İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL ROBOTLARDA PARÇACIK FİLTRESİ KULLANARAK EŞ ZAMANLI LOKALİZASYON VE HARİTALAMA

YÜKSEK LİSANS TEZİ

Ali KULELİ

Anabilim Dalı : Elektrik Mühendisliği

Programı : Kontrol ve Otomasyon

ŞUBAT 2009

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

MOBİL ROBOTLARDA PARÇACIK FİLTRESİ KULLANARAK EŞ ZAMANLI LOKALİZASYON VE HARİTALAMA

YÜKSEK LİSANS TEZİ
Ali KULELİ
(504061101)

Tezin Enstitüye Verildiği Tarih : 22 Aralık 2008
Tezin Savunulduğu Tarih : 19 Ocak 2009

Tez Danışmanı : Prof. Dr. Hakan TEMELTAŞ (İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. Metin GÖKAŞAN(İTÜ)
Doç. Dr. Osman Kaan EROL(İTÜ)

ŞUBAT 2009

ÖNSÖZ

Parçacık filtresi ile ilgili yaptığım çalışmaların ışığında hazırladığım bu tezin "Eş Zamanlı Lokalizasyon ve Haritalama" problemine ilgi duyan veya çalışmak isteyen akademisyenlerin yararına olduğuna düşünüyorum. Uğraşılan problemin büyük çevrelerdeki uygulama yöntemleri üzerinde, bilim dünyasının hala aktif olarak üzerinde çalıştığı düşünülürse parçacık filtresinin konuya bir bakış açısı kazandırdığı düşünülebilir, filtreye bağlı yeni çözüm yolları geliştirilebilir. Hala İTÜ'de sürmekte olan ve TÜBİTAK tarafından desteklenen "Eş Zamanlı Lokalizasyon ve Haritalama" probleminde araştırdığım filtreyi uygulama imkânı bulduğum ve yüksek lisansım süresince bana verdiği desteklerden dolayı Tübitak'a, konu üzerinde fikirlerimin olgunlaşması için bana yol gösteren ve yardım eden Prof. Dr. Hakan Temeltaş'a, filtre simülasyonunun programlaması sırasında bana yardımcı olan proje grubuma ve çalışmalarıma üniversite yaşamım boyunca hep inanan aileme teşekkürlerimi borç bilirim.

Aralık 2008

Ali Kuleli

Elektrik Mühendisi
Elektronik ve Haberleşme Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

SLAM **:** Simultaneous Localization And Mapping, Eş Zamanlı Lokalizasyon ve Haritalama

FastSLAM **:** Hızlı olarak Eş Zamanlı Lokalizasyon ve Haritalama

RBPF **:** Rao-Blackwellised Parçacık Filtreleri

GKF **:** Genişletilmiş Kalman Filtresi

CML **:** Ardışıl Harita İşleme

MCL **:** Monte Carlo Lokalizasyonu

SIR **:** Ardışıl Önem Örneklemesi(Sequential Importance Resampling)

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

MOBİL ROBOTLARDA PARÇACIK FİLTRESİ KULLANARAK EŞ ZAMANLI LOKALİZASYON VE HARİTALAMA

ÖZET

Parçacık filtreleri, günümüzde mobil robotların kullanıldığı yön bulma ve haritalama işlemleri için, varolan eş zamanlı lokalizasyon ve haritalama probleminin(SLAM) çözümlenmesini sağlayacak Bayes teoremi temeline dayanan bir alternatif filtreleme yöntemidir. Parçacık filtresi, genişletilmiş Kalman filtresi(GKF) gibi doğrusal olmayan denklemleri, doğrusal gauss tipi denklemlere yakınsatarak SLAM probleminin çözülmek yerine, gauss tipi olmayan süreçleri ve dağılımları içeren ardışıl Monte Carlo tekniklerini kullanır.

Tezin amacı, parçacık filtresinin SLAM problemindeki başarısını ölçmektir. Tezde popüler olarak tercih edilen parçacık filtrelerinin özelleştirilmiş bir modeli olan, Rao-Blackwellisation adlı variyans azaltma tekniğini içinde barındıran FastSLAM adlı Rao-Blackwellised parçacık filtreleri incelenmiştir. FastSLAM SLAM problemini robotun yer bulma problemi ve işaretçi nesnelerin yerlerinin tahmin edilmesi problemi olarak ikiye ayırır. Her parçacık, çoklu veri eşleştirmesine izin veren bir biçimde, çevre haritasını ve robot lokalizasyonunu saklar. Mobil robotun yer bulma probleminde Monte Carlo lokalizasyonu(MCL) tarafından işaretçi nesnelerin konumlarının kestirimi ise GKF ile yapılır. Bu yapı ile gauss tipi işaretçi nesne kestirimi parçacık filtresine kazandırılmış olur.

RBPF'nin anlaşılması açısından iteratif Bayes kestirimi, Kalman filtresi, MCL, maksimum benzerlik kestirimi konuları açıklanmış ve filtrenin adımlarını olan örnekleme, ardışıl önem ağırlıkları hesaplaması, haritalama, yeniden örnekleme ve kriterlerinden bahsedilmiştir.

Simülasyon ortamının oluşturulması için MATLAB programından faydalanılmıştır. Kare biçimindeki simülasyon alanında belirli aralıklarla lazer tarayıcı yardımı ile gözlemler yapılarak parçacıklar oluşturulmuştur. Filtrenin başarısının tespiti için, farklı parametrelerin değişiminin filtrenin çalışmasına etkisi incelenmiştir. Ölçüm ve süreç gürültülerinin, farklı parçacık sayılarının, çevre karmaşıklığının, farklı yeniden örnekleme kriterlerinin filtrenin çalışmasına etkileri ve SLAM'in zorluklarına sundukları cevaplar gözlenmiştir.

SIMULTANEOUS LOCALIZATION AND MAPPING FOR MOBILE ROBOTS WITH PARTICLE FILTERS

SUMMARY

Nowadays, particle filters which consist on the Bayes theorem offer an alternative filtering method to solve the simultaneous localization and mapping problem (SLAM) in the robotic navigation and mapping areas. Particle filters which contain non-linear Gaussian process and distributions use sequential Monte Carlo techniques instead of approximating the non-linear, non-Gaussian process to the linear Gaussian ones such as made by the extended Kalman filter (EKF).

The purpose of the thesis is to measure the success rate of the particle filters for the SLAM problem. In the thesis, popularly used and specialized form of the particle filters, the FastSLAM algorithm in which a variance reduction technique named as Rao-Blackwellisation is found, is observed. FastSLAM divides the SLAM problem into two categories such as the robotic mapping problem and the estimation of the locations of the landmarks. Each particle of the algorithm stores the landmarks' map and the robot pose rendering the multiple hypothesis data associations. The robot pose estimation is computed by Monte Carlo localization (MCL) algorithms and the coordinates and their uncertainties of the landmarks are estimated by the EKF. This model implements the Gaussian estimation of the observations into the particle filters.

Sequential Bayes estimation, Kalman filter, MCL, maximum likelihood estimation and the application steps which are sampling, importance weighting, mapping, resampling and its criteria are included in the thesis for a better comprehension of RBPF.

A suitable simulation area is created with the MATLAB program. The particle clusters are formed with observations made by the laser scanner. The scanner scans the square shaped area on convenient spots. For measuring the success rate, changes in the particle numbers, the process noise, the laser measurement noise, the complexity of the environment, different resampling methods are tested and their effects are observed in order to decrease the difficulties of the SLAM.

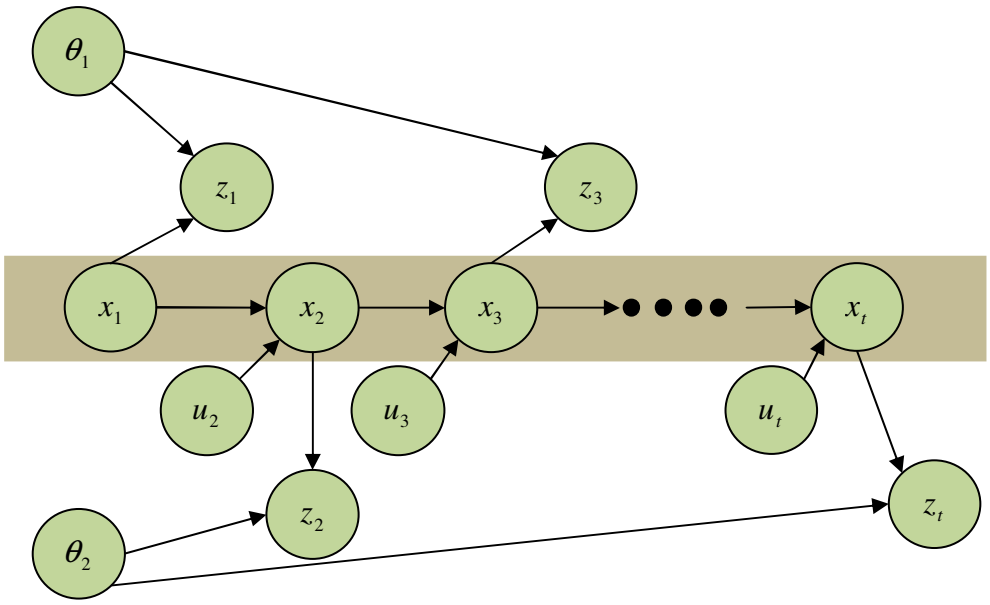
1. GİRİŞ

Bilinmeyen bir çevrenin haritasını çıkarma ve öğrenme işi mobil robot sorunlarının en önemlilerinden biridir. Literatürde, eş zamanlı lokalizasyon ve haritalama problemi(SLAM) olarak geçer[1-5]. Çünkü robot çevresini keşfederken, hem üzerindeki algılayıcılardan etrafındaki cisimlerin veya algıladığı işaretlerin haritasını oluşturur hem de kendi bulunduğu noktayı kestirir. Genel olarak, daha geniş alanlarda gezdikçe, haritanın eski kısımlarına ait olan cisim belirsizlikleri artar. Problemin anahtar noktası, robotun bilmediği pozisyonunu bilinmeyen bir çevrede veya daha önceden tahmin ettiği çevrede kestirebilmesidir. Bu ise bir tavuk yumurta problemine benzer bir yaklaşımdır[4].

SLAM problemi Smith, Self ve Cheeseman tarafından bir bildiri olarak yayınlandıktan sonra artımsal kestirimlerin dağılımı için önerdikleri genişletilmiş Kalman filtresi(GKF) çözümü bilimsel olarak robot alanında yönetici bir yaklaşım olarak yerini almıştır. Son yıllarda yapılan deneyler bu GKF yaklaşımını temel alarak, sorunu az sayıdaki işaretçi nesneler bulunan dar çevrelerden çıkartarak çok sayıdaki işaretçi nesneler bulunan geniş alanların haritalanmasını sağlamıştır. Böylece SLAM problemi, GKF'nin işlemsel karmaşıklığından soyutlanarak yeni algoritmaların uygulanma alanı haline gelmiştir[3].

GKF 'nin güçlü tarafı robotun konumlarından ve işaretçi nesnelerin yerinden, robotun kendi konumunun soncul olasılığını çok iyi bir şekilde tahmin edebilmesidir. Ancak zayıf tarafı, mobil robotun hareket modelinin çok iyi yapılmış olması gerektirir ve algılayıcı gürültülerinin çok iyi bilinmesini gerektirir. Ayrıca işaretçi nesne sayısını 'm' olarak alırsak, algılayıcının gözlemlerini yenileyip tek bir 'yeni' işaretçi nesne bulmasında bile kendi kovariyans matrisinin yeniden oluşturulması için $O\left(M^2\right)$ 'lik[1] elemanın algoritmada yeniden işlenmesi lazımdır ki[3,4], bu da birkaç yüz işaretçi nesneyi aşan çevrelerde veri eşleştirme problemi de düşünülürse süreçte karmaşıklığı arttıran bir durumdur.

Bu çalışmada SLAM problemi Bayes teori ile açıklanmıştır. Şekil 1.1'de bulunan Dinamik Bayes Ağı SLAM'ın gövde yapısını açıklamaktadır. Robotun konumları $x_1, x_2 \ldots, x_t$ zamanla $u_1, u_2 \ldots, u_t$ kontrol işaretlerine göre değişir. Her işaret gözlemi $z_1, z_2 \ldots, z_t$ işaretlerin gerçek pozisyonu olan $\theta_{1:M}$'lerin zamana göre fonksiyonlarıdır. Bu şekilde, SLAM sorununun bağlı olduğu şartlar gözükür. Robotun $x_1, x_2 \ldots, x_t$ konum bilgisi işaretçi nesnelerin konumu ile bağıntılıdır. Eğer robotun gerçek yolu bilinirse, araç süreçten dışlanarak M tane θ'nın tahmini durumuna döner[1].



Şekil 1.1 : Dinamik Bayes Ağı

Tahmin algoritmalarını kabaca içerdiği çalışma mantıklarına göre sınıflayabiliriz[2]. Bunların içlerinde en popülerleri şu anda bu çalışmada yer alan GKF, maksimum benzerlik teknikleri ve parçacık filtreleridir.

Kullanılan maksimum benzerlik algortimaları sensör okumalarına bakarak gerçeğe en yakın haritayı oluşturmaya çalışır ve ilişki kümeleri yaratarak robotun yerini kestirir. Gutmann bu algoritmayı artımsal bir biçimde kullanarak robotun gezdiği döngüleri doğru bir biçimde tahmin etmeyi başarmıştır. Hahnel bu algoritmada veri ilişkilendirme kullanmıştır[2]. Ancak, bu algoritma ile kullanılan veri ilişkilendirme ağaçları gerçek zamanlı çalışmalara izin vermemektedir.

Murphy'nin çalışmasında SLAM problemini çözmek için etkili bir FastSLAM yolu olan Rao-Blackwellised Parçacık Filtreleri(RBPF) önerilmiştir. Bu çalışma, Montemerlo tarafından farklı işaretçi nesne haritaları kullanılarak genişletilmiştir[2]. Şu anda da popüler olarak üzerinde geliştirmeler yapılmaktadır. Çalışmamızda parçacık filtrelerinin gelişmiş bir modeli olan Rao-Blackwellised Parçacık Filtresini teorik olarak inceledik ve simülasyonlar bu filtre tekniğine göre yapıldı.

Veri eşleşim algoritmalarından yararlanılmadan FastSLAM için n'yi parçacık sayısı alırsak ; Montemerlo algoritma zamanını O(nm) zamana indirebilmiş ve eklediği veri eşleştirme algoritmaları ile O(nlogm) zamanına ulaşmıştır[1].

1.1 Tezin Amacı

Tezin amacı, parçacık filtresinin eş zamanlı yer belirleme ve haritalama problemindeki başarısını ölçmektir.

Bayes kestirimine dayanan filtrelerin genel olarak açıklaması yapılacaktır, daha sonra parçacık filtresinin RBPF modelinde kullanılan GKF teorik olarak açıklanacaktır. Parçacık filtresinin özelleştirilmiş bir modeli olan Rao-BlackWellised parçacık filtresinin kullanım sebebi GKF filtresini kendi içinde ihtiva etmesi ve son yıllarda üzerinde çalışmalar yapılmış olmasından ileri gelmektedir. Simülasyonda kullanılan araç hareket modeli, gözlem modellerinden bahsedilecektir. SLAM problemi incelenecek ve karşılaşılan sorunlardan bahsedilecektir. Anlatılmış yapıların üzerine parçacık filtresi, çalışma modeli ve adımları anlatılacaktır. Konudaki deneylerin yapılmasını sağlayan MATLAB RBPF simülasyunundan bahsedilecektir. Bu bağlamda RBPF'nin çizelge 1.1'deki paratmetrelerinin değişmesinin SLAM problemine etkisi ve yapılan hata oranları ile ölçülecektir.

Çizelge 1.1 : Parçacık Filtresi Başarısı Karşılaştırma Kriterleri

Gürültü	İşaret Sayısı	Parçacık Sayısı	Yeniden Örnekleme
Lazer Tarayıcı	0 -50	10	Basit
Süreç	51-200	10-100	Adaptif Sistematik

2. MOBİL ROBOTLAR

Mobil robotların yapıları ve haritalama işlemesinin teorik olarak anlatılmasından önce tarihinin anlatılması yerinde olur.

2.1 Robotik Haritalamanın Tarihi

Robotik haritalamanın uzun bir tarihi vardır. 1980'lerde ve 1990'ların başında, haritalama biliminde topolojic ve metrik yaklaşımlardan yararlanılırdı. Metrik haritalar bir çevrenin geometrik içeriklerini ihtiva ederdi, bunun yanında topolojik haritalar farklı bölgeleri eğrilerle birbirlerine bağlanan önemli yerler olarak bağlantısını tanımlardı. Haritalamada algoritmaların gelişmesi Elfes ve Moravec'in ızgara işgal haritalama algoritması yaratıcı bir fikir olarak karşımıza çıkan ilk örneklerdendir. Bu örnekte o anda meşgul(dolu) veya boş olan olarak harita karelere bölünmüştür. Bu yöntem daha sonra robotik sistemlerde de kullanılmıştır. Buna bir seçenek olarak ortaya çıkan Chatila ve Laumond'un metrik haritalaması çevreyi polyhedra kümeleri ile tanımlamıştır.

Thrun'a göre[6], ancak topolojik ve metrik harita arasındaki ayrım her zaman bulanıktır, çünkü topolojik yaklaşımlar her zaman geometrik bilgilere dayanmak zorundadırlar. Robotik alanda yüksek çözünürlüklü metrik haritalar her zaman işlevsel zorluğu olmasına rağmen zor problemlerin çözülmesinde daha çok yardımcı olurlar. Oysa topolojik olarak her zaman ek bir geometrik bağıntı gerekir.

Bu iki yaklaşımın yanında dikkat edilmesi gereken bir başlık da haritanın hareket eden robota göre temelleneceği mi, yoksa çevreye göre mi temelleneceğiydi. Çevre merkezli haritalarda global koordinat uzayında tanımlanırken, çevre bilgilerinden algılayıcı ölçüm bilgileri çıkartılamazdı. Robot merkezli haritalar ise ölçüm uzayında tanımlanırdı ve robotun çeşitli noktalarda aldığı algılayıcı bilgilerinden faydalanılırdı. Robotta haritalama için robot merkezli bir haritayı oluşturmak daha kolay gelse de iki ana zorlukla karşılaşılır. Ölçümden ölçüme ilişki kurmak her zaman kolay olmaz; özellikle daha önce hiç bilinmeyen bir bölgede kolay olmaz.

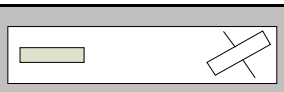
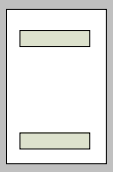
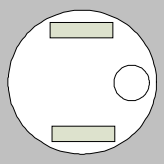
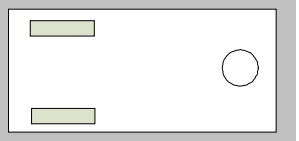
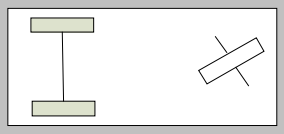
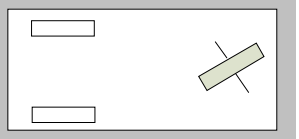
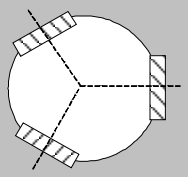
Öte yandan birbirine benzeyen bilinmeyen çevreler robotla ziyaret edilecekse robot merkezli yaklaşım onları birbirlerinden ayırt etmekte zorlanır. Bu yüzden önemli mobil robotla yaklaşımlar çevre merkezli olarak yapılırlar.

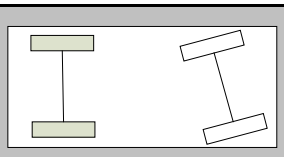
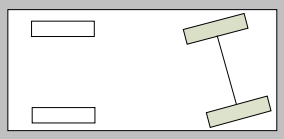
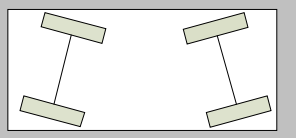
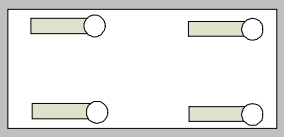
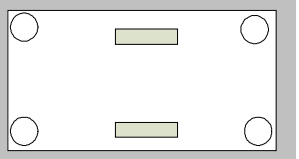
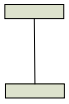
1990'ların başında Smith, Self ve Cheeseman sürekli olarak robotun sürekli büyüyen bir çevrede bulunduğu yeri haritalaması ve bu haritaya bağlı olarak kendi yerini bulmasını sağlayan kuvvetli bir istatiksel çalışma yayınlar. Bu süreç *Eş Zamanlı Lokalizasyon ve Haritalama*(SLAM) veya *Ardışıl Haritalama ve Lokalizasyon*(CML) olarak adlandırılır. Problemin çözümü için Kalman filtreleri kullanıldı. Oluşturulan haritalarda işaretçi nesnelerin yerleri saptandı ve sonraki uygulamalarda ise, lazer ölçüm bilgilerinden çevrenin haritası çıkartıldı. Kalman filtresine Alternatif bir algoritma ailesi de Dempster'ın *Beklenen Değer Maksimizasyonudur*. Bu yaklaşımlar veri ilişkilendirme problemini beraberinde getirmişlerdir. Bu problem farklı zamanlarda mobil robotlarda yapılan ölçümlerin fiziksel çevrede aynı noktaya düşüp düşmemesi ile ilgilidir. Haritalamadaki diğer bir yaklaşım ise çevredeki cisimlerin tanımlanması ile ilgilenir. Bunlar duvar, tavan, taban, mobilyalar, pencere veya dinamik hareket eden cisimler olabilir ki bu teknikler görüntü işleme dalı ile bağıntılıdır. Bahsedilen yöntemlerin bugünkü yaklaşımları yine Thrun'a göre aç gözlü olarak nitelendirilir[6]. Çünkü bu yaklaşımlar maksimum haritalama bilgisinin sağlanması için robotun güvenlik sınırları içinde çalışmasını engelleyebilir.

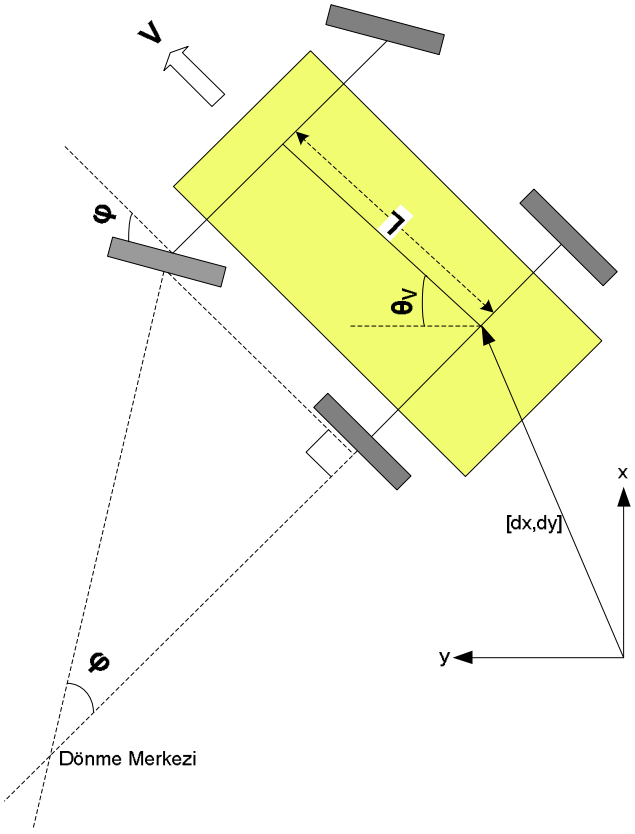
2.2 Mobil Robot Hareketi

Mobil Robotun hareketi çalışmada önemli bir yer kaplamaktadır. Kontrol işaretinin uygulanması tamamen kontrol edenin veya hazırlanmış bir algoritmanın denetiminde olsa da, aracın fiziksel modelinin filtrelere olan etkisi önemlidir. Bundan dolayı çalışmada mobil aracın hareket denklemlerinden faydalanılmıştır. Üretim öncesi bir çok tekerlek sistemi göz önüne alınmıştır. En uygun olanı aralarından seçilmiştir.

Çizelge 2.1 : Tekerlekli kara araçlarında çeşitli teker konfigürasyonları (Z.KurtY.L.Tezi)

Teker sayısı	**Teker Düzeni**	**Açıklama**	Örnek
2		Yön belirleyen bir ön teker, gövdeyi ön tekerin çektiği yere sürükleyen bir arka teker vardır.	Bisiklet, Motorsiklet
		İki teker bir mille bağlıdır, ağırlık merkezi milin orta noktasıdır.	
3		Önde bir teker, arkada farklı sürüş merkezleri olan iki teker bulunmaktadır.	
		Önde çok yönlü bir teker, arkada iki bağımsız teker vardır.	Çoğu kapalı ortam robotları (Plgmalion ve Alice)
		Önde yön veren bir teker, arkada bir mille birbirine bağlı iki ayrı teker vardır.	
		Önde yön veren bir teker, arkada iki teker vardır.	Neptune (Carnegie Mellon Üniversitesi)
		Bir üçgen düzenine oturtulmuş üç adet çok yönlü İsveç tekeri veya küresel teker.	Tribolo EPFL, Palm Pilot Robot Kit (Carneige Mellon Üniversitesi)

4		Önde yön veren iki teker, arkada motorlu iki teker bulunmaktadır.	Arkadan sürüşlü arabalar
		Önde iki motorlu ve yön veren teker, arkada iki serbest teker vardır.	Önden sürüşlü arabalar
		Dört adet yön belirleyen ve motorlu teker vardır.	Dört tekerli sürüşe sahip Hyperion (Carnegie Mellon Üniversitesi)
		Dört adet motorlu ve yön belirleyen taşıyıcı tekerler vardır.	Nomad XR4000
6		Merkezde iki adet çekici (sürükleyici) teker, her köşede birer tane çok yönlü teker vardır.	Terragator (Carnegie Mellon Üniversitesi)
	Güç verilmemiş çok yönlü teker (küresel, nakliye tekeri, İsveç tekeri)		
	Motorlu İsveç tekeri (Stanford tekeri)		
	Güç verilmemiş standat teker		
	Motorlu standart teker		
	Motorlu ve yön belirleyen nakliye tekeri		
	Yön belirleyen standart teker		
	Bağlı tekerler		



Şekil 2.1 : Holonomik olmayan Ackermann tekerlek sistemi

Çizelge 2.2 : Ackermann Tekerlek Sisteminde Kullanılan Semboller

L	**Aracın tekerlekler arası boyu**
V(k)	Aracın k anındaki hızı
Φ(k)	Aracın tekerleğine k anında uygulanan dönme açısı
θ(k)	Aracın xy koordinat düzlemine göre dönme miktarı
(x,y)	Aracın düzlemdeki konumu

Günümüz araçlarında şekil 2.1'deki Achermann tekerlek sistemi esas alınarak önden, arkadan veya dört tekerlekten çekiş sistemi ile üretilmektedir. Holonomik olmayan(yerel dönme açısı global dönme açısından küçük olan) Ackermann tekerlek sisteminde aracın 2 boyutlu yüzeydeki koordinaatları denklem 2.1'deki gibi verilmiştir.

$$x_v = \begin{bmatrix} x_v \\ y_v \\ \theta_v \end{bmatrix} \tag{2.1}$$

$$\begin{aligned} \dot{x} &= V.Cos(\theta) \\ \dot{y} &= V.Sin(\theta) \\ \dot{\theta} &= V\frac{Tan(\phi)}{L} \end{aligned} \tag{2.2}$$

$$u(k) = \begin{bmatrix} V(t) \\ \varphi(t) \end{bmatrix} \tag{2.3}$$

Araç kontrol işaretine ayrıca bir gürültü eklendiği de düşünülebilir.

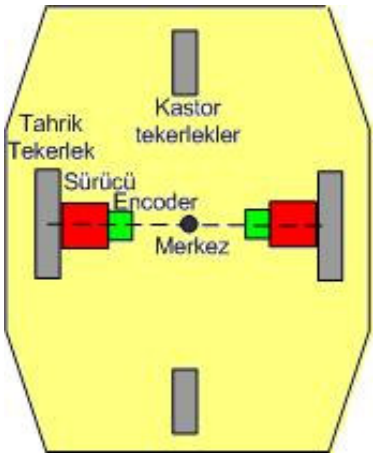
$$u(t) = u(t) + v(t) \tag{2.4}$$

Aracın t zamanından t+1 zamanına ötelendiğini düşünürsek uygulamalarda kullanılabilecek dinamik denklem modeli ise aşağıdaki gibi gösterilir[7].

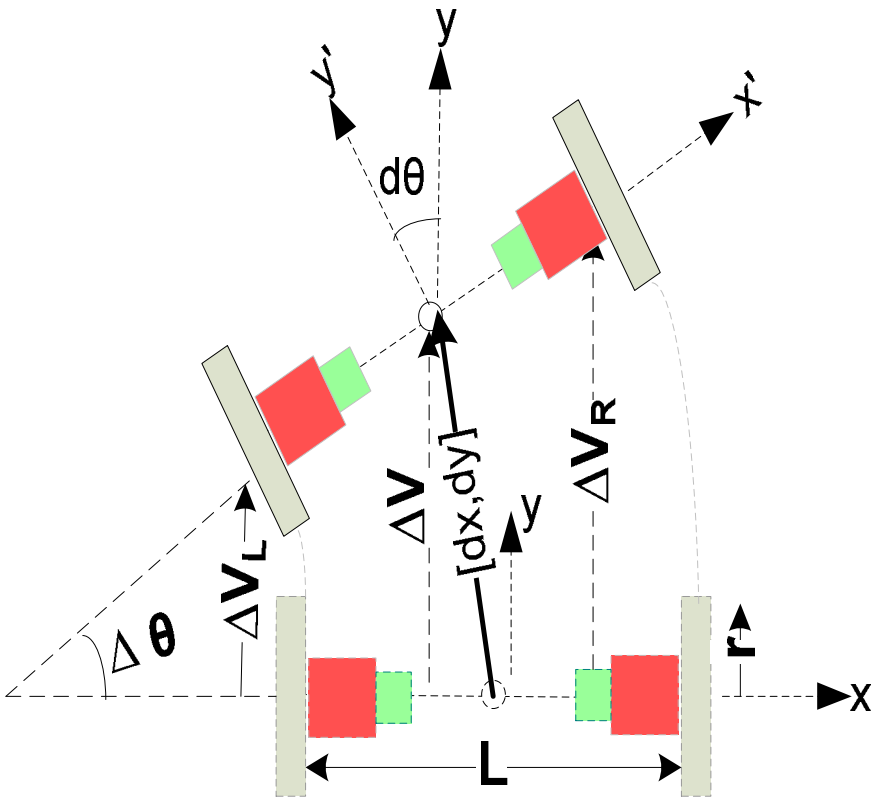
$$\begin{bmatrix} x_v(t+1) \\ y_v(t+1) \\ \theta_v(t+1) \end{bmatrix} = \begin{bmatrix} x_v(t) + \Delta T.V(t).Cos(\theta_v(t)) \\ y_v(t) + \Delta T.V(t).Sin(\theta_v(t)) \\ \theta_v(t) + \dfrac{\Delta T.V(t).Tan(\varphi_v(t))}{L} \end{bmatrix} \tag{2.5}$$

Deneylerde diferansiyel tekerlek modeli tercih edilmiştir. Robotun kullanılan modeli şekil 2.3'deki gibi resmedilmiştir. Bu model çizelge 2.1'deki 3 numaralı modellere benzemektedir, farklı olarak bir yerine önde ve arkada olmak üzere iki kastor tekerlek kullanılmıştır. Bu modelin çok kullanılan Ackermann tekerlek modeli yerine tercih edebilmesinin nedeni ana tekerleklere merkezli servo motorlarla olduğu yerde yön değiştirmesi ve olduğu yerden fazla manevra gerktirmeden geri geri çıkabilmesidir. Böylece önünün arka, arkasının da ön haline gelip çalışmaya devam edilebilmesi düşünülebilir. Aracın ön ve arka ucunda bulunan kastor tekerleklerde servo motorlardan bağımsız olarak araç yükünü kaldırdıkları gibi, bir hareket serbestliği de sağlarlar.

Şekil2.2'de üretilmiş aracın modeli ve çizelge 2.3'te şekil 2.3'e ait semboller gözükmektedir.



Şekil 2.2 : Önde ve Arkada birer serbest tekerlek, ortada servo motor eksenli 2 tane ana tekerlek



Şekil 2.3 : Diferansiyel tekerlek yapısının ötelenmesi

Çizelge 2.3 : Diferansiyel Tekerlek Sisteminde Kullanılan Semboller

$\mathbf{D_n}$	nominal tekerlek çapı
$\mathbf{C_e}$	encoder çözünürlüğü
n	dişli oranı
$\mathbf{\Delta U_L}$	sol tekerlek hız değişimi
$\mathbf{\Delta U_R}$	sağ tekerlek hız değişimi
$\mathbf{\Delta\theta}$	aracın yönündeki açı değişimi
$\mathbf{N_L}$	sol tekerlek encoder değeri farkı
$\mathbf{N_R}$	sağ tekerlek encoder değeri farkı

Araç t zamanından t+1 zamanına ötelenirse kinematik denklemleri şöyle yazılabilir.

$$c_m = \frac{\pi . D_n}{n . C_e} \tag{2.6}$$

$$\begin{bmatrix} dx \\ dy \\ d\theta \end{bmatrix} = \begin{bmatrix} \Delta V . Cos(\theta(t+1)) \\ \Delta V . Sin(\theta(t+1)) \\ \Delta\theta \end{bmatrix} \tag{2.7}$$

$$\left.\begin{aligned} \Delta V_L &= c_m . N_L \\ \Delta V_R &= c_m . N_R \end{aligned}\right\} \Rightarrow \Delta V = \frac{\Delta V_R + \Delta V_L}{2} ; \Delta\theta = \frac{\Delta V_R - \Delta V_L}{L} \tag{2.8}$$

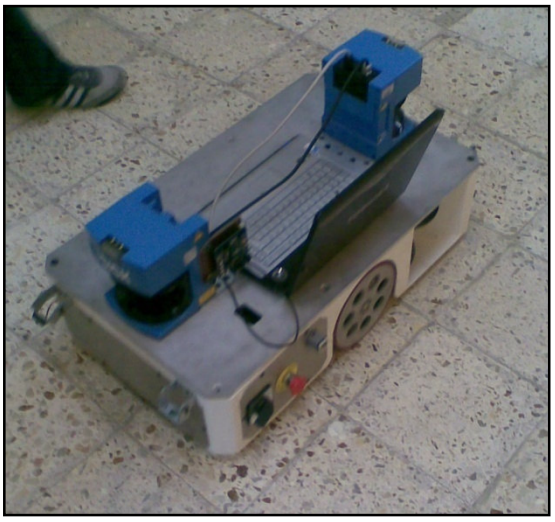
$$\begin{aligned} \Delta V &= r.(\Delta\omega_R + \Delta\omega_L)/2 \\ \Delta\omega &= r.(\Delta\omega_R - \Delta\omega_L)/L \\ \Delta\theta &= \Delta\omega \end{aligned} \tag{2.9}$$

$$\begin{aligned} x_v(t+1) &= x_v(t) + \Delta V . Cos(\theta(t) + \Delta\theta) \\ y_v(t+1) &= y_v(t) + \Delta V . Sin(\theta(t) + \Delta\theta) \\ \theta(t+1) &= \theta(t) + \Delta\theta \end{aligned} \tag{2.10}$$

2.3 Mobil Robottaki Algılayıcılar

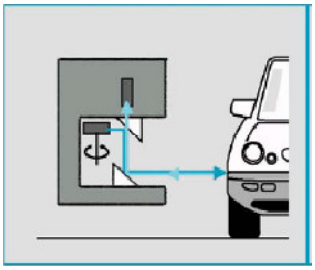
Aracın etrafındaki cisimleri hareket esnasında algılaması için laser tarayıcı kullanıldı. Lazer tipi algılayıcı kullanılmasının sebebi ultrasonik sensörler kadar etrafın gürültüsünden etkilenmemesidir. Kamera gibi görüntü işleme aygıtları hem daha pahalıdır, hem kullanılan ortamlardaki ışık şiddetinin değişimlerinden çabuk etkilenirler, hem de şu anki çalışma konusunun dışında olmasıdır.

İleriki çalışmalar için bu sensörlerden birer tanesi hem aracın önüne hem de arkasına yerleştirildi. Böylece araç geri hareket ettiğinde arka tarafının önü olarak kullanılması mümkün kılınmıştır. Şekil 2.4 üretilmiş SLAM aracının görüntüsü göstermektedir.



Şekil 2.4 : Mobil Robottaki 2 SICK LMS200'ün yerleşimi

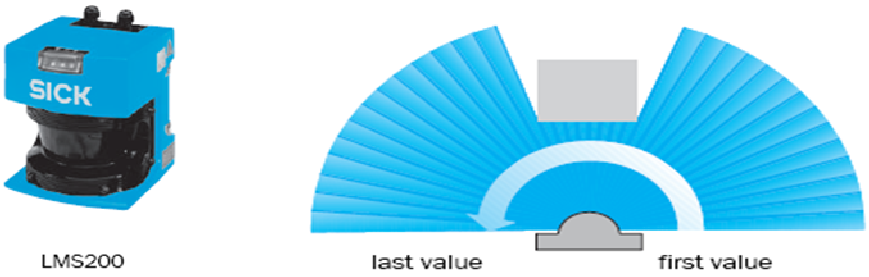
LMS200 tipi lazer tarayıcı LIDAR veya laser radar olarak anılırlar ve uçuş zamanı prensibine göre çalışırlar. Belirli bir zaman aralığında yayılan ışık hedef cisimden yansır ve gönderildiği aynı yoldan geri alınır. Işık gönderildiği sırada bir sayaç çalışmaya başlar ve tekrar ışık alındığında durur. Sayaç değeri yol ile gönderilen yol ilişkilendirilir ev hedef cismin mesafesi ortaya çıkar. Işın geri döndükten sonra lazer tarayıcının bağlı bulunduğu servo motor ilgili derece kadar dönerek yeni mesafe okumasını gerçekleştirir. Işığın gidip gelmesi ışık hızında olduğundan aynanın dönüşünün ölçüme bir etkisi olmaz[8,9].



Şekil 2.5 : Lazer Tarayıcının Çalışmasının yandan görünüşü

2.3.1. Lazer tarayıcının kullanımı

SICK marka LMS200 kapalı alan tipi lazer tarayıcı bu uygulamalardaki popülerliğinden ve haberleşme kolaylığından dolayı tercih edilmiştir. T2000 DSP kontrol kartı ile lazer tarayıcı arasında haberleşme ortamı olarak RS232, telegram adlı haberleşme tipini kullanılmıştır.



Şekil 2.6 : LMS200'ün tarama 13ms'lik 1° aralıkla 180°'lik taraması

Lazer tarayıcı ile 180°'lik bir görüş alanı taranabilmekte ve en fazla 0,25°'lik tarama hassasiyetiyle ölçüm yapılabilmektedir. Çalışmamızda 1°'lik açı hassasiyeti ile 180°'lik görüş alanları taranmıştır. Veri haberleşmesindeki 13ms süren her tarama için 373 byte'lık yükün çok büyük olmasına rağmen sadece belirli noktalarda ölçüm yapılması bu dar boğazın aşılmasında yardımcı olmuştur ve parçacık filtresi simülasyonu da bu yaklaşımla düzenlenmiştir. Aracın DSP kartına yüklenmesi işlemi ise tarayıcı kodlarının MATLAB kodlarının SIMULINK bloklarına çevrilmesiyle mümkün olmuştur. LMS kodları için EKA.1'de yer alan 'LMSMatlab.m' koduna, LMS200 programlaması için 8 numaralı, cihazın teknik özelliklerine ise 9 numaralı kaynaktan ulaşılabilir.

Lazer tarayıcı ile *Sunucu-Müşteri* haberleşme yapısı seçeneği tercih edilmiştir. Tarama modu olrak 8 reflektör seviyeli tarama tercih edilmiştir. Tarama

noktalarında istenen bilgiler lazer tarayıcından telegram kodları ile talep edilir, LMS200 ise tarama sonucunu gönderir.

Description	STX	Address	Length		Command	Data	Checksum	
Byte position	1	2	3	4	5	6	7	8
Hex. value	02	00	02	00	30	01	31	18

Okuma Emri 2 byte

Emir Paketi genişliği 8 byte

Description	STX	Address	Length		Response	Data		Checksum	
						Data	LMS status		
Byte position	1	2	3	4	5	6 to 729	730	731	732
Hex. value	02	80	D6	02	B0	724 bytes	10	15	D4

Cevap 1°'lik aralıklar için 375 byte

Cevap Paketi genişliği 381 byte

Şekil 2.7 : LMS200'e gönderilen bir emir ve karşılığında gelen bir telegram paketi

2.3.2. Lazer tarayıcı modeli

Simülasyonun ve pratik denemelerin başarılı olması açısından ölçüm yaptığımız lazer tarayıcının her algılayıcı gibi ne kadar hata yapıldığının bulunması gerekli idi. Ayrıca parçacık filtresinin içinde yer alan Kalman filtresi de ölçüm hata kovarians matrisine ihtiyaç duyuyordu.

$$R = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} \tag{2.11}$$

Şekil 2.11'deki matris kartezyen koordinatlarda şekil 2.12'deki gibi yazılabilir.

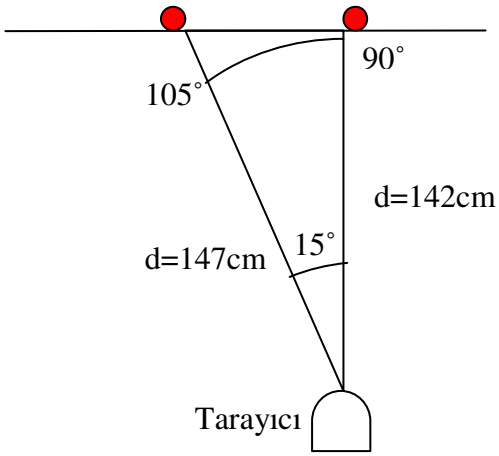
$$R = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{2.12}$$

SICK LMS lazer tarayıcı çalışması ile algılayıcı karakteristiği oluşturularak buradan bir ölçüm hata matrisi R çıkartılmıştır. Yapılan çalışmalar düz bir kapalı

zeminde (kapalı alan sensörü olduğundan dolayı) gerçekleştirilmiştir. Çalıştırılan sensörün konfigürasyonu ektedir.

Reflektör ve lazer tarayıcı sabitlenerek her 2 deney için de 100 tane ölçüm alınmıştır. Uzaklık ölçümleri reflektör lazer tarayıcının lazer ışığının tam karşısına, dik gelecek biçimde 142cm uzaklığa yerleştirilmiştir. Böylece açısal hata dikkate alınmadan ölçümler yapılmıştır. Algılayıcı ışığının tam karşısı 90° kabul edilmektedir. Bütün okumalar 90° yapılmıştır.

Açı ölçümleri sensörü karşılayan 90° 15° fazla olacak biçimde yapılmıştır. Tarayıcının yeri sabit tutulup reflektörün yeri 15° açı yapacak biçimde sabitlenmiştir. Sabitleme kolaylığından dolayı 90°'lik tarama yayı tercih edilmeyip tarama işlemi, aynı eksende, gerçek uzaklığı 147 cm olan bir noktada yapılmıştır. Bu durum Şekil 2.8.de gösterilmiştir.



Şekil 2.8 : SICK LMS200 için test ölçümlerinin elde edilmesi

Lazer tarayıcı karşısına dik olarak gördüğü cisimleri 1.42m'de 1 cm'den daha iyi bir hassasiyetle ölçebilmektedir. Ölçümler sırasında tarayıcının çalışma süresine bağlı olarak ölçüm değerlerinin değiştiği dikkat çekmiştir. Çizelge 2.3'e göre, gerçek uzaklığa yakın ölçümler yaptığı söylenebilir.

Ancak yüksek uzaklıklarda açısal olarak gerçek değeri 15° olması gereken açının ortalama değeri 19,02 derece olarak bulunmuştur. Bu ise uzaktaki 10m'lik objelerin konumlarının saptanmasında büyük sayılabilecek hatalara yol açabilir.

Ayrıca yansıma değeri yüksek olmayan reflektörler belli açıların dışına çıkıldığına tanımlanamamaktadır. Bu tip reflektörler tespit edilirken $90° \pm 20°$'lik açılarda 1m ve üzerinde reflektörler algılamalarında problemler oluşmuştur. Matlab simülasyonunda lazerin algılama mesafesi 180°'lik yayda 1m ile 2m arasında sınırlandırılmıştır; daha uzakta görülen işaretçi nesneler dikkate alınmamıştır.

Laboratuarda yapılan ölçüm sonuçları ise çizelge 2.4 ve 2.5'te verilmiştir.

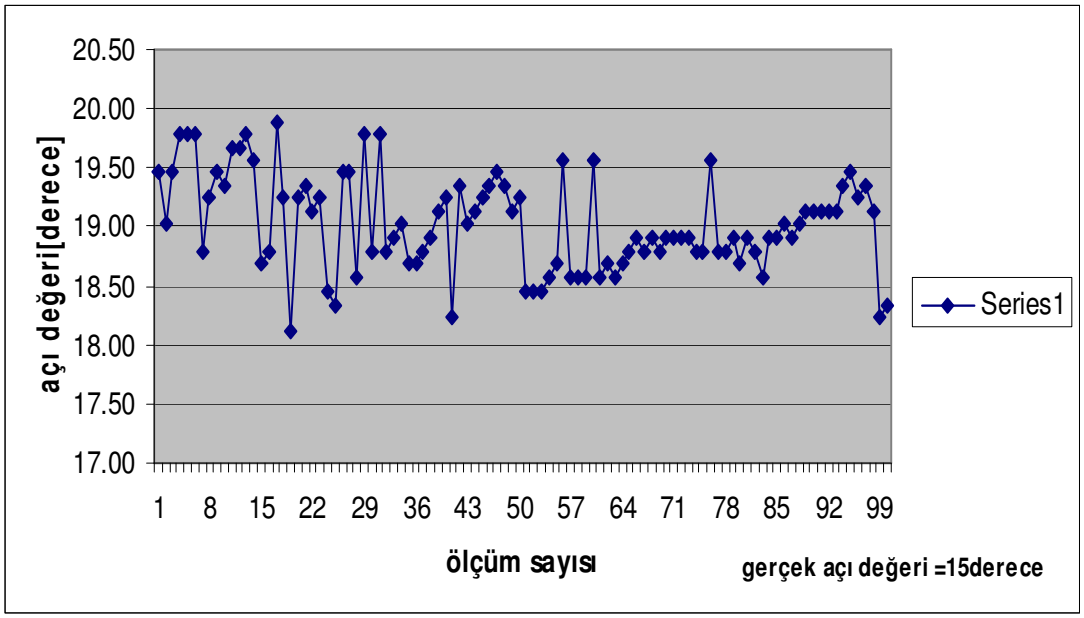
Çizelge 2.4 : Lazer Tarayıcının Açısal Kovariyansının hesaplanması

Gerçek uzaklık d=1470mm				105derecede-90=15 derecelik bir uzaklık			
Ölçüm sayısı	**Ölçülen d[mm]**	**Açı[°]**	**Açı farkı[°]**	**Ölçüm sayısı**	**Ölçülen d[mm]**	**Açı[°]**	**Açı farkı[°]**
1	1506	19.46	4.46	51	1497	18.46	3.46
2	1502	19.02	4.02	52	1497	18.46	3.46
3	1506	19.46	4.46	53	1497	18.46	3.46
4	1509	19.78	4.78	54	1498	18.57	3.57
5	1509	19.78	4.78	55	1499	18.68	3.68
6	1509	19.78	4.78	56	1507	19.56	4.56
7	1500	18.80	3.80	57	1498	18.57	3.57
8	1504	19.24	4.24	58	1498	18.57	3.57
9	1506	19.46	4.46	59	1498	18.57	3.57
10	1505	19.35	4.35	60	1507	19.56	4.56
11	1508	19.67	4.67	61	1498	18.57	3.57
12	1508	19.67	4.67	62	1499	18.68	3.68
13	1509	19.78	4.78	63	1498	18.57	3.57
14	1507	19.56	4.56	64	1499	18.68	3.68
15	1499	18.68	3.68	65	1500	18.80	3.80
16	1500	18.80	3.80	66	1501	18.91	3.91
17	1510	19.88	4.88	67	1500	18.80	3.80
18	1504	19.24	4.24	68	1501	18.91	3.91
19	1494	18.11	3.11	69	1500	18.80	3.80
20	1504	19.24	4.24	70	1501	18.91	3.91
21	1505	19.35	4.35	71	1501	18.91	3.91
22	1503	19.13	4.13	72	1501	18.91	3.91
23	1504	19.24	4.24	73	1501	18.91	3.91
24	1497	18.46	3.46	74	1500	18.80	3.80
25	1496	18.34	3.34	75	1500	18.80	3.80
26	1506	19.46	4.46	76	1507	19.56	4.56
27	1506	19.46	4.46	77	1500	18.80	3.80
28	1498	18.57	3.57	78	1500	18.80	3.80
29	1509	19.78	4.78	79	1501	18.91	3.91
30	1500	18.80	3.80	80	1499	18.68	3.68
31	1509	19.78	4.78	81	1501	18.91	3.91
32	1500	18.80	3.80	82	1500	18.80	3.80
33	1501	18.91	3.91	83	1498	18.57	3.57
34	1502	19.02	4.02	84	1501	18.91	3.91
35	1499	18.68	3.68	85	1501	18.91	3.91
36	1499	18.68	3.68	86	1502	19.02	4.02
37	1500	18.80	3.80	87	1501	18.91	3.91
38	1501	18.91	3.91	88	1502	19.02	4.02

39	1503	19.13	4.13	89	1503	19.13	4.13
40	1504	19.24	4.24	90	1503	19.13	4.13
41	1495	18.23	3.23	91	1503	19.13	4.13
42	1505	19.35	4.35	92	1503	19.13	4.13
43	1502	19.02	4.02	93	1503	19.13	4.13
44	1503	19.13	4.13	94	1505	19.35	4.35
45	1504	19.24	4.24	95	1506	19.46	4.46
46	1505	19.35	4.35	96	1504	19.24	4.24
47	1506	19.46	4.46	97	1505	19.35	4.35
48	1505	19.35	4.35	98	1503	19.13	4.13
49	1503	19.13	4.13	99	1495	18.23	3.23
50	1504	19.24	4.24	100	1496	18.34	3.34

1.kısım	**19.17479°**	2.kısım	**18.87890225°**
	ortalama	**19.02684414°**	

Bu tabloya ilişkin açı değerlerinin değişmesinin grafiği aşağıda verilmiştir.



Şekil 2.9 : Açı değerlerinin grafiksel değişimi

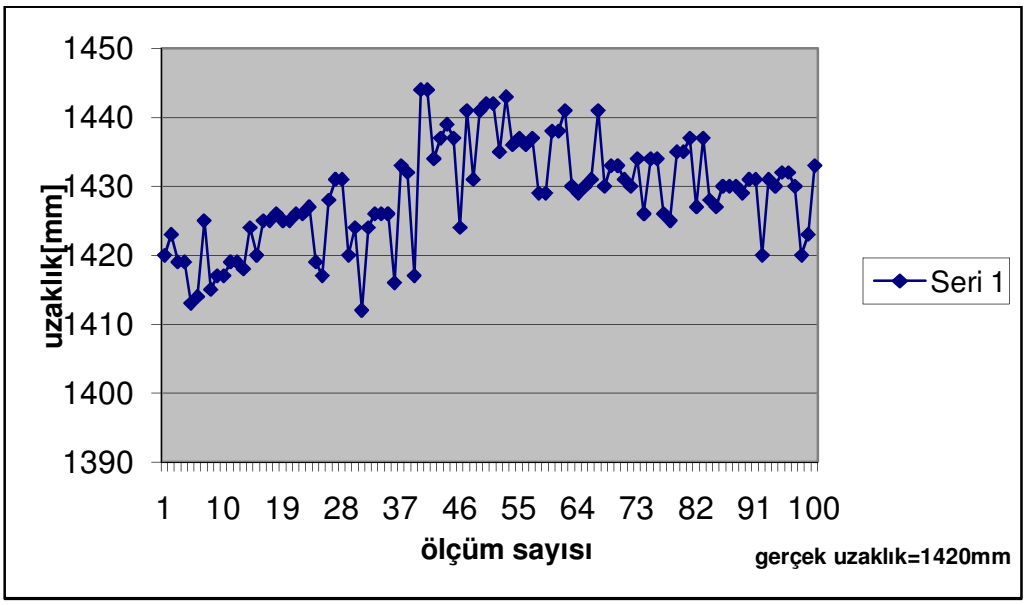
Açının standart sapması 0.40586° derece, variansı 0.16472° olarak bulunmuştur

Çizelge 2.5 : Lazer Tarayıcının Uzaklıksal Kovariyansının Hesaplanması

gerçek uzaklık d= 1420mm					
ölçüm sayısı	**Ölçülen [mm]**	**Fark[mm]**	**ölçüm sayısı**	**Ölçülen [mm]**	**Fark[mm]**
1	1420	0	51	1442	22
2	1423	3	52	1435	15
3	1419	-1	53	1443	23
4	1419	-1	54	1436	16
5	1413	-7	55	1437	17
6	1414	-6	56	1436	16

7	1425	5	57	1437	17
8	1415	-5	58	1429	9
9	1417	-3	59	1429	9
10	1417	-3	60	1438	18
11	1419	-1	61	1438	18
12	1419	-1	62	1441	21
13	1418	-2	63	1430	10
14	1424	4	64	1429	9
15	1420	0	65	1430	10
16	1425	5	66	1431	11
17	1425	5	67	1441	21
18	1426	6	68	1430	10
19	1425	5	69	1433	13
20	1425	5	70	1433	13
21	1426	6	71	1431	11
22	1426	6	72	1430	10
23	1427	7	73	1434	14
24	1419	-1	74	1426	6
25	1417	-3	75	1434	14
26	1428	8	76	1434	14
27	1431	11	77	1426	6
28	1431	11	78	1425	5
29	1420	0	79	1435	15
30	1424	4	80	1435	15
31	1412	-8	81	1437	17
32	1424	4	82	1427	7
33	1426	6	83	1437	17
34	1426	6	84	1428	8
35	1426	6	85	1427	7
36	1416	-4	86	1430	10
37	1433	13	87	1430	10
38	1432	12	88	1430	10
39	1417	-3	89	1429	9
40	1444	24	90	1431	11
41	1444	24	91	1431	11
42	1434	14	92	1420	0
43	1437	17	93	1431	11
44	1439	19	94	1430	10
45	1437	17	95	1432	12
46	1424	4	96	1432	12
47	1441	21	97	1430	10
48	1431	11	98	1420	0
49	1441	21	99	1423	3
50	1442	22	100	1433	13
1.kısım	**1425,66**		2.kısım	**1431,92**	
	ortalama	**1428,79**mm			

Bu tabloya ilişkin ölçülen uzaklık değerlerinin değişmesinin grafiği aşağıda verilmiştir.



Şekil 2.10 : Uzaklık Değerlerinin Grafiksel Değişimi

Uzaklığın standart sapması 0.76082 mm, variansı 0.57885mm^2 olarak bulunmuştur.

Yukarıdaki ölçümlerden hareketle lazer ölçüm görültüsü formül 2.11'deki gibi yazılabilir.

$$R = \begin{bmatrix} 0,58 & 0 \\ 0 & 0,16 \end{bmatrix} \qquad \textbf{(2.13)}$$

Denklem 2.13, RBPF simülasyonunda denklem 2.12'deki gibi kartezyen koordinaatlar ve açı bilgisi ile denklem 2.14'teki gibi yazılmıştır.

$$R = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} \qquad \textbf{(2.14)}$$

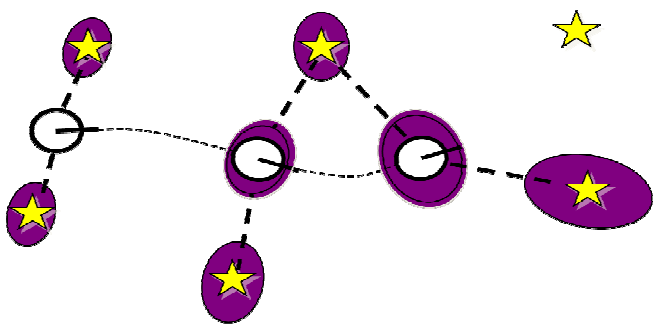
3. SLAM PROBLEMİ

3.1 Amaç

Bu bölümdeki amaç SLAM problemini tanıtmak ve SLAM problemini çözerken karşılaşılan sorunları göstermektir. Bir sonraki bölümde anlatılacak olan filtrelerin SLAM problemini çözerken nerelerde kullanılacağı burada anlatılacaktır.

3.2 SLAM Nedir?

SLAM genel anlamda bir yön bulma problemidir. Bir mobil robotun herhangi bir başlangıç bilgisi olmadan bilinmeyen bir çevrede hareket ettirilmesi ve etrafının haritasının çıkarılmasına dayanır. Ek olarak, mobil robot çıkardığı haritadan faydalanarak yönünü bulacak(lokalizasyon işlemi) ve böylelikle gideceği yolu planlayıp, haritalama süreci boyunca bu yolu kontrol edebilecektir. Şekil 3.1'de bir robotun hareket ederken bulduğu nesneler, kendi yolu ve buna bağlı oluşturduğu belirsizlikler gözükmektedir. Haritasız yön bulma konusundaki robot uygulamaları çok çeşitlidir. Bu uygulamardan SLAM'i ayıran özellik Mars yüzeyinin keşfi, denizlerin tabanın haritalandırılması, felaket bölgeleri gibi uygulama alanlarının insanın giremeyeceği, pratik olmayan veya haberleşmenin çok zor olduğu yerlerde kullanılabilmesidir. Aynı zamanda otononom olarak dayanıklı ve güvenilir yön bulmanın hayattaki kullanım alanları çok geniştir. Otonom yön bulma uzun yıllardır aktif bir araştırma alanıdır[7].



Şekil 3.1 : SLAM probleminin resmedilmesi

SLAM probleminin çözümü robot topluluğunun geçen 10 yılda kazandığı dikkate değer başarılardan biridir. SLAM problemi teorik olarak birkaç farklı şekilde çözüme kavuşmuştur. Bu ise onun kapalı alan, dış mekan , su altı ve havacılık yön bulma uygulamalarında kullanımına sebep olmuştur. SLAM problemi teorik ve öğrenilebilir anlamda çözülmüştür. Ancak pratik olarak hala, bazı alt başlıklarının gerçeklenmesinde ve SLAM ile çok işaretli zengin haritalar oluşturmada problem yaşanmaktadır[10].

SLAM uygulamalarında önceden öğretilmiş,bilinen veya GPS gibi mekan dışı algılama özelliklerinden kullanımına ihtiyaç yoktur. Bir SLAM algoritması hem araç yolunun, hem de çevrenin tutarlı bir haritasını ataletsel, odometrik gibi gürültüye bağlı ve araca bağlı radar, kamera, lazer gibi algılayıcılarla çizer. Göreceli gözlemlerini bulunduğu yere göre yapsa da asıl amaç bütün alanın haritasını kestirmesidir. Kullanılan algılayıcıların sınıflandırılması çizelge 3.1'de verilmiştir.

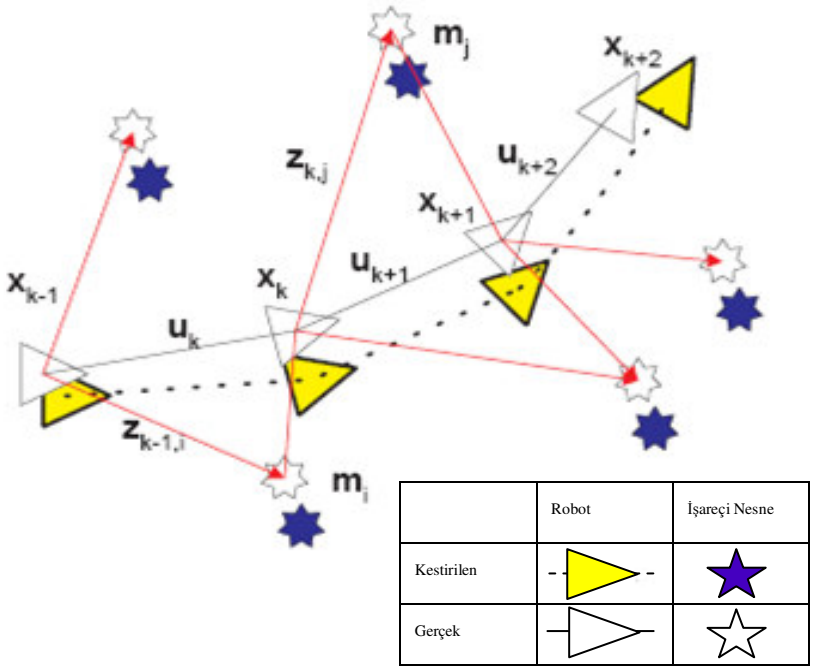
Otonom araç çalışmalarında SLAM'in kullanılması büyük bir öneme sahiptir. Çünkü makine öğrenimini, belirsizliklerin düzenlenmesini, ön bilgi veya yardım olmadan çevreyi algılama ve keşfetmeyi mümkün kılar[10].

Çizelge 3.1 : Haritalama ve Lokalizasyonda kullanılan algılayıcıların sınıflandırılması

	Kullanım Alanları	Kestirim Teknikleri	SLAM Konusunda
Haritalama	Otonom açık alanda maden tetkiki	GPS, Radar, Atalet	LIDAR,RADAR (Uçuş Zamanı), Sonar,ultrasonik (uçuş zamanı), Mono, stereo görme (Kamera), GPS
	Savaş alanı gözlemi	Kamera, GPS, Atalet	
	Yer çatlaklarının belirlenmesi	X-ray	
	Deniz altı petrol yataklarının belirlenmesi	Akustik Vericiler, Atalet	
Lokalizasyon	GPS	Uydu, Uçuş zamanı	Odometri, Jiroskop, Akselometre (ataletsel)
	Müze Rehberliği	Sonar, Lazer Tarayıcı,Kamera	
	Hastane Tedarik sistemi	Radyo işaretleri, lazer kameralar	

3.3 SLAM Probleminin Teoriksel Açıklaması

SLAM problemi herhangi bir ön bilgi olmaksızın robotun hareket ederkenbulunduğu çevrenin haritasını çıkarma ve bu haritadan kendi yerini tespit edebilme sürecidir[10]. Bu süreç sürekli bir biçimde işler. Çevre haritasının çıkarılması için işaretçi nesneler kullanılabilir. Nesneleri herhangi bir haritada keşfedilmeyi bekleyen önemli noktalar veya bölgeler olarak görebiliriz. Şekil 3.2'de SLAM problemi tanıtılmıştır.



Şekil 3.2 : Temel SLAM Problemi

Şekil 3.2'de k anında, mobil robottan tüm algılayıcı bilgileri okuduğumuz varsayalım. x_k aracın bulunduğu noktayı ve yönünü barındıran durum vektörünü, u_k k-1 anında araca uygulanan kontrol vektörünü, m_i zamanla değiştiği kabul edilen işaretçi nesneyi, $z_{k,i}$'de k anında m_i işaretinde alınan gözlem bilgisini ifade eder. Buradan yola çıkarak aşağıdaki tanımları oluşturabiliriz:

- $X_{0:k} = \{x_0, x_1, ..., x_k\} = \{X_{0:k-1}, x_k\}$ Aracın konum bilgileri
- $U_{0:k} = \{u_0, u_1, ..., u_k\} = \{U_{0:k-1}, u_k\}$ Aracın kontrol işaretleri
- $m = \{m_1, m_2, ..., m_n\}$ İşaret kümesi
- $Z_{0:k} = \{z_0, z_1, ..., z_k\} = \{Z_{0:k-1}, z_k\}$ İşaretlerin gözlem kümesi

3.3.1. Bayes teoremi

Thrun'a göre SLAM probleminde olasılık tekniklerinin popülerliğinin sebebi robotlu haritalamanın belirsizlikler ortaya çıkarması ve algılayıcı gürültülerinin işleme dahil olmasıdır[6]. Olasılık tabanlı SLAM algoritmaları algılayıcı gürültü kaynaklarını ve çevreye etkilerini modellerler. Robotla haritalama karmaşık bir problemdir ve robot alanında haritalama en zor algılama konusudur. Böylelikle akademik olarak tek bir bakış açısı ile yaklaşılmıştır ve çözümü için tek bir matematik metodolojisi kullanılmıştır. Her haritalama algoritmasının altında denklem 3.1'de verilmiş basit Bayes teoremi yatar. Benzerlik anlamında Bayes teoremi 3.2'deki gibi yazılır.

$$p(x / z) = \frac{p(z / x)\, p(x)}{p(z)} \tag{3.1}$$

$$p(x / z) = \eta\, p(z / x)\, p(x) \tag{3.2}$$

Bayes teoremine göre, eğer aracın yerinin işaretler bağlı öğrenilmesi istenirse bu 3.2'deki iki terimin çarpılmasıyla mümkündür. Birinci terim olan $p(z / x)$ x'e bağlı gözlemlerin koşullu olasılığıdır. $p(z / x)$ üretici model olarak tanımlanabilir ki bu ise algılayıcı ölçümlerini farklı x durumlarında tanımlar. İkinci terim $p(x)$ aracın öncül olasılığıdır.(thrun) Öncül olasılık, o anda herhangi bir algılayıcı ölçüm değeri alınmadan aracın konumunu gösterir. η ise denklemin sol tarafındaki olasılığı sağlayan normalizasyon katsayısı olarak düşünülebilir.

Robotik haritalamada, böyle değişen verileri kullanmaktaki en belirgin yöntem Bayes Filtrelerinden geçer. Bayes filtreleri Kalman filtreleri, gizli Markov modelleri, dinamik Bayes ağları ve kısmense Markov karar süreçleri ile ilişkilidir. Bayes filtreleri Bayes kuralını kestirim problemlerine genişletir. Bu filtrenin görevi, harita gibi doğrudan gözlemlenemeyen değerlerin soncul olasılık dağılımlarını iteratif kestirimlerle hesaplamaktır. Bu konudaki bilinmeyen durum x_t olarak kabul edilirse, genel Bayes filtresi iteratif olarak x_t 'yi denklem 3.3 ile açıklanır[6].

$$p(x_t \mid z^t, u^t) = \eta\, p(z_t \mid x_t) \int p(x_t \mid u^t, x_{t-1} \mid z^{t-1}, u^{t-1}) dx_{t-1} \tag{3.3}$$

Denklem 3.3 'te t anına kadar olan gözlem ve araç konumlarının kümesi z^t ve x^t olarak belirtilir. $p(x_t \mid z^t, u^t)$ koşullu olasılığının iteratif olarak hep bir önceki adımdaki aynı olasılıktan hesaplandığı görülür. Başlangıç olasılığı t=0 anında $p(x_0 \mid z^0, u^0) = p(x_0)$ olduğu kabul edilir. Denklem 3.3'ün iteratif olduğundan zamanın sabit olarak değiştirilmesi önerilir.

Bayes Filtreleri kullanılırken en can alıcı gereklilik, x^t 'nin sisteme zamanın birçok noktasında etki edecek tüm bilinmeyen algılayıcı ölçüm değerlerini içermesi gerekliliğidir. Robot haritalama bağlamında algılayıcı ölçümlerine etki edecek iki önemli bilinmeyenden bahsedilebilir: Birincisi robotun konumu, ikincisi ise çevrenin haritasıdır. Bundan dolayı, robotla haritalamada probleminde olasılık yöntemleri kullanılırken, hem robot haritasının, hem de robot konumunun birlikte tahmin edilmesi gerekir[6]. Eğer haritanın tanımlanması için *m* değişkenini kullanacak olursak denklem 3.3'ten denklem 3.4 elde edilir.

$$p(x_t, m_t \mid z^t, u^t) = \eta\, p(z_t \mid x_t, m_t) \iint p(x_t, m_t \mid u_t, x_{t-1}, m_{t-1})\, p(x_{t-1}, m_{t-1} \mid z^{t-1}, u^{t-1})\, dx_{t-1} dm_{t-1} \quad \textbf{(3.4)}$$

Birçok haritalama algoritması çevrenin değişmez olduğunu düşünür, bu yüzden m'yi t zamanından bağımsız olarak düşünürler. Bu bağlamda denklem 3.4'ün düzenlenmesi ile denklem 3.5 oluşturulur.

$$p(x_t, m \mid z^t, u^t) = \eta\, p(z_t \mid x_t, m) \int p(x_t \mid u_t, x_{t-1})\, p(x_{t-1}, m \mid z^{t-1}, u^{t-1})\, dx_{t-1} \quad \textbf{(3.5)}$$

Denklem 3.5'deki tahmin denklem 3.4'ten farklı olarak m haritalarında herhangi bir entegrasyon istemez. Böyle bir entegral işleminde çok boyutlu haritalarda zorluklarla karşılaşılır, bundan dolayı denklem 3.5'in pratik uygulamalar için önemi büyüktür.Denklem 3.5 kestirimini kullanmak için denklem 3.6 ve denklem 3.7 olasılıklarının belirtilmesi gerekir.

$$p(z_t \mid x_t, m) \quad \textbf{(3.6)}$$

$$p(x_t \mid x_{t-1}, u_t) \quad \textbf{(3.7)}$$

Zamandan bağımsız olarak düşünülürse bu olasılıklar $p(z \mid x, m)$ ve $p(x \mid x', u)$ olarak yazılabilir. Robotikte $p(z \mid x, m)$ *gözlem modeli* olarak adlandırılır. Çünkü z

gözlemlerinin farklı x robot konumu ve m haritalarından veya içindeki farklı işaretlerden tahmin edildiğini gösterir. $p(x \mid u, x')$ olasılığında ise x robot konumunun u'ya bağlı tahmin edildiğini gösterir ve robotikte *hareket modeli* olarak adlandırılır.

Denklem 3.6 sayısal biçimde kullanılamaz, sebebi koşullu olasılığı bütün haritaların zamanı üzerinde dağılmıştır ve robot konumu da sürekli zaman üzerinde dağılmıştır. Bundan dolayı bütün robotla haritalama algoritmalarında ek kabullere ihtiyaç duyulmaktadır[6].

3.3.2. Olasılıksal SLAM

Eş zamanlı lokalizasyon ve haritalama probleminin aracın ilk durumu ile birlikte k anındaki işaretleri ve aracın konumunu gösteren birleşik olasılık yoğunluk fonksiyonu denklem 3.8'deki gibidir.

$$P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) \tag{3.8}$$

İteratif bir sonuç tercih edilen SLAM probleminde, k-1 anındaki $P(x_{k-1}, m \mid Z_{0:k-1}, U_{0:k-1})$ bileşik olasılık u_k ve z_k işaretleriyle Bayes teoremi ile açıklanır. Bayes filtresinin sonucunda teorik olarak hareket ve gözlem modelleri denklem 3.6 ve 3.7'de belirtilmiştir. Gözlem modelinde eğer robotun konumu ve haritadaki işaretlerin yeri bilinirse gözlemler haritadan ve robotun konumundan koşullu olarak bağımsız düşünülebilir. Hareket modeli ise Markov süreci olarak kabul edilir . x robot konumu u'ya ve bir önceki andaki x'e bağlıdır. Hareket modeli hem gözlemden, hem de haritadan bağımsız olarak düşünülebilir.

SLAM algoritması denklem 3.6 ve 3.7'e bağlı olarak iki ardışıl öngörü(zaman güncellemesi) ve düzeltme(ölçüm güncellemesi) algoritması biçiminde yazılır.

ZamanGüncellemesi

$$p(x_t, m \mid z_{0:t-1}, u_{0:t}, x_0) = \int p(x_t \mid x_{t-1}, u_t)\, p(x_{t-1}, m \mid z_{0:t-1}, u_{0:t-1}, x_0)\, dx_{t-1} \tag{3.9}$$

ÖlçümGüncellemesi

$$P(x_t, m \mid z_{0:t}, x_0) = \frac{P(z_t \mid x_t, m) P(x_t, m \mid z_{0:t-1}, u_{0:t}, x_0)}{P(z_t \mid z_{0:t-1}, u_{0:t})} \tag{3.10}$$

Denklem 3.9 ve 3.10 ile t anına kadar bütün $z_{0:t}$ gözlemlerine ve $u_{0:t}$ kontrol işaretlerine bağlı olarak x_t robot durum vektörü , m işaretçi nesneler için $P(x_t, m \mid z_{0:t}, x_0)$ bileşik koşullu olasılığı bulunur.

Burada dikkat edilmesi gereken nokta, haritalama probleminin koşullu olasılık yoğunluk fonksiyonu $p(m \mid x_{0:t}, z_{0:t}, u_{0:t})$ ile hesaplanabilmesidir. Bu harita hesaplaması aracın ilk konum bilgisi göre bütün t anlarındaki konumlarının bilinmesini içerir. Zıt olarak lokalizasyon probleminin de olasılık yoğunluk fonksiyonu $p(x_t \mid z_{0:t}, u_{0:t}, m)$ ile hesaplanabilmesidir. Hesaplamanın içeriği bütün işaret konumlarının kesin suretle bilinerek bu işaretlere göre aracın lokalizasyonunun tahmin edilmesidir.

3.3.3. Olasılıksal SLAM'in yapısı

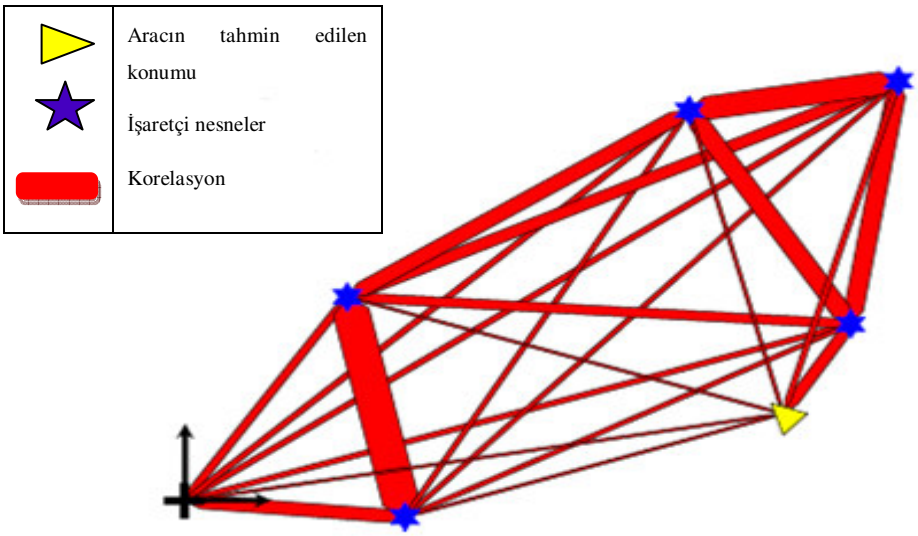
Denklem 3.6'daki gözlem modeli hem araç, hem de işaret konumlarına bağlıdır ve bileşik olasılık yoğunluk fonkisyonunun denklem 3.11'deki gibi yazılamaz.

$$P(x_t, m \mid z_k) \neq P(x_t \mid z_t) P(m \mid z_t) \qquad \textbf{(3.11)}$$

Şekil 3.2 incelenirse SLAM problemindeki hataların büyük kısmının işaretlerin gözlemlendiği sırada mobil robotun nerede olduğunun bilinememesinden kaynaklandığı görülür. Buna rağmen işaretlerin yerinin tahminindeki hatalar birbirleriyle ilişkilidir. Pratik olarak, iki işaret arasındaki bir göreceli konum iyi bir şekilde tahmin edilir. Örnek olarak m_i'nin yeri tam olarak bilinmese de m_i-m_j arasındaki konum iyi olarak tahmin edilebilir. Bu da demek olur ki, $P(m_i)$ değeri az olsa bile, m_i ve m_j işaret çiftinin birleşik olasılık yoğunluk fonksiyonu $P(m_i,m_j)$'nin değeri büyük olur. SLAM'in yapısındaki önemli noktalardan birisi, gözlem yaptıkça işaretlerin korelasyon değerlerinin monotonik olarak artmasıdır[10]. İşaretlerin göreceli yer bilgisi robot hareketinden bağımsız olarak gözlem yaptıkça sürekli güncellenir, işaretlerin birleşik olasılık yoğunluk fonksiyonlarının değerleri büyür.

Bu olayın oluşmasının sebebi, mobil robotun yaptığı gözlemlerin işaretler arası uzaklıktan hemen hemen bağımsız oluşudur. Tam bağımsız olarak düşünülemez, çünkü gözlem hataları aracın gittiği konumlarla ilişkilendirilir. Şekil 3.2'deki mobil araç x_k konumunda iken m_i ve m_j gözlemlerinin gerçekleştirir. Ancak iki işaretin aralarındaki göreceli uzaklık aracın koordinat ekseninden bağımsızdır. Eğer robot

x_{t+1} konumuna gittiğinde yeniden mj'yi gözlemlerse aracın yeri ve işaretin x_k anındaki konumu yeni gözleme göre güncellenir. Bu gözlem aynı zamanda mi işaretinin güncellemesine de yansır, böylece mi işareti aracın yeni konumunda görülmese bile işlemden etkilenir. Güncellemenin etkisi iki işaret arasındaki korelasyon ne kadar fazla ise o kadar fazladır.



Şekil 3.3 : Korelasyonların yay tipinde modellenmesi, korelasyon büyükse yaylar kalın, küçükse yaylar ince gösterilmiştir.

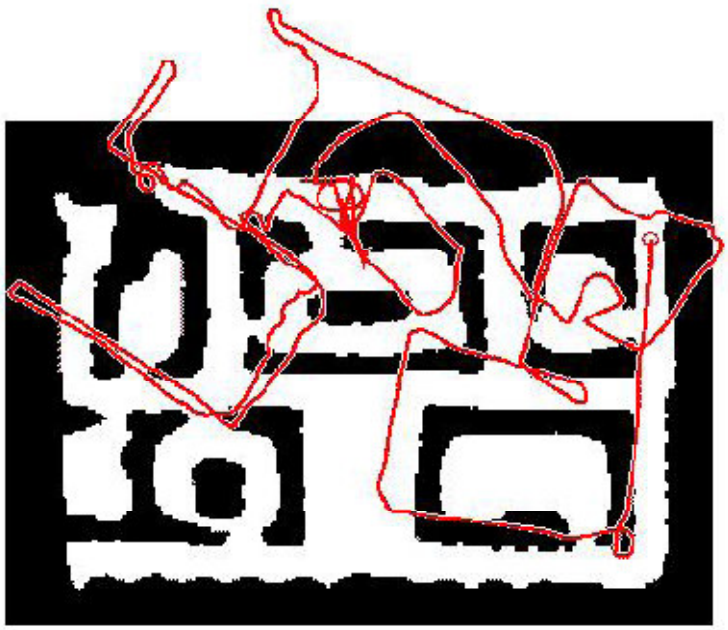
Göreceli ilişkilendirme şekil 3.3'teki gibi yay tipi ağ modelindeki gibi şekillenir. Bu modelin yapısı dinamik Bayes Ağlarına benzer. İşaretler arasındaki komuşuluk ilişkisi(korelasyon) ne kadar fazla ise yaylar daha kalınlaşır. Yayların kalınlaşması aracın aynı haritada dolaşarak yeni gözlemler yapması ile oluşur. Araç yeni bir bölgeyi ziyaret ettiğinde ise yeni işaretlerle önceden gözlemlenmiş işaretlerle olan bağları daha zayıf olacaktır. Tüm harita yaratıldıldığında ise robotun lokalizayonu sadece haritanın iyi bir şekilde oluşturulmuş olmasına ve algılayıcıların göreceli ölçümüne bağlıdır. Sonuç olarak aracın lokalizasyonun kalitesi işaretlerin yerlerinin belirlenme kalitesi ile ilişkilidir ve kalitesinin limiti işaret yerlerinin kalitesi ile ölçülür.

3.4 SLAM Uygulanırken Karşılaşılan Sorunlar

Robotik haritalamın çevre haritasının robot uzayında koordinatlara dökülmesi olduğundan bölüm 2.2 ve 3.2'de bahsedilmişti. Robotun dış dünyayı algılaması için sensörlere ihtiyacı olduğunu biliyoruz. Ancak, her algılayıcı ölçümlerini az veya çok

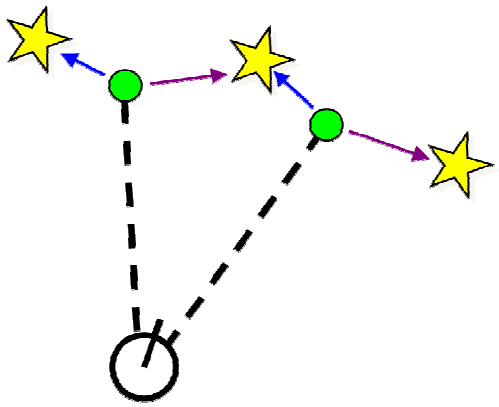
hatalı yapar. Algılayıcı sınıflarından çizelge 3.1'de bahsedilmişti. Ölçüm kaynaklı hatalar ölçüm gürültüsü olarak adlandırılır. Bundan dolayı simülasyonda kullanılacak olan lazer tarayıcıların bölüm 2.3.2'de ölçüm gürültüsü variyansları bulunmuştu. Bunun yanında her algılayıcı uzaklık ve ortam sınırlamalarına sahiptir. Simülasyonumuzda kullanılan SICK LMS200'ün 32 metrelik maksimum ölçüm sınırı uzaklık sınırlamasından mağdur olmamamızı sağlamıştır. Örnek olarak, ışık ve ses duvarları geçemediği gibi mobil robotun kapalı alanlarda rahat olarak harita oluşturmasını engeller, kameralar ortamın ışık şiddetinden etkilenir. Robot hareketi de hataları beraberinde getirir(şekil 3.4). Özellikle açık çevrim kontrol algoritmaları robotun konumunun belirlenmesinde hatalara sebep olur.

Karşılaşılan ilk ana problem, ölçüm gürültüsünden kaynaklanır. Eğer haritalamada yapılan ölçümler birbirlerinden istatiksel olarak bağımsızsa ölçüm gürültüsünü çözmek basittir. Aynı noktada birden çok ölçüm alınması gürültüyü azaltmada yararlı olur. Ancak robotik haritalamada ölçüm gürültüleri istatiksel olarak bağımlıdır; ölçüm hataları birikerek algılayıcı zaman ve ölçüm güncellemesine(3.9 , 3.10) etki eder. Gürültülere göre tahmin modellerinin teorisini ve uygulamalarını her zaman karmaşıklaştırır[6]. Sonuçta robotu çevreyi gözlemlemesi sırasında aldığı her ölçümde sistematik, ilişkili hatalara rastlanır. Ölçüm gürültülerini bilerek haritalama yapmak, robotik haritalamanın anahtarıdır.



Şekil 3.4 : Odometri Hatasının giderilememesi

Karşılaşılan ikinci problem haritalanacak mekanların çok boyutlu olmasından kaynaklanır. Çok boyutluluk problemi, haritalanmış bir mekanın gerçek mekana benzemesi için mobil robotun kaç farklı mekanda gezdiğini anlaması ile eşdeğerdir. Küçük oda, büyük oda, koridor gibi tanımlamalar 2 boyutlu ufak kapalı mekanlar için harita çıkarmaya yeteilir. Ancak iç içe geçmiş mekanlar, dış alanlar veya okyanus yüzeyi gibi ortamların 2 veya 3 boyutlu haritalamaları bir çok tanımlama ve tahmin gerektirir.



Şekil 3.5 : Veri eşleştirme sorunu, aracın yaptığı gözlemlerin hangi fiziksel cisimlere ait olduğunu tanımlayamamasından kaynaklanır.

En zor problem ise *veri eşleştirme* problemidir. İlişkilendirme sorusu farklı zamanlarda alınmış gözlemlerin gerçek dünyada aynı fiziksel cismin üzerine gelip gelmediğinin sorusunu arar. Özellikle karmaşık ve kapalı alanlarda problem olabilir. Bu durumda robotun 3 temel durumundan bahsedilir[5]:

- Robot bilinmeyen bir çevrede geziyordur.
- Robot bulunduğu çevreyi gezmeye tamamlayıp döngüsünü tamamlamıştır.
- Robot bulunduğu çevrede yeni bir döngüye başlamıştır.

Bu üç durum mobil robot tarafından tanımlanabilirse veri eşleştirme problemini aşmak kolaylaşır, yoksa robotun konumu ile bir çok hipotez oluşur, zamanla ekponasiyel olarak artar. Sonuçta robotun konum hatası büyür.

Dördüncü olarak çevrenin dinamik değişiminden söz edilebilir. Bazı değişimler yavaş, bazıları hızlı gerçekleşir. Örnek olarak bir kapının açılması veya daha önce robot olan bir yolu kapatması normalden farklı algılayıcı bilgilerinin oluşmasına sebep olur ve veri eşleştirme problemini doğurur. Çevrenin değişmesi sadece 2 boyutlu olarak algılanmamalı üçüncü boyut da düşünülmelidir. Bundan dolayı

robotun yerden yüksekliği veya genişliği gibi ayrıntılar önem kazanır. Şu ana kadar dinamik çevreleri haritalamaya yarayan bir algoritma yoktur, ancak haritalamada sadece robotun zamanla değişen bir yolu olduğu ve arta kalan her şeyin bir gürültü olarak kabul edildiğinde, oluşan dinamik değişimin zamanla değişen bir gürültü olduğu düşünülebilir.

Beşinci zorluk ise robotun haritadaki keşif yöntemleridir. Tam olarak oluşturulmuş veya yapılan simülayondaki gibi önceden belirlenenen bir yolda robotun hareket etmesi kolaydır, ancak tam haritası tamamlanmamış ve bilinmedik bir ortamda robotun etrafı keşfetmesi ek bir zorluktur. Böylece keşfetme yolları ve bunları uygulama bir planlama problemidir. Düzgün bir haritanın oluşması için nereye hareket edileceğinin seçiminin fiyatı zaman ve enerji olarak düşünülür. Yanlış seçimlerde robot yolunu da kaybedebilir, enerjisi de bitebilir. Bu tercihler gerçek zamanlı yapılacağından dolayı ise bir çok haritalama algoritmasının uygulanmasını engeller.

Son zorluk olarak hesaplama yöntemlerinden bahsedilebilir. Haritalama yapmak büyük veri tabanlarının yaratılmasını ve bu veri tabanlarının gerekli algoritamalarla işlenmesini gerektirir. Gelişen teknoloji işlem zamanlarında çok yardımcı olsa da büyük çevrelerin haritalanması büyük işlem zamanları demektir. Gerçek zamanlı olarak düşünülürse, matematiksel işlemler CPU'da fazla zaman alır. CPU bu işlemlerle uğraşırken robotun anlık ihtiyacı olan verileri sağlayamamakta ve haritların yanlış oluşmalarına sebebiyet vermektedir.

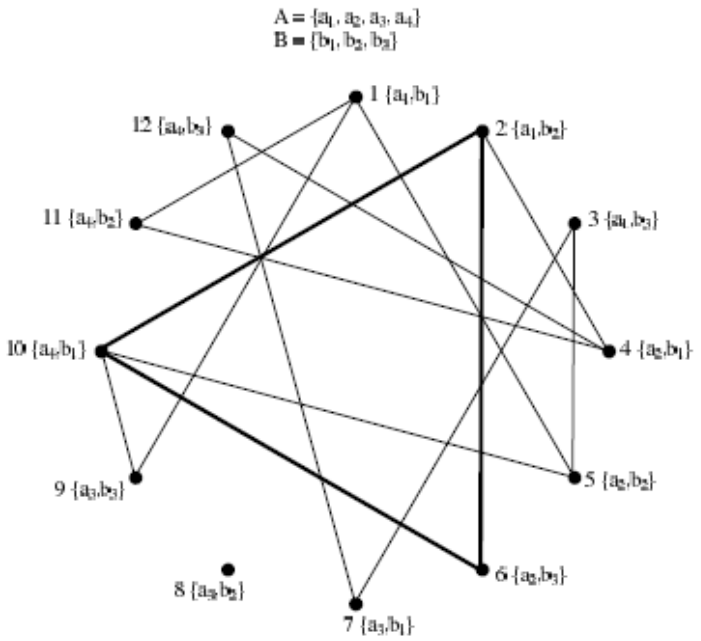
3.5 Veri Eşleştirmesinde Kullanılan Yöntemler

Veri eşleştirmesinin SLAM'ı çözerken karşılaşılan sorunlardaki en zoru olduğundan bir önceki bölümde bahsedilmiştir. Bir noktanın yanlış eşleşmesi bütün bir haritanın bozulmasına sebep olabilir. Rao-Blackwellised Parçacık filtresinin kullandığı *Çok hipotezli veri eşleştirme* yönteminin pratik olarak uygulaması ilgili bölümde açıklanacak olup giriş anlamındaki anıtımı ve diğer genel yöntemler aşağıda açıklanacaktır. İlk önerilen SLAM algoritmalarında veri eşleştirmesi, *tekil veri toplama* yöntemini içerir. Bu yöntem yeni ölçülen yöntemlerin teker teker ölçülen eski verilerle belirli bir hata payı ile karşılaştırılmasından ibarettir. Ancak aracın belirsizliği çok farklı olursa, okunan gözlemlerin sürekli eskileriyle karşılaştırılması güvenilir bir yöntem değildir. Öte yandan, yapılan parçacık filtrelerinin

simülasyonunda aracın göreceli yolu önceden belirlenmiş olduğundan dolayı aracın belirsizliklerin atanan partiküllere eş değer olduğu hatırlatılmalıdır. Bu sebepten ve tekil veri toplamanın basitliğinden dolayı, çoklu veri eşletirmesinin yanında simülasyonda tekil olarak gözlemlerin eşleştirilmesine de başvurulmuştur. Tekil veri eşleştirmesine simülasyonların anlatımı sırasında yer verilecektir.

3.5.1. Grup doğrulama

Hemen hemen bütün veri eşleştirme algoritmaları hedef izleme yöndemlerindeki gibi istatiksel doğrulama kullanır[11]. Grup doğrulamada geliştirilen önemli bir yöntem de çoklu eşleştirmelerin dikkate alındığı grup doğrulama yöntemidir. Bir ağaç tipi arama yöntemi olan birleşik uyumlu dallanma ve bağlanma(JCBB); graf yöntemine dayananansa tümleşik sınırlı veri eşleştirme (CCDA) iki bilinen yoludur. Daha sonra aracın yer bilgisinin bilinmemesine dayanan rastgele JCBB yöntemi de geliştirilmiştir.



Şekil 3.6 : CCDA yakınlık graflarında uçları uygun eşleştirmeleri, kalın çizgili üçgen ise karşılıklı olarak bağlı eşleştirmeleri gösterir.

Eşleştirme ilişkileri darsa, grup doğrulama kendine başına güvenilir bir veri eşleştirmesi sağlar. Yanlış bir işaretin eşleştirilmesi sırasında oluşan hatalarının etkisi küçük olacaktır. Ancak geniş alanların eşleştirilmesinde çok hipotezli veri eşleştirme gibi daha karmaşık eşleştirme algoritmalarına ihtiyaç duyulur.

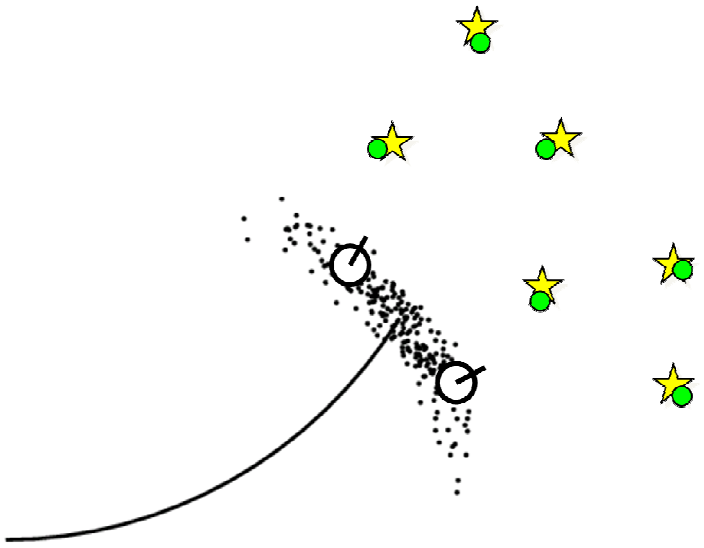
3.5.2. Görünüm tanıma

Sadece geometriye dayalı izler bulma veri ilişkilendirme de güvenilir değildir. Görüntü gibi birçok algılama modeli cismin yüzeyi, şekli ve rengi hakkında bilgi sahibi olmamızı sağlar, bu artılar da veri ilişkilendirme probleminde kullanılabilecek özelliklerdir. SLAM'da görünüm tanıma veri ilişkilendirme, döngü tamamlama, veri toplama süreçlerinde ek bir araç olarak kullanılabilir.

İlk olarak görünüm tanıma ve görüntü benzerliği ölçütleri görüntü veritabanına adresleme yapmak ve topolojik haritadaki bölgeleri tanımlamak için kullanılırdı[11]. Daha sonra, SLAM'da döngü tamamlanmasının tanımlanmasında kullanıldı. Newman SLAM çalışmasında iki önemli yenilikten söz eder[11]. Görüntü ölçüleri tek görüntü yerine birçok görüntü üzerinden işlenir. İkinci olarak öz değerler ortak benzerliklerin kaldırılmasında kullanılmıştır. Böylece, aranan eş görüntüler yerine, hatalı pozitiflerin bulunması önemli ölçüde azaltılmıştır.

3.5.3. Çok hipotezli veri ilişkilendirmesi

Çok hipotezli veri ilişkilendirmesi bir haritada işaretlerin dağınık olarak yer alması durumunda dayanıklı bir şekilde işaretleri gözlemlemeye yarar. Her hipotez için aynı veya farklı biçimlerde robot yolu öngörüsü oluşturularak ilişkilendirme hataları çözülür. Hipotezin sayısı hesaplama hızı, işlem hacmi, enerji gibi problemlerle sınırlıdır. Bu problemlerin üstesinden gelmek, az benzerliği olan hipotezlerin silinmesi ile giderilir.



Şekil 3.7 : Çok hipotezli veri eşleştirmesinde bir gözlem anındaki robot konumları

Örnek olarak bir robotun programı bölüm 3.4'te anlatıldığı farklı veri eşleştirme tiplerine sahip olabilir. Robot döngünün kapanmasından şüphelendiğinde veya benzer işaretler algıladığında farklı veri ilişkilendirme algoritmaları uygular. Bu tip algoritmalar SLAM sorununa ancak yeni uygulanmaya başlamıştır ve şu anki genel yönelim her hipotezin oluşturduğu tahmini haritaların ayrı saklanması yönündedir[11]. Yapılan simülasyonda kullanılan FastSLAM yöntemi birçok haritaya sahiptir ve çoklu hipotez yöntemini temel alır. Ek bir veri ilişkilendirme algoritması olarak tekil veri eşleştirmesi kullanılır. FastSLAM'in avantajı her parçacık için veri ilişkilendirme yapmasındadır ki, özelliklerinde parçacık filtresi bölümünde bahsedilecektir.

3.6 SLAM Problemine Çözümler

SLAM probleminin çözümü denklem 3.6'da verilen hareket modeline ve denklem 3.7'de verilen gözlem modeline çözümler bulmaktan geçer. Bulunan çözümler denklem 3.9'daki zaman ve denklem 3.10'daki ölçüm güncellemesi hesaplamalarını tutarlı bir şekilde gerçekleştirebilmelidir. Verilen çözümlerden şu ana kadar en genel olanı ek Gauss gürültüsü ile durum uzayı modelini bize sunan GKF'dir. GKF'ye alternatif olarak hareket modelini çoklu hipotezler olarak gören ve bu hipotezlerin oluşturulmasında gauss tipi olmayan dağılım kullanan parçacık filtresi (FastSLAM) görülebilir. İki önemli çözüm filtresinden de parçacık filtreleri bölümünde bahsedilecektir.

4. PARÇACIK FİLTRESİ

4.1 Amaç

Anlatılan SLAM problemine bir çözüm olarak tezde parçacık filtresi çözümü öne sürülmüştür. İlk olarak filtrenin tanımı yapılacaktır, daha sonra kullanılan RBPF'de kullanılan Kalman Filtresi anlatılacaktır. Parçacık filtresi tanımı ve daha sonra parçacık filtresinin özelleştirilmiş bir çeşidi olan RBPF anlatılacaktır. Parçacık filtresi ile ilgili yapılmış simülasyon ve parçacık filtresinin nasıl kullanıldığı bu bölümde açıklancak, parçacık filtresinin SLAM problemindeki başarısı çizelge 1.1'e göre gösterilecektir. Ayrıca, incelenen filtrenin yapılmakta olan pratik uygulamada nasıl kullanılacağı anlatılacaktır.

4.2 Filtrelerin SLAM Probleminde Kullanımı

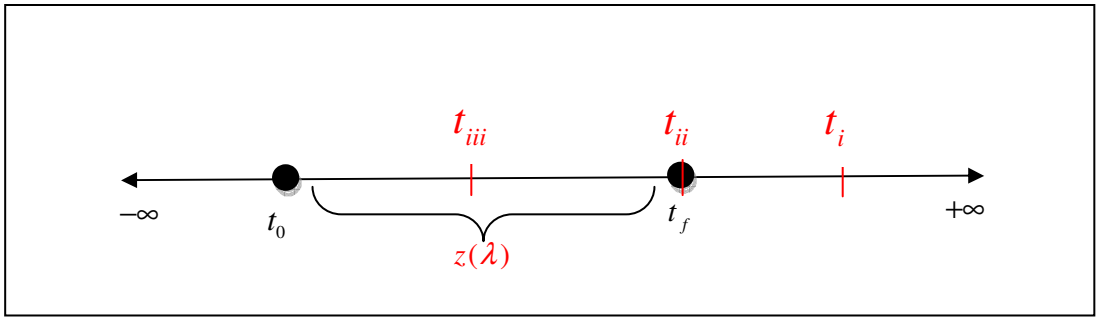
Haritalamada kullanılan bütün durum algoritmalarının olasılıksal olma gibi ortak bir özellikleri vardır. Hepsi, mobil robotun ve çevrenin olasılıksal modellerini uygulayarak algılayıcı ölçmelerini haritalamada kullanırlar. Kullanılan bazı yöntemler olasılıksal gözükmese de hepsine uygun önermelerle olasılıksal olarak bakılabilir[6].

Analog haberleşme, radarla hedef izleme,model tanıma gibi birçok fiziksel durum çevreden toplanan verilerin işlenmesi ve sonucunda tahmin edilmesine dayanır[12]. Tahmindeki hatayı azaltmak için alınan veriler için filtre uygulanır.Çünkü SLAM sürecindeki gözlemin amacı gürültülü ortamdaki verilerden en iyi tahmini çıkarmaktır. Başka bir deyişle, $\lambda \in I$ ve $I = [t_0, t_f]$ önermeleriyle sürekli veya ayrık zamanda y(t) tahminini toplanan $z(\lambda)$ verilerinden oluşturmak istersek t zamanı'nın değerine göre şekil 4.1'de gösterilen üç belirgin problemle karşılaşırız:

i. $t > t_f$ ise, öngörü veya ekstrapolasyon problemi vardır

ii. $t = t_f$ ise, filtreleme problemi vardır.

iii. $t_0 \le t < t_f$ ise, uydurma(smoothing) veya interpolasyon problemi vardır.



Şekil 4.1 : Gözlem doğrusu

Bölüm 3.3.1'de anlatılan Bayes Teoremine bağlı ardışıl filtre uygulamalarında yukarıdaki üç problem de filtre yardımıyla ile çözülür.

4.3 Kalman Filtresi

Kalman Filtresi öngörü, düzeltme tipi kesitiricileri içinde barındıran matematik denklemler kümesidir. Ön şartlar belirtildiği takdirde, tahmin edilen hata kovariansını azalttığından dolayı optimal bir çözüm kabul edilir. Çıkışı ile birlikte Kalman Filtresi özellikle otonom ve yardımcı yön bulma da geniş bir uygulama ve araştırma alanına sahip olmuştur. Filtrenin sayısal hesaplamalarda kullanılabilmesi kendisini pratik kılmıştır, ancak bu pratiklik filtrenin basit ve dayanıklı yapısından ileri gelir. Gerçek dünyada, nadiren optimalliği engelleyen şartlar oluşsa da, filtre birçok uygulamada iyi sonuçlar vermektedir[13].

Genişletilmiş Kalman filteresinin temel hali Kalman filtresidir. Kalman filtresinin kestirimde kullanılabilmesi için araç ve gözlem modellerinin lineer olması gerekmektedir. Fakat gerçek hayatta bu modeller lineer olmayan biçimdedir. Bunun için araştırmacılar lineer olmayan modeller için Kalman Filtresinin kullanabilirliğini araştırmışlardır ve bu modelleri lineer hale getirerek Kalman Filtresinin lineer olmayan mo7deller içinde kestirim metodu olarak kullanılabileceğini bulmuşlardır. Kalman filtresi ardışıl Bayes kestirim metoduna benzer. Kalman filtresi ve Genişletilmiş Kalman Filtresi'nin temeldeki yapısı ve mantığı aynıdır. Kalman filtresi formülleri içerisinde sadece küçük bir değişiklik yapılarak Genişletilmiş Kalman filtresi(GKF) ortaya atılmıştır. Ancak, öncelikle lineer Kalman filtresinin yapısı GKF'ye giriş anlamında takip eden bölümde verilmiştir[14].

4.3.1. Ayrık doğrusal Kalman filtresi

Ayrık Kalman filtresi, doğrusal stokastik fark denklemi ile kontrol edilen, denklem 4.1'de verilmiş ayrık zamanlı kontrol prosesinin durumunu denklem 4.2'deki gözlem ile tahmin etmeye çalışır. Dinamik bir sistemin durumunu bütün sistemin geçmişini tanımlayan en ufak vektör demektir. Gürültüsüz bir şekilde ele alındığında sistem tanımı deterministik olrak sisteminin geleceğini öngörür.

$$x_t = Ax_{t-1} + Bu_t + w_{t-1} \quad , x \in \Re^n \tag{4.1}$$

$$z_t = Hx_t + v_t \quad , z \in \Re^m \tag{4.2}$$

w_t rastgele değişkeni proses gürültüsünü, v_t rastgele değişkeni ise ölçüm gürültsünü simgelerler. İkisi de birbirinden bağımsız, beyaz ve normal dağılımlı gürültülerdir ve sırasıyla denklem 4.3 ve 4.4' teki gibi gösterilirler.

$$p(w) \sim N(0, Q) \tag{4.3}$$

$$p(v) \sim N(0, R) \tag{4.4}$$

Pratikte Q ve R matrislerinin uygulama süresince değiştiği gözlense de Q ve R matrisi için sabit matris değerleri alınmıştır. R matrisi için denklem 2.12'de hesaplanan SICK LMS200'ün karakteristiği kullanılmıştır. Denklem 4.1'deki A matrisi nxn boyutlu olup durumu t zamanı t-1 zamanına bağlar. A her adımda değişebilir. B matrisi nx1 boyutlu olup optimal kontrol girişini x durumuna bağlar. Denklem 4.2'deki H matrisi x_t durumuna z_t gözlemine bağlar. H her adımda değişebilir. $x_{t|t-1}$, $x_{t|t-1} \in \Re^n$ olarak bir önceki adıma bağlı olarak öncül olasılık olsa; $x_{t|t}$, $x_{t|t} \in \Re^n$ olarak bir t anındaki z_t gözlemlerine bağlı soncul olasılık olsa öncül ve soncul hata kestirimleri sırasıyla denklem 4.5 ve 4.6'daki gibi yazılır.

$$e_{t|t-1} \equiv x_t - x_{t|t-1} \tag{4.5}$$

$$e_t \equiv x_t - x_{t|t} \tag{4.6}$$

Öncül ve soncul tahmin hatası kovariyansı denklem 4.7 ve 4.8'deki gibidir.

$$P_{t|t-1} = E[e_{t|t-1} e_{t|t-1}^T] \tag{4.7}$$

$$P_{t|t} = E[e_t e_t^T] \tag{4.8}$$

Şu anki amaç, $x_{t|t}$ soncul olasılığının $x_{t|t-1}$ öncül olasılığı, z_t gözlemi ile ölçüm öngörüsü $Hx_{t|t-1}$ arasındaki farka bağlı hesaplanmasını sağlayan bir denklem bulmaktır. Durum soncul olasılığı, Bayes kuralına göre $x_{t|t-1}$ durum öncül olasılığının bütün önceki z_t gözlemlerine bağlı olması ile açıklanır ve denklem 4.9'daki gibi yazılır.

$$x_{t|t} = x_{t|t-1} + K(z_t - Hx_{t|t-1}) \quad \textbf{(4.9)}$$

Denklem 4.9'da yazılı olan fark ifadesi $(z_t - Hx_{t|t-1})$ inovasyon veya kalan olarak adlandırılır. İnovasyon z_t ile $Hx_{t|t-1}$ arasındaki uyumsuzluğu yansıtır; fark sıfır ise öngürü ile ölçüm aynıdır. nxm boyutlu K matrisi Kalman kazancı olarak adlandırılır ve soncul hata kovariyansını ufaltır. K matrisi denklem 4.6 ve 4.8'e bağlı olarak denklem 4.10'daki gibi yazılır.

$$K = P_{t|t-1}H^T(HP_{t|t-1}H^T + R)^{-1}$$
$$K = \frac{P_{t|t-1}H^T}{HP_{t|t-1}H^T + R} \quad \textbf{(4.10)}$$

Ölçüm gürültüsü sıfıra yaklaşırken K daha fazla hissedilir, denklem 4.11'deki gibi yazılır. Hata kovariyansı $P_{t|t-1}$ sıfıra yaklaşırken K daha az hissedilir[13].(denklem 4.12)

$$\lim_{R\to 0} K = H^{-1} \quad \textbf{(4.11)}$$

$$\lim_{P_{t|t-1}\to 0} K = 0 \quad \textbf{(4.12)}$$

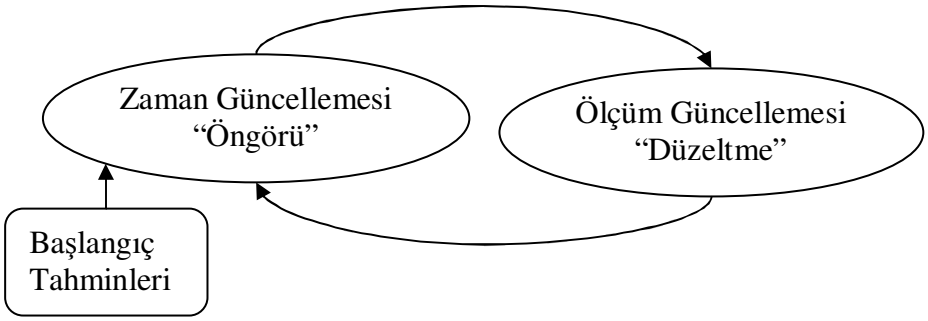
Sonuçta Kalman filtresi olasılıksal olarak, durum dağılımının ilk iki momenti içerir(Denklem 4.13-14). Soncul durum tahmin denklemi 4.9, beklenen değerin ilk momentini içerir. Soncul hata kovariyansı ise durum dağılımının variyansını gösterir(Denklem 4.15).

$$x_{t|t-1} = E[x_t] \quad \textbf{(4.13)}$$

$$P_t = E[(x_t - x_{t|t-1})(x_t - x_{t|t-1})^T] \quad \textbf{(4.14)}$$

$$P(x_k \mid z_k) \sim N(E[x_k], E[(x_t - x_{t|t-1})(x_t - x_{t|t-1})^T]) \quad \textbf{(4.15)}$$

Kalman filtresinin algoritma olarak kullanılması ise sürecin şekil 4.2'de gösterildiği üzere geri besleme ile kontrol edilmesine eşdeğerdir. Filtre durumu tahmin eder ve belirli bir zaman sonra algılayıcılardan ölçüm geri beslemesi alır. SLAM'in Denklem 3.9 ve 3.10'dakine benzer bir algoritma grubu oluşur ve bu iki grup sırası ile zaman güncellemesi ve ölçüm güncellemesi olarak adlandırılırlar. Zaman güncellemesi için öngörü, ölçüm güncellemesi için düzeltme adları da kullanılır.



Şekil 4.2 : Kalman Filtresi Algoritması

Çizelge 4.1 : Kalman Filtresi Algoritması Denklemleri

Zaman Güncellemesi(Öngörü)	
$x_{t\|t-1} = Ax_{t-1} + Bu_t$	**(4.16)**
$P_{t\|t-1} = AP_{t-1}A^T + Q$	**(4.17)**
Başlangıç Tahminleri	
x_{t-1}, P_{t-1}	
Ölçüm Güncellemesi (Düzeltme)	
$K = P_{t\|t-1}H^T(HP_{t\|t-1}H^T + R)^{-1}$	**(4.18)**
$x_t = x_{t\|t-1} + K(z_t - Hx_{t\|t-1})$	**(4.19)**
$P_t = (I - KH)P_{t\|t-1}$	**(4.20)**

Çizelge 4.1'te t-1 ve t zamanı arasındaki denklem ilişkileri gözükmektedir. Zaman güncellemesindeki A ve B matrisleri denklem 4.1'den gelir; aynı şekilde Q denklem 4.3'den gelir Düzeltmedeki amaç denklem 4.18'deki Kalman kazancını hesaplamaktır. Sonraki adım algılayıcıdan gözlem bilgilerini alarak denklem 4.9,19'daki soncul durum tahminini bulmaktır.

Her tahmin ve düzeltme adımında süreç bir önceki soncul tahminlerle devam eder. Bu ardışıl davranış Kalman filtresinin özelliklerindendir. Bu ardışıl davranışta o anki tahmin geçmişteki bütün tahminlere dayanarak bulunur.

4.3.2. Genişletilmiş Kalman filtresi(GKF)

Kalman filtresi üç basit kabule dayanır. İlk olarak, bir sonraki durum denklemi, ki bu SLAM probleminde mobil robotun hareket modelidir Gauss gürültüsü eklenmiş bir biçimde doğrusal olmalıdır. İkinci olarak aynı doğrusallık gözlem modelinde de olmalıdır. Üçüncü olarak başlangıç belirsizliği gauss tipinde olmalıdır. Ayrık doğrusal Kalman filtresinde süreç, denklem 4.1'deki lineer stokastik fark denklemi ile yönetilmiştir. Ardışıl algoritma ise doğrusal filtredeki gibi, şekil 4.2'dekinin aynısıdır.

Bir doğrusal tahmin fonksiyonunda robotun hareketi x_t ve harita m_t doğrusal olarak x_{t-1} , m_{t-1} ve u_t 'ye bağlıdır. Kabullere bakarsak harita değişmemelidir. Ama dinamik olarak harita değişebilir. Ayrıca robotun konumu x_t genel olarak lineer olmayan trigonometrik fonsiyonlar ile oluşturulur. Bu fonksiyonlar ise doğrusal olmayan biçimde x_{t-1} ve u_t 'ye bağlıdır. Böyle doğrusal olmayan durumları engellemek için Kalman filtresinde robotun hareket modelinin doğrusal fonksiyonu Taylor serilerine açılımla doğrusal olmayan modele yakınsanır[6].

Teorik olarak göstermek gerekirse, denklem 4.21 ve 4.22'de aracın hareket ve gözlem modeli doğrusal olmayan stokastik fark denklemleri biçiminde gösterilmiştir. Ancak w_{t-1} ve v_t 'nin her adım değiştiği düşünülmelidir ve bunlar her adımda ölçülemez. Bundan dolayı, hareket ve gözlem modeli denklem 4.22 ve 4.23'teki gibi gürültüler olmadan yakınsanabilir.

$$x_t = f(x_{t-1}, u_t, w_{t-1}) \quad \textbf{(4.21)}$$

$$z_t = h(x_t, v_t) \quad \textbf{(4.22)}$$

$$\widetilde{x}_t = f(x_{t-1}, u_t, 0) \quad \textbf{(4.23)}$$

$$\widetilde{z}_t = h(\widetilde{x}_t, 0) \quad \textbf{(4.24)}$$

Doğrusal olmayan denklemler olan 4.23 ve 4.24, doğrusal denklemler 4.1 ve 4.2'deki gibi aşağıda yazılmıştır.

$$x_t = \widetilde{x}_t + A(x_{t-1} - x_{t|t-1}) + Ww_{t-1} \quad \textbf{(4.25)}$$

$$z_t = \widetilde{z}_t + H(x_t - \widetilde{x}_t) + Vv_t \quad \textbf{(4.26)}$$

Sonuç olarak, yakınsama işlemi ile doğrusal olan Kalman filtresinden farklı olarak robot hareket modeli $\widetilde{x}_t$ ve gözlem modeli $\widetilde{z}_t$ yakınsatılmış biçimlerde yer almaktadır[13]; süreç ve ölçme gürültüleri matris olarak aynı kalmıştır. Ek olarak ilişkilendirme matrisleri kismi türevli jakobiyen matrisleri biçiminde çizelge 4.2'deki gibi yazılmaktadır. Her adımda değişmesine rağmen basitlik açısından jakobiyen kısmi türevlerin matrislerinde *t* zaman indisi kullanılmamıştır.

Çizelge 4.2 : GKF' de kullanılan Jakobiyen Matris Tablosu

İsim	Jakobiyen denklemler	Simülasyon Gösterimi	Denklem no
Araç konum jakobiyeni	$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[f]}}(x_{t-1}, u_t, 0)$	∇F_x	**(4.27)**
Süreç gürültüsü jakobiyeni	$W_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[f]}}(x_{t-1}, u_t, 0)$	∇F_v	**(4.28)**
Gözlem jakobiyeni	$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[f]}}(\widetilde{x}_t, 0)$	∇H_x	**(4.29)**
Ölçüm gürültüsü jakobiyeni	$V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[f]}}(\widetilde{x}_t, 0)$	-	**(4.30)**

Çizelge 4.1'dekine benzer olarak GKF algoritmaları çizelge 4.4'te verilmiştir. Çizelge 4.3'te jakobiyen değişkenlerin açık matris hali gözükmektedir.

Çizelge 4.3 : Jakobiyen kısmi türev matrislerinin açık hali

$$\nabla F_x = \frac{\partial f}{\partial x} = \underbrace{\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix}}_{\hat{x}(t-1|t-1)\text{ de hesaplanan}} \qquad \nabla F_v = \frac{\partial f}{\partial u} = \underbrace{\begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{bmatrix}}_{u(t)'dahesaplanan}$$

$$\nabla H_x = \frac{\partial h}{\partial x} = \underbrace{\begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_m} \\ \vdots & \cdots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \cdots & \frac{\partial h_n}{\partial x_m} \end{bmatrix}}_{\hat{x}(t|t-1)\text{ de hesaplanan}}$$

Çizelge 4.4 : GKF Algoritması Denklemleri

Zaman Güncellemesi(Öngörü)

$$\widetilde{x_t} = f(x_{t-1}, u_t, 0) \qquad \textbf{(4.34)}$$

$$P_{t|t-1} = A_t P_{t-1} A_t^T + W_t Q_{t-1} W_t^T \qquad \textbf{(4.35)}$$

Başlangıç Tahminleri

$$x_{t-1}, P_{t-1}$$

Ölçüm Güncellemesi (Düzeltme)

$$K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + V_t R V_t^T)^{-1} \qquad \textbf{(4.36)}$$

$$x_t = x_{t|t-1} + K_t (z_t - h(\widetilde{x_t}, 0)) \qquad \textbf{(4.37)}$$

$$P_t = (I - K_t H_t) P_{t|t-1} \qquad \textbf{(4.38)}$$

Yukarıdaki denklemlerle simülasyonda kullanılan denklem takımı aynı olsa da, proje grubunun Tübitak'a sunulan rapor doğrultusunda GKF formüllerinin sembollerinde bazı farklılıklar olmuştur. Ayrıca okunanan makalelerde RBPF filtrelerindeki GKF denklemleri de dikkate alınarak çizelge 4.6'daki denklemler oluşturulmuş ve kullanılmıştır.

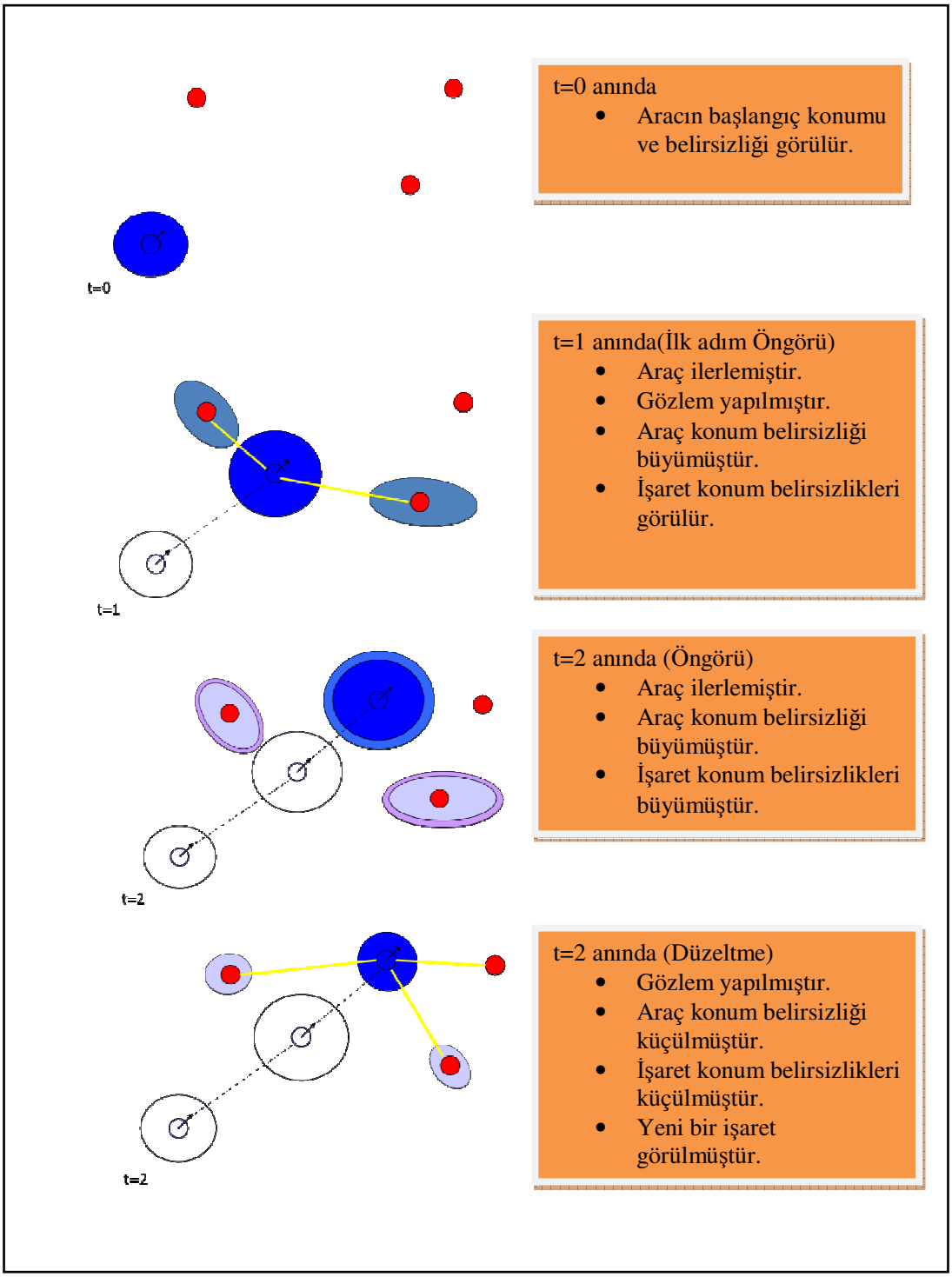
4.3.3. GKF'nin SLAM problemindeki yeri

SLAM ilk olarak ortaya atıldığında , herkesin üzerinde uzlaştığı nokta Kalman filtreli bir çözüm getirilmesi olmuştu. Şekil3.2'deki SLAM problemini ele alan Şekil 4.3 Kalman filtresinin SLAM probleminde kullanılmasını göstermiştir.

Araç t=0 anından t=1 anına ilerlediğinde gözlem yapar , kendi belirsizliğini ve işaretlerin konumunu öngörür. Burada belirsizliklerin değişimini görebiliriz. İlk adımda sadece gözlem yapar ve ilerler. Bundan sonraki adımlarda hem gözlem hem de düzeltme yapılmalıdır. t=2 anında araç yeniden bulunduğu konumu ve hafızasındaki işaretleri öngörür; daha sonra düzeltme için algılayıcı gözlemlerine ihtiyaç duyar. Böylece kendi konum belirsizliğini ve işaretlerin konum belirsizliklerini düzeltir.

GKF'de pratik olarak kullanımda, haritadaki işaretlerin sayısı birkaç düzine ile birkaç yüz arasında değişir. GKF'de en uzun süren işlem matris çarpım ve ters alma işlemleridir. Haritadaki işaret sayısı m olarak alınırsa algılayıcının gözlemlerini yenileyip tek bir 'yeni' işaretçi nesne bulmasında bile kendi kovariyans matrisinin yeniden oluşturulması için $O(m^2)$ 'lik elemanın algoritmada yeniden işlenmesi gerekir[1]. Fiziksel çevrede işaretlerin sayısı bilinemez, işaretçi nesne sayısı her gözlemde dinamik olarak artar, birkaç yüz işaretçi nesneyi aşan çevrelerde veri eşleştirme problemi de düşünülürse süreçte karmaşıklığı arttıran bir durumdur. Az işaretli bölgeler için, bir bölgede birkaç kere dolaşıldığında yeni görülen nesneler için aday işaret listesi oluşuturulabilir. Bu liste de başka bir Kalman filtresi ile kontrol edilir. Sürekli aynı yerde işaret bu listeye girerse işaretin benzerliği kontrol edilir ve orada bir işaret vardır mıdır , yoksa bu işaret hatalı bir ölçme midir anlaşılır[6].

GKF 'nin güçlü tarafı robotun konumlarından ve işaretçi nesnelerin yerinden, robotun kendi konumunun koşullu olasılığını gerçek zamanlı olarak çok iyi bir şekilde tahmin edebilmesidir. Bunu ancak gaussian metodları içinde barından yöntemler yapabilir. Ancak zayıf tarafı, mobil robotun hareket modelinin çok iyi yapılmış olması gerektirir ve algılayıcı gürültülerinin çok iyi bilinmesini gerektirir.



Şekil 4.3 : GKF SLAM'in t=0,1,2 anları için işlemesi

GKF diğer SLAM çözümleri ile karşılaştırıldığında Thrun'un hazırladığı yayındaki[6] tabloya göre GKF'nin yakınsaması yüksek, gerçek zamanlı olarak işaret tahmini yapabilen bölgesel minimallerin olmadığı, artımsal bir algortimaya sahip olan, mutlak robot konumunun bilinmesine gerek olmayan, gaussian algılayıcı gürültüsüne sahip, işaretçi sayısı 1000'i geçmeyen çevrelerde kullanılan, veri

ilişkilendirmeyi ek bir yöntem kullanmadan yapamayan, saf algılayıcı ölçümlerini kullanamayan, dinamik çevrede kullanımı kısıtlı bir filtredir. Yakınsamasının diğer metodlardan yüksek olması, her işareti her adımda görmeye ihtiyaç duymamasından gelir. GKF'yi en fazla sınırlayan nokta, gürültülerin gaussian biçimde olmasıdır. İşaret algılayıcı gürültüleri algoritmadan bağımsız olarak değişirler. Bundan dolayı Kalman filtreleri ölçümleri işaretler ile ilişkilendirme problemi ile baş edemezler.

4.4 Parçacık Filtreleri ve SLAM Problemindeki Yeri

Tahmin algoritmalarını kabaca içerdiği çalışma mantıklarına göre sınıflayabiliriz. Bunların içlerinde en popülerleri şu anda bu çalışmada yer alan genişletilmiş GKF, maksimum benzerlik teknikleri ve parçacık filtreleridir[2].

GKF 'nin öne çıkan güçlü tarafının robotun konumundan ve işaretçi nesnelerin yerinden, robotun kendi konumunun koşullu olasılığını çok iyi bir şekilde tahmin edebilmesi olduğundan, ancak zayıf tarafının, algılayıcı gürültülerinin gauss tipinde olması çok ve işaretçi nesne sayısını 'm' olarak alırsak, algılayıcının gözlemlerini yenileyip tek bir 'yeni' işaretçi nesne bulmasında bile kendi kovariyans matrisinin yeniden oluşturulması için $O\left(m^2\right)$'lik elemanın algoritmada yeniden işlenmesi gerekliliğinden ve veri eşleştirme sorunundan bölüm 4.3.3'te bahsedilmişti.

Kullanılan Maksimum Benzerlik Algortimaları sensör okumalarına bakarak gerçeğe en yakın haritayı oluşturmaya çalışır ve ilişki kümeleri yaratarak robotun yerini kestirir. Gutmann bu algoritmayı artımsal bir biçimde kullanarak robotun gezdiği döngüleri doğru bir biçimde tahmin etmeyi başarmıştır. Hahnel bu algoritmada veri ilişkilendirme kullanmıştır. Lakin, bu algoritma ile kullanılan veri ilişkilendirme ağaçları gerçek zamanlı çalışmalara izin vermemektedir. Ancak çoklu hipotezlerdeki benzerliklerin bulunmasında parçacık filtrelerinde yer alan maksimum benzerlik algoritması kullanılmıştır.

Montemerlo tarafından önerilen FastSLAM algoritması ardışıl olasılıksal SLAM'da mantıksal bir atlama yaratmıştır. Çünkü, doksanlı yıllarda kullanılan GKF'nin tek hipotezden oluşan lineer gauss tipi algoritmalarının tersine, FastSLAM ardışıl Monte Carlo tekniklerini kullanır ve çok hipotezli bir yapı ihtiva eder. Bu yapı gauss tipi olmayan süreçleri ve dağılımlarını içermektedir. Ayrıca çok hipotezli olması

algoritmayı kendiliğinden veri eşleştirmesi yapan bir model kılmıştır. GKF gibi parçacık filtreleri de Bayes kuralı temellidir.

Murphy'nin çalışmasında SLAM problemini çözmek için etkili bir yol olan RBPF önerilmiştir. Bu çalışma, Montemerlo tarafından farklı işaretçi nesne haritaları kullanılarak genişletilmiştir[2]. Şu anda da popüler olarak üzerinde geliştirmeler yapılmaktadır.

Veri eşleşim algoritmalarından yararlanılmadan FastSLAM için n'yi parçacık sayısı alırsak; Montemerlo algoritma zamanını O(nm) zamana indirebilmiş ve gerekli veri eşleştirme algoritmaları ile O(nlogm) zamanına ulaşmıştır[1].

4.5 Parçacık Filtresinin Teorik Olarak Açıklanması

Görüldüğü üzere RBPF filtreleri hem GKF, hem maksimum benzerlik algoritması, hem de Monte Carlo Lokalizasyonu(MCL) algoritmalarını içermektedir. Tezin konusu parçacık filtresinin SLAM problemine uygulanması olduğundan parçacık filtresinin içinde kullanılan algoritmalardan bahsedilecektir. Daha sonra da bu filtrenin bir uygulamaya ve simülasyonda kullanılan RBPF'ye nasıl dönüştürüldüğünden bahsedilecektir.

Parçacık filtresinin tanımını yapmak gerekirse, bu filtre parçacıkların dağılımlarını örnekleyen bir ardışıl Monte Carlo tekniğidir[1,15]. Gauss dağılımlı olmayan ve doğrusal olmayan süreçlerin tahmin edilmesinde kullanılır. Kalman filtresindeki amaç çevredeki işaretlerin yerini tahmin etmektir,bu tahminlere bağlı olaraksa mobil robotun yerini tahmin etmektir. Oysa parçacık filtresinin amacı, mobil robotun geçeceği olası yolları tahmin etmektir. Ortam büyüklüğüne uygun parçacık sayısı ile optimal Bayes kestirimine yaklaşılır. Parçacık yapısından gerekli kavramlar verildikten sonra bahsedilecektir.

4.5.1. Monte Carlo Lokalizasyonu

SLAM problemindeki mobil robotun yeri bilinmediğinden dolayı parçacık filtresini kullanırken aracın yerinin tahmin edilmesine ihtiyaç vardır. Tahmin etme işlemi GKF'den farklı olarak tek bir aracın belirsizliğini bulmak yerine o bölgedeki araç için bir çok yer bulunur. Araç yeri parçacık sayısı ile sınırlıdır. Bu yerlerin

bulunması için bir öneri dağılımı oluşturulur. Bu öneri dağılımı da MCL ile gerçekleştirilir.

MCL aslında parçacık filtresi ve GKF gibi robot konumunu algılayıcı okumalarından tahmin eden bir algoritmadır. MCL'nin anafikri aracın olasılıksal lokalizasyonunu parçacıklara veya *inançlara* göre tahmin etmesidir.(roboust mcl) İnanç terimi Thrun, Fox, Burgard ve Dellaert[15] tarafından yazılarında kullanılmaktadır. Başka bir değişle soncul olasılıkları Kalman Filtresi veya Markov Lokalizasyonundaki parametrik formlara yakınsamak yerine, MCL her adımda robotun gözlemlerine bağlı kümeler oluşturur[15,16]. Bu örnek kümeler, robotun geçmişteki konumları dikkate alınarak robotun konumları için oluşturulan öneri dağılımından oluşturulur. MCL basit olarak soncul olasılıkları örnek kümelerin ağırlıkları ile gösterir. İstatikte parçacık filtresi, görüntü işlemede condensation algoritması olarak geçer[15].

Lokalizasyon bağlamında bakıldığında, parçacık tanımını diğer yaklaşımlardan ayıran bazı karakteristikleri vardır.

i. Parçacık filtreleri hareket dinamikleri ve gürültü dağılımı algılayıcı dinamiklerini içinde barındırabilir[15].

ii. Parçacık filtreleri genel anlamda yoğunluk yakınsayıcılarıdır. Bunun sebebi, gauss gürültüsü ve gauss tipi başlangıç belirsizliği gibi zorlayıcı kabullere gerek duymamasıdır[15].

iii. Parçacık filtreleri oransal örneklemeden, soncul benzerliğe kadar hesaplama algoritmalarına odaklanır. Çok çeşitli hesaplamalar esnek olarak kullanılabilir[17].

iv. Parçacık sayısı gerçek zamanlı olarak değişebildiğinden mobil robot kaynaklarına uygun çalışan bir filtredir[15].

v. SLAM problemine uygulaması kolaydır, bundan dolayı birçok araştırma grubu tarafından benimsenmiştir[15,17].

Ancak bazı eksileri de vardır. Örnek sayısının azlığı filtrein kötü çalışmasına sebep olacaktır, böylece yeri belli olan bir robot yerini MCL'nin düzgün örnek üretememesinden dolayı kaybedebilir. Çok kaliteli bir gürültüsü az olan algılayıcılar

örneklerin arasında seçim yapma şansını azaltırlar. Bu sorunlara çözüm olarak çok hipotezli Kalman filtresi yapıları veya örnek sayısını arttırma önerilir[15].

MCL'nin teorisi denklem 3.3'teki ardışıl Bayes filtresine dayanır. Bayes filtresi kısmi gözlemlenebilir Markov zincirinden oluşan mobil robotun x_t konumunu algılayıcı gözlemlerinden tahmin eder. Kaynak 15'e göre parçacık denklem 4.39'a göre yazılır.

$$\mathrm{B}el(x_t) = p(x_t \mid z_t, u_{0:t}) \tag{4.39}$$

Bayes Kuralını $\mathrm{B}el(x_t)$ ile yazılabilir.

$$\eta = p(z_t \mid u_{0:t-1})^{-1} \tag{4.40}$$

$$\mathrm{B}el(x_t) = \eta\, p(z_t \mid x_t, u_{0:t-1})\, p(x_t \mid u_{0:t-1}) \tag{4.41}$$

Markov kabulu ile denklem 4.42 yazılabilir.

$$p(z_t \mid x_t, u_{0:t-1}) = p(z_t \mid x_t) \tag{4.42}$$

Denklem 4.41 ve 4.42 birleştirilip açılırsa denklem 4.43 yazılır.

$$\mathrm{B}el(x_t) = \eta\, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_{0:t-1}) p(x_{t-1} \mid u_{0:t-1}) dx_{t-1} \tag{4.43}$$

Yine Markov kabulune göre denklem 4.44 yazılır.

$$p(x_t \mid x_{t-1}, u_{0:t-1}) = p(x_t \mid x_{t-1}, u_{t-1}) \tag{4.44}$$

Denklem 4.44 ve 4.39 denklem 4.43'te yerlerine yazılır.

$$\mathrm{B}el(x_t) = \eta\, p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_{t-1}) \mathrm{B}el(x_{t-1}) dx_{t-1} \tag{4.45}$$

Aynen denklem 3.5'teki gibi, iteratif Bayes denklemi denklem 4.45'deki gibi yazılır. Karşılaştırma için denklem 3.5 denklem 4.46'da yeniden verilmiştir.

$$p(x_t, m \mid z^t, u^t) = \eta\, p(z_t \mid x_t, m) \int p(x_t \mid u_t, x_{t-1})\, p(x_{t-1}, m \mid z^{t-1}, u^{t-1}) dx_{t-1} \tag{4.46}$$

İteratif MCL algoritması 3 ana adımda gerçekleşir.

i. Örnekleme: Oluşturulmuş bütün örnekler robotun bulunabileceği konumları gösterir. Robotun bulunduğu eski yerlerden yeni yerler türetilir. Başlangıç için simülasyonda robota bir kare içinde rastgele noktalar atanmıştır.

Başlangıç olasılıkları her parçacık için eşittir; n tane parçacık s için denklem 4.47'de başlangıç olasılığı verilmiştir. Denklem 4.48 ise *i* parçacık numarası olmak üzere standart parçacıkğı ifade eder.

$$\mathrm{B}el(x_0) = \frac{1}{n} \tag{4.47}$$

$$x_{t-1}^{(i)} \sim \mathrm{B}el(x_{t-1}) \tag{4.48}$$

ii. Önem ağırlıklarının hesaplanması: Her yeni örneğin ağırlığı hesaplanır. Amaç lazer tarayıcının işaretleri, parçacığın bulunduğu yerden nasıl gördüğü ve buna bağlı olarak bulunduğu yerin konumunu tahmin etmektir.Denklem 4.49'da öneri dağılımı aşağıdaki gibi ifade edilir.

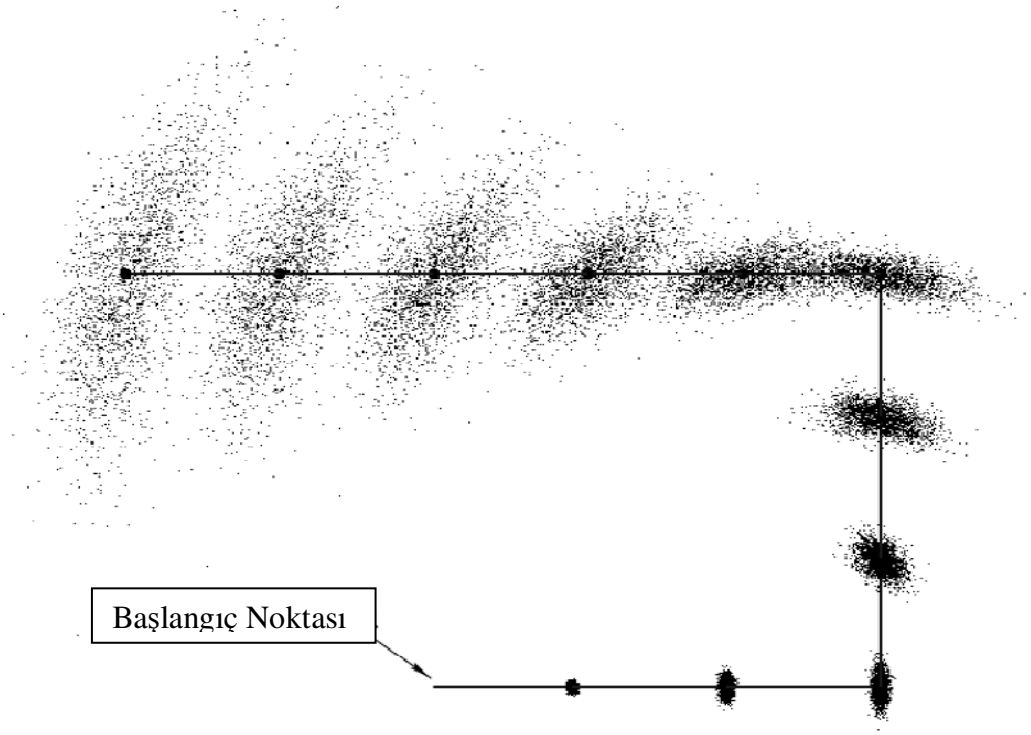
$$w_t^{(i)} = \frac{hedefdağılım}{öneridağılımı} = \frac{p(x)}{q(x)} \tag{4.49}$$

$$x_t^{(i)} \sim p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1}) \tag{4.50}$$

$$q_t^{(i)} = p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1}) \times Bel(x_{t-1}^{(i)}) \tag{4.51}$$

$w_t^{(i)}$'nin ardışıl olarak çeşitli yöntemlerle hesaplanması ardışıl önem örneklemesi(SIR) olarak adlandırılır. Öneri dağılımı SIR'da q(x) olarak geçer. Öneri dağılımı olarak adlandırılan denklem 4.49 ve 4.51'de bulunan q(x)'in amacı istenen soncul olasılığı denklem 4.45'e göre vermektir. Ancak sadece öneridir, soncul olasılığa eşit değildir.

iii. Yeniden örnekleme: Önem ağırlıkları arasında normalizyondan sonra seçim yapılır, düşük önem ağırlıklarına sahip örnekler yok olur; büyük önem ağırlıklarına sahip örnekler bir sonraki adımda kullanılmak üzere kopyalanır. Bu işlem genelde ağırlık çarkı denen yöntemle yapılır. Bir rulet çarkı gibi çarka yerleştirilen değerler önem ağırlıklarına yer kaplarlar, parçacık sayısı N kadar çevrilen bu çarktan yeni adıma geçmesi gereken N tane örnek toplanır. Susan Fox'a göre olasılıkla yönetilen rastgelelik MCL'nin en önemli noktasıdır[16]. Yapılan RBPF simülasyonunda yeniden örnekleme için farklı yöntemler önerilmiştir. Bu yöntemlere daha sonra değinilecektir. Yeniden örnekleme filtrenin güncellenebilir olmasını sağlayan en önemli adımdır.



Şekil 4.4 : Sadece odometriye bağlı yapılmış MCL. Noktalar farklı adımlardaki parçacıkları gösterir.

4.5.2. Maksimum benzerlik kestirimi

Yapılan z gözlemlerinin x durumuna bağlı olduğunu SLAM probleminden bilinmektedir. Ancak algılayıcıların düzgün ölçüm yapamaması ve ortamdaki gürültünün varlığı gözlemleri etkilemektedir. Hem belirsizliklerin hem de gözlemlerin içinde barınabileceği bir benzerlik fonksiyonu denklem 4.52'de verilmiştir[7].

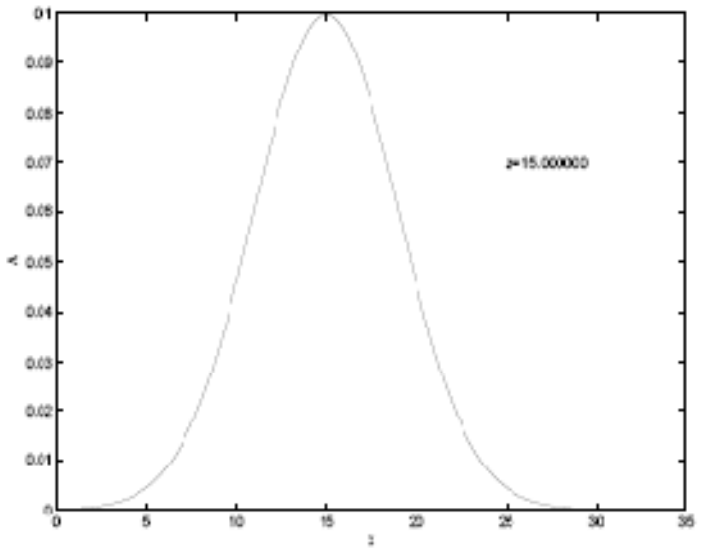
$$L \triangleq p(z \mid x) \tag{4.52}$$

Denklem 4.52'deki dağılım z gözlemlerinin x durumlarına bağlı koşullu olasılığını ifade eder. Şekil 4.5'de ve denklem 4.53'te gauss tipli böyle bir dağılım görülmektedir. C normal sabitidir.

$$p(z \mid x) = \frac{1}{C} e^{-\frac{1}{2}(z-x)^T P^{-1}(z-x)} \tag{4.53}$$

Denklem 4.53'teki gibi L olasılık yoğunluk fonksiyonu ve z gözlemlerinden benzerliği maksimum kılan x değerini bulmak için kullanılacak maksimum benzerlik kestiricisi $\hat{x}_{m.l}$ denklem 4.54'te verilmiştir.

$$\hat{x}_{m.l} = \arg\max_{x} p(z \mid x) \qquad \textbf{(4.54)}$$



Şekil 4.5 : z skaler gözlemleri ile oluşturulmuş gauss tipi benzerlik fonksiyonu

Yapılan simülasyonda, maksimum benzerlik kestirminden parçacıklarda atanmış ağırlık değerleri arasında en yüksek olanının seçilmesinde yararlanılmıştır. Değeri en yüksek olan önem ağırlıklı parçacığın RBPF'de gösterilecek yeniden örnekleme adımında, bozulmuş önem ağırlıklı parçacıkların yerine kopyalanmasında kullanılmıştır.

4.6 Rao-Blackwellised Parçacık Filtresi

Rao-Blackwellisation Monte Carlo algoritmalarında kullanılan bir variyans azaltma tekniğidir. Birleşik olasılık yoğunluk fonksiyonunun simülasyon uzayı böylece azaltılmış olur.

Monte Carlo tekniğinde amaç bölüm 4.5.1'de belirtildiği üzere çok boyutlu bir uzayda verilmiş bir olasılık fonsiyonu ile yapılan gözlemlerden örnek kümeler bulmaktır. Bu ise Parçacık Filtrelerinin temelini oluşturur.

$x_1, ..., x_N$ 'den kadar olan rastlantı değişkenlerinin emprik kestirimi denklem 4.55'de şöyle yazılabilir.

$$E(f(x) \mid p) = \int f(x)\, p(x) dx \tag{4.55}$$

Monte Carlo emprik kestrimi ise denklem 4.56'de yazılmıştır.

$$\widehat{E_N}(f) = \int f(x) dp_N(x) = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{4.56}$$

Denklem 4.57'deki tanıma göre, sadece $p(x_2 \mid x_1)$ analitik olarak hesaplanabilirse simülasyon denklem 4.58'deki gibi $p(x_1)$ uzayı ile sınırlandırılabilir ve denklem 4.59 mümkün olur. Böylece denklem 4.60'daki gibi marjinal kestirimler toplama ile bulunur. Denklem 4.60 denklem 4.56'ya benzemektedir.

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{4.57}$$

$$x_1^{(i)} \sim p(x_1) \tag{4.58}$$

$$p(x) = p(x_2 \mid x_1)\, p(x_1) \tag{4.59}$$

$$p(x_2) = \frac{1}{N} \sum_{i=1}^{N} p(x_2 \mid x_1^{(i)}) \tag{4.60}$$

Sabit sayıdaki örnekler için, Rao-Blackwellised örneklemesi ile bulunan rastgele variyans her zaman uzayın tümünün örneklenmesinden daha küçük veya eşittir[17].

Rao-Blackwellisation MCL'ye uygulanarak parçacık filtresinin örnek uzayının $[x_t, m]^T$ 'den sadece aracın konumu olan x_t 'ye inmesini sağlamıştır. Robot yolu ağırlıklı örneklerden oluşur ve harita analitik olarak hesaplanır. Her parça

birbirinden bağımsız gauss tipi dağılım gösterirler. Parçacık filtresinin bölüm 4.5.1'deki özelliklerini gösterir. RBPF filtresi FastSLAM olarak da adlandırılır. Bu filtrede örnek kümelerimiz kestirilen robot yerine bağlı olarak kestirilen 'işaretçi nesnelerden ve robotun kestirilen yerinden' oluşur. Parçacık Filtrelerinde popüler olarak SIR algoritmasından yararlanılır. Bu algoritmada Kalman Filtreleri ek bir adım olarak kullanılır. En önemli özellikleri, SLAM'de kullanılan bir çok filtre gibi Bayes tahminine dayanır, gauss dağılımlı olmayan ve lineer olmayan durumları kestirebilir. Diğer önemli özelliği ise, FastSLAM'in SLAM problemini Robotun yer bulma problemi ve işaretçi nesnelerin yerlerinin tahmin edilmesi problemi olarak ikiye ayırmasıdır. Kullanılan FastSLAM'da her parçacıkta m tane işaretçi nesne varsa m tane de GKF bulunmaktadır. Yani işaretçi nesne kestirimlerinde GKF'den yararlanılmaktadır.

Çalışmamız gereği EKF'nin ve FastSLAM'in ayrıldığı noktaya dikkat çekmek gerekir. Kalman filtresinin amacı işaretçi nesnelerin yerinden mobil robotun yerini tespit etmektir; FastSLAM'de ise amaç, araçın potansiyel konumları arasından aracın kontrol işareti ve gözlemleri kullanılarak bulunduğu yeri ve haritayı tespit edilmesidir. Çizelge 4.5'te FastSLAM'de kullanılan sembollerin açıklamaları verilmiştir. Bundan önceki bölümlerden farklı olarak beklenen değer ve kovariyans değerlerinin sembolleri FastSLAM çalışmalarında kullanılan biçimde değiştirilmiştir.

Çizelge 4.5 Denklemlerde Kullanılan Sembollerin Açıklamaları

Semboller		Anlamları
x	**x**	Aracın x konumu
y		Aracın y konumu
Φ		Aracın açısı
Θ,m		Robotun yerine bağlı kestirilen işaret yerleri
n,i		Toplam Parçacık sayısı, parçacık numarası
m		İşaretçi nesne sayısı
μ		Beklenen değer
Σ		kovariyans
u		Kontrol işareti
z		Gözlem
t		Zaman

Her parçacık, aracın izleyebileceği yolun tahminidir. Parçacıklar genel olarak $S_t^{[i]}$ ile gösterilir. Robotun konum bilgilerinden, işaretçi nesnelerin beklenen değerlerinden ve kovariyanslarından çizelge 4.5 yardımı ile denklem 4.61'deki parçacık yapısı oluşturulabilir. Her parçacığın ilk elemanı x^t mobil robotun o parçacık için önerilmiş konumunu ve o ana kadar kendi haritasında görülmüş işaretlerin $\mu_{m,t}$ beklenen değer ve $\sum_{m,t}$ kovariyansları gözükmektedir.

$$S_t^{[i]} = \langle x^{t,[i]}, \mu_{1,t}^{[i]}, \Sigma_{1,t}^{[i]}, \ldots\ldots, \mu_{m,t}^{[i]}, \Sigma_{m,t}^{[i]} \rangle \quad \textbf{(4.61)}$$

$$p(x_{1:t}, \theta_{1:m} \mid z_{1:t}, u_{0:t-1}) = p(x_{1:t} \mid z_{1:t}, u_{0:t-1})\, p(\theta_{1:m} \mid x_{1:t}, z_{1:t}, u_{0:t-1}) \quad \textbf{(4.62)}$$

$$\begin{aligned} p(x_{1:t}, \theta_{1:m} \mid z_{1:t}, u_{0:t-1}) &= p(x_{1:t} \mid z_{1:t}, u_{0:t-1})[p(\theta_1 \mid x_{1:t}, z_{1:t}, u_{0:t-1})\ldots \\ &\ldots p(\theta_m \mid \theta_{m-1}, \ldots, \theta_1, x_{1:t}, z_{1:t}, u_{0:t-1})] \end{aligned} \quad \textbf{(4.63)}$$

SLAM problemi parçacık filtrelerine göre denklem 4.64'te yazılmıştır.

$$p(x_{1:t}, \theta_{1:m} \mid z_{1:t}, u_{0:t-1}) = p(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \prod_{j=1}^{m} p(\theta_t^{\,j} \mid x_{1:t}, u_{0:t-1}) \quad \textbf{(4.64)}$$

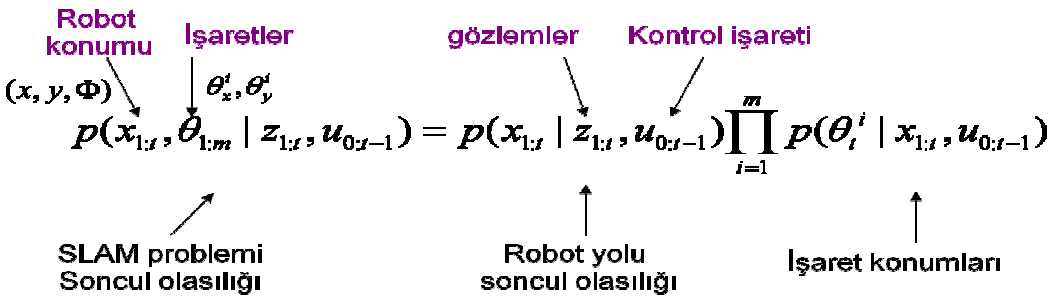
Denklem 4.64'te mobil robot için 1 tane, işaretçi nesneler için ise m tane olmak üzere m+1 tane kestirici kullanılmıştır.

$$p\,(x_{1:t}, \theta_{1:m} \mid z_{1:t}, u_{0:t-1}) \quad \textbf{(4.65)}$$

$$p\,(x_{1:t} \mid z_{1:t}, u_{0:t-1}) \quad \textbf{(4.66)}$$

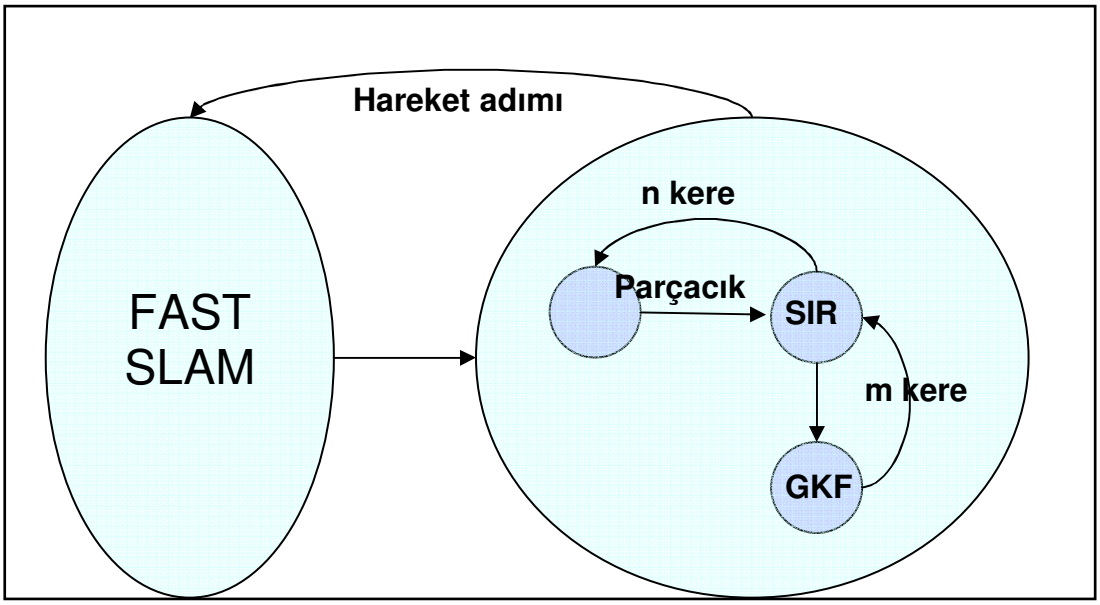
$$\prod_{j=1}^{m} p\,(\theta_t^{\,j} \mid x_{1:t}, u_{0:t-1}) \quad \textbf{(4.67)}$$

Denklem 4.65 SLAM problemini gösterir, denklem 4.66 aracın konumunu gözlem ve kontrol işaretlerine göre gösterir. Denklem 4.67 ise işaretçi nesneleri gösterir. Denklem 4.64'ten de anlaşılacağı üzere her araç konumu içi farklı işaretçi nesne haritaları yaratılmaktadır. Önem ağırlığı en fazla olan robot haritaları aynı MCL' de olduğu gibi bir sonraki adıma saklanır. Bozulmuş olanlarsa silinir veya iyi olanlar ile değiştirilir. Denklem 4.64'ün açıklamalı hali şekil 4.5'te verilmiştir.



Şekil 4.5 : SLAM Probleminin FastSLAM Hali

FastSLAM algoritmasının daha rahat anlaşılması için şekil 4.6 oluşturulmuştur.



Şekil 4.6 : FastSLAM şematik gösterimi

4.7 RBPF Algoritmasının Adımları

Çalışmalarda FastSLAM algortiması aşağıdaki adımlarda uygunlanmıştır[2]. RBPF araştırıldığında bazı yayınlarda Haritalama, örnekleme adımı ile birlikte kullanılmıştır. Böyle bir yapı düşünüldüğünde şüphesizdir ki RBPF tam anlamı ile MCL algoritmasına benzer. Yapı olarak aynı adımları içerse de, bir ve iki numaralı adımların uygulanmasında büyük farklılıklar göze çarpar.

1) Örnekleme
2) Önem ağırlıklarının hesaplanması
3) Yeniden örnekleme
4) Haritalama

Bu adımların uygulanması sırasında ise belli başlı problemlere çözümler aranır. Bu problemler değişen çevreye göre kullanılması gereken parçacık sayısı, yeterli parçacık sayısının bulunması, parçacıkların hangi veri ilişkilendirme yöntemlerine göre çalıştırılması, yeniden örnekleme adımının hangi şartlarda yapılması gerektiği olarak sıralanabilir.

4.7.1. Örnekleme

SLAM algoritmasının RBPF olarak araç yolu kestirimi ve m tane işaretin kestirimi olmak üzere iki ana bölüme ayrıldığı bölüm 4.6'da belirtilmişti. Her parçacık birbirinden bağımsız olarak çevre haritası içerdiğinden ve araç yolları da önerisel bir dağılım biçiminde belirlendiğinden çok hipotezli veri eşleştirmesinden bahsedilebilir.

4.7.1.1 Araç yolu kestirimi

Araç yolu kestirilirken MCL algoritmasına benzer bir öneri dağılımı oluşturulur.Aşağıda MCL hareket denklemi 4.50 gözükmektedir.

$$x_t^{[i]} \sim p(x_t \mid u_t, x_{t-1}^{[i]}) \tag{4.68}$$

Mobil robota başlangıçta ağırlıkları eşit, belirli bir bölge içinde tahminler atanmıştır. Bu tahminler her bir parçacığın ilk elemanını oluşturan konum matrisleridir. MCL konumlarının simülasyondaki oluşturulma matrisi denklem 4.69'daki gibidir, yapılan işlemler sonrası bu matrisin elemanları denklem 4.61'e katılmıştır.

$$S_t^{[i]} = \left\langle x^{t,[i]} \right\rangle_n = \left\langle x_1^{[i]}, x_2^{[i]}, \ldots., x_t^{[i]} \right\rangle_n \tag{4.69}$$

Yapılan gözlemler sonrası aracın yeni konumu denklem 4.71'e göre hesaplanır.

$$w_t^{[i]} = \frac{p(x)}{q(x)} = \frac{hedefdağılım}{önerilendağılım} = \frac{p(x^{t,[i]} \mid z^t, u^t)}{p(x^{t,[i]} \mid z^{t-1}, u^t)} \tag{4.71}$$

Bu hesaplamanın yapılabilmesi için aracın işaret gözlemlerine ve odometri bilgilerine ihtiyaç vardır. 2 numaralı kaynağa göre önerisel dağılım için yazının üçüncü bölümünde açıklanan Bayes iteratif denkleminden yola çıkmıştır.

$$p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) = \frac{p(z_t \mid m_{t-1}^{(i)}, x^t)\, p(x_t \mid x_{t-1}^{(i)}, u_{t-1})}{\int p(z_t \mid m_{t-1}^{(i)}, x_{t-1})\, p(x_{t-1} \mid x_{t-1}^{(i)}, u_{t-1}) dx_{t-1}} \tag{4.72}$$

Denklem 4.72'nin anlamlı kısmını gözlem olasılığını yönetmektedir. Denklem 4.73'te anlamlı bir aralık tanımlanır.

$$L^{(i)} = \left\{ x \mid p(z_t \mid m_{t-1}^{(i)}, x) > \in \right\} \tag{4.73}$$

Yanındaki araç konumu olasılığı k sabiti yakınsanır ve denklem 4.72'te düzenleme yapılır.

$$p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) \cong \frac{p(z_t \mid m_{t-1}^{(i)}, x^t)}{\int_{x_{t-1} \in L^{(i)}} p(z_t \mid m_{t-1}^{(i)}, x_{t-1}) dx_{t-1}} \tag{4.74}$$

Denklem 4.74, maksimum odometri hatası ile sınırlandırılmış olan $p(z_t \mid m_{t-1}^{(i)}, x^t)$ hesaplanarak ya da, alternatif olarak, bir parçacıktaki aracın konumuna ait gözlemlerinden yola çıkılarak beklenen değer ve kovariyansının bulunması ile mümkün olur. Bir parçacıkta alınan aracın konumu tek olduğundan, yapılacak işlem gözlemlerden yola çıkarak aracın konumu hesaplamaktan geçer. Araçla işaretler arasından lazer tarayıcı ile ölçüm alarak ilişki kurmak için SLAM'de denklem 4.75'e ihtiyaç vardır. Kullanılan denklem aynı zamanda GKF hesaplamalarında kullanılan $x_t^{(i)}$ durumunu $z_t^{(i)}$'ye bağlayan H matrisidir.

$$z_{(i)} = h_i(x_v) = \begin{bmatrix} \sqrt{(x_i - x_v)^2 + (y_i - y_v)^2} \\ \arctan \dfrac{(y_i - y_v)}{(x_i - x_v)} - \Phi_v \end{bmatrix} = \begin{bmatrix} (x_i - x_v) & 0 & 0 \\ 0 & (y_i - y_v) & 0 \\ 0 & 0 & \arctan \dfrac{(y_i - y_v)}{(x_i - x_v)} - \Phi_v \end{bmatrix} \tag{4.75}$$

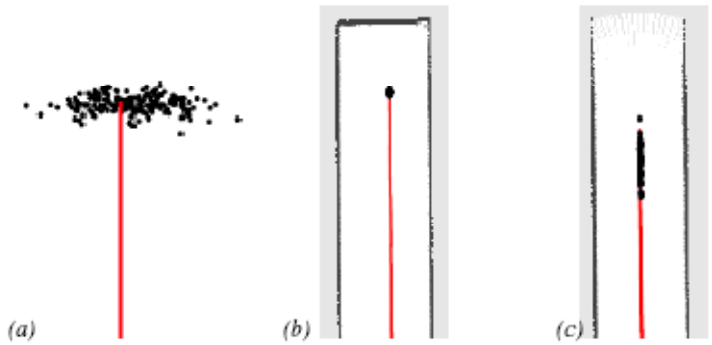
Bu işlemlere bağlı olarak hesaplanan $z_{(i)}$ Grisetti,Stachniss ve Burgard tarafından önerilen denklem 4.76'daki $p\left(z_t \mid m_{t-1}^{(i)}, x_j\right)$ yerine kullanılır[2]. Denklem 4.76'da hesaplanan beklenen değerlerle bir adımdaki parçacıklar için araç konum dağılımına ulaşılır. Araç konumları için denklem 4.78'deki matris yapısı kullanılır.

$$\mu_t^{(i)} = \frac{1}{\eta^{(i)}} \sum_{j=1}^{m} x_j p\left(z_t \mid m_{t-1}^{(i)}, x_j\right) \tag{4.76}$$

$$\sum_t^{(i)} = \frac{1}{\eta^{(i)}} \sum_{j=1}^{m} p\left(z_t \mid m_{t-1}^{(i)}, x_j\right)\left(x_j - \mu_t^{(i)}\right)\left(x_j - \mu_t^{(i)}\right)^T \tag{4.77}$$

$$x_t^{(i)} = \begin{bmatrix} \mu_{v_x} & 0 & 0 \\ 0 & \mu_{v_y} & 0 \\ 0 & 0 & \Phi_v \end{bmatrix} \quad \textbf{(4.78)}$$

Şekil 4.7'de önerisel dağılım ortama göre izlediği dağılım biçimleri gözükmektedir. Görüldüğü üzere açık alan ve uzun bir koridor ve ucu kapalı yolda öneri dağılımının önerdiği dağılımlar birbirinden gözle görülür biçimde ayrılırlar. Bu ise bize araç yolu kestirimindeki esnekliği gösterir.



Şekil 4.7 : Önerisel dağılımın aldığı farklı biçimler (a),(b) ,(c) bölümlerinde gösterilmektedir.

4.7.1.2 İşaretçi nesne yerlerinin kestirimi

Denklem 4.61'deki parçacığın içeriğinde ilk terimden sonra gelen bütün terimlerin işaretlerin $\mu_{m,t}^{[i]}$ beklenen değer ve $\Sigma_{m,t}^{[i]}$ kovariyanslarını oluşturduğunu belirtilmişti. Beklenen değer ve kovariyans matrislerinin simülasyonda kullanılan hali denklem 4.79 ve denklem 4.80'de verilmiştir.

$$\mu_{k,t}^{(i)} = \begin{bmatrix} \mu_x & 0 & 0 \\ 0 & \mu_y & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \textbf{(4.79)}$$

$$\Sigma_{k,t}^{(i)} = \begin{bmatrix} \Sigma_x & 0 & 0 \\ 0 & \Sigma_y & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \textbf{(4.80)}$$

$$p(\theta_k \mid x^t, z^t, u^t) \overset{Bayes}{\propto} p(z^t \mid \theta_k, x^t, z^{t-1}, u^t)\, p(\theta_k \mid x^t, z^{t-1}, u^t)$$

$$\overset{Markov}{=} p(z^t \mid \theta_k, x^t)\, p(\theta_k \mid x^{t-1}, z^{t-1}, u^{t-1}) \quad \textbf{(4.81)}$$

$$p(\theta_k \mid z^t, x^t, u^t) = p(\theta_k \mid x^{t-1}, z^{t-1}, u^{t-1}) \quad \textbf{(4.82)}$$

Denklem 4.81'deki düzeltme algoritması GKF kullanılarak gerçekleştirilmiştir. SLAM'de GKF kullanıldığında filtre doğrusallaştırılmış $p(z \mid x, \theta)$ gözlem modelini yakınsardı. Oysa, FastSLAM'in GKF'sinde $p(z \mid x, \theta)$ gözlem modeli doğrusal gauss tipi fonksiyon kullanılarak yakınsanır. Burada önemli olan, hareket modeli doğrusal olmasa bile, doğrusal gauss tipi gözlem modeli kullanılarak sonuç dağılımı kesinlikle gauss tipidir[1].

Kullanılan GKF denklemleri çizelge 4.6 ve 4.7'de verilmiştir. Bu denklemler çizelge 4.4'de verilen GKF algoritması denklemleri ile benzerdir. Sadece kullanılan semboller parçacık filtresi literatürüne uygun hale getirilmiştir.

Çizelge 4.6 : FastSLAM GKF Denklemleri

Zaman Güncellemesi(Öngörü)*

$$\hat{\mu}(t|\text{t-1}) = \overbrace{\text{f}\,(\ \underbrace{\hat{\mu}(t-1|\text{t-1})}_{\text{bir önceki kestirim değeri}}\ ,\ \underbrace{\text{u(t)}}_{\text{kontrol}}\ ,\text{t})}^{\text{sistem modeli}} \quad \textbf{(4.83)}$$

$$\underbrace{\Sigma(t|\text{t-1})}_{\text{tahmini kovaryans}} = \nabla \text{F}_{\mu} . \underbrace{\Sigma(t-1|\text{t-1})}_{\text{bir önceki kestirilen kovaryans}} . \nabla \text{F}_{\mu}^{\ T} + \underbrace{\nabla F_v . Q . \nabla F_v^{\ T}}_{\text{sistem gürültüsü}} \quad \textbf{(4.84)}$$

$$\underbrace{z(t|\text{t-1})}_{\text{tahmini gözlemler}} = \overbrace{h(\hat{\mu}(t|\text{t-1}),\text{u(t)},\text{t})}^{\text{gözlem modeli}} \quad \textbf{(4.85)}$$

Zaman Güncellemesi Simülasyon Formu

$$\begin{aligned} \mu_{t|t-1}^{(i)} &= \nabla F_{\mu_t}^{(i)} \cdot \mu_{t-1}^{(i)} + F(\mu_t^{(i)}) \cdot u_t \\ \Sigma_{t|t-1}^{(i)} &= \nabla F_{\mu_t}^{(i)} \cdot \Sigma_{t|t-1}^{(i)} \cdot \nabla F_{\mu_t}^{(i)T} + \nabla F_{z_t}^{(i)} \cdot Q \cdot \nabla F_{z_t}^{(i)T} \\ z_{t|t-1}^{(i)} &= \nabla H_{\mu_t}^{(i)} \mu_{t|t-1}^{(i)} + G(\mu_t^{(i)}) u_t \end{aligned} \quad \textbf{(4.86)}$$

**

*Denklemlerdeki jakobiyen formlar çizelge 4.3'teki gibidir

**Denklem 4.86'daki $G(\mu_t^{(i)})$ Mobil robota göre işaret yerlerinin dinamik değişmesini gösteren bir matristir. Simülasyonda yoktur.

Ölçüm Güncellemesi(Düzeltme)*

$$\upsilon(t) = \overbrace{z(t)}^{gözlemler} - z(t|t\text{-}1) \qquad \textbf{(4.87)}$$

$$W = \underbrace{P(t|t\text{-}1).\nabla H_{\mu}{}^{T}.S^{-1}}_{Kalman-kazancı}$$

$$S = \underbrace{\nabla H_{\mu}.P(t|t\text{-}1).\nabla H_{\mu}{}^{T} + \underbrace{R}_{ölçümgürültüsü}}_{inovasyon-kovaryansı}$$

Ölçüm Güncellemesi Simülasyon Formu

$$\underbrace{\hat{\mu}(t|t)}_{yeni-kestirilen-durumlar} = \overbrace{\underbrace{\hat{\mu}(t|t\text{-}1)}_{tahmin edilen,\mu} + W.\underbrace{\upsilon(t)}_{inovasyon}}^{\text{tahmin ve düzeltme}} \qquad \textbf{(4.88)}$$

$$\underbrace{\Sigma(t|t)}_{yeni-kestirilen-kovaryans} = \Sigma(t|t\text{-}1) - WSW^{T} \qquad \textbf{(4.89)}$$

$$W^{(i)} = \Sigma^{(i)}_{t|t-1} \cdot \nabla H^{(i)T}_{\mu_t} \cdot S^{-1(i)} \qquad \textbf{(4.90)}$$

$$S^{(i)} = \nabla H^{(i)}_{\mu_t} \cdot \Sigma^{(i)}_{t|t-1} \cdot \nabla H^{(i)T}_{\mu_t} + R$$

*Denklemlerdeki jakobiyen formlar çizelge 4.3'teki gibidir

Simülasyonlarda kullanılan jakobiyen biçimdeki denklemler çizelge 4.7'de verilmiştir

Çizelge 4.7 : Jakobiyenlerin simülasyonda Kullanılan Halleri

Jakobiyenlerin simülasyonda Kullanılan Halleri

$$\nabla F^{(i)}_{\mu_t} = \begin{bmatrix} 1 & 0 & -(\mu_{t_x} - \mu_{vt-1_x})\sin(\Phi_v) - (\mu_{t_y} - \mu_{vt-1_y})\cos(\Phi_v) \\ 0 & 1 & (\mu_{t_x} - \mu_{vt-1_x})\cos(\Phi_v) - (\mu_{t_y} - \mu_{vt-1_y})\sin(\Phi_v) \\ 0 & 0 & 1 \end{bmatrix} \qquad \textbf{(4.91)}$$

$$\nabla F_{\mu_t}^{(i)} = \begin{bmatrix} \cos\Phi_v & -\sin\Phi_v & 0 \\ \sin\Phi_v & \cos\Phi_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(4.92)}$$

$$\nabla H_x^{(i)} = \begin{bmatrix} \frac{(\mu_{t_x} - \mu_{vt-1_x})}{r} & \frac{(\mu_{t_y} - \mu_{vt-1_y})}{r} & 0 \\ \frac{(\mu_{t_y} - \mu_{vt-1_y})}{r^2} & -\frac{(\mu_{t_x} - \mu_{vt-1_x})}{r^2} & -1 \end{bmatrix} \quad \textbf{(4.93)}$$

4.7.2. Önem ağırlıklarının hesaplanması

Yeniden örnekleme için parçacıkların ağırlıklarının belirlenmesi gereklidir. Amaç denklem 4.94'ü bulmaktır.

$$w_t^{[i]} = \frac{p(x)}{q(x)} = \frac{hedefdağılım}{önerilendağılım} = \frac{p(x^{t,[i]} \mid z^t, u^t)}{p(x^{t,[i]} \mid z^{t-1}, u^t)} \quad \textbf{(4.94)}$$

Parçacıkların başlangıç ağırlıkları denklem 4.47'deki gibidir. Sonraki adımlarda, parçacıkta yer alan bütün işaretlerin ne kadar değiştiği önemlidir. Bunu anlamak için 4.87'den inovasyon matrisi, denklem 4.88'den beklenen değerler ve denklem 4.89'dan kovariyans değerleri alınır. Bu değerlerden normal olasılık yoğunluk fonksiyonu bulunur. Bu fonksiyondan bölüm 4.5.2'de anlatıldığı gibi maksimum benzerlikler bulunur. Ancak bulunan değer doğrudan ağırlık olarak kaydedilmemelidir. Doğru bir yakınsama olması açısından ağırlıkların bulunmasında çeşitli yöntemler uygulanmaktadır. Bolic'e göre denklem 4.64'ün yazın formuna gelmiş bir parçacık filtresinden, denklem 4.94 denklem 4.95'teki gibi belirtilebilir.(Bolic) Elde edilen denklemde görülmesi gereken, bir önceki değer ile yapılan çarpma işlemidir.

$$w_t = w_{t-1} \frac{p(z_t \mid x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)})}{\Pi(x_t^{(i)} \mid x_{1:t-1}^{(i)}, z_{1:t})} \quad \textbf{(4.95)}$$

$$w_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^{n} w_t^{(j)}} \quad \textbf{(4.96)}$$

Yapılan simülasyonda denklem 4.95'deki gibi ağırlıklar parçacığın bir önceki adımındaki değeri ile çarpılır, ardından denklem 4.96 ile bulunan bütün ağırlıklara normalizasyon uygulanır. Böylece parçacıkların toplam değeri bir olan önem ağırlık

kümeleri elde edilmiş olur. Burada dikkat edilmesi gereken işaretler *2x2* boyutunda olduğundan x ve y düzlemlerinde ayrı yada işaret uzaklıkları r biçiminde değerlendirilerek ağırlıkların bulunması gereğidir. Çizelge 4.8'in 3. sütununda örnek olrak hesaplanan ağırlık değerleri gözükmektedir.

Çizelge 4.8 : FastSLAM simülasyonunda ikinci adımdaki on parçacığın ağırlıkları

weight_mon(:,:,2) =		
0.1118	0.0452	0.0511
0.0981	0.1471	0.1456
0.0981	0.0821	0.0813
0.0981	0.0536	0.0531
0.0981	0.1340	0.1327
0.1037	0.0583	0.0610
0.0981	0.1340	0.1327
0.0981	0.1340	0.1327
0.0981	0.0647	0.0641
0.0981	0.1471	0.1456

4.7.3. Yeniden örnekleme

Yeniden örnekleme önerilen dağılımın önceki dağılımlardan çok farklı olduğu durumlarda uygulanmalıdır. Zamanında yapılan yeniden örnekleme parçacık çeşitliliğini arttırmaya yardımcı olur, filtrenin bozulmasını engeller[4].(Parçacık bozulması problemi) Araştırılmış yayınlarda iki farklı işleme rastlanır. Örnekleme sonucunda düşük önem ağırlıklı parçacıklar elenmeli veya yüksek önem ağırlıklı yeni parçacıklarla değiştirilmelidir. Periyodik olarak yapılmasına gerek yoktur. Her parçacık aracın önceki $x_{1:t}^{[i]}$ konumlarını içerir. Zamanla örnek uzayındaki benzerlikler azalır. Parçacık filtrelerinin örnekleme variyansları artar ve düşük değerli bir parçacık bulana kadar filtreleme prosesi uzar. Yeniden Örnekleme tam bu noktada uygulanmalıdır. Ek veri eşleştirme algoritmaları kullanılırken dikkat edilecek unsur, yanlış bir parçacığın silinmesi ile o ana kadar doğru olabilecek bir yol ve harita bilgisinin silinmesi anlamına gelir. Bölüm 4.7'nin altında bahsedilen problemlerin çözüm yerinin yeniden örnekleme adımı olması gerekir. Çünkü yeniden örnekleme parçacık sayısına etki eder. Şu ana kadar dünyada yapılan araştırmalar parçacık sayısının belirli bir değerin üstünde olmasının filtrenin çalışmasına olumlu yönde etki edeceğini göstermiştir. Araç yerinin tahmininde her parçacık aynı öneme sahip olmaz.Amaç, efektif parçacık sayısının bulunmasıdır.

Böylece algoritmanın işlem yükü azaltır. Parçacık sayısının silinmesi veya yeniden kopyalanmasını gerektirecek eşikler veya eklenen ek algoritmalar çok çeşitlidir. Simülasyonda kullanılan üç farklı yeniden örneklemenin yapılma zamanı bulan algoritmadan bahsedilecektir.Yeniden örnekleme yöntemleri aşağıdaki gibi adlandırılır.

1. Basit
2. Adaptif(Seçkin)[2,19]
3. Sistematik[18]

Yeniden örneklemede kullanılan yöntemlerin RBPF'ye nasıl etki ettiği parçacık filtresinin başarısı sonuçları verilirken görülecektir.

Basit yeniden örneklemede denklem 4.47'deki 1/n başlangıç ağırlığı değerinden ufak keyfi bir değer seçilir.Simülasyonda alınan değer denklem 4.48'de gösterilmiştir.

$$thr = \frac{1}{2n} \tag{4.97}$$

Eğer bu değerin altına düşülürse parçacığa yeniden örnekleme uygulanır, yoksa devam edilir. Bu basit eşiğin problemi, ancak düşük parçacık sayılarında işe yaramasıdır. Yirmi veya daha büyük sayılı parçacıklarda bir çok parçacık bozulup bu eşik değerinin altına düşme işlemi çok fazla olacağından her adımda yeniden örnekleme yapılır. Her adımda yeniden örnekleme her ne kadar sağlıklı olsa da enerji ve hesaplama açısından istenmeyen bir durumdur. Ek olarak bozulduğu kabul edilen parçanın silinmesi durumunda veya daha yüksek ağırlıklı bir parça ile değişmesi durumunda her adımda parçalar arası benzerlik artar. Oysa, parçacık filtresinin amacı çeşitliliği sağlayan öneri dağılımını kullanarak SLAM problemini çözmektir.

Yeniden örneklemenin gerekli ama sıklıkla kullanılmaması gerektiği bu bölümün başında anlatılmıştı. Popüler olarak yeniden örnekleme kriteri olarak kullanılan adaptif yeniden örnekleme olası kaç tane parçacığın filtrenin düzgün çalışması için yeterli olduğunu söyler. Bu tahmini yapabilmek için de denklem 4.49'daki n_{eff} kriterinden faydalanılır.

$$n_{eff} = \frac{1}{\sum_{i=1}^{n} (w^{(i)})^2} \tag{4.98}$$

n_{eff} keyfi bir tahminden öte bir değerdir. Çünkü, eğer örnekler gerçek sonculdan alınmış ise örneklerin önem ağırlıkları ilk adımdaki gibi birbirlerine eşit olurlar. Daha kötü yakınsamada ağırlıkların kovariyansları daha yüksek olur. Eğer n_{eff} parçacık silme kriteri ise, hangi parçacığın aracın gerçek yerine yakınsattığına bakmak iyi bir karar kriteridir[2].

Boliç tarafından tanıtılan sistematik yeniden örneklemede[18] 0 ile 1/n arasından rastgele bir sayı seçilir.Yeniden örnekleme algoritmasında bütün parçacık ağırlıkları toplanır. Eğer toplanan sayı rastgele seçilen sayıdan büyükse yeniden örnekleme yapılır ve rastgele sayının değeri düzenli bir şekilde arttırılır. Simülasyon değişkenleri ile verilen sistematik yeniden örnekleme çizelge 4.9'da verilmiştir.

Çizelge 4.9 : Sistematik yeniden örnekleme algoritması

Amaç: Sistematik Yeniden Örneklemenin uygulanması(SR)
Giren: Parçacık ağırlık dizisi $\left\{w^{(i)}\right\}_{i=1}^{n}$,yeniden örnekleme sayısı *r*, örneklenen parçacık sayısı *p,* parçacık sayısı *n*
Metot:

```
U~U(0, 1/n]                 // Rastgele U sayısının yaratılması
s = 0                       //Ağırlık toplama değişkeni
p = n                       //Örneklenecek parçacık sayısının atanması
for  i = 1:n                //Ana döngü
        r = 0               //Yeniden örnekleme sayacı
        s = s + w(i)        //Ağırlıkların toplanması
        while s > U         //Yeniden örnekleme döngüsü
                r = r + 1
                U=U + 1/p
        end
end
```

4.7.4. Haritalama

Haritalama GKF düzeltmelerinin yapıldığı bölümdür. Bu adım yapılan simülasyonda örnekleme adımı ile beraber işlenmiştir. Her araç konumu $x_t^{[i]}$'ye bağlı olarak işaretçi nesnelerin, $p(\theta_t^{[i]} \mid x_{1:t}^{[i]}, z_{1:t})$ 'nin , kestirimi yapılır.

4.8 RBPF Simülasyonu

Araştırmaların somutluk kazanması açısında teoriğin pratikte de uygulanabilirliğini görmek gerekir. Bu yüzden, MATLAB programı kullanılarak bir simülasyon programı hazırlanmıştır. Başarı kriterlerinin içeren çizelge 4.10 yeniden aşağıda verilmiştir. Bu başarıları test etmek açısıdan diğer test edilecek değerler içlerinden birinin test işlemi süresince sabit bırakılacaktır.Bu sabit değerler çizelge 4.11'de yazılı olan değerlerdir.

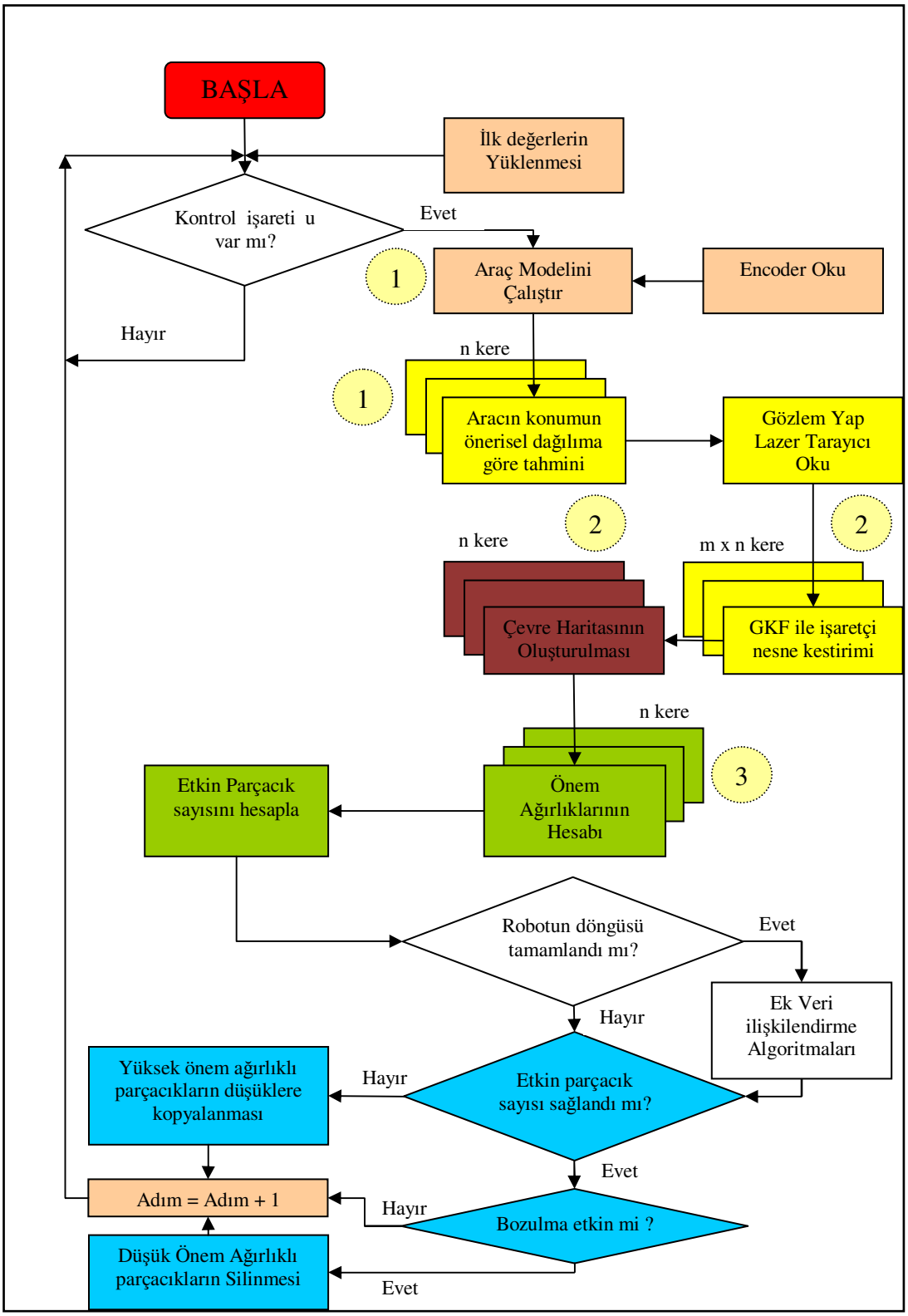
Çizelge 4.10 : RBPF Başarı Kriterleri

Gürültü	**İşaret Sayısı**	**Parçacık Sayısı**	**Yeniden Örnekleme**
Lazer Tarayıcı	0 -50	1	Basit
Süreç	51-500	10-500	Adaptif Sistematik

Çizelge 4.11 : Kriterlerin testi sırasında diğer kriterlerin standart değerleri

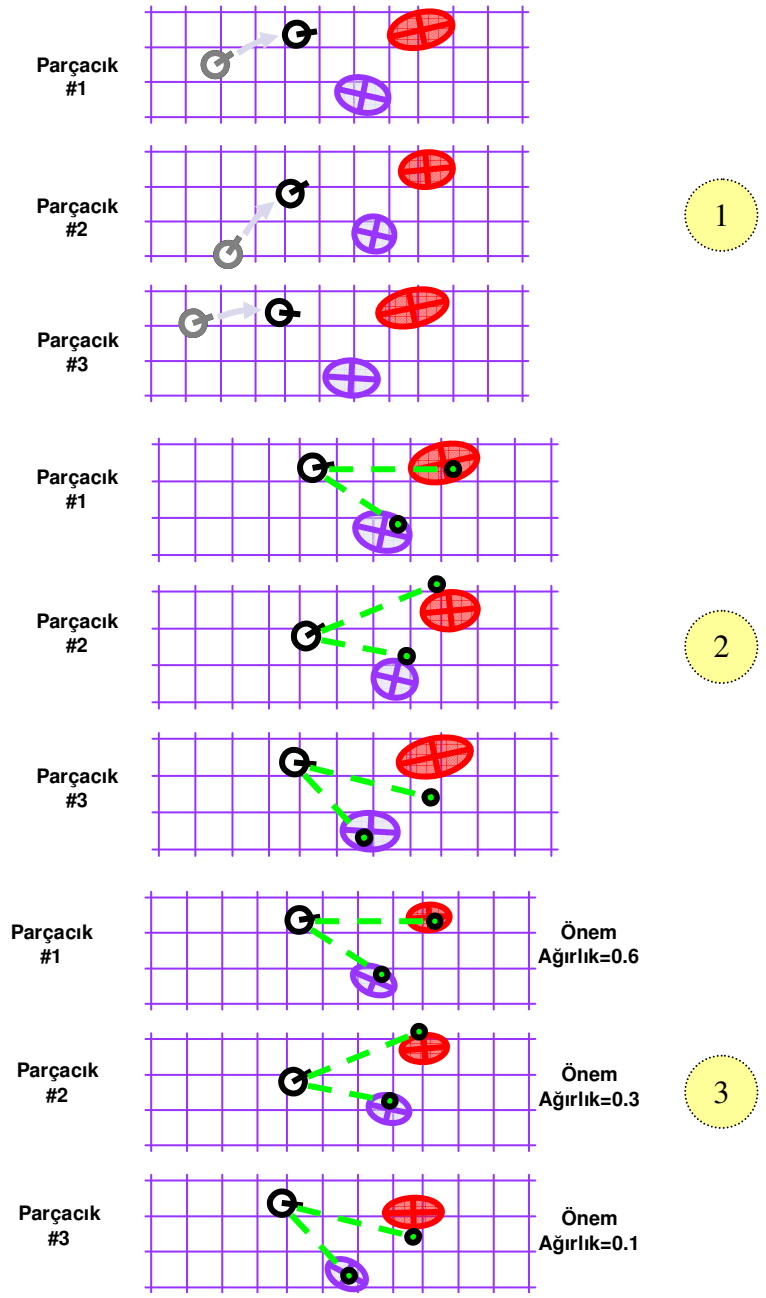
Kriter	**Değer-Durum**
Lazer Tarayıcı gürültüsü	*Denklem 2.14*
Süreç gürültüsü	$2e^{-5}$
İşaret sayısı	100
Parçacık Sayısı	25
Yeniden Örnekleme	Seçkin(Adaptif)

RBPF filtresinin adımları akış şeması bölüm 4.7'de anlatımına göre şekil 4.8'de verilmiştir.



Şekil 4.8 : RBPF Akış Şeması. Sarı Örnekleme, yeşil önem ağırlıklarının hesaplanmasını, mavi yeniden örneklemeyi ve kahverengi harita oluşturmayı temsil eder.

Şekil 4.9'da RBPF akış şemasına göre robotun yaptığı işler gözükmektedir.

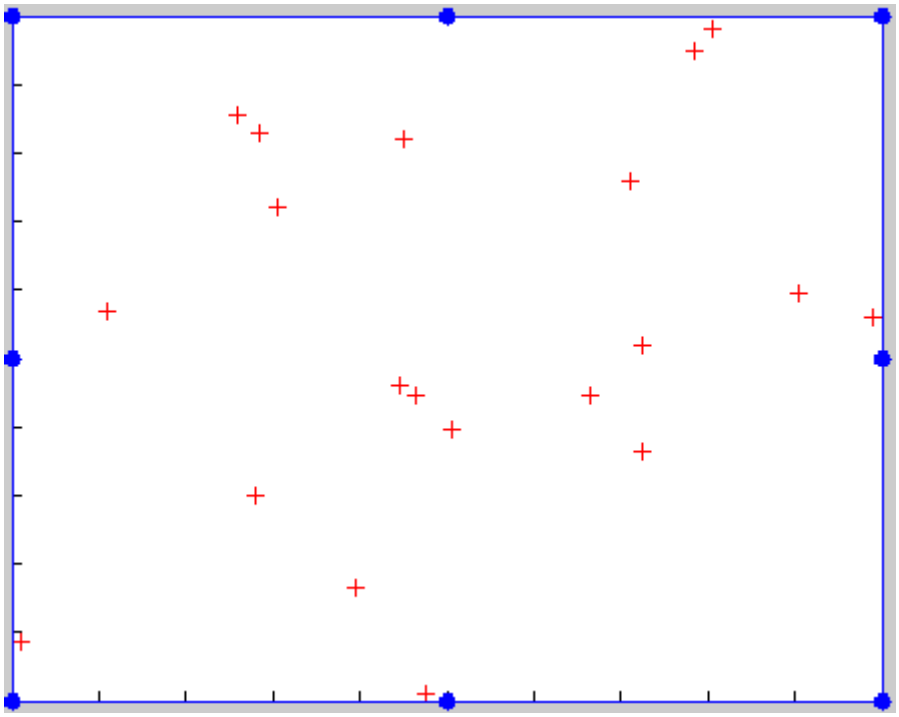


Şekil 4.9 : Akış Şemasına Göre Mobil robotun yaptıkları

4.8.1. Yörünge senaryosu

Simülasyon için bir deney alanı oluşturuldu. Bu deney alanının MATLAB görüntüsü şekil 4.10'da gözükmektedir. Mobil robot her mavi noktayı ziyaret ettiğinde ve her mavi noktada 90° 'lik dönüş yaptığında lazer tarayıcıdan ölçüm alır. Toplam 12 adımda bulunduğu noktaya gelir. Deneylerde kolaylık sağlaması açısından çevresi 16m'lik bir karenin her kenarının ortasına ve kare köşelerine birer gözlem noktası yerleştirilmiştir. Böylece 2m'lik aralıklarla düzgün doğrultularda

gözlem alabilme şansı olacaktır. Bu deney düzeneği TÜBİTAK projesi çalışması sırasında gerçek zamanlı gerçekleştirilmiştir.



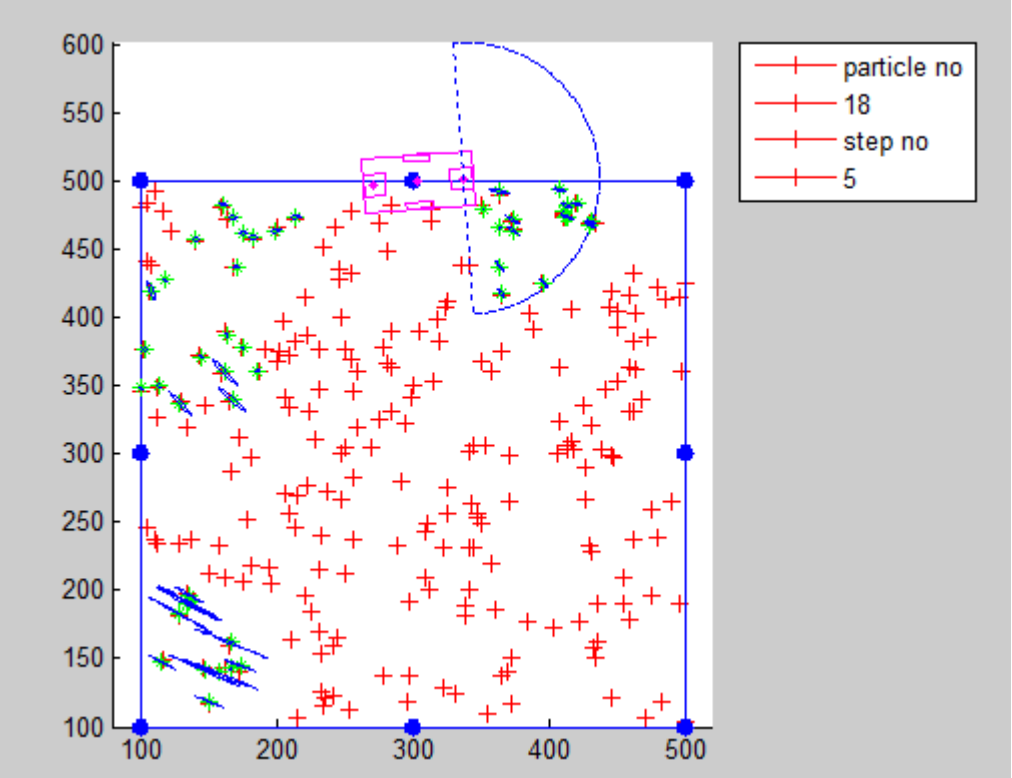
Şekil 4.10 Uygulama Alanı

Senaryoya göre kırmızı '+'larla gösterilen reflektörlerimiz deney alanına rastgele yerleştirilir. Buradaki amacımız reflektörlerin yerlerini doğru tahmin ederek aracın kontrolü sonucunda oluşturduğu hareket yolunu doğru oluşturabilmektir.

Aracın görüntüsü, MATLAB'ta gerçek ölçüleri ile çizilmiştir. Lazerin tarama alanı 80 metreye kadar çıkmasına rağmen deney alanımızın boyutlarına uygunluk için programda 2 metre olarak alınmıştır. Ancak, şu anda tek lazerli çalışma yapıldığından dolayı ikinci lazerin tarama alanı gösterilmemiştir.

4.8.2. RBPF deneyleri

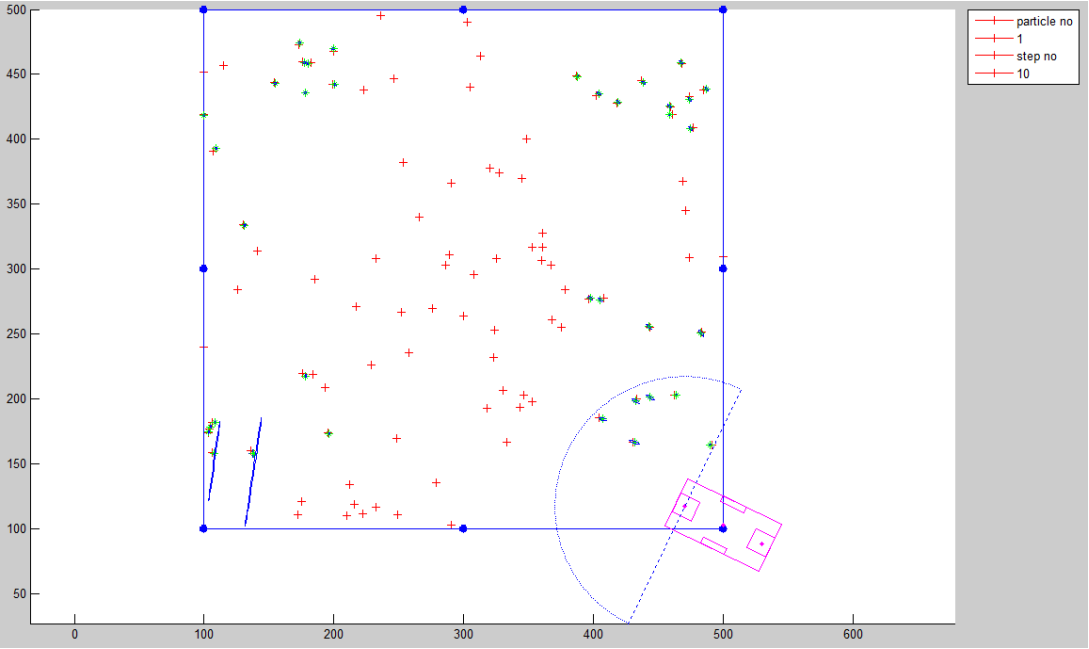
Kriterleri karşılaştırmadan önce kriterlerin çizelge 4.11 'deki sabit değerleri ile simülasyon çalıştırılmıştır. Bu standart çalışmaya ait çalışma değerleri öncelikle incelenecek ve simülayonun çalışması anlatılacaktır. RBPF filtresinde anlatıldığı gibi her parçacık kendi haritasını oluşturmaya başlar. Somut olarak bu haritalama işleminin yapılmasını şekil 4.11'de görmekteyiz. Şekilde 18 numaralı parçacığın simülasyonun 8. Adımında iken oluşturduğu hata elipslerini ve parçacığın tahmin ettiği işaretçi nesneler gözükmektedir.



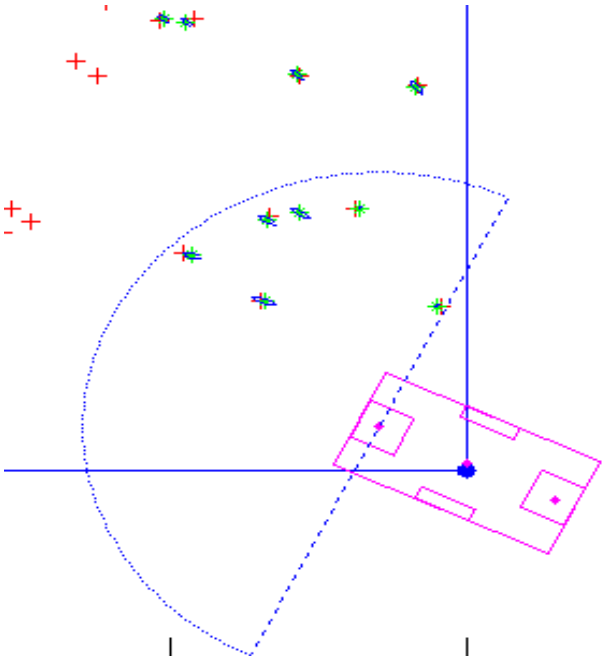
Şekil 4.11 : Mobil robot haritalamanın 5. adımında 18 numaralı parçacığın soncul olasılıklarını hesaplamış olarak gözükmektedir.

Şekil 4.13'de kırmızı '+' işaretler işaretçi nesnelerin konulduğu gerçek yerleri, Mavi '+''lar lazer tarayıcı ile alınmış, işlenmemiş işaretçi nesne yerlerini, yeşil '*' biçiminde olanlar tahmin edilen işaret yerlerini yeşil yıldızların etrafındaki mavi halkalar belirsizlik elipslerini, pembe aracın ön tarafındaki mavi hare ise lazer tarayıcının işleme alınan 180° 'lik görüş alanını göstermektir. Mavi hare içindeki kırmızı işaretler önce mavi '+''larla saptanır, filtre işleminden sonra bu işarete uygun belirsizlik elipsi ve yeşil yıldızlarla beklenen değeri atanır. Haritalarda dikkat edilmesi nokta, aracın her hareketinden sonra, belirli parçacıkların haritalarında bulunan işaretçi nesnelerin belirsizlik elipslerinin değer olarak ve gözle görülür bir biçimde büyümesidir. Bu büyüme birkaç adım önce taranmış eski işaretlerde, özellikle haritada aracın yaptığı 90° 'lik dönüşlerden sonra artar. İşaretçi nesnelerin belirsizliklerinin büyümesi yeniden örnekleme adımına kadar sürer. Eğer o parçacığın önem ağırlığı eşik değerinin altında kalırsa silinir ve daha iyi örneklenmiş bir harita ile yer değiştirilir. Böylece belirsizlikler azaltılmış olur. Çoklu veri eşleştirmesinin yanında her adımda, kayıtlı bir işaretin yeniden sisteme kaydedilmesini sisteme engelleyip, eski işaretleri tanıyabilmek maksadı ile tekil veri eşleştirmesinden faydalanılmıştır. Bir işaretin belirli bir yarıçapı içersinde eğer yeni

bir işaret tanımlanırsa, yeni işaret silinir, gözlem eski işaretle eşleştirilir. Şekil 4.12'de yeniden örneklenmiş bir parçacığın haritası gözükmektedir. Bir önceki adımlardaki örnekler gözle görülür biçimde azalmış olsa da, en fazla hatanın ilk adımda gözlemlenen işaretçi nesnelerde olduğu göze çarpmaktadır.

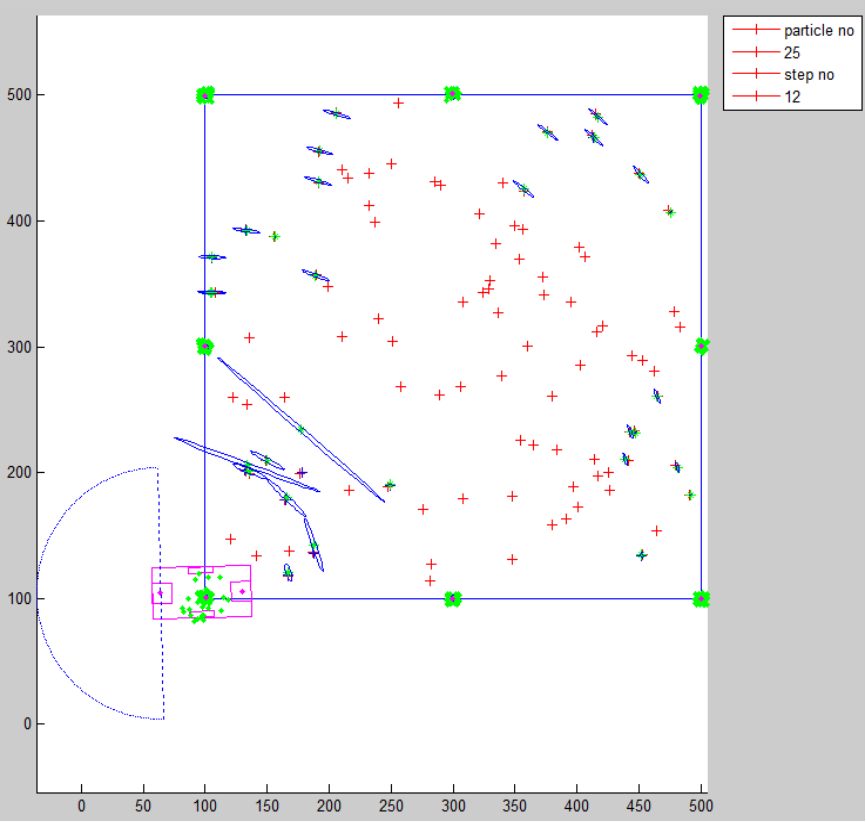


Şekil 4.12 : Mobil robot dönme işlemini gerçekleştirdikten sonra 10. adımında 1 numaralı parçacığın soncul olasılıklarını hesaplamış olarak gözükmektedir.

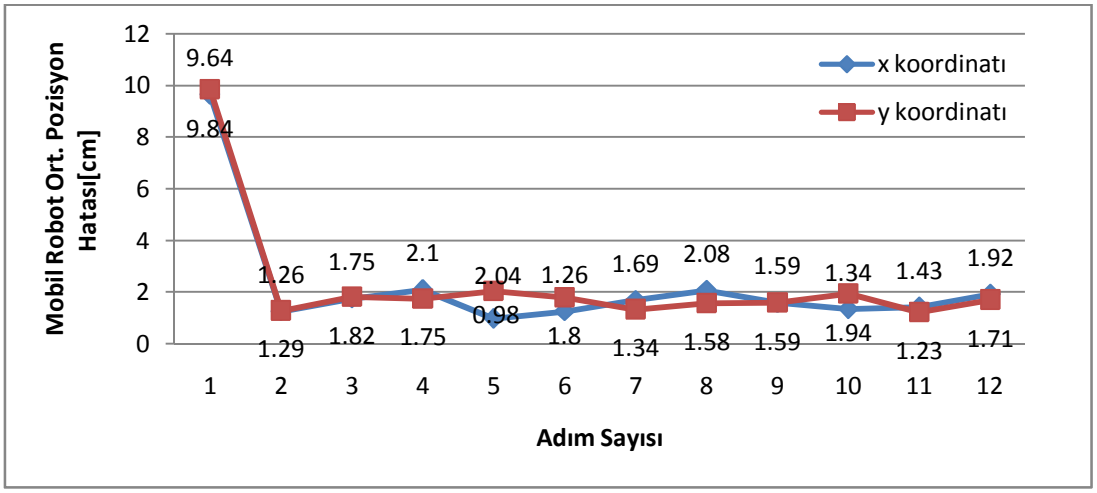


Şekil 4.13 : Mobil robotun lazer tarayıcı alanındaki işaretçi nesneleri görmesi ve filtreyi çalıştırması

Şekil 4.14'de Mobil robot haritalama işlemini tamamladıktan sonra, 25 numaralı parçacık için oluşturulmuş olan haritayı ve öneri dağılımına göre oluşturulmuş bütün parçacıklardaki robot konumları gözükmektedir. Bu şekilde dikkat edilmesi gereken nokta ilk adımda belirli bir kare alanı içerisinde rastgele noktalarda gözleme başlayan mobil robot, daha sonraki adımlarda öneri dağılımına göre belirsizliği daha ufak olan bir alanda gözlemlerini yapmıştır ve şekil 4.7'deki açık alana benzer bir şekilde konumlar dağılmıştır. Oluşturulmuş robot konum noktaları için denklem 4.76'dan yararlanılmıştır. Burada da gözüktüğü üzere, robotun konumunun hesaplanması, aracın enkoderlerinin ölçtüğü odometri hatası ve yapılan gözlemlerle ilişkilidir.

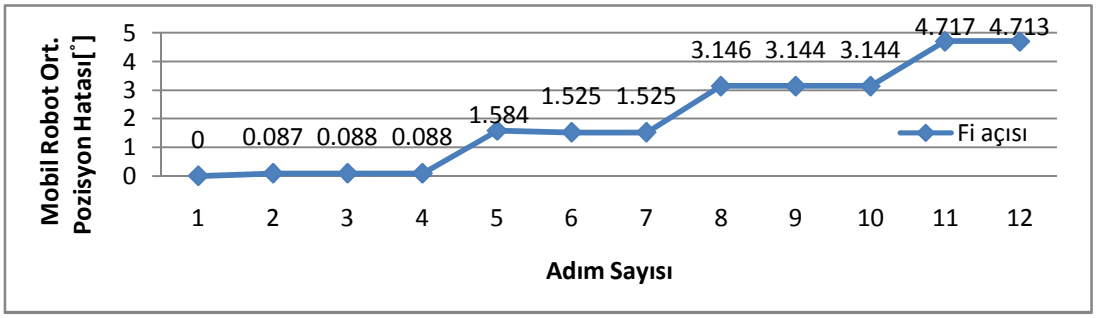


Şekil 4.14 : Bir parçacığın tamamlanmış bir haritası ve her adımda gözlem yapılmış robot konumları



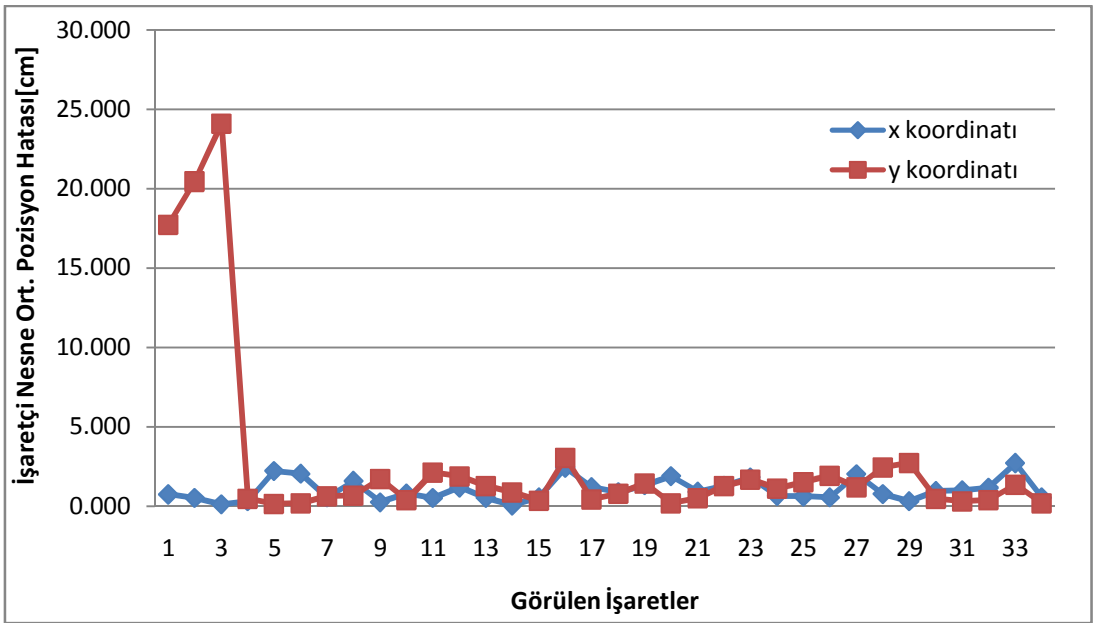
Şekil 4.15 : Standart kriterler için Kartezyen koordinatlarda mobil robot ortalama konum hatası

Çizelge 4.11'e göre yapılmış simülasyon sonuçlarına bakıldığı zaman, her adımda mobil robotun bulunduğu konum hatası (bütün parçacıkların ortalaması alınarak) x,y ve yönelim açısı Φ olarak şekil 4.15-16'da gösterilmiştir. X ve y koordinatlarındaki hatalara bakıldığında ilk adımda rastgele robot konumu belirlemeden dolayı oluşan robot konum hatalarının büyüklüğü göze çarpmaktadır. Daha sonraki adımlarda , MCL'na oluşturulan öneri dağılımı ile aracın bulunması gereken konumlarının hatası 2.1 cm'i geçmemiştir. Bu sırada mobil robotların kovariyans matrisleri önem taşımamaktadır, çünkü araç konumlarının belirlenmesinde doğrudan aracın yeri belirtilmiştir. Öte yandan, aracın açı hatasına bakarsak ilk adımda bütün konumlar $0°$ verilmiştir. Ancak ilerledikçe artan bir açı hatası görülmektedir. Bu açı hatası 5., 9. ve 11. adımlarda göze çarpmaktadır. Bu adımlar aracın $90°$ 'lik dönüş yaptıktan sonra adımlar. Dönüşten sonra yapılan öneri dağılımlarında böyle bir artış gözlenmiştir. Bu hatayı gidermek için, robot döngüsünü tamamladığını anladığında yendien örnekleme yapılmalı ve döngünün tamamlandığının anlaşılmasını sağlayacak ek algoritmalar kullanılmalıdır.



Şekil 4.16 : Standart kriterler için mobil robotta ortalama açı hatası

Şekil 4.17'de ilgili simülasyonun 17 numaralı parçacığının haritalama işlemi sonucunda gördüğü işaretçi nesnelerin belirsizlik çemberlerinin merkezleri ile gerçek işaretlerin bulunduğu noktalar arasındaki fark gözükmektedir. Dikkat edilmesi gereken nokta, ilk 4 işaretin konum hatalarının büyüklüğüdür. Bu parçacık ilk adımda görülen parçaları konumlarını iyi tahmin edememiştir. Bunun sebebi işaretlerin çok önce bir kere gözlemlenmiş olmasından kaynaklanır. Yeniden işaretler görülürse hataları azalacaktır, yoksa hatalar gitgide büyüyecek, parçacığın önem ağırlığını azaltacak ve parçacığın silinmesini mümkün kılacaktır. Çizelge 4.12'de grafiğe sığmadığından fark değerleri aşağıda verilmiştir.

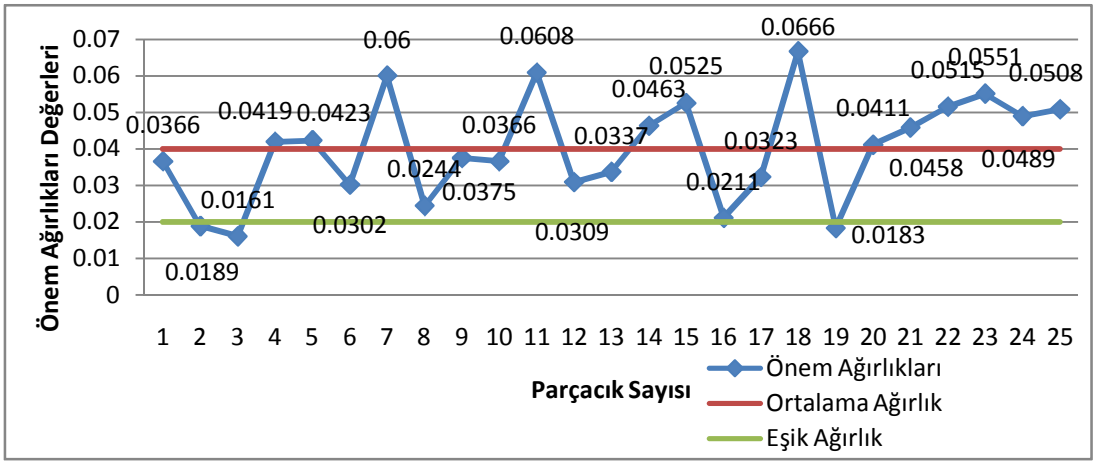


Şekil 4.17 : Standart kriterler için Kartezyen koordinatlarda işaretçi nesnelerin konum hatası

Çizelge 4.12 : Şekil 4.17'de bulunan 34 işaretçi nesnenin cm. olarak fark değerleri

1_6		7_12		13_18		19_24		25_30		31_34	
x	y	x	y	x	y	x	y	x	y	x	y
0,737	17,709	0,549	0,594	0,517	1,261	1,321	1,401	0,609	1,508	0,982	0,287
0,510	20,449	1,579	0,651	0,050	0,854	1,885	0,169	0,539	1,913	1,157	0,349
0,110	24,101	0,239	1,691	0,546	0,314	0,888	0,478	2,009	1,155	2,705	1,348
0,303	0,451	0,780	0,360	2,420	3,040	1,274	1,243	0,760	2,409	0,527	0,149
2,212	0,125	0,511	2,100	1,188	0,420	1,804	1,667	0,303	2,706		
2,035	0,147	1,175	1,848	0,885	0,751	0,639	1,099	0,951	0,447		

Şekil 4.18'deki grafikte simülasyon sonuçlandıktan sonra parçacıkların bütün adımlardaki ortalama önem ağırlıklarını gösterilmiştir. Yukarıdaki grafik ile karşılaştıracak olursak şekil 4.17'deki parçacığın 25 parçacık arasından ortalamanın altında yer almaktadır. 1 kere yeniden örneklemenin gerçekleştiği simülasyonda haritalamada en çok başarılı olan parçacıkların 7,11,15 ve 18 numaralı parçacıklar olduğunu görürüz. En başarısız olanlar ise 2,3,16 ve 19 numaralı parçacıklardır. Grafik tek bir adım olarak düşünülürse parçacıkların,denklem 4.78'deki n_{eff} değeri parçacık sayısının yarısından küçük olursa sisteme yeniden örnekleme uygulanır. Grafikteki belirtilen ortalama değerin yarısı olan eşik değeri ise basit,sistematik yeniden örneklemede örnekleme eşiği olarak kullanılır.



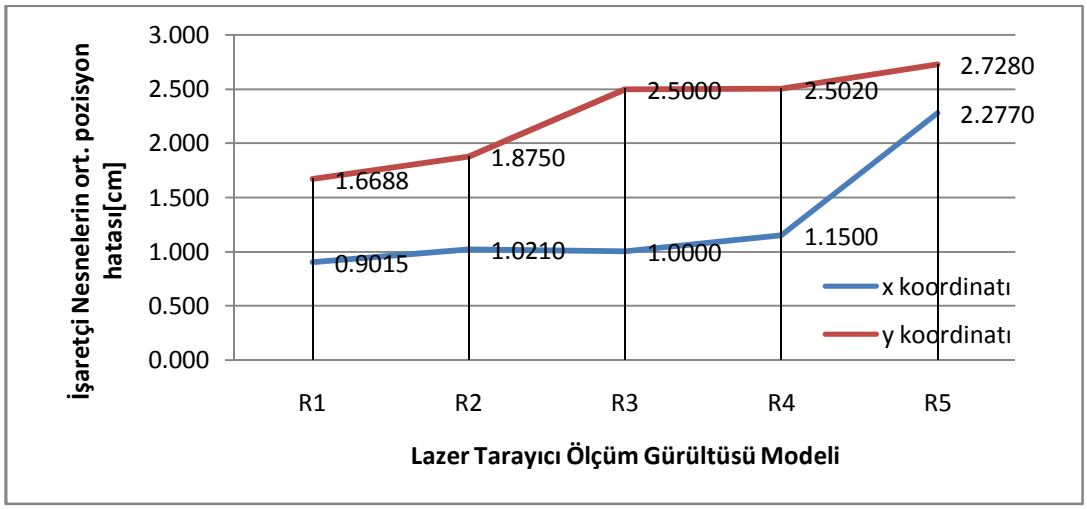
Şekil 4.18 : Simülasyon sonundaki her parçacığın ortalama önem ağırlığı

Karşılaştırma kriterleri olarak gürültüleri alırsak sistemde lazer ölçüm gürültüsü ve süreç gürültüsü olmak üzere toplam iki ana gürültüden bahsedilebilir. İki ana gürültü de doğrudan RBPF içinde GKF'yi etkilemektedir. MCL algoritmasını doğrudan etkileyen ise maksimum odometri hatasıdır. Maksimum odometri hatası her yönde 5 cm. olarak alınmıştır.

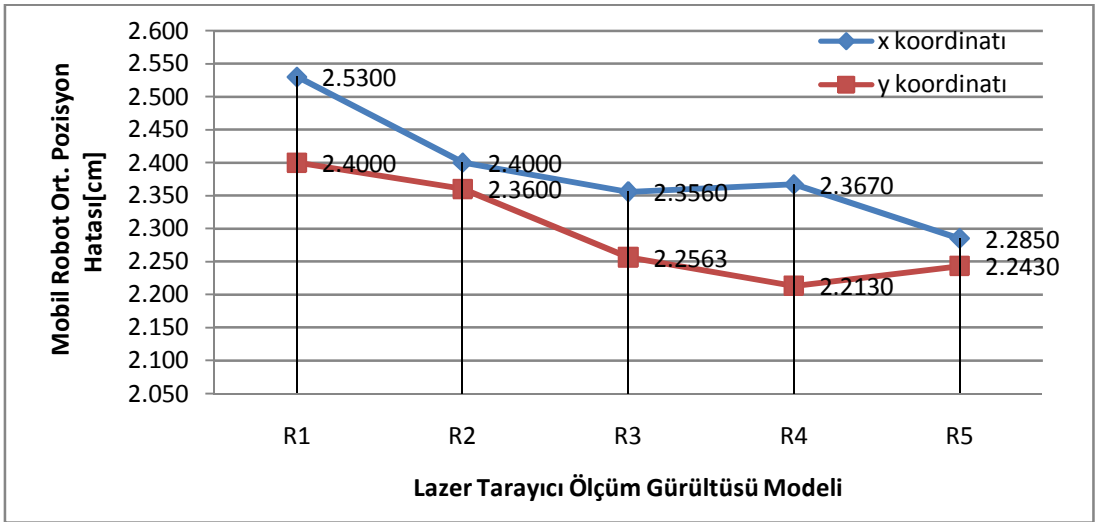
Çizelge 4.13 : Kullanılan Lazer Tarayıcısı Gürültü Matrisleri

R_1	R_2	R_3	R_4	R_5
$\begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix}$	$\begin{bmatrix} 0.3 & 0 \\ 0 & 0.08 \end{bmatrix}$	$\begin{bmatrix} 0.58 & 0 \\ 0 & 0.16 \end{bmatrix}$	$\begin{bmatrix} 1.16 & 0 \\ 0 & 0.32 \end{bmatrix}$	$\begin{bmatrix} 2.9 & 0 \\ 0 & 0.8 \end{bmatrix}$

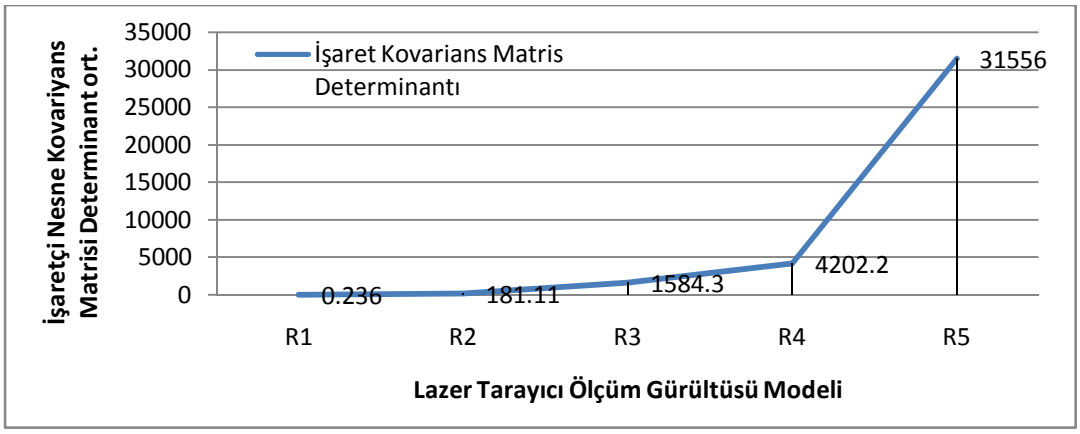
Lazer ölçüm gürültüsü değiştirilerek işaretçi nesnelerin beklenen değerlerinin, mobil robot konumlarının ve işaretçi nesne kovariyans matrislerinin determinantı gözlemlenmiştir. Değişimlerde, lazer tarayıcı modeli gürültü kovariyans matrisinin standart değerin katları biçiminde veya ihmal edilecek derecede değiştirilmiştir. Kullanılan matrisler çizelge 4.13'te verilmiştir. Ortaya çıkan sonuçlara göre, gürültü arttırılarak işaretçi nesnelerin beklenen değerler matrisi elemanlarını oluşturan x ve y konumlarının hatalarında en büyüğü 1.37 cm. olan ufak artışlar şekil 4.19'da gözlemlenmiştir. Şekil 4.20'de araç konumları için yapılan simülasyonlarda büyük bir monotonik değişiklikler gözlenmemiştir. Ancak en büyük artış işaretçi nesnelerin kovariyans matrislerinin determinantlarının ortalama değerlerinde gözlemlenmiştir. Şekil 4.21'de işaretçi nesnelerin belirsizliklerinin büyüyen gürültü matrisi ile büyük bir hızla arttığı gözlemlenmiştir.



Şekil 4.19 : Değişen lazer tarayıcı gürültüsü için işaretçi nesne konum hatası grafiği

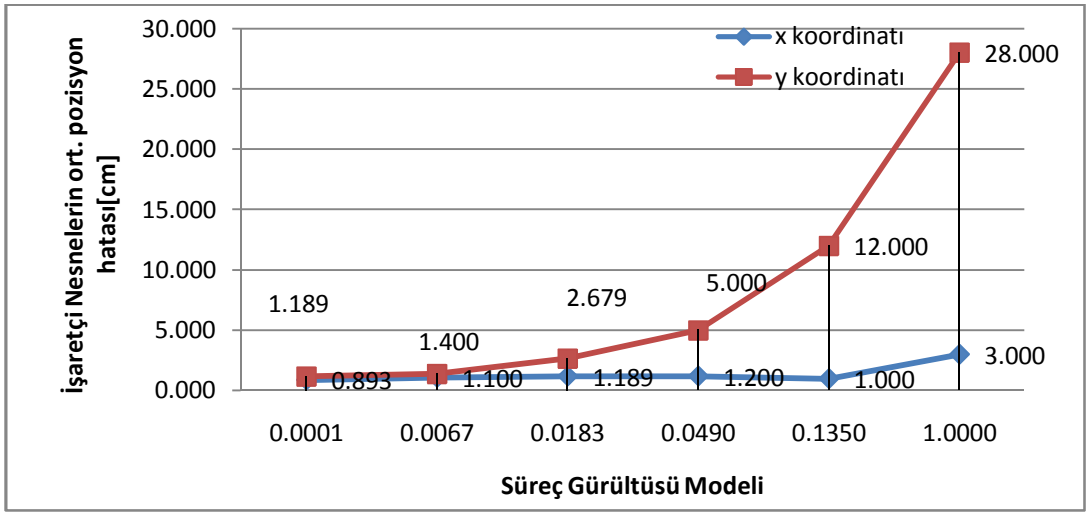


Şekil 4.20 : Değişen lazer tarayıcı gürültüsü için mobil robot konum hatası grafiği

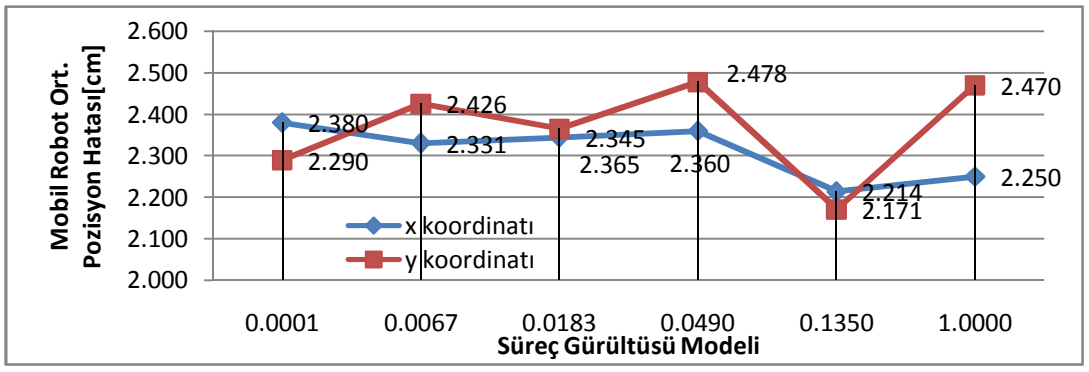


Şekil 4.21 : Değişen lazer tarayıcı gürültüsü için işaretçi nesne kovariyans matrislerinin determinantlarının ortalaması

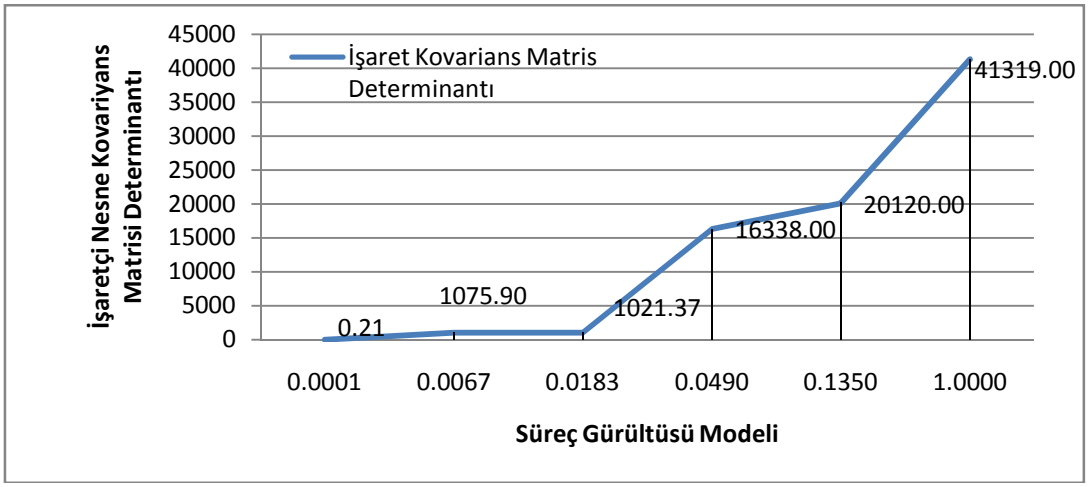
Süreç gürültüsü değiştirilerek işaretçi nesnelerin beklenen değerlerinin, mobil robot konumlarının ve işaretçi nesne kovariyans matrislerinin determinantı gözlemlenmiştir. Değişimlerde, süreç gürültü kovariyans matrisinin standart değerinin katları biçiminde veya ihmal edilecek derecede değiştirilmiştir. Ortaya çıkan sonuçlara göre, gürültü arttırılarak işaretçi nesnelerin beklenen değerler matrisi elemanlarını oluşturan x ve y konumlarının hatalarında en büyüğü 16 cm. belirgin artışlara şekil 4.22'de rastlanmıştır. Süreç gürültüsünün işaretçi nesnelere ve belirsizliklerine daha fazla etki ettiği açıktır. Şekil 4.23'de araç konumları için yapılan simülasyonlarda büyük bir monotonik değişiklikler gözlenmemiştir. Ancak belirgin bir artış ölçüm gürültüsündeki gibi işaretçi nesnelerin kovariyans matrislerinin determinantlarının ortalama değerlerinde gözlemlenmiştir. Şekil 4.24'te işaretçi nesnelerin belirsizliklerinin büyüyen gürültü matrisi ile büyük bir hızla arttığı gözlemlenmiştir.



Şekil 4.22 : Değişen süreç gürültüsü için işaretçi nesne konum hatası grafiği

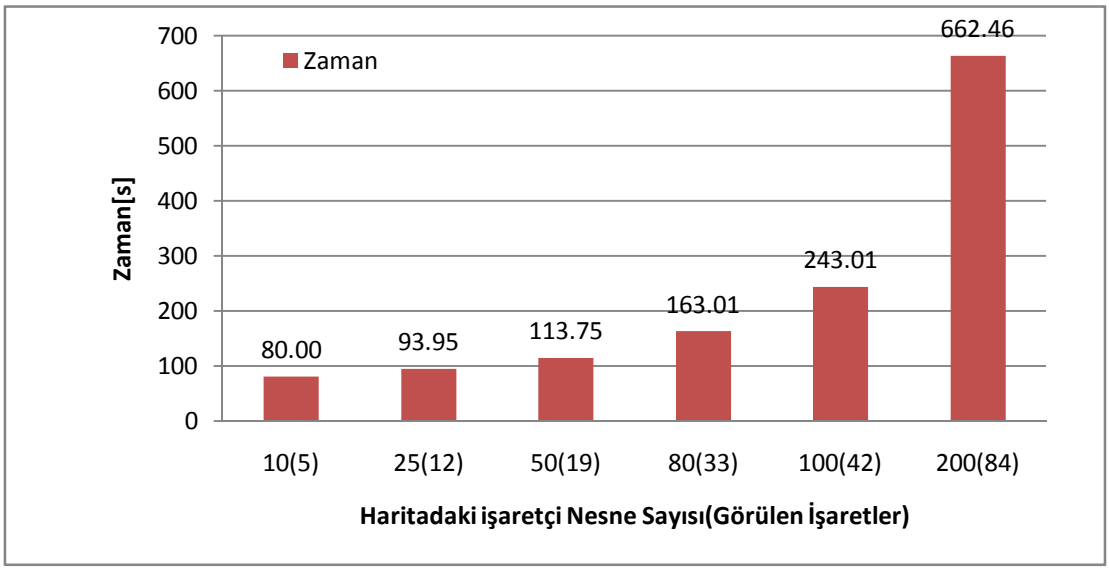


Şekil 4.23 : Değişen süreç gürültüsü için mobil robot konum hatası grafiği



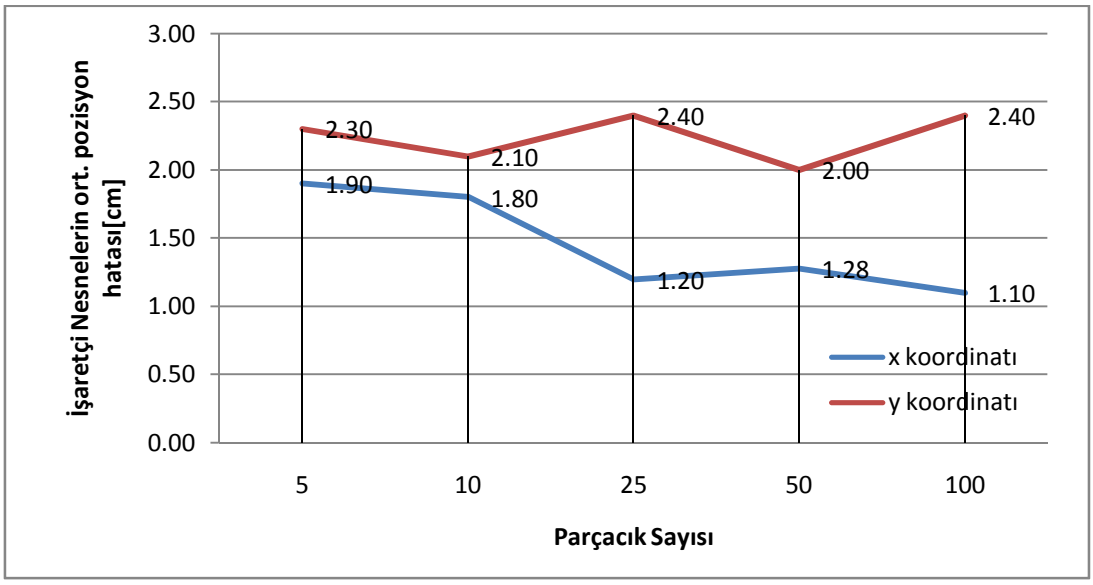
Şekil 4.24 : Değişen süreç gürültüsü için işaretçi nesne kovariyans matrislerinin determinantlarının ortalaması

SLAM probleminde işlem zamanları önemli bir yer tutmaktadır. Bu sebepten ötürü, simülasyonun işlem zamanlarının hesaplanması gerekirdi. 12 adımlık simülasyon haritası değişik sayılardaki işaretçi nesnelerle doldurularak işlem zamanları kontrol edildi. Şekil 4.25'te haritadaki işaret sayısı ile beraber her parçacığın ortalama kaç işaret gözlemlendiği ve işlem yaptığı dikkate alınmıştır. Sonuç olarak haritadaki işaretlerin artması RBPF filtresinin çalışmasını önemli ölçüde yavaşlatmaktadır. Bu sonuç geniş alanlarda SLAM probleminin çözülmesinin güçlüğünü desteklemektedir. Simülasyon sırasında harita çizimi ve bazı istatiksel hesaplar zamanlamanın içinde yer almıştır. Bu yüzden, gerçek zamanlı çalışmada RBPF işlem zamanların daha düşük çıkacağı düşünülmektedir.

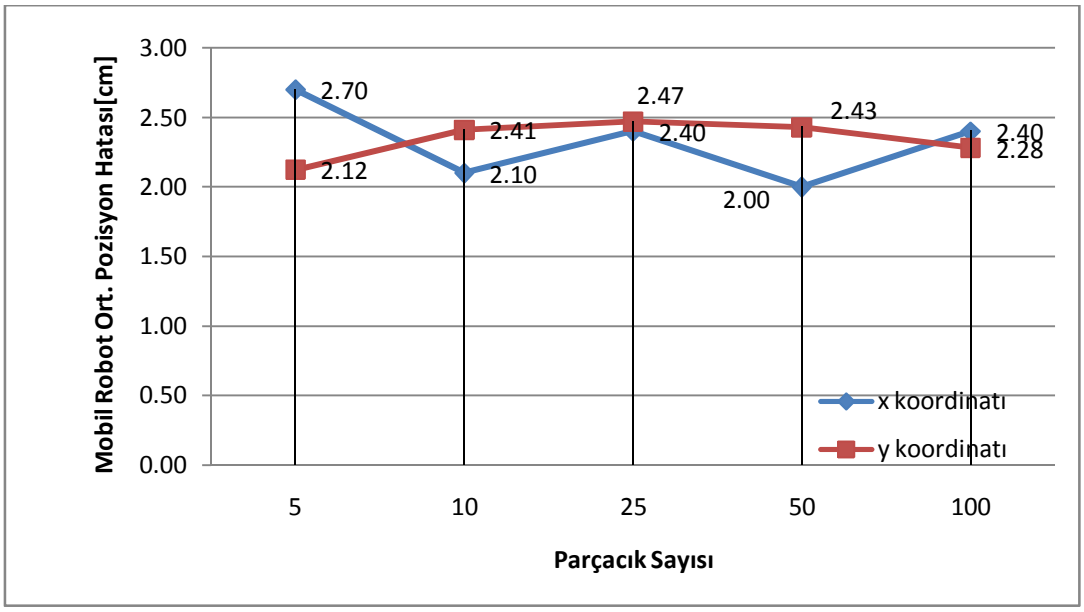


Şekil 4.25 : Değişen işaretçi nesne sayısına göre oluşturulmuş zaman grafiği

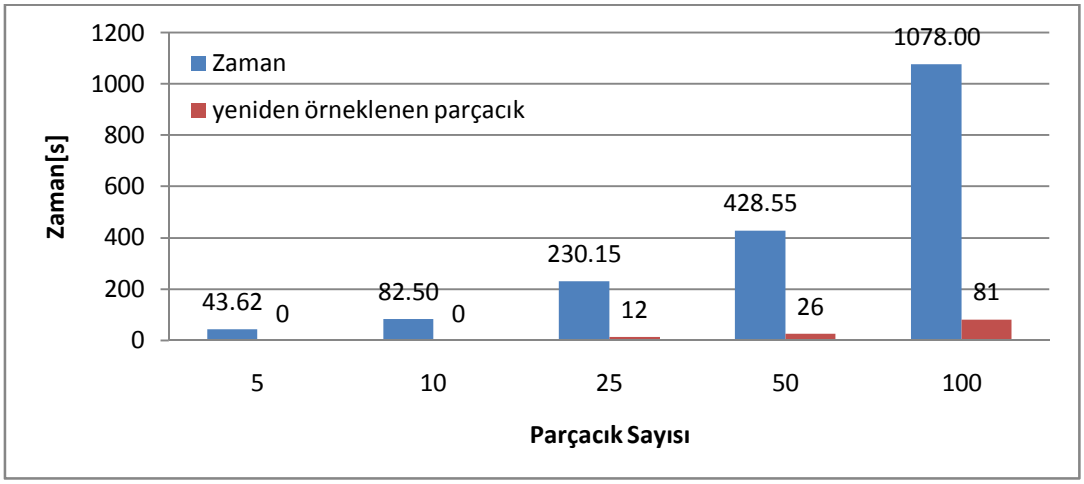
Parçacık sayısı değiştirilerek RBPF filtresine etkisi gözlenmiştir. Bu gözlem, işaretçi nesnelerin beklenen değer konum hataları, araç konum hataları ve RBPF işlem zamanı konularında yapılmıştır. Sonuçlara göre, şekil 4.26'da artan parçacık sayısı ile işaretçi nesne beklenen değer konum hatalarında büyük değişiklikler gözlenmemiştir. Şekil 4.27'de de araç konum hatalarında büyük hata değişiklikleri yoktur. Ancak şekil 4.28'de gözüken zaman kıyaslaması sırasında ciddi zaman değişimleri gözlemlenmiştir. Beklenildiği üzere zamana etki eden bir unsur olduğundan dolayı yeniden örneklenen parçacık sayısı da sonuçlara dahil edilmiştir.



Şekil 4.26 : Değişen parçacık sayısı için işaretçi nesne konum hatası grafiği



Şekil 4.27 : Değişen parçacık sayısı için mobil robot konum hatası grafiği

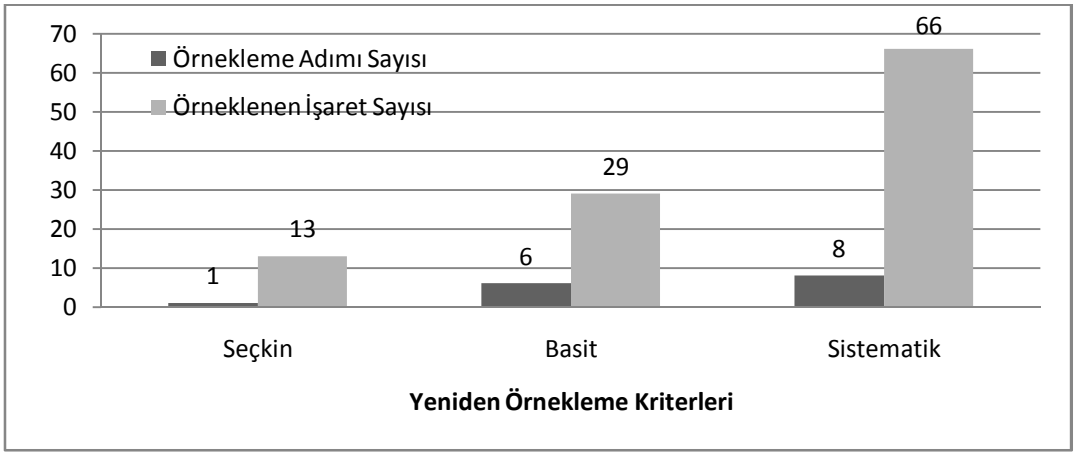


Şekil 4.28 : Değişen parçacık sayısına göre oluşturulmuş zaman grafiği

Sonuçlara bakıldığında açık olarak bu tipte bir haritanın çok yüksek bir parçacık sayısına ihtiyaç duymadığı söylenebilir. Oluşturulan filtre az sayıda parçacık ile de olsa, işaretçi nesnelerin beklenen değerlerinde ve mobil araç konumlarında uygun bir hata ile işlevini yapmaktadır. 25 ile 50 parçacık arası hem şekil 4.26'daki hatalar dikkate alındığında uygundur, hem de az parçacık sayısı seçilerek örnek sayısı çok azaltılmamış olur. Bunun sebebi, bölüm 4.7.3 bölümünde bahsedildiği gibi sürekli aynı yerlerin haritalamasında yüksek önem ağırlığına sahip örnekler sürekli düşüklerin önem ağırlıklı olanların yerine kopyalanabilir ve parçacık bozulmaları olur, çoklu hipotezler azalır, yanlış veri eşleştirmeleri yapılabilir.

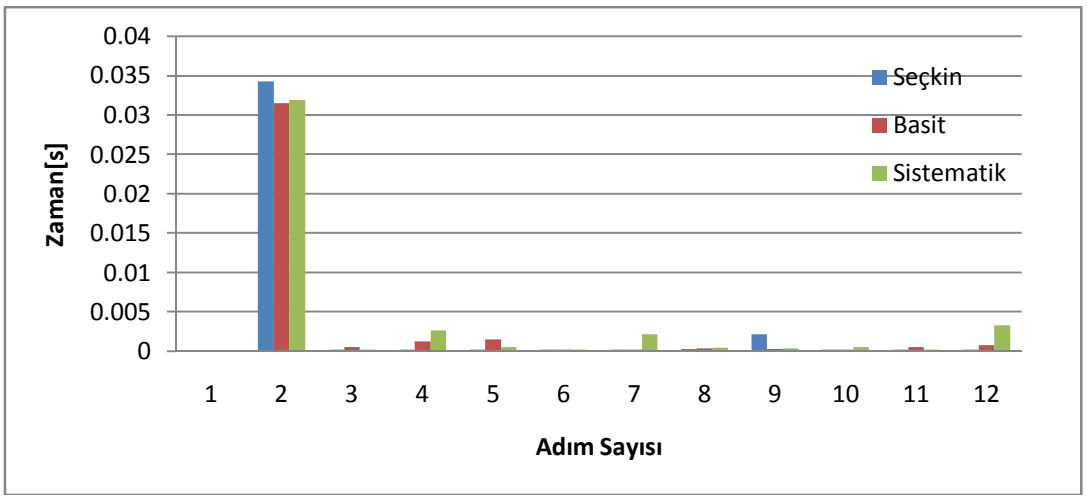
Parçacık filtresinin önemli özelliklerinden biri her adımına değişik algoritmalar eklenebilmesidir. Yeniden örneklemede hem yanlış denebilecek yollar ve haritalar silinir, hem de parçacıklarına harcanacak zamandan kazanılır. Son olarak yeniden örnekleme adımındaki algoritmalar değiştirilerek yeniden örnekleme algoritmaları arasındaki başarı test edilmiştir. Yapılan testler, zaman değişimi, örnekleme sayısı, işaretçi nesnelerin beklenen değerlerindeki hata değişimleri konuları olmuştur.

Şekil 4.29'a göre yeniden örnekleme işleminin en fazla uygulandığı algoritmalar sırasıyla sitematik, basit, seçkin(adaptif) algoritmalarıdır. Yapılan testlerde sistematik ile basit örnekleme adımı sayısı olarak yakın çıksa da sistematik algoritmadaki problem başlangıç eşiğinin rastgele yaratılmasıdır. Çizelge 4.9'daki algoritmada bu rastgele sayı ne kadar ufak olursa o kadar çok yeniden örnekleme yapılır. Bu sayının büyük olduğu algoritmalarda ise daha az yeniden örnekleme uygulanmaktadır. Basit yeniden örneklemede her eşik değerini geçildiğinde kopyalama prosedürü başlar.



Şekil 4.29 : Farklı yeniden örnekleme algoritmalarına göre örneklenen adım ve işaret sayıları

Şekil 4.30'da her adımda algoritmalara harcanan zaman gözükmektedir. İlk adımda sadece işaretlere öngörü uygulandığından dolayı önem ağırlıklarının hesaplanması ve yeniden örneklemeye ait veri tabanlarının oluşumu 2. adıma rastlamaktadır. Bu adımdaki zaman kaybı aşikardır. Diğer adımlar incelendiğinde kopyalama işlemlerinin beklenenden düşük sürdüğü gözlemlenmiştir. Adımlardaki belirgin zaman farklarını yaratan işlem yeniden örneklemedir. Hangi örneklemenin hangi adımda olduğu çizelge 4.14'te verilmiştir.

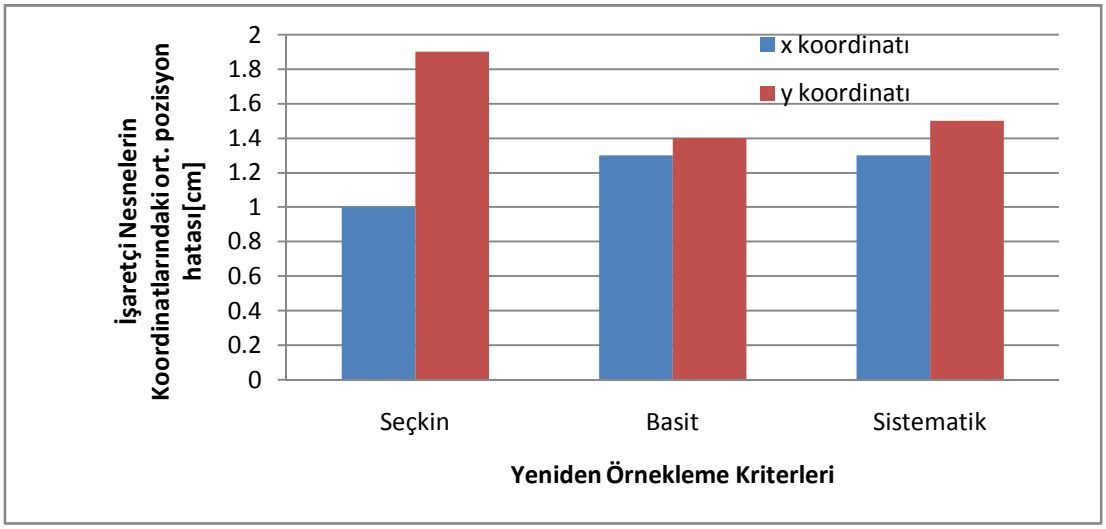


Şekil 4.30 : Farklı yeniden örnekleme algoritmalarına göre adımlarda yeniden örneklemeye harcanan zamanlar

Çizelge 4.14 : Yeniden örneklemenin yapıldığı adımlar

	1	2	3	4	5	6	7	8	9	10	11	12
Seçkin									X			
Basit			X	X	X			X			X	X
Sistematik		X		X	X		X	X	X	X		X

Yeniden örnekleme algoritmaları işaretçi nesne beklenen değer hata değişimleri açısından şekil 4.31'de incelendiğinde 3 hesaplamada da en büyük gerçek işaretten uzaklık 2cm.'yi geçmemiştir, bu sonuç daha önceki başarı kriterleri karşılaştırmalarına benzer bir sonuçtur.



Şekil 4.31 : Farklı yeniden örnekleme algoritmalarına göre işaretçi nesne konum hatası grafiği

5. RBPF SİMÜLASYONU SONUÇLARI VE TARTIŞMA

Yapılan simülasyonlar ve bölüm 4.7'deki grafikler başarı kriterlerine göre incelendiğinde, haritalamada, ölçüm ve süreç gürültüsünün işaretçi nesnelerin kovariyans matris değerlerini büyüttüğünü söyleyebiliriz. Algoritması GKF'den farklı olsa da, RBPF kendi içinde GKF içerdiğinden GKF'yi etkileyen tüm gürültü tipleri kendisini de etkileyecektir. Ölçüm gürültüsünde algılayıcının kalitesi, süreç gürültüsün de ise sürece etki eden sistemlerin tasarım kalitesi filtrenin başarısına etki etmektedir.

Filtre başarılı bir şekilde artan işaret ve parçacık sayısına rağmen hesaplama yapabilmiştir. $16m^2$'lik bir alanda, 16 metrelik yol boyunca, işaretçi nesne sayısı artımlarına rağmen, standart çalışmada işaretçi nesne hatalarını birkaç santimetre ile sınırlamayı başarmıştır. Ancak artan parçacık sayısı bu sınırı azaltamamıştır. Her çevre için yüksek sayılı parçacıklı filtre kullanmak kullanıcıya ek bir avantaj sağlamayacaktır, aksine işlem zamanını yükseltecektir. Birkaç yüz tane parçacık kullanarak haritalama işlemi gürültünün ve işaretin çok olduğu dış ve geniş çevrelerde tercih edilmelidir.

Mobil robotun konumunu tahmin etme işleminde MCL denklemlerindeki işaretçi gözlemlerinden yararlanılsa da işaretçi nesne konumlarını etkileyen süreç ve ölçüm gürültüsünün tahmin konumlarına etkisi çok azdır. Mobil robot konumuna etki eden asıl parametre araçtaki odometri hataları olacaktır. Ayrıca MCL algoritmasının bir noktada kümelenmeye benzeyen dağılımından farklı önem dağılımlarının gözlemlenmesi, ancak etrafı kapalı veya sınırlandırılmış ortamlarda test edilmesi ile mümkün olacaktır.

Yeniden örnekleme algoritmaları kendi aralarında test edilmiştir. Seçkin önem örneklemesi bu konuda diğer algoritmalara göre daha başarılıdır. Çünkü, ufak bir alanda sadece bir kere yeniden örnekleme yaparak diğer algoritmalar kadar işaretçi nesne konumlama hatası yapmıştır ve işlemci zamanını azaltmayı başarmıştır. Diğer algoritmalar ufak alanlarda kullanılmaya elverişli olsa da, artan parçacık sayısı ile geniş alanlarda her adımda yeniden örnekleme yapabilirler, işlem yükünü

arttırabilirler. Öte yandan, sürekli yeniden örnekleme yapmak harita çeşitliliğini azaltır ve doğru haritaların bozularak yanlış haritalara dönüşmesine yol açabilir.

RBPF SLAM problemini araç pozisyonu ve haritalama olarak ikiye ayırabilmiştir. Her bir işarete uygulanan GKF, işlem zamanını arttırsa da yanlış bir işaretin konumunun tespiti filtreyi ardışıl hatalara götürmez. Bir parçacıktaki hataların artımı diğer bir parçacığın yeniden örneklenmesi ile telafi edilir. Böylece, ek bir veri eşleştirme algoritması kullanılmasından parçacıklar arası çoklu veri eşleştirmesi yapılmış olur.

Yapılan çalışmalara göre, RBPF SLAM probleminde GKF'ye iyi bir alternatiftir. Parçacık filtreleri, SLAM 'in olduğu her konuda yer alabilir. Sualtı tabanı haritalaması, enkaz bölgelerinin araştırılması, AGV tipli endüstriyel taşıma, Mars yüzeyinin keşfi, askeri uygulamalar en çok kullanılacağı alanlardır.

Bundan sonraki çalışmalarda aracın döngüyü tamamladıktan sonraki davranışları ve haritalamada kullanılan ızgara tipi haritalamanın RBPF'ye uygulanması tercih edilebilir.

KAYNAKLAR

[1] Montemerlo, M., Thrun, S., Koller, D. ve Wegbreit, B. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem.

[2] Grisetti, G., Stachniss, S. ve Burgard, W. FastSLAM: Improving Grid-based SLAM with Rao-Blackwellized ParticleFilters by Adaptive Proposals and Selective Resampling, *Makale*

[3] Gao, S. ve Lotun, R. FastSLAM with Look-ahead RBPF

[4] Hahnel, D., Burgard, W., Fox, D. ve Thrun, S. An Efficient FastSLAM Algorithm for Generating Maps of Large-Scale Cyclic Environments from Raw Laser Range Measurements

[5] Grisetti, G., Tipaldi, G.D., Stachniss, S., Burgard, W. ve Nardi, D. Fast and Accurate SLAM with Rao-Blackwellized Particle Filters

[6] Thrun, S., 2006. Robotic Mapping: A Survey. *School of Computer Science Carnegie Mellon University*

[7] Newman, P.M., 2006. EKF Based Navigation and SLAM, *SLAM Summer School 2006 Background Materials and Notes* , Oxford University, Oxford.

[8] SICK, 2006. Telegrams for Configuring and Operating the LMS2xx Laser Measurement Systems

[9] SICK, 2006. LMS200/211/221/291 Laser Measurement Systems, Technical Description

[10] Bailey, T. ve Durrant-Whyte, H.F., 2006. Simultaneous Localization and Mapping (SLAM): Part I, *IEEE Robotics and Automation Magazine,* **June 2006**

[11] Bailey, T. ve Durrant-Whyte, H.F., 2006. Simultaneous Localization and Mapping (SLAM): Part II, *IEEE Robotics and Automation Magazine,* **September 2006,** 108-117.

[12] Çelebi, E., 2007. İTÜ *Detection and Estimation Ders Notları*

[13] Welch, G. ve Bishop, G., 2001. An Introduction to the Kalman Filter. SIGGRAPH 2001 University of North Carolina at Chapel Hill, Department of Computer Science Chapel Hill, NC 25799-3175, sayfa 19-25.

[14] Kavak, D., 2007. İnsansız Kara Araçları Navigasyonunda Genişletilmiş Kalman (EKF) ve Sıkıştırılmış Kalman Filtre (CEKF) Tabanlı SLAM

Yöntemlerinin Geliştirilmesi ve Karşılaştırılması. İTÜ Kontrol Kumanda Sistemleri Yüksek Lisans Tezi. Sayfa 36

[15] Thrun, S., Fox, D., Burgard, W. ve Dellaert., F. Robust Monte Carlo Localization for Mobile Robots

[16] Fox, S., 2004. Monte Carlo Localization COMP/NEUR 484

[17] Bailey, T., Nieto, J. ve Nebot, E., 2006. Consistency of the FastSLAM Algorithm. *ICRA 2006*

[18] Bolic, **M.,** 2004. Architectures for Efficient Implementation of Particle Filters, Presented Dissertion for PhD, Electrical Eng., Stony Brook University.

[19] Lim, F.L., Leoputra,W. ve Tan, T., 2007. Non-overlapping Distributed Tracking System Utilizing Particle Filter.

EKLER

EK A.1 : SICK LMS200 MATLAB RS232 veri haberleşme program kodu

EK A.1

```
clc;
LMS=serial('COM2','BaudRate',9600,'DataBits',8);
A= [2 0 2 0 48 1 49 24];
B= [6 2 128 110 1 176 181 0];
clear RcvB;
flagg=0;
%x=[]; rho=[]; tita=[];
mvalues=zeros(181,1);
k = strcmp('open',LMS.Status)
if k == 0
    fopen(LMS);
end

fwrite(LMS,A);

while flagg == 0
    if LMS.BytesAvailable == 373
        RcvB = fread(LMS,LMS.BytesAvailable);
        flagg = 1;
    end
end

flagg = 0;
fclose(LMS);

if RcvB(1:8,1) == B'
    for n=1:181
                RcvD(n,:)=cat(2,RcvB((2*n)+7,1),RcvB((2*n)+8,1));
                bRcvD(n,:)=de2bi(RcvD(n,2),'right-msb',8);

                for m=8:12

                  mvalues(n,:)=((2^m)*bRcvD(n,(m7))+mvalues(n,:));
                end
                mvalues(n,:)=mvalues(n,:)+RcvD(n,1);
    end
end

for n=1:181

    if bRcvD(n,6) | bRcvD(n,7) | bRcvD(n,8) == 1

        mvalues(n,2) = 1;

    end

end
%for n=1:180

% polar((pi/180)*n,mvalues(n,1)/100,'--r')
% hold on

%end
rvalues = mvalues;
```

```
for n=1:180

    rvalues(n,3)=n;

    if rvalues(n,2) & rvalues(n+1,2) & rvalues(n+2,2) &
rvalues(n+3,2) & (n < 178) == 1

        rvalues(n+1,2) = 0;
        rvalues(n+2,2) = 0;
        rvalues(n+3,2) = 0;
    else if rvalues(n,2) & rvalues(n+1,2) & rvalues(n+2,2) & (n <
179 )== 1

            rvalues(n+1,2) = 0;
            rvalues(n+2,2) = 0;
        else if rvalues(n,2) & rvalues(n+1,2) & (n < 180) == 1

                rvalues(n+1,2) = 0;
        end

    end
    end
end
```

Şekil A.1 : SICK LMS200 MATLAB RS232 veri haberleşme program kodu

ÖZGEÇMİŞ

Ad Soyad: Ali Kuleli

Doğum Yeri ve Tarihi: 1982, İstanbul

Adres: Kartal , İstanbul

Lisans Üniversite: Yıldız Teknik Üniversitesi Elektrik Müh., 2005

Yıldız Teknik Üniversitesi Elektronik ve Haberleşme Müh. 2006

Çalıştığı Kurum : KaleAltınay Robotik ve Otomasyon A.Ş. , İstanbul

Onur ve Başarılar:

-Yüksek Onur Öğrencisi Yıldız Teknik Üniversitesi, Elektrik ve Elektronik Fakültesi, 2006

-Elektrik Müh. Bölüm Birinciliği Yıldız Teknik Üniversitesi, 2005

-Elektrik ve Elektronik Fakültesi Bölüm İkinciliği, 2005

-Saint-Benoit Fransız Lisesi Teşekkür Belgesi(1997-1998 Güz, 1997-1998 Bahar,1998-1999 Güz, 1998-1999 Bahar,1999-2000 Bahar, 1999-2000 Güz, 2000-2001 Güz)

Yayın Listesi:

Kuleli, A., 2006. Ev Güvenlik Sistemi bildirisi *ASYU-INISTA Haziran 2006* İstanbul, Türkiye