YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

GEZGİN ROBOTLARLA EŞ ZAMANLI KONUM BELİRLEME VE HARİTALAMA

Bilgisayar Mühendisi Afig HABİBOV

FBE Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı: Yrd. Doç. Dr. Sırma YAVUZ (YTÜ)

İSTANBUL, 2011

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

x	Robotun konumu
u	Kontrol işareti
z	Algılayıcı tarafından elde edilen ölçüm
Θ	Harita
t	Zaman
μ	Gauss dağılımında ortalama değer
Σ	Kovaryans
p	Olasılıksal fonksiyon
ε	Rastgele durum değişkeni
δ	Rastgele ölçüm değişkeni
χ	Parçacık filtresinde parçacıkların kümesi
θ_t	Harita detayı
M	Parçacık sayı
η	Normalizasyon değeri
n_t	Harita detayının numarası
w	Parçacığın ağırlık değeri

KISALTMA LİSTESİ

CCITT	Comité Consultatif International Télégraphique et Téléphonique (Uluslararası Telgraf ve Telefon Danışma Komitesi)
CRC	Cyclic Redundancy Check (Dönüssel Artıklık Denetimi)
DC	Direct Current (Doğru Akım)
EZKBH	Eş Zamanlı Konum Belirleme ve Haritalama
GKF	Genişlendirilmiş Kalman Filtresi
hex	Hexadecimal (Onaltılı Sayı Sistemi)
IDE	Integrated Development Environment (Tümleşik Geliştirme Ortamı)
ITU-T	Telecommunication Standardization Sector (Telekomünikasyon Standardizasyon Sektörü)
jnilib	Java Native Interface Library
JRE	Java Runtime Environment (Java Çalışma Ortamı)
KF	Kalman Filtresi
PF	Parçacık Filtresi
SLAM	Simultaneous Localization and Mapping

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖNSÖZ

Robotlar, ister sosyal, ister endüstriyel, isterse de askeri çalışmalarda başrolleri oynama yolunda emin adımlarla ilerliyorlar. Herhangi bir robotun bulunduğu ortamdaki konumu bilmesi ve o ortam ile ilgili bilgileri anlaması maalesef bir tek komutla mümkün değildir. Bunun için çok fazla yöntemler geliştirilmiş ve uygulanmıştır. Bu konu benim de ilgimi çok uzun zamandır çekiyordu. Ama konuya aşina olmadığım için kendime özgüvenim azdı. Tez danışmanım ve değerli hocam Yrd. Doç. Dr. Sırma YAVUZ' dan aldığımı bilgiler sayesinde konuya yavaş-yavaş hâkim olmaya başladım ve sonunda ortaya böyle bir çalışma çıktı. Desteklerinden dolayı hocama sonsuz teşekkürlerimi sunuyorum.

Ayrıca, eşsiz bilgilerini paylaştıklarından dolayı bölümümüzün araştırma görevlilerinden Zeynep KURT ve Ozan ÖZIŞIK' a teşekkür etmeden geçemiyorum. Bu tez sürecinde her zaman yanımda olan, mühendislik bilgisini benimle paylaşan Mete KART' a ve son olarak hiçbir zaman maddi ve manevi desteklerini esirgemeyen, vatanımdan uzakta, İstanbul' da ailem olan, dostlarım Rasim BABAYEV ve Şamil ALİYEV' e ne kadar teşekkür etsem az.

ÖZET

Son zamanlarda artan gezgin robot uygulamalarında ortamın haritasın çıkarılması ve senkronize olarak robotun bulunduğu konumu belirlemesi problemine farklı yaklaşımları göre biliyoruz. Herhangi bir bilinmeyen ortamda robotun kendi konumunu öğrenmesi ve gerçek konum bilgilerine maksimum yaklaşması, bu işlemi yapması için ortamın haritasının çıkarması meselesi Eş Zamanlı Konum Belirleme ve Haritalama olarak bilinmektedir.

Haritalama için çözümler sunulduğunda ilk önce duyarga sistemlerinin konfigürasyonu araştırılmakta ve en iyi sonuçların alınması için pahalı tarayıcılar kullanılmaktadır. Bu çalışmada ise her şeyden önce maliyet düşürülerek kızılötesi algılayıcılar kullanılmıştır. Haritalamada ise soncul (posterior) değerleri çarpımı yani FastSLAM algoritması kullanılmıştır. Eş Zamanlı Konum Belirleme ve Haritalama çalışmasının doğruluğu Yıldız Teknik Üniversitesi, Bilgisayar Mühendisliği, Yüksek Lisans mezunu Ersin DENİZ' in geliştirdiği ERSIM simülasyon ortamında denemiştir. Simülasyon kumanda ortamı, NetBeans tümleşik geliştirme ortamı kullanılarak Java programlama dilinde geliştirilmiştir.

Elde edilen sonuçlar kıyaslanarak en iyi sonuca varılmaya çalışılmıştır.

Anahtar kelimeler: Eş Zamanlı Konum Belirleme ve Haritalama, FastSLAM, Gezgin Robotla Haritalama, Gezgin Robot Simülasyonu

ABSTRACT

Recently, we can see the different approaches to the problem of determining location according to synchronized built map of the environment on increased mobile robot applications. Determining the location and convergence it to the real position according the mapping of an any unknown environment is called Simultaneous Locatization and Mapping problem.

For the representing the solutions to the mapping it is important to analyze the good configuration for sensor systems and for getting the best results, expensive laser sensor systems are used. In this thesis, first of all are used infrared sensors for reducing the cost of operation. At the mapping stage there was used the Factored SLAM Posterior Method (FastSLAM). Accuracy and results of the Simultaneous Localization and Mapping algorithm has been tested and seen on ERSIM simulation platform which has developed in 2009 by MS's degree graduate student Ersin DENIZ in Yıldız Technical University, Computer Engineering department. Simulation command environment is developed using NetBeans IDE in Java programming language.

The obtained results was compared for the getting best one.

Keywords: Simultaeous Localization and Mapping, FastSLAM, Mobile Robot Mapping, Mobile Robot Simulation.

1. GİRİŞ

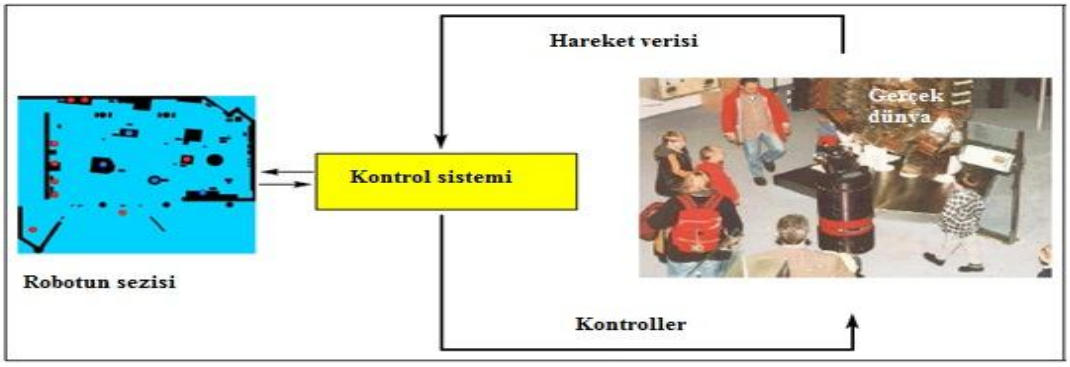
Tezde kullanılan robot simülasyonu, geliştirilen algoritma yardımı ile çevrenin ön topolojik bilgilerine ya da birtakım referans nesnelerinin yer bilgisine sahip olmadan ve ihtiyaç duymadan Eş Zamanlı Konum Belirleme ve Haritalama (EZKBH) yapmaktadır. Dolayısıyla robot bilinmeyen bir ortamda bilinmeyen bir noktadan harekete başlayarak bir taraftan bu ortamın haritasını çıkarırken, bir taraftan da kendi yerini tahmin etmekte ve başlangıç noktasına döndüğünü de algılamaktadır. Tasarlanan simülasyonda robot gezinme esnasında etrafındaki olayları algılayıp, nereye gideceğine yine kendi başına karar verecektir. Kullanılan robot simülasyonu sadece kızılötesi algılayıcılar kullanarak etrafındaki cisimlere olan uzaklığını ölçmektedir. Bu ölçümlerden yola çıkarak yoluna nasıl devam edeceğine karar vermekte ve konumunu bildirmektedir.

1.1 .Tezin Amacı ve Kapsamı

EZKBH yöntemleri ile yapılan çalışmalarda algılayıcıların geri döndüğü değerlerin doğruluğu büyük önem taşımaktadır. Genellikle bu tür sorunlarla karşılaşmamak için pahalı lazer duyarga sistemleri kullanılıyor. Sosyal-günlük yaşamda kullanılan akıllı robot tipi makinelerde ise böyle çok maliyetli sistemler kullanılmadığından çalışmada altı kızılötesi algılayıcı, üç tekerlekli, dikdörtgen şeklinde olan robot modeli kullanılmıştır. Robot modeli ve özellikleri hakkında daha detaylı bilgiler sonraki bölümlerde verilmiştir. Tezde yukarıda belirtilen özelliklerden yararlanarak 'Çevirim İçi' haritalama yapılmaktadır. Kullanılan EZKBH algoritması ise FastSLAM' dır. Yapılan çalışmanın sonuçları ise ERSIM ortamı çalıştırarak elde edilmiştir (Deniz, 2009).

1.2 .Robotikte İnanç ve Durum

Aşağıdaki Şekil 1.1' de robotun etrafındaki hareketli veya hareketsiz nesneleri kendinin oluşturduğu ortam modelindeki görünümü verilmiştir. Robotlar ortam hakkındaki bilgiyi duyargaları yardımıyla edinmektedirler. Bazen duyarga, bazen ortam, bazen robot sezisi (inanç), bazen de nesnelerin hareketleri ortamın haritasını (modelini) tam çıkarmayı önlemektedir (Thrun vd., 2005).



Şekil 1.1 Robot ortam algılaması (Thrun vd., 2005)

Reel dünyadan makine ortamına geçişte ortam bilgileri durumlarla karakterize edilir. Durum zamana ve nesnelerin hareketine göre değişebilir veya herhangi bir düz duvarın takip edilmesi zaman sabit kalabilir. Bununla birlikte kendinde robota ait; robotun konumu, hızı, duyargaların doğru çalışması gibi bazı değişkenleri de tutmaktadır. Bundan sonra t zamanındaki durumu x_t ile göstereceğiz. Bazı durum değişenleri aşağıdakilerdir:

- Robot konumu, onun genel haritaya göre nerede olmasını Kartezyen koordinatlarla gösterilmesidir. Bu bilgiler iki koordinat ve yön bilgisi olarak verilir. Bu bilgilere aynı zamanda Kinematik durum olarak da bilinir.
- Robot hızı
- Robot bileşenlerinin konfigürasyonu
- Ortamdaki objelerin hakkında bilgiler; onların konumu ve s.
- Hareketli nesnelerin hızı ve konumları ve s.

x_t durum bilgisi gelecek durumla ilgili bilgiyi tek başına hepsini verebiliyorsa ‘kapalı’ veya ‘tam’ durum bilgisi oluyor (Montemerlo, 2003). Başlangıç anında $t=0$ olarak alındığı varsayılmaktadır.

Robotun ortam bilgilerini anlaması, belli bir anda nerede olması gibi sayısal değerleri aşağıda yazılmış olan verilerle belirleniyor:

- **Duyarga ölçümleri:** Kamera, lazer, kızılötesi ve başka donanımlar yardımıyla elde olunan verilerdir ki, bunlar anlık gecikmeyle toplanır ve bu da robotun hareketini biraz yavaşlatır.
- **Kontrol bilgileri:** Bunlar robotun hareketi ve ortamdaki objelerin yer değişmelerine bağlı olmaktadır. Robot her hangi bir anda durmuş olabilir ama objelerin yerlerinin değişebilmesi yine robota yeni kontrol bilgilerinin gelmesine neden olacaktır.

- **Ölçüm verileri:** Bu veriler ortam hakkındaki anlık bilgileri kendinde tutmaktadır. *t* anındaki ölçümü z_t ile göstereceğiz.

$$z_{t_1:t_2} = z_{t_1}, z_{t_1+1}, z_{t_1+2}, \dots, z_{t_2} \tag{1.1}$$

t_1, t_2 arasında elde edilen bütün ölçümlerin kümesidir $(t_1 \leq t_2)$.

- **Kontrol verisi:** Kendisinde ortamın durumunun değişmesi hakkında bilgiyi tutmaktadır, mesela hız gibi. Her 5 saniyede 10 cm giden robotun on saniye sonra başlangıç yerinden 20 cm uzaklaşması, yeni kontrol emrinden önceki en gerekli bilgilerden biridir. Bu veri genel olarak robot tekerleğinin çapına bağlı olup, onun dönme sayısı ile doğru orantılıdır. Kontrol verisi u_t ile $(t-1, t]$ aralığındaki durum değişikliği ile bağlı olacaktır. $(t_1 \leq t_2)$ için

$$u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, u_{t_1+2}, \dots, u_{t_2} \tag{1.2}$$

bilgisini içerecektir.

1.3 .Olasılıksal Denklemler

Durum ve ölçümler olasılıksal denklemlerle gösteriliyor. x_t ve z_t bilgisi genel olarak stokastik olarak oluşturuluyor ve şöyledir:

$$p(x_t | x_{0:t-1}, z_{1:t-1}, u_{1:t}) \tag{1.3}$$

$$p(z_t | x_{0:t}, z_{1:t-1}, u_{1:t}) \tag{1.4}$$

Eğer x_t bir 'tam' durum verisi olursa o zaman yukarıdaki denklem aşağıdaki gibi yazılabilir:

$$p(x_t | x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t | x_{t-1}, u_t) \tag{1.5}$$

$$p(z_t | x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t | x_t) \tag{1.6}$$

2.OLASILIKSAL KONUM BELİRLEME YÖNTEMLERİ

Olasılıksal konum belirleme yöntemlerinde yapılan işlemler hareket ve duyarga güncellemesi olarak ayrılabilir. Hareket güncellemesi robotun hareketi zamanı her hangi bir $t-1$ zamanı için kontrol işareti ile t anında nerede olabileceğini anlayabilmesidir. Algı aşamasında ise robot belli bir zaman aralığında yaptığı ölçümler sonucunda kendi konumunu belirler.

Kullanılan temel parametrik yöntemlerden Kalman, parametrik olmayanlardan ise Parçacık filtrelerini gösterebiliriz.

2.1 .Kalman Filtreleri (KF)

Devamlı sistemler için en iyi öğrenilmiş ve gerçekleştirilmiş Bayes filtresi Kalman Filtresidir. KF Bayes filtresinin parametrize edilmiş formasıdır. Bu filtrede durum bilgisi Gauss fonksiyonu şeklinde hesaplanır. Uygun olarak kontrol ve ölçüm değerleri de Gauss gürültüsü eklenmiş parametreler ile gösterilir.

Kalman filtresinde inanç, μ_t ortalama değerli, Σ_t kovaryanslı Gauss fonksiyonu gibi yazılır.

$$p(x) = \det(2\pi)^{-\frac{1}{2}}\ exp\left\{-\frac{1}{2}\ (x-\mu)^{T-1}(x-\mu)\right\} \tag{2.1}$$

(2.1) eşitliğindeki μ vektörü x'la aynı boyutta, □ ise, x'in boyutunda kare matristir.

Yukarıdaki denklemden yola çıkarsak ve x durum vektörü için $p(x_t|x_{t-1}, u_t)$ ihtimalini yazacak olursak, bu ihtimal o zaman parametrelerinin Gauss gürültüsü eklenmiş liner fonksiyonu olacaktır.

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \tag{2.2}$$

Burada x_t ve x_{t-1} t anındaki durum vektörleri, u_t kontrol vektörüdür. Onları aşağıdaki gibi dikey vektörler şeklinde gösterebiliriz:

$$x_t = \begin{pmatrix} x_{1,t} \\ x_{2,t} \\ . \\ . \\ . \\ x_{n,t} \end{pmatrix} \qquad u_t = \begin{pmatrix} u_{1,t} \\ u_{2,t} \\ . \\ . \\ . \\ u_{m,t} \end{pmatrix} \tag{2.3}$$

A_t kare matris olup $n \times n$ boyutlarında, B_t ise $n \times m$ boyutlarında geçiş matrisleridir. Burada n ve m sırasıyla x_t ve u_t vektörlerinin uzunluklarıdır. ε_t rastgele değişeni durum değişkeni ile aynı ölçülerde olup, $\mu = 0$ ve □ = R_t şartları ödemektedir. (2.1) denklemini $A_t x_{t-1} + B_t u_t$ ortalama değerli ve R_t kovaryansı için yazarsak:

$$p(x_t|x_{t-1}, u_t) = \det(2\pi R_t)^{-\frac{1}{2}} \, exp\left\{-\frac{1}{2}\,(x_t - A_t x_{t-1} - B_t u_t)^T {R_t}^{-1}(x_t - A_t x_{t-1} - B_t u_t)\right\} \quad (2.4)$$

bu görünümü kazanır.

$p(z_t|x_t)$ de kendi parametrelerinin doğrusal fonksiyonu olduğuna göre

$$z_t = C_t x_t + \delta_t \quad (2.5)$$

olmalıdır. Burada da C_t $k \times n$ ölçülü geçiş matrisi ($k - z_t$ ölçüm vektörünün uzunluğudur). δ_t ise ölçüm gürültüsünü belirtmektedir. δ_t için de $\mu = 0$ ve □ = Q_t değerleri temel alınacaktır. Bütün bunları yeni denklemde yerine yazarsak, o zaman denklemimiz

$$p(z_t|x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} \, exp\left\{-\tfrac{1}{2}\,(z_t - C_t x_t)^T {Q_t}^{-1}(z_t - C_t x_t)\right\} \quad (2.6)$$

şeklini almaktadır.

Son olarak başlangıç inanç olan $bel(x_0)$' da normal dağılımlı olmalıdır.

$$bel(x_0) = p(x_0) = \det(2\pi_0)^{-\frac{1}{2}} \, exp\left\{-\tfrac{1}{2}\,(x_0 - \mu_0)^T {}_0{}^{-1}(x_0 - \mu_0)\right\} \quad (2.7)$$

2.1.1 Kalman filtresi algoritması

Yukarıda belirttiğimiz gibi Kalman filtresi zamanı bütün veriler, yani hareket ve ölçüm modelleri Gauss dağılımı ile gösterilmeli, her bir inanç fonksiyonu kendine özgün ortalama değer ve kovaryans ile belirtilmelidir. Bir önceki ortalama değer ve kovaryans hareket ve duyarga bilgilerine dayanarak sonraki ortalama değer ve kovaryanslar hesaplanmalıdır. Kalman sürecinde kontrol verileri bellidir. Ama sistemin gürültüsünü bilmediğimizden sürecin sonunda elde edeceğimiz küme gerçek kümeye ola bildiğince yakın olmalıdır.

Kalman filtresi geniş kullanılmasına rağmen veri ilişkilendirme gibi önemli bir probleme çözüm getirememektedir. Bundan başka Kalman filtresinin haritalamadaki kısıtları aşağıdakilerdir:

- Yapay sınırlara veya duvarlara ihtiyaç duyulur
- Direk ham algılayıcı datası kullanılmaz, veriler ön işlemlerden geçirilmelidir
- Algılayıcı verisinin sahip olduğu gürültü tipi Gauss dağılımında olmalıdır
- Başlangıçta robot pozisyonunda kayma olmadıysa harita iyi çıkar, ancak kayma olduysa, zaman geçtikçe bu kayma büyüyeceğinden kötü sonuçlar alınmaktadır.
- Sadece doğrusal sistemler için çalışabilmektedir.

2.1.2 Genişletilmiş kalman filtresi (GKF)

Bir önceki başlıkta Kalman filtresinin bazı artı ve eksileri gösterilmiştir. Eksilerinden en önemlisi ise sadece doğrusal sistemler için çalışmasıdır. Yani, durum ve ölçüm modelleri doğrusal denklemlerle ifade edilmelidir. Ama ya robot herhangi bir dairesel yolu takip ederse veya kontrol fonksiyonu doğrusal değilse? Bu sorulara cevap vermek için filtremiz doğrusal olmayan yani non-linear sistemler için çalışmalıdır (Kalman, 1960). Bu yüzden Kalman filtresinin daha genel ifadesi olan Genişletilmiş Kalman Filtresi meydana çıkmıştır.

GKF' de bir sonraki durum ve ölçüm olasılık fonksiyonları sırasıyla g ve h ile kumanda edilmektedir.

$$x_t = g(u_t, x_{t-1}) + \varepsilon(t) \tag{2.8}$$

$$z_t = h(x_t) + \delta(t) \tag{2.9}$$

Burada g fonksiyonu Kalman filtresindeki A_t ve B_t' in, h ise C_t' in yerine geçmiştir. Ama KF' den farklı olarak GKF gerçek inanç değerine yakın bir değer elde etmektedir.

(2.8) ve (2.9) denklemlerindeki g ve h fonksiyonları Gauss dağılımına sahip olmadıklarını biliyoruz. GKF ilk adımı bu fonksiyonları doğrusallaştırma olacaktır. Bu işlemi de Taylor sırası açılımına göre yazabiliriz. O zaman,

$$g'(u_t, x_{t-1}) = \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}} \tag{2.10}$$

olacaktır. Doğrusal olmayan g fonksiyonunu Gauss fonksiyonu doğrultusunda doğrusallaştırmak için parametreleri daha uygun seçmek gerekir. Bu durumda g için gerçek değerlere yakın bir parametre olarak μ_{t-1} seçilmiştir.

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \underbrace{g'(u_t, \mu_{t-1})}_{=:G_t}(x_{t-1} - \mu_{t-1}) =$$

$$= g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}) \tag{2.11}$$

O zaman bir sonraki durum ihtimali yaklaşık olarak Gauss fonksiyonu gibi şöyle yazılabilir:

$$p(x_t|x_{t-1}, u_t) \approx \det(2\pi R_t)^{-\frac{1}{2}} exp\begin{Bmatrix} -\frac{1}{2}[x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})]^T \\ R_t^{-1}[x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})] \end{Bmatrix}$$

(2.12)

Burada $G_t$matrisi $n \times n$ ölçülerinde olup (n – durum verisinin ölçüsüdür) Jacobian olarak da bilinmektedir. Bu Jacobian' ın değeri u_t ve μ_{t-1}değerlerine bağlıdır ve her bir nokta için farklıdır.

Aynı yolla bütün bu denklemleri h ölçüm fonksiyonu için yazabiliriz:

$$h'(x_t) = \frac{\partial h(x_t)}{\partial x_t} \tag{2.13}$$

$$h(x_t) \approx h(\bar{\mu}_t) + \underbrace{h'(\bar{\mu}_t)}_{=:H_t}(x_t - \bar{\mu}_t) = h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t) \tag{2.14}$$

(2.14)' e göre Gauss fonksiyonunu yazacak olursak:

$$p(z_t|x_t) \approx \det(2\pi Q_t)^{-\frac{1}{2}} exp\begin{Bmatrix} -\frac{1}{2}[z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t)]^T \\ Q_t^{-1}[z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t)] \end{Bmatrix} \tag{2.15}$$

GKF' in bazı artıları olduğu gibi birkaç eksisi de mevcuttur.

- Yaklaşık Gauss fonksiyonu hesaplanması zamanı sistemin derecesi belli olmadığından, bu işlemlerin karmaşıklığını de beraberinde getirir. Sistemin derecesi arttıkça Taylor sırası uzar.
- Jacobian matrislerinin hesaplanması işlemlerin sayının artmasına getirir.
- Doğrusallaştırma, bizi gerçek inanç değerinden uzaklaştırır.

2.2 .Parçacık Filtresi (PF)

Parçacık filtresi Bayes filtrelerinin parametrik olmayan alternatif realizasyonu ve sonlu sayıda parametrenin reel inanç değerine yakınsamasıdır. PF' in ana konusu $bel(x_t)$ değerini, ondan önceki $bel(x_{t-1})$ değerinden, rastgele durum örnekleri kümesi ile oluşturulmasıdır (Crisan ve Doucet, 2002). Ama GKF' de olduğu gibi bu değerin kendisini değil de ona yakın bir değeri bulmaktadır. Bunu yaparken dağılımların Gauss dağılımı olması zorunluluğu da yoktur.

PF' de önceki durum dağılımlarını 'parçacık' adlandırılmakta ve aşağıdaki gibi gösterilmektedir:

$$\chi_t := x_t^{[1]}, x_t^{[2]}, \dots, x_t^{[M]} \tag{2.16}$$

Burada her bir $x_t^{[m]}$ $(1 \leq m \leq M)$ parçacığı, t anındaki durumu gösteriyor. M ise burada χ_t parçacık kümesindeki parçacıklarını sayını göstermektedir. Genellikle parçacık sayısı $M = 1000$ gibi büyük rakamlar oluyor. Bazen de M, t'in fonksiyonu olabilir. Yukarıdaki gösterimden yola çıkarsak:

$$x_t^{[m]} \sim p(x_t | z_{1:t}, u_{1:t}) \tag{2.17}$$

gibi yazabiliriz.

PF algoritması da GKF gibi inanç değerlerini yaklaşık olarak hesaplamaktadır. Tam dakik olmayan böyle hesaplamalar zamanı yakınsama hataları meydana çıkmaktadır. Aşağıda bazı hatalar ve oluşma sebepleri verilmiştir:

- Parçacık sayısının sonlu olması: Mesela $M = 1$ varsayalım. O zaman döngü bir alt aşamaya geçmeyerek ölçüm verilerini kullanmayacak ve yeni parçacıklar sadece $p(x_t, u_{1:t})$ değerinden hesaplanacaktır (Thrun vd., 2005).
- 'Resampling' aşamasının rastgele olması: Bu hatayı göstermek için en uç örneği vermemiz lazım. Varsayalım ki robot durumu değişmiyor, yani $x_t = x_{t-1}$. Yani,

hareket etmeyen bir robot örneğini inceliyoruz. Bir de robotun sensor ölçümü yapmadan durum algılaması yapamayacağını düşünelim. Bu durumda ise yeni durum örneği hesaplanamayacak. Sonuç olarak, bütün durum örnekleri bir birinin kopyaları olarak belirecek ve robotun sensorlarının çalışmadığı gibi hiç de gerçek olmayan bir bilgi elde edilmiş olacaktır (Montemerlo, 2003).

- Hedef ve göreceli dağılımların oranı: Sensorlara ve robot hareketine bağlı olarak bu oran değişmektedir. Sensor verilerinin gürültülü olduğu durumlarda $p(z \mid x)$ değeri bütün durumlar için sıfır değeri alacak (Hahnel vd., 2003). Bu durumda dağılım fonksiyonu z ölçüm değerleri dâhilinde örnek x üretmeyecektir. Böyle olan halde ağırlık değerli de sıfıra eşit olmaktadır. Sonraki döngü ise devam etmeyecektir.
- Yüksek dereceli sistemler daha fazla sayıda parçacık ile filtreleme talep ediyor ki bu da beraberinde işlem karmaşıklığını dolayısıyla sistemin pahalı olmasına getirip çıkarıyor.

3.EŞ ZAMANLI KONUM BELİRLEME VE HARİTALAMA

Gezgin robotlarla harita çıkarımı son yıllarda aktif olan bir araştırma konusudur. Haritalama, gezgin robotların, gerçek dünyadaki ortamın x-y-z düzlemindeki iz düşümlerine göre gösterme problemidir. Otonom olan gezgin robotların başlıca problemi harita oluşturmaktır. Bulunan çözümler genellikle olasılıksal yöntemler olup olasılıksak denklemlerle ifade edilmektedir. Bunun sebebi ise robotun hareketi zamanı ve algı ölçümleri sırasında oluşan belirsizliktir ki, bu bilgilerde oluşan hatalar haritanın bütününe etki etmektedir. Robotun hareket ve ölçüm verilerinin kesin olmaması haritalama problemi için olasılıksal yöntemleri eşsiz kılmaktadır. Eş zamanlı konum belirleme ve harita oluşturma problemi, ilk defa 1980'lerin sonunda, 1990'ların başında teknolojinin daha da gelişmesi ile ortaya çıkarılmıştır. İlk yıllarda temel olarak metrik ve topolojik yöntemler olarak ikiye ayrılmıştır. Metrik haritalar çevrenin geometrik özelliklerini gösterirken, topolojik haritalar farklı bölgelerin birbirine bağlı olup olmadığını gösterir (Kurt 2007). Chatila ve Laumond (1985) çevrenin geometrik özelliklerini tanımlamak amacıyla bir dizi çokgen kullanarak metrik haritalama yöntemlerini geliştirmişlerdir. Topolojik haritalama yöntemleri çalışmalarına ise Matari´c (1990) ile Kuipers ve Byun (1991)' un çalışmalarını örnek gösterebiliriz. Metrik ve topolojik haritalama arasındaki fark yeterince belirgin olmadığından bazen bu alandaki çalışma sonuçları bir birine benziyor. Ama bununla beraber aralarında farklar vardır. Bu farklar ve belirgin özellikler aşağıda belirtilmiştir (Huang vd., 2006):

a) Metrik haritalar topolojik haritalardan daha çok detay barındırdığından veri ilişkilendirme problemine iyi çözüm getirir.
b) Yüksek çözünürlük gerektirdiğinden ve hesapsal yükü ağır olması ise metrik haritaların topolojik olanlar karşısındaki eksilerindendir.

Daha sonra EZKBH algoritmalarının gelişim sürecinde dünya merkezli ve robot-merkezli olmak üzere ikinci bir sınıflandırma daha yapılmıştır (Kurt 2007). 90'lardan başlayaraksa haritalamada olasılıksal yöntemler daha sık kullanılmaya başlanmıştır. Harita ve robot pozisyonu belirlenmesinde istatistiksel yöntemlerden en sık kullanılanı Kalman Filtreleri olmakla beraber alternatif haritalama ve konum belirleme algoritmalarına Dempster'ın (1977) geliştirdiği "expectation maximization (EM)" (beklenti enbüyültme - BE), FASTSLAM örnek olarak gösterilebilir. Robot gezinmesi ve haritalama probleminde daha iyi sonuçların elde edilmesi için pahalı çözümler geliştirilmektedir (Julier ve Uhlmann, 2001). Günümüzde üretilen çözümlerde çoğunlukla pahalı duyarga sistemleri kullanılarak bilgi kazanımını artırmak hedeflenmektedir.

3.1 .Haritalama Sürecinin Özellikleri

Haritalama sürecinin en belirgin özellikleri ve karşılaşılan problem aşağıda sıralanmıştır:

a) Harita çıkarma problemi, genellikle robot pozisyonu belirleme yani lokalizasyon problemi ile birlikte ele alınır. Çünkü etraftaki nesnelerin konumunu bilmek için başka bir deyişle harita çıkarabilmek için robot kendi konumunu ve buna bağlı olarak da algılayıcılarının konumunu bilmelidir; robotun kendi konumunu belirleyebilmesi için de gezindiği ortamda etrafında bulunan nesnelerin konumunu bilmelidir (Kurt 2007).

b) Robot, haritasını çıkarmayı amaçladığı ortamda gezinirken dış dünyayı algılayabilmelidir. Bunun için de kamera; sonar, lazer veya kızılötesi teknolojilerini kullanan mesafe ölçerler, pusulalar veya GPS'ler kullanabilirler. Ancak algılayıcıların verisi tamamen güvenilir değildir, ölçümler gürültülü olabilir, ayrıca çoğu algılayıcının görme – algılama mesafesi kısıtlıdır. Örneğin ışık ve ses duvarlardan geçemez (Siegwart ve Nourbakhsh, 2004).

c) Haritası çıkarılacak ortamda gezinirken üretilen hareket (kontrol) komutları, farklı algılayıcı ölçümlerinin alındığı konumlarla ilgili bilgi içerdikleri için haritalamada önemlidir (Siegwart ve Nourbakhsh, 2004). Robot aldığı komuta göre hareket ederken birtakım problemler çıkabilir, dolayısıyla robotun yeni pozisyonunu belirlerken sadece kontrol işaretlerine göre verilerin kesin olduğuna inanmak doğru değildir (Kurt 2007).

3.2 .Eş Zamanlı Konum Belirleme ve Haritalama Probleminde Kullanılan Temel Yöntemler

Eş zamanlı konum belirleme ve harita oluşturma problemi için geliştirilen algoritmaların bazıları artımlı yöntemler olduğundan, verinin tümünü bilmesine gerek yoktur, sadece bir önceki adımı bilmesi yeterlidir, dolayısıyla gerçek zamanlı uygulamalarda kullanılabilirler ancak bazısı da önceden geçilen tüm adımları bilmek zorundadır dolayısıyla gerçek zamanlı uygulamalarda kullanılamaz (Chatila vd., 1985). Bunlardan bazıları harita oluşturabilmek için robotun kesin pozisyonunu bilmek zorundadır. Diğerleri ise "veri ilişkilendirme" problemine çözüm bulabilmek için özel olarak geliştirilmiştir (Montemerlo, 2003) . Ancak kullanılan tüm yöntemlerin tek bir ortak özelliği istatistiksel yöntemlerle çözüm yolunun bulunmasıdır. Harita oluşturma ve robotun konumunun belirlenmesi esnasında algılayıcı gürültüsü, ölçüm belirsizliği, kontrol işaretlerinin bir problem yüzünden istendiği gibi uygulaması, vb durumlar la karşılaşılır ve çözüm olarak da istatistiksel kestirim yöntemleri kullanılır (Kurt 2007).

En çok kullanılan yöntemlerden bazılarını yukarıdaki bölümlerde verilmiştir. Bu algoritmaların içinde bir tanesi var ki, diğerlerinden işlem sayısına ve karmaşıklık derecesi göre farklıdır. Bu yöntem FastSLAM adlandırılmaktadır. İlerideki bölümlerde FastSLAM yönteminin geniş incelemesi verilmiştir.

3.3 .Önceki Çalışmalar

EZKBH alanında son yıllarda çok sayıda çalışmalar olmuştur. Bilinen algoritmalar geliştirilerek optimum seviyede çalıştırılması amaçlanmakta ve ya eski bilinen algoritmaların yerine onların yetersiz kaldığı noktaları daha da aydınlatmak sebebiyle yeni algoritmalar geliştirilmektedir.

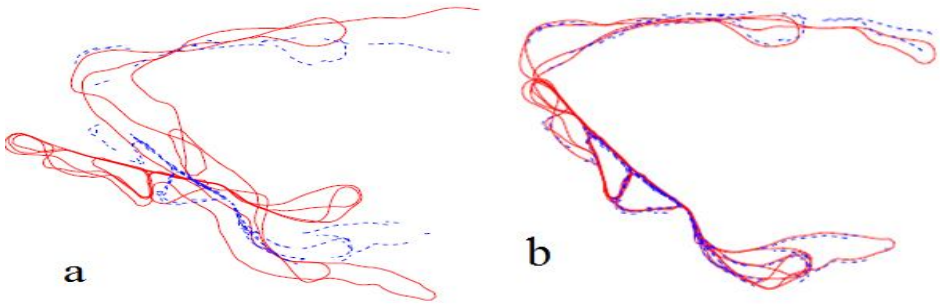
Bunlara örnek olarak Kalyaev vd. (1997) tarafından yapılan çalışmada algılayıcı olarak pahalı lazer sistemi kullanılmış, bilinmeyen her hangi bir ortamda gezinen robotun algılanan bu veriler yardımıyla konumunu belirlemesi ve düzeltme yapması amaçlanmıştır. Sonuçların başarılı olmasından dolayı Rus savunma sanayisinde kullanılmıştır.

Yun vd. (1998) tarafından, doğrusal özellikler kullanılarak robotun kendi konumunu belirlemesi çalışmasının yapılmasını görebiliriz.

Karpov vd. (2001) yapmış olduğu çalışmada ise her hangi bir komut almadan robotun bir ortamda kendi konumunu ve ortamın haritasının çıkarılması için yeni bir çözüm geliştirilmeye çalışılmıştır. Algılayıcı verisinden En Küçük Kareler yöntemi ile doğrular uydurulmuş, doğruların köşe oluşturduğu yerler belirlenmiş ve bunun sonucunda robot kendi konumunu kesinleştirmiştir.

Tardos vd. (2002) tarafından yapılan çalışmada ise algılayıcı olarak sonar algılayıcılar kullanılarak, elde edilen sonuçlara Hough Dönüşümü uygulanmıştır. Bunu sonucunda harita özellikleri doğrusal ve noktasal olarak elde edilmiştir.

Thrun vd. (2005) Victoria Park' ta gezinen lazer sensorlu araba çalışmasında o güne kadar çok az kullanılmış bir yöntemle parktaki detaylar, özellikler belirlenmiştir. Odometri verisi kullanılmadan sadece doğrusal ve açısal hız kontrolünden yararlanarak araba 30 dakika parkta gezdirilmiştir. Kullanılan algoritma FastSLAM olmuştur. Parktaki detaylar bulunarak gerçek haritadan ne kadar sapma olduğunun sebepleri araştırılmış ve ortadan kaldırılmaya çalışılmıştır.



Şekil 3.1 GKF(a) ve FASTSLAM(b) kullanılarak Victoria Park' ta gezinme (Thrun vd., 2003)

Bailey vd. (2004) tarafından yapılmış çalışmada, FastSLAM algoritması kullanılarak haritalamanın başlıca sorunlarından biri olan 'veri ilişkilendirme' sorununa çözüm aranmıştır. Alınan sonuçları GKF yöntemi ile elde edilen sonuçlarla kıyasladıktan sonra işlem sayında ve işlemlerin karmaşıklık derecesindeki farklar araştırılmıştır. Bununla beraber FastSLAM algoritmalarının da kendi aralarındaki ortak ve farklı yönleri belirlenmiş kullanılan robot modeline ve ortama göre hangi algoritmanın kullanılmasının uygunluğu gözden geçirilmiştir.

Beevers ve Huang (2006) tarafından yapılan çalışmada sınırlı sayıda algılayıcıya sahip bir robot ile Parçacık Filtresi kullanılarak EZKBH yapılmıştır. Diğer çalışmalarda olduğu gibi bu çalışmada da algı verileri kullanılarak elde edilen noktalardan doğrular çıkarılmıştır. Bundan önce noktalar kümelenmiş, her kümeye İteratif Uç Nokta uygulanarak gerektiği takdirde bu kümeler daha küçük kümelere ayrılmıştır. Bir eşik değer belirlendikten sonra bunun üstündeki sayıda noktası bulunan kümeler belirlenmiştir. Bu nokta kümelerinden doğrular çıkarılarak sadece doğruların uç noktaları özellik olarak alınmıştır.

3.4 .Eş Zamanlı Konum Belirleme ve Haritalama Meselesi

Yukarıdaki bölümlerde EZKBH konusunda, onun problemleri ve gerçekleştirme algoritmaları hakkında bazı bilgiler verilmiştir. Önceki konulardaki algoritmalara daha bir parametre, - harita durumu parametresi - ekleyerek EZKBH meselesini daha bütün bir şekilde göstereceğiz. Genel olarak bu parametreyi Θ olarak belirleyelim. Bu harita, her biri θ_n ile gösterilen detaylardan oluşmaktadır. Haritadaki bütün stasionar(durgun) detayların sayısı ise N olsun. Gezgin robotun konumunu s_t parametresi ile göstereceğiz. Konum ise iki ölçülü Kartezyen koordinatları şeklinde gösteriliyor. s^t ile t anına kadarki bütün konumları ifade edeceğiz (Montemerlo, 2003).

Robotun haritayı oluşturabilmesi için sensor kullanması şarttır. t anındaki sensor ölçümünü z_t ile ifade edeceğiz. Ölçüm ise kendiliğinde bulunan detaya olan uzaklığı göstermektedir. Her

bir z_t bulunan harita detayını $n_t \in \{1, \dots, N\}$ ile numaralandıracağız. Kontrol verimizi ise u_t ile gösteriyoruz. Bu veri ise direk robotun kinematik hareket modeline bağlıdır. O zaman GKF' de olduğu gibi ölçüm ve konum modelini birkaç değişiklik yaparak aşağıdaki gibi yazabiliriz.

$$p(z_t \mid s_t, \theta_{n_t}, n_t) = g(s_t, \theta_{n_t}) + \varepsilon(t) \tag{3.1}$$

$$p(s_t \mid u_t, s_{t-1}) = h(u_t, s_{t-1}) + \delta(t) \tag{3.2}$$

Burada $\varepsilon(t)$ ve $\delta(t)$ rastgele değişenleri, sırasıyla sıfır ortalama değerli ölçüm ve hareket gürültüleridir. Kovaryansları ise sırayla R_t ve P_t değerleridir.

$$p(s_t, \Theta \mid z^t, u^t, n^t) \tag{3.3}$$

Eğer, (3.3) olasılık fonksiyonu bir önceki adımdaki fonksiyondan rekursif olarak hesaplanırsa, o zaman aşağıdaki gibi gösterebiliriz:

$$p(s_t, \Theta \mid z^t, u^t, n^t) = \tag{3.4}$$

$$= \eta\, p(z_t \mid s_t, \theta_{n_t}, n_t) \int p(s_t \mid u_t, s_{t-1})\, p(s_{t-1}, \Theta \mid z^{t-1}, u^{t-1}, n^{t-1}) ds_{t-1}$$

$$p(s_t, \Theta \mid z^t, u^t, n^t) = \tag{3.5}$$

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

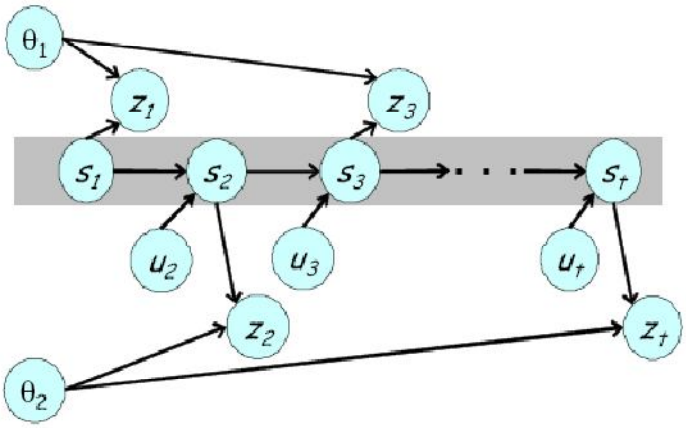
η burada normalizasyon değeridir. Soncul değerin hesaplanması zamanı hiçbir parametre ona etki etmediğinden genelleştirme yapabiliriz.

Bayes filtresinin birçok EZKBH algoritmalarının temelinde olduğunu biliyoruz. g ve h fonksiyonların doğrusal olduğu durum ise bildiğimiz Kalman filtresi oluyor. (3.3) formülü EZKBH için 'altın standart' gösterimidir (Perdomo, 2009). (3.5) ise bütün FASTSLAM algoritmalarının temelini oluşturmaktadır.

EZKBH posterior aşağıdaki gibi çarpım formatında gösterilebilir:

$$p(s^t, \Theta \mid z^t, u^t, n^t) = p(s^t \mid z^t, u^t, n^t) \prod_{n=1}^{N} p(\theta_n \mid z^t, n^t, s^t) \tag{3.6}$$

Bu çarpım posterior değerini (harita ve yol), bir yol çarpanı ve N tane detay konumu olmakla $N+1$ çarpan şeklinde gösterilebilmektedir. Bu çarpım özelliği daha önceki filtrelemelerde olduğu gibi yaklaşık değer değil, kesindir. Grafik bir gösterimle bunu ifade edelim:



Şekil 3.2 Çarpımlarla gösterilen harita detaylarıyla EZKBH meselesinin grafik gösterimi (Hahnel vd., 2003)

Şekil 3.1' de Robot $\boldsymbol{s_1}$ konumundan $\boldsymbol{u_{1:t}}$ kontrolleri etkisinde hareket etmekte olup, yakınında yerleşen detayları bulmaktadır. $\boldsymbol{t = 1}$ anında $\boldsymbol{\theta_1}$ detayını, $\boldsymbol{t = 2}$' de $\boldsymbol{\theta_2}$, $\boldsymbol{t = 3}$ zamanında ise yeniden $\boldsymbol{\theta_1}$ detayını fark ediyor. Gri renkli yol ise şartlı bağımsızlık ilişkisini gösteriyor.

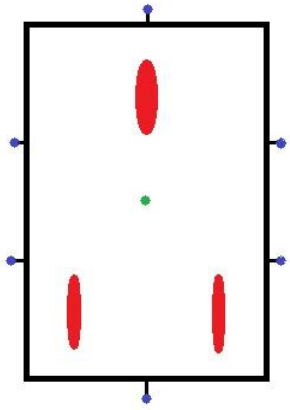
4.HARİTALAMA İÇİN ÖNERİLEN YÖNTEM

Bizim önerdiğimiz yöntem ile bundan önce yapılmış çalışmalardan farklı olarak EZKBH algoritması olarak FastSLAM kullanışının sadece harita detayı olarak çıkarılmasından öte, bulunan harita detaylarının yardımıyla ortamın optimum haritasının çıkarılması hedeflenmektedir. Çünkü daha önceki FastSLAM çalışmalarında ortamdaki detay(landmark) lar bulunmuştur ama ortamın sınırları bu detaylara göre çizdirilmemiştir. Buna örnek olarak Thrun (2003, 2005) ve Bailey (2004) in çalışmalarını gösterebiliriz. Çalışmamızı diğer çalışmalardan farklı kılan diğer bir gelişme ve bir anlamda artısı, bölümümüzde daha önceden yapılmış olan robot simülasyon ortamının kullanılmasıdır. Bu da bölümümüzde robot haritalaması alanının bir ilgi alanı olmasını ve çalışmaların sürekliliğini gösteriyor.

4.1 .Kullanılan Robot Modeli

Çalışmada Deniz (2009) tarafından simülasyonda gezdirilen robot modeli temel alınmıştır. Aynı robot modeli daha önce de belirttiğimiz gibi YTÜ, Bilgisayar Mühendisliği, Olasılıksal Robotik grubunun çalışmalarında da yer almıştır. Kullandığımız robot altı kızılötesi algılayıcıya sahip. Bunlardan birer tanesi ön ve arkada, ikişer tanesi ise robotun sağında ve solundadırlar. Yanlarda ikişer tane algılayıcının bulunmasının sebebi robotun duvara paralel gidip gitmediğini belirleyebilmemize yarayacaktır.

Hareketi sağlayan motorumuz ön tekerleğe bağlı olmakla robotun üç tane tekerleği var. Geri kalan iki tekerlek ise robotun arka kısmında robotun merkezine göre simetrik yerleştirilmiş ve bir birinden bağımsız olarak hareket etmektedirler.



Şekil 4.1 Simülasyonda kullanılan robotun temsili görünümü

Ön tekerlek mili üzerine yerleştirilmiş olan bir adet artımlı optik kodlayıcı (*incremental encoder*) ile alınan yol miktarı hesaplanabilmektedir. Artımlı kodlayıcının mekanik bağlantısı, ön tekerleğe göre yatay eksende eşmerkezli olarak yapılmıştır. Kodlayıcıya ait, 2 bayt ileri devir sayısı bilgisi, 2 bayt geri devir sayısı bilgisi ve 1 bayt da tekerlek devir bilgisi tutulmaktadır. İleri ve geri devir sayısı tekerin 1 tam turunda 512 artım yapmaktadır. Bundan başka robotumuzun parçaları aşağıda belirtilmiştir (Deniz, 2009).

a) Servo Motor: Alınan komuta göre ön tekerleği sağa veya sola döndürmektedir.
b) Döner Kodlayıcı: Ön tekerleğin dönme açısının ölçülmesini sağlamaktadır.
c) Mesafe Duyargaları: Robotun kenarlarına yerleştirilmiş olan çeşitli tiplerdeki duyargalar ile ölçüm yapılarak ortam hakkında veri toplanmaktadır
d) DC Motor: Robotun ileri ve geri yönlü hareketini sağlamaktadır.
e) Artımlı Optik Kodlayıcı: Ön tekerleğin ileri ve geri yönde dönme miktarlarını ve devir sayısını hesaplayabilmeyi sağlamaktadır.

Robotun eski pozisyondan yeni konuma geçmesi için rotasyon matrisi (Bailey vd., 2004, Manseur, 2006) aşağıdaki gibi olacaktır:

$$\begin{pmatrix} 1 & 0 & -tVsin(\theta+\alpha) \\ 0 & 1 & tVcos(\theta+\alpha) \\ 0 & 0 & 1 \end{pmatrix} \tag{4.1}$$

Burada V - doğrusal hız, θ- robotun yönünü koordinat eksenleri göre bildiren açı, α ön tekerleğin dönme açısıdır, d ise ön teker ile arka tekerlekleri birleştiren doğrunun orta noktası arasındaki mesafedir.

4.2 .Robotun Hareket Modeli

Robotun gittiği yolun uzunluğunu belirlemek için Artımlı Optik Kodlayıcıdan iki zaman arasında gelen bilgi kullanacağız (Kurt 2007, Deniz, 2009).

$$Gidilen\ mesafe\ (ds) = \frac{encoder_t - encoder_{t-1}}{512} * 2\pi r_{tekerlek} \tag{4.2}$$

Alınan mesafeyi bu zaman aralığına bölersek robotun hızını almış oluruz. Kontrol işareti olarak kullanacağımız diğer veri ise açısal hızdır ki bunu ise robotun dönme zamanı çizdiği

yaya uygun gelen merkezi açının dönme zamanına oranı olarak belirliyoruz. Böylelikle $(v, w)^T$ kontrol verimizi almış oluyoruz.

Robotun kinematik modeline göre bir sonraki konum (yani t anındaki) ise aşağıdaki gibi olacaktır:

$$\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} tVcos(\theta + \alpha) \\ tVsin(\theta + \alpha) \\ (tV/d)sin\alpha \end{pmatrix} \quad (4.3)$$

Yukarıdaki gösterimde $(x_{t-1}, y_{t-1}, \theta_{t-1})^T$ bir önceki konum bilgilerini, $(x_t, y_t, \theta_t)^T$ ise şu andaki konum bilgilerini ifade etmektedir.

Kontrol işareti olara odometri verisini de kullanabilirdik ama işlemlerin karmaşık olması sebebiyle Bailey vd., (2004) ve Manseur (2006) çalışmalarında olduğu gibi hız hareket modeli kullanmanın verimli olacağı belirlenmiştir.

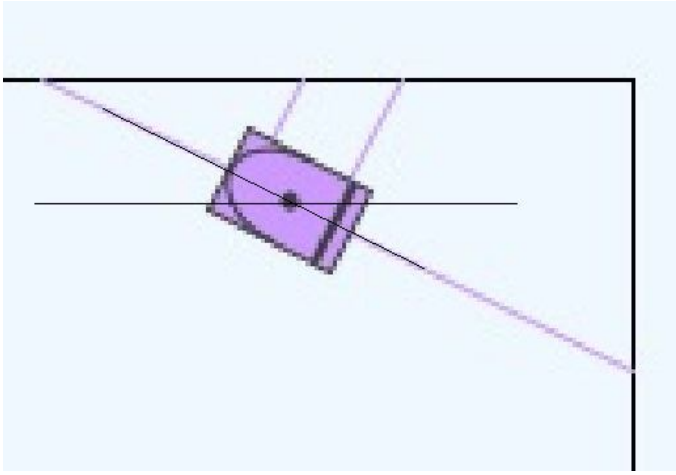
4.3 .Robotun Sensor Modeli

Robotun sensor modeli genellikle algılayıcı ile robot gezinmesi zamanı bulunan harita detayı arasındaki mesafe ve bu detayın robota göre konumunu açı olarak gösterilmesidir (Montemerlo, 2003).

Bu bilgilerden yola çıkarak simülasyonda kullanılan robotun sensor modeli ise aşağıdaki gibi olacaktır:

$$z_s = \begin{pmatrix} r \\ \theta_s \end{pmatrix} = \begin{pmatrix} \sqrt{(x_s - x_i)^2 + (y_s - y_i)^2} \\ arctan\frac{y_s - y_i}{x_s - x_i} - \theta \end{pmatrix} \quad (4.4)$$

Burada r - ölçüm mesafesi (algılayıcıdan harita detayına kadar olan uzaklık), θ_s – harita detayı ile robot arasındaki açı, (x_s, y_s) algılayıcının konumu, (x_i, y_i) ise bulunan detayın konumudur. Şekil 4.2' de simule edilen robotun ortamda gezinmesi zamanı ölçüm verilerini toplaması gözterilmiştir.



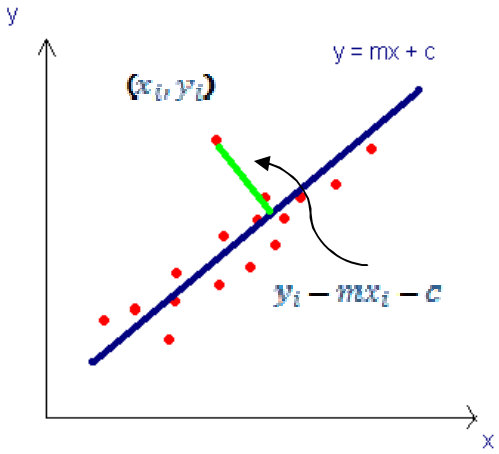
Şekil 4.2 Algılayıcıların duvar tespit etmesi durumu

4.4 .Engel Takip Etme Yöntemi ile Haritalama

Robotun kontrolü ve ortamda gezinmesi için engel takip etme algoritması kullanılmıştır. Simülasyonda taklit edilen robot, ortamda her hangi bir engel bulduktan sonra kendini bu engele paralel hale getirmeye çalışmaktadır. Bulunan engelin takip edilmesi sonucunda ise robotun bütün ortamı gezinmesi amaçlanmaktadır (Astapkovich, 2009). Engel Takip Etme Yönteminin adımları şöyledir: Robotun başlangıç konumuna bir değer atandıktan sonra ortamda bir engel bulunup robot bu engele paralel konuma getirilir. Sonraki adımlar sürekli olarak veya başlangıç konumuna dönene kadar tekrarlanır. Duyarga ölçüm değerleri ile ortamdaki nesnelerin konumları hesaplandıktan sonra ise ortam hakkında bilgilerden yararlanarak takip edilen engelin köşe ve ya düz duvar olması tahmin edilir (Deniz 2009).

4.5 .Duvarların ve Köşelerin Çizdirilmesi

Gezgin robotun bilinmeyen ortamda gezdirilmesi zamanı elde edilen ölçümler ışığında ortamın haritası çıkarılmaktadır. Harita çıkarılırken dikkat edilen en önemli noktalardan biri elde edilmiş harita detaylarından yararlanarak ortamın sınırları olan duvarları ve köşeleri en gerçekçi haliyle çizebilmektir (Astapkovich, 2009). Duvarların ve harita sınırlarının çizdirilmesi için birçok yöntem mevcuttur. Literatürde En Küçük Kareler, İteratif Uç Nokta Uydurma, Hough Transform, Ardışık Kenar Takibi, Doğru Bağlanımı ve s. yöntemler kullanılmaktadır (Perdomo, 2009). Yapılan çalışmada elde edilen harita detayları, algılanan noktalardan En Küçük Kareler yöntemi ile doğrular oluşturularak ortamın haritası çıkarılmaya çalışılmıştır. Aşağıda bu yöntemin adımları verilmiştir:



Şekil 4.3 Noktanın doğruya uzaklığı ve en optimal doğru

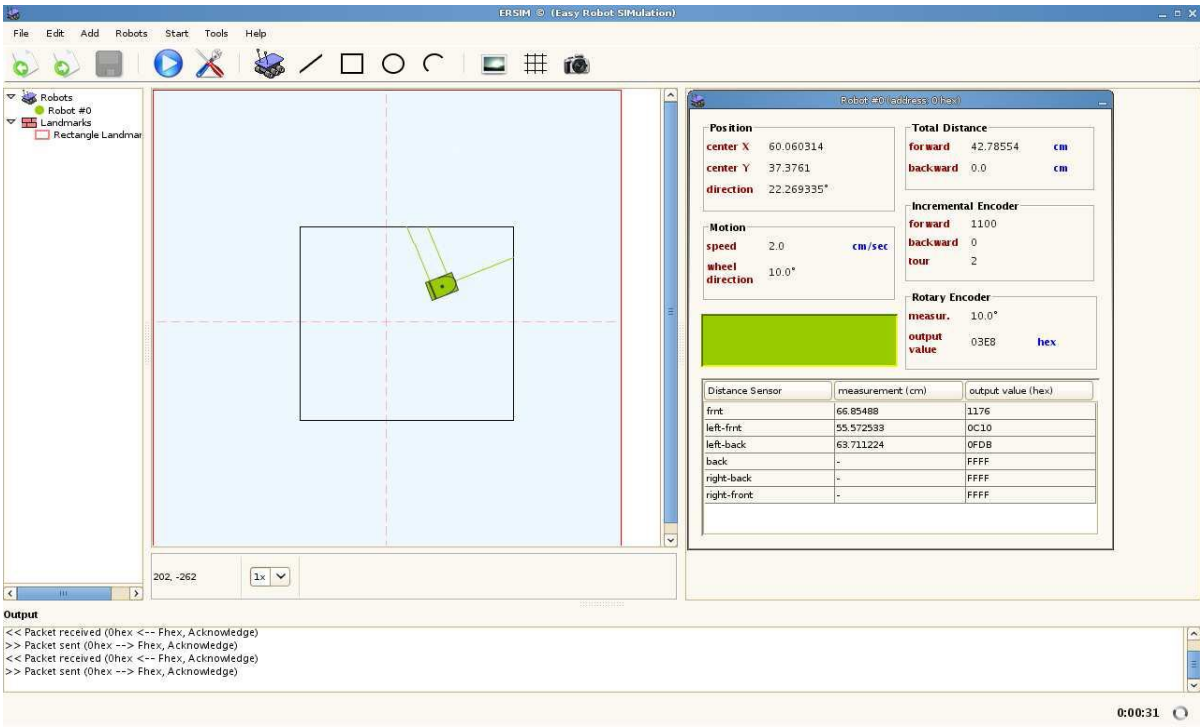
- (x_i, y_i) şeklinde verilmiş N sayıda nokta üzerinden geçebilecek en optimal doğruyu çizdirmek;
- $y = mx + c$ bu doğrunun grafiği olsun. (m, c) değerlerini bulmak için;
- $\frac{\partial E}{\partial m} = 0$ ve $\frac{\partial E}{\partial c} = 0$ kullanarak $E = \sum_i \frac{(y_i - mx_i - c)^2}{N}$ ortalama kare uzaklığı değerini minimize etmeli
- $\bar{y} = \frac{\sum_i y_i}{N}$ ve $\bar{x} = \frac{\sum_i x_i}{N}$ olarak kabul edilir
- O zaman $m = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$ ve $c = \bar{y} - m\bar{x}$ olacaktır.

4.6 .ERSIM Simülasyon Ortamı

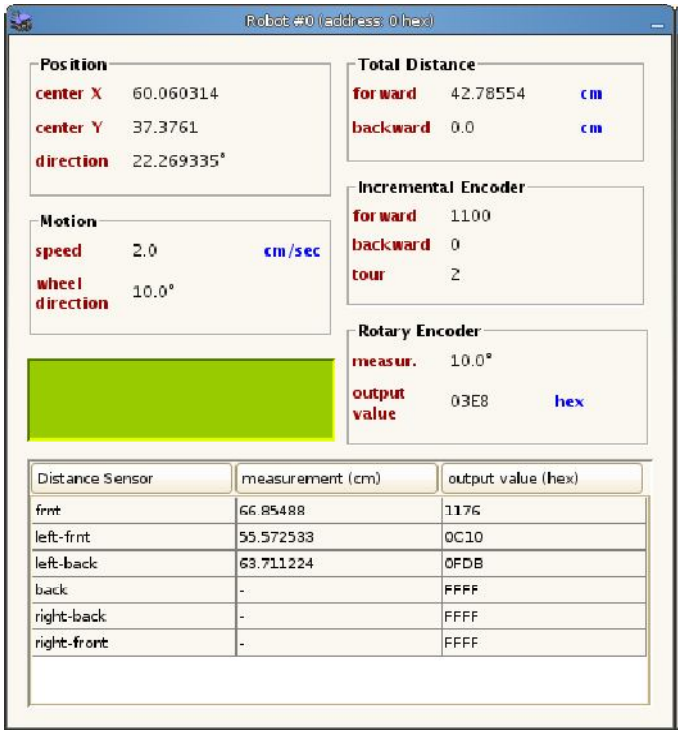
ERSIM simülasyon ortamı, NetBeans IDE 6.5 kullanılarak Java programlama dilinde geliştirilmiştir. Simülasyon, Yıldız Teknik Üniversitesi, Bilgisayar Mühendisliği bölümündeki Olasılıksal Robotik grubunun kullandığı robotu birebir modellemiştir. Aşağıdaki şekilde (Şekil 4.4) simüle edilen gerçekçi çizimi verilmiştir:

Şekil 4.4 YTÜ, Bilgisayar Mühendisliği, Olasılıksal robotik grubunun kullandığı robot modeli

İletişim, seri port üzerinden gerçekleştirilmiş ve simülasyon kontrolü bu şekilde sağlanmaktadır. Seri port erişiminde, GNU LGPL lisansı altında dağıtımı yapılan RXTX kütüphanesi kullanılmaktadır. Seri port erişimi, yapısı gereği işletim sistemi bağımlı bir işlem olduğu için RXTX kütüphanesi seri port erişimi için her işletim sisteminde, o sistem için özel hazırlanmış dinamik erişimli bir kütüphaneye çalışma anında ihtiyaç duymaktadır. Farklı dosya türlerinde olabilen (so, dll veya jnilib) bu kütüphane, uygulama ilk çalıştırıldığında kontrol edilmekte bulunamaması halinde ise işletim sistemine göre gerekli dosya JRE' nin kullanabileceği uygun konuma yerleştirilmektedir. Böylece JRE' nin yüklü olduğu ve seri portu bulunan herhangi bir bilgisayarda uygulamanın çalıştırılması mümkün olmaktadır (Deniz 2009). Simülasyonun mümkün olduğunca kolay kullanımlı ve anlaşılır bir yapıda olmasına gayret gösterilmiştir. Uygulama ara yüzü de buna paralel olarak tasarlanmıştır. Şekil 4.5 ve Şekil 4.6'de ERSIM simülasyon ortamının çalışma ekranı ve her hangi bir çalışma anında durumla ilgili bilgileri içeren ekran gösterilmiştir.



Şekil 4.5 ERSIM simülasyon ortamı ara yüzü görünümü



Şekil 4.6 ERSIM, Çalışma anında robot bilgileri gösterim penceresinin görünümü

4.6.1.Simülasyon ortamı, arayüzün haberleşmesi ve kontrolü

Robot simülasyonu yukarıda belirtilen robotik grubunun kullandığı robotu birebire temsil ettiği için grup tarafından tanımlanmış olan paket yapısı kullanılmaktadır. Haberleşme paketinin yapısı aşağıdaki gibidir (Kurt, 2007):

1) SYNCH (*Synchronization*): Eş zamanlama baytı olup, haberleşmenin başladığını belirtmektedir. Değeri, 55_{hex} (01010101_2) olarak tanımlanmıştır.
2) SFD (*Start Frame Delimiter*): Paket sınırlayıcısı başlangıç baytı olmakla verinin başladığını belirtmektedir. Değeri, $7E_{hex}$ (01111110_2) olarak tanımlanmıştır.
3) LENGTH: Kendisinden sonra gelen verinin uzunluğunu tanımlayan bayt olup Address, Control, Payload ve CRC baytlarının toplam uzunluğunu ifade etmektedir. Değerleri 0-255 arasında pozitif tam sayılardan oluşmaktadır.
4) ADDRESS: İletişim taraflarını belirleyen bayttır. Birinci dört bit gönderen tarafın, ikinci dört bit ise alıcı tarafın adresini tanımlamaktadır.
5) CONTROL: Kontrol verisini içeren bayt olup, paketin içeriğini tanımlamaktadır. Alabileceği değerler ve ifade ettikleri anlam Çizelge 4.1' de görüldüğü gibidir

Çizelge 4.1 Haberleşme protokolünde kullanılan kontrol baytları ve anlamları

Kontrol baytı	Değer	Açıklama
Acknowledge	00_{hex}	Son paket başarıyla alındı
Negative-Acknowledge	01_{hex}	Son paket alınırken zaman aşımı oldu
Collision	02_{hex}	Robot bir yere çarptı
Error	03_{hex}	Hatalı veri paketi
Hello	04_{hex}	Bağlantı kuruldu onayı (*Handshaking*)

6) PAYLOAD: Pakette asıl verinin bulunduğu bölüm olarak, paketin esnek bir yapı kazanmasını sağlamaktadır. Bu bölümde, robotta bulunan her aygıta verilmiş olan ve o aygıta has olarak tanımlanan bir baytlık adres değerleri kullanılmaktadır.

7) CRC (*Cyclic Redundancy Check*): Veri paketinin iletimi esnasında oluşabilecek hataların tespiti amacıyla kullanılan 2 baytlık hata tespit kodudur. Farklı boyutlarda olabilen ve değişik polinomlar kullanılarak hesaplanabilen CRC kodu için ITU-T tarafından V.41 tavsiye kararında tanımlanan ve CRC-16-CCITT ismiyle bilinen $x_{16} + x_{12} + x_5 + 1$ $(1021hex)$ polinomu kullanılmaktadır (Deniz, 2009). CRC kodu, haberleşme paketindeki Address, Control ve Payload baytları için ayrı-ayrı hesaplanmaktadır.

4.7 .FastSLAM Yöntemi İle Haritalama

FastSLAM algoritması parçacık filtresi ile GKF algoritmasının bileşkesinden oluşmuştur. Bu yöntemle haritalama zamanı, robot konumlarına ve harita durumuna bağlı olarak öncül ihtimalleri çarpım şeklinde gösteriyoruz. Bunu Parçacık filtrelerinin çarpımı gibi de düşünebiliriz. Bu çarpım, her bir parçacık için ayrı olasılık fonksiyonu kuruluyor ve bunların bileşkesi gibi gösteriliyor. Geri kalan bütün durum güncellemesi zamanı haritadaki detayları algılamak için GKF kullanılıyor. FastSLAM algoritmasını en önemli üstünlüğü, veri ilişkilendirme kararlarında, parçacıklar üzerinden işlem yapmasıdır (Montemerlo, 2003). Diğer bir üstün özelliği ise FastSLAM algoritmasında PF kullanıldığından ve bu filtreleme yönteminde hareket modelinin doğrusal olmayabilmesidir ki, bu özellik onu GKF' den de seviyece yukarıya taşımaktadır. Çünkü GKF işlemleri zamanı kullanılan yaklaşık modelden daha iyi sonuçlar vermektedir.

4.7.1.FastSLAM 1.0 algoritması

FASTSLAM algoritmaları posterior' leri robotun yolu ve ya konumu olan $\boldsymbol{p(s^t \mid z^t, u^t, n^t)}$ üzerinden hesaplıyor. (3.6) formülündeki ikinci çarpan olan $\boldsymbol{N}$ adet detay çarpanındaki detaylar ise GKF ile hesaplanıyor.

Çizelge 4.2 FASTSLAM' da parçacıklar

	Robot konumu	*1. detay*	*2. detay*	*…*	*N. detay*
1. Parçacık	$x\ y\ \theta$	μ_1, Σ_1	μ_2, Σ_2	…	μ_N, Σ_N
2. Parçacık	$x\ y\ \theta$	μ_1, Σ_1	μ_2, Σ_2	…	μ_N, Σ_N
… …	… …	… …	… …	… …	… …
M. Parçacık	$x\ y\ \theta$	μ_1, Σ_1	μ_2, Σ_2	…	μ_N, Σ_N

Yukarıdaki Çizelge 4.2' de M adet parçacıktan oluşan FASTSLAM görüyoruz. Her bir parçacık için (4.5) yazabiliriz.

$$S_t^{[m]} = \langle s^{t,[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \dots, \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]} \rangle \tag{4.5}$$

$[\boldsymbol{m}]$ - parçacık numarası, $\boldsymbol{s^{t,[m]}}$ yol durumu, $\boldsymbol{\mu_{n,t}^{[m]}, \Sigma_{n,t}^{[m]}}$ ise $\boldsymbol{n}$ numaralı detay için Gauss gösteriminin ortalama değeri ve kovaryansıdır. Bütün bu yazılımların toplamı ise $\boldsymbol{S_t^{[m]}}$ oluyor.

Filtreleme ise anındaki posterior'dan zamanındakini, parçacık kümesinden yeni kümesini oluşturarak hesaplamaktır. Bu işlemlerle senkronize olarak yeni kontrol ve ölçüm bilgisini (' e bağlı olarak) oluşturuluyor. Bütün bunlar aşağıdaki adımlarla gerçekleştiriliyor.

Sampling (Sadeleştirme – Yol, konum posterioru'u yeni konumları sadeleştirerek genişlendirme) :

(4.6)

Updating (Güncelleme - Elde olunmuş harita detaylarının güncellenmesi) : Bu adımda bulunan her detay için ortalama değer ve kovaryans değerlerinin güncelleştirilmesi yapılacaktır. Yani, güncelleme aşaması anındaki fark edilmiş gürültü harita detayı mı değimli sorusunun cevabını araştırıyoruz.

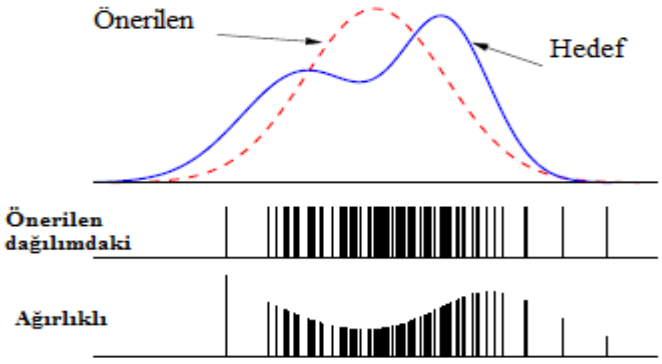
(4.7)

(4.7) atamasında durumu gösterilmiştir. Eğer ise o zaman

$$p(\theta_{n_t} \mid s^t, n^t, z^t) \;=\; \eta\, p(z_t \mid s_t, \theta_{n_t}, n_t)\; p(\theta_{n_t} \mid s^{t-1}, n^{t-1}, z^{t-1}) \qquad (4.8)$$

eşitliğinden yararlanarak GKF aşamalarını bir-bir gerçekleştirilmektedir.

Resampling: Birinci asım olan Sampling' de konumları ölçüm değeri dikkate alınmadan kontrol değerlerine göre oluşturuluyor. Böylelikle örtüşmezlikler meydana çıkıyor. Bu örtüşmezlikleri aradan kaldırmak için ise Resampling adımı gerçekleştirilmektedir. FASTSLAM' da önerilen dağılım ' den bağımsızdır. Ama hedef dağılım değil. Parçacıklara ağırlık değerleri atayarak ve sonra bu değerlere göre tekraren oluşturulan örnek parçacıklar aslında hedef dağılıma yakınsamış oluyor. Her bir parçacığın ağırlık değeri o parçacığın 'önemlilik faktörü' oluyor.



Şekil 4.7. Hedef ve önerilen dağılımlar ve ağırlıkların histogramlarla gösterilmesi (Thrun vd., 2005)

Şekil 4.7' de önerilen ve hedef dağılımlar grafiklerle, önerile ve ağırlık değerleri bulunmuş parçacıklar histogramlarla gösterilmiştir.

Resampling aşaması aslında hedef ve önerilen dağılımlar arasındaki fark ortaya çıkarmaktadır.

$$w_t^{[m]} = \frac{hedef\ dağılım}{önerilen\ dağılım} = \frac{p(s^{t,[m]} \mid z^t, u^t, n^t)}{p(s^{t,[m]} \mid z^{t-1}, u^t, n^{t-1})} = \eta\, p(z_t \mid s^{t,[m]}, z^{t-1}, n^t) \qquad (4.9)$$

w değerlerinin hesaplanması zamanı ise Taylor sırası kullanılarak doğrusallaştırma işlemi yapıldıktan sonra $\mathcal{N}(x;\, \mu, \Sigma)$ değerli Gauss dağılımlı yaklaşık değer, 2. aşama olan güncelleme adımında hesaplanıyor.

Bu üç aşamadan oluşan işlemlerin sonucunda bunu diyebiliriz ki, zaman ve hafıza gereksinimlerinin hiçbiri t anına bağlı değil.

4.7.2.FastSLAM 2.0 algoritması

2.0 algoritması genellik daha önceki FASTSLAM algoritmasına benziyor. Ama en önemli farklılığı, önerilen dağılım z_t göz önünde bulundurarak hesaplanmasıdır. Bu da FASTSLAM 1.0' daki sorunları aradan kaldırıyor. FASTSLAM 1.0' da Sampling aşamasında konumları belirlemek için u_t kontrol verisini, Resampling aşamasında ise z_t ölçüm verisini kullanıyor. Robot sensorlarının kontrol verilerine az bağlı olduğu durumlarda ise sorunlarla karşılaşılıyor. Önerilen dağılım fonksiyonunun oluşturduğu çok sayıda örneklerden yalnız çok az bir kısmı

yüksek benzerlik göstermektedir (Bailey vd., 2004, Perdomo, 2009). Ona göre de önerilen dağılımın içine sensor verilerini de eklersek daha verimli bir çalışma yapmış oluruz.

4.8 .Veri İlişkilendirme Problemi

Sensor ölçümleri yardımıyla bulunan harita özellikleri oluşturulan haritadaki detaylarla kıyaslandığı zaman farklılıklar meydana çıkmaktadır. Bu problem veri ilişkilendirme problemi olarak bilinmekte ve EZKBH meselesinin en karmaşık problemidir (Durrant vd., 2006).

Her iki FASTSLAM algoritmasının en büyük kısıtlaması bilinen veri ilişkilendirme üzerinden işlem yapmalarıdır. Gerçek dünyanın özellikleri belirsiz olduğundan bu bölümde n^t değişkenlerinin, kendi aralarında haberleşmesinin bilinmeyen olduğunu varsayarak anlatmaya çalışalım. Veri ilişkilendirmeni kısaca şöyle tanımlayabiliriz: t anında n_t değişkenini mümkün verilere dayanarak belirlemek. Veri ilişkilendirmenin en eski çözümü ise n^t'i sensor ölçümlerinin maksimum bezerlik gösterdiği için seçmektir. Yani

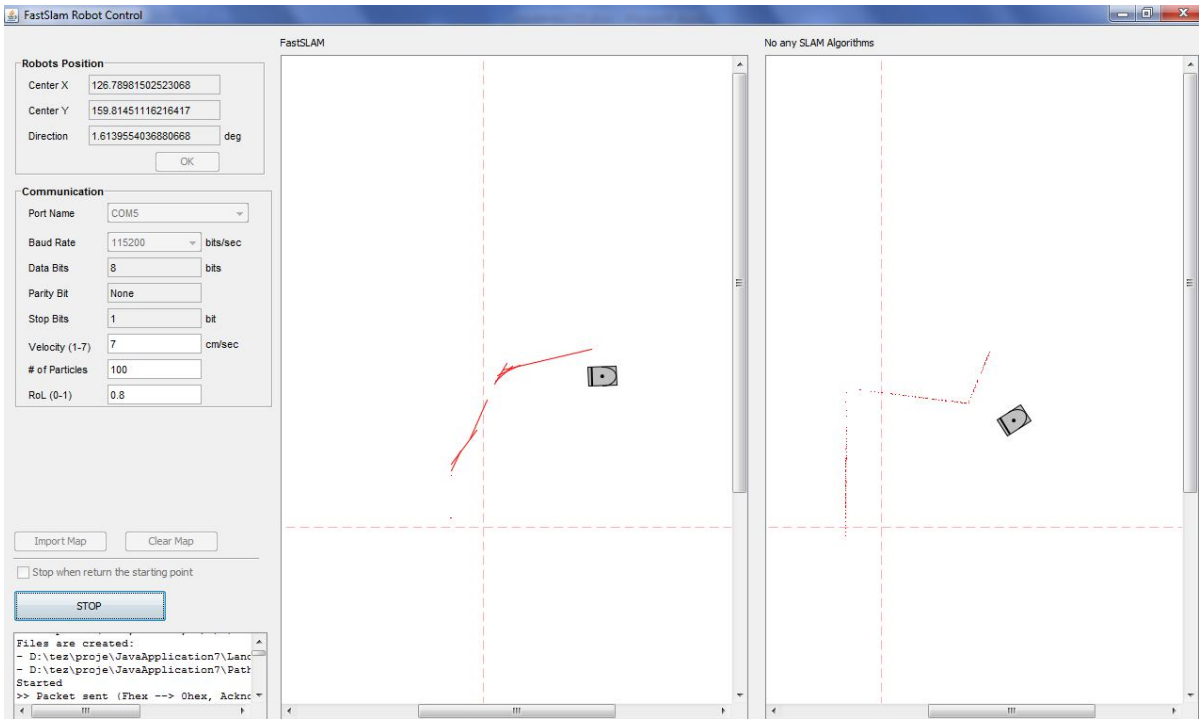
$$\hat{n}_t = \text{argmax}\ p(z_t \mid n_{t,},\ \hat{n}^{t-1}, s^t, z^{t-1}, u^{t-1}) \tag{4.9}$$

Böyle bir tahmin 'maksimum olabilirlik' adlandırılıyor.

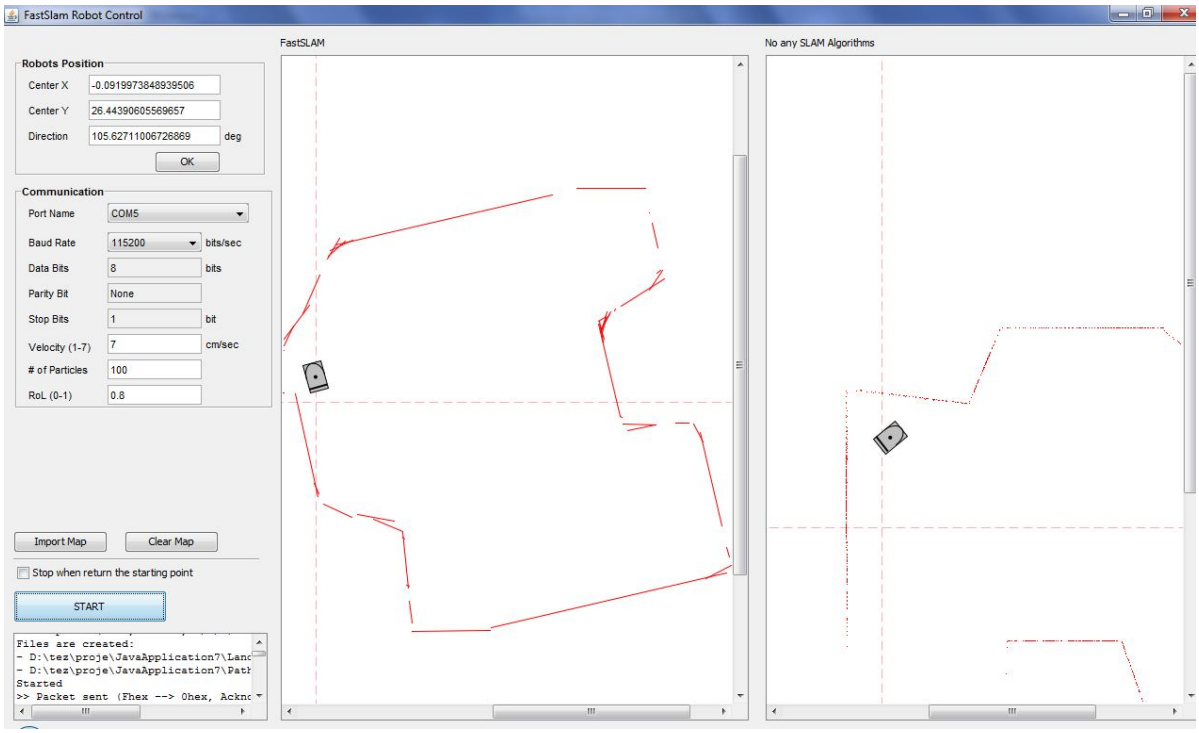
FASTSLAM algoritmalarında ise her bir parçacık için $\hat{n}_t^{[m]}$- veri ilişkilendirme değişkenleri kümesi tanımlanmaktadır. Sensor ölçüm dağılımı belirlenmiş bir eşik değer etrafında toplanmışsa bu kümeye yeni detay özelliği eklenerek küme doldurulmaktadır.

5.GEZGİN ROBOTLARLA EŞ ZAMANLI KONUM BELİRLEME VE HARİTALAMA

Yapılan çalışmada simule edilen robot ERSIM simülasyon ortamından verilen çeşitli haritalarda gezdirilmiştir. Doğrusal duvarlara sahip haritalarla beraber eğimli ve çember şekilli ortamlarda geliştirilen algoritmanın verimliliği kontrol edilmiştir. Gezdirme algoritması olarak duvar takip etme, özellik olarak doğrular En Küçük Kareler yöntemi ile çıkarılmıştır. Haritalar 1, 10, 50, 75, 100, 125 parçacık kullanılarak, algı, tekerlek, patinaj ve zemin tipine göre oluşturulmuştur. Robotun hızı genel olarak 7cm/sn olarak belirlense de bazı deneylerde 5cm/sn denenmiş ve sonuçlar kıyaslanmıştır. Şekil 5.1 ve Şekil 5.2' de simülasyondan çalışma esnasında alınmış resimleri gösterilmiştir.

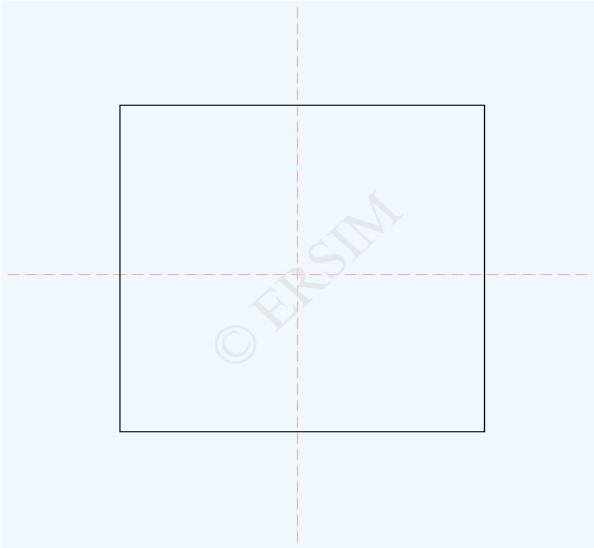


Şekil 5.1 Simülasyon arayüzünün haritalama zamanı görünümü (I)

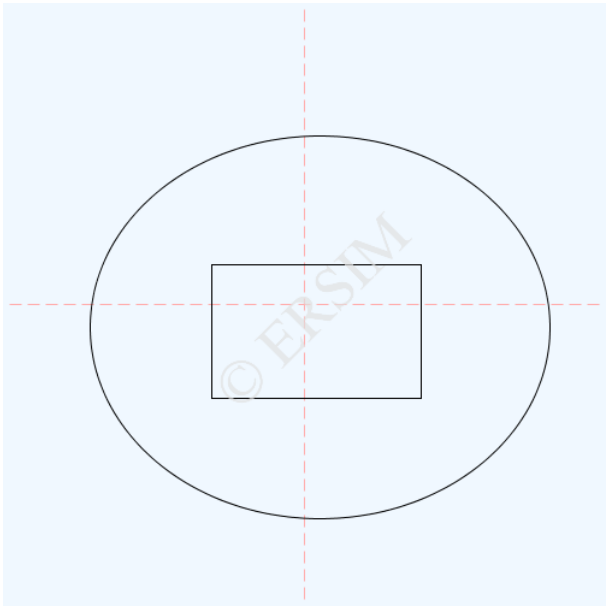


Şekil 5.2 Simülasyon arayüzünün haritalama zamanı görünümü (II)

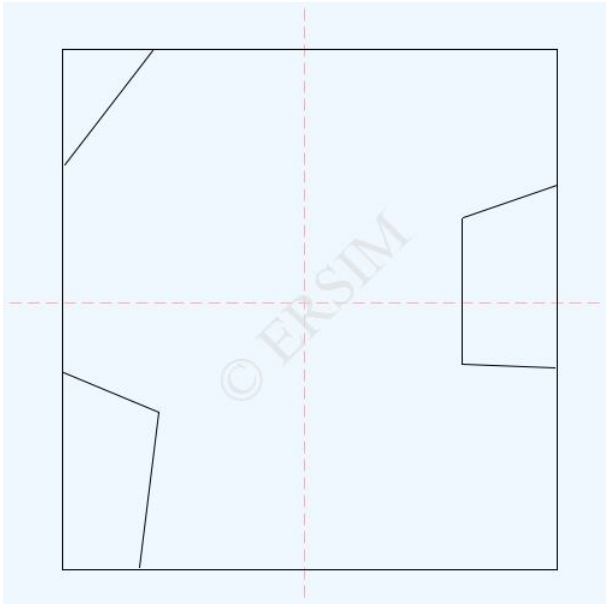
Duyarga olarak kızılötesi algılayıcılar kullanıldığından 80 cm maksimum ölçüm mesafesi belirlenmiştir. Belirlenen eşik değerin altına düştüğünde ise ölçüm yapabilme kabiliyetini kaybetmektedir. Harita detayı algılamada FastSLAM 2.0 algoritması kullanılmıştır. Bu algoritma Parçacık tabanlı filtreleme yaptığından parçacıkların sayı arttıkça çıkarılan harita kalitesinin de arttığı belirlenmiştir. Aşağıda, Şekil 5.3, Şekil 5.4 ve Şekil 5.5' de hedeflenen ortam haritaları verilmiştir:



Şekil 5.3 I Ortam için hedeflenen harita görünümü



Şekil 5.4 II Ortam için hedeflenen harita görünümü



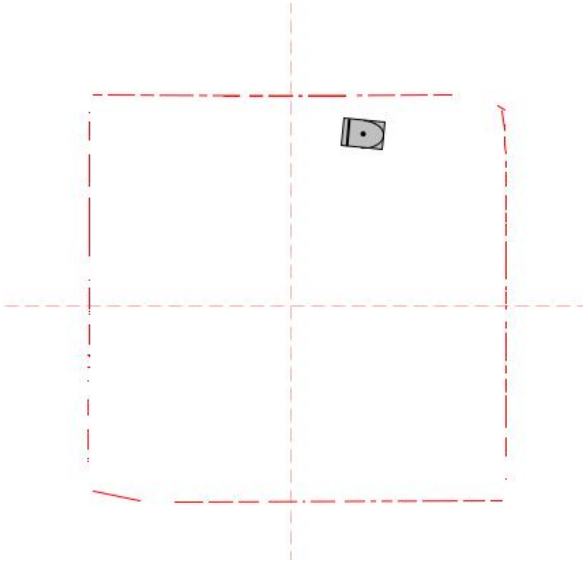
Şekil 5.5 III Ortam için hedeflenen harita görünümü

5.1 .Deneysel Sonuçlar

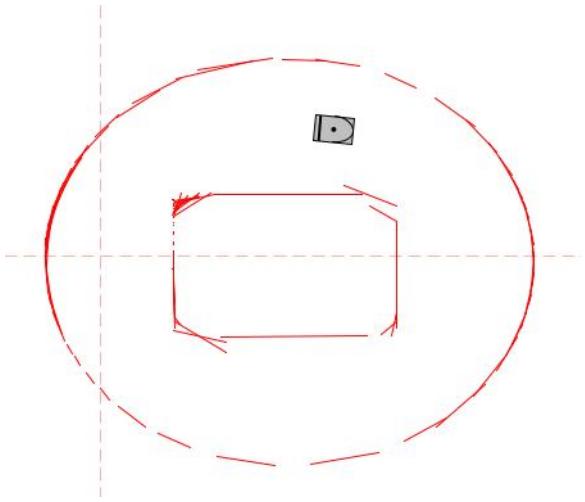
Aşağıdaki Çizelge 5.1' de kullanılan EZKBH yöntemi ile oluşturulan haritalar ve sürece etki eden faktörler belirtilmiştir. Çok sayıda yapılmış deneylerde Algılayıcı, Tekerlek, Döner Kodlayıcı (Patinaj) ve Zemin tipi gibi gürültülerden biri ve ya bazıları kullanılarak sonuçlar incelenmiştir. Genel olarak, simülasyonun çalışma süresi ise robotun bir tur attıktan sonra başlangıç konumuna en yakın pozisyonda bulunması süresi ele alınmıştır. Deneylerde, robotun haritayı bir tur gezindikten sonra algoritmadan alınan predict, yani tahmini konum bilgilerinin gerçek konuma yaklaştırmaya çalışması sonucunda başlangıç konumuna geldiğini anlamaktadır. Yukarıda adları belirtilen özellik belirleyicilerin Gauss (μ, σ) gürültüsünün değişik ortalama değer ve sapması eklendiğinde alınan sonuçları incelenmiştir.

Çizelge 5.1 FastSLAM'la elde edilen sonuçlar

Deneme Sırası	Ortam Numarası	Parçacık Sayı	Duyarga Gürültüsü (cm)	Tekerlek Gürültüsü (°)	Döner Kodlayıcı Gürültüsü (°)	Geçen zaman (saniye)	Elde Edilen Harita
1	I	1	-	-	-	163.26	Şekil 5.6
2	II	1	-	-	-	160.51	Şekil 5.7
3	II	1	Gauss (0,2)	-	-	154.59	Şekil 5.8
4	III	1	Gauss (0,2)	-	-	219.75	Şekil 5.9
5	I	10	-	-	-	166.34	Şekil 5.10
6	II	10	-	-	-	163.80	Şekil 5.11
7	I	100	-	-	-	166.7	Şekil 5.12
8	I	100	-	-	Gauss (0,2)	174.14	Şekil 5.13
9	I	100	-	Gauss (0,2)	-	159.56	Şekil 5.14
10	I	100	-	-	Gauss (0,4)	176.73	Şekil 5.15
11	I	100	Gauss (0,4)	-	-	152.51	Şekil 5.16
12	I	100	-	Gauss (0,4)	-	159.11	Şekil 5.17
13	II	100	-	-	-	172.57	Şekil 5.18
14	II	100	-	-	-	225.13	Şekil 5.19
15	II	100	Gauss (0,2)	-	-	161.05	Şekil 5.20
16	II	100	Gauss (0,4)	-	-	163.26	Şekil 5.21
17	II	100	Gauss (0,2)	Gauss (0,2)	-	212.01	Şekil 5.22
18	III	100	-	-	-	221.22	Şekil 5.23
19	III	100	Gauss (0,2)	-	-	223.1	Şekil 5.24
20	III	100	Gauss (0,4)	-	-	223.51	Şekil 5.25
21	III	100	Gauss (0,2)	Gauss (0,2)	Gauss (0,2)	225.14	Şekil 5.26
22	I	50	-	-	-	160.07	Şekil 5.27
23	I	50	Gauss (0,4)	-	-	160.72	Şekil 5.28
24	I	50	-	Gauss (0,2)	-	160.94	Şekil 5.29

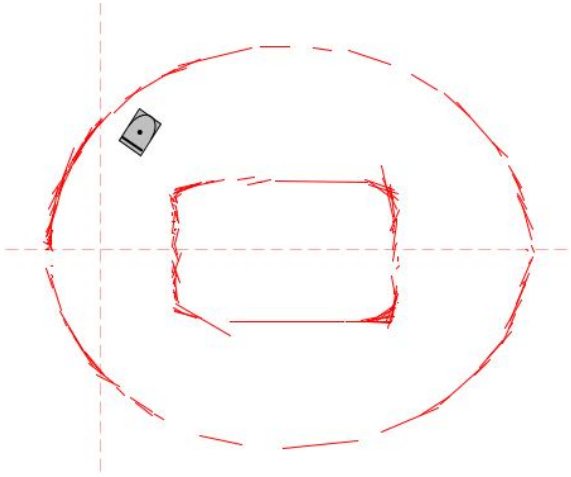


Şekil 5.6 FastSLAM yöntemi ile 1. denemede çıkarılan harita

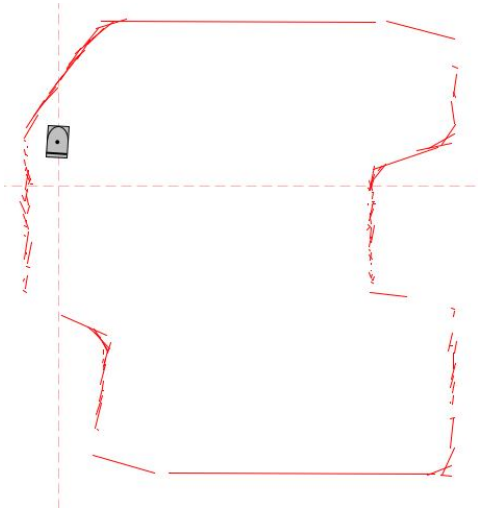


Şekil 5.7 FastSLAM yöntemi ile 2. denemede çıkarılan harita

Şekil 5.6 ve Şekil 5.7'de çıkarılan haritadan gördüğümüz gibi, parçacık sayısı haritanın kalitesine direk etki ettiğinden ve işlemler sadece bir parçacık için hesaplandığından haritada tam ve bütün olması gerek duvarlar kesik-kesik çıkarılmıştır.

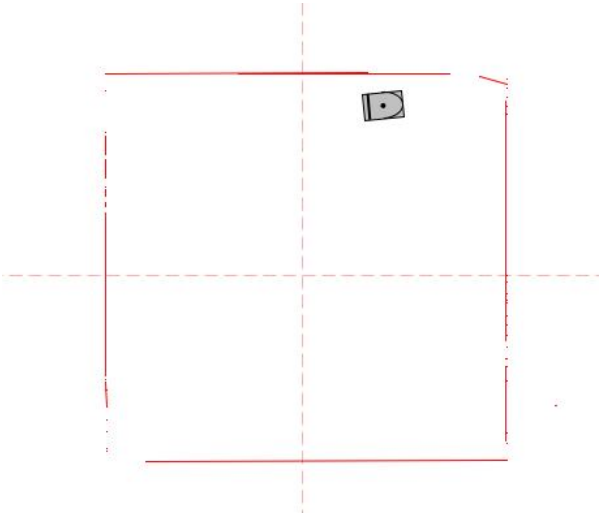


Şekil 5.8 FastSLAM yöntemi ile 3. denemede çıkarılan harita

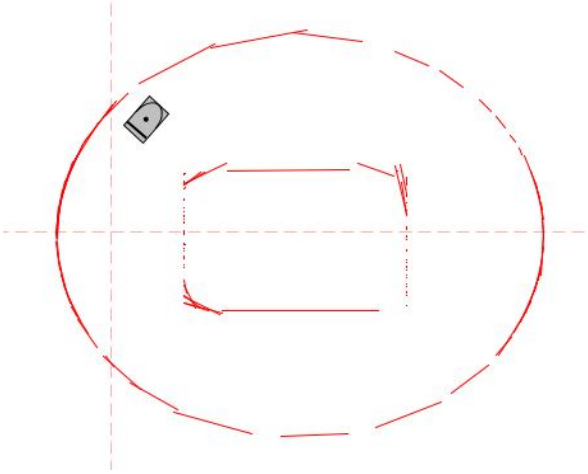


Şekil 5.9 FastSLAM yöntemi ile 4. denemede çıkarılan harita

Şekil 5.8 ve Şekil 5.9' de daha önceki haritalarda olduğu gibi tek bir parçacık etkisi haritalara yansımıştır. Yani elde edilen haritalar ortam gürültülerini düzeltememektedir.

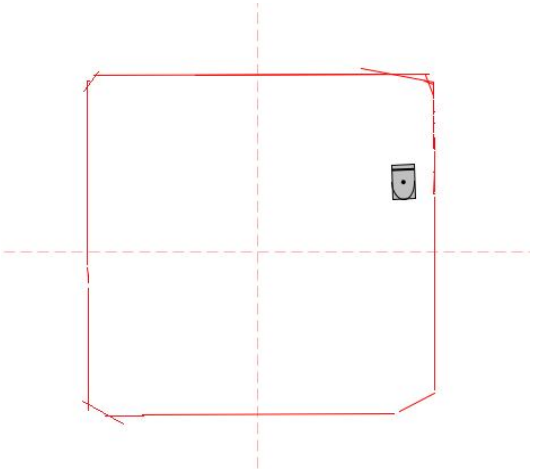


Şekil 5.10 FastSLAM yöntemi ile 5. denemede çıkarılan harita

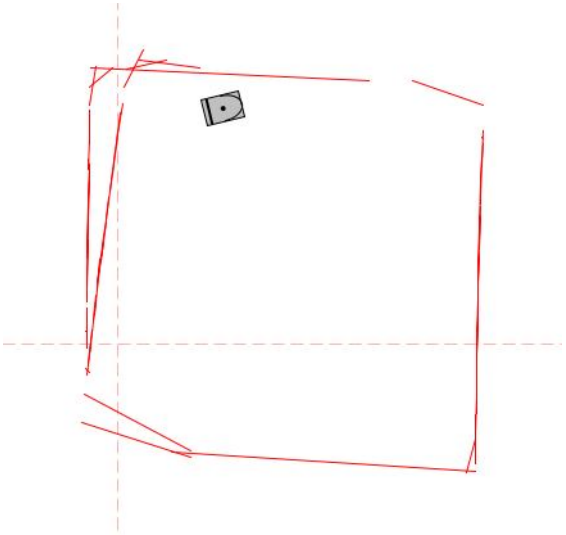


Şekil 5.11 FastSLAM yöntemi ile 6. denemede çıkarılan harita

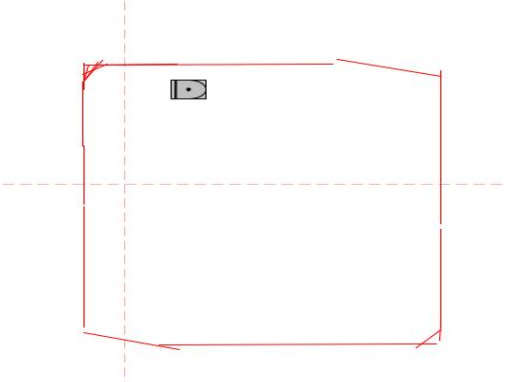
Yukarıdaki Şekil 5.10 ve Şekil 5.11 de ise parçacık sayısının artması ile çıkarılan haritanın da değiştiğini görüyoruz.



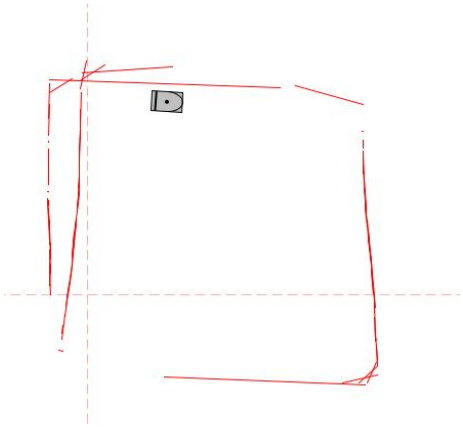
Şekil 5.12 FastSLAM yöntemi ile 7. denemede çıkarılan harita



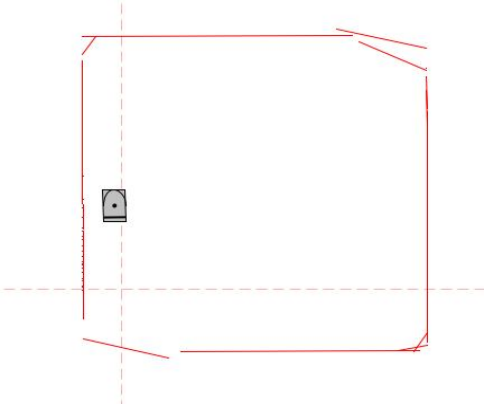
Şekil 5.13 FastSLAM yöntemi ile 8. denemede çıkarılan harita



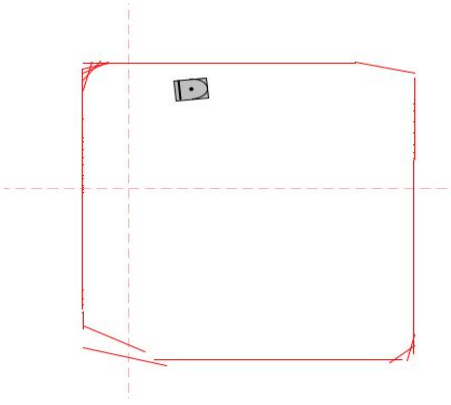
Şekil 5.14 FastSLAM yöntemi ile 9. denemede çıkarılan harita



Şekil 5.15 FastSLAM yöntemi ile 10. denemede çıkarılan harita

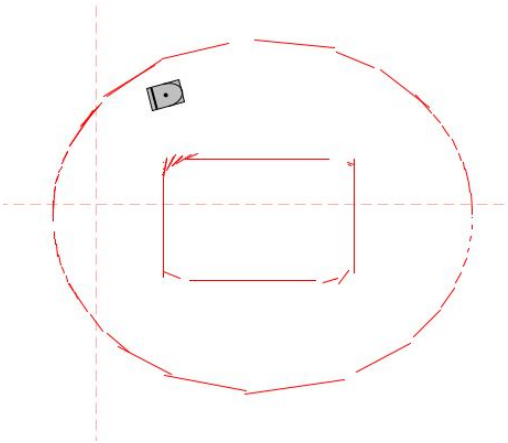


Şekil 5.16 FastSLAM yöntemi ile 11. denemede çıkarılan harita

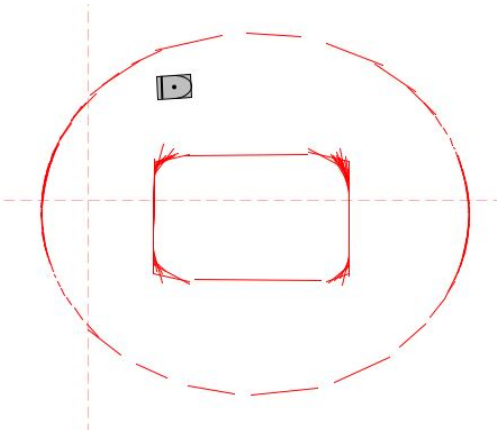


Şekil 5.17 FastSLAM yöntemi ile 12. denemede çıkarılan harita

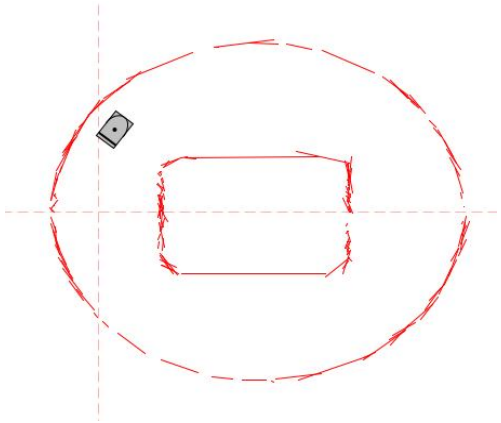
Yukarıda Şekil 5.12’ den Şekil 5.17’ e kadar çıkarılan I Ortam haritalarına dikkat edersek genellikle daha önceki haritalardan farklı olduklarını seze biliriz. Lakin Şekil 5.13 ve Şekil 5.15’ de döner kodlayıcı gürültüsü haritanın kalitesini bozmuştur. FastSLAM algoritması sensor ve hareket gürültüsünü minimuma indirse de döner kodlayıcı gürültüsünü düzeltememesi görülmüştür.



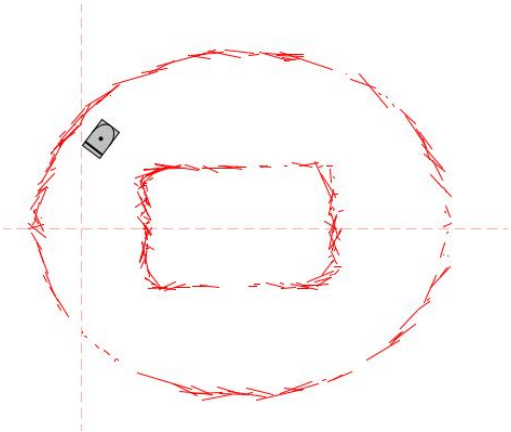
Şekil 5.18 FastSLAM yöntemi ile 13. denemede çıkarılan harita



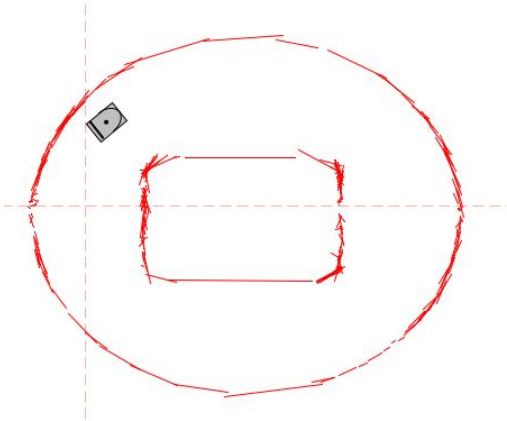
Şekil 5.19 FastSLAM yöntemi ile 14. denemede çıkarılan harita



Şekil 5.20 FastSLAM yöntemi ile 15. denemede çıkarılan harita

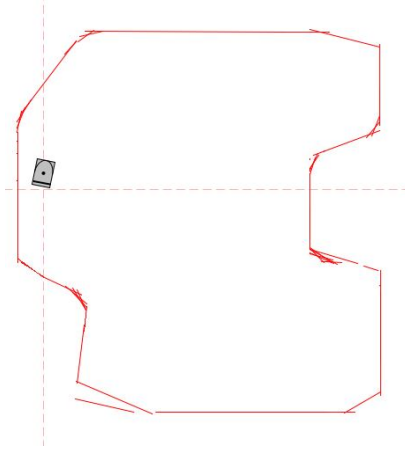


Şekil 5.21 FastSLAM yöntemi ile 16. denemede çıkarılan harita

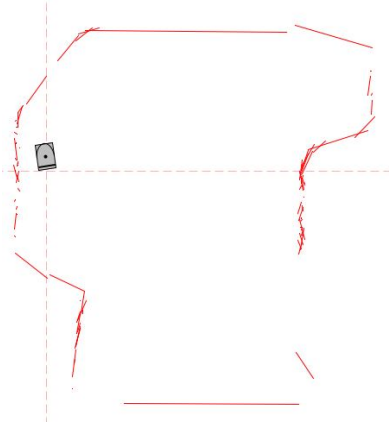


Şekil 5.22 FastSLAM yöntemi ile 17. denemede çıkarılan harita

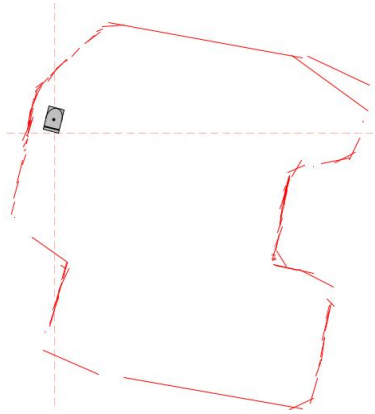
Şekil 5.20, 5.21 ve Şekil 5.22' den de görüldüğü gibi sensor gürültüsünün varlığında karesel olmaya yani yuvarlak ortamların haritalanma süresi biraz uzamış ve sınırların çizdirilme kalitesi düşmüştür.



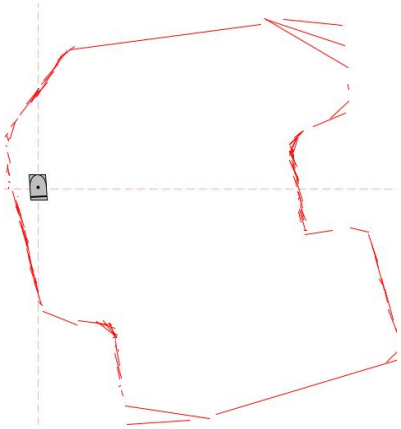
Şekil 5.23 FastSLAM yöntemi ile 18. denemede çıkarılan harita



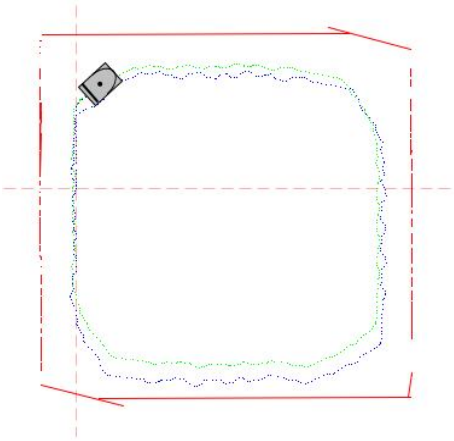
Şekil 5.24 FastSLAM yöntemi ile 19. denemede çıkarılan harita



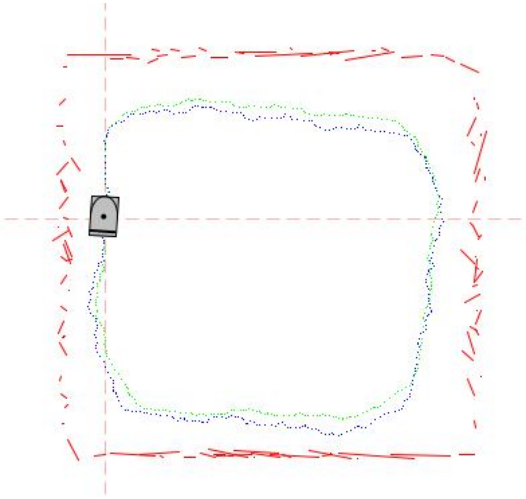
Şekil 5.25 FastSLAM yöntemi ile 20. denemede çıkarılan harita



Şekil 5.26 FastSLAM yöntemi ile 21. denemede çıkarılan harita

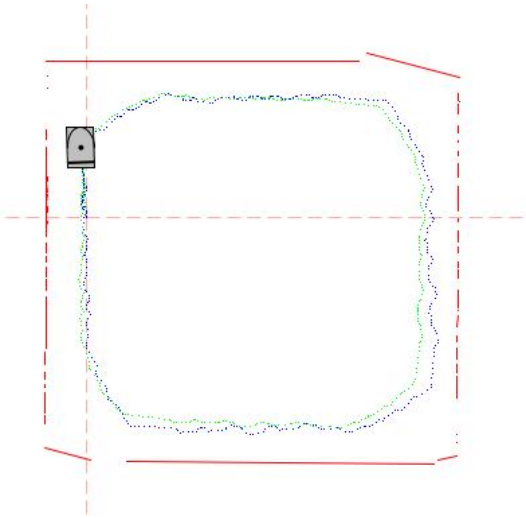


Şekil 5.27 FastSLAM yöntemi ile 22. denemede çıkarılan harita



Şekil 5.28 FastSLAM yöntemi ile 23. denemede çıkarılan harita

Yukarıdaki şekilde sensor gürültüsü artırılarak Gauss (0,4) yapılmış ama parçacık sayısı elliye düşürülmüştür. Bunun sonucunda ise elde edilen haritada duvarların doğruluğu bozulmuştur. Ama robotun takip ettiği yoldan ve kendini zannettiği konumdan da gördüğümüz gibi FastSLAM algoritması robotun asıl konumunu iyi tahmin etmekte ve robotu o konuma maksimum yaklaştırmaya çalışmaktadır.

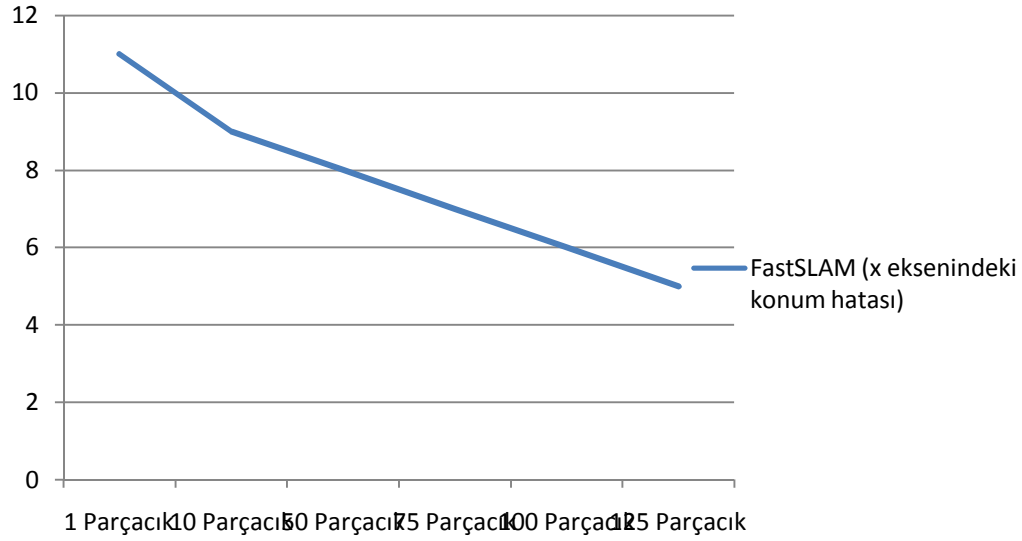


Şekil 5.29 FastSLAM yöntemi ile 24. denemede çıkarılan harita

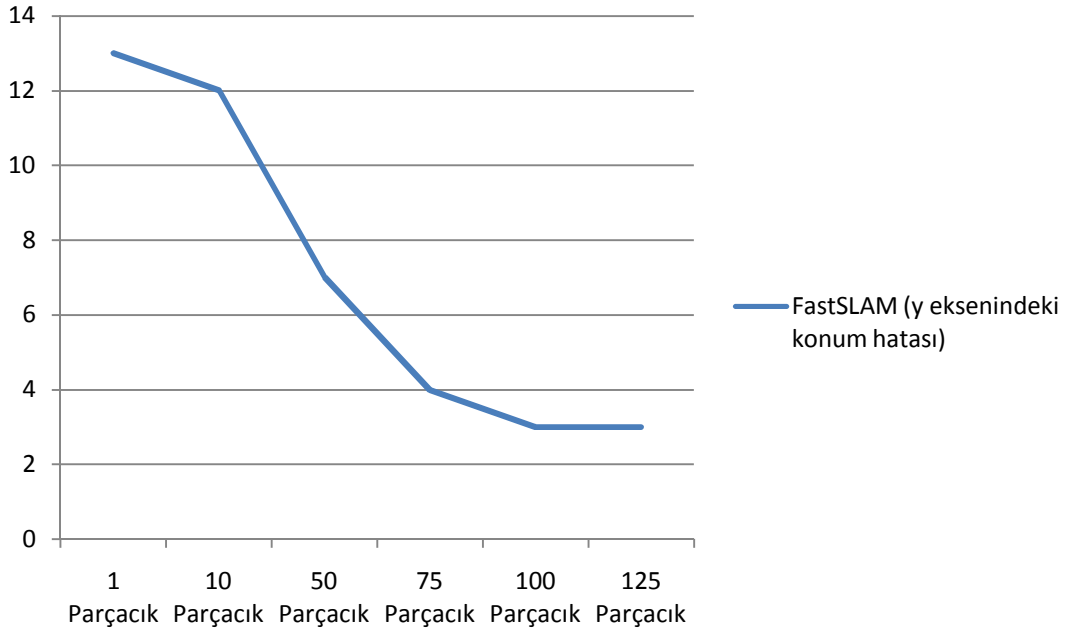
5.2 .Sonuçlar

Haritalanmaya çalışılan üç ortamda da göründüğü gibi harita detaylarının doğrulara dönüştürülmesi zamanı genel bir problemden bahsedebiliriz. Köşeleri istenilen seviyede çıkarılamaması büyük bir eksiktir. Parçacık sayında artış olduğu durumlarda işlemler yavaşlasa da elde edilen sonuçlar daha az parçacık kullanılarak elde edilen sonuçlardan daha verimlidir. Çoklu sayıda denemeden sonra kullanılan yöntemdeki parçacık sayısının simülasyonun gerçekleştirildiği bilgisayardaki en uygun sonuçları çıkarabilecek değeri 100 olarak belirlenmiştir. Ama Çizelge 5.1' de belirtilmemesine rağmen 200, 1000, 10000 parçacık için algoritma denemiş lakin bilgisayar hafızası yetmediğinden işlemlerde sonuca ulaşılamamıştır. Daha yüksek seviyeli sistemlerde daha iyi sonuçlar alınabilir. Bir başka tespit Robotun hızı ile ilgilidir ki, bu sisteme göre hız optimum tutulması lazım. Yüksek hızlarda alınan ölçümlerin kalitesinde genel bir düşüş görülmektedir. Robotun çok yavaş hızla hareket ettiği durumlarda ise hem sistemin genel yavaşlamasına sebep olmaktadır hem de ölçümlerin fazlalığı işlem karmaşıklığını doğurur. Mesela, 14. denemede, genelde denemelerin yapıldığı 7 cm/san değil, 5 cm/san doğrusal hızlı robot seçilmiştir. Buna uygun olarak simülasyonun çalışma süresi, yani robotun bir tam tur yapması gözle görülecek derecede gecikmiştir. Fakat oluşturulan harita kalitesi yükselmiştir. Ayrıca gerçekleştirilen simülasyonun algılayıcı ve hareket gürültüsüne de dayanıklı olduğu açıkça görülmektedir. Bunu aşağıdaki grafikte görebiliriz. Grafiği oluşturmak için parçacık sayısı, gerçek ve algılanan konum farklarının yaklaşık değerleri veri olarak kullanılmıştır. Grafiklerde Y ekseninde konum hatası cm olarak, X ekseninde ise parçacık sayıları verilmiştir.

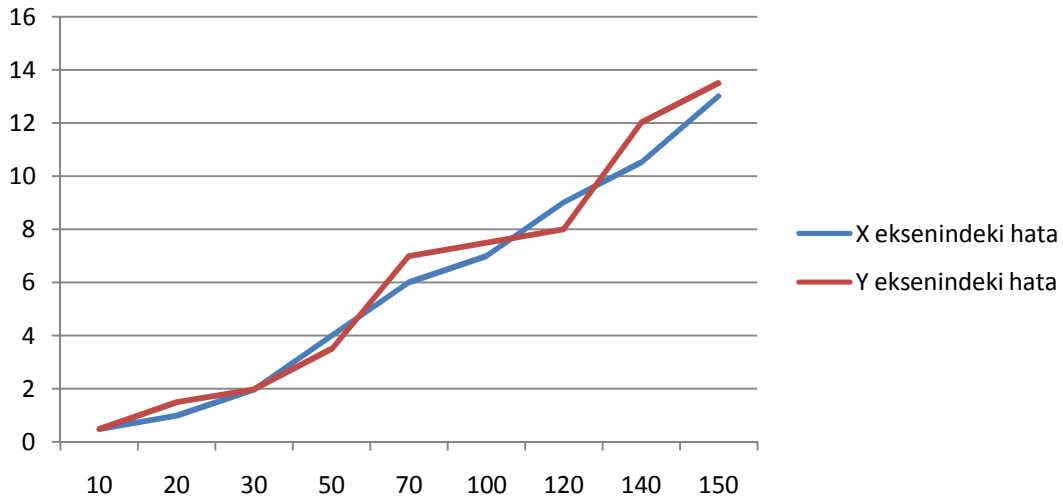
Çizelge 5.2 FastSLAM algoritması ile I Ortamın gezinmesi zamanı X eksenindeki konum hatası (FastSLAM)



Çizelge 5.3 FastSLAM algoritması ile I Ortamın gezinmesi zamanı Y eksenindeki konum hatası (FastSLAM)

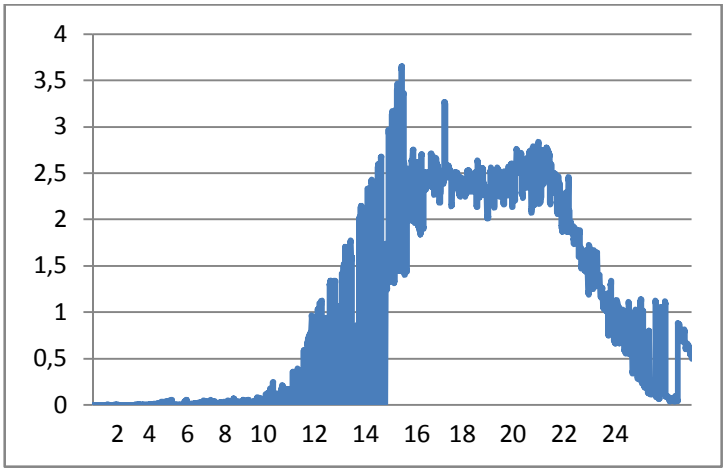


Çizelge 5.4 FastSLAM kullanılmadan X ve Y eksenlerindeki konum

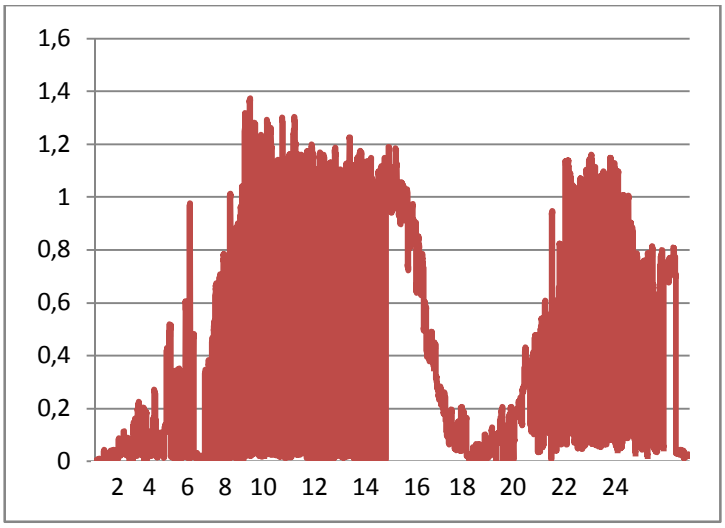


Ayrıca bütün bu denemelerden sonra I Ortamın 2 tur gezinmesi ile konum hatası gözden geçirilmiştir. Konum hatası dediğimizde (x,y,θ) verileri teker-teker incelenmiştir. Grafikleri oluşturduğumuzda yatay eksende robotun gittiği yol metre olarak, dikey eksende ise robotun kendini zannettiği ve gerçek konumları arasındaki farkların mutlak değeri cm olarak verilmiştir. θ açısı ise robotun incelenen andaki yönünü gösterdiğinden dikey eksen olan Y ekseninde gerçek ve tahmin edilen yön farkları radyan olarak verilmiştir.

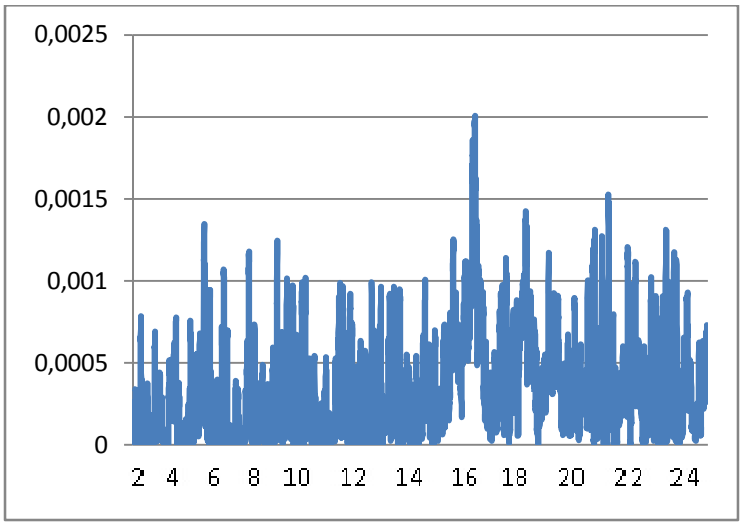
Çizelge 5.5 FastSLAM kullanarak I Ortam'ın iki tur gezdirilmesi zamanı X eksenindeki konum hatası



Çizelge 5.6 FastSLAM kullanarak I Ortam'ın iki tur gezdirilmesi zamanı Y eksenindeki konum hatası



Çizelge 5.7 FastSLAM kullanarak I Ortam'ın iki tur gezdirilmesi zamanı θ eksenindeki konum hatası



Yukarıdaki grafiklerden de gördüğümüz gibi konum hatasının mutlak değeri bir aralıkta tutulmaya çalışılmıştır.

6.SONUÇ

Çalışmada robotla her hangi bir ortamın haritasının çıkarılması amaçlanmış, FastSLAM yöntemi ile Eş Zamanlı Konum Belirleme ve Haritalama yapılmıştır. Literatürdeki çalışmaların hemen-hemen hepsinde FastSLAM algoritmaları pahalı lazer algılayıcılar beraberinde kullanılır. Çünkü aynı anda 500-1000 ölçüm yapabilen lazer algılayıcılar robotun kendi konumunu gerçek konumuna daha da yaklaştırarak, ortam haritasını maksimum düzeyde çıkarır. Bu çalışmada ise daha önceki bölümlerde de belirtildiği gibi altı adet kızılötesi algılayıcı kullanılmıştır. Simülasyondan çıkan genel sonuçlardan biri de bu algılayıcıların robot üzerindeki konumunda ve sayısında değişiklik etmekle daha verimli sonuçlara varılabilir. Ayrıca, EZKBH en büyük sorunu olan veri ilişkilendirme problemine de uygulamada çözüm aranmıştır.

Haritalanması amaçlanan ortamların şekli sadece bir-birine dik yerleşen duvarları değil, çember görünümlü ve eğik sınırları da kapsamaktadır. Yuvarlak sınırları olan ortamlarda algı ölçümlerinin ve kullanılan sınır çizdirme yöntemlerini optimal tutarak istenilen sonuca ulaşılabilir. Uygulamada En Küçük Kareler yöntemi ile doğrular çıkarılmıştır. Gelecek çalışma olarak bu yöntemin daha da geliştirilmesi ve ya başka daha verimli yöntemlerin kullanılmasını gösterebiliriz. Ayrıca bir başka gelecek çalışmalara örnek olarak robotun ortamda gezinmesi için duvar takip etme yöntemi ile beraber başka metotlar kullanılabilir ki bu da mesela, ortamın tamamının gezinmesi sonucunda daha gerçekçi harita çıkarılabilir. Bundan başka haritalama problemi için robot takimi da simule edilebilir ki, bu da gezinme süresini ve harita kalitesini artıracaktır.

KAYNAKLAR

Astapkovich A.M. (2009), Virtual Mobile Robot SOFA-2009, Saint-Petersburg University of Aerocosmics, Saint-Petersburg.

Bailey T., Nieto J. ve Nebot E. (2004), ‘‘Consistency of the FastSLAM Algorithm’’, Australian Centre for Field Robotics, University of Sydney, NSW.

Chatila, R. ve Laumond, J.-P (1985), “Position Referencing and Consistent World Modeling for Mobile Robots”, In Proceedings of the IEEE International Conference on Robotics and Automation, 2:138-145.

Crisan D. ve Doucet A. (2002), ‘‘A survey of convergence results on particle filtering methods for practitioners’’, IEEE Transactions on Signal Processing, 50(3):736–746.

Dempster, A.P., Laird, A.N. ve Rubin, D.B. (1977), “Maximum likelihood from incomplete data via the EM algorithm”, Journal of the Royal Statistical Society, Series B, 39(1):1–38.

Deniz E. (2009), ‘‘Çoklu Robot Simülasyonu ve Kumanda Ortamı’’, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Bilgisayar Mühendisliği, İstanbul.

Durrant W.H. ve Bailey, T. (2006), “Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms”, Robotics and Automation Magazine, 13:99-110.

Hahnel D. , Burgard W. , Fox D. ve Thrun S. (2003) , ‘‘An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements’’, IEEE/RSJ International Conference on Intelligent Robots and Systems, 206–211.

Huang, W. H. ve Beevers, K. R. (2006), “Topological mapping with sensing-limited robots”, Algorithmic Foundations of Robotics VI, 17:235-250.

Julier S.J. ve Uhlmann J.K. (2001) ‘‘A counter example to the theory of simultaneous localization and map building’’, IEEE International Conference on Robotics and Automation, 4238–4243.

Kalman, R. E. (1960), “A New Approach to Linear Filtering and Prediction Problems”, Trans. ASME, Journal of Basic Engineering, 82:35–45.

Karpov V. E. ve Platonova M. V. (2001), ‘‘The Navigation System Of Mobile Robot’’, MEI TU press, Moskova,

Kuipers, B. ve Byun, Y.-T. (1991), “A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations”, Journal of Robotics and Autonomous Systems, 8:47–63.

Kurt, Z. (2007), “Es Zamanlı Konum Belirleme ve Haritalamaya Yönelik Akıllı Algoritmaların Gelistirilmesi”, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, İstanbul.

Manseur, R. (2006), Robot Modeling & Kinematics, Charles River Media, Boston.

Matari´c, M. J. (1990), A Distributed Model for Mobile Robot Environment-learning and Navigation. Master's thesis, MIT.

Montemerlo M. (2003) ‘‘FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association’’, Doktora tezi, Carnegie Mellon University.

Perdomo E.F., (2009), Test and Evaluation of the FastSLAM Algorithm in a Mobile Robot, MS's thesis, University of Las Palmas de Gran Canaria, University Institute of Sistemas Inteligentes y Aplicaciones Numericas en Ingeniera (SIANI).

Siegwart, R. ve Nourbakhsh, I.R. (2004), Introduction to Autonomous Mobile Robots, The MIT Pres, Cambridge, Massachusetts.

Thrun, S. , Burgard, W. ve Fox, D. (2005), Probabilistic Robotics, The MIT Press, Cambridge, Massachusetts.

Thrun, S., Bücken, A., Burgard, W., Fox, D., Fröhlinghaus, T., Hennig, D., Hofmann, T., Krell, M. and Schmidt, T. (2001), "Map Learning and HighSpeed Navigation in RHINO", Research Reports on Information Science and Electrical Engineering of Kyushu University, 6(1):77-82.

Thrun, S., Fox, D. ve Burgard, W. (1998), "A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots", Machine Learning, 31:29–53. Also appeared in Autonomous Robots 5:253–271.

Wang C.C., (2004), ‘‘Simultaneous Localization and Mapping and Moving Object Tracking’’, PhD thesis, Robotics Institute, Carnegie Mellon University.

EKLER

Ek 1 FastSLAM Algoritmaları ve açıklamalar

Ek 2 FastSLAM Algoritmasının uygulaması

Ek-1 FastSLAM Algoritmaları ve Açıklamalar

FastSLAM algoritması 1.0 (z_t, u_t, S_{t-1}):

1. For $m = 1$ to M do

   S_{t-1} 'den $\langle s_{t-1}^{[m]},\ N_{t-1}^{[m]},\ \langle \mu_{1,t-1}^{[m]}, \Sigma_{1,t-1}^{[m]}, i_1^{[m]} \rangle, \dots, \langle \mu_{N_{t-1}^{[m]},t-1}^{[m]}, \Sigma_{N_{t-1}^{[m]},t-1}^{[m]}, i_{N_{t-1}^{[m]}}^{[m]} \rangle \rangle$ oku

   Bütün parçacıkları oluşturulması, işaretlenmesi

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

   Yeni robot konumunun oluşturulması

3. For n = 1 to $N_{t-1}^{[m]}$ do

   Sensor ölçüm benzerliklerinin hesaplanması

4. $\hat{z}_n = g(\mu_{n,t-1}^{[m]}, s_t^{[m]})$

   Ölçüm tahmini hesaplanması

5. $G_n = g'(s_t^{[m]}, \mu_{n,t-1}^{[m]})$

   Jakobyen hesaplanması

6. $Q_n = G_n^T \Sigma_{n,t-1}^{[m]} G_n + R_t$

   Sensor ölçüm kovaryansının hesaplanması

7. $w_n = |2\pi Q_n|^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(z_t - \hat{z}_n)^T Q_n^{-1}(z_t - \hat{z}_n)\right\}$

   Haberleşmenin benzerliğinin bulunması

   Endfor

8. $w_{N_{t-1}^{[m]}+1} = p_0$

   Yeni harita detayının önemlilik derecesinin hesaplanması

9. $\hat{n} = \underbrace{argmax}_{n=1,\dots,N_{t-1}^{[m]}+1} w_n$

   Max benzerlik haberleşmesi

10. $N_t^{[m]} = max\left\{N_{t-1}^{[m]},\ \hat{n}\right\}$

    Haritadaki detayların yeni sayı

11. For $n = 0$ to $N_t^{[m]}$ do

    Kalman filtresinin güncellenmesi

12. İf $n = N_{t-1}^{[m]} + 1$ then

    Yeni harita detayı mı?

13. $\mu_{n,t}^{[m]} = g^{-1}(z_t, s_t^{[m]})$

Ortalama değerin alınması

14. $\Sigma_{n,t}^{[m]} = G_{\hat{n}}^{-1} R_t (G_{\hat{n}}^{-1})^T$

Kovaryansın alınması

15. $i_{n,t}^{[m]} = 1$

Sayacın başlatılması

16. Else if $n = \hat{n}\ then$

Daha önce görülmüş detay mı?

17. $K = \Sigma_{n,t-1}^{[m]} G_{\hat{n}} Q_{\hat{n}}^{-1}$

Kalman sayısının(gain) hesaplanması

18. $\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]} + K(z_t - \hat{z}_n)^T$

Ortalama değer güncellenmesi

19. $\Sigma_{n,t}^{[m]} = (I - K G_{\hat{n}}^T) \Sigma_{n,t-1}^{[m]}$

Artımlı sayaç

20. $i_{n,t}^{[m]} = i_{n,t-1}^{[m]} + 1$

İşlemler diğer kalan detaylar için tekrarlanıyor

21. Else

Bir önceki ortalama değerin alınması

22. $\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]}$

Bir önceki kovaryansın alınması

23. $\Sigma_{n,t}^{[m]} = \Sigma_{n,t-1}^{[m]}$

Detay bulunmuş olabilir mi?

24. İf $\mu_{n,t-1}^{[m]}$, $s_t^{[m]}$ ' algı alanında değilse then

Hayırsa, hiçbir şeyi değiştirme

25. $i_{n,t}^{[m]} = i_{n,t-1}^{[m]}$

Evetse, sayıyı bir azalt

Else

26. $i_{n,t}^{[m]} = i_{n,t-1}^{[m]} - 1$

Tekrarlanan detayları temizle

27. İf $i_{n,t-1}^{[m]} < 0$ then n numaralı harita özelliğini at, endif

Endif

Endif

Endfor

S_{aux} ' e $\langle s_t^{[m]},\ N_t^{[m]},\ \langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, i_1^{[m]} \rangle, \ldots,\ \langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]} \rangle \rangle$ ekle

Yeni parçacık kümesi oluştur

Endfor

28. $S_t = \emptyset$

M adet parçacığı bütünleştir

29. For $m' = 1\ to\ M\ do$

30. $\propto w_t^{[m]}$ Olasılıklı m rastgele indisini oluştur

S_t 'e $\langle s_t^{[m]},\ N_t^{[m]},\ \langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, i_1^{[m]} \rangle, \ldots,\ \langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]} \rangle \rangle$ ekle

Bütünleştirme aşaması

Endfor

Return S_t

End(algoritma)

FastSLAM algoritması 2.0 (z_t, u_t, S_{t-1}):

1. For $m = 1$ to M do

   S_{t-1} 'den $\langle s_{t-1}^{[m]},\ N_{t-1}^{[m]},\ \langle \mu_{1,t-1}^{[m]}, \Sigma_{1,t-1}^{[m]}, i_1^{[m]} \rangle, \ldots,\ \langle \mu_{N_{t-1}^{[m]},t-1}^{[m]}, \Sigma_{N_{t-1}^{[m]},t-1}^{[m]}, i_{N_{t-1}^{[m]}}^{[m]} \rangle \rangle$ oku

   Bütün parçacıkları oluşturulması, işaretlenmesi

2. For n = 1 to $N_{t-1}^{[m]}$ do

   $\hat{s}_t^{[m]} = h\left(s_{t-1}^{[m]}, u_t\right);\ \hat{z}_{t,n_t}^{[m]} = g(\mu_{n_t,t-1}^{[m]}, \hat{s}_t^{[m]})$

   $G_{\theta,n_t} = \nabla_{\theta_{n_t}} g(\theta_{n_t}, s_t)\Big|_{s_t=\hat{s}_t^{[m]};\theta_{n_t}=\mu_{n_t,t-1}^{[m]}};\ G_{s,n_t} = \nabla_{s_t} g(\theta_{n_t}, s_t)\Big|_{s_t=\hat{s}_t^{[m]};\theta_{n_t}=\mu_{n_t,t-1}^{[m]}}$

   $Q_{t,n_t}^{[m]} = R_t + G_{\theta,n_t} \Sigma_{n_t,t-1}^{[m]} G_{\theta,n_t}^T$

   $\Sigma_{s_t,n_t}^{[m]} = [G_{s,n_t}^T Q_{t,n_t}^{[m]-1} G_{s,n_t} + P_t^{-1}]^{-1}$; $\mu_{s_t,n_t}^{[m]} = \Sigma_{s_t,n_t}^{[m]} G_{s,n_t}^T Q_{t,n_t}^{[m]-1} \left(z_t - \hat{z}_{t,n_t}^{[m]}\right) + \hat{s}_t^{[m]}$

   Sampling dağılımının hesaplanması

3. $s_{n_t,t}^{[m]} \sim \mathcal{N}\left(\mu_{s_t,n_t}^{[m]}, \Sigma_{s_t,n_t}^{[m]}\right)$

   $$p_{n_t} = |2\pi\, Q_{t,n_t}^{[m]}|^{-\frac{1}{2}} exp\left\{-\tfrac{1}{2}(z_t - g(\mu_{n_t,t-1}^{[m]}, s_{n_t,t}^{[m]}))^T Q_{t,n_t}^{[m]-1} (z_t - g(\mu_{n_t,t-1}^{[m]}, s_{n_t,t}^{[m]}))\right\}$$

   Örnek konumun bulunması

   Endfor

4. $p_{N_{t-1}^{[m]}+1} = p_0$

   Yeni detayın benzerliği

5. $\hat{n}_t^{[m]} = argmax_{n_t} p_{n_t}$ Ve ya $\propto p_{n_t}$ olasılıklı $\hat{n}_t^{[m]}$ 'i rastgele oluştur

   Veri ilişkilendirme

6. For $n = 1\ to\ N_{t-1}^{[m]} + 1\ do$

   Proses ölçümü

7. İf $n_t = \hat{n}_t \leq N_{t-1}^{[m]}$ then

   $N_t^{[m]} = N_{t-1}^{[m]};\ \tau_{n_t,t}^{[m]} = \tau_{n_t,t}^{[m]} + \rho^+;\ K_t^{[m]} = \Sigma_{\hat{n}_t,t-1}^{[m]} G_{\theta,\hat{n}_t}^T Q_{t,\hat{n}_t}^{[m]-1}$

   $\mu_{\hat{n}_t,t}^{[m]} = \mu_{\hat{n}_t,t-1}^{[m]} + K_t^{[m]} \left(z_t - \hat{z}_{t,\hat{n}_t}^{[m]}\right);\ \Sigma_{\hat{n}_t,t}^{[m]} = (I - K_t^{[m]} G_{\theta,\hat{n}_t}) \Sigma_{\hat{n}_t,t-1}^{[m]}$

   $L_t^{[t]} = G_{s,\hat{n}_t} P_t G_{s,\hat{n}_t}^T + G_{\theta,\hat{n}_t} \Sigma_{n_t,t-1}^{[m]} G_{\theta,\hat{n}_t}^T + R_t$

   $$w_t^{[m]} = |2\pi\, L_t^{[t]}|^{-\frac{1}{2}} exp\left\{-\tfrac{1}{2}(z_t - \hat{z}_{t,\hat{n}_t})^T L_t^{[t]-1} (z_t - \hat{z}_{t,\hat{n}_t})\right\}$$

   Daha önceden bilinen detay mı?

8. Else if $n_t = \hat{n}_t = N_{t-1}^{[m]}$ +1 then

$n_t = N_t^{[m]} = N_{t-1}^{[m]} + 1;\ \tau_{n_t,t}^{[m]} = \rho^{+};\ w_t^{[m]} = p_0$

$s_{n,t}^{[m]} \sim p(s_t \mid s_{t-1}^{[m]}, u_t);\ G_{\theta,n} = \nabla_{\theta_n} g(\theta_n, s_t)\big|_{s_t = s_{n,t}^{[m]}; \theta_n = \mu_{n,t}^{[m]}}$

$\mu_{n,t}^{[m]} = g^{-1}\left(z_t,\ s_{n,t}^{[m]}\right);\ \Sigma_{n,t}^{[m]} = (G_{\theta,n} R_t^{-1} G_{\theta,n}^T)^{-1}$

Yeni detay mı?

9. Else if $n_t \neq \hat{n}_t\ and\ n_t \leq N_{t-1}^{[m]}$

   $\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]};\ \Sigma_{n,t}^{[m]} = \Sigma_{n,t-1}^{[m]}$

   Bulunamamış detayların ele alınması

10. İf $\mu_{n,t}^{[m]}$, $(s_{\hat{n}_1 t}^{[m]})$ algı alanı dışında kalıyorsa then

    Sensor algı alanının dışında mı kalıyor?

11. $\tau_{n_t,t}^{[m]} = \tau_{n_t,t-1}^{[m]}$

    Sensor algı alanının içinde mi kalıyor?

12. Else

    Detaydan vazgeçelim mi?

13. $\tau_{n_t,t}^{[m]} = \tau_{n_t,t-1}^{[m]} - \rho^{-}$; if $\tau_{n_t,t}^{[m]} < 0$ then n_t' at

    Endif

    endif

    endfor

    S_{aux} 'e $\langle s_t^{[m]},\ N_t^{[m]},\ \langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, i_1^{[m]} \rangle, \ldots,\ \langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]} \rangle \rangle$ ekle

    Bütün parçacıkların alınmasının durdurulması

    Endfor

14. $S_t = \emptyset$

    Yeni parçacık kümesini oluştur

15. For $m' = 1\ to\ M\ do$

    M adet parçacık oluştur

16. $\propto w_t^{[m]}$ Olasılıklı m rastgele indisini oluştur

    S_t 'e $\langle s_t^{[m]},\ N_t^{[m]},\ \langle \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, i_1^{[m]} \rangle, \ldots,\ \langle \mu_{N_t^{[m]},t}^{[m]}, \Sigma_{N_t^{[m]},t}^{[m]}, i_{N_t^{[m]}}^{[m]} \rangle \rangle$ ekle

    Bütünleştir (resample)

    Endfor

    Return S_t

    Endalgoritma

Ek 2 FastSLAM Algoritmasının Uygulaması

FastSLAM algoritmasının iki farklı sürümü bulunmaktadır. Çalışmada FastSLAM 2.0 algoritması kullanılmıştır. Simülasyonun çalışma sürecinde çalışılan sisteme ve robot konfigürasyonuna has birtakım parametreler mevcuttur. Bu parametrelerin değer atamaları sistemde 'Constants' klasında yapılmaktadır. Harita detaylarının (engel, detay) pozisyonu ise simülasyonda engel ölçümlerini hesaplamak için ve alınan ölçümlerden yola çıkılarak bulunan engel pozisyonları ile gerçek engel pozisyonlarını haritada gösterebilmek için gereklidir.

Uygulamada, kontrol parametreleri, doğrusal ve açısal hızdan oluşmaktadır. Sırayla [**V**; **G**] değişkenleriyle gösterilir. Birimleri sırayla cm/sn ve radyan/sn verilmiştir. İlk önce robot konfigürasyonuna dair atamalar yapılır. Örneğin, ön tekerin kontrol işaretleri sonucunda sahip olabileceği maksimum açı değeri radyan cinsinden **MAXG**'de tutulmaktadır. Ön tekerin saniyedeki maksimum değişim oranı (açısal hızı) radyan/sn cinsinden **RATEG**'de; robotun aks (dingil) yani ön teker ve arka teker arasında uzanan birleştirici doğrunun uzunluğu (ön teker-arka teker mesafesi) metre cinsinden **WHEELBASE**'de tutulmaktadır. Simülasyonda kullanılan robot modelinde bu mesafe 16 cm olarak belirlenmiştir.

Bunların dışında algı ile ilgili belirlenmesi gereken parametreler aşağıdaki gibidir:

- Peş-peşe gönderilen iki kontrol işareti arasında geçen süre saniye cinsinden **DT_CONTROLS**'te,
- Kontrol işaretlerini (doğrusal ve açısal hızlar) gönderirken oluşabilecek (Gaussian tipteki) gürültülerin standart sapma değerleri sırasıyla **sigmaV** (cm/sn cinsinden) ve **sigmaG**'de (rad/sn cinsinden),
- Alınan gözlemlerde görülebilecek maksimum mesafe santimetre cinsinden **MAX_RANGE**' de,
- Alınan gözlemler arasında geçen süre sn cinsinden **DT_OBSERVE**'de;
- Gözlem esnasında oluşabilecek gürültülerin standart sapmaları **sigmaR** (santimetre cinsinden-doğrusal mesafe gürültüsü için) ve **sigmaB**'de (rad cinsinden-açısal fark (eğim farkı) gürültüsü için);
- Gezinme sürecinin kaç defa yapılacağı **NUMBER_LOOPS**'ta;
- Parçacık adedi **NPARTICLES**'ta;

- Yeniden örnekleme (resampling) yapmadan önce işe yarayan (efektif) parçacık adedi **NEFFECTIVE**'de;
- Kontrol, gözlem ve tahmin süreçlerine gürültü eklenip eklenmeyeceğine karar verecek olan kontroller sırayla **SWITCH_CONTROL_NOISE**,

  SWITCH_SENSOR_NOISE, **SWITCH_PREDICT_NOISE**'te tutulmaktadır.

Öncelikle NPARTICLES adet parçacığa ilk değer ataması yapılır. İlk robot pozisyonu (xtrue) belirlendikten sonra ilk açısal hız (G) değerine 0 atanır. Q: kontrol gürültülerinin (doğrusal ve açısal hız için) standart sapmalarının karelerini (varyans) diagonelinde bulunduran kovaryans matrisi; R: gözlem gürültülerinin (doğrusal ve açısal mesafe ölçümü için) varyanslarını diagonelinde bulunduran kovaryans matrisidir. Q'da ve R'de doğrusal ve açısal gürültüler birbirinden bağımsızdır, dolayısıyla sadece diagonelleri doludur. (Yani Q(1,2)=Q(2,1)=0 olur: doğrusal ve açısal hız gürültü ilişkisizdir; aynı şekilde R(1,2)=R(2,1)=0'dır.)

Daha sonra yeni robot pozisyonu hesaplandıktan (**xtrue)**. Kontrol gürültüsü eklenerek gürültülü kontrol datası üretiliyor (**[Vn,Gn]**)**.**

Tahmin adımında her bir parçacık robot pozisyonuna dair kendi tahminini buna kendi tahmin gürültüsünü ekleyerek yapar. Sonra gözlem işlemleri yapılır. Gözlem adımında peş-peşe yapılan iki gözlem arasında geçmesi gereken min süreye ulaşılmışsa, gözlem alınır ve sayaç sıfırlanır(**dtsum**). Robotun görüş alanına giren harita detaylarının gerçek ölçüm değerleri (robot-detay arası gerçek mesafeler) ve bu engellerin indisleri bulunur (**[z,ftag_visible]**). Hesaplanan gerçek ölçümlere (**z**) gürültü eklenir.

Nf: 1.parçacıkta associated (önceden görülen) detay sayısıdır. Tüm parçacıklar eşit sayıda detay içerdiği için herhangi birindeki detay sayısını (**xf**' deki eleman sayısını) almak yeterlidir.

Daha sonra ‘‘veri ilişkilendirme ’’ olup olmadığı hesaplanır ve **da_table** değişkeninde önceden görülen detayların indisleri tutulur.**[zf,idf,zn,da_table]**

Burada **zf** ve **idf**: Önceden görülen detayların ölçüm ve indis değerleri; **zn**: yeni bulunan detayların ölçümünü göstermektedir.

Güncelleme adımında önceden görülen detay'ler kümesi **zf** doluysa: **i** numaralı parçacığın ağırlığı (**w**), **i** numaralı parçacığı, **zf**, **idf** ve **R**-gözlem gürültü matrisini kullanarak bulunur. Sonraki adımda **i** numaralı parçacığın ağırlığı (w alanı) güncellenir. **i** numaralı parçacığın detay konumu tahmin dağılımının Multivariate Gauss olduğunu varsayılmaktadır. Detay

konum tahmininde dağılımın **xf** ortalama değer vektorü ve **Pf** kovaryans matrisi güncellenir. Bir sonraki adımda yeni detay bulunduysa harita büyütülmeli ve parçacıklara ekleme yapılmalıdır ve işlem sırasında **i** numaralı parçacığa **zn** ve **R**'yi kullanarak, yeni bulunan ağırlıkların tahminini eklenmelidir. Bundan sonra ise parçacıklar, önerilen dağılıma göre yeniden örneklenir (Resampling) ve çıktı olarak, son durumu gösteren parçacıklar kümesi döndürülür (**data= particles**).

FastSLAM'de gereken ilk değer atamaları yapıldıktan sonra **NPARTICLES** adet parçacığın ilk değer atamaları yapılır. (**np**: **NPARTICLES**). **1/ NPARTICLES** ağırlık değerleri başta hepsine eşit atanır. **xv i** numaralı parçacığın robot pozisyonu tahmini "**xv**" alanındadır ve [x;y;teta] şeklinde, **xf** i numaralı parçacığın detay pozisyonlarının tahminidir ve [x;y;teta] şeklindedir. **Pf** ise i. parçacığın detay konum tahmin dağılımının Multivariate Gauss olduğunu yukarıda kabul etmiştik. Detay konum tahminde **xf** bu dağılımın ortalama değer vektorü, **Pf** ise kovaryans matrisidir.

Prior state tahmini aşamasında ise **[xfi,Pfi]**, innovasyon matrisleri **[v=zp-z, R]** ve doğrusallaştırılmış gözlem modeli **(H)** kullanılarak KF güncelleştirmesini yapılırken kullanılan geçiş matrisidir. **Pf** boyutu: (2x2xdetay sayısı)'dir.

Gerçek pozisyondan, kontrol parametreleri [V; G]'den, **dt** ve robot konfigürasyon parametrelerinden yola çıkılarak, yeni robot pozisyonu hesaplanır ve döndürülür (**xv**).

Bir adım sonra ise kontrol gürültüsü eklenerek gürültülü kontrol verisi **(Vn, Gn)** üretilip gönderilecektir.

Harita detaylarının algılanması aşamasında detaylarla robot konumu arasındaki mesafe ve eğim açısı farkları bulunur (**dx, dy, phi) z**, doğrusal ve açısal gözlem mesafeleri hatası bağımsız olduğundan R kovaryans matrisinin sadece diyagonali kullanılarak buna gürültü eklenir. Sonra sırasıyla **[zp,Hv,Hf,Sf]** Jakobian hesapları yapılır. Burada;

zp: Parçacığın tahmin ettiği ölçüm (robot-landmark arası doğrusal ve açısal mesafe);

Hv: robot durumunu hesaplayan Jacobian matrisi;

Hf: detay durumunu hesaplayan Jacobian matrisidir ve Kalman Filtresinde gözlem modeli gibidir;

Sf: robot durumu verildiğinde, detay durumunu hesaplayan matristir ve robotun konumundan harita detayı değerini bulan yeni bir kovaryans matristir.

Yapılan bütün bu işlemlerden sonra parçacıklar yeniden örneklenmelidir. Bu aşamada ağırlık dizisini (**w**) üzerinden; stratified (çok katmanlı) yeniden örnekleme yapılır ve çıkışta korunacak parçacıkların indisi ve efektif olan parçacık adedi elde edilir. Yeniden örnekleme sonucunda bazı parçacıklar yeni kümeye giremez, bazısı ise birden fazla kez girer. Hata ne kadar düşükse ağırlık o kadar yüksek olur; hata arttıkça da ağırlık değeri azalır.

ÖZGEÇMİŞ

Doğum tarihi	10.07.1985	
Doğum yeri	Sumgait, Azerbaycan	
Lise	1997 – 2002	Sumgait Özel Türk Lisesi
Lisans	2002 – 2006	Bakü Devlet Üniversitesi, Uygulamalı Matematik ve Sibernetik bölümü
Yüksek Lisans	2007 – 2011	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği bölümü