\theta}{2}, \frac{\theta}{2}] \tag{2.3}$$

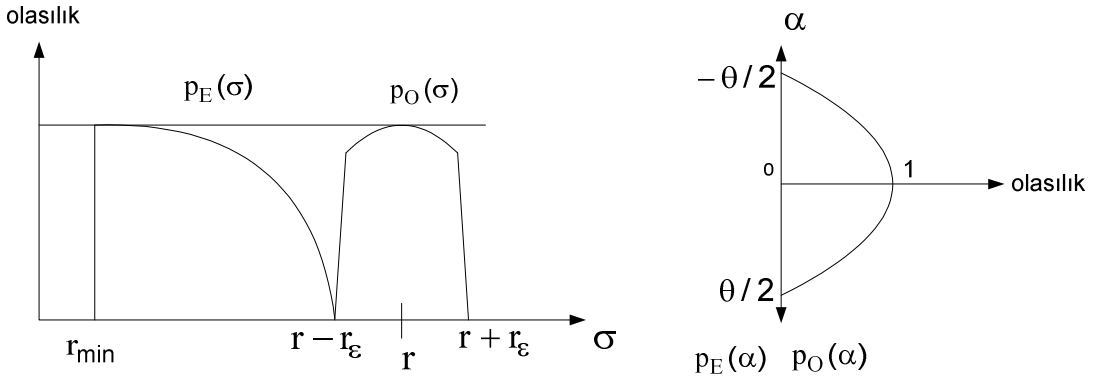
Aynı şekilde p_O tanımlanırsa,

$$p_O(x,y) = p[\text{hücre}(x,y) \quad \text{dolu}] = O_r(\sigma).O_a(\alpha) \tag{2.4}$$

$O_r(\sigma), O_a(\alpha)$ sırasıyla, uzaklık ve açısal bağımlılıkların olasılık yoğunluk fonksiyonlarıdır.

$$O_r(\sigma) = \begin{cases} 1-[(\sigma - r)/(r_\varepsilon)]^2, & \sigma \in [r - r_\varepsilon, r + r_\varepsilon] \\ 0 \quad \text{Diger Durumlarda} \end{cases} \tag{2.5}$$

$$O_a(\theta) = 1-(2\alpha/\theta)^2, \qquad \alpha \in [-\frac{\theta}{2}, \frac{\theta}{2}] \tag{2.6}$$



Şekil 2.4 p_E ve p_O olasılık yoğunluk fonksiyonları

Harita oluşturma süreci, tüm hücrelere başlangıçta sıfır değerinin verilmesi ile başlar. Her bir sonarın görüş alanı içindeki $[r_{min}, r - r_\varepsilon]$ aralığına düşen tüm hücrelerde (2.2), (2.3) eşitlikleri ve $[r - r_\varepsilon, r + r_\varepsilon]$ aralığına düşen hücrelerde ise (2.5), (2.6) eşitlikleri kullanılarak hücrenin boş ve dolu olma olasılık değerleri hesaplanır. Daha sonra Bayesian teoremleri ile hücre olasılık değerleri güncellenir.

Hücrenin boş olma olasılığı,

$$p_E(\text{hücre}) = p_E(\text{hücre}) + p_E(\text{ölçüm}) - p_E(\text{hücre}) * p_E(\text{ölçüm}) \quad (2.7)$$

hücrenin dolu olma olasılığı,

$$P_O(\text{hücre}) = p_O(\text{hücre}) + p_O(\text{ölçüm}) - p_O(\text{hücre}) * p_O(\text{ölçüm}) \quad (2.8)$$

(2.7) ve (2.8) eşitlikleri ile güncellenir.

Böylece ortam haritası her yeni sonar ölçüm için ilgili hücrelerin olasılıksal değerlerinin güncellenmesiyle oluşturulur. Güncellenmiş p_E ve p_O olasılık değerleri [-1,1] aralığında değişir. Bu değerler [0,1] aralığını düşecek şekilde dönüştürme işlemi uygulanır. Daha sonra belirlenen eşik seviyesi altında kalan hücreler boş, eşik seviyesinin üstünde kalan hücreler dolu olarak yorumlanır [19].

Ancak bu yöntemin iki önemli dezavantajı vardır:

- **Bellek İhtiyacı**

Harita boyutuna bağlı olarak, bellek gereksimi artar.

- **Hesaplama Karmaşıklığı**

Izgara güncellemesi yapılabilmesi için, hücre değerlerinin her ölçüm sonunda yeniden hesaplanması gerekmektedir. Bu süreci hızlandırmak adına, bazı ilintili

varsayımlar yapılmasına karşın, her bir hücre üzerindeki cebirsel hesaplamalar nedeniyle hesaplama karmaşıklığı bu yöntem için önemli bir dezavantajıdır.

Izgara tabanlı haritalar ise, konum izleme problemlerinin çözümünde sıkça tercih edilir [3,20].

2.2.2 Öznitelik Tabanlı Haritalama

Öznitelik tabanlı haritalar ortam içindeki nesnelerin şekli hakkında fikir veren ve bu nesnelerin konumlarına göre ortamın betimlendiği haritalardır. Bu öznitelikler konum belirleme problemlerinde birer yer gösterici (işaretçi) olarak kullanılır. Robot bu yer göstericilere bakarak konumunu belirler. Yer göstericiler yapay veya doğal yer göstericiler olmak üzere ikiye ayrılır.

2.2.2.1 Yapay yer göstericiler

Yer gösterici yerini belirtmek için kendisi ortama işaret gönderiyorsa aktif, göndermiyorsa pasif yer gösterici olarak adlandırılır.

Aktif yer göstericiler robota pozisyon bilgisi gönderen işaretçilerdir. Açık alan uygulamalarımda en çok bilinen ve başarılı aktif işaretçiler GPS'lerdir. Bu sistem, uydudan gönderilen radyo işaretlerinin uçuş süresi prensibine dayanır. Ancak kapalı alan uygulamalarında kısıtlı olarak kullanılır.

Pasif yer göstericiler robot tarafından algılanırlar. Robotun bulunduğu her ortama yerleştirilmesi gerekir. Robot algılayıcıları tarafından algılanabilecek her hangi bir nesne olabilir. Robot konumunu ortama sonradan ilave edilmiş bu pasif yer göstericilere bakarak belirler.

2.2.2.2 Doğal yer göstericiler

Robotun gezindiği ortam içinde kendiliğinden var olan nesnelerdir. Robot bu nesneleri algılayarak, sınıflayarak ve konumlarını tespit ederek haritalama yapar. Kapalı alanlar için kapı dikmesi, duvarlar, masa ayakları, kenar ve köşe noktalar

doğal yer göstericilere örnek olarak verilebilir. Ortamın iki boyutlu olarak tanımlandığı durumlarda, yer gösterici tabanlı haritalar genel olarak düz çizgi ve noktalardan oluşan bir yapıdadır. Düz çizgiler, ortamdaki düzlemsel bölgeleri (duvarlar, mobilya yüzeyleri); noktalar ise ortamdaki kenar ve köşeleri (masa ve koltuk ayakları, odanın kenar veya köşe kısımlarını) temsil eder.

2.3 Konum Belirleme Teknikleri

Gezgin robotlarda konum belirleme problemleri için kullanılan en bilindik yöntem olasılıksal durum kestirim yöntemleridir. Konum belirleme problemleri zorluk derecelerin göre alt başlıklar halinde aşağıda sıralanmıştır. Her bir problem için literatürde yer alan çözüm yöntemleri hakkında bilgi verilmiştir.

2.2.1 Konum İzleme

Konum belirleme problemlerinden başlangıç olarak ele alınanı konum izleme problemidir. Bu problemde robotun başlangıç konumu ve ortam haritası bilinerek çözüme gidilir. Şekil 2.5'de başlangıç konumu bilinen robotun hareketi geziniş sırasında izlenir ve son konumu belirlenmeye çalışılır. Bu arada ortamdaki doğal yer gösterici konumlarının da bilindiği varsayılır. Şekil 2.5'de bu yer göstericiler ortamdaki kenar noktaları temsil etmek üzere koyu siyah noktalar ile gösterilmiştir. Bu problemin çözümünü sağlayan yordamlar izleme veya bölgesel yordamlar olarak adlandırılır.

Konum izleme ve kestirim probleminin çözümünde genel olarak GKS 'den yararlanılır. Bu yaklaşım ile birçok uygulamada başarılı sonuçlar alınmıştır. Ancak GKS varsayımlarının yetersiz kaldığı durumlarda, özellikle yüksek dereceli doğrusal olmayan sistem modelleri ile çalışıldığında, başarısız sonuçlar da elde edilmiştir. Ancak alternatiflerinin az olması sebebiyle birçok uygulama için yine de tercih edilmektedir.

Gezgin robot konum izleme problemlerinde hareket ve ölçüm için doğrusal olmayan durum uzay modelleri çıkarılır.

$$x_{k+1}=f(x_k,u_k)+w_k \tag{2.9}$$

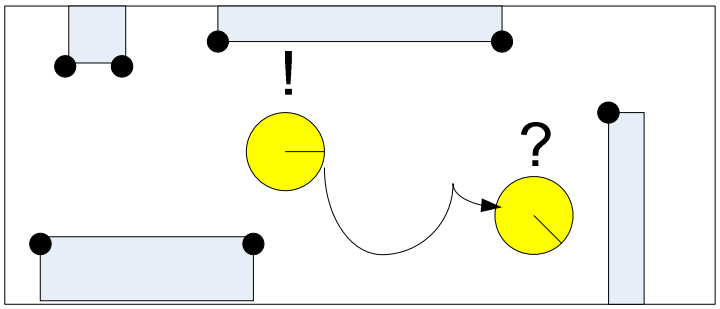
$$z_k=h(x_k)+v_k \tag{2.10}$$

Modeldeki durum vektörü x_k (2.11)'deki durum değişkenleri ile ifade edilir.

$$x_k = [x_r \quad y_r \quad \theta_r]^T \tag{2.11}$$

(x_r,y_r) robotun ortam içindeki konumunu, θ_r ise yönelim açısını göstermektedir. Denetim girdisi u_k ise hareket modeli için teker kodlayıcılarından aldığı bilgileri içerir. Sistem gürültüsü w_k ve ölçüm gürültüsü v_k birbirlerinden bağımsız, sıfır ortalamalı Gauss beyaz gürültü olarak alınır.

Doğrusal olmayan hareket $f(x_k,u_k)$ ve ölçüm $h(x_k)$ geçiş işlevlerinin doğrusallaştırılması GKS ile her döngüde yeniden hesaplanarak yapılır.

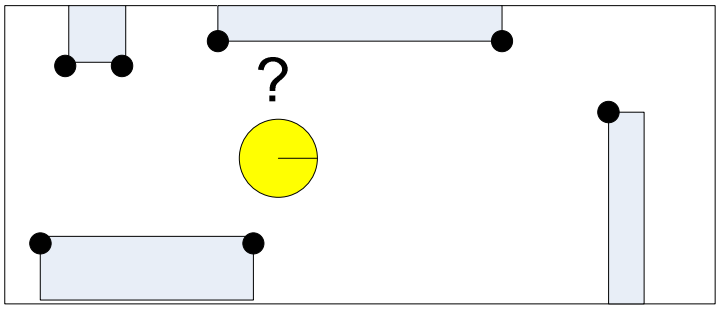


Şekil 2.5 Konum izleme problemi

2.2.2 Bütünsel Konum Belirleme

Uyanma yada bütünsel konum belirleme problemi, konum izleme problemine göre daha zordur. Şekil 2.6'da görüldüğü gibi robotun elinde kullanabileceği referans bir harita mevcuttur. Ancak harita üzerinde nerede konumlandığının bilgisine sahip değildir. Bu nedenle robot başlangıç durumunu bilmez. Robotun çözmesi gereken problem "Ben neredeyim?" sorusunun cevabını bulmaktır. Robotun konumunu belirlemesi için çabalaması gerekir. Dolayısıyla robot bulunabileceği konum hakkında çok farklı bilgilere sahip olabilir. Bu problemin

çözümünde kullanılan yordamlar bütünsel yordamlar olarak adlandırılır. Bütünsel ve bölgesel konum belirleme yöntemleri için geliştirilen birçok yönteme oranla Monte-Carlo parçacık süzgeci ile konum belirleme yöntemi daha etkili ve daha doğru sonuçlar ortaya koymuştur [16].



Şekil 2.6 Bütünsel konum belirleme problemi

2.2.3 Eş Zamanlı Konum Belirleme ve Haritalama

Şekil 2.7'de görüldüğü gibi bu problemde robot ortam hakkında ön bir bilgiye sahip değildir. Daha önceki konum belirleme problemlerinden farklı olarak yer gösterici konumları önceden bilinmez. Dolayısıyla robotun kendi başına ortam haritasını oluştururken bu harita üzerindeki konumunu da belirlemesi gerekmektedir.

EKBH problemi için genel olarak, konum izleme probleminde tanımlanan doğrusal olmayan durum uzay modelleri aynen kullanılarak GKS ile çözüm aranır.

$$x_{k+1}=f(x_k,u_k)+w_k \tag{2.12}$$

$$z_k=h(x_k)+v_k \tag{2.13}$$

Ancak konum izleme problemine göre en büyük fark EKBH'deki durum vektörünün boyutundaki değişikliktir. Konum belirleme problemlerinde referans haritanın önceden bilindiği ve doğru olduğu kabul edilir. Bu yüzden durum vektörü x_k sadece robot konum değişkenlerini ($x_k = [x_r \quad y_r \quad \theta_r]^T$) içerir. Ancak EKBH

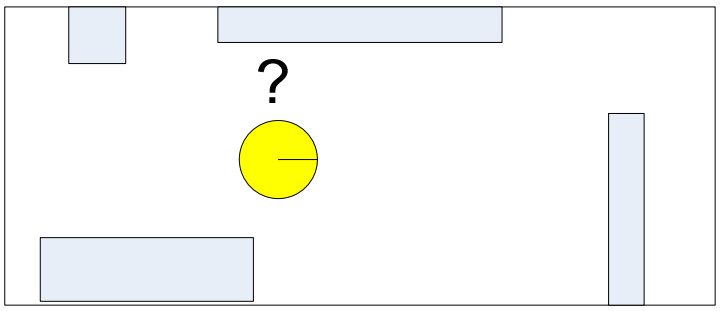
yaklaşımında, konumlama ve haritalama eş zamanlı olarak yapılacağı için durum vektörü hem robot konumunu hem de ortamdaki yer gösterici nesnelerin konumsal bilgilerini $[m_1,m_2,\ldots,m_N]^T$ kapsar. Ayrıca her yeni yer gösterici tespit edildiğinde durum vektörü içersinde tanımlanarak, durum vektörü ve durum hata kovaryans matrisleri genişletilir. Bu nedenle tanımlanan vektör ve matris boyutları dinamiktir.

$$x_k = \begin{bmatrix} x_r & y_r & \theta_r & m_1 & m_2 & . & . & . & m_N \end{bmatrix}^T \tag{2.14}$$

Genişlemiş durum kovaryans matrisi de (2.15) denklemindeki gibi olacaktır.

$$P_{k,k} = \begin{bmatrix} P_{rr} & P_{r1} & \ldots & P_{rN} \\ P_{1r} & P_{11} & \ldots & P_{1N} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ P_{Nr} & P_{N1} & \ldots & P_{NN} \end{bmatrix} \tag{2.15}$$

P_{rr}, P_{ri} ve P_{ii} (i=1,2,…,N) sırasıyla robotun-robota, robotun-yer göstericilere ve yer göstericilerin-yer göstericilere göre karşılıklı ilişkilerini ifade eder. Burada çözülmesi gereken bir diğer konu ise büyük boyuttaki dinamik matrislerden dolayı algoritmanın hesaplama karmaşıklığıdır. GKS algoritmasının EKBH uygulamalarındaki hesaplama karmaşıklığı $O(N)^3$'dür. (N: yer gösterici sayısı)



Şekil 2.7 Eş zamanlı konum belirleme ve haritalama problemi

Bu nedenle durum vektöründeki yer gösterici sayısının çok fazla olması (1000'in üzeri) gerçek zamanlı uygulamalara ciddi bir kısıtlama getirir.

EKBH problemi için literatürde yer alan bir diğer çözüm, Rao-Blackwellized parçacık süzgeci ile tanımlanır [16]. FastSLAM algoritması, Rao-Blackwellized parçacık süzgecinin en yaygın uygulamasıdır. FastSLAM algoritması konum izleme kestirimi yapmak için birden fazla parçacık süzgecini bir arada kullanır. Bu algoritmada haritalama problemi ise haritadaki her bir özniteliğin bağımsız problemler olarak ele alınmasıyla çözülür. FastSLAM algoritması harita öznitelik konumlarının kestirimi için birbirinden bağımsız GKS'ler kullanır. Her bir öznitelik için ağırlıklandırılmış bağımsız bir GKS kullanılması daha düşük boyutlarda matrisler ile harita kestirimi yapılmasına olanak sağlamaktadır. Bu sayede yer gösterici sayısı N'e bağlı olarak, standart GKS'deki hesaplama karmaşıklığı $O(N)^3$ 'den, FastSLAM algoritması ile $O(\log N)$'e düşürülmüştür.

FastSLAM yönteminin diğer bir avantajı ise doğrusal olmayan robot sistem modelleri ile kullanılabilir olmasıdır. GKS yönteminde doğrusal olmayan sistem modelleri doğrusallaştırılarak kullanılmaktadır. Özellikle yüksek dereceli doğrusal olmayan sistem modellerinin doğrusallaştırılmasıyla, GKS doğru kestirim yapamayabilir. Bu nedenle yüksek dereceli doğrusal olmayan sistem modelleri ile çalışıldığında parçacık süzgeci ve FastSLAM algoritması ilk tercih edilen yöntemdir [16].

Yukarıdaki avantajları nedeniyle FastSLAM gerçek zamanlı uygulamalarda daha çok tercih edilir [16].

3. YER GÖSTERİCİ ÇIKARMA TEKNİKLERİ

Öznitelik tabanlı haritalama, EKBH uygulamalarında daha çok tercih edilen haritalama tekniğidir. EKBH'nin yapılabilmesi için gezinim sırasında ortamdaki özniteliklerin çıkartılması ve çıkartılan bu özniteliklerin birer yer gösterici olarak kullanılması gerekir. EKBH için iki boyutlu düzlemde harita oluşturulması ve haritadaki özniteliklerin çıkartılması, robot algılayıcılarının (sonar, lazer mesafe ölçer) ortam üzerinden elde edecekleri verilerin işlenmesi ile olur. Birden fazla algılayıcı bir arada kullanılarak daha maliyetli (lazer mesafe ölçer+kamere) çözümlerde literatürde yer almaktadır [21, 22]. Ancak sadece mesafe verileri (sonar, lazer mesafe ölçer) kullanılarak da ortamdaki düzlemsel bölgelerin ve kenar-köşe noktaların konumsal gösterimiyle öznitelik tabanlı haritalar oluşturulabilir.

Bu durumda ortamı tanımlayan bu özniteliklerin mesafe ölçerler ile tespit edilmesi ve sınıflandırılması gerekir. Sonar algılayıcı tabanlı çalışmalarda Hough Dönüşümü (doğru belirleme) veya Üçgenleme Tabanlı Birleşim (kenar belirleme) teknikleri ile çözüme gidilir [3,14,17,19,23]. Her iki teknikte de sonar ölçümler, şekil 2.3'deki gibi sabit açılı(θ) birer yay olarak modellenmiştir.

3.1 Hough Dönüşüm Metodu

Hough dönüşümü, P.V.C Hough (Hough1962) tarafından geliştirilen,

$$g(x,c)=0,\ x \in X,\ \ c \in C \tag{3.1}$$

(3.1) eşitliği kullanılarak, $D \in X$ veri dizisinden eğri tanımlama yöntemidir. Burada x noktasal konum verilerinden oluşan bir vektör, c ise vektör katsayılarıdır. Hough dönüşüm tekniğinin en genel uygulaması nokta veri dizisinden doğru parçası tanımlama olarak karşımıza çıkar.

$$D=\{(x_1,y_1),(x_2,y_2),(x_3,y_3),\ldots(x_n,y_n)\} \tag{3.2}$$

Bu durumda g(x,c) fonksiyonu (3.11) eşitliğindeki gibi tanımlanır.

$$x\cos(\varphi) + y\sin(\varphi) - \rho = 0, \quad (x, y) \in \Re, \quad (\varphi, \rho) \in C \tag{3.3}$$

Eğer amaç doğru parçası bulmak ise, C için parametre uzayı (Hough uzayı) aşağıdaki gibidir.

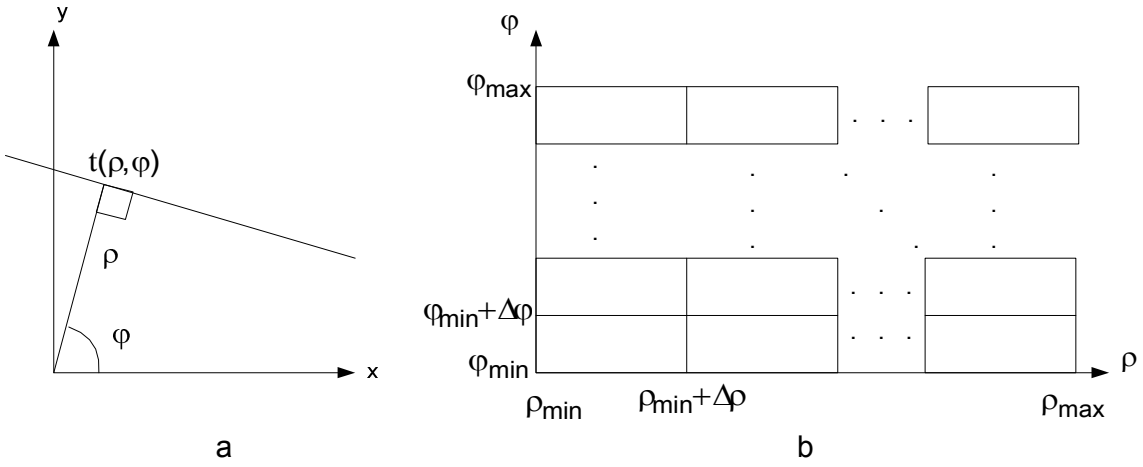
$$C = \{(\rho, \varphi) \mid \rho \in \Re, \quad -\frac{\pi}{2} \le \varphi \le \frac{\pi}{2}\} \tag{3.4}$$

Bir çok uygulamada ilgili parametre uzayı sınırlandırılmıştır. Bu durumda C yeniden tanımlanacak olursa,

$$C = \{(\rho, \varphi) \mid \rho_{min} \le \rho \le p_{max}, \quad \varphi_{min} \le \varphi \le \varphi_{max}\} \tag{3.5}$$

gibi olur.

Hough algoritması bir çeşit oylama yöntemidir. Bu oylama kesikli parametrik uzayda yani Hough uzayında yapılır. Hough uzayı tüm olası özniteliklerin konumlarını gösterir. Hough uzayında en çok oy alan hücre ilgili özniteliğin konumu ile ilgili polar koordinatları verir.



Şekil 3.1 a Doğru parçasının kartezyen koordinat sistemindeki gösterimi.
b Toplayıcı hücrelerin bulunduğu $\rho\varphi$ düzlemi (Hough uzayı).

Şekil 3.1 a'da kartezyen koordinat uzayında doğru parçasının gösterimi yer almaktadır. Bu tanımlama doğrunun orijine göre olan yönelim açısı (φ) ve uzaklık değeri (ρ) parametreleriyle yapılır. Şekil 3.1 b'de $\Delta\varphi$ ve $\Delta\rho$ Hough uzayını belirli miktarda hücrelere böler. Izgara yapıdaki uzayda her bir hücre toplayıcı (A_{ij}) olarak adlandırılır.

1: **procedure** HoughTransform (x,y,ρ,φ)

2: **for** $i \leftarrow 1, \frac{\rho_{max}-\rho_{min}}{\Delta\rho}$ **do**

for $j \leftarrow 1, \frac{\varphi_{max}-\varphi_{min}}{\Delta\varphi}$ **do**

3: $A_{ij} \leftarrow 0$

end for

4: **end for**

5: **for all** $(x_k, y_k) \in D$ **do**

6: **for** $\varphi \leftarrow \varphi_{min}$, $\Delta\varphi \rightarrow \varphi_{max}$ **do**

7: $\rho \leftarrow x_k \cos(\varphi) + y_k \sin(\varphi)$

8: **if** $\rho_{min} \le \rho \le \rho_{max}$ **then**

9: $A_{ij} \leftarrow A_{ij} + 1$

10: **end if**

11: **end for**

12: **end for**

13: **end procedure**

Şekil 3.2 Hough dönüşüm algoritması

Her toplayıcı sayısal bir değer tutar. Algoritma başlangıcında tüm toplayıcılar sıfırlanır ($A_{ij}=0 \quad i = 1,2,...,\frac{\rho_{max}-\rho_{min}}{\Delta\rho}, \quad j = 1,2,...,\frac{\varphi_{max}-\varphi_{min}}{\Delta\varphi}$). Şekil 3.2'deki

Hough dönüşüm algoritmasına göre D veri kümesine ait tüm noktalar sırasıyla kartezyen koordinat uzayından Hough uzayına aktarılarak A_{ij} hücre güncellemesi yapılır. Böylece tüm veri kümesi ele alındıktan sonra, Hough uzayındaki en yüksek A_{ijmax} değerine sahip hücre, doğru parçasına teğet olan polar koordinatın t(ρ, φ) yakınındaki bir değeri gösterir.

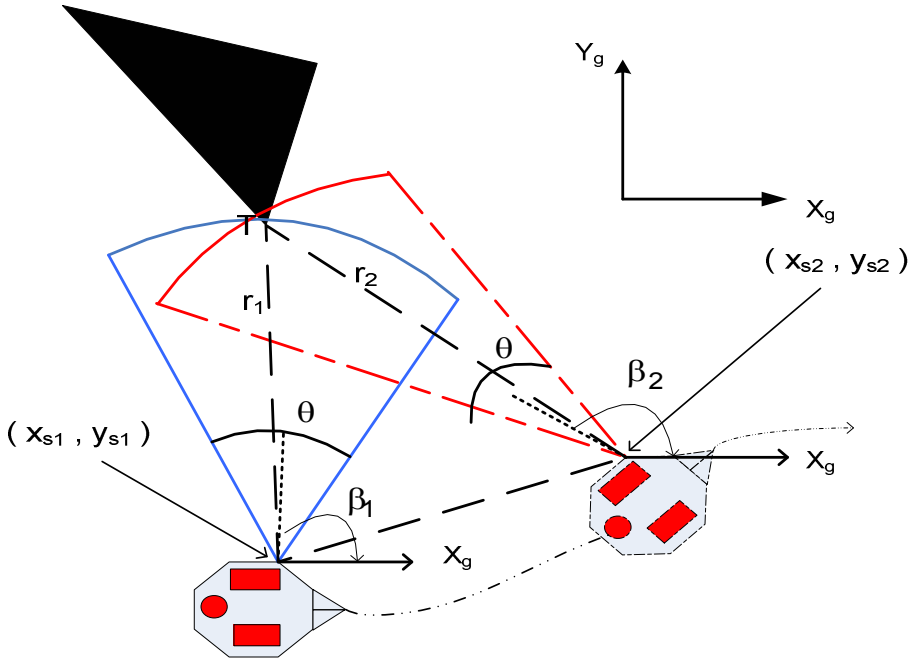
Hough dönüşüm algoritması, bir çok EKBH uygulamasında kullanılmıştır. [14]'e göre bu algoritma kullanılarak, ortamdaki hem düzlemsel bölgeler hem de köşe-kenar noktalar, sonar algılayıcılar kullanılarak tespit edilmiştir. Her bir veri kümesi robotun ortam içindeki 1.5 metrelik gezinimi sonunda elde ettiği sonar ölçümlerden oluşmuştur. $\Delta\varphi$ ve $\Delta\rho$ sırasıyla, 3.5^0 ve 4cm olarak seçilmiştir.

3.2 Üçgenleme Tabanlı Birleşim Metodu

Ses ötesi algılayıcılar ile ortam haritası oluşturmak için kullanılan yöntemlerden biri de Üçgenleme Tabanlı Birleşim metodudur. ÜTB metodu ortam içinde bulunan dikey kenarların (masa ayakları, kapı dikmesi gibi…) belirlenmesi ve konumlarının çıkartılması için geliştirilmiş bir yöntemdir.

Bu yöntemde sonar ölçümler sabit açılı yay olarak modellenir. Temel prensip aynı nesne üzerinden farklı konumlarda alınmış sonar ölçümler arasında kesişim noktaları bulmak ve kenar sınıflaması yapmaktır.

ÜTB metodunun Hough metodundan farkı ortam içindeki geometriksel nesnelerin direk tespit edebilmesidir. Herhangi bir dönüşüm işlemi uygulanmaz. Fakat ortamdaki düzlemleri tespit etmek için kullanılan bir yöntem değildir. Sadece kenar noktaları çıkarır [19,22]. Hough metodu ile benzer yanı ise algoritmanın oylama yöntemi ile kenar noktaları tespit ediyor olmasıdır.



Şekil 3.3 Üçgenleme yöntemi

Şekil 3.3'de farklı robot konumlarından alınan iki sonar ölçüm görülmektedir. Burada r algılayıcının ölçtüğü mesafeyi, β sonar merkezi ile bütünsel yatay düzlem arasında kalan açıyı gösterir. Bu açı sonar merkezinin bütünsel düzlem üzerindeki yönelim doğrultusunu diğer bir ifade ile görüş alanını tanımlar. θ ise yay açıklığını ifade eden açısal bir değerdir. Bu kullanılan sonar algılayıcıların fiziksel yapısına göre değişebilir [3]. Şekil 3.3'da iki ölçümün de $T(x_T,y_T)$ noktası üzerinden alındığı varsayılmıştır. Sadece tek bir sonar ölçüm ile bu konum tespit edilemez. Çünkü sonarın hüzme açıklığı (θ) nedeniyle yay üzerindeki herhangi bir yerden yansıma alınmış olabilir. Konumsal belirsizlik ancak ikinci bir ölçümün daha aynı nesne üzerinden alınmasıyla düzeltilir. Bundan dolayı bu yöntem üçgenleme tabanlı olarak adlandırılır.

Yayların kesişim noktası (T), her iki sonardan elde edilen bilgilerle (3.6) ve (3.7) eşitlikleri kullanılarak hesaplanır (şekil 3.4).

$$(x_T - x_{si})^2 + (y_T - y_{si})^2 = r_i^2, \quad i = 1, 2 \qquad (3.6)$$

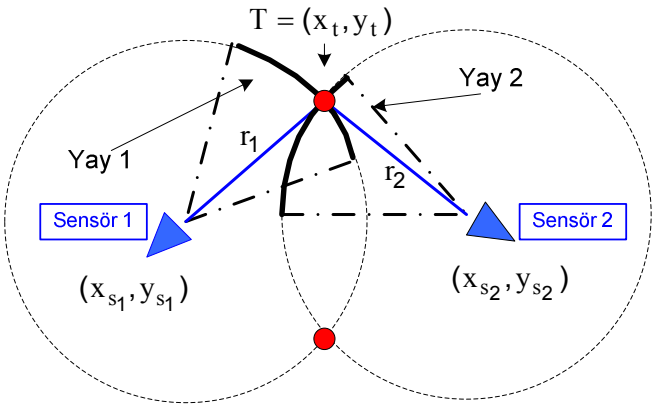
$$\arctan\left(\frac{y_T - y_{s_i}}{x_T - x_{s_i}}\right) \in \left[\beta_i - \frac{\theta}{2}, \beta_i + \frac{\theta}{2}\right] \quad i = 1, 2 \tag{3.7}$$

(3.6) eşitliğinin çözüm uzayı (3.8) ve (3.9) eşitliğinden çıkarılır.

$$\hat{x}_T = x_{s_2} - \frac{1}{d_s^2}\left(d_{x_s} d_r^2 \pm \left|d_{y_s}\right| \sqrt{r_2^2 d_s^2 - d_r^4}\right) \tag{3.8}$$

$$\hat{y}_T = y_{s_2} - \frac{1}{d_s^2}\left(d_{y_s} d_r^2 \pm \left|d_{x_s}\right| \sqrt{r_2^2 d_s^2 - d_r^4}\right) \tag{3.9}$$

$$\begin{aligned} d_{x_s} &= x_{s_1} - x_{s_2} \\ d_{y_s} &= y_{s_1} - y_{s_2} \\ d_s^2 &= d_{x_s}^2 + d_{y_s}^2 \\ d_r^2 &= \frac{r_1^2 - r_2^2 - d_s^2}{2} \end{aligned} \tag{3.10}$$



Şekil 3.4 Çember kesişim noktaları

(3.8) ve (3.9) eşitlikleri kullanıldığında T kesişim noktası için 4 farklı çözüm kümesi elde edilir(T_1, T_2, T_3, T_4). Bu çözümlerin (3.6) ve (3.7) eşitlikleri üzerinden sağlaması yapıldığında doğru kesişim noktası bulunur (T).

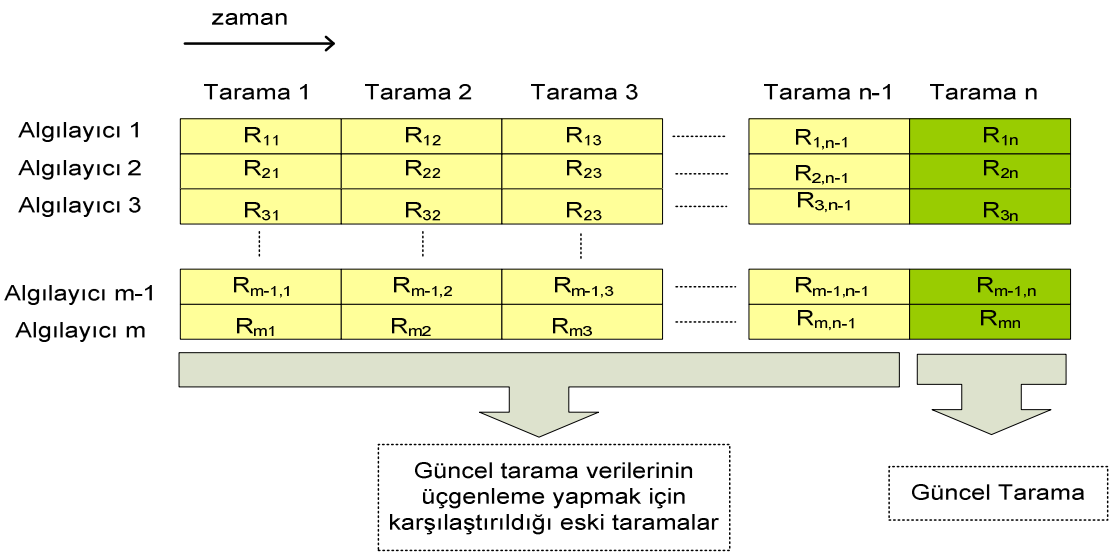
Böylece sonarın fiziksel kısıtlamaları nedeniyle konumsal belirsizlik üçgenleme yöntemi ile düzeltilir. Bu yöntem için önemli bir parametre de kesişim

noktalarının kaçar üçgenleme adımı sonunda bulunduğunu gösteren sayısal değerdir. Bu sayısal değer algoritmada n_t (number of triangulation) ile gösterilir.

ÜTB algoritmasını uygulamak için öncelikle robotun hareketi esnasında elde ettiği sonar verilerin tutulduğu bir veri tabanı oluşturulur. Bu veri tabanı hareketli pencere olarak da adlandırılır. Bu pencere şekil 3.5'te görüldüğü gibi m satır ve n sütundan oluşmaktadır. m kullanılan sonar algılayıcı sayısını ifade eder. n ise robot gezinimi sırasında yapmış olduğu sonar ölçümlerin toplamını gösterir. Pencerenin her bir hücresindeki R_{ij} (i=1,…,m ve j=1,…,n) veri paketi bütünsel düzlem üzerinde sonar yayları tanımlayan sonar algılayıcı bilgilerini içerir (3.11).

$$R_{ij}= (x_{sij}, y_{sij}, \beta_{ij}, r_{ij}) \qquad (3.11)$$

x_{sij} ve y_{sij} bütünsel düzlem üzerindeki algılayıcı konumu, β_{ij} sensör merkezinin bütünsel yatay eksen ile yapmış olduğu açıyı, r_{ij} ise algılayıcı ile engel arasındaki uzaklığı ifade eder. Bu bilgiler ile aslında ortam üzerinde merkez koordinatı (x_{sij},y_{sij}), yarıçapı r_{ij} ve görüş alanı ($\beta_{ij}+\frac{\theta}{2}, \beta_{ij}-\frac{\theta}{2}$) olan bir yay tanımlanmıştır (şekil 3.4).



Şekil 3.5 Sonar algılayıcı verilerinin tutulduğu hareketli pencere.

ÜTB algoritması, hareketli pencere içersinde tanımlanan sonar yaylar arasında kesişim noktaları tespit eder. Bu pencerenin en sağ sütunu en güncel sonar ölçümleri göstermektedir. Amaç bu güncel ölçümler ile daha önceden yapılan ölçümler arasında kesişim noktası yakalayabilmektir. Bunun içinde en sağdaki sütunun birinci satırındaki sonar ölçüm R_{1n}, diğer sütunlardaki her bir sonar ölçüm ile karşılaştırılır. Bu karşılaştırma sonucunda hesaplanan her kesişim noktası bir öncekinin üzerine eklenerek ortalaması alınır ve R_{1n} için kesişim sayacı bir artırılır. Başlangıçta kesişim sayısı sıfır olarak atanır ve her başarılı kesişim sonunda n_t bir artırılır ($n_t=n_t+1$). Diğer bir husus ise birden fazla kesişim olduğunda başlangıç varsayımına göre başarılı kesişimler arasındaki maksimum sapmanın ne kadar değiştiğidir. Bu değişim aslında alınan ölçümün kenar bölgeden mi yoksa duvar gibi düzlemsel bir bölgeden mi geldiğini göstermek için kullanılır. Diğer bir ifade ile kenar-düzlem sınıflaması bu sapma göz önünde bulundurularak yapılır.

ÜTB algoritması şekil 3.6'da yer almaktadır. Algoritma içindeki adımlar numaralandırılmış ve detayları bir sonraki sayfada açıklanmıştır.

1: **procedure** ÜTB (x_s, y_s, β_s, r_s)

2: **repeat**

3: **for** $i \leftarrow 1, m$ **do**

4: **if** $r_{in} < r_{max}$ **then**

5: $n_t \leftarrow 0,\ x_T \leftarrow x_{s_{in}} + r_{in}\cos(\beta_{in}),\ y_T \leftarrow y_{s_{in}} + r_{in}\sin(\beta_{in})$

6: $x_{min} \leftarrow x_T,\ x_{max} \leftarrow x_T,\ y_{min} \leftarrow y_T,\ y_{max} \leftarrow y_T$

7: **for** $j \leftarrow n-1, 1$ **do**

8: **for** $k \leftarrow 1, m$ **do**

9: **if** $r_{kj} < r_{max}$ **then**

10: **if** $(x_T, y_T) \in R_{kj}$ **then**

11: $r_e = \sqrt{(x_T - x_{s_{kj}})^2 - (y_T - y_{s_{kj}})^2}$

12: **if** $|r_e - r_{kj}| < \frac{d_1}{n_t + 1}$ **then**

13: **if** Kesişim var mı? (çıkış: $x^{üçg}, y^{üçg}$) **then**

14: $x_T \leftarrow \frac{n_t x_T + x^{üçg}}{n_t + 1}$

15: $y_T \leftarrow \frac{n_t y_T + y^{üçg}}{n_t + 1}$

16: $n_t \leftarrow n_t + 1$

17: **if** $x^{üçg} < x_{min}$ **then** $x_{min} \leftarrow x^{üçg}$ **end if**

18: **if** $x^{üçg} > x_{max}$ **then** $x_{max} \leftarrow x^{üçg}$ **end if**

19: **if** $y^{üçg} < y_{min}$ **then** $y_{min} \leftarrow y^{üçg}$ **end if**

20: **if** $y^{üçg} > y_{max}$ **then** $y_{max} \leftarrow y^{üçg}$ **end if**

21: **end if**

22: **end if**

23: **end if**

24: **end if**

25: **end for**

26: **end for**

27: **end if**

Şekil 3.6 ÜTB algoritması

```
28:          if n_t ≥ 1 then
29:               if (x_max − x_min) + (y_max − y_min) > d_2  then
30:                    n_t = −n_t
31:               end if
32:          end if
33:     end for
34:     if Yeni ölçümler geldi mi? then
35:          Pencere sütunlarını bir sola kaydır, do
36:          Güncel ölçümleri pencerenin en sağ sütuna yerleştir, do
37:     end if
38:   until gezinim bitene kadar
39: end procedure
```

Şekil 3.6 Devam ediyor

3. adım: Pencerenin en sağ sütununda yer alan her bir sonar veri paketi (R_{ij}) için algoritma kesişim noktası arar. Yukarıdaki algoritmada i=1 için adımlar detaylı olarak açıklanmıştır.

4. adım: r_{1n} mesafe değerinin, maksimum ölçülebilen mesafeden küçük olması şartı aranır. Maksimum ölçülebilen mesafe sonarın fiziksel özelliğine bağlı olarak değişebilir.

5. adım: n_t, x_T, y_T değişkenlerine başlangıç koşulları atanır. Başlangıç koşulu olarak n_t = 0'dır. Bu koşul yay üzerinde kesişim noktası bulunmadığı durumdur. Bu durumda yankının sonar merkez doğrultusundan geldiği kabul edilir.

6. adım: Üçgenlemeler arasındaki maksimum sapmayı hesaplamak için başlangıç koşulu atanır.

7-8. adımlar: (n-1)'nci sütundan 1'nci sütuna (sağdan-sola doğru) tüm hücreler gezilecek şekilde döngü yapılır.

9-12. adımlar: İlk olarak 4. adımdaki koşulu r_{kj}'nin de sağlayıp sağlamadığı kontrol edilir. Konum kestirimi yapılan T (x_T, y_T) noktasının R_{kj} yayının görüş alanı içinde olup olmadığına bakılır. Bunun için (3.7) eşitliği kullanılır. Son olarak beklenen mesafe r_e ile gerçek mesafe r_{kj} arasındaki fark hesaplanır. Bu farkın d_1 / (n_t +1) 'den küçük olması beklenir. İzin verilen farkın n_t arttıkça ,yada diğer bir ifade ile kesişim sayısı arttıkça azaldığı görülür.

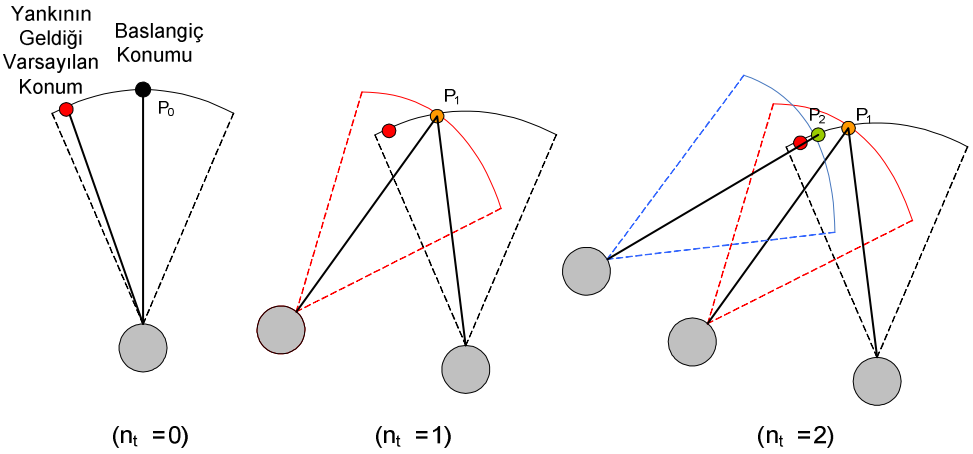
13. adım: Eğer algoritmada bu adıma gelindiyse, büyük olasılıkla R_{1n}, R_{kj} ölçümleri aynı nesne üzerinden alınmıştır. (3.6) ve (3.7) eşitlikleri kullanılarak yaylar arasındaki kesişim noktaları bulunur.

14-16. adımlar: Bu adımda 13. adımdan gelen her (x^{tri},y^{tri}) kesişim konumu ile yankının geldiği noktanın kestirimi yinelemeli olarak güncellenir. Böylece engelin (nesnenin) konumunu kestirilmeye çalışılmış olur. Her bir üçgenleme işlemi sonunda üçgenleme sayacı (n_t) bir artırılır.

17-20. adımlar: Başarılı üçgenlemeler arasında maksimum sapma güncellemesi yapılır.

28-36. adımlar: Başarılı üçgenlemeler arasındaki maksimum sapma belirlenen eşik değerinin üzerinde çıkarsa bu noktanın kenar ifade etmediği, büyük olasılıkla düzlem üzerinde olduğu sonucuna varılır. Bu sınıflamayı yapmak için yayların kesişim noktaları arasındaki maksimum sapma, eşik değerini aşarsa n_t işaret değiştirir ve negatif değer alır. Negatif değerli ($-n_t$) noktalar ortamdaki düzlemsel bölgeleri ifade etmektedir. Pozitif yüksek kesişim sayısına (n_t) sahip noktalar ortamdaki kenar bölgeleri temsil eder. Artık R_{1n} için pencere içindeki tüm hücreler gezilmiştir. En sağ sütundaki en son ölçüm R_{16n} için de aynı işlemler tekrarlandıktan sonra yeni tarama ölçümlerinin pencereye alınması için sütunlar sola kaydırılır. En sol sütundaki tarama pencere dışına atılırken, yeni gelen güncel tarama en sağ sütundaki yerini alır. Böylece hareketli pencere güncellemesi de yapılmış olur.

Şekil 3.7'de soldaki varsayım, yankının algılayıcının merkez doğrultusundan geldiği varsayılan başlangıç koşuludur $P_0=T_0=[\ x_t\ ,\ y_t\ ]$, ($n_t$ =0). Ortadaki varsayım yaylar arasında tek bir kesişim koşulunun sağlandığı durumdur $P_1=T_1=[\ x^{tri},\ y^{tri}\ ]$, ($n_t$ =1). Sağdaki varsayım ise iki kesişim koşulunun sağlandığı durumu göstermektedir. Böylece yankının geldiği konum iki kesişim sonunda algoritmanın 14. ve 15. adımdaki eşitliklere göre $T_2=(\ P_1 + P_2\ ) / 2$ olarak bulunur ve yankının geldiği gerçek konuma yaklaşmış olur.



Şekil 3.7 Üçgenleme varsayımları

Kenarlar, başarılı üçgenlemeler arasındaki maksimum sapmanın belirlenen eşik değerinin altında kaldığı ve bu başarılı üçgenleme sayısının (n_t) yine belirlenen eşik seviyesinin üstünde olduğu durumlarda tespit edilir. Bu eşik değerler deneysel olarak belirlenebilir. [3]' de bu değerler 0.1m ve 5 olarak seçilmiştir.

4. OLASILIKSAL EŞ ZAMANLI KONUM BELİRLEME VE HARİTALAMA YAKLAŞIMI

EKBH'de en genel çözüm genişletilmiş Kalman süzgeci ile yapılır. Bu süzgeç Bayesian tabanlı durum uzayı kestirim yordamıdır.

Bayesian teoremi, bir durumun oluş olasılığının durum hakkında ek bilgi veya bilgiler edinilmesi halinde nasıl değişeceğini gösteren bir teoremdir. Bu teoremde kullanılan iki önemli olasılık yaklaşımı vardır. Bunlar bileşik ve koşullu olasılıklardır.

Herhangi iki olayın bileşik olasılığı, bu iki olayın bir arada olma olasılığıdır ve P(X,Y) veya $P(X \cap Y)$ olarak gösterilir. Eğer X ve Y birbirinden bağımsız olaylar ise ($X \cap Y = \varnothing$), bileşik olasılıkları P(X,Y)=P(X)P(Y)'e eşit olur.

Koşullu olasılık ise, bir olayın diğer olayın gerçekleştiği durumda olma olasılığı olarak tanımlanır. $P(X|Y)$ Y olayı gerçekleştiğinde X olayın olma olasılığını gösterir. Eğer bu iki olay birbirinden bağımsız ise $P(X|Y)$=P(X) olur. X ve Y bağımlı olaylarsa bileşik olasılıkları, koşullu olasılık cinsinden yazılabilir. Bu durumda

$$P(X,Y)= P(X|Y)\,P(Y) \text{ veya } P(X,Y)= P(Y|X)\,P(X) \tag{4.1}$$

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)},\ P(Y) \neq 0 \tag{4.2}$$

olarak ifade edilir.

(4.2)'de payda, X değişkeni üzerinden Y'nin marjinal olasılığı cinsinden yazılırsa,

$$P(X_j|Y) = \frac{P(Y|X_j)P(X_j)}{\sum_{i=0}^{n} P(Y|X_i)P(X_i)} \tag{4.3}$$

gibi ifade edilir.

(4.3)'deki Bayesian kuralı sürekli olasılık yoğunluk fonksiyonları için de tanımlanabilir.

Bu durumda x,y rassal değişkenlerinin f(x) ve f(y) yoğunluk fonksiyonları cinsinden koşullu olasılığı,

$$f(x|y) = \frac{f(y|x)f(x)}{\int_{-\infty}^{\infty} f(y|x)f(x)dx} \tag{4.4}$$

gibi olur.

(4.3) eşitliğindeki $P(X_j)$, X olayının önsel olasılığı olarak adlandırılır. Önsel olasılık rassal bir değişkenin marjinal olasılığıdır. Marjinal olasılık ise birbiri ile bağımlı iki olaydan herhangi birinin koşulsuz olma olasılığı olarak tanımlanır. Bayesian kuralları uygulandıktan sonra koşullu olasılığa sonraki olasılık da denir . Sensör ölçümleri ile çevreden elde edeceği bilgiye göre robot konum olasılığının hesaplanmasında Bayesian kurallarından yararlanılır.

4.1 Özyinelemeli Bayesian Süzgeçlemesi

Markov zinciri sürecinin her bir adımında Bayesian kuramları kullanılarak durum güncellemesi yapılmasına özyinelemeli Bayesian süzgeçlemesi denir. Markov zinciri sürecinde güncel durum sadece önceki duruma bağlıdır.

$$p(x_{t+1}|x_0, x_1, ..., x_{t-1}, x_t) = p(x_{t+1}|x_t) \tag{4.5}$$

Markov zinciri varsayımının sonucunda t+1 anındaki değişkenin olasılığı aşağıdaki gibi hesaplanabilir.

$$p(x_{t+1}) = \int p(x_{t+1}|x_t) \;\; p(x_t) \;\; dx_t \tag{4.6}$$

Özyinelemeli Bayesian süzgeçlemesi, sensör ölçümleri veya daha başka veriler kullanılarak rassal bir durum vektörünün kestirimi ile ilgili çözüm sunar.

Temelde bu yöntem ile koşullu bir durumun sonraki olasılık dağılım fonksiyonu kestirilmeye çalışılır. Robot konum belirleme problemlerinde, robotun ortam içinde gezinirken çevre üzerinden yaptığı ölçümlere göre konumu kestirilebilir. Bu durumda, robot konumunun koşullu olasılığı,

$$p(s_t \left| z^t, u^t \right.), \quad z^t = \{z_1, z_2, ..., z_t\}, \quad u^t = \{u_1, u_2, ..., u_t\} \tag{4.7}$$

gibi tanımlanabilir. Burada s_t robotun t anındaki konumunu, z^t t zamanına kadar yapılan ölçümleri, u^t ise yine t anına kadarki denetim girdilerini gösterir. Bayesian kuralları kullanılarak (4.7) eşitliği aşağıdaki gibi tekrar tanımlanabilir.

$$p(s_t \left| z^t, u^t \right.) = \frac{p(z_t \left| s_t, z^{t-1}, u^t \right.) \;\; p(s_t \left| z^{t-1}, u^t \right.)}{\int p(z_t \left| s_t, z^{t-1}, u^t \right.) \;\; p(s_t \left| z^{t-1}, u^t \right.) \;\; ds_t} \tag{4.8}$$

$$= \eta^{-1} p(z_t \left| s_t, z^{t-1}, u^t \right.) \;\; p(s_t \left| z^{t-1}, u^t \right.) \tag{4.9}$$

$\eta^{-1} = \int p(z_t \left| s_t, z^{t-1}, u^t \right.) \;\; p(s_t \left| z^{t-1}, u^t \right.) \;\; ds_t$ normalizasyon katsayısı olarak gösterilir.

(4.9) eşitliğindeki ilk terim ölçüm modelini göstermektedir. Eğer ölçümlerin birbirlerinden bağımsız ve sadece robot konumuna bağlı olduğu varsayılırsa ifade (4.10)'daki gibi sadeleştirilebilir.

$$p(z_t \left| s_t, z^{t-1}, u^t \right.) = p(z_t \left| s_t \right.) \tag{4.10}$$

(4.9) eşitliğindeki ikinci terim ise robot konumunun t anına kadarki ölçüm ve denetim girdilerine göre koşullu (önsel) olasılığını gösterir. Eğer robot konumunun ölçümlerden bağımsız olduğu kabul edilirse, ifade aşağıdaki gibi sadeleştirilebilir.

$$p(s_t \left| z^{t-1}, u^t \right.) = p(s_t \left| u^t \right.) \tag{4.11}$$

Bu durumda (4.9) eşitliğinin sadeleştirilmiş şekli yeniden yazılırsa,

$$p(s_t \left| z^t, u^t \right.) = \eta \underbrace{p(z_t \left| s_t \right.)}_{\text{Ölçüm Modeli}} \underbrace{p(s_t \left| u^t \right.)}_{\text{Robot Konumu}} \tag{4.12}$$

gibi olur.

Sonuç olarak Markov varsayımları kullanılarak robotun önsel konum olasılığı robotun bir önceki konumuna bağlı olarak ifade edilirse, (4.13)'deki genel ifade elde edilir.

$$p(s_t \left| z^t, u^t \right.) = \eta \underbrace{p(z_t \left| s_t \right.)}_{\text{Ölçüm Modeli}} \int \underbrace{p(s_t \left| s_{t-1}, u_t \right.)}_{\text{Hareket Modeli}} p(s_t \left| u^{t-1} \right.) \, ds_{t-1} \tag{4.13}$$

Özyinelemeli Bayesian süzgeci ile durum kestirimi iki adımda hesaplanır. Bunlardan ilki sensör ölçümleri kullanılarak önsel olasılığın belirlemesi ile başlar. Bu adım öngörü adımı olarak da tanımlanır. Öngörü safhası, (4.14) eşitliğinde tanımlanan $p^-(s_t \left| u^t \right.)$'nin olasılık yoğunluk fonksiyonunun kestirimi ile yapılır.

$$p^-(s_t \left| u^t \right.) = \int p(s_t \left| s_{t-1}, u_t \right.) \, p(s_t \left| u^{t-1} \right.) \, ds_{t-1} \tag{4.14}$$

(4.14) eşitliği sürecin öngörüsünü tanımlar. İntegral içindeki ilk terim $p(s_t \left| s_{t-1}, u_t \right.)$ sistemin hareket modelini gösterir. Bu model doğrusal veya doğrusal olmayan fonksiyonlar ile tanımlanabilir. Doğrusal olmayan hareket modellerinde denetim girdileri açısal ve doğrusal hız değişimleridir. İntegral içindeki ikinci terim

$p(s_t|u^{t-1})$ ise t-1 anına kadarki denetim girdileriyle robotun bir önceki konumda olma olasılığını gösterir.

Özyinelime Bayesian süzgeçlemesi için ikinci adım, ölçümlerin öngörüye katılarak durum kestiriminin yapıldığı safhadır. Bu adım ölçüm güncellemesi olarak da adlandırılır ve (4.15) deki gibi tanımlanır.

$$p^+(s_t|u^t) \leftarrow p(z_t|s_t)\ \ p^-(s_t|u^t) \tag{4.15}$$

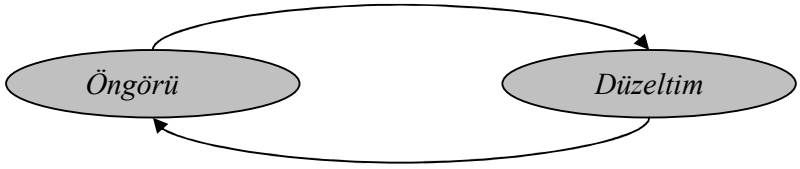
Böylece öngörülen olasılık fonksiyonu $p^-(s_t|u^t)$ ve ölçüm verileri kullanılarak, t anındaki durum kestirimi yapılmış olur.

Özyinelemeli Bayesian süzgeçlemesi yaklaşımıyla durum uzay kestirimi yapmak için kullanılan en iyi yöntem Kalman süzgecidir.

4.2 Kalman Süzgeci

Bayesian süzgecinin en genel uygulaması Kalman'dır. Doğrusal durum uzayı modelleri için yinelemeli veri işleyen sayısal bir süzgeçtir. Kalman süzgecinde tüm olasılık yoğunluk fonksiyonları Gauss dağılımı olarak modellenebilir. Bu sayede olasılık yoğunluk fonksiyonu kestirimi daha kolay bir hal alır. Çünkü Gauss olasılık dağılımı sadece ortalama (μ) ve varyans (σ^2) değerleri ile tanımlanabilir.

Kalman Süzgeci öngörü-düzeltim yapısının özyineli olarak işletilmesi ile durum kestirimini gerçekleştirir. Bu özyineli yapı, inanış işlevinin (durum) önceden öngörülüp daha sonra sistemden algılanan verilerle, öngörülen durumun düzeltilmesi işlemlerinin tekrarlanması ile gerçekleşir (Şekil 4.1).

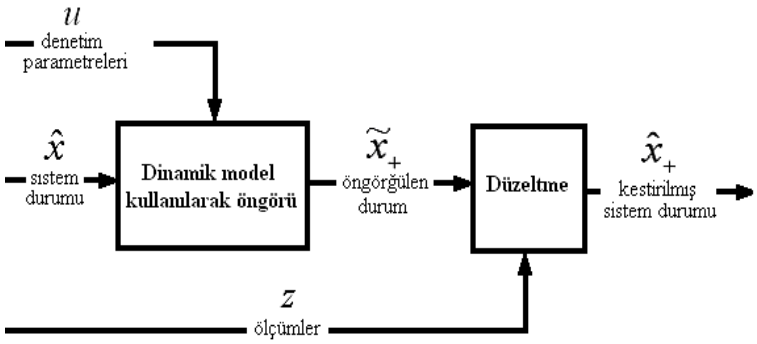


Şekil 4.1 Kalman Süzgeci Döngüsü

Kalman süzgecinin kullanım amacı, durum kestiriminin yapılarak doğru veya olması gereken durumun elde edilmesidir. Eğer durum, zamanla değişen bir yapıdaysa Kalman süzgecinin doğru kestirim yapması zorlaşır. Doğru durum kestirimini zorlaştıran başka bir etken de sistem değişkenlerinin doğrudan gözlemlenemiyor olmasıdır. Bu durumda Kalman süzgeci kullanılarak algılayıcılardan gelen veriler ile sistem değişkenlerinin gözlenmesi sağlanabilir. Algılayıcılardan gelen verilerin gürültülü olacağı da göz önünde bulundurulduğunda durum kestiriminin yapılması daha zor bir probleme dönüşür. Ancak Kalman süzgeci bu olumsuz etkilere karşın durum kestirimini en iyi şekilde gerçekleştirir.

4.2.1 Öngörü ve Düzeltim

Kalman süzgecinin durum kestirimi yapabilmesi için öncelikle bir önceki durumun öngörüsü hesaplamalı, daha sonra ölçümler ile bir sonraki durum kestiriminin belirlenmesi gerekmektedir.



Şekil 4.2 Standart model-tabanlı Kalman Süzgeci tasarım şekli

Şekil 4.2'deki dinamik model, bir sonraki durumun $\tilde{x}_+$ öngörülmesinde, ölçümler ise bu öngörüyü düzeltmede kullanılmıştır.

4.2.2 Dinamik Sistem Modelleri

Kalman süzgecinde sistem dinamikleri, bir önceki durumun doğrusal dönüşümü ile bulunur. Doğrusal dinamik sistem modelleri (4.16) eşitliğinde olduğu gibi hareket modeli ve (4.17) eşitliğinde olduğu gibi ölçüm modeli olarak tanımlanabilir.

$$\hat{x}_k = A.\hat{x}_{k-1} + B.u_{k-1} + w_{k-1} \tag{4.16}$$

$$z_k = H.x_k + v_k \tag{4.17}$$

Durum gürültülü ölçümler ile belirlenmeye çalışıldığında, sistemden ve ölçümden gelen gürültü de modele eklenmelidir. Doğrusal kalman süzgecinde bu model sistemin o anki durumu ve algılayıcı kaynaklı gürültüsü ile tanımlanır.

Yukarıda belirtilen her iki modelde de gürültüler birbirinden bağımsız sıfır ortalamalı Gauss olasılıklı olduğu kabul edilir.

$$w_k \sim N(0,Q) \quad , \quad v_k \sim N(0,R) \tag{4.18}$$

(4.18)'de belirtilen Q ve R sıfır ortalamalı sistem ve ölçüm gürültülerinin kovaryanslarını belirtmektedir. Kalman Süzgeci dinamik sistemlerin denetimi için kullanıldığında durum kestirimi amaçlıdır. Sistemin denetimi sırasında sistem hakkında alınabilecek en fazla bilginin elde edilmesi durum kestiriminin daha doğru yapılmasını sağlar. Sistem hakkında her bilginin elde edilmesi bazen mümkün değildir. Kalman Süzgeci, elde edilemeyen verilerin diğer elde ettiği veriler aracılığıyla kestirimini yaparak durum analizini gerçekleştirir.

4.2.3 Kalman Süzgeci Algoritması

Kalman süzgeci algoritması üç temel adımdan oluşmaktadır.

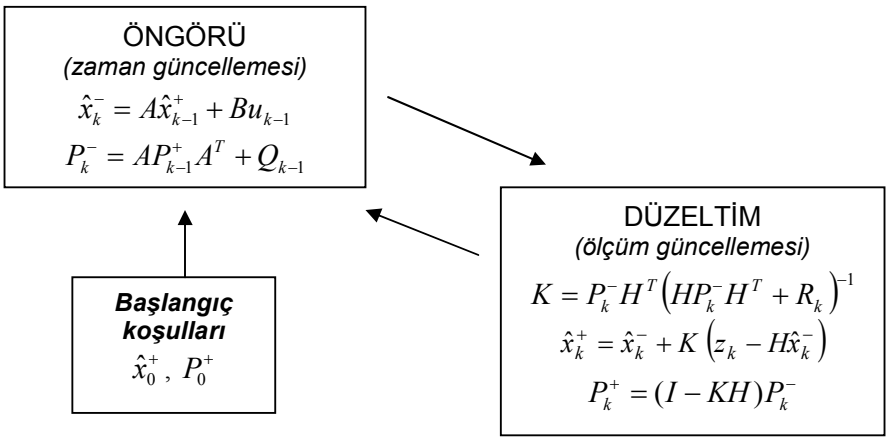
- Başlangıç koşulu

- Öngörü
- Düzeltim

Kalman algoritması özyinelemeli olarak çalıştığı için başlangıç koşulunun belirlenmesi gerekmektedir. Başlangıç durumu $\hat{x}_0^+$ ve başlangıç durumuna ait belirsizlik P_0^+ verilmelidir.

Öngörünün yapılabilmesi için bir önceki durumun öngörüsü $\hat{x}_k^-$ ve bu öngörüye ait belirsizlik P_k^- hesaplanmalıdır. Öngörü bölümü zaman güncellemesinin yapıldığı bölümdür.

Düzeltimin yapılabilmesi için bir sonraki durumun kestirimi $\hat{x}_k^+$ ve düzeltimdeki belirsizlik P_k^+ hesaplanmalıdır. Böylece bir sonraki durumun kestirimi gerçekleştirilmiş olur. Düzeltim sonucu elde edilmiş bir sonraki inanış işlevi ($\hat{x}_k^+, P_k^+$), bir zaman sonra bir önceki inanış işlevinin ($\hat{x}_k^-, P_k^-$) yerini alır. Bu özyinelemeli algoritma şekil 4.3'de gösterilmiştir.



Şekil 4.3 Doğrusal Kalman Süzgeci Algoritması

Sonuç olarak, zaman güncellemesinde sistem durumunun ve belirsizliğinin öngörüsü yapılır. Ölçüm güncellemesiyle de sistemden alınan ölçümler kullanılarak öngörü düzeltilir, sistem durumunun ve belirsizliğinin k anındaki kestirimi yapılmış olur. Çizelge 4.1'de doğrusal Kalman süzgeci algoritmasındaki simgeler, matris boyutları ve tanımları yer almaktadır.

Çizelge 4.1 Kalman süzgeci algoritmasında kullanılan simgeler ve açıklamaları

Simge	Boyut	Tanım
$\hat{x}_k$	n x 1	Durum değişkenleri vektörü
z_k	m x 1	Ölçüm vektörü
A	n x n	Durumun zamana göre dönüşüm matrisi
U_k	j x 1	Denetim girdileri
B	n x j	Denetim girdileri için dönüşüm matrisi
P_k	n x n	Durum hata kovaryans matrisi
Q	n x n	Sistem gürültüsü kovaryans matrisi
R	m x m	Ölçüm gürültüsü kovaryans matrisi
K	n x m	Kalman kazanç matrisi
H	m x n	Ölçümler zaman göre dönüşüm matrisi
I	n x n	Birim matris
$\hat{x}_k^-$	n x 1	Öngörülen durum vektörü
P_k^-	n x n	Öngörülen durum kovaryans matrisi

4.3 Genişletilmiş Kalman Süzgeci

Kalman süzgeci doğrusal olmayan sistemlerin kestiriminde de kullanılabilir. Bunun için bazı tanımlamalar yapılmalıdır. Gerçekte birçok sistem (4.16) ve (4.17) eşitliğinden olduğu gibi doğrusal olarak modellenemez. Doğrusal olmayan

sistemlerde sistem dinamikleri bir önceki durumun f ve h dönüşüm işlevleri sonrasında (4.18) ve (4.19)'deki gibi tanımlanabilir.

$$\hat{x}_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \tag{4.18}$$

$$z_k = h(x_k, v_k) \tag{4.19}$$

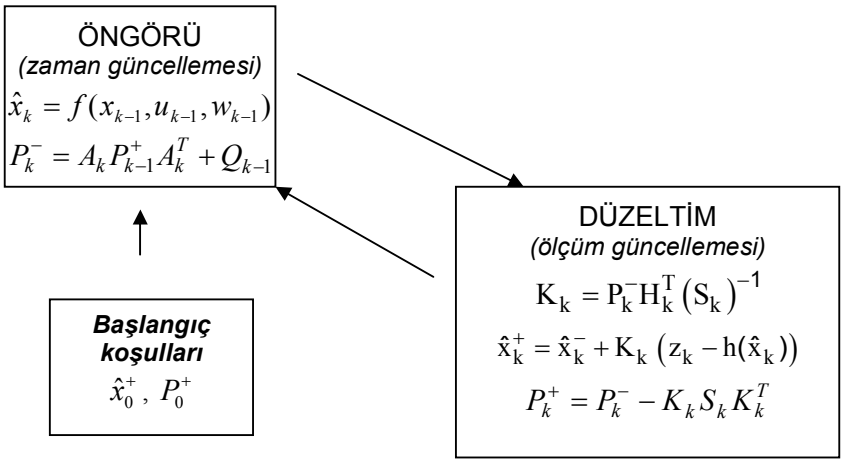
Kalman süzgecinin doğrusal olmayan sistemlerde kullanılması f ve h işlevlerinin doğrusallaştırılması ile olur. Bunun nedeni, doğrusal olmayan dönüşümler sonrası Gauss olasılık yoğunluk işlevinin, Gauss özelliği gösterememesidir. Bu nedenle durum denklemindeki doğrusal olmayan f ve h işlevleri, durum denklemini sağlayan $(x_k^{nom} : k = 0,1,2,...)$ ve $(z_k^{nom} : k = 0,1,2,...)$ dizilerini etrafında Taylor serisi açılarak doğrusallaştırılabilir. Böylece f ve h işlevlerinin durum vektörüne göre birinci dereceden kısmi türevleri alındığında A_k ve H_k Jacobian matrisleri (4.20) ve (4.21) eşitliklerindeki gibi elde edilir.

$$A_k = \left.\frac{\partial f(x_{k-1})}{\partial x}\right|_{x_{k-1}=x_{k-1}^{nom}} \tag{4.20}$$

$$H_k = \left.\frac{\partial h(x_k)}{\partial x}\right|_{x_k=x_k^{nom}} \tag{4.21}$$

4.3.1 Genişletilmiş Kalman Süzgeci Algoritması

GKS algoritması, doğrusal Kalman süzgeci algoritmasından çok farklı değildir. Ancak burada dikkat edilmesi gereken konu A_k ve H_k Jacobian matrislerinin bir önceki durum kestirimine göre yeniden hesaplanıyor olmasıdır. Dolayısıyla A_k ve H_k Jacobian matrisleri çevrimiçi olarak, durumun her yeni kestiriminde yeniden hesaplanacaktır. GKS algoritması şekil 4.4'de gösterildiği gibidir.



Şekil 4.4 GKS Algoritması

Bir sonraki durum belirsizliğinin hesaplanmasında kısaca S_k ile belirtilen matris Kalman kazancının hesaplanmasında kullanılan yenilenme kovaryans matrisidir (4.22).

$$S_k = H_k P_k^- H_k^T + R_k \tag{4.22}$$

4.4 EKBH'nin Olasılıksal Modelinin Çıkartılması

EKBH uygulamalarındaki amaç eş zamanlı olarak robotun konumlandırılması yapmak ve ortam haritasını doğru şekilde belirleyebilmektir. Bu problem robot yerinin ve harita konumlarının olasılık dağılım fonksiyonlarıyla ifade edilerek Bayes tabanlı kestirim yöntemleri kullanılarak çözümlendirilmiştir. EKBH problemin çözümü için harita Θ ve robot konum s_t olasılık yoğunluklarının, gözlemler $z^t = \{z_1 , z_2 , \ldots , z_t\}$, denetim girdiler $u^t = \{u_1 , u_2 , \ldots , u_t\}$, harita Θ ile gözlem z^t arasındaki veri ilişkilendirmeleri $n^t = \{n_1 , n_2 , \ldots , n_t\}$ sonucunda kestirimi yapılır. Bu durumda EKBH çözümü için koşullu olasılık (4.23)'deki tanımlanmıştır.

$$p(s_t, \Theta | z^t, u^t, n^t) \tag{4.23}$$

Hareket ve ölçüm modellerinin olasılık yoğunlukları,

$$p(s_t | s_{t-1}, u_t) \tag{4.24}$$

$$p(z_t | s_t, \Theta, n^t) \tag{4.25}$$

(4.24) ve (4.25) deki gibi ifade edilebilir.

Bayes kuralları kullanılarak (4.23)'deki koşullu olasılık,

$$p(s_t, \Theta | z^t, u^t, n^t) = \eta p(z_t | s_t, \Theta, z^{t-1} u^t, n^t) p(s_t, \Theta | z^{t-1}, u^t, n^t) \tag{4.26}$$

(4.26) denklemindeki gibi ifade edilir. Burada η normalizasyon katsayısıdır ve olasılık değerinin [0,1] aralığında kalması sağlar. Denklemdeki ikinci terimde her bir ölçümün geri kalan diğer ölçümlerden bağımsız olduğu ve ölçümlerin de denetim girdilerine bağımlı olmadığı düşünülürse, ikinci terim (4.27) deki gibi sadeleştirilebilir.

$$p(z_t | s_t, \Theta, z^{t-1}, u^t, n^t) \rightarrow p(z_t | s_t, \Theta, n^t) \tag{4.27}$$

Bu durumda (4.26) eşitliği yeniden yazılırsa,

$$p(s_t, \Theta | z^t, u^t, n^t) = \eta \underbrace{p(z_t | s_t, \Theta, n^t)}_{\text{Ölçüm Modeli}} p(s_t, \Theta | z^{t-1}, u^t, n^t) \tag{4.28}$$

gibi olur.

(4.28) eşitliğindeki üçüncü terim için toplam olasılık yoğunluk fonksiyonu, Markov varsayımları kullanıldığında, (4.29)'daki gibi ifade edilir.

$$p(s_t, \Theta | z^{t-1}, u^t, n^t) \rightarrow \int p(s_t, \Theta | s_{t-1}, z^{t-1}, u^t, n^t)\ p(s_{t-1} | z^{t-1}, u^t, n^t)\ d_{s_{t-1}} \quad (4.29)$$

Böylece EKBH çözümü için koşullu olasılık yeniden ifade edilecek olursa,

$$\begin{aligned} p(s_t, \Theta | z^t, u^t, n^t) = \eta\ & p(z_t | s_t, \Theta z^{t-1} u^t, n^t) \times \\ & \int p(s_t, \Theta | s_{t-1}, z^{t-1}, u^t, n^t)\ p(s_{t-1} | z^{t-1}, u^t, n^t)\ d_{s_{t-1}} \end{aligned} \quad (4.30)$$

gibi olur.

(4.30) eşitliğinde, integral içindeki ilk terim koşullu olasılık tanımına göre parçalara ayrılırsa,

$$p(s_t, \Theta | s_{t-1}, z^{t-1}, u^t, n^t \rightarrow p(\Theta | s_{t-1}, z^{t-1}, u^t, n^t)\ \ p(s_t | s_{t-1}, \Theta, z^{t-1}, u^t, n^t) \quad (4.31)$$

gibi dağılır.

(4.31) eşitliğindeki üçüncü terim için robot pozisyonun t anındaki olasılığı (s_t), sadece bir zaman önceki t-1 (s_{t-1}) olasılık değerine ve yine t anındaki denetim girdisine bağlıdır. Diğer değişkenlerden bağımsız olduğu kabul edilirse (4.32)'deki gibi sadeleştirilebilir.

$$p(s_t | s_{t-1}, \Theta, z^{t-1}, u^t, n^t) \rightarrow p(s_t | s_{t-1}, u^t) \quad (4.32)$$

Bu durumda (4.31) ve (4.32) eşitliklerinde yapılan sadeleştirmelerle, EKBH için toplam olasılık dağılımı yeniden ifade edilecek olursa

$$p(s_t, \Theta | z^t, u^t, n^t) = \eta\ \ p(z_t | s_t, \Theta, n^t)\ \int \underbrace{p(s_t | s_{t-1}, u^t)}_{\text{Hareket Modeli}}\ \ p(\Theta | s_{t-1}, z^{t-1}, u^t, n^t)\ p(s_{t-1} | z^{t-1}, u^t, n^t)\ d_{s_{t-1}} \quad (4.33)$$

(4.33) eşitliğindeki gibi olur.

İntegral içindeki son iki terimde koşullu olasılık teoreminden yararlanarak birleştirilebilir.

$$p(\Theta|s_{t-1}, z^{t-1}, u^t, n^t)\ \ p(s_{t-1}|z^{t-1}, u^t, n^t) \rightarrow \qquad p(s_{t-1}, \Theta|z^{t-1}, u^t, n^t) \tag{4.34}$$

EKBH için Bayes süzgeç eşitliğinin matematiksel ifadesi (4.35)'deki gibi olur.

$$p(s_t, \Theta|z^t, u^t, n^t) = \eta\ \underbrace{p(z_t|s_t, \Theta, n^t)}_{\text{Ölçüm Modeli}}\ \int\ \underbrace{p(s_t|s_{t-1}, u^t)}_{\text{Hareket Modeli}}\ \ p(s_{t-1}, \Theta|z^{t-1}, u^{t-1}, n^{t-1})\ \ ds_{t-1} \tag{4.35}$$

(4.35) eşitliğinde de görüldüğü gibi ölçüm ve hareket modeli sisteme tek bir matematiksel ifade ile dahil edilmiştir.

4.4 Eş Zamanlı Konum Belirleme Ve Haritalamanın Olasılıksal Modelinin Genişletilmiş Kalman Süzgecine Uygulanması

(4.35) eşitliğini uygulamak için, dinamik sistem modellerinin ve olasılık yoğunluk fonksiyonun tanımlanması gerekir. Dinamik sistem modelleri robotun hareket ve ölçüm modelleridir. GKS yaklaşımında tüm olasılık dağılım fonksiyonları çok değişkenli Gauss dağılımı olarak tanımlanır. Bu durumda (4.35)'deki olasılık dağılımının μ_t ortalama vektörü ve $\sum_t$ kovaryans matrisine sahip Gauss dağılımı olduğu kabul edilir (4.36).

$$p(s_t, \Theta|z^t, u^t, n^t) \sim N(\mu_t, \sum_t) \tag{4.36}$$

Gauss dağılımın ortalama vektörü μ_t güncel sistem durumu gösterir. Sistem durum değişkenleri (4.37) eşitliğinde olduğu gibi robot ve harita konumların güncel değerleridir.

$$x = [s_t, \theta_0, \theta_1, ..., \theta_n] \tag{4.37}$$

Gauss dağılımı ile x durum değişkenleri üzerinden olasılık dağılım fonksiyonun tanımlanabilmesi için ortalama vektör ve kovaryans matrisin hesaplanması gerekir.

$$\hat{x} = E[x] \tag{4.38}$$

$$\tilde{x} = x - \hat{x} \tag{4.39}$$

$$\sum(x) = E[\tilde{x} . \tilde{x}^T] \tag{4.40}$$

$$\int p(s_t | s_{t-1}, u^t)\ p(s_{t-1}, \Theta | z^{t-1}, u^{t-1}, n^{t-1}) ds_{t-1} \ \sim\ N(\hat{x}_t^-, P_t^-) \tag{4.41}$$

Bayesian süzgecindeki önsel olasılık (4.41) eşitliğinde görüldüğü gibi $\hat{x}_t^-$ ortalama vektörü ve P_t^- kovaryans matrisi ile Gauss dağılmaktadır. Bu ortalama vektör ve kovaryans matris GKS algoritmasının öngörü adımındaki eşitlikler ile hesaplanır. Böylece bir sonraki durumun öngörüsü gerçekleştirilmiş olur. (zaman - güncellemesi).

$$p(z_t | s_t, \Theta, n^t) p^-(s_t, \Theta | u^t) \tag{4.42}$$

Bayesian süzgecindeki öngörünün düzeltilmesi (4.35) eşitliğindeki ölçüm modeline ait olasılık yoğunluk fonksiyonun tanımlanması veya öngörülen ortalama vektörün ve kovaryans matrisin güncellemesiyle yapılmalıdır. Bu da GKS algoritmasının ölçüm güncellemesi adımındaki eşitliklerin hesaplanması ile gerçekleştirilir (ölçüm-güncellemesi). Böylece bir sonraki durum kestirimi gerçekleştirilir. Sonuç olarak (4.42)'deki sonraki olasılık yoğunluk fonksiyonu üzerinden t zamanındaki robot konumunun ve ortam haritasının eş zamanlı olarak konum kestirimi yapılmış olur.

5. ROBOT YAPISI VE SİSTEM MODELLERİ

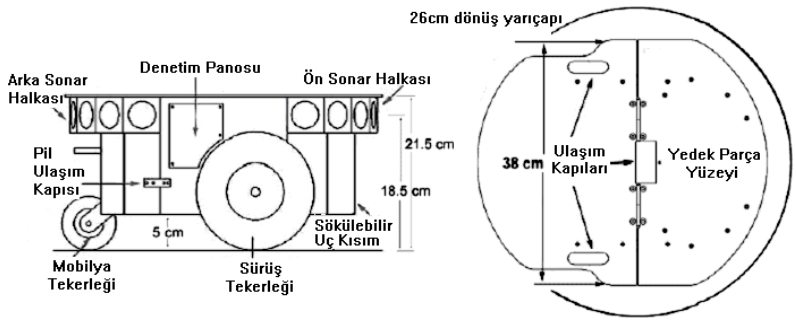
5.1 Robot Yapısı

Şekil 5.1'de görüldüğü gibi ActiveMedia firması tarafından üretilen Pioneer 3DX robotu, iki adet diferansiyel ve bir adet mobilya tekerden oluşan, gezgin özelliğine sahip, çok amaçlı kullanılabilen, üzeriden bulunan işlemci ve gezgin robot yazılımları sayesinde yönetilebilen bir yapıdadır.

Şekil 5.1 Pioneer 3DX robotun önden görünüşü

Pioneer 3DX modelindeki standart donanımlar :

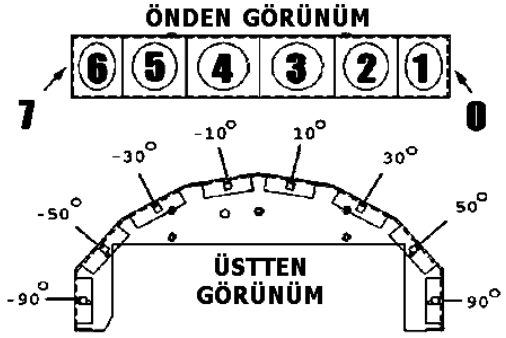
- Yüksek çözünürlükte kodlayıcılar (500 vurum/dönüm)
- 1 adet RS232 seri kanal portu, 9.6 ile 115.2 kilobaud arasında ayarlanabilir
- 3 adet batarya, toplam 9 kg
- 8 ön ve 8 arka kısımda olmak üzere toplam 16 sonar algılayıcı
- Reset ve Motor basma düğmeleri
- Kullanıcı denetim paneli
- Siemens C166 Mikroişlemci
- Lazer algılayıcı (isteğe bağlı)
- Kamera (isteğe bağlı)
- 2-DOF kıskaç (isteğe bağlı)
- Pusula



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

Pioneer 3DX robot, 44cm x 38cm x 22cm boyutunda olup alüminyum gövdeye sahiptir. Robot gövdesi üzerinde maksimum 23 kg'a kadar yük taşıyabilir. Dönüş yarıçapı şekil 5.2'de gösterildiği gibi 26 cm'dir. Sonar algılayıcıların yerden yüksekliği ise 18 cm'dir. Pioneer 3DX gezgin robotunun maksimum ulaşabileceği hız 1.6 m/s' dir.

Pioneer 3DX üzerindeki ses ötesi algılayıcılar robotun önünde ve arkasında 2 dizi halinde olmak üzere yerleştirilmiştir. Bu ses ötesi algılayıcı dizileri sayesinde, neredeyse 360 derecelik bir alanın taranması sağlanmıştır. Böylece bu mesafe ölçümlerinin değerlendirilmesiyle robotlar hem kendi konumlarını hem de çevresindeki nesnelerin kendilerine göre konumlarını tespit edebilir. Algılayıcıların hassasiyeti yaklaşık 15cm ile 5m arasındadır. Şekil 5.3'de robotun ön tarafında yer alan 8 adet algılayıcının dizilimi görülmektedir. Aynı dizilim robotun arka tarafında da yer almaktadır.



Şekil 5.3 Ön taraftaki sonar dizisi dağılımı

Pioneer 3DX'de robotun hareketini sağlayan tekerler yüksek torklu ve yüksek hızlı çift yönlü düz akım motorlar ile sürülür. Bu motorların bağlı olduğu iki adet 38.3:1 oranında dişli çark mevcuttur. Motor şaftı üzerinde yüksek çözünürlükteki kodlayıcılar ile robotun hız, yön ve aldığı mesafe bilgileri elde edilebilir. Kullanılan kodlayıcılar 500 vurum/dönüm özelliğine sahiptir. Ayrıca kodlayıcı bir mm yer değişimde 128 vurum üretir (128 vurum/mm). Bu sayede robot kendi konumunu doğru bir şekilde saptayabilir.

5.1.1 Robot Yazılımları

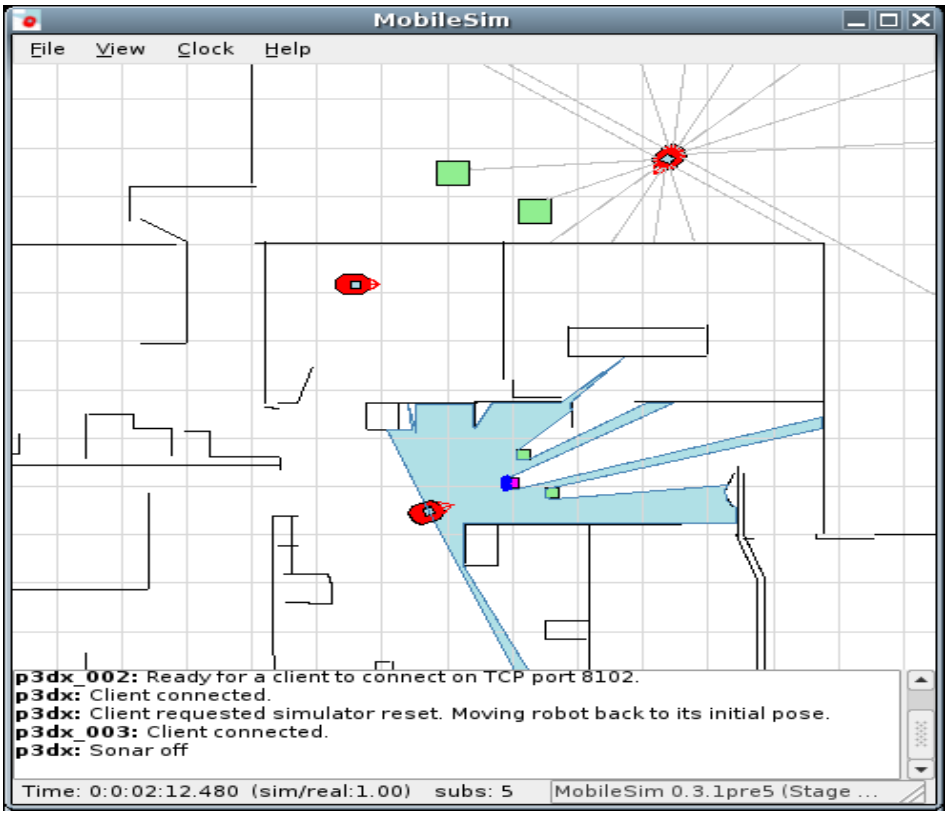
5.1.1.1 ARIA

ARIA (Advanced Robotics Interface for Applications) arayüzü Pioneer robotlarla kullanıcı arasındaki iletişimi sağlamak için geliştirilmiş bir yazılımdır. ARIA, Windows ve Linux işletim sistemlerinde çalışabilen, C++ tabanlı, açık kaynak geliştirmeli ve nesne yönelimli bir yazılımdır. Esnek bir yapıya sahiptir ve robot ile ilgili tek bir davranışı gerçekleştirebileceği gibi birçok davranışı da aynı anda gerçekleştirebilmektedir. ARIA güçlü ve büyük bir API'e (Application Programming Interface) sahiptir. API, robotun hareket denetimi için gerekli üst düzey sınıfları ve kütüphaneleri içermektedir. ARIA robot ile iletişimini sunucu/istemci ilişkisi kullanarak ya seri kanal üzerinden (robot ile haberleştiğinde) yada TCP kanal (MobileSim benzeticisi ile haberleştiğinde) üzerinden bağlantı kurarak yapar. ARIA kütüphaneleri, Windows işletim sistemi kullanılıyorsa Ms Visual C++ .NET (7.1), Linux işletim sistemi kullanılıyorsa G++ 3.x derleyicileri ile programlanır.

5.1.1.2 MobileSim Benzetimcisi

MobileSim, Pioneer robotlar için hazırlanmış bir benzetimci (simulator) programıdır. MobileSim ile ARIA arayüzü kullanılarak hazırlanan programlarlar veya uygulamalar robot üzeriden denenmeden önce bu benzetici kullanılarak test edilebilir. Böylece bu benzetici ile hataların ayıklanması ve uygulamanın doğruluğunun test edilebilmesi imkanı sağlanır. MobileSim çevredeki duvarları ve engelleri tanımlayabilmek için çizgi verileri kullanır. Mapper3-Basic programı ile

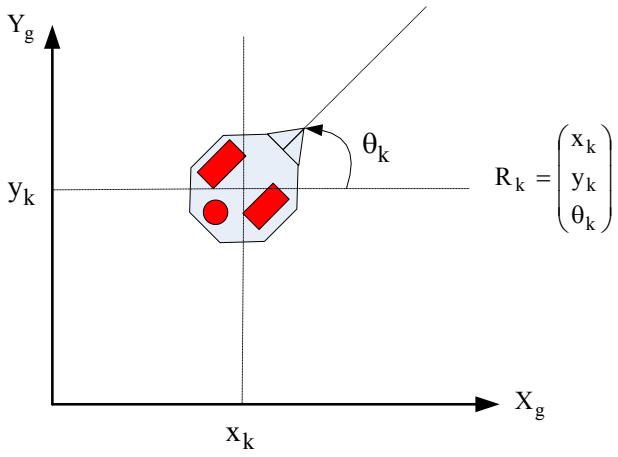
çizgi temelli harita oluşturulabilir. Böylece tanımlanan ortam içinde robotun gerçekte yapması gereken uygulamalar MobileSim ile taklit edilmiş olur. Şekil 5.4'de MobileSim benzetimcisine ait bir görünüm yer almaktadır.



Şekil 5.4 MobileSim programının görünümü

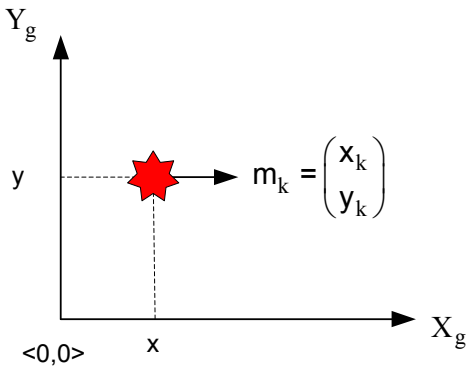
5.2 Hareket Modeli

Robotun bütünsel ortamdaki duruşu x_k, y_k, θ_k parametreleriyle tanımlanır. x_k ve y_k parametreleri robotun k anında ortamdaki konumunu, θ_k ise robotun bütünsel düzleme göre açısal durumunu ifade eder (Şekil 5.5).



Şekil 5.5 Robot konumu

İki boyutlu düzlem içinde noktasal yer göstericiler şekil 5.6'da görüldüğü gibi x_k, y_k parametreleriyle tanımlanır.



Şekil 5.6 İki boyutlu düzlemde noktasal konum parametreleri

GKS'de sistem durum vektörü, robotun konumsal verileri ve noktasal yer gösterici konumlarından oluşur. Artırılmış sistem durum vektörü aşağıdaki gibi tanımlanabilir :

$$x_k = \left[x_r \quad m_1 \quad m_2 \quad . \quad . \quad . \quad m_N \right]^T \qquad (5.1)$$

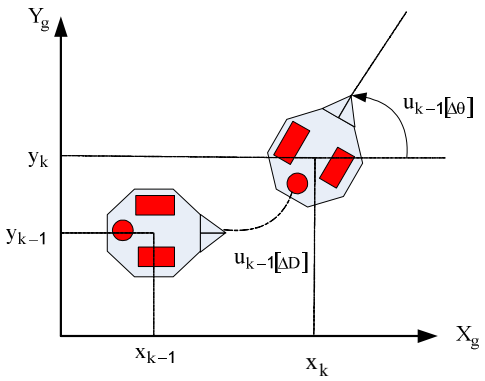
Sistem durumu vektöründeki $x_r = [x_k \quad y_k \quad \theta_k]^T$ robot konumunu, $m_j = [x_j \quad y_j]^T$ ise ortamdaki yer göstericileri ifade eder (j=1,2,…N). Bu durumda, (5.2) eşitliğine göre durum hata kovaryans matrisi aşağıdaki gibi oluşacaktır:

$$P_{k,k} = \begin{bmatrix} P_{rr} & P_{r1} & \cdots & P_{rN} \\ P_{1r} & P_{11} & \cdots & P_{1N} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ P_{Nr} & P_{N1} & \cdots & P_{NN} \end{bmatrix} \tag{5.2}$$

Durum hata kovaryans matrisindeki alt matrisler P_{rr} , P_{ri} , P_{ii} sırasıyla, robotun robota, robotun yer göstericiye, yer göstericinin yer göstericiye göre kovaryans matrislerini ifade eder. Böylece sistemin durum uzay gösterimi, her bir k anı için durum hata kovaryansı $P_{k,k}$ olan, tek bir durum vektörü x_k ile ifade edilir.

Şekil 5.7'de robot, u_k denetim girdileri sayesinde (x_{k-1},y_{k-1}) konumundan (x_k,y_k) konumuna hareket etmiştir. Bu hareket sonrasında robotun ΔD kadar mesafesi ve $\Delta\theta$ kadar da açısı değişmiştir [2]. Böylece u denetim girdileri sayesinde robotun ortam içinde ne kadar yer değiştirdiği belirlenebilir.

$$u_k = [u_{k[\Delta D]} \quad u_{k[\Delta\theta]}]^T \tag{5.3}$$



Şekil 5.7 Denetim girdileri ile robot hareketi

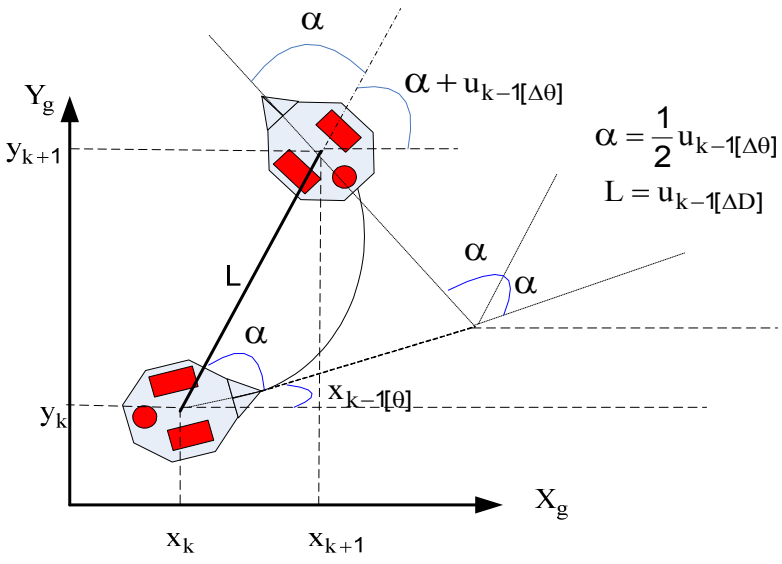
Bu tez çalışmasında denetim girdileri, robotun doğrusal hızı ve dönme hızı olarak seçilmiştir. Robotun doğrusal hızı $u_{k[\Delta D]}$, her iki tekerleğinin dönüş hızlarının ortalamalarından, robotun dönüş hızı $u_{k[\Delta\theta]}$ ise her bir tekerleğin dönüş hızları arasındaki fark ve tekerleklerin merkezleri arasındaki uzaklık kullanılarak hesaplanabilir.

$$v_k = \frac{v_{kR} + v_{kL}}{2} = \frac{R.w_{kR} + R.w_{kL}}{2} \qquad u_{k[\Delta D]} = v_k \,.\, t_{samp} \quad , u_{k|\Delta\theta|} = 0 \tag{5.4}$$

$$w_k = \frac{v_{kR} - v_{kL}}{b} = \frac{R.w_{kR} - R.w_{kL}}{b} \qquad u_k[\Delta\theta] = w_k \,.\, t_{samp} \tag{5.5}$$

(5.4) ve (5.5) eşitliklerindeki t_{samp} *k-1* anından *k* anına kadar geçen süreyi kısacası örnekleme zamanını belirtir. Böylece robotun bu süre içersindeki yer değişimi doğrusal ve açısal hızlarının zaman ile çarpımına eşit olur.

Robotun denetim girdileri ile bütünsel düzlemde ne kadar yer değiştirdiği şekil 5.8'de gösterilmiştir. Robot u_k denetim girdileri ile *k* anında (x_k, y_k, θ_k) konumundan bir zaman sonra *k+1* anında $(x_{k+1}, y_{k+1}, \theta_{k+1})$ konumuna gelmiştir.



Şekil 5.8 Robotun dönüş hareketi

Elde edilmek istenen yeni konum $\hat{x}_k$ sistemin hareket modeliyle tanımlanır. Bu tez çalışmasında ortamın durağan olduğu varsayılmıştır. Dolayısıyla ortamdaki yer göstericilerin konumları sabittir ve zamana bağlı olarak değişmez. Yer göstericiler için durum geçiş işlevi (5.6)'daki gibi tanımlanır. Bu durumda bir sonraki durum vektörü, doğrusal olmayan sistem modeli ile tanımlanırsa, yeni durum vektörü (5.7) eşitliğindeki gibi olur.

$$m_{k,j} = m_{k-1,j} \quad M = \left[m_1 . \quad . \quad . \quad m_N \right]^T \tag{5.6}$$

$$\hat{x}_k = f(x_{k-1}, u_{k-1}, w_{k-1}) = \begin{bmatrix} f(x_{k-1}^{robot}, u_{k-1}, w_{k-1}) \\ M \end{bmatrix} \tag{5.7}$$

$$\hat{x}_k = f(x_{k-1}, u_{k-1}) = \begin{bmatrix} x_{k-1[x]} + u_{k-1[\Delta D]} \cos(x_{k-1[\theta]} + \frac{1}{2} u_{k-1[\Delta\theta]}) \\ x_{k-1[y]} + u_{k-1[\Delta D]} \sin(x_{k-1[\theta]} + \frac{1}{2} u_{k-1[\Delta\theta]}) \\ x_{k-1[\theta]} + u_{k-1[\Delta\theta]} \\ m_1 \\ . \\ . \\ . \\ m_N \end{bmatrix}_{(2N+3)\,x\,1} \tag{5.8}$$

(5.8) eşitliğinde doğrusal olmayan hareket modeliyle bir sonraki durumun matematiksel ifadesi yer almaktadır. Bu eşitlik içinde süreç gürültüsü tanımlanmamıştır. Süreç gürültüsü denetim girdilerinden kaynaklanır. Robot tekerleri üzerinde bulunan kodlayıcıların hassasiyeti sistem gürültüsünün değişimini etkiler. (5.9) eşitliğinin (5.8)'e dahil edilmesiyle bir sonraki sistem durumu elde edilmiş olur.

$$u_k = \begin{bmatrix} u_{k[\Delta D]} \\ u_{k[\Delta\theta]} \end{bmatrix} + \begin{bmatrix} q_{k[\Delta D]} \\ q_{k[\Delta\theta]} \end{bmatrix} \tag{5.9}$$

$$\hat{x}_k = f(x_{k-1}, u_{k-1}, w_{k-1}) = \begin{bmatrix} x_{k-1[x]} + (u_{k-1[\Delta D]} + q_{k[\Delta D]}).\cos\left(x_{k-1[\theta]} + \frac{1}{2}(u_{k-1[\Delta\theta]} + q_{k[\Delta\theta]})\right) \\ x_{k-1[y]} + (u_{k-1[\Delta D]} + q_{k[\Delta D]}).\sin\left(x_{k-1[\theta]} + \frac{1}{2}(u_{k-1[\Delta\theta]} + q_{k[\Delta\theta]})\right) \\ x_{k-1[\theta]} + (u_{k-1[\Delta\theta]} + q_{k[\Delta\theta]}) \\ m_1 \\ . \\ . \\ . \\ m_N \end{bmatrix}_{(2N+3)\,x\,1} \tag{5.10}$$

Süreç gürültüsünün sıfır ortalamalı Gauss olduğu varsayılırsa, gürültünün kovaryans matrisi (5.13)'deki gibi tanımlanabilir.

$$q_k \sim N(\hat{q}_k, U_k) \tag{5.11}$$

$$\hat{q}_k = [0 \quad 0]^T \tag{5.12}$$

$$U_k = \begin{bmatrix} \sigma^2_{q_{k[\Delta D]}} & 0 \\ 0 & \sigma^2_{q_{k[\Delta\theta]}} \end{bmatrix}_{2x2} \tag{5.13}$$

(5.13) eşitliğinde köşegenler dışında kalan terimler sıfırdır. Bunun nedeni gürültü kaynaklarının birbirinden bağımsız olduklarının kabul edilmesidir.

Doğrusal olmayan durum geçiş işlevi ile bir zaman önceki durum ile şimdiki durum ilişkilendirilmiştir. Denetim girdilerinden kaynaklanan gürültünün diğer bir ifadeyle süreç gürültüsünün sisteme kattığı belirsizliğin de hesaplanması gerekir. Bunun için Q_{k-1} öngörülemeyen gürültü kovaryans matrisi (5.14) eşitliği ile bulunur.

$$Q_{k-1} = \Delta f \; x \; U_{k-1} \; x \; \Delta f^T \tag{5.14}$$

$$\Delta f = [\frac{\partial f(x_{k-1})}{\partial u_{k[\Delta D]}} \quad \frac{\partial f(x_{k-1})}{\partial u_{k[\Delta\theta]}}] \tag{5.15}$$

Burada Δf Jacobian matrisi, durum geçiş işlevinin denetim girdilerine göre kısmı türevlerinin alınması ile bulunur (5.16).

$$\Delta f = \begin{bmatrix} \cos(x_{k-1[\theta]} + \frac{1}{2}\Delta\theta) & -\frac{\Delta D}{2}.\sin(x_{k-1[\theta]} + \frac{1}{2}\Delta\theta) \\ \sin(x_{k-1[\theta]} + \frac{1}{2}\Delta\theta) & \frac{\Delta D}{2}.\cos(x_{k-1[\theta]} + \frac{1}{2}\Delta\theta) \\ 0 & 1 \\ 0 & 0 \\ . & . \\ . & . \\ . & . \\ 0 & 0 \end{bmatrix}_{(2N+3)x2} \quad \begin{matrix} \Delta D = (u_{k-1[\Delta D]} + q_{k[\Delta D]}) \\ ,\Delta\theta = (u_{k-1[\Delta\theta]} + q_{k[\Delta\theta]}) \end{matrix} \qquad (5.16)$$

Bir başka belirsizlikte robotun k-1 anındaki belirsizliğinin bir sonraki zamanı nasıl etkileyeceğini gösteren sistem durum belirsizliğidir. Bunun için (5.17) eşitliğinde gösterilen A_{k-1} Jacobian matrisinin hesaplanması gerekir. Bu matris durum geçiş işlevin her bir durum değişkenine göre kısmi türevinin alınmasıyla belirlenir.

$$A_{k-1} = \left[\frac{\partial f(x_{k-1})}{\partial x}\right] = \left[\frac{\partial f(x_{k-1})}{\partial x_{k-1[x]}} \quad \frac{\partial f(x_{k-1})}{\partial x_{k-1[y]}} \quad \frac{\partial f(x_{k-1})}{\partial x_{k-1[\theta]}} \quad \frac{\partial f(x_{k-1})}{\partial m_1} \quad . \quad . \quad . \quad \frac{\partial f(x_{k-1})}{\partial m_N}\right] \qquad (5.17)$$

Böylece A_{k-1} Jacobian matrisi (5.18) deki ifade edilir.

$$A_{k-1} = \begin{pmatrix} J & 0 & \cdots & 0 \\ 0 & I & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & I \end{pmatrix}_{(2N+3)x(2N+3)} \qquad (5.18)$$

(5.18)'deki J matrisi durum geçiş işlevinin robot konum parametrelerine göre alınan kısmı türevini göstermektedir (5.19). A_{k-1} Jacobian matrisinin köşegenleri ise birim matris ile ifade edilir.

$$J = \begin{pmatrix} 1 & 0 & \frac{-\Delta D}{2}.\sin(x_{k-1[\theta]} + \frac{1}{2}\Delta\theta) \\ 0 & 1 & \frac{\Delta D}{2}.\cos(x_{k-1[\theta]} + \frac{1}{2}\Delta\theta) \\ 0 & 0 & 1 \end{pmatrix}, \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \begin{matrix} \Delta D = (u_{k-1[\Delta D]} + q_{k[\Delta D]}) \\ \Delta\theta = (u_{k-1[\Delta\theta]} + q_{k[\Delta\theta]}) \end{matrix} \qquad (5.19)$$

Daha önceki durum belirsizliğin bir adım sonraki durum belirsizliğini nasıl etkileyeceği, bir önceki adımdaki durum belirsizliği P_{k-1}^{+}'in, A_{k-1} Jacobian matrisi ile (5.20)'deki gibi çarpılıp, süreç gürültüsünden gelen belirsizliğin de üzerine eklenmesi ile bulunur (P_k^{-}).

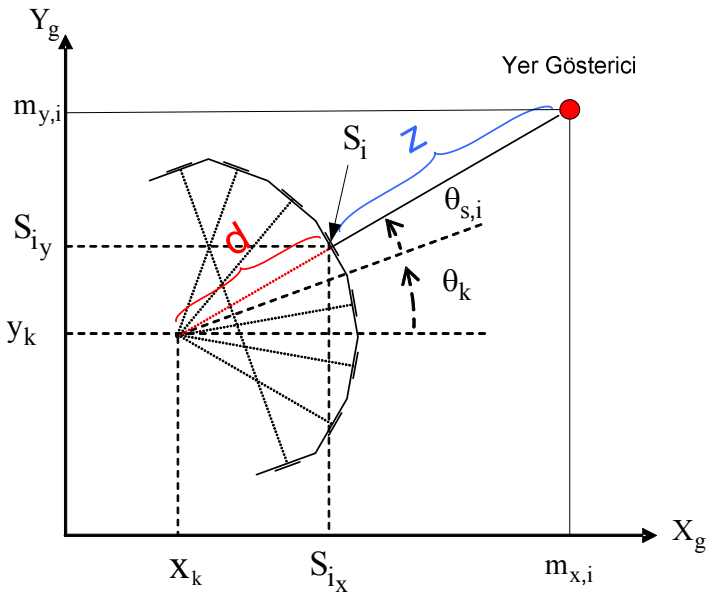
$$P_k^{-} = AP_{k-1}^{+}A^T + Q_{k-1} \qquad (5.20)$$

5.3 Ölçüm Modeli

Robot konumunun izlenmesinde kullanılan denetim girdileri robot tekerleri üzerinde bulunan kodlayıcılardan elde edilir. Ancak robotun mutlak ölçümler yardımıyla odometri kaynaklı hataları düzeltmesi beklenir. Bunun için de robot üzerinde konumlanmış algılayıcılardan faydalanılır. Bu algılayıcılar ile robotun çevredeki yer göstericilere olan uzaklık ve açısal fark bilgileri elde edilir.

Gerçek hayatta sonar algılayıcılar sesin yayılma prensibine göre çalıştıkları için mesafe bilgisi Bölüm 3'de anlatıldığı gibi koni şeklinde bir alandan elde edilir. Ancak sonarın algılama doğrultusunun koni yerine, doğrusal olduğu varsayılırsa sistemin ölçüm modeli (5.21) eşitliğindeki gibi tanımlanabilir.

$$h(x_k, v_k) = \begin{bmatrix} z_r^j \\ z_\beta^j \end{bmatrix} + v_k = \begin{bmatrix} \sqrt{(m_{x,j} - x_k)^2 + (m_{y,j} - y_k)^2} \\ a\tan\left(\frac{(m_{y,j} - y_k)}{(m_{x,j} - x_k)}\right) - \theta_k \end{bmatrix}_{2x1} + v_k \qquad (5.21)$$

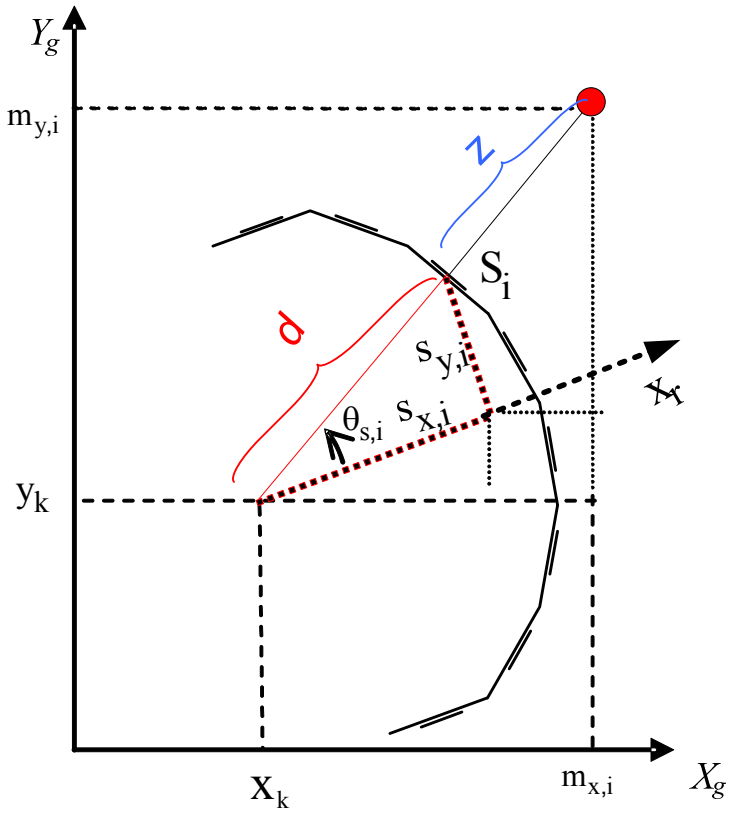


Şekil 5.9 Sonarların algılama doğrultusu

(5.21) eşitliğinde de görüldüğü gibi j numaralı yer gösterici ile robot merkezi arasındaki uzaklık z_r^j, robotun yatay ekseni ile yer gösterici arasındaki açı z_β^j ile gösterilmiştir.

Şekil 5.9'da ölçümlerin yer gösterici ile sonar algılayıcı arasında olduğu görülmektedir. Ancak (5.21) eşitliğindeki z_r^j ise robot merkezi ile yer gösterici arasındaki uzaklıktır. Bu durumda çizelge 6.5'de tanımlanan her bir sonarın robot merkezine olan yatay ve dikey uzaklıklarından d mesafesi hesaplanarak, sonar ölçümüne eklenmelidir. Böylece S_i sonardan alınan mesafe ve konum açısı ölçümü (5.22) deki gibi tanımlanabilir (Şekil 5.10).

$$z_{k,i} = z_{S_i} = \begin{bmatrix} d + z \\ \theta_{s,i} \end{bmatrix}, \quad d = \sqrt{s_{x,i}^2 + s_{y,i}^2} \tag{5.22}$$



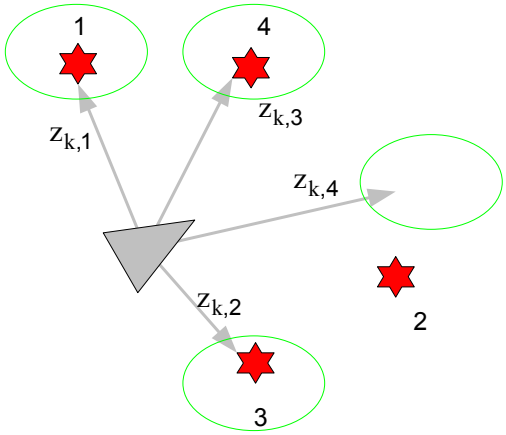
Şekil 5.10 Sonardan alınan mesafe ve açı verisi

5.3.1 Veri İlişkilendirme

(5.22)'de tanımlanan sonar ölçümlerin ortamdaki yer göstericilerden herhangi birine ait olup olmadığının belirlenmesi gerekir. Veri ilişkilendirme süreci, k anında robot üzerindeki m adet sonardan (S_i i=1,…,m) ölçüm alınacağı düşünülürse, durum vektöründe tanımlanan N adet yer göstericiden (m_j j=1,…,N) hangisi ile eşleştirilmesi gerektiğini gösterir.

$$H_k = [j_1\ j_2 \dots j_m] \tag{5.23}$$

(5.23)'deki eşleştirme matrisi her bir $z_{k,i}$ ölçümüne ait ilişkilendirilmiş yer gösterici numarasını göstermektedir. Buna göre $H_k = [1\,3\,4\,0]$ olduğu düşünülürse 1,2,3 numaralı sonar ölçümlerin sırasıyla 1,3,4 numaralı yer göstericilerle eşleştirildiği anlaşılır. 4 numaralı sonar ölçümü ise hiçbir yer gösterici ile eşleştirilmemiştir (Şekil 5.11).



Şekil 5.11 Veri ilişkilendirme

$z_{k,i}$ ölçümü m_j yer göstericisi ile ilişkilendirilmişse robotun elde etmesi gereken ölçüm ile elde ettiği ölçüm arasındaki matematiksel bağıntı (5.24)'deki gibi olur.

$$z_{k,i} = h_{k,j}(x_k) + v_{k,i} \tag{5.24}$$

Her bir sonar algılayıcıdan elde edilen veri, belirli bir gürültüye sahiptir. Bu gürültüler ($v_{k,i}$) (5.24) eşitliğine dahil edilmiştir. Algılayıcı gürültülerin sıfır ortalamalı Gauss olasılık dağılımına sahip oldukları varsayılırsa ölçüm gürültüsünün kovaryans matrisi (5.26)'daki gibi tanımlanabilir.

$$v_{k,i} \sim N(0, R_{k,i}) \tag{5.25}$$

$$R_{k,i} = \begin{bmatrix} \sigma_{r,i}^2 & 0 \\ 0 & \sigma_{\theta,i}^2 \end{bmatrix}_{2x2} \tag{5.26}$$

$R_{k,i}$ i'nci sonarın gürültü kovaryans matrisini göstermektedir. Köşegen dışında kalan terimlerin sıfır olmasının nedeni uzaklık ve açı ölçümlerinin birbirinden bağımsız olduklarının kabul edilmesidir.

(5.24) eşitliğindeki hata, robotun öngörülen konuma geldiğinde elde etmesi gereken ölçüm ile elde ettiği ölçüm arasındaki farka eşittir. Bu hatanın bir önceki durum belirsizliğine dahil edilmesi gerekir. Bu nedenle ölçüm modelinin durum değişkenlerine göre kısmi türevleri hesaplanmalıdır. Böylece H_k Jacobian matrisi (5.27)'de gibi yazılabilir.

$$H_{k,j} = \left[\frac{\partial h(x)}{\partial x}\right] = \left[\frac{\partial h(x)}{\partial x_{k[x]}} \quad \frac{\partial h(x)}{\partial x_{k[y]}} \quad \frac{\partial h(x)}{\partial x_{k[\theta]}} \quad \frac{\partial h(x)}{\partial m_1} \quad . \quad . \quad . \quad \frac{\partial h(x)}{\partial m_N}\right]$$

$$H_{k,j} = \begin{bmatrix} \frac{\Delta x}{\Delta} & \frac{\Delta y}{\Delta} & 0 & 0 & 0 & \dots & -\frac{\Delta x}{\Delta} & -\frac{\Delta y}{\Delta} & \dots & 0 & 0 \\ -\frac{\Delta y}{\Delta^2} & \frac{\Delta x}{\Delta^2} & -1 & 0 & 0 & \dots & \frac{\Delta y}{\Delta^2} & -\frac{\Delta x}{\Delta^2} & \dots & 0 & 0 \end{bmatrix}_{2x(2N+3)} \tag{5.27}$$

$$\Delta x = \left(m_{x,j} - x_k\right), \quad \Delta y = \left(m_{y,j} - y_k\right), \quad \Delta = \sqrt{\Delta x^2 + \Delta y^2} \tag{5.28}$$

(5.27) deki H_k Jacobian matrisinde $z_{k,i}$ ile ilişkilendirilmemiş diğer yer göstericilerin kısmi türevlerinin sıfır olduğu görülmektedir. Bu durum ölçüm modelinde sadece veri ilişkilendirmesi yapılan yer gösterici konumlarına ($m_{x,j}$ ve $m_{y,j}$) göre kısmı türevlerin alınmasından kaynaklanmıştır.

Bu durumda i'nci sonarın j'nci yer gösterici ile ilişkilendirilmesi sonrasında, ölçüm farkı (yenilenme terimi) ve yenilenme kovaryans matrisi aşağıdaki gibi tanımlanabilir.

$$v_{k,ij} = z_{k,i} - h_{k,j}(x_k) \qquad S_{k,ij} = H_{k,j} P_k^- H_{k,j}^T + R_{k,i} \tag{5.29}$$

(5.29)'daki eşitlikler ancak i'nci sonar ölçümü j'nci yer gösterici yakınlarından alınmış ise hesaplanabilir. Bu eşleştirme ise (5.30)'daki eşitsizliğin sağlamasıyla gerçekleştirilir.

$$D_{k,ij}^2 = v_{k,ij}^T S_{k,ij}^{-1} v_{k,ij} < \chi_{d,1-\alpha}^2 \tag{5.30}$$

5.3.1.1 Mahalanobis Uzaklığı

Mahalanobis uzaklığı, 1936 yılında P.C Mahalanobis tarafından öne sürülen bağımsız değişkenler uzayındaki merkezden veya örneklem ortalamasından tek bir veri noktasının uzaklığını ölçen istatistiksel bir ölçüm değeridir [27]. Öklit uzaklığından farklı olarak veri kümesindeki değişkenlerin korelasyonları da göz önünde bulundurularak hesaplama yapılır.

Çok değişkenli vektör $x=(x_1,x_2,\ldots,x_p)$ için ortalama değerleri $\mu=(\mu_1, \mu_2,\ldots, \mu_p)$ ve kovaryans matrisi Σ olarak tanımlanırsa, Mahalanobis uzaklığı (5.31) eşitliğindeki gibi hesaplanır.

$$D_M(x) = \sqrt{(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{5.31}$$

Mahalanobis uzaklığı iki rassal vektör arasındaki benzeşmezliği de ifade etmek için kullanılabilir. Bunun için aynı dağılıma ait rassal vektörlerin $\vec{x}$ ve $\vec{y}$ ve kovaryans matrislerinin Σ_{xy} olduğu varsayılırsa, Mahalanobis uzaklığı (5.32) eşitliğinde olduğu gibi hesaplanır.

$$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T \Sigma_{xy}^{-1}(\vec{x}-\vec{y})} \tag{5.32}$$

Mahalanobis uzaklığı kümeleme analizleri ve verilerin sınıflandırmasında oldukça sık kullanılan bir ölçümdür. Mahalanobis uzaklığı kullanılarak herhangi bir test noktasının, N adet sınıf içinden hangisine ait olduğunu belirlemek için öncelikle her bir sınıfla olan kovaryans matrisinin hesaplanması gerekir. Daha sonra test noktasının diğer sınıflarla olan Mahalanobis uzaklığı hesaplanarak en küçük Mahalanobis uzaklık değerine sahip olan sınıfla eşleştirilmesi yapılabilir.

Mahalanobis uzaklığı EKBH uygulamalarında robot algılayıcı ölçümleri ile ortamdaki yer göstericiler arasındaki eşleştirmeler için kullanılır. Bu eşleştirmelerin doğru şekilde yapılması GKS'nin ölçüm güncellemesinin de doğru şekilde yapılması anlamına gelir. Böylece bir sonraki durum kestirimin doğru yapılabilmesi ve GKS'nin yakınsaması bu eşleştirmelerin tutarlılığına bağlıdır.

(5.29) eşitliğindeki hata terimi $v_{k,ij}$ ve bu hata terimine ait kovaryans matris $S_{k,ij}$, (5.32) eşitliği kullanılarak Mahalanobis uzaklığı hesaplanabilir.

(5.30)'daki Mahalanobis uzaklığı yenilenme kovaryans matris tersi ile yenilenme teriminin sağdan kendisi soldan ise devriğiyle çarpımı sonucu elde edilir. Daha sonra elde edilen bu değerin güvenilirlik aralığı $(1-\alpha)$ ve özgürlük derecesi d olarak tanımlanan Chi-kare dağılımı içinde kalıp kalmadığına bakılır. Burada d, $h_{k,j}(x_k)$ vektörünün boyutudur. Şekil 5.12'de Chi-kare dağılımında özgürlük derecesi 2 ve güvenilirlik aralığı $0.95\,(\%95)$ seçildiğinde dağılımın [0,5.99] aralığında değiştiği görülmektedir [24]. Eğer Mahalanobis uzaklığı bu testi başarıyla geçerse sonar ölçüm $z_{k,i}$, söz konusu yer gösterici ile eşleştirilir.

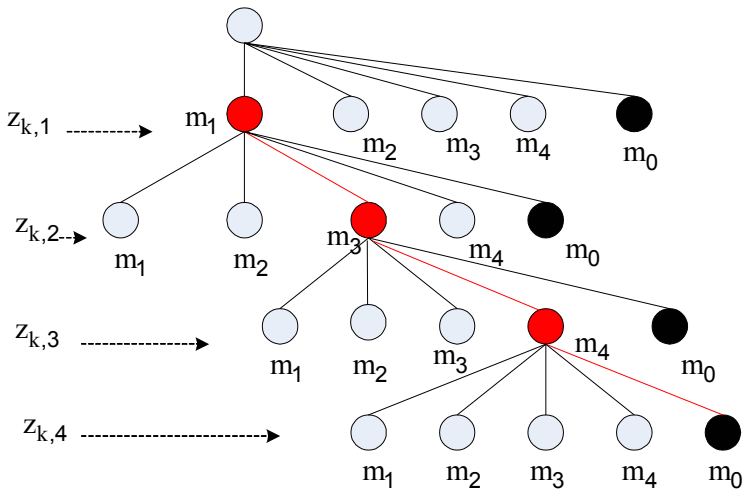
Şekil 5.12 Chi-kare olasılık yoğunluk fonksiyonu $\chi^2_{2,0.95}$

Şekil 5.13'de gösterilen "Bireysel Uyumluluk ile En Yakın Komşuluk" (BUEYK) algoritmasına ve Mahalanobis uzaklık ölçümü kullanılarak yukarıda bahsedilen eşleştirmeler yapılabilir. Bu algoritma her bir $z_{k,i}$ ölçüm ile her bir yer gösterici m_j arasındaki en küçük Mahalanobis uzaklığın [0,5.99] aralığında kaldığı eşleşmeleri bulur. (5.23)'de tanımlanan $H_k = [1\,3\,4\,0]$ matrisi için, BUEYK algoritması kullanıldığında eşleştirme ile ilgili ağaç yapısı şekil 5.14'de gösterilmiştir.

1: **procedure** BUEYK ($z_k, h_k, H_k, P_k^-, R_k$)
2: **for** $i \leftarrow 1$, m do
3: $D^2_{min} \leftarrow MD(z_{k,i}, h_{k,1}, R_{k,i})$
4: nearest $\leftarrow 1$
5: **for** $j \leftarrow 2$, N do
6: $D^2_{ij} \leftarrow MD(z_{k,i}, h_{k,j}, R_{k,i})$
7: **if** $D^2_{ij} < D^2_{min}$ then
8: nearest $\leftarrow j$
9: $D^2_{min} \leftarrow D^2_{ij}$
10: **end if**
11: **end for**
12: **if** $D^2_{min} \leq \chi^2_{d,1-\alpha}$ then
13: $H_i \leftarrow$ nearest
14: **else**
15: $H_i \leftarrow 0$
16: **end if**
17: **end for**
18: **return** *H*
19: **end procedure**

20: **procedure** MD ($z_k, h_k, H_k, P_k^-, R_k$)
21: $v_k = z_k - h_k$ **do**
22: $S_k = H_k P_k^- H_k^T + R_k$ **do**
23: **return** $v_k^T . S_k^{-1} . v_k$
24: **end procedure**

Şekil 5.13 BUEYK algoritması



Şekil 5.14 Bireysel Uyumluluk ile En Yakın Komşuluk algoritması ağaç yapısı

k anında Q ($Q \leq N$) adet eşleştirme yapıldığı varsayılırsa, *H* matrisindeki ilgili eşleştirmelere göre GKS'nin ölçüm modelinde kullanılan matris boyutları aşağıda gibi değişir.

$$z_k = \begin{bmatrix} z_1 \\ \vdots \\ z_Q \end{bmatrix}_{2Qx1} = \begin{bmatrix} h_1 \\ \vdots \\ h_Q \end{bmatrix}_{2Qx1} + \begin{bmatrix} v_1 \\ \vdots \\ v_Q \end{bmatrix}_{2Qx1} \tag{5.33}$$

$$R_k = \begin{bmatrix} R_1 & \cdots & 0 \\ & \ddots & \\ 0 & & R_Q \end{bmatrix}_{2Qx2Q} \tag{5.34}$$

$$H_k = \begin{bmatrix} H_1 \\ \vdots \\ H_Q \end{bmatrix}_{2Qx(2N+3)Q} \tag{5.35}$$

Böylece Kalman kazancının güncellenmesinde kullanılan yenileme kovaryans matrisi (5.36)'daki gibi elde edilir.

$$S_k = H_k P_k^- H_k^T + R_k \tag{5.36}$$

Kalman kazancı (5.37) eşitliğindeki gibi hesaplanır.

$$K_k = P_k^- H_k^T S_k^{-1} \tag{5.37}$$

Bu hesaplamalar sonucunda, bir sonraki durum kestirimi (5.38)'deki Kalman kazancının hata terimi ile çarpıldıktan sonra öngörülen duruma eklenmesi ile bulunur.

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - h_k) \tag{5.38}$$

5.3.2 Yeni Yer Göstericilerin Sisteme Eklenmesi

Ortam üzerinde yeni bir yer gösterici tespit edildiğinde $m_{yeni} = \begin{bmatrix} r_m & \theta_m \end{bmatrix}$, durum vektörüne eklenerek, durum hata kovaryans matrisi güncellenir. Yeni yer göstericinin iki boyutlu düzlemde noktasal konumu (5.39) eşitliği ile ifade edilebilir. (5.40) ve (5.41) eşitlikleri sırasıyla genişletilmiş durum vektörü ve genişletilmiş durum hata kovarysans matrisini göstermektedir.

$$m_{N+1} = L(x_r, m_{yeni}) = \begin{bmatrix} x_{r[x]} + r_m \cos(x_{r[\theta]} + \theta_m) \\ x_{r[y]} + r_m \sin(x_{r[\theta]} + \theta_m) \end{bmatrix} \tag{5.39}$$

$$x_k \leftarrow \begin{bmatrix} x_k \\ m_{N+1} \end{bmatrix} \tag{5.40}$$

$$P_{k,k} = \begin{bmatrix} P_{rr} & P_{r1} & \cdots & P_{rN} & \mathbf{P_{rN+1}} \\ P_{1r} & P_{11} & & P_{1N} & \mathbf{P_{1N+1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ P_{Nr} & & & P_{NN} & \\ \mathbf{P_{N+1r}} & \mathbf{P_{N+11}} & \cdots & & \mathbf{P_{N+1N+1}} \end{bmatrix} \tag{5.41}$$

$$P_{N+1N+1} = J_{xr} P_{rr} J_{xr}^T + J_z R J_z^T = \begin{bmatrix} . & . \\ . & . \end{bmatrix}_{2x2} \tag{5.42}$$

.

$$P_{rN+1} = P_{N+1r}^T = P_{rr} J_{xr}^T = \begin{bmatrix} . & . \\ . & . \\ . & . \end{bmatrix}_{3x2} \tag{5.43}$$

$$P_{N+1i} = P_{iN+1}^T = J_{xr} P_{ri}^T = \begin{bmatrix} . & . \\ . & . \end{bmatrix}_{2x2} \quad (i=1,2\ldots,N) \tag{5.44}$$

(5.42), (5.43) ve (5.44) eşitlikleri kullanılarak (5.41)'deki durum hata kovaryans matrisinin son satır ve son sütundaki alt kovaryans matrisleri elde edilmiş olur.

J_{xr} ve J_z Jacobian matrisleri (5.45) ve (5.46) eşitliklerinde olduğu gibi sırayla $L(x_r, m_{yeni})$'nin robot (x_r) ve yer gösterici (m_{yeni}) durum değişkenlerine göre kısmı türevlerinin alınmasıyla hesaplanır.

$$J_{xr} = \frac{\partial L(x_r, m_{yeni})}{\partial x_r} = \begin{bmatrix} 1 & 0 & -r_m \sin(x_{r[\theta]} + \theta_m) \\ 0 & 1 & r_m \cos(x_{r[\theta]} + \theta_m) \end{bmatrix} \tag{5.45}$$

$$J_z = \frac{\partial L(x_r, m_{yeni})}{\partial m_{yeni}} = \begin{bmatrix} \cos(x_{r[\theta]} + \theta_m) & -r_m \sin(x_{r[\theta]} + \theta_m) \\ \sin(x_{r[\theta]} + \theta_m) & r_m \cos(x_{r[\theta]} + \theta_m) \end{bmatrix} \tag{5.46}$$

Şekil 5.15'de yeni bir yer göstericinin ((N+1)'inci yer gösterici) durum vektörüne ilave edilmesi sonrasında güncellenen durum hata kovaryans matrisi görülmektedir. Bu matris içindeki gri renkli bloklar yeni yer göstericiye ait kovaryans matrisleri gösteriri. Matris boyutları (5.42), (5.43) ve (5.44) eşitliklerinde tanımlanmıştır.

P_{rr}	P_{r1}	… … … … … …	P_{rN+1}
P_{1r}	P_{11}	… … … …	P_{1N+1}
… … … … … …	… … … …	… … … …	… … … …
P_{N+1r}	P_{N+11}	… … … …	P_{N+1N+1}

Şekil 5.15 Genişletilmiş durum hata kovaryans matrisi

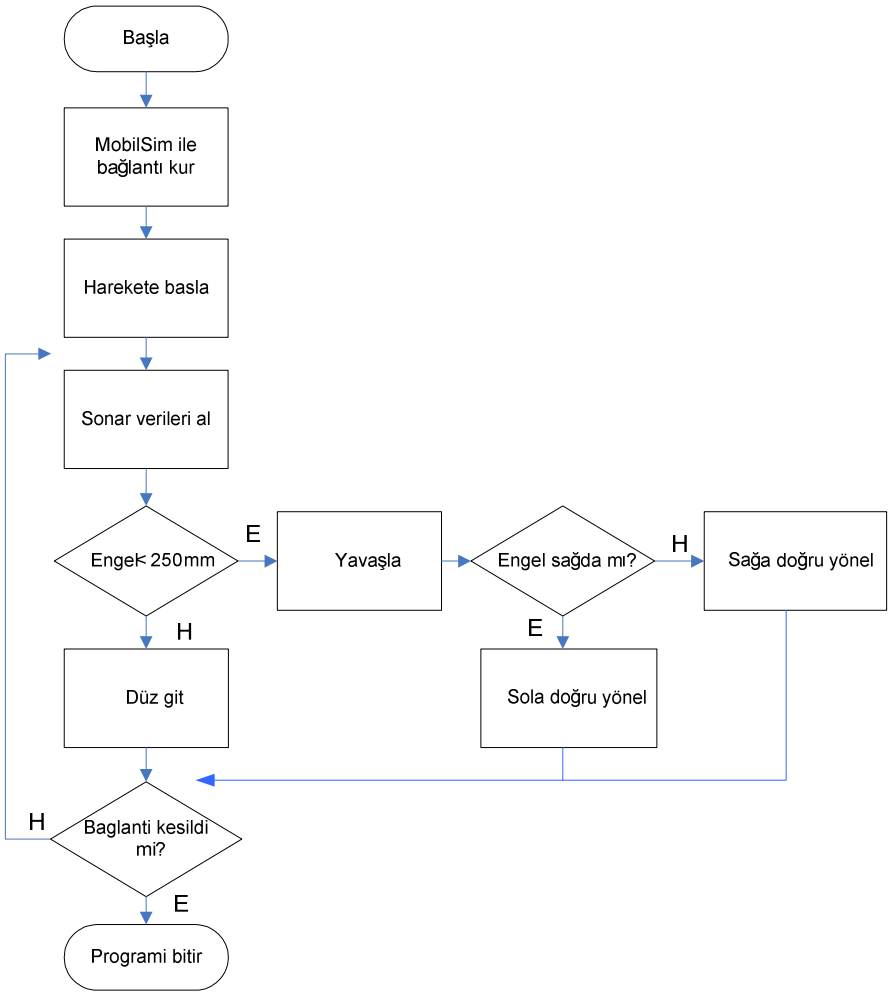
6. UYGULAMALAR

Eş zamanlı konum belirleme ve haritalama uygulamasına yönelik Pioneer robotlar için hazırlanmış MobilSim benzetimcisi ve Matlab programı kullanılarak benzetim tabanlı bir çalışma gerçekleştirilmiştir. Bölüm 6.1'de ÜTB algoritması kullanılarak benzetim ortamında tanımlanan çevrenin öznitelik tabanlı haritasının çıkartılması ile ilgili çalışmalar yer almaktadır. Bu harita ortamdaki kenar nokta konumlarını göstermektedir. Haritadaki her bir kenar nokta robot konumu belirlenirken birer yer gösterici olarak kullanılmıştır. Beşinci bölümde anlatılan sistem modelleri esas alınarak GKS algoritması kullanılarak, yer gösterici ve robot konum kestirimin eş zamanlı olarak yapılması ile ilgili çalışmalar bölüm 6.2'de yer almaktadır. Gerçekleştirilen benzetim çalışmasında robot sistem modelinde hata olmadığı varsayılmıştır. Diğer bir ifade ile robot tekerlerinin çapları, genişlikleri ve aralarındaki mesafe bilgisinde hata olmadığı kabul edilmiştir. Ayrıca robot hareketi iki boyut dışına çıkmamış ve yüzey değişiklikleri olmadığı varsayılmıştır.

Robotun sanal ortam içersinde engellere çarpmadan gezinmesi için ARIA arayüz uygulaması kullanılarak C++ tabanlı bir program yazılmıştır [25].

Bu program ile robot sonar algılayıcılardan elde ettiği mesafe verilerini kullanarak yakınında engel olup olmadığı kararını verir. Engel olmadığı sürece robot yönelim doğrultusunda doğrusal olarak hareket eder. Engel ile robot arasındaki mesafe 25 cm'nin altına düştüğünde robot yavaşlar ve engel olmayan alana doğru yönelir. Programda robotun maksimum hızı 300 mm/s olarak ayarlanmıştır. Program ile ilgili akış diyagramı şekil 6.1 de verilmiştir.

Bu program ile ayrıca robotun gezinimi sırasında 16 adet sonar algılayıcıdan elde edilen mesafe ölçümleri, kodlayıcılardan elde edilen robot konum verileri ile robotun doğrusal ve dönüş hızları 300 ms aralıklarla örneklenerek bir dosya içine kaydedilmiştir. Dosyada kayıtlı olan bu veriler daha sonra Matlab programı aracılığıyla hem ÜTB hem de GKS algoritmalarında kullanılmıştır. Dosyadaki kayıtlı veriler ile ilgili örnek çizelge 6.1, 6.2 ve 6.3'de gösterilmiştir.



Şekil 6.1 Robotun gezinim için yazılan programın akış diyagramı

Çizelge 6.1 Sonar veriler (mm)

Sonar 1	Sonar 2	Sonar 3	Sonar 4	Sonar 4	Sonar 6	Sonar 7	Sonar 8
1278	1122	1002	845	829	425	606	452

Sonar 9	Sonar 10	Sonar 11	Sonar 12	Sonar 13	Sonar 14	Sonar 15	Sonar 16
442	575	921	1206	1770	1755	1773	1288

Çizelge 6.2 Robot konumu

Robot Konumu		
x (mm)	y (mm)	θ(derece)
-460.75	420.01	58.98

Çizelge 6.3 Robotun doğrusal ve dönüş hızı

Doğrusal Hız (mm/sn)	Dönüş hızı (derece/sn)
185.5	13

Çizelge 6.3 ve 6.2'e bakıldığında robotun konumu (-460.75, 420.01, 58.98^{o}) iken doğrusal hızı 185.5 mm/sn, dönüş hızı ise 13 derece/sn dir. Robotun bu konumdayken sonarlardan okuduğu veriler mm cinsinden çizelge 6.1'de görülmektedir.

Her bir örnek arasındaki süre ise MobilSim benzetimcisi ile bağlantı kurulduktan sonra her bir sonraki örneklemenin alındığı zamandan şimdiki örnekleme zamanın çıkarılmasıyla elde edilmiştir. Çizelge 6.4'de 300ms aralıklarla alınan örneklerin başlangıç anından itibaren örnekleme zamanları görülmektedir.

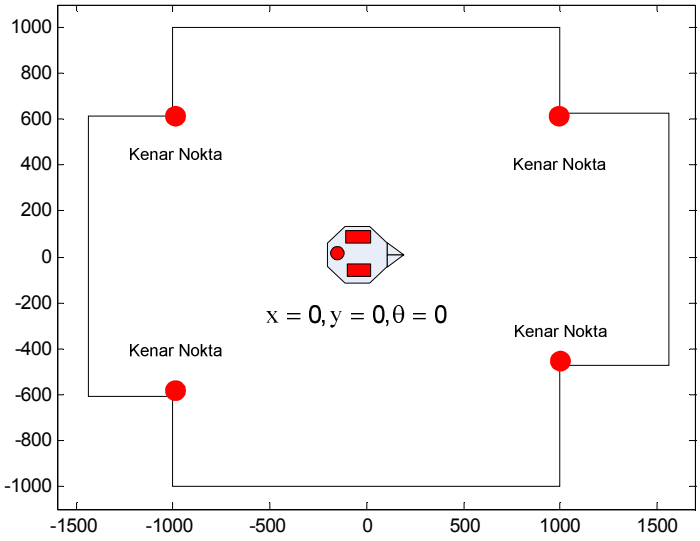
$$\Delta t = t_{samp} = 27980 - 27669 = 285\text{ms} \ (\cong 300\text{msn})$$

Çizelge 6.4 Örnekleme zamanı

Süre (msn)
0
⋮
27669
27980
⋮

6.1 Üçgenleme Tabanlı Birleşim Algoritması ile Öznitelik Tabanlı Harita Oluşturma

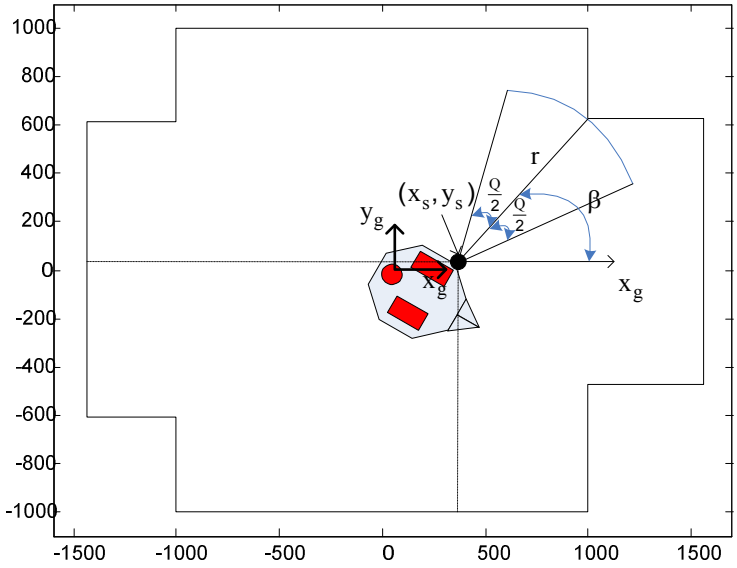
Bölüm 3.2'de bahsedilen ÜTB algoritmasının uygulamasını yapabilmek için Mapper3-Basic programı ile şekil 6.2'deki gibi sanal bir ortam oluşturulmuştur (3x2 metre boyutlarında boş bir oda). MobilSim benzetimcisi ile robot sanal ortam üzerinde şekil 6.1'deki robot gezinim programı kullanılarak dolaştırılmış ve çizelge 6.1-6.4'de örnek olarak gösterilen veriler dosyaya kaydedilmiştir. Daha sonra ortamdaki 4 adet dikey kenar konumlarını tespit etmek için standart ÜTB algoritması Matlab programı kullanılarak test edilmiştir (çevrim dışı).



Şekil 6.2 3x2 m^2 boyutunda tanımlanan boş sanal ortam

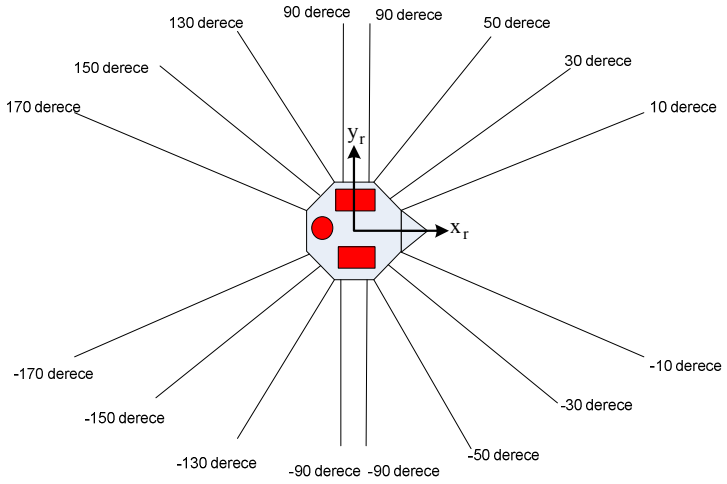
Öncelikle robotun hareketi esnasında elde ettiği sonar verilerin gruplandığı hareketli pencere oluşturulmuştur. Bunun için 16x80'lik iki boyutlu bir matris tanımlanmıştır. Bölüm 3.2 de gösterilen her bir hücre 1x4'lük vektörlerle ifade edilmiştir. Robot üzerindeki 16 sonardan da ölçüm alındığı için pencere satır sayısı (m) 16, her hücre 1x4'lük vektörden oluştuğu için pencere sütun sayısı (n) 80 olarak seçilmiştir.

Her bir hücredeki sonar yayların tanımlanabilmesi için örneklemenin yapıldığı andaki sonar koordinatlarına (x_s,y_s) ve sonar merkezinin bütünsel yatay düzlem ile yaptığı açı (β) bilgisine ihtiyaç vardır (şekil 6.3).



Şekil 6.3 Sonar ölçümün ortam üzerindeki modeli

Bu bilgilerin elde edilmesi için her bir sonarın robot merkezine göre konumlarının ve yönelim açılarının bilinmesi gerekmektedir (şekil 6.4). Pioneer 3DX robot üzerinde yer alan sonar konumları çizelge 6.5 gösterilmiştir.



Şekil 6.4 Robot sonar ağı

Çizelge 6.5 Pioneer 3-DX Sonar Konumları

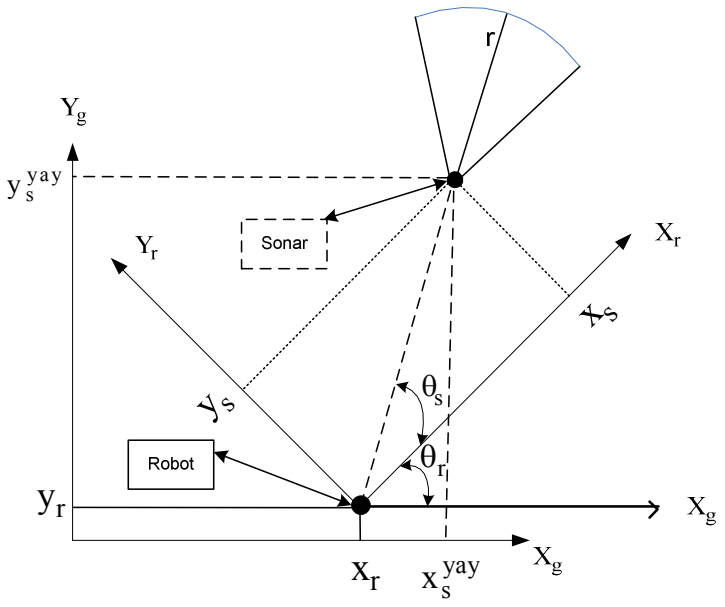
Sonar Numarası	X_s(mm)	Y_s(mm)	θ_S (derece)
1	69	136	90
2	114	119	50
3	148	78	30
4	166	27	10
5	166	-27	-10
6	148	-78	-30
7	114	-119	-50
8	69	-136	-90
9	-157	-136	-90
10	-203	-119	-130
11	-237	-78	-150
12	-255	-27	-170
13	-255	27	170
14	-237	78	150
15	-203	119	130
16	-157	136	90

Aşağıdaki döşüm formülleri kullanılarak her bir sonar ölçüm için modellenen yayın merkez konumu ve merkez doğrultusun yatay eksen ile yaptığı açı hesaplanmıştır (şekil 6.5).

$$x_{s_i}^{yay} = x_{robot} + x_{s_i}\cos\theta_r - y_{s_i}\sin\theta_r, \qquad i = 1,2,...16 \tag{6.1}$$

$$y_{s_i}^{yay} = y_{robot} + x_{s_i}\sin\theta_r + y_{s_i}\cos\theta_r, \tag{6.2}$$

$$\theta_{s_i}^{yay} = \theta_{robot} + \theta_{s_i} \tag{6.3}$$



Şekil 6.5 Yay konumu ve görüş alanı

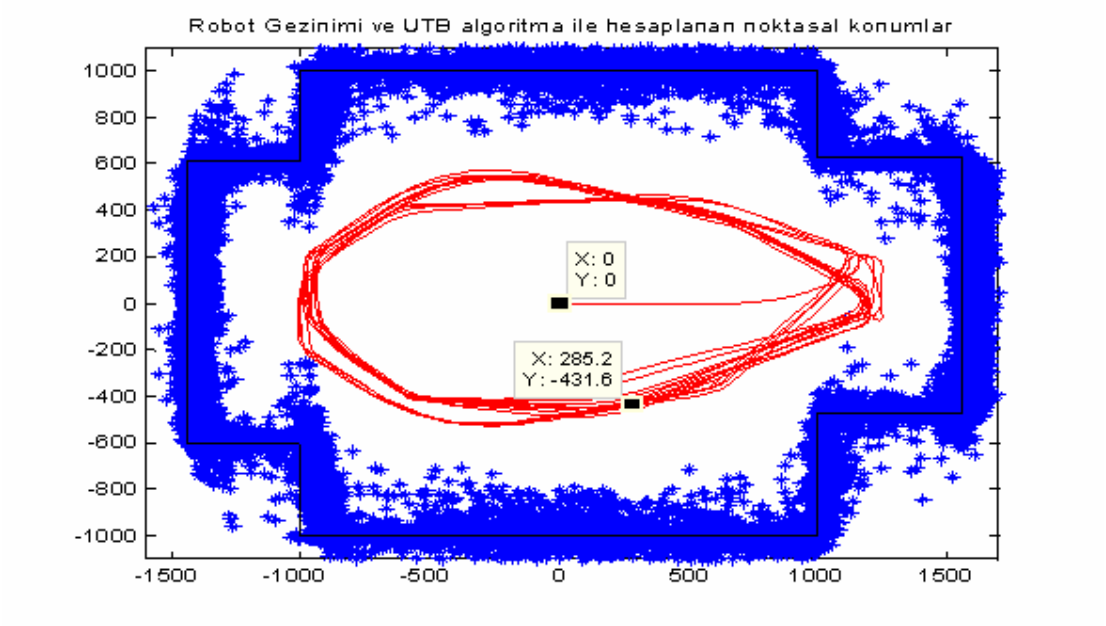
Çizelge 6.6'da yer alan eşik değerleri yapılan deneysel çalışmalar sonucunda belirlenmiş ve algoritmada bu değerler kullanılmıştır.

Çizelge 6.6 Eşik değerleri

Simge	Değer
d_1	300 mm
d_2	35 mm
n_t	6

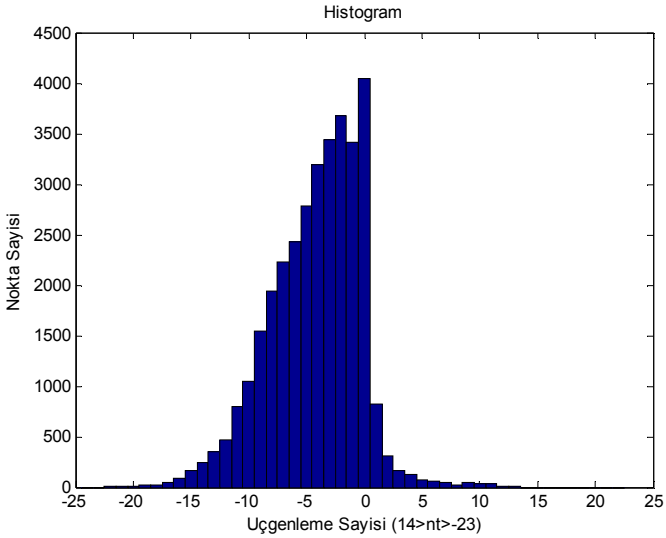
Şekil 6.6'da robotun gezinim sırasında yapmış olduğu hareket görülmektedir. Robotun harekete başladığı ilk konum ve son konum işaretlenmiştir. Haritadaki 33696 adet nokta ÜTB algoritması ile belirlenen noktasal konumları göstermektedir. Bazı ölçümlerin ortam sınırı dışında bazılarının ise engel olmamasına rağmen engel varmış gibi ortam içinde çıktığı görülmektedir. Bunun nedeni ÜTB algoritmasının sadece kenar noktaları tespit edebiliyor olmasıdır.

Dolayısıyla düzlem üzerinden alınan ölçümlerdeki konumsal sapmalar çok fazla olmuştur.



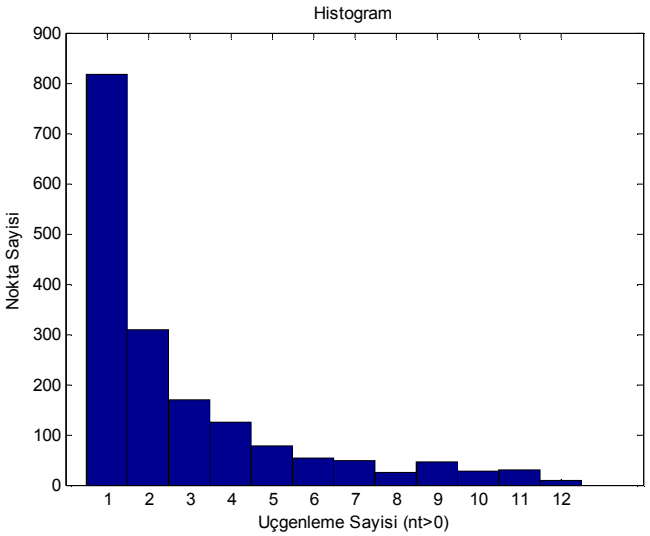
Şekil 6.6 Robot gezinimi ve ÜTB algoritma ile hesaplanan noktasal konumlar

Şekil 6.7'de gezinim sırasında çıkartılan noktaların kesişim sayılarına göre histogramı yer almaktadır. Negatif işaretli üçgenleme sayıları düzlemsel bölgeleri, pozitif işaretliler ise kenar potansiyelli bölgeleri temsil etmektedir. Kenar veya düzlemsel bölge üzerinden elde edilen noktalar bu işaret değişikliği ile sınıflandırılır. Buna göre histogramın sol yarım ekseninde kalan dağılım, sağ yarım eksene göre oldukça yoğundur. Bunun nedeni ortam içindeki düzlemsel bölgelerden elde edilen noktaların, kenar bölgelerden elde edilenlere göre çok daha fazla sayıda olmasıdır.



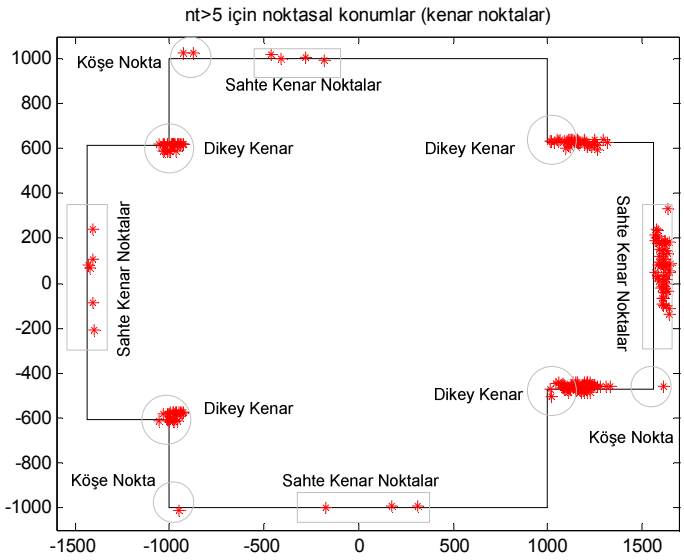
Şekil 6.7 Noktasal histogram ($13 \geq n_t \geq -22$)

Kesişim sayılarının pozitif olduğu noktalar için ($n_t \geq 1$) histogram Şekil 6.8'de yer almaktadır. Kenar noktaları tespit etmek için en az 6 üçgenleme ($n_t \geq 6$) sonunda hesaplanan noktasal konumlar göz önünde bulundurulmuştur.



Şekil 6.8 Noktasal histogram ($n_t \geq 1$)

Şekil 6.9'da kesişim sayısı 6 veya daha fazla olan noktalar için öznitelik tabanlı harita çizdirilmiştir. Bu haritaya göre ortam içindeki dikey kenarlar gezinim sırasında tespit edilmiştir. Ancak haritada kenar bölgeler dışında, düzlemsel bölgeler (ortamdaki duvarlardan) üzerinde de sahte kenar noktalar yer almaktadır. Sahte kenar noktalardan kurtulmak için temel ÜTB algoritmasında iyileştirme yapılmıştır.



Şekil 6.9 $n_t \geq 6$ için kenar nokta konumları

6.1.1 Geliştirilmiş Üçgenleme Tabanlı Birleşim Algoritması

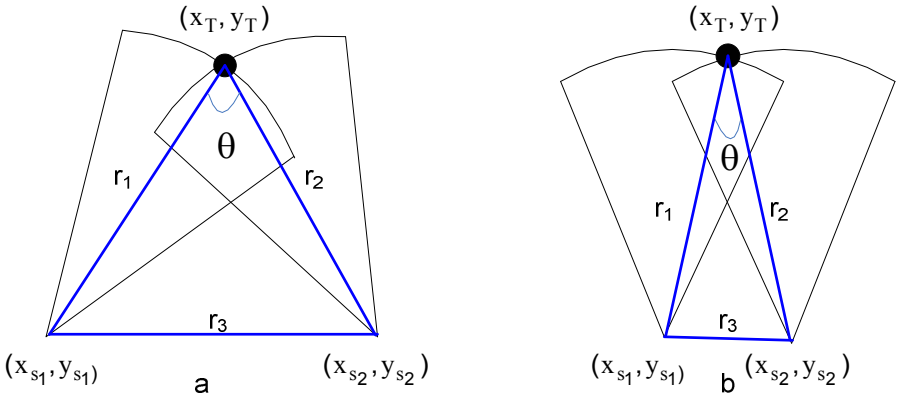
Algoritmanın performansı artırmak ve haritadaki sahte kenar noktaları temizlemek için standart ÜTB algoritmasına iki ek süreç ilave edilmiştir.

6.1.1.1 Kararlı kesişim koşulu

Kararlı kesişim iki sonar yay arasında kalan açının belirlenen bir eşik değerinin üzerinde olduğu durum olarak tanımlanır [13].

$$\theta_{kesisim} > \theta_{esik}$$

Kararlı kesişim koşulunun kullanılmasının kenar nokta belirlemede iki önemli avantajı vardır. Birincisi bu koşulu sağlayan yaylar arasındaki kesişim noktaları sonar algılayıcı kaynaklı hatalardan (mesafe ölçümlerinde veya sonar konumlarından) önemli ölçüde etkilenmeyecektir (şekil 6.10a). Buna karşın kararsız kesişim koşulu ile tespit edilen noktasal konumlar, algılayıcı kaynaklı hatalar nedeniyle önemli ölçüde etkilenebilir (şekil 6.10b). Bu yüzden kararlı kesişim koşulu kullanılarak elde edilen noktalar sonar algılayıcı hatalarına karşı daha güvenilirdir. İkincisi ise düzlemsel bölgeler üzerinde tespit edilen sahte noktaları filtrelemektir. Düzlem üzerinde tespit edilen noktalar kararsız kesişim koşulu ile tespit edilmiş olabilir.



a) Kararlı kesişim **b)** Kararsız kesişim

Şekil 6.10 Kararlı kesişim koşulu

Kesişim açısını hesaplamak için kosinüs teoreminden yararlanılmıştır. Bunun için gerekli olan üçgen taban uzunluğu r_3 (3.6) eşitliğiyle elde edilmiştir. (6.4)'deki kosinüs teoremi kullanılarak θ (kesişim açısı) açısı bulunmuştur.

$$\theta = \cos^{-1}\left(\frac{r_1^2 + r_2^2 - r_3^2}{2r_1r_2}\right) \tag{6.4}$$

6.1.1.2 Etkin hareketli pencere güncellemesi

Temel ÜTB algoritmasında yapılan ikinci iyileştirme, kenar nokta konumu belirlemek için etkin hareketli pencere güncellemesidir. Standart ÜTB algoritmasında hareketli pencere güncellemesi sensör konumlarının belirli bir eşik seviyesinin üstünde yer değiştirmesi sonucunda yapılmaktadır.

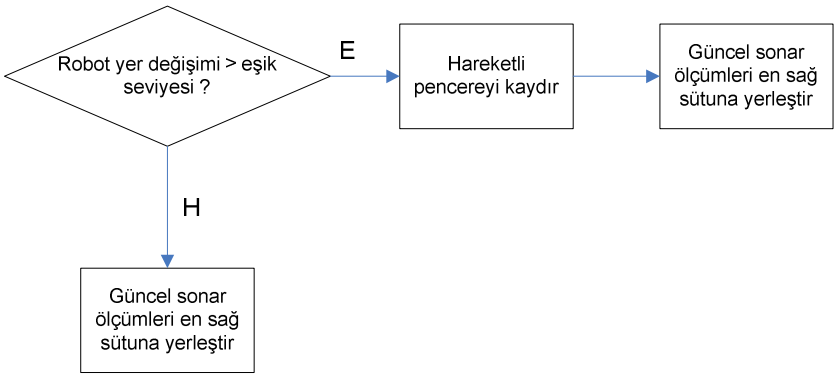
$$\sqrt{(x_{s_{yeni}} - x_{s_{önceki}})^2 - (y_{s_{yeni}} - y_{s_{önceki}})^2} > d_t \qquad (6.5)$$

Ancak, bu yaklaşım daha az sayıda kenar nokta tespit edilmesine veya hiç tespit edilememesine neden olabilir. Özellikle bu yaklaşım robotun kendi ekseni etrafında yaptığı döngüsel hareket sonucu daha az sayıda kenar nokta tespit etme ihtimalini artırabilir. Robot sabit bir pozisyon etrafında döndüğü zaman hareketli pencere sürekli olarak güncellenir. Bunun nedeni bu hareket sonucunda sensör konumlarının değişecek olmasıdır. Böylece hareketli pencere içindeki tüm sensör ölçümleri sabit bir robot konumundan elde edilmiş olur. Bu da kenar noktaların daha az kesişim sayısıyla tespit edilmesine neden olabilir. Böylece kenar noktaların tespiti doğru olarak yapılamayabilir.

Hareketli pencere güncellemesinin daha etkin bir şekilde yapılması için, sensör konumlarına göre değil robotun yer değişimine bakılarak yapılmalıdır [13].

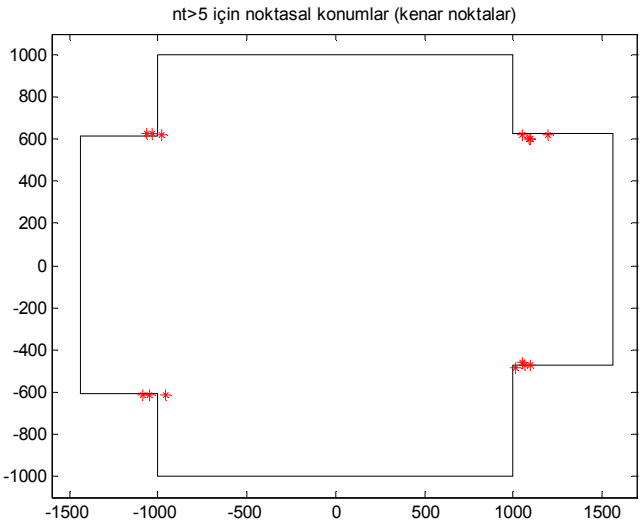
$$\sqrt{(x_{R_{yeni}} - x_{R_{önceki}})^2 - (y_{R_{yeni}} - y_{R_{önceki}})^2} > d_t' \qquad (6.6)$$

Eğer robotun bir önceki ve bir sonraki konumu arasıdaki mesafe belirlenen eşik değerinin altında kalırsa, hareketli penceredeki güncel tarama yenisi ile değiştirilir. Hareketli pencere üzerinde kaydırma işlemi yapılmaz (şekil 6.11). Bu sayede özellikle robotun merkez ekseni etrafında yaptığı döngüsel hareketler sonrasında daha fazla sayıda kenar nokta tespit edilebilir [13].

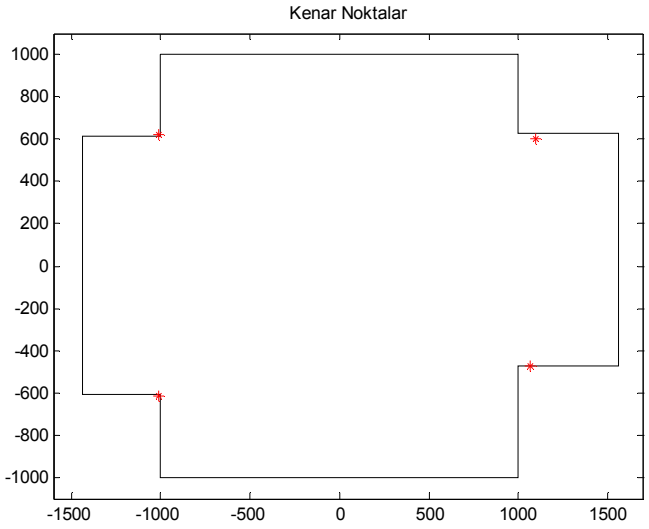


Şekil 6.11 Etkin pencere güncellemesi akış diyagramı

Yapılan deneysel çalışmalar sonucunda tanımlanan sanal ortam için $\theta_{eşik}$ = 62 ^{o} ve $d_t^{'}$ = 2.5 cm olarak seçildiğinde ortamdaki sahte kenar noktalar temizlenmiş ve gerçek kenar noktalar doğru şekilde tespit edilebilmiştir. Şekil 6.12'de kenar bölgeler üzerinde tespit edilen noktalar yer almaktadır. Harita üzerindeki noktalar ortamdaki dikey kenarların konumlarını göstermektedir. Bu noktasal konumların ortalamaları alındığında ortamdaki kenar bölge konumları Şekil 6.13'de gösterildiği gibi belirlenmiştir.



Şekil 6.12 Kenar bölge üzerinde tespit edilen noktalar



Şekil 6.13 Kenar nokta konumları

Çizelge 6.7'de sanal harita üzerindeki kenar noktaların gerçek konumları mm cinsinden yer almaktadır. Konum hataları yüzdesel olarak (6.7), (6.8), (6.9) eşitlikleri kullanılarak hesaplanmıştır. Dört kenar nokta için gerçek ve hesaplanan konum değerleri çizelge 6.7'de karşılaştırılmıştır.

Çizelge 6.7 Konum karşılaştırması

	Yatay Eksen (x)		Düşey Eksen (y)	
	Gerçek Konum (mm)	Hesaplana Konum (mm)	Gerçek Konum (mm)	Hesaplana Konum (mm)
Sol Üst Kenar	-1000	-1012	616	621.8
Sol Alt Kenar	-998	-1014	-605	-615.4
Sağ Alt Kenar	1002	1064	-471	-474.1
Sağ Üst Kenar	999	1093	625	603.2

$$\text{Yatay Eksen Hata Oranı} = \frac{x_{\text{Gerçek}} - x_{\text{Ölçülen}}}{x_{\text{Gerçek}}} \%100 \qquad (6.7)$$

$$\text{Düşey Eksen Hata Oranı} = \frac{y_{\text{Gerçek}} - y_{\text{Ölçülen}}}{y_{\text{Gerçek}}} \%100 \quad (6.8)$$

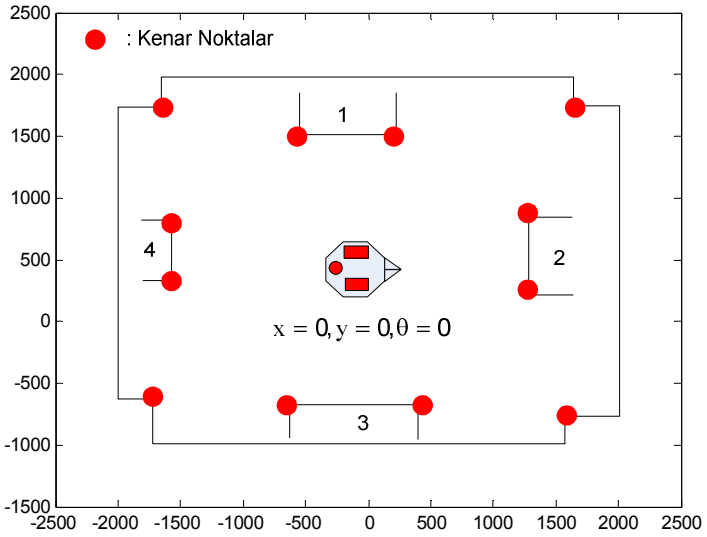
$$\text{Ortalama Hata} = \frac{\text{YEHO} + \text{DEHO}}{2} \quad (6.9)$$

Çizelge 6.8 Konumsal hata oranları

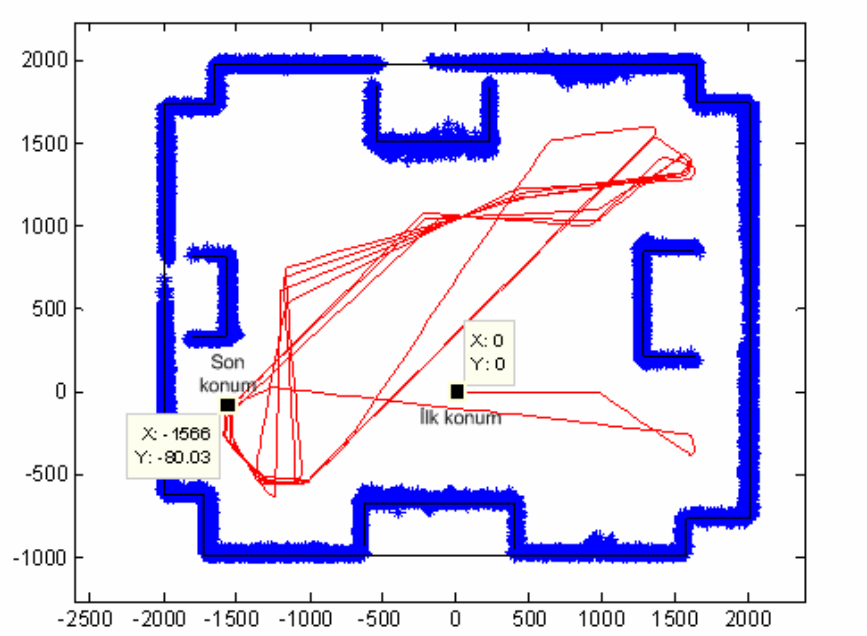
	YEHO %	**DEHO %**	**Ortalama Hata %**
Sol Üst Kenar	1.2	0.9	1.05
Sol Alt Kenar	1.6	1.7	1.65
Sağ Alt Kenar	6.2	0.7	3.45
Sağ Üst Kenar	9.4	3.4	6.4

Sonuç olarak çizelge 6.8'e bakıldığında ÜTB algoritmasıyla ortamdaki dört kenar bölgenin noktasal konumları minimum % 1.05, maksimum % 6.4 ortalama hata oranıyla tespit edilmiştir.

GÜTB algoritması kullanılarak daha karmaşık ortamlarda da çalışmalar yapılmıştır. Bunun için şekil 6.14'deki gibi 4x3 m^2 boyutunda içinde dört adet dikdörtgen yapıda engel bulunan başka bir sanal ortam tanımlanmıştır. Ortam içinde toplam 12 adet kenar nokta yer almıştır. Hareketli pencere boyutu başlangıçta 16x20 olarak seçildiğinde, robotun üzerindeki mevcut sonar sayısının yetersiz kalması ve asimetrik dizilim gibi nedenlerden dolayı birçok kenar noktaların tespit edilemediği görülmüştür. Bunun nedeni tespit edilemeyen kenar noktalar üzerinden yeterli sayıda ölçüm alınamamış olmasıdır. Kenar noktalar üzerinden ne kadar fazla ölçüm alınırsa algoritmanın o oranda kenar nokta belirleme ihtimali artacaktır [26]. Çok daha fazla kenar noktanın belirlenebilmesi ya mevcut sonar sayısının artırılarak ölçüm çözünürlüğünün yükseltilmesiyle yada hareketli pencerenin genişletilmesiyle sağlanabilir. Ancak bu durum algoritmanın çalışma zamanını doğrudan artıracağı için gerçek zamanlı uygulamalara kısıtlama getirecektir. Bu yüzden yapılacak uygulamanın gerçek zamanlı olup olmamasına göre pencere boyutu en uygun boyutta seçilmelidir.



Şekil 6.14 4x3 m^2 'lik karmaşık sanal ortam

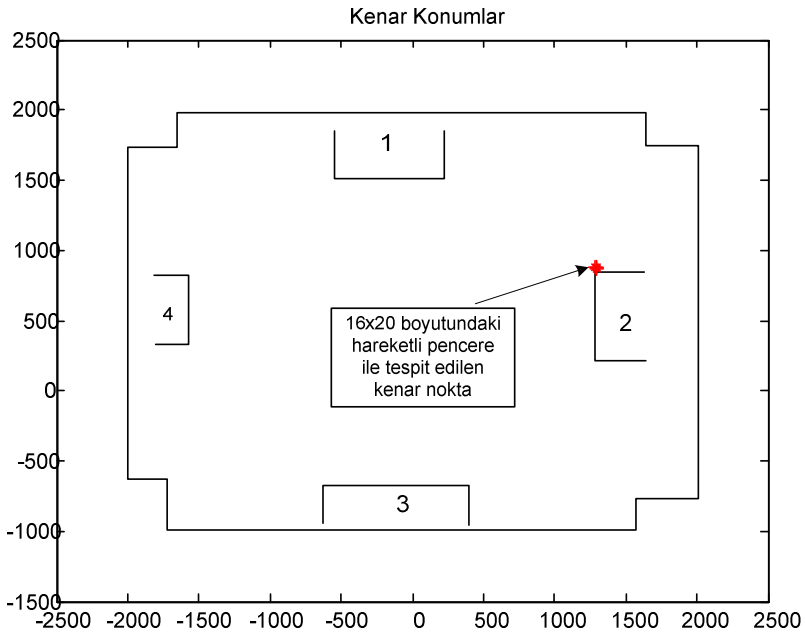


Şekil 6.15 Robot gezinimi ve GÜTB algoritması hesaplanan noktasal konumlar

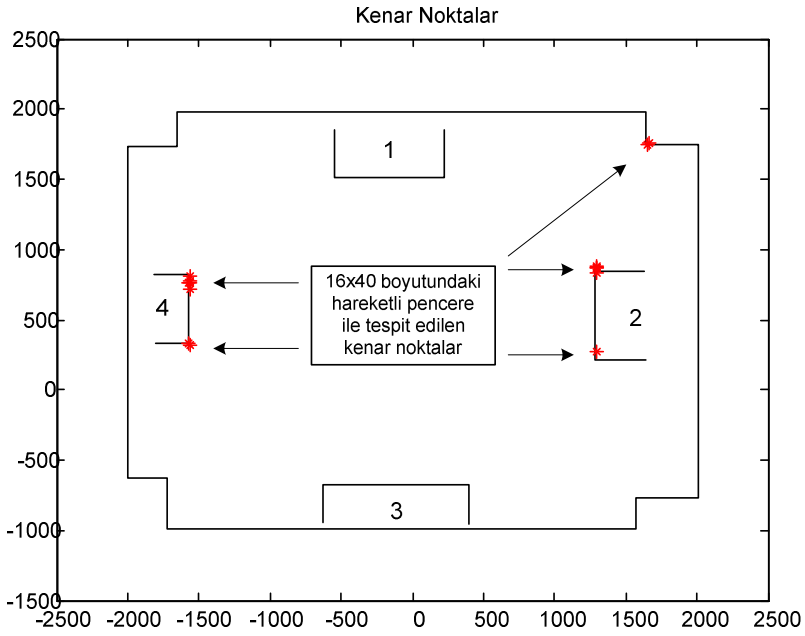
Şekil 6.15'de robotun karmaşık sanal ortam içindeki gezinimi görülmektedir. Robot (0,0) konumundan harekete başlayarak 47204 mm yol aldıktan sonra (1566,-80.03) konumunda gezinimi sonlandırılmıştır.

Şekil 6.16'de GÜTB algoritmasıyla çizelge 6.9'da yer alan eşik değerleri kullanıldığında sadece bir kenar noktanın tespit edilebildiği görülmüştür. Bunun en önemli nedeni robotun gezinim sırasında kenar bölge üzerinden aldığı sonar ölçümlerin belirlenen eşik değerlerine göre yeterli sayıda olmamasıdır. Bu ölçümler hareketli pencere içinde depolandığından pencere boyutu genişletilerek daha fazla ölçüm arasından daha çok kenar nokta tespit edilebilir.

Şekil 6.16'de tanımlanan ortam üzerinde sadece 2 numaralı engele ait kenarlar biri tespit edilebilmiştir. Daha fazla kenar nokta tespit etmek için pencere sütün sayısı (n) iki katına çıkartılarak tekrar denemeler yapılmıştır. 16x40'lık bir pencere tanımlandıktan sonra (robotun ortam üzerinden aldığı son 40 ölçüm ile) GÜTB algoritması kullanıldığında daha fazla sayıda kenar nokta tespit edilmiştir (Şekil 6.17).



Şekil 6.16 Karmaşık ortamda tespit edilen kenar nokta



Şekil 6.17 Genişletilmiş pencere boyutu ile tespit edilen kenar noktalar

Çizelge 6.9 Pencere boyutu ve eşik değerlerler

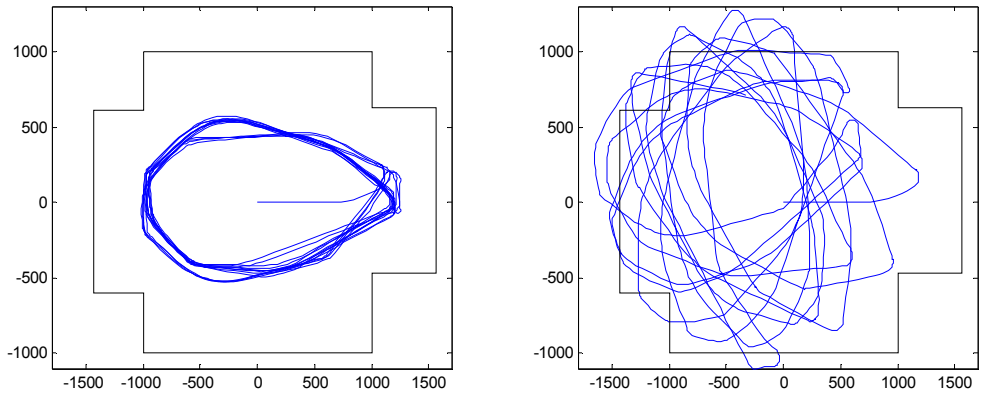
		Pencere Boyutu	
		16x20	16x40
Kesişim sayısı	n_t	6	6
Maksimum sapma	max_{sapma}	35mm	35mm
Kesişim açısı	$\theta_{eşik}$	67 derece	67 derece
Robot yer değişimi	$d_t^{'}$	3.5 cm	3.5cm
Tespit edilen kenar sayısı		1	5

Literatürdeki ÜTB algoritması ile yapılan uygulamalarda 24x10'luk pencereler ile çalışılmıştır [3,12]. Bu tez çalışmasında pencere sütun sayısı 20 seçildiğinde tanımlanan boş sanal ortamdaki kenar noktaların hepsi tespit edilmiştir. Ortam daha büyük ve daha çok kenar noktadan oluşan daha karmaşık bir yapıda tanımlandığında ise 16x20'lik pencere boyutu ve çizelge 6.9'da tanımlanan eşik değerleri ile yalnızca 1 kenar noktanın tespit edildiği görülmüştür. Pencere boyutu 16x40'lık olarak genişletildiğinde toplam 12 kenar noktadan 5'i tespit edilmiştir.

6.2 Eş Zamanlı Konum Belirleme Ve Haritalama Uygulaması

Robot konumunun ve bölüm 6.1'de elde edilen yer gösterici konumlarının eş zamanlı olarak kestirimi için GKS'den faydalanılmıştır. Bölüm 5'de açıklanan sistem modeli aynen kullanılmış ve sonar algılayıcılarından elde edilen ölçümler ile yer göstericiler arasındaki eşleştirmeler Mahalanobis uzaklığı yaklaşımı ile BUEK algoritması kullanılarak ilişkilendirilmiştir.

Benzetim ortamında yapılan ikinci çalışma robot konumunun odometrik veriler kullanılarak izlenmesi olmuştur. Odometrik veriler çizelge 6.3 ve 6.4'de örnek olarak gösterilen robotun doğrusal ve dönüş hızlarıdır. Bu hız değerleri (5.4) ve (5.5) eşitliklerine göre örnekleme süresiyle çarpıldığında sistem denetim girdileri elde edilmiş olur. Sistem modelinde belirtilen denetim girdileri sayesinde robot konumunun izlenmesi sağlanmıştır.



a **b**

Şekil 6.18 a Robotun gerçek gezinimi
b Robotun odometrik gezinimi

Robot benzetim ortamında 656527 sn sürede toplam 69276 mm mesafe yol almıştır. Bu süre sonunda robot gerçekte ortam üzerinde (232 mm, -442.8 mm, -167.96^0) konumuna gelmiştir. Bu gezinim sırasında robotun yapmış oluğu hareket ise şekil 6.18 a'da görülmektedir. Robotun konumunun izlenmesi odometri ile yapıldığında robotun ortam üzerinde geldiği son konum (-327 mm, 710 mm, -15^0) dir. Robotun odometrik verilere göre gezinimi ise şekil 6.18 b'de görülmektedir.

Çizelge 6.10 Gerçek ve odometrik son konum değerleri

	X(mm)	Y(mm)	θ (derece)
Gerçek Son Konum	285.18	-431.65	-164.97
Odometrik Son Konum	-327.72	710.08	-15.70

$$\text{Konum Hata Oranı} = \frac{\text{gerçek son konum} - \text{odometrik son konum}}{\text{toplam alınan yol}}.100\% \qquad (6.10)$$

$$\text{Yönelim Hata Oranı} = \left|\frac{\text{gerçek yönelim} - \text{odometrik yönelim}}{\text{gerçek yönelim}}\right|.100\% \qquad (6.11)$$

$$\text{Ortalama Hata Oranı} = \frac{\text{Konum Hata Oranı} + \text{Yönelim Hata Oranı}}{2} \qquad (6.12)$$

Çizelge 6.10'da robot son konumlarının gerçek ve odometrik karşılaştırması yer almaktadır. Odometriye dayalı konum ve yönelim hatalarının yüzdesel ifadeleri (6.10) ve (6.11) eşitlikleri kullanılarak hesaplanmıştır. Ortalama hata oranı ise konum ve yönelim hata oranlarının ortalaması alınarak belirlenmiştir (6.12). Bu durumda hesaplanan hata oranları aşağıdaki gibidir.

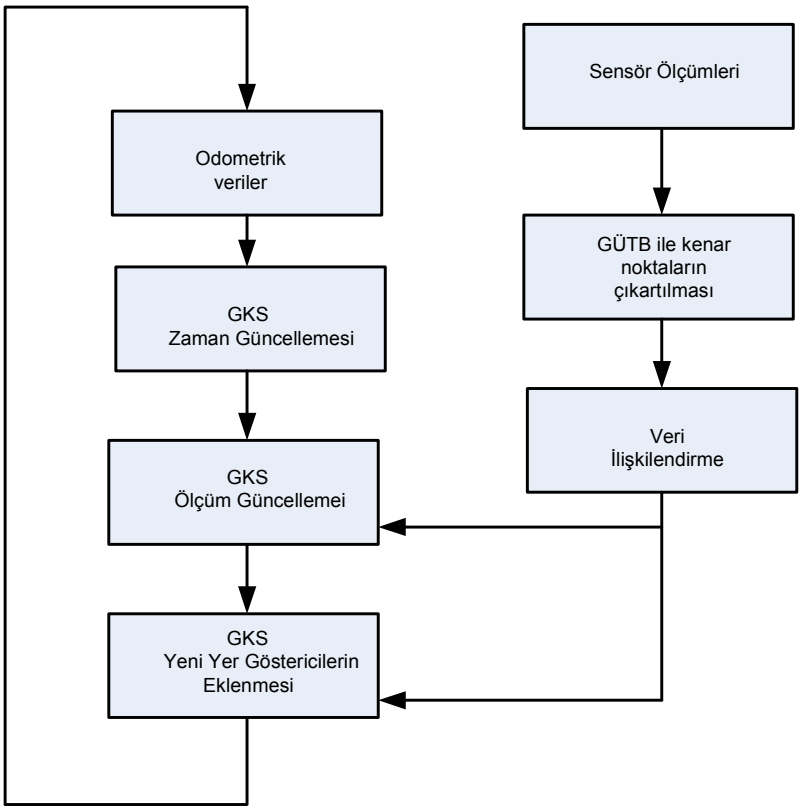
$$\text{Konum Hata Oranı} = \frac{\sqrt{(285.18 + 327.72)^2 + (-431.65 - 710.08)^2}}{69276}.100\% = 1.87\%$$

$$\text{Yönelim Hata Oranı} = \left|\frac{-164.97 + 15.7}{-164.97}\right|.100\% = 90.48\%$$

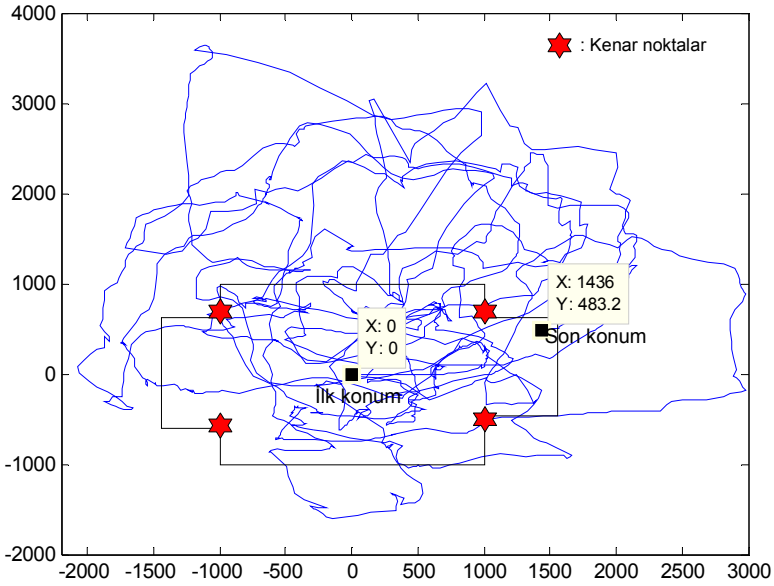
$$\text{Ortalama Hata Oranı} = \frac{1.87 + 90.48}{2} = 46.18\%$$

Sonuç olarak robot hareketinin izlenmesi sadece odometrik verilere bağlı olarak yapılması hatalı sonuçlar doğuracaktır. Benzetimde robotun sistem modelinde hata olmadığı varsayılmıştır. Burada odometrik hatanın nedeni, robot hareketi süresince alınan örneklerin kesikli olmasıdır. Diğer bir ifade ile alınan örnekler arasındaki sürede robot hareketinin bilinmemesidir. Bu yüzden odometrik hatanın oluşmasındaki en önemli neden zaman sorunudur.

Benzetim ortamında yapılan üçüncü çalışma ise robot konumunun ve ortam haritasının eş zamanlı olarak bölüm 5.2 ve 5.3 de anlatılan hareket ve ölçüm modelleri kullanılarak GKS ile kestirimidir. Bu çalışmada GKS algoritması içine bölüm 6.1'de test edilen GÜTB algoritması dahil edilmiştir. Bu sayede ortamdaki dikey kenar konumları ve robot konumu kestirimi eş zamanlı olarak yapılmıştır. Şekil 6.19'da EKBH sürecine ait blok şema görülmektedir. Buna göre robotun hareket etmesiyle (odometrik veriler ile) yeni konuma ait belirsizlik, GKS ile öngörülür. Robot hareketiyle birlikte ortam üzerinden alınan ölçümler kullanılarak yer göstericiler tespit edilir. Bu ölçümler ile daha önceden çıkarılmış olan yer göstericiler arasında eşleştirme yapılarak, robot ve yer gösterici konumlarının eş zamanlı olarak kestirimi yapılır. Tespit edilen yer göstericilerden daha önce gözlemlenmemiş olanlar (yeni yer göstericiler) GKS'e eklenir.



Şekil 6.19 EKBH sürecine ait blok şema



Şekil 6.20 Dört yer gösterici ile konum kestirimi

GKS ile robot konum kestirimi yapmak için öncelikle bölüm 6.1'de GÜTB algoritması ile tespit edilen ortamdaki 4 kenar nokta başlangıç anında (t_0) durum vektörü içinde (6.13) eşitliğindeki gibi tanımlanmıştır.

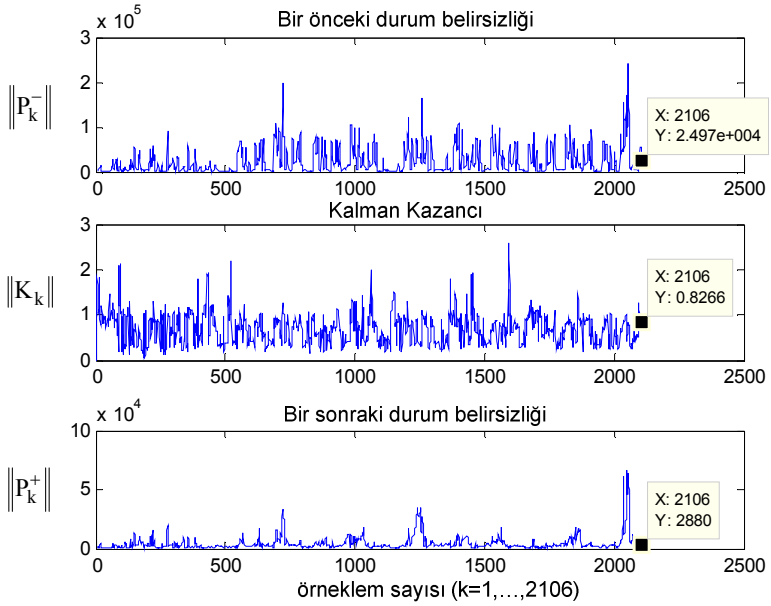
$$\hat{x}_0^+ = \begin{bmatrix} x_k^r \\ m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix}_{11x1} \qquad P_0^+ = \begin{bmatrix} P_{rr} & \cdots & & & 0 \\ & P_{11} & & & \\ & & P_{22} & & \\ & & & P_{33} & \\ 0 & \cdots & & & P_{44} \end{bmatrix}_{11x11} \tag{6.13}$$

$$x_k^r = (0,0,0)^T \qquad P_{rr} = 0 \tag{6.14}$$

Robot başlangıç konumu ve hata kovaryans matrisi (6.14)'deki gibi seçilmiştir. Durum hata kovaryans matrisinin köşegenleri dışında kalan terimlerin sıfır olmasının nedeni de robot konum belirsizliğinin başlangıçta sıfır kabul edilmesidir [14]. Bu yüzden sadece her bir yer göstericinin kendisi ile olan kovaryans matrisleri yer almaktadır. Bu matrisler (5.42) eşitliği kullanılarak hesaplanmıştır.

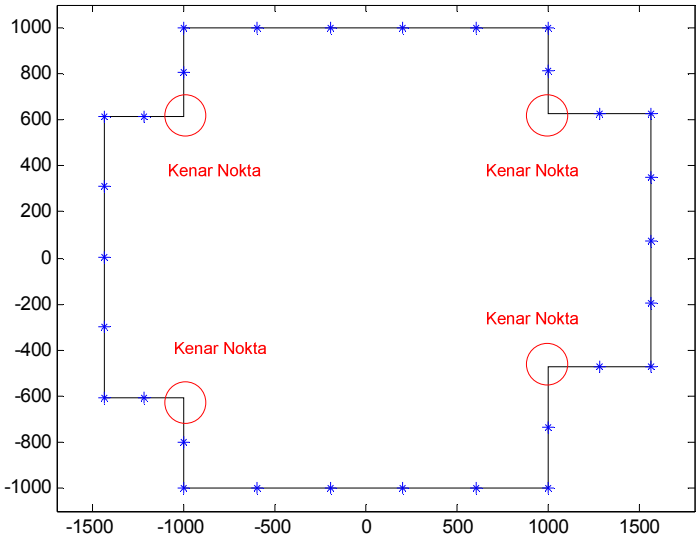
Ortamdaki sadece kenar noktalar ile GKS kullanılarak robot ve yer gösterici konumlarının kestirimini yapmak üzere çalışmalar yapılmıştır. GKS ile doğru konum kestirimi yapmak doğru veri ilişkilendirmesi yapmakla mümkündür [1]. Veri ilişkilendirmesi (5.29) eşitliğindeki denklemler ile hesaplanan Mahalanobis uzaklığına ve en yakın komşuluk ilişkisine göre yapılmaktadır. (5.29) eşitliğindeki kovaryans matriste ($S_k = H_k P_k^- H_k^T + R_k$) eğer bir sonraki öngörülen belirsizlik (P_k^-) çok fazla ise belirsizliğin etkisi yanlış eşleştirmelere neden olabilir [1]. Bu öngörülen belirsizlik ölçüm güncellemesi adımında süzgecin her doğru veri ilişkilendirmesi sonrasında azalmaktadır. Bu sayede belirsizlik azaldığı sürece süzgeç doğru şekilde çalışarak doğru konum kestirimi gerçekleştirecektir.

Şekil 6.20'de robot algılayıcıların durum vektörü içinde tanımlanan dört yer gösterici üzerinden her döngüde ölçüm alamadığı ve GKS ölçüm güncellemesi adımının her döngüde düzeltim yapamaması nedeniyle filtre bir süre sonra işlevselliğini kaybetmiştir ve robot konum kestirimi hatalı yapılmıştır. Şekil 6.21'de sistemin her bir döngüsünde hesaplanan bir önceki ve sonraki belirsizlik ile Kalman kazancı normları yer almaktadır. Bir sonraki durum belirsizliği bir önceki durum belirsizliği ile karşılaştırıldığında belirsizliğin yeteri kadar düşmediği görülmektedir. Bu nedenle doğru eşleştirmeler yapılamadığı için doğru konum kestirimi de gerçekleştirilememiştir.



Şekil 6.21 Sistem belirsizlikleri ve Kalman kazancı normları

Bu tez çalışmasında başlangıç anında ortam haritası üzerinde belirli sayıda yer göstericinin robot tarafından çıkarıldığı kabul edilmiştir. Şekil 6.22'de ortamda başlangıç anından önce (t_0^-) çıkarıldığı varsayılan yer göstericiler görülmektedir. Bu varsayımın yapılmasının sebebi mevcut sonar sayısı ile sadece ortamda bulunan kenar konumlar çıkartılarak kestirimin yapılamayacak olmasıdır. Çünkü Kalman süzgeci algoritmasında durum öngörüsünün kestirimi (5.21) eşitliğinde tanımlanan ölçüm modelini ile yapılır. Bu modeldeki ölçüm hatası robotun öngörülen konuma geldiğinde elde etmesi gereken ölçüm ile elde ettiği ölçüm arasındaki farkı gösterir. Dolayısıyla öngörü kestirimi bu hata farkına göre hesaplanan Kalman kazancının durum vektörü üzerine eklenmesiyle gerçekleştirilir (5.41). Bu sürecin işleyebilmesi için sonar ölçümlerin ortam üzerindeki yer göstericilerden en az biriyle ilişkilendirilmesi diğer bir ifade ile eşleştirilmesi gerekmektedir. [14]'e göre GKS ile yapılan EKBH uygulamasında belirli sayıda yer gösterici başlangıç koşulu olarak durum vektörü içinde tanımlanmıştır. Bu tez çalışmasında da sanal ortamdaki her bir sınır çizgisi, uzunluğa bağlı olarak, eşit aralıklara bölünerek sonar ölçümlerin her döngüde en az bir yer gösterici ile eşleştirilmesi sağlanmıştır. Şekil 6.22'de gösterildiği gibi başlangıç anında 30 yer gösterici konumu durum vektörü içersinde tanımlanmıştır. Ortam üzerindeki kenar noktalar gezinim sırasında tespit edildikçe durum vektörüne eklenmiştir.



Şekil 6.22 Başlangıç anındaki yer gösterici konumları

Bu duruma göre başlangıç anında sistem durum vektörü ve durum hata kovaryans matrisi aşağıdaki gibi tanımlanmıştır.

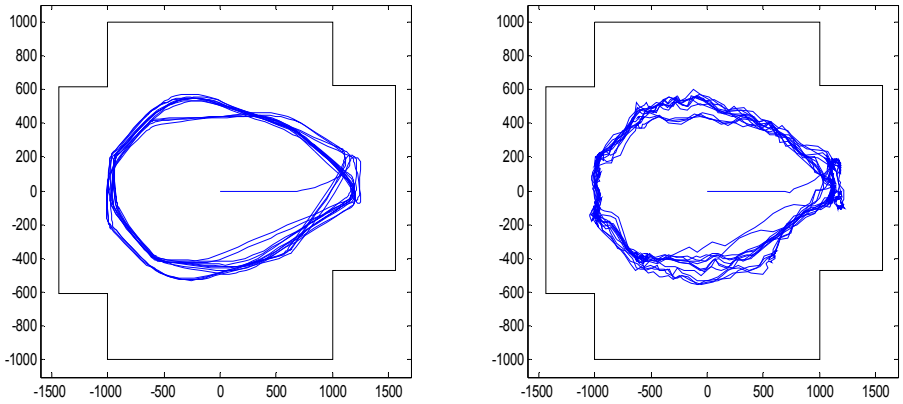
$$\hat{x}_k = \begin{bmatrix} x_k^r \\ m_1 \\ \vdots \\ m_{30} \end{bmatrix}_{63x1} \qquad P_0^+ = \begin{bmatrix} P_{rr} & \cdots & 0 \\ & P_{11} & \\ & & \ddots \\ 0 & \cdots & P_{3030} \end{bmatrix}_{63x63} \tag{6.15}$$

Robotun başlangıç konumu ve konum hata kovaryans matrisi (6.16)'da gösterildiği gibi seçilmiştir.

$$x_k^r = (0,0,0)^T \qquad P_{rr} = 0 \tag{6.16}$$

Robot konum hata kovaryans matrisi P_{rr}, başlangıç anında sıfırdır. Bu durum robotun başlangıç konumu hakkında mutlak bilgi sahibi olduğunu gösterir. Durum hata kovaryans matrisinin köşegenleri dışında kalan terimlerinin sıfır olmasının nedeni robot konum belirsizliğinin başlangıç anında sıfır seçilmesinden kaynaklanmıştır. Bu yüzden sadece her bir yer göstericinin kendisi ile olan kovaryans matrisleri yer almaktadır (6.15). Bu matrisler (5.42) eşitliği kullanılarak hesaplanmıştır. (5.42)'deki R gürültü kovaryans matrisi $\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$ seçilmiştir.

GKS ile odometrik veriler kullanıldığında robotun gerçek hareketine yaklaştığı görülmüştür. Şekil 6.23'de robotun izlediği yolun, gerçekte olması gereken yolu takip etmeye çalıştığı görülmektedir. Çizelge 6.11'de robotun son konumları karşılaştırılmıştır.



Şekil 6.23 a Robotun gerçek hareketi

b Robotun GKS ile kestirilmiş hareketi

Çizelge 6.11 Gerçek ve GKS ile son konumlar

	X(mm)	Y(mm)	θ (derece)
Gerçek son konum	285.18	-431.65	-164.97
GKS ile son konum	231.88	-420.35	-162.40

GKS ile son durumdaki robot konum ve yönelim hata oranları (6.10), (6.11) ve (6.12) eşitlikleri kullanılarak tekrar hesaplanmıştır.

$$\text{Konum Hata Oranı} = \frac{\sqrt{(285.18-231.88)^2+(-431.65+420.35)^2}}{69276}.100\% = 0.08\%$$

$$\text{Yönelim Hata Oranı} = \left|\frac{-164.97+162.40}{-164.97}\right|.100\% = 1.55\%$$

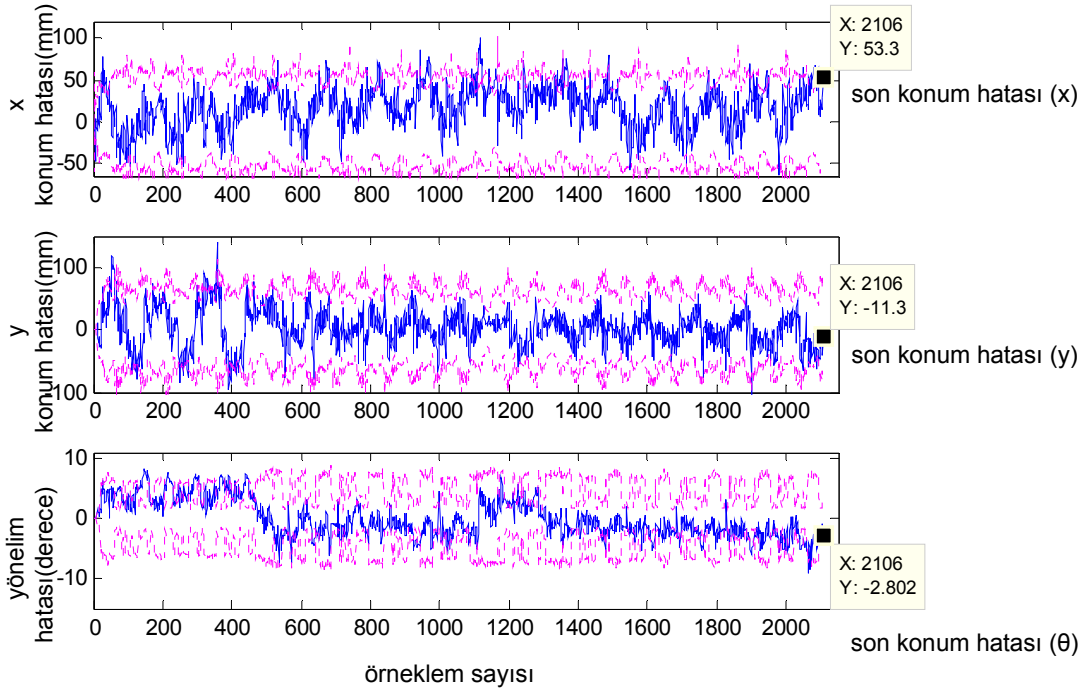
$$\text{Ortalama Hata Oranı} = \frac{0.08+1.55}{2} = 0.82\%$$

Kalman süzgeci ile sadece odometrik veriler kullanılarak elde edilen son konum verilerindeki hata oranları çizelge 6.12'de karşılaştırılmıştır.

Çizelge 6.12 Hata oranlarının karşılaştırılması

	Odometrik Hata	GKS Hata
Konum Hatası %	1,87	0,08
Yönelim Hatası %	90,48	1,55
Ortalama Hata %	46,18	0,82

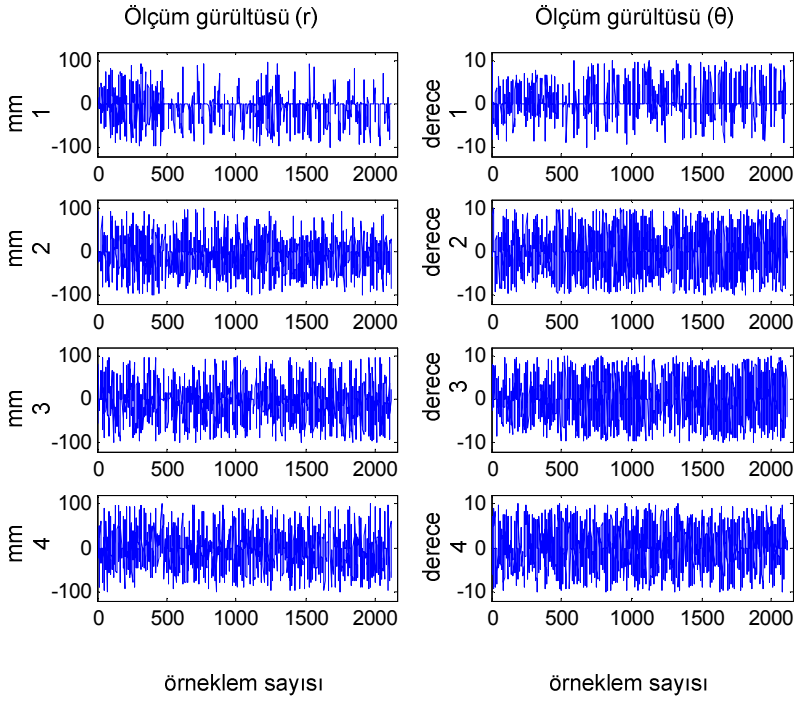
Çizelge 6.12'de görüldüğü gibi odometrik veriler GKS ile kullanıldığında son durumdaki konumsal hata oranlarında çok belirgin bir iyileşme olmuştur. Robotun gezinim sırasındaki konumsal ve yönelim hata farkları şekil 6.24'de gösterilmiştir. Konum ve yönelim hata farklarının 2σ belirsizlik sınırı içinde kaldığı görülmüştür. Bu sonuç GKS tabanlı yapılan EKBH uygulamasının tutarlılığını göstermektedir [11,13,14].



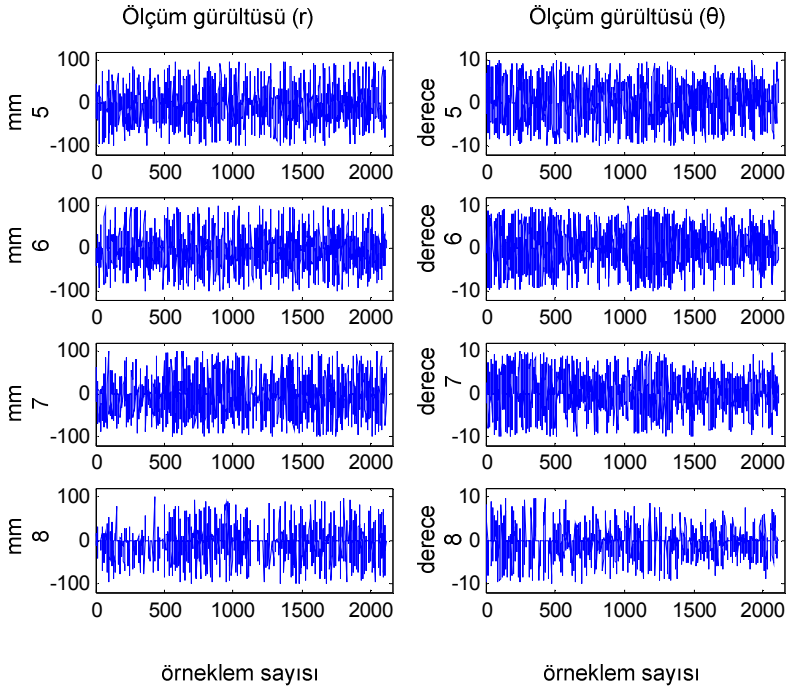
Şekil 6.24 Konum ve yönelim hata farkı.

(6.15)'deki başlangıç durum belirsizliği P_0^+ için 63x63'lük matris tanımlanmıştır. Başlangıç anındaki belirsizlik normu $\left\|P_0^+\right\| = 2832$ olarak hesaplanmıştır. Sisteme verilen denetim girdileri ile durum öngörüsü yapılmış ve durum öngörüsü belirsizliği her döngüde durum hata kovaryans matrisinin norm'u alınarak hesaplanmıştır.

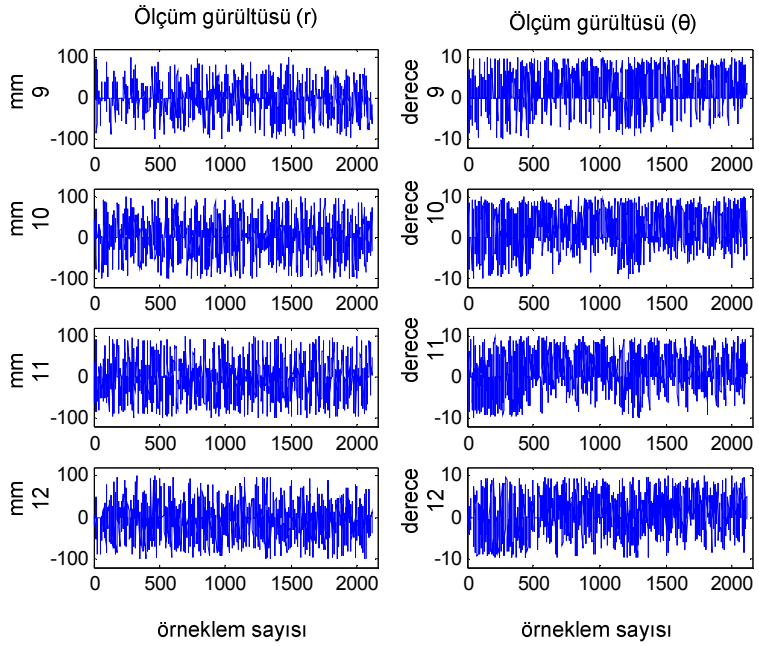
Kalman süzgeci algoritmasının ölçüm güncellemesi bölümünde öngörülen konumda elde edilmesi gereken ölçüm değerleri ile robotun elde ettiği değerler arasındaki fark (yenilenme terime) ve yenilenme kovaryans matrisi her döngüde yeniden hesaplanmıştır. (5.30) eşitliği kullanılarak her bir ölçüm ile yer gösterici arasındaki Mahalanobis uzaklığı ayrı ayrı hesaplanmış, BUEK algoritması kullanılarak en küçük Mahalanobis değerinin, özgürlük derecesi 2 olan Chi-kare dağılımın %95'lik güvenilirlik düzeyi içinde kalıp kalmadığı kontrol edilmiştir. Bu sayede her bir ölçümün hangi yer gösterici üzerinden elde edildiği belirlenmiş ve (5.31), (5.32), (5.33)'deki matrisler bu doğrultuda oluşturulmuştur. Uygun eşleştirmeler sonucunda 16 sonar için ölçüm gürültüleri şekil 6.25, 6.26, 6.27 ve 6.28'de gösterilmiştir.



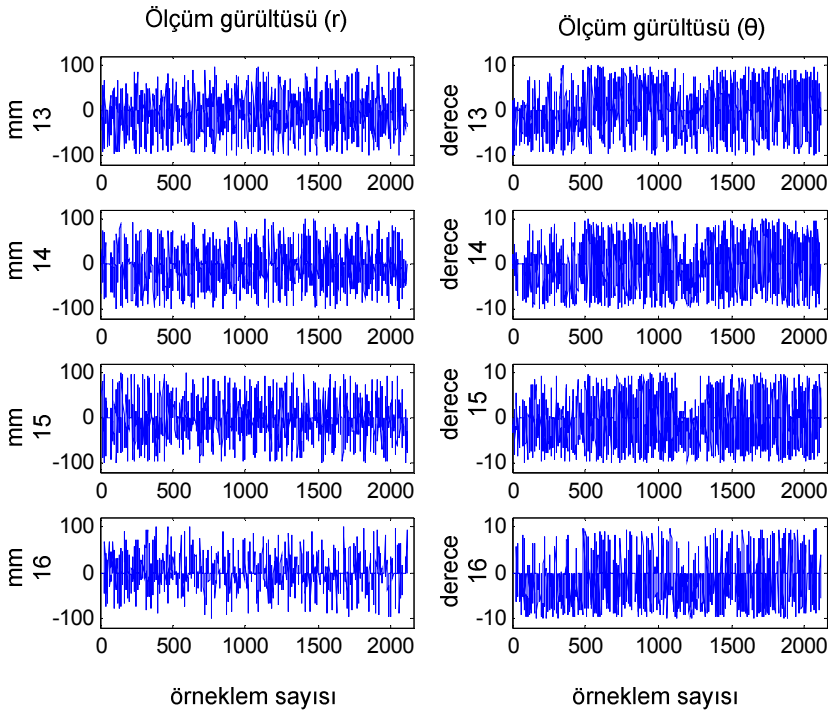
Şekil 6.25 1-4 numaralı sonarlara ait gürültü değerleri



Şekil 6.26 5-8 numaralı sonarlara ait gürültü değerleri

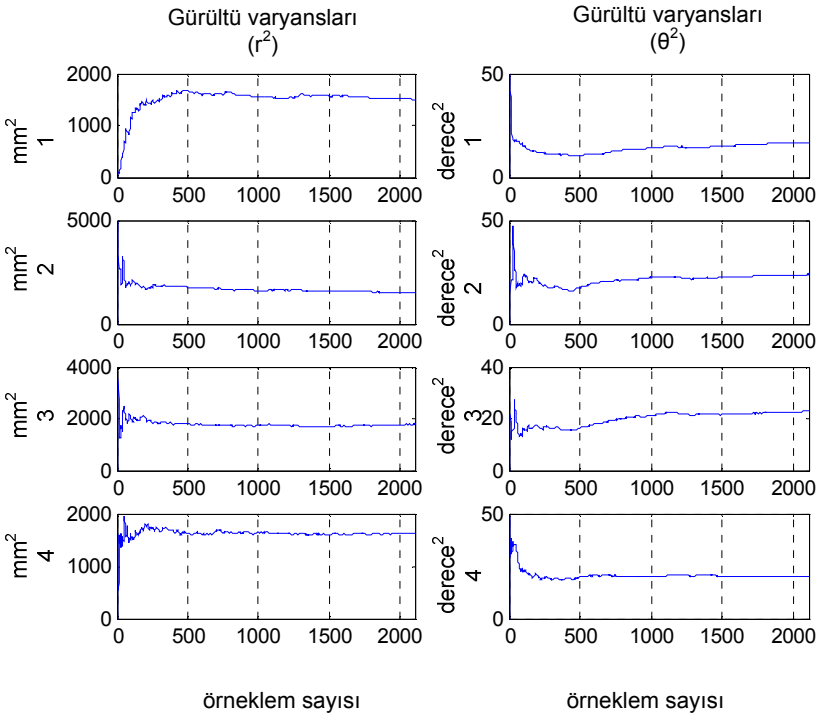


Şekil 6.27 9-12 numaralı sonarlara ait gürültü değerleri

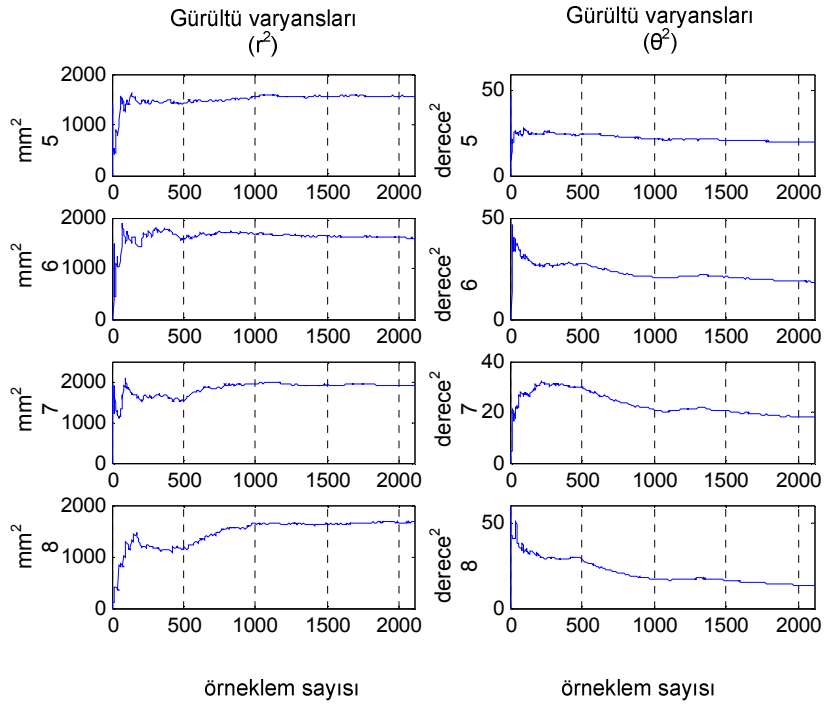


Şekil 6.28 13-16 numaralı sonarlara ait gürültü değerleri

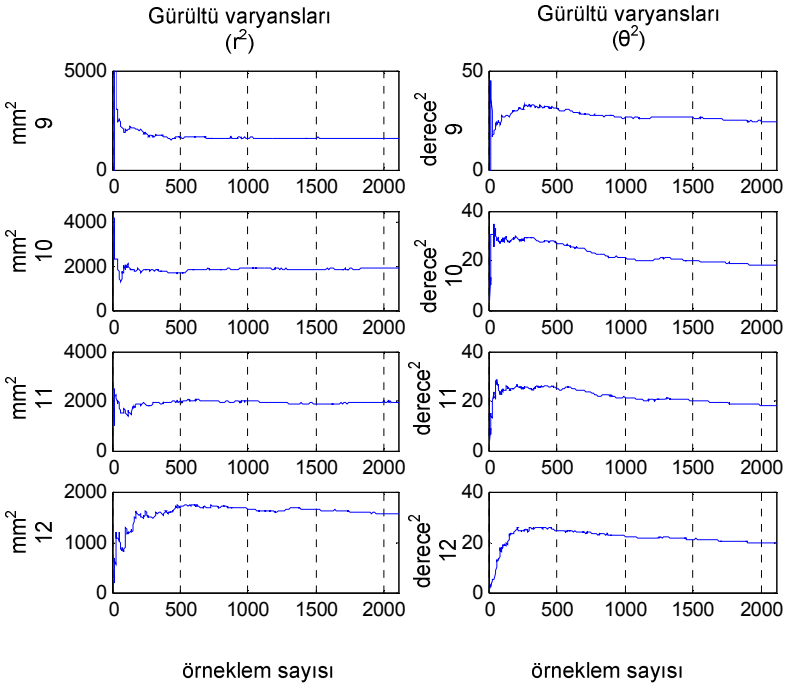
Şekil 6.25, 6.26, 6.27, 6.28'deki her bir algılayıcıya ait mesafe ve açısal gürültüler, (5.29) eşitliğinde tanımlanan v_k ölçüm farkına diğer bir ifadeyle ölçüm ile eşleştirmenin yapıldığı yer gösterici (ölçüm modelinden gelen ölçüm) arasındaki gürültü değerine göre her döngüde yeniden hesaplanarak bulunmuştur. Bu nedenle her bir ayrık zamandaki algılayıcı gürültüleri aslında eşleşmenin yapıldığı ölçüm ile yer gösterici arasındaki v_k ölçüm farkının [-100mm,100mm] ve [-10^0,10^0] aralığında kaldığını göstermektedir. Kısaca verilerin ilişkilendirilmesi yukarıdaki gürültü aralığında yapılmıştır. Algılayıcı gürültüleri Kalman kazancını değiştirecektir. Gürültülerin varyansları Kalman kazancına (5.40) eşitliğinde tanımlandığı gibi eklenir. Gürültü varyans değerlerinin artması yani gürültüde oluşacak değişimlerin artması, bir sonraki durum öngörüsü belirsizliğini artıracaktır Elde edilen gürültü varyansları 16 sonar algılayıcı için şekil 6.29, 6.30, 6.31 ve 6.32'de gösterilmiştir.



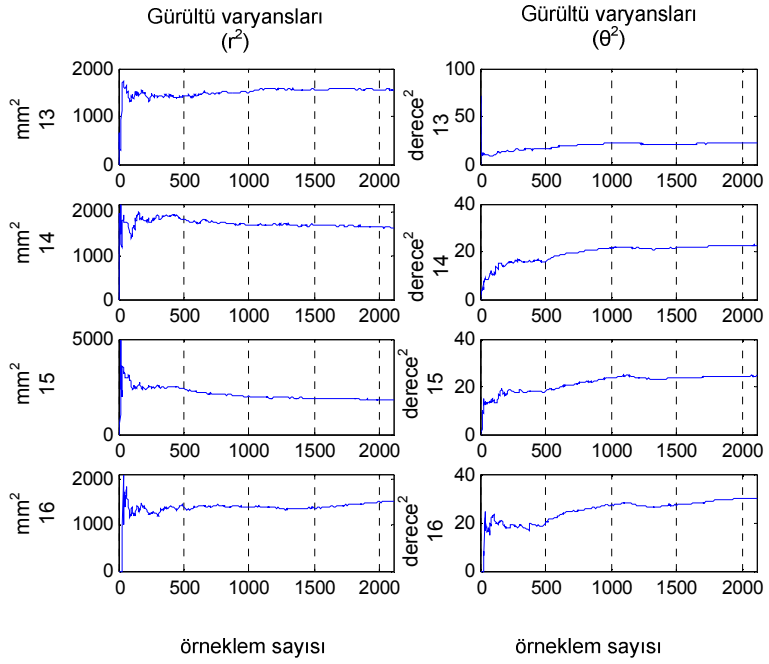
Şekil 6.29 1-4 numaralı sonarların gürültü kovaryansları R_k



Şekil 6.30 5-8 numaralı sonarların gürültü kovaryansları R_k



Şekil 6.31 9-12 numaralı sonarların gürültü kovaryansları R_k



Şekil 6.32 13-16 numaralı sonarların gürültü kovaryansları R_k

Elde edilen son Kalman kazancı $K_{2106} = \begin{bmatrix} K_{1,1} & \cdots & K_{1,22} \\ \vdots & \ddots & \vdots \\ K_{71,1} & \cdots & K_{71,22} \end{bmatrix}_{71x22}$ boyutundadır.

Bunun nedeni son döngüde 16 sonar ölçümün, 11 yer gösterici ile eşleştirilmiş olmasıdır. 2106 örnek sayısıdır.

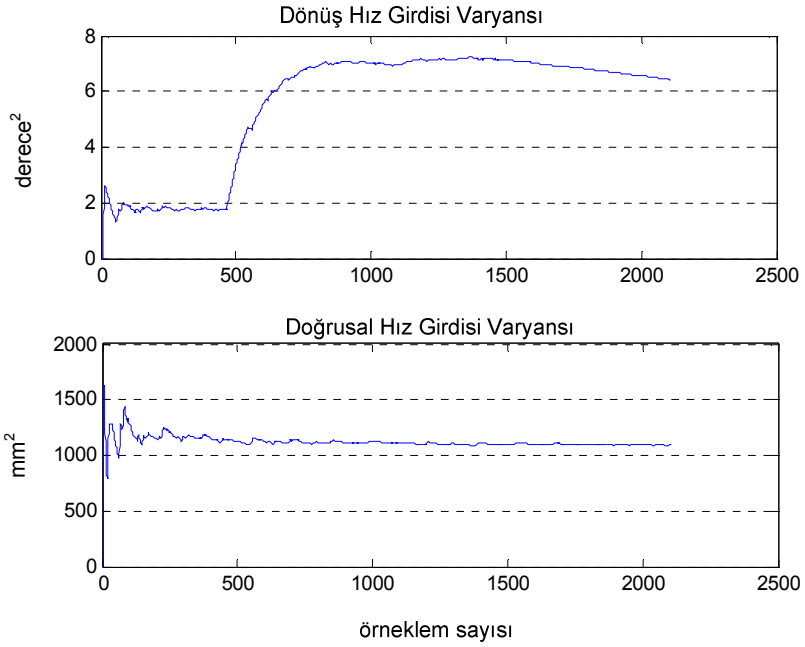
Çizelge 6.13 Ölçüm-yer gösterici eşleştirme matrisi H_{2106}

	Sonar Ölçümler (1-8)							
	#1	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**
YG numarası	0	2	0	30	27	25	22	0
MD	10,999	5,8293	55,2896	0,29339	0,45338	0,16219	0,72342	20,19915

	Sonar Ölçümler (9-16)							
	#9	**#10**	**#11**	**#12**	**#13**	**#14**	**#15**	**#16**
YG numarası	18	15	13	11	8	0	5	0
MD	0,017795	1,669	0,91096	0,43779	1,1582	9,2914	2,8822	19,1137

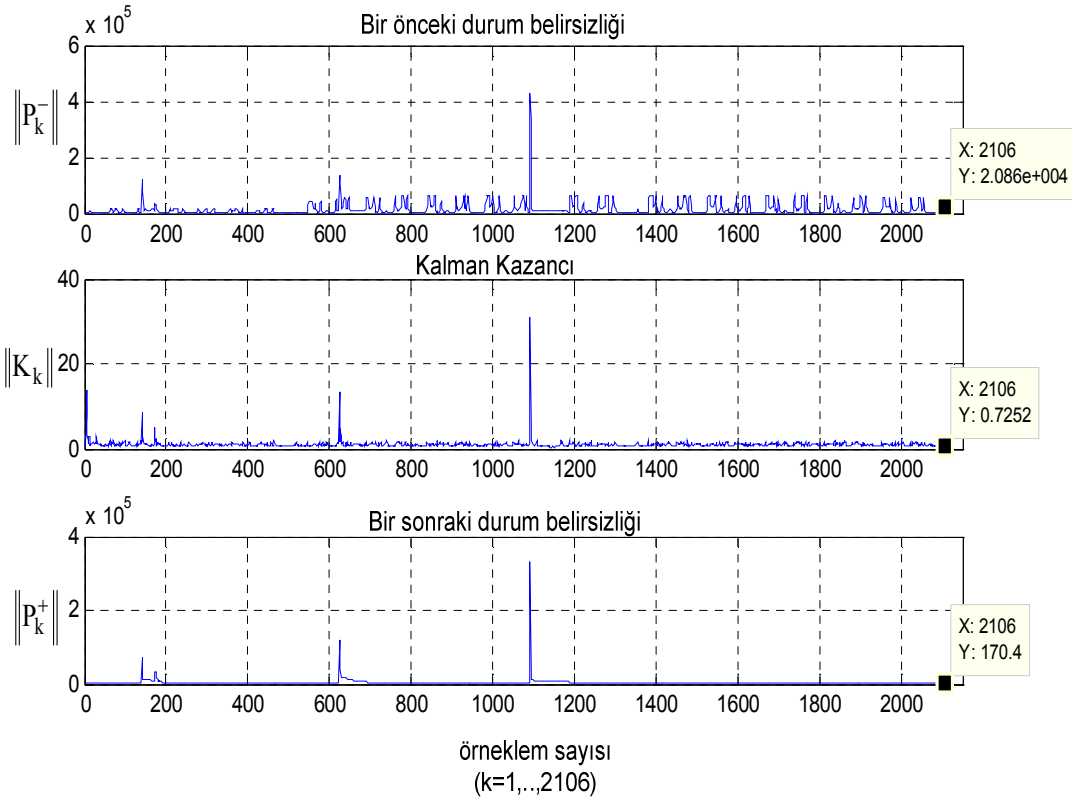
Çizelge 6.13'e bakıldığında son konumda sonar ölçümlerden sadece 1, 3, 8, 14 ve 16 numaralılar her hangi bir yer gösterici ile eşleşmediği görülmektedir. Bu durum sonar ölçümlere ait hesaplanan en küçük Mahalanobis uzaklık değerlerinin 5.99'dan daha büyük olmasından kaynaklanmaktadır. 2, 4, 5, 6, 7, 9, 10, 11, 12, 13, 15 numaralı diğer sonar ölçümler ise sırasıyla 2, 30, 27, 25, 22, 18, 15, 13, 11, 8, 5 numaralı yer göstericiler ile eşleşmiştir. Yer göstericiler durum vektörü içinde tanımlandıkları sıraya göre numaralandırılmıştır.

GKS uygulamalarında gürültü varyansları sabit değerler olarak alınır. Ancak bu uygulamada zaman sorunu olmadığı için hem denetim girdisi hem de ölçüm gürültü varyansları her döngüde yeniden hesaplanmıştır. Şekil 6.33'de hesaplanan denetim girdileri varyansları görülmektedir.



Şekil 6.33 Denetim girdileri varyansları

Kalman algoritması kullanılarak her döngüde bir önceki durum belirsizliği P_k^-, Kalman kazancı K_k ve bir sonraki durum belirsizliği P_k^+ her döngüde yeniden hesaplanmıştır. Sistemin bir sonraki durum belirsizliğinin norm'u $\|P_k^+\|$, bir önceki durum belirsizliği norm'u $\|P_k^-\|$ ile karşılaştırıldığında her örneklemede aldığı değerln daha küçük olduğu görülmektedir (şekil 6.34). Bunun anlamı öngörülen robot ve yer gösterici konumlarındaki düzeltme işlemi başarı ile gerçekleşmiştir. Ancak yeni yer göstericilerin (kenar noktalar) sisteme eklendiği 139, 171,623,1091 döngü adımlarında durum belirsizliğinin arttığı görülmektedir. Sistemin bir önceki belirsizliğindeki bu artış bir sonraki döngüde Kalman kazancı normunun $\|K_k\|$ yükselmesine neden olmuştur. Böylece bir sonraki durumun belirsizliği daha yüksek bir kazanç değeri ile düzeltilmiştir. Yeni yer göstericilerin bir önceki durum belirsizliğini artırmasının nedeni sonar gürültü varyanslarının yüksek olmasıdır. Buda tespit edilen yer gösterici konumuna ait belirsizliğin başlangıçta yüksek olduğunu göstermektedir.

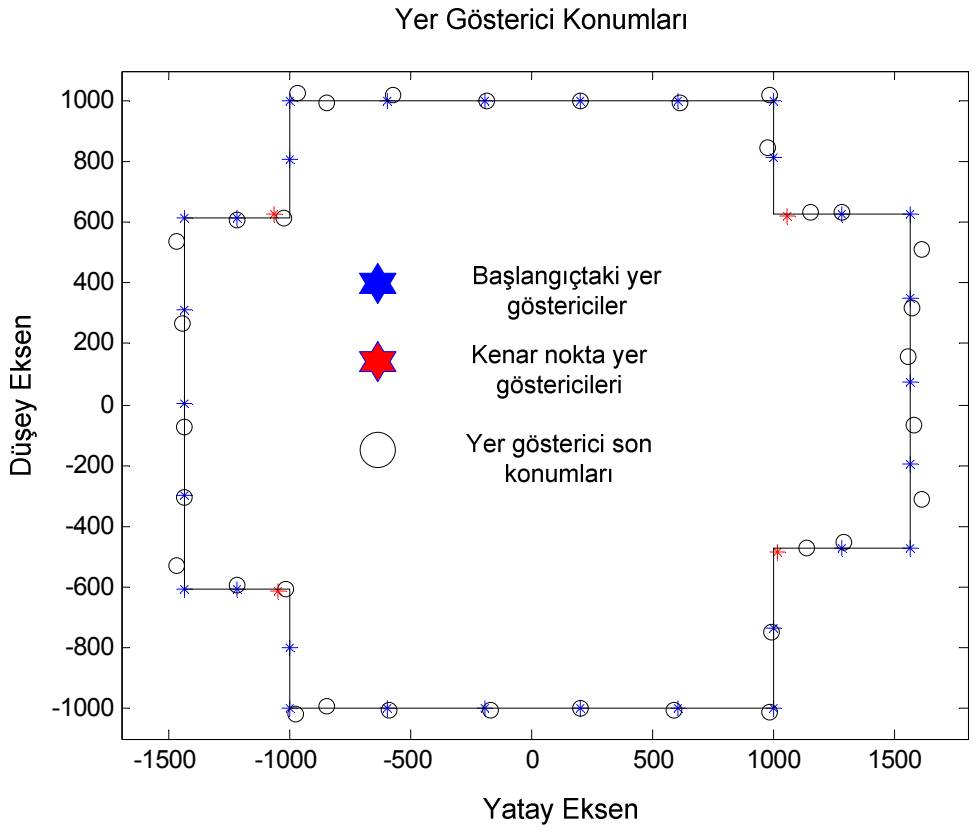


Şekil 6.34 P_k^-, P_k^+ ve K_k norm değerleri

Yeni yer göstericiler durum vektörüne eklendikten sonra, durum hata kovaryans matrisinin güncellenmesi sırayla (5.42), (5.43) ve (5.44) eşitlikleri kullanılarak yapılmıştır. (5.42) eşitliğindeki gürültü varyansı yer göstericinin tespit edildiği algılayıcıya ait olan varyanstır. Dört yer gösterici içinde aynı süreç tekrarlanmıştır. Gezinim tamamlandığında 71x71 boyutunda durum hata kovarys matrisi elde edilmiştir.

Şekil 6.35'de gezinim tamamlandığında ortamdaki tüm yer göstericilerin son konumları çember sembolü ile gösterilmiştir. Yer gösterici konumlarındaki değişimi gösterebilmek için harita üzerindeki başlangıç konumlar yıldız sembolü ile işaretlenmiştir. Mavi renkli yıldızlar başlangıç anından önce tespit edildiği kabul edilen yer göstericileri, kırmızı renkli yıldızlar ise gezinim sırasında tespit edilen ve yeni durum değişkeni olarak GKS'e eklenen kenar nokta konumlarını temsil etmektedir. Bu durumda Kalman algoritması doğrultusunda sonar ölçüm ve yer

gösterici eşleştirmelerine göre durum öngörüsü düzeltildiğinde, yer gösterici konumlarının da kestirimi yapılmış olur. Son durumdaki yer gösterici konumları çizelge 6.14'de verilmiştir.



Şekil 6.35 Yer gösterici konumları

Çizelge 6.14 Yer gösterici konumları

	Başlangıç Konumları (mm)		GKS ile son Konumlar (mm)		**Fark (mm)**	
Yer Gösterici Numarası	x	y	x	y	**Δx**	**Δy**
1	-600	-1000	-593,42	-1003,9	**-6,58**	**3,9**
2	-200	-1000	-168,23	-1006,4	**-31,77**	**6,4**
3	200	-1000	195,34	-1000,9	**4,66**	**0,9**
4	600	-1000	588,11	-1007,1	**11,89**	**7,1**
5	1000	-1000	984,32	-1015,7	**15,68**	**15,7**
6	1000	-735	991,52	-746,54	**8,48**	**11,54**
7	1281	-470	1286	-456,44	**-5**	**-13,56**
8	1562	-470	1610,2	-309,9	**-48,2**	**-160,1**
9	1562	-199	1578,9	-66,115	**-16,9**	**-132,885**
10	1562	76	1558	157,9	**4**	**-81,9**
11	1562	351	1569,7	316,72	**-7,7**	**34,28**
12	1562	626	1607,1	513,43	**-45,1**	**112,57**
13	1281	626	1278,2	631,66	**2,8**	**-5,66**
14	1000	813	971,17	848,45	**28,83**	**-35,45**
15	1000	1000	982,67	1017,3	**17,33**	**-17,3**
16	600	1000	611,93	992,84	**-11,93**	**7,16**
17	200	1000	195,73	1000,9	**4,27**	**-0,9**
18	-200	1000	-190,16	1002	**-9,84**	**-2**
19	-600	1000	-571,96	1016,8	**-28,04**	**-16,8**
20	-1000	1000	-971,39	1028,6	**-28,61**	**-28,6**
21	-1000	808	-848,95	994,95	**-151,05**	**-186,95**
22	-1219	615	-1223	607,17	**4**	**7,83**
23	-1437	615	-1469,9	538,13	**32,9**	**76,87**
24	-1437	310	-1446,8	264,59	**9,8**	**45,41**
25	-1437	5	-1437,3	-73,188	**0,3**	**78,188**
26	-1437	-300	-1435,2	-308,42	**-1,8**	**8,42**
27	-1437	-605	-1469,4	-527,9	**32,4**	**-77,1**
28	-1219	-605	-1222,8	-597,35	**3,8**	**-7,65**
29	-1000	-803	-846,13	-994,61	**-153,87**	**191,61**
30	-1000	-1000	-979,54	-1020,4	**-20,46**	**20,4**
31	-1049	-614,94	-1015,7	-607,1	**-33,3**	**-7,84**
32	1011,15	-485,88	1136,8	-469,95	**-125,65**	**-15,93**
33	-1063,3	624,64	-1024,5	614,25	**-38,8**	**10,39**
34	1056,6	621,85	1149,8	630,78	**-93,2**	**-8,93**

7. SONUÇ

Eş zamanlı konum belirleme ve haritalama uygulamasına yönelik Pioneer robotlar için hazırlanmış MobilSim benzetimcisi ve Matlab programı kullanılarak benzetim tabanlı bir çalışma gerçekleştirilmiştir.

Çalışma iki aşamada değerlendirilmiştir. İlk aşamasında Üçgenleme Tabanlı Birleşim algoritması kullanılarak benzetim ortamında tanımlanan çevrenin öznitelik tabanlı haritası çıkartılmıştır. Harita ile ilgili çıkartılan her bir konum verisi, robot konumu belirlenirken birer yer gösterici olarak kullanılmıştır. Standart ÜTB algoritması kullanılarak çizelge 6.6 gösterilen eşik değerleri ile sanal ortamdaki kenar noktalar tespit edilmiştir. Ancak düzlem üzerinde gerçekte olmaması gereken sahte kenar noktaların da oluşturulan haritada yer aldığı görülmüştür. Sahte kenar noktalardan kurtulmak için standart ÜTB algoritmasında iyileştirmeler yapılmıştır. Bu doğrultuda "kararlı kesişim koşulu" ve "etkin hareketli pencere güncellemesi" süreçleri algoritmaya dahil edilmiştir. Bu sayede Geliştirilmiş Üçgenleme Tabanlı Birleşim algoritması kullanılarak düzlem üzerinde tespit edilen sahte kenar noktalar harita üzerinden çıkartılmıştır. Harita üzerinde yer alan 4 kenar noktaya ait konumlar çizelge 6.8'de belirtildiği gibi %1.05, %1.65, %3.49, %6.4 ortalama konum hata oranlarıyla tespit edilmiştir. Bu tez çalışmasında sonar ölçümlerin hafızada tutulduğu pencere boyutu 16x20 olarak seçilmiştir. Yapılan deneysel çalışmalarda 16x20'lik pencere boyutunun kenar nokta belirlemede ortamın genişliğine ve ortamdaki kenar nokta sayısına bağlı olarak tüm kenar noktaların tespit edilmesinde yetersiz kalabildiği görülmüştür. Bu durumda ya donanımsal olarak algılayıcı sayısı artırılarak yada daha fazla sonar ölçüm hafızada tutularak çözüm aranmalıdır. Kısacası pencere boyutu genişletilerek daha fazla örnek arasından kenar nokta tespiti yapılmalıdır. Pencere boyutunun çok artması daha fazla kenar nokta tespit etme olasılığını artırırken, algoritmanın çalışma süresini de artıracaktır. Buda eş zamanlı konum belirleme ve haritalama uygulamalarının gerçek zamanlı olarak yapılması konusunda kısıtlamalar getirebilir. Literatürdeki ÜTB algoritması ile yapılan uygulamalar kullanılan robot yapılarına ve donanımsal özelliklerine bağlı olarak 24x10 boyutundaki pencereler ile gerçekleştirilmiştir. Bu tez çalışmasında pencere sütun sayısı 20 seçildiğinde tanımlanan boş sanal ortamdaki kenar noktaların hepsi tespit edilmiştir. Kenar

noktaların belirlenmesinde GÜTB algoritmasında yer alan eşik seviyeleri çizelge 6.6 belirtildiği gibi seçilmiştir. Ortam daha büyük ve daha çok kenar noktadan oluşan daha karmaşık bir yapıda tanımlandığında ise çizelge 6.9'da tanımlanan eşik değerleri ve 16x20'lik pencere ile 12 kenar noktadan yalnızca 1'ini tespit edilebildiği görülmüştür. Pencere boyutu 16x40 olarak genişletildiğinde aynı eşik değerleriyle 12 kenar noktadan 5'i tespit edilmiştir. Bu nedenle daha karmaşık ortamlarda yapılacak çalışmalarda daha fazla algılayıcı sayısına sahip ve algılayıcıların robot üzerinde simetrik şekilde dizildikleri gezgin robotlar ile çalışılması daha az kenar noktanın kaçırılmasına neden olabilir.

Çalışmanın ikinci aşamasında genişletilmiş Kalman süzgeci algoritması ve GÜTB algoritması bir arada kullanılarak eş zamanlı robot ve yer gösterici konum kestirimi yapılmıştır.

Her bir kesikli zaman dilimi içersinde ortam üzerinde sadece 16 noktadan örnek alınması başlangıç anında bazı yer göstericilerin daha önceden tespit edilmiş olduğu varsayımının yapılması zorunluluğunu doğurmuştur. Bu zorunluluk Kalman algoritmasında ölçüm güncelleme adımının her döngüde yapılabilmesi için en az bir yer gösterici üzerinden ölçüm alınmış olması gerekliliğinden kaynaklanmaktadır. Ayrıca literatürde bir çok uygulama çok hassas algılayıcılarla yapılmadığı için gezgin robotlar ortam içinde birkaç kez tur attırıldıktan sonra topladığı ön bilgiler eşliğinde çalışılmıştır [14]. Ortam üzerindeki tüm kenar noktalar GÜTB algoritması ile tespit edilebildiği için düzlem üzerinde belirli sayıda yer gösterici başlangıç anında durum vektörü ve durum hata kovaryans matrisi içinde tanımlanmıştır. Gezinim sırasında tespit edilen kenar noktalar ise GKS'e eklenerek durum vektörü ve durum hata kovaryans matrisleri genişletilmiştir. Robotun konumunun belirlenmesinde sadece odometrik veriler ile yapılmasının hatalı sonuçlar doğuracağı görülmüştür. Çizelge 6.12'de robot son konum hata oranları ile ilgili sonuçlar yer almaktadır. Bu sonuçlarda GKS kullanılarak yapılan konum belirleme işleminin çok daha başarılı olduğu belirlenmiştir. Yapılan konum belirleme işleminde son konumdaki ortalama hata odometrik veriler kullanıldığında %46.18 iken GKS kullanıldığında %0.82 değerine düşmüştür. GKS ile tüm gezinim sırasında kestirilen robot konumu ve gerçek konum arasındaki farkın (hata) 2σ

sınırı içersinde kaldığı görülmüştür. Bu sonuç EKBH tabanlı GKS süzgecinin konum belirlemedeki tutarlılığını göstermiştir.

Genişletilmiş Kalman süzgeci algoritmasında öngörülemeyen sistem gürültüsünün kovaryans değeri Q ve ölçüm gürültüsünün kovaryans değeri R bu tez çalışmasında sabit alınmamış, her döngüde yeniden hesaplanmıştır. Sistem gürültüsünü oluşturan denetim girdileri varyanslarından, dönme hızı varyansı ve doğrusal hız varyansı şekil 6.33'de gösterilmiştir. Buradaki varyans değişimlerine bakılarak robotun yaptığı hareketler anlaşılabilir. Örneğin robotun başlangıç anından itibaren yaklaşık ilk 50 örnekleme süresince dönüş hareketi yapmadığı ve bu süre sonunda varyans değerinin sıfırdan anlık olarak değiştiği görülmektedir. Aynı şekilde robotun başlangıç anında hareketinin doğrusal olarak değiştiği doğrusal hız varyansının başlangıçtaki ani değişimine bakılarak yorumlanabilir. Her iki varyans değişiminin testere dişi şeklinde olduğu anlar robotun dönüş hareketinin gerçekleştiği zamanları göstermektedir. Sonar ölçümlerdeki mesafe ve açı hatalarının varyansları 16 sonar algılayıcı için ayrı ayrı şekil 6.29-6.30.6.31-6.32'de gösterilmiştir. Bu varyans değerlerinin belirli bir örnekleme sayısı sonrasında sabit bir değere yakınsadığı görülmektedir. GKS algoritmasındaki R arttığı sürece Kalman kazancı azalarak, durumun daha yavaş gerçek duruma yakınsamasını sağlar. Diğer bir ifade ile ölçüm verilerine güvenilmez. R azaldıkça bu yakınsama hızı artacaktır. Kısaca istenilen R'nin azalmasıdır. R değerinin azalamaya başladığı sıralarda tekrar değerinin artarak belirli bir aralıkta kalması, denetim girdilerindeki varyans değerlerinin sürekli olarak değişmesinden kaynaklanmaktadır. Her iki varyansın (Q ve R) belirli değere yakınsamasının nedeni budur. Kalman süzgecinin çalışması robotun yaptığı hareketlere ve ölçüm değerlerine göre olur. Q kovaryansı artarsa bir önceki durum belirsizliği artar ve durum kestiriminde ölçüm verilerinin etkisi artmış olur. R kovaryansı artarsa bir sonraki durum belirsizliği artar ve durum kestirimde denetim girdilerinden alınan verilerin etkisi artmış olur. Her iki varyans değeri değişiminin belirli değerlere yakınsamasıyla Kalman kazancıda belirli aralıkta kalarak (bu tez çalışmasında kazancın [0.5,1] aralığında yakınsadığı tespit edilmiştir) durum kestirimini gerçekleştirir.

Bu tez çalışmasında sadece kenar konum tespiti yapılmıştır. Gelecek çalışmalarda Hough dönüşüm algoritması kullanılarak düzlem konumlarının da harita üzerinde ifade edilmesi düşünülmektedir.

Benzetim çalışmalarında Pioneer 3DX robotu model olarak alınmıştır. Bu modeldeki gerek sonar sayısı ve gerekse sonarların diziliş şekli nedeniyle ortam üzerindeki kenar konumlar tanımlanan 16x20'lik pencere boyutu ile kısa süreli aralıklarda tespit edilememiştir. Bu yüzden sonar algılayıcı yerine lazer algılayıcı kullanılması daha kısa aralıklarla daha çok yer göstericinin tanımlanmasını sağlayabilir. Lazer algılayıcı ile yapılan çalışmalarda tanımlanan ortam haritası sonar algılayıcı ile yapılanlara göre oldukça başarılıdır. Ancak lazer algılayıcıların yüksek maliyetli oluşu birçok uygulamada sonar algılayıcıların tercih edilmesine neden olmaktadır.

Sonuç olarak düşük maliyetli algılayıcılar kullanılarak yer gösterici konumlarının ve robot konumunun eş zamanlı olarak genişletilmiş Kalman süzgeci ile kestirimi başarıyla gerçekleştirilmiştir.

KAYNAKLAR

[1] Hogue Oral, “Simultaneous Localization and Mapping Techniques Oral Exam”: June 20,2005

[2] Hasan Hatipoğlu, “Genişletilmiş Kalman Süzgeci ile Gezgin Robot Konumunun Belirlenmesi” Ocak, 2007.

[3] Olle Wijk, Henrik I. Christensen “Triangulation-Based Fusion of Sonar Data with Application in Robot Pose Tracking”, 2000 IEEE Transactions on Robotics and Automation Vol.16, No.6

[4] A. Elfes, “Occupancy Grids: a probabilistic framework for robot perception and navigation PhD thesis”, Carnegie Mellon University, 1989

[5] A. Elfes, “Using Occupancy Grids for Mobile Perception and Navigation. Computer”, 22(6):46-57,1989

[6] H. Moravec and A. Elfes, “High-Resolution Maps from Wide-Angle Sonar”. In robotics and Automation, Proceeding ICRA’85, IEEE International Conference on, Los Alamitos, Calif., 1985. CS Pres.

[7] J. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. Robotics and Automation, IEEE Transactions on, 7(3):376-382,June 1991

[8] I.Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. Robotics and Automation, IEEE Transaction on,7(2):193-204, April 1991

[9] R. Chatila and J. Laumond. Position Referencing and Consistent World Modelling for Mobile Robots. In Robotics and Automation. Proceedings Society Press, March 1985

[10] R.Smith, M.Self, and P. Cheeseman. A Stochastic Map for Uncertain Spatial Relationships. In S. Iyengar and A. Elfes, editors, Autonomous Mobile Robots: Perception, Mapping, and Navigation, pages 323-330. IEEE Computer Society Pres 1991.

[11] M. Csorba. Simultaneous Localization and Map Building PhD thesis, University of Oxford,1997

[12] G. Zunino, H. I. Christensen. SLAM in Domestic Environments, Multisensor Fusion and Integration for Intelligent System. International Conference on 20-22 Aug.2001 Page(s): 67-72

[13] J. Choi, S. Ahn and W. K. Chung Robust Sonar Feature Detection for SLAM of Mobile Robot. IROS 2005. 2005 IEEE/RSJ International Conference on 2-6 Aug. Page(s) 3415-3420

[14] J. D. Tardos, J. Neira, P. Newman,J. Leonard, Robust Mapping and Localization in Indoor Environments Using Sonar Data, The International Journal of Robotics Research vol.21, n.4 April 2002 pp:311-330

[15] D. Ribas Towards Simultaneous Localization and Mapping for an AUV using an Imaging Sonar, Report on The Research Project, Girona University june, 2005

[16] S.Thrun, W. Burgard, D. Fox. “Probabilistic Robotics”, Intelligent Robotics and Autonomous Agents series, The Massachusetts Institude of Technology Press. Htpp://mitpress.mit.edu

[17] Jose A. Castellanos, Jose Neira, Juan D. Tardos, Map Building and SLAM Algorithm, webdiis.unizar.es/GRPTR/pubs/Caste_AMR_2006.pdf, 2005 pp:335-337

[18] Edouard Ivanjko, Mario Vasak, and Kalman Filter Based Mobile Robot Pose Tracking Using Occupancy Grid Maps, 2005 International Conference on Control and Automation (ICCA2005) June 27-29, 2005, Budapest, Hungary

[19] Billur Barshan, Directional Processing of Ultrasonic Arc Maps and its Comparison with Existing Techniques, 2007 The International Journal of Robotics

[20] Elif Eroğlu, Gezgin Robotlarda Ultrasonik Mesafe Algılayıcılarla Robot Davranışlarının Kontrolü ve Çevre Haritalama, Yüksek Lisans Tezi, Haziran 2006

[21] Thrun, S., Fox, D. and Burgard, W.,1998, A probabilistic approach to concurrent mapping and localization for mobile robots, Machine Learning, Vol.31

[22] Davison A.J. and Murray D.W., 1998, Mobile robot localization using active vision. In Proceedings of the 5th European Conference on Computer Vision, Freiburg, 809-825

[23] O. Wijk, P. Jensfelt, H.I Christensen Triangulation Based Fusion of Ultrasonic Data, 1998 Proceeding of the 1998 IEEE International Conference on Robotics & Automation Leuven, Belgium

[24] http://www.hostsrv.com/webmaa/app1/MSP/webm1010/chi2

[25] htpp://www.mobilerobots.com

[26] Olle Wijk, Triangulation Based Fusion of Sonar Data with Application in Mobile Robot Mapping and Localization PhD thesis, Royal Institute of Technology (KTH), Sweden 2001.

[27] R. De Maesschalck, D. Jouan-Rimbaud, D.L. Massart, Tutorial The Mahalanobis Distance, Chemometrics and Intelligent Labortory Systems 50 (2000) 1-18