**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ**

# İNSANSIZ KARA ARAÇLARI NAVİGASYONUNDA GENİŞLETİLMİŞ KALMAN (GKF) VE SIKIŞTIRILMIŞ GENİŞLETİLMİŞ KALMAN FİLTRE (SGKF) TABANLI SLAM YÖNTEMLERİNİN GELİŞTİRİLMESİ VE KARŞILAŞTIRILMASI

**YÜKSEK LİSANS TEZİ**

**Elektrik-Elektronik Müh. Deniz KAVAK**

**Anabilim Dalı : ELEKTRİK MÜHENDİSLİĞİ**

**Programı : KONTROL VE OTOMASYON**

**OCAK 2008**

**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ**

# İNSANSIZ KARA ARAÇLARI NAVİGASYONUNDA GENİŞLETİLMİŞ KALMAN (GKF) VE SIKIŞTIRILMIŞ GENİŞLETİLMİŞ KALMAN FİLTRE (SGKF) TABANLI SLAM YÖNTEMLERİNİN GELİŞTİRİLMESİ VE KARŞILAŞTIRILMASI

**YÜKSEK LİSANS TEZİ**

**Elektrik-Elektronik Müh. Deniz KAVAK**
**(504041126)**

**Tezin Enstitüye Verildiği Tarih : 24 Aralık 2007**
**Tezin Savunulduğu Tarih : 28 Ocak 2008**

**Tez Danışmanı : Prof.Dr. Hakan TEMELTAŞ**
**Diğer Jüri Üyeleri : Prof.Dr. Metin GÖKAŞAN**
**Doç.Dr. Ata MUĞAN**

**OCAK 2008**

## ÖNSÖZ

Tez danışmanım Prof. Dr. Hakan TEMELTAŞ'a bu çalışmada verdiği destek, öneri ve tavsiyeleri, çalışmam sırasındaki cesaretlendirici ve anlayışlı tutumundan dolayı teşekkür ederim. Çalışma şevki ve azmini örnek aldığım değerli hocamla çalışmak benim için gurur verici bir paylaşım olmuştur.

Çalışmam süresince her türlü desteğini esirgemeyen, doğumu sırasında bile çalışmalarımı düşünen fedakar eşim Nergizhan KAVAK'a ve doğumuyla bana bir moral ve uğur getiren güzel kızım Zehra Nur KAVAK'a teşekkürü bir borç bilirim.

Hayatım boyunca her konuda beni destekleyip maddi ve manevi karşılıksız özverilerini sunan babam Bilal KAVAK ve annem Fehri KAVAK'a şükranlarımı sunarım.

Tez çalışmam boyunca beni cesaretlendirip her türlü desteğini veren, iş hayatımda çalışma azmini takdirle karşıladığım başmühendisim Mustafa BULDUM'a teşekkürü borç bilirim.

**Ocak, 2008** **Deniz KAVAK**

**İÇİNDEKİLER**

## KISALTMALAR

| | |
|---|---|
| SLAM | : Eş Zamanlı Lokalizasyon ve Haritalama (Simultaneous Localization and Mapping) |
| İKA (UGV) | : İnsansız Kara Aracı |
| GKF (EKF) | : Genişletilmiş Kalman Filtresi |
| SGKF (CEKF) | : Sıkıştırılmış Genişletilmiş Kalman Filtresi |
| BUYK (ICNN) | : Bireysel Uyumlu Yakın Komşuluk |
| BUDB (JCBB) | : Bileşik Uyumlu Dallanma ve Bağlanma |
| SLAF | : Sınırlandırılmış Lokal Altharitalama Filtresi |
| SRAF | : Sınırlandırılmış Rölatif Altharitalama Filtresi |
| MAP | : Maksimum Soncul |
| ÇHİ (MHT) | : Çoklu Hipotez İzleme |
| BSVİ (CCDA) | : Birleşik Sınırlandırılmış Veri İlişkilendirme |
| SZE | : Sabit Zamanlı SLAM |
| GPS | : Genel Pozisyonlama Sistemi |
| BU (IC) | : Bireysel Uygunluk |
| BRU (JC) | : Bileşik Uygunluk |
| YK (NN) | : Yakın Komşu |
| d-r | : mesafe sayacından lokalizasyon hesabı (dead-reckoned) |

## TABLO LİSTESİ

## ŞEKİL LİSTESİ

## SEMBOL LİSTESİ

**x** : Gerçek durum vektörü
$\hat{x}$ **:** Kestirilen durum vektörü
**P** **:** Kestirilen kovaryans matrisi
**Q** **:** Sistem model gürültü kovaryans matrisi
**R** **:** Gözlem modeli gürültü kovaryans matrisi
$\nabla H$ **:** Gözlem modeli jacobian matrisi
$\nabla F$ **:** Sistem modeli kovaryans matrisi

## İNSANSIZ KARA ARAÇLARI NAVİGASYONUNDA GENİŞLETİLMİŞ KALMAN (GKF) VE SIKIŞTIRILMIŞ KALMAN FİLTRE (SGKF) TABANLI SLAM YÖNTEMLERİNİN GELİŞTİRİLMESİ VE KARŞILAŞTIRILMASI

## ÖZET

Eş Zamanlı Lokalizasyon ve Haritalama (SLAM) otonom mobil robot navigasyonunda önemli bir görevdir. Genişletilmiş Kalman Filtre (GKF) tabanlı navigasyon son yirmi yıldır en iyi bilinen ve iyi geliştirilmiş bir tekniktir. GKF denklemlerini işlemek için gereken hesaplama zamanı işaretçi nesne sayısı arttıkça önemli ölçüde artmaktadır. GKF tabanlı SLAM'nin hesaplama problemine çözüm olarak global haritadaki lokal alanı baz alan Sıkıştırılmış Genişletilmiş Kalman Filtresi (SGKF) kullanılmıştır. Gözlemleri haritadaki mevcut nesnelerle ilişkilendirme olarak bilinen veri ilişkilendirme problemi her iki teknik içinde önemlidir. Bu çalışmada SGKF ve GKF tabanlı SLAM'yi veri ilişkilendirme algoritmalarını kullanarak inceledik.

Doğru ve kararlı harita oluşturmak için, güvenilir ve etkin veri ilişkilendirme algoritmalarına ihtiyaç vardır. Biz çalışmamızda Bireysel Uyumlu Yakın Komşu (BUYK) ve Bileşik Uyumlu Dallanma ve Bağlanma (BUDB) veri ilişkilendirme algoritmalarını GKF ve SGKF filtreleri ile karşılaştırmalı olarak inceledik. Otonom insansız kara aracı ortamda dolaşırken haritadaki nesne sayısı zamanla büyümektedir. Büyümüş durum vektörüne ait matris işlemleri ve özellikle inovasyon matrisinin tersi alınırken harcanan zaman GKF tabanlı SLAM'nin gerçek zamanlı navigasyonda uygulanmasını güç kılmaktadır. Bu probleme çözüm olarak sunulan Sıkıştırılmış Genişletilmiş Kalman Filtresi önemli oranda hesaplama problemini çözmektedir. SGKF haritanın tümü yerine aktif durumlar olarak adlandırdığı lokal bölgedeki işaretçi nesnelerle ilgilenmektedir. Lokal haritanın dışında kalan işaretçi nesneler ise pasif durumlar olarak adlandırılmaktadırlar. Önemli bir soru SGKF hesaplama derecesini düşürürken veri ilişkilendirme nasıl etkilenmektedir. Bu çalışmada BUYK ve BUDB algoritmalarının SGKF ve GKF ile nasıl çalıştıklarını inceleyerek karşılaştırmalarını yaptık. Matlab ortamında hazırladığımız çalışma ile diferansiyel araç ve LMS 200 lazer sensörünü modelleyerek karesel bir yörüngede değişik sayıda rastgele atanmış işaretçi nesneler ile GKF ve SGKF tabanlı SLAM algoritmalarının BUYK ve BUDB veri ilişkilendirme algoritmalarıyla beraber simulasyonunu yaptık.

**COMPARISON AND IMPROVEMENT OF EXTENDED KALMAN FILTER (EKF) AND COMPRESSED EXTENDED KALMAN FILTER (CEKF) BASED SLAM METHODS FOR UNMANNED GROUND VEHICLE (UGV) NAVIGATION**

## SUMMARY

The Simultaneous Localization and Mapping (SLAM) methodologies are of an important task in navigation of a mobile robot particularly autonomous capabilities are required. Extended Kalman Filter (EKF) based approaches are well developed over the last two decades. However, enormous computational efforts are required while computation of EKF equations when the number of landmarks are increased inside the robot's environment. One of the solutions of the problem is to use Compressed Extended Kalman Filter (CEKF) since it considers only local area map avoiding large number of variables inside the global map. However those studies do not encounter the data association problem which recognizes features by preventing them from re-assignment in the global map. In our study, we proposed CEKF based SLAM methodology solving data association problem for autonomous mobile robots.

To build correct and robust map, very robust and efficient data association algorithms are needed. We compare Individual Compatibility Nearest Neighbour (ICNN) and Joint Compatibility Branch and Bound (JCBB) data association algorithms with EKF and CEKF. While building a map feature size is growing with the time and map area that the UGV walks around. To implement EKF in real-time navigation system becomes difficult and inappropriate due to high dimension matrix operation and calculation of innovation matrix inversion. Compressed EKF (CEKF) was proposed for the real time navigation system as a solution of computational complexity while map size is growing. In CEKF robot is just dealing with local area of the global map. In that local area we call that features are active states. The other features not in the local area but exist in global area called are passive states. Therefore matrix operations are reduced to the size of active states that matrix are low dimension. The other question is that while we are reducing the matrix with CEKF how the data association will be effected. We investigate how well CEKF works with different data association algorithms such as ICNN and JCBB. We give some simulation results with EKF and CEKF SLAM using ICNN and JCBB data association algorithms in Matlab. For the simulation we used differential drive with wheel odometer and LMS 200 2D laser-range finder sensor. Features are randomly scattered in the global area and differential robot moves a dedicated trajectory with a closed loop.

# 1. GİRİŞ

İnsansız kara araçları konuları yörünge kontrolü için algoritma ve metodlar dizayn etme, engellerden kaçınma, lokalizasyon ve haritalama gibi bir çok konuyu kapsamaktadır. Başarılı bir yörünge planlaması ve navigasyon için aracın lokasyonunun başarılı bir şekilde kestirilmesi ve çevrenin hassas bir şekilde algılanmasına bağlıdır. İnsansız kara araçlarının navigasyonunda çeşitli teknikler kullanılmaktadır. Bunların en temeli kişi tarafından araç üzerinden yada uzaktan kumanda vasıtasıyla kontrol edilmesidir. Fakat araç kendi yolunu belirleyip ortamın haritasını oluşturamaz. Bunun için SLAM olarak bilinen Eş Zamanlı Konum Belirleme ve Haritalama algoritmaları kullanılmaktadır. SLAM'yi üç ana katmandan oluştuğunu varsayarsak bu katmanlardan ilki sensörden k anında çevredeki nesnelere ait bilgileri okuma, bu verileri düzenleme, ikinci katmanı haritadaki nesnelerle yapılan gözlemleri eşleştirme, üçüncü katmanı k-1 anında kestirilen haritayı bu gözlemlerle birleştirerek güncelleme olarak düşünebiliriz. SLAM ile ilgili ilk çalışmalar 1980'li yılların başında ortaya çıkmıştır. [2]

Haritalama için en basit yaklaşım aracın konumunu mesafe sayacından lokalizasyon hesabı bilgisine dayanarak kestirilmesi üzerine kuruludur. Fakat 1999 yılında Tardos, Neira ve Castellanos'un yaptığı SPMap çalışması göstermiştirki uzun süreli çalışmalarda bu yaklaşım çok güvenilir sonuçlar vermemekte, aracın lokasyonundaki belirsizlik elipsleride giderek büyümektedir [12]. Hem ölçümdeki hemde aracın lokasyonundaki belirsizliklerin azaltılması için son on yılda literatürde oldukça fazla sayıda çalışmalar yapılmış ve yapılmaya devam etmektedir. SLAM problemini ilk ele alan çalışmalar 1986 yılında Cheeseman, Crovley, Durrant-Whyte'ın IEEE Robotik ve Otomasyon konferansında sundukları ve olasılıksal metodlar üzerine kurulu olan çalışmalardır. Haritalama ve Lokalizasyon problemlerine teorik-kestirim metodların uygulanması üzerine çalışmışlardır. SLAM konusunda en iyi bilinen çalışma Cheesman ve Smith'in 1988 yılında navigasyonu yapılacak ortamın yapısının ayrık-zamanlı durum uzay gösterimiyle ifade edilmesi temeline dayanan istatistiksel haritalamadır ki Smith bu çalışmayı daha sonra Genişletilmiş Kalman Filtre tabanlı SLAM olarak geliştirmiştir. Gene aynı yıllarda Ayache ve Paugeras

görüntü tabanlı navigasyon, Chatila ve Laumond 1985 yılında, Crovley'de 1989 yılında sonar tabanlı Kalman Filtresi algoritmaları kullanılarak mobil robot navigasyonu üzerinde çalışmalar yapmışlardır.

Genişletilmiş Kalman Filtre tabanlı SLAM temelinde robot lokasyonu ve işaretçi nesnelerin lokasyonunun birleştirilmesiyle oluşan ayrık zamanlı büyüyen durum vektörü bulunmaktadır. Bu durum vektörü sensörden alınan gözlem ve araç modelinin oluşturduğu bilgilerle ardışıl olarak güncellenmektedir. Lokasyonlar kesin bilgiler içermemekte modellerden ve gürültülerden gelen belirsizliklere sahiptir. Belirsizlikler durum vektörünün, hareket modelinin ve sensör gözlemlerinin olasılıksal dağılım fonksiyonu (pdf) ile ifade edilirler. Ardışıl olarak yapılan işlemler neticesindeki güncellemelerle bu olasılıksal dağılım fonksiyonlarının ortalama değer ve kovaryanslarının durumların kestirimi için optimal bir değere yakınsayacağı düşünülür. [2]

SLAM'nin en ciddi problemlerinden biri hesaplama yoğunluğudur. Gerçek zamanlı SLAM'de robotun hareket halindeyken haritalama yapabilmesi gerekmektedir. Eğer kestirim algoritmaları işlemcide çok zaman alırsa robot hareket halindeyken ortamın haritasını doğru çıkaramaz ve bir çok işaretçi nesne algılanamaz. GKF tabanlı SLAM yaklaşımında işlem zamanı tam kovaryans matrisini işlemek için geçen zamana karşılık gelir ki bu zamanla artmaktadır. GKF tabanlı SLAM'da n ortamdaki işaretçi nesne sayısı ve araç lokasyon bilgisi olmak üzere $O(n^2)$'dir[2]. Büyük ortamlarda SLAM'nin hesaplama derecesini azaltacak bir çok çalışma yapılmaktadır. Mevcut bir çok metodta hesaplama problemini haritanın belli bir lokaldeki bögesine indirgeme üzerine kuruludur. 2001 yılında Nebot ve Guivant'ın yaptığı çalışmada Sıkıştırılmış Filtre [9] ve gene aynı yılda Length ve Davison'un yaptığı Erteleme [8] metodları hesaplama derecelerini GKF'ye göre kestirimde herhangi bir hassasiyet kaybı olmadan önemli ölçüde azaltmaktadırlar. Gene aynı yılda Julier ve Uhlmann'ın milyon nesneli harita oluşturma isimli çalışmasında kullandıkları Parçalanmış Kovaryans Kesişim Metodu da hesaplama derecesini önemli ölçüde azaltmakla beraber kestirim hassasiyetini de yitirdiği görülmüştür. 2003 yılında Lui ve Thrun'un dış ortamda SLAM uygulamasında kullandıkları Dağınık Genişletilmiş Bilgi Filtresi ile kapalı çevrim yörüngeleri hariç her adımda sabit bir zamanla yaklaşık bir harita elde edebilmektedir.

Yukarıda anlatılan tüm bu çalışmalar tek bir harita için yapılmıştır. Büyük alanların haritalarında işlemci zamanı çok önemlidir ve yukarıdaki teknikler yetersiz kalmaktadırlar. Tardos, Neira ve Newmann'ın 2002'de sonar verilerini kullanarak yaptıkları çalışmada kullandıkları Lokal Harita Ekleme ve Williams'ın gene aynı yılda doktora tezinde ortaya attığı Sınırlandırılmış Lokal Altharitalama Filtresi (SLAF) [5] metodları bir lokal referans eksenine göre alt haritalardan global stokastik haritalar oluşturmak için ortaya atılmış ve büyük alanlarda oldukça iyi sonuçlar vermişlerdir. Büyük haritalar lokal haritalara bölünerek hesaplama derecesi lokal haritadaki işaretçi nesne sayısının karesine indirgenmiştir. Yukarıdaki iki teknikte hesaplama derecesini düşürmekle beraber lineerizasyon hatalarını artırmaktadırlar. Bunu engellemek için Williams'ın 2001'de hazırladığı doktora tezinde Sınırlandırılmış Relatif Altharitalama Filtresi (CRSF) [5] metodunu önermiştir. Bu metod da herbir lokal harita belli bir global referans noktasına göre haritaya yerleştirilmeyip yanındaki lokal haritaların pozisyonlarına göre göreceli olarak yerleştirilmektedirler. Bütün bu teknikler bir çok yönden iyi gibi görünseler de kapalı çevrimlerde kararlılıklar ve tutarlılıkları bazen azalmakta ve bozulabilmektedir. 2005 yılında Estrada, Neira ve Tardos'un yaptığı Hiyerarşik SLAM çalışması kapalı çevrim kararlılık ve tutarlığını önemli ölçüde artırmaktadır. 2002 yılında Montemerlo ve Thrun'un ortaya attığı Hızlı SLAM algoritması ise farklı bir yapıya sahiptir. Hızlı SLAM (FastSLAM) tekniği araç yörüngesini kestirmek için parçacık filtre yapısı kullanmaktadır. Haritadaki her bir işaretçi nesne lokasyonunu kestirmek için her bir parçacık filtresi birbirinden bağımsız Genişletilmiş Kalman Filtre grubuyla ilişkilendirilir. Bu yapıdaki SLAM tekniğinde hesaplama derecesi, n haritadaki işaretçi sayısı olmak üzere O(log(n)) ile orantılıdır [2].

SLAM'ın en ciddi problemlerinden biri de veri ilişkilendirme problemidir. Veri ilişkilendirme mobil robotun o anda elde ettiği gözlemlerin haritadaki hangi işaretçi nesnesine ait olduğu yada yapılan gözlemin yeni bir nesneye mi ait olduğu bilgisini veren hipotezler üretir. Veri ilişkilendirmeyi üç gruba ayırabiliriz. Bunlardan ilki Grup Doğrulama tekniğidir. Bu teknik nesneler arasındaki geometik yapıyı baz almaktadır. Bu tekniğin iki formu mevcuttur. Neira ve Tardos'un 2001 yılında ortaya koyduğu Bileşik Uyumluluk Dallanma ve Bağlanma (BUDB) metodu grup doğrulamanın ilk formudur ki ağaç arama metodu olarak bilinir [10]. İkinci form, Tim Bailey'in 2002 yılında hazırladığı doktora tez çalışmasında ortaya koyduğu Birleştirilmiş Sınırlandırılmış Veri İlişkilendirme (CCDA)'dirki grafik arama metodu

olarak bilinir. Grup doğrulama tekniği büyük alanlarda çok iyi sonuçlar vermemektedir. Büyük alanlarda veri ilişkilendirmenin 3. grubunda bulunan Çokluhipotez İzleme (MHT) tekniği kullanılmaktadır [8]. Bu teknik ilk olarak 1994 yılında Cox ve Leonard tarafından kullanılmıştır. Hızlı SLAM metodu da doğal bir MHT içerir. Veri ilişkilendirmenin 2. grubunda yer alan Görünüm İmzaları olarak bilinen ve görüntü tabanlı çalışan bir tekniktir. Bu teknikte sadece nesneler arası geometrik ilişki kullanılmaz. Aynı zamanda nesnelerin görünümleri, renkleri ve yapılarıda veri ilişkilendirme algoritmasıda dahil edilmektedir [8]. Bu konuda bilinen en önemli çalışmalar 1999 ve 2004 yıllarında Konolige ve Gutmann tarafından yapılmıştır.

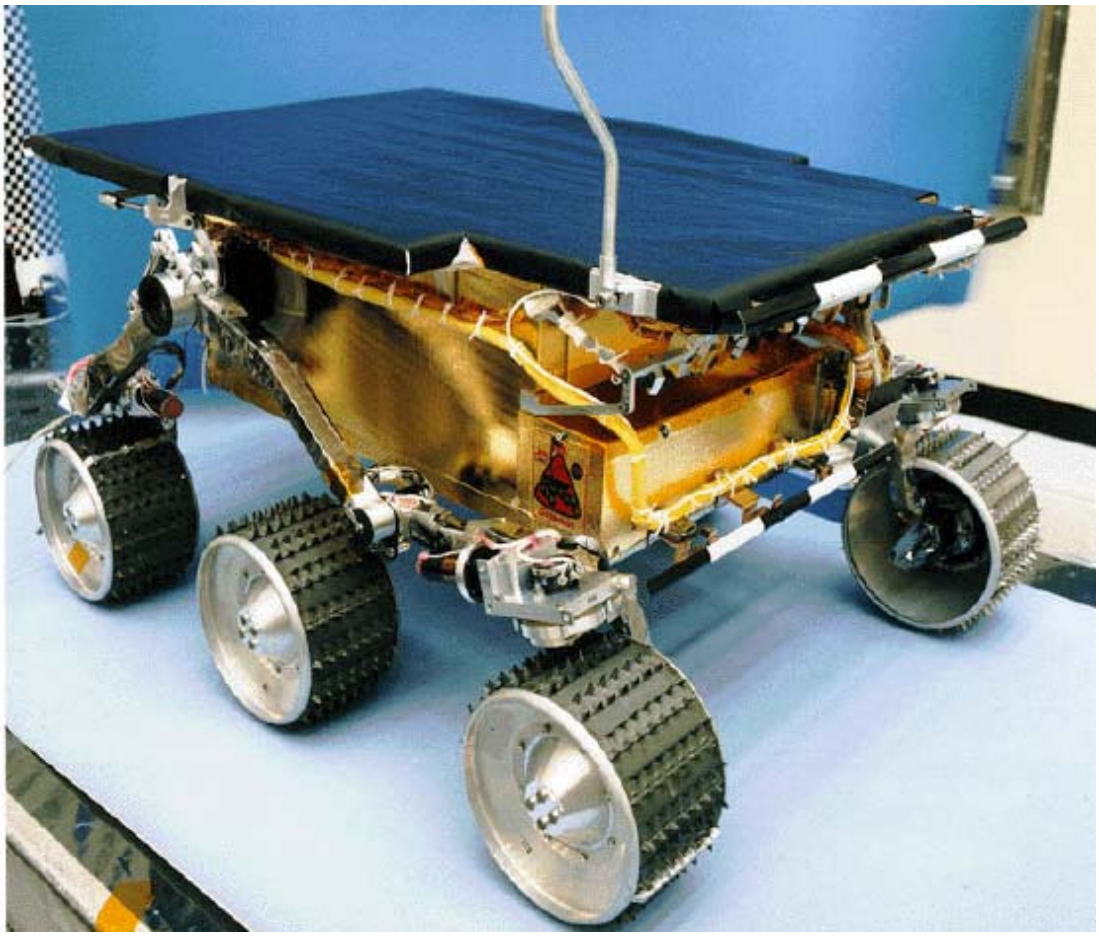
SLAM teknikleri ile ilgili çalışmalar son yıllarda artmış ve gelecekte de artmaya devam edecektir. GKF tabanlı SLAM algoritmaları sıkıntılarıda olsa halen üzerinde çalışmalar yapılan ve geliştirilen ve en iyi bilinen algoritmalardır. Özellikle Hızlı SLAM ve Bilgi Filtresi tabanlı SLAM teknikleri ileri ki yıllarda daha çok çalışmalara kapı açacak nitelikte birer SLAM algoritmalarıdır. Bununlar beraber son yıllarda üzerinde durulan ve literatürde önemli çalışmalar yapılmakta olan çok araçlı SLAM'da gelecekte önemli bir çalışma alanı olarak karşımızda durmaktadır. Çok araçlı SLAM özellikle büyük alanların haritalanmasında kullanılmaktadır. Hem sanayi hem askeri amaçlı olarak önemli ölçüde fayda sağlayacak bir SLAM tekniğidir ki son çalışmalar bu teknikle ilgili çalışmaların ileri ki yıllarda ön sıralarda yer alacağını göstermektedir. Bütün bu SLAM metodların yanında haritanın kararlığı, nesnelerin ve gözlemlerin ilişkilendirmesi, kapalı çevrim tutarlılığı ve kararlığı, hesaplama karmaşıklığı problemleri ile de önemli çalışmalar yapılmaya devam edilmektedir.

Bu tezde bu algoritmalardan en temeli olan GKF tabanlı SLAM ve GKF'nin getirdiği hesaplama problemine yönelik geliştirilen SGKF tabanlı SLAM'yı veri ilişkilendirme algoritmalarıyla beraber karşılaştırdık. İlk uygulamada GKF tabanlı SLAM ile SGKF tabanlı SLAM teknikleri hesaplama dereceleri bakımından karşılaştırılmış ve kovaryans matrisleri arasındaki hata incelenmiştir. İkinci uygulamada 10m'lik karesel bir yörüngede diferansiyel aracın GKF ve SGKF tabanlı SLAM'ın BUYK ve BUDB veri ilişkilendirme algoritmalarıyla kullanımı incelenmiştir. Araç pozisyonundaki hatalar; harita kestirimlerinin gerçek haritadan sapmaları; veri ilişkilendirme ve güncelleme işlemleri için harcanan işlemci zamanları incelenmiştir.

# 2. İNSANSIZ KARA ARAÇLARI

## 2.1 Tanım

Mobil robotlar son yıllarda insan hayatında önemli bir yer edinmeye başladılar. Mobil robotlar kara, hava, denizaltı ve uzay ortamlarında insanların hizmetine sunulmuş ve son yıllarda askeri ve araştırma amaçlı çalışmalarda kullanılmışlardır. Denizaltında kullanılan mobil robotlar denizaltında sismik hareketleri kontrol etmek ve denizaltı yaşamını araştırmak; insansız hava araçları deprem, heyelan gibi karasal hareketleri veya kasırga, tsunami gibi afetleri havadan tespit edip içerisinden bilgi toplamak; insansız kara araçları da askeri amaçlı bomba ve mayın imha etmek, kütüphaneler ve hastanelerde insanlara kılavuzluk yapmak ve bilgi vermek, sanayide ise malzeme taşımak ve üretime katkıda bulunmak gibi çok geniş yelpaze de insanlığın hizmetine sunulmuşlardır. İnsansız kara araçları daha çok askeri amaçlı kullanılmasına rağmen sanayi, kütüphane, hastane gibi mekanlarda insanların hizmetine sunulmuşlardır. İnsansız kara araçlarının deprem, yangın ve kanalizasyon tıkanıklığı gibi durumlarda insanların ulaşamayacağı veya insanlar için tehlike arz eden yerlerde kullanılmasından doğan ihtiyaç giderek artmaktadır. Aşağıdaki resimlerde çeşitli amaçlı için üretilmiş mobil robotlar bulunmaktadır. Şekil 2.1'de Mars'a gönderilen ilk insansız kara aracı Sojourner'in resmini görebiliriz. Şekil 2.2'de bir hava gözlemleri yapan bir insansız hava aracı, şekil 2.3'de ise denizaltında çeşitli araştırmalar için kullanılan insansız denizaltı robotuna ait örnek resimler bulunmaktadır.



**Şekil 2.1:** Mars'a Gönderilen İlk İnsansız Kara Aracı, Sojourner

**Şekil 2.2:** Hava Mobil Robotu

**Şekil 2.3:** Denizaltı Mobil Robotu

### 2.1.1 İKA Tipleri

İnsansız Kara Aracları (İKA) temelde iki tip'dir.

#### 2.1.1.1. Uzaktan Kontrollü İnsansız Kara Aracı

Uzaktan kontrollü insansız kara araçları bir operatör tarafından kontrol edilirler. Çeşitli haberleşme altyapısı ile kontrol edilebilirler. En çok tercih edilen kontrol tipi uzaktan kumanda ile kablosuz kontroldür. Bu tip kontrol günümüz oyuncak arabalarda çok sık kullanılmaktadır. Askeri amaçla kullanılan insansız kara araçları en çok bu tiptedir.

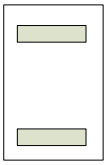
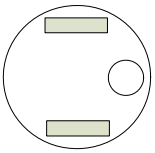
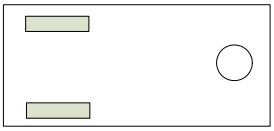
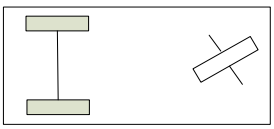
#### 2.1.1.2. Otonom İnsansız Kara Aracı

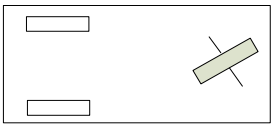
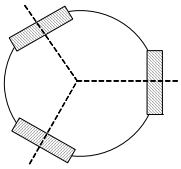
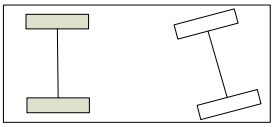
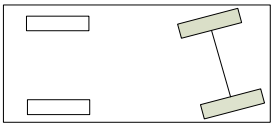
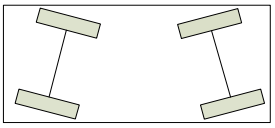
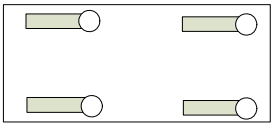
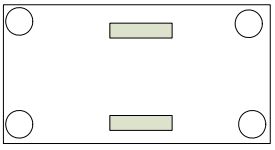
Son yıllarda ortaya çıkan otonom insansız kara araçları, aracın kendi kendisini kontrol ettiği insandan kontrolünden bağımsız çalışan insansız kara araçlarıdır. Bu tip araçlar avantajlı olmakla beraber uygulaması oldukça pahalı ve zordur. Kendisine verilen belirli bir görevi ( yükü bir yerden bir yere taşıma, görme engelli bir hastayı evden işe götürme gibi) yolunu etraftaki engellerden sakınarak belirleyen ve etraftaki yapıları, duvarları, araçları tespit ederek uygulamaya çalışmaktadır. Bu tip araçların kullanılmasındaki en önemli sebep insan kontrolünün olmadığı bölgeler veya zamanlarda robotun verilen işlevi yerine getirebilmesidir. Otonom insansız kara araçlarının konumlarının belirleyebilmesi ve etraftaki nesnelerin yerlerini ve tipleri haritalayabilmesi için çeşitli araştırmalar yapılmaktadır. Bu algoritmalara da genel olarak SLAM algoritmaları denmektedir.

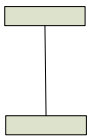
## 2.2 Tahrik Sistemleri

İnsansız kara araçlarının çevrede dolaşabilmesi için çeşitli tahrik mekanizmalarına ihtiyaçları vardır. Bu mekanizmalar tekerleklikli olabileceği gibi, paletli, ayaklı da olabilmektedirler. Bunun için robotun hangi amaçla kullanılacağı tahrik sistemini belirlemek için önemlidir. En çok kullanılan tahrik sistemi tekerlektir. Aracın hareket kabiliyeti açısından tekerlek yapıları çok önemlidir. Tekerlekli insansız kara araçlarında kullanılan standart, nakliye, isveç ve küresel olmak üzere dört temel tekerlek sınıfı mevcuttur. Tablo 2.1'de çeşitli teker konfigürasyon yapıları görülmektedir.

**Tablo 2.1.**Tekerlekli kara araçlarında çeşitli teker konfigürasyonları (Z.Kurt Y.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. | |

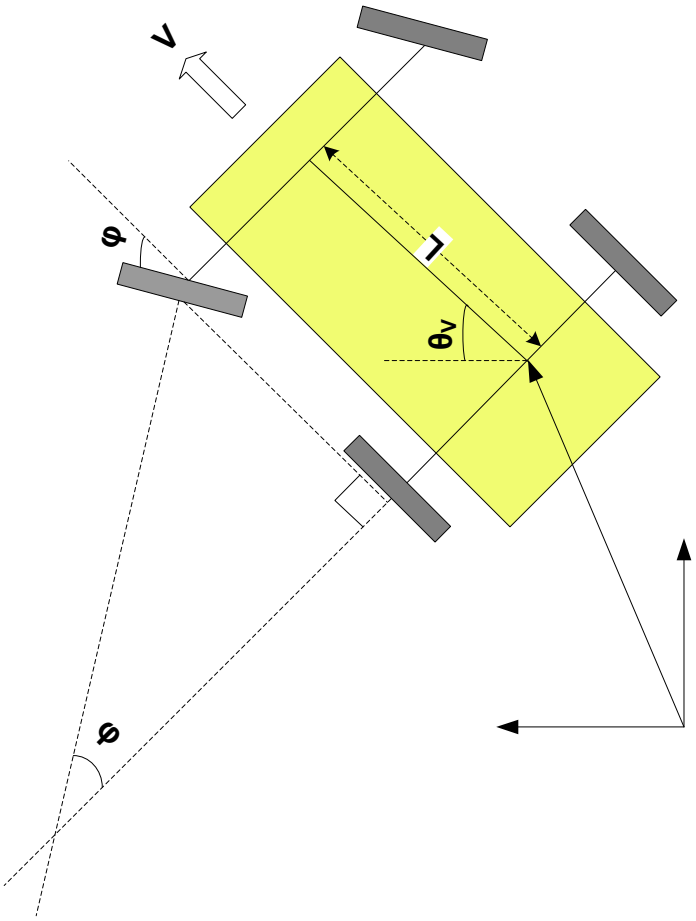
| | | | |
|---|---|---|---|
| |  | Önde yön veren bir teker, arkada iki teker vardır. | Neptune (Carnegie Mellon Üniversitesi) |
| |  | Bir üçgen düzenine oturtulmuş üç adet çok yönlü İsveç tekeri veya küresel teker. | Tribolo EPFL, Palm Pilot Robot Kit (Carneige Mellon Üniversitesi) |
| 4 |  | Önde 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) |
| Yukarıda gösterilen teker tiplerindeki ikonların anlamı | | | |

| | |
|---|---|
|  | 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 |

Robot uygulamalarında yukarıdaki konfigürasyondan bazıları çok kullanılmamaktadır. En çok kullanılan tekerlek yapılarından ilki Ackermann tekerlek yapısı diye bildiğimiz ve 4 tekerlek yapısının ilk sırasında bulunan konfigürasyona benzemektedir. En çok kullanılan ikinci tekerlek yapısı diferansiyel sürücü tipidir ki 3 tekerlekli konfigürasyonun ilk sırasındaki gibi olabileceği gibi 6 tekerlekli terragator yapısındada olabilir. 4 tekerlekli önde ve arkada kastor tekerlekler ve her iki yan da sürükleyici iki teker yapısında da olabilir. Biz uygulamamızda diferansiyel tekerlek yapısındaki konfigurasyonları kullanacağız.

### 2.2.1 Ackermann Tekerlek Modeli

Ackermann tekerlek modeli 1817 yılında Rudolph Ackerman tarafından ortaya çıkarılmış bir modeldir. Her ne kadar modern araçlar sırf Ackermann modelini kullanmasa da temelde bu modeli baz alarak kendi modellerini oluşturmuşlardır. SLAM algoritmaları araştırmalarında bu model oldukça sık kullanılmakta ve makalelerde rastlanmaktadır. Ackermann modeli yukarıdaki konfigürasyon tablosunda 4 tekerlekli yapının ilk konfigürasyonunda gösterilmiştir. Ackermann modelinde tahrik ön taraftaki iki tekerlekten olabileceği gibi arka iki tekerlekten de olabilmektedir.




**Şekil 2.4:** Ackermann Tekerlek Modeli

SLAM uygulamalarında araç modeline ait dinamik denklemler önemlidir. Aracı ötelemek için araca *k* anında uygulanan kontrol işareti *u(k)* olsun. Aracın *k+1* anındaki pozisyonunu bulabilmek için araca ait dinamik modeli bilmemiz gerekiyor. f(.) aracın kinematik modelini belirtmektedir.

V(k): Aracın k anındaki hızı

Φ(k): Aracın tekerleğine k anında uygulanan dönme açısı

θ(k): Aracın xy koordinat düzlemine göre dönme miktarı

L: Aracın tekerlekler arası boyu

(x,y): Aracın düzlemdeki konumu

Aracın kinematik denklemleri aşağıdaki gibi olur.

$$\left.\begin{aligned} \dot{x} &= V.Cos(\theta) \\ \dot{y} &= V.Sin(\theta) \\ \dot{\theta} &= V\frac{Tan(\phi)}{L} \end{aligned}\right\} \qquad \textbf{(2.1)}$$

$$u(k) = \begin{bmatrix} V(k) \\ \phi(k) \end{bmatrix} \tag{2.2}$$

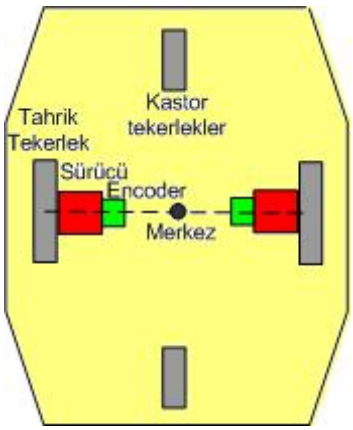
$$x_v(k+1) = f(x_v(k), u(k)) \tag{2.3}$$

Araca yukarıdaki u(k) kontrol işareti uygulandığında aracın k+1 anındaki pozisyonu aşağıdaki denklemle bulunur.

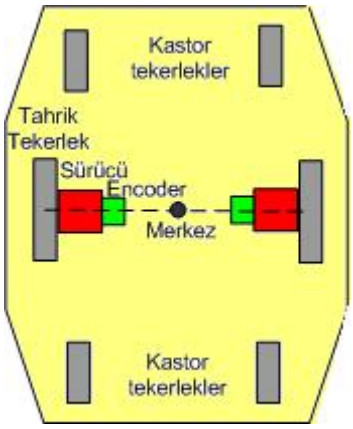
$$\begin{bmatrix} x_v(k+1) \\ y_v(k+1) \\ \theta_v(k+1) \end{bmatrix} = \begin{bmatrix} x_v(k) + \Delta T.V(k).Cos(\theta_v(k)) \\ y_v(k) + \Delta T.V(k).Sin(\theta_v(k)) \\ \theta_v(k) + \dfrac{\Delta T.V(k).Tan(\phi_v(k))}{L} \end{bmatrix} \tag{2.4}$$

### 2.2.2 Diferansiyel Tekerlek Modeli

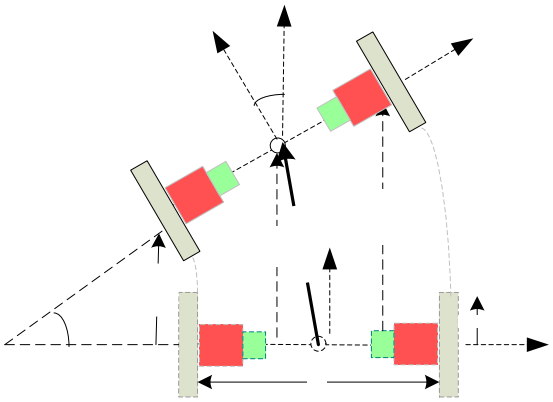
Diferansiyel tekerlek yapısı Ackermann ve bir çok tekerlek yapılarına göre kontrolü karmaşık ve zor olsa da bir çok uygulama da hareket yapısının esnek olması, dönme, geri geri gelme, engellerden hızlı bir şekilde yön değiştirebilme gibi istenilen hareketleri rahatlıkla yapabilme kabiliyetlerinden dolayı tercih edilmektedir. Diferansiyel tekerlek yapısı gerek 3 tekerlekli, gerek 4 tekerlekli gerekse de 6 tekerlekli yapıda olabilmektedir. 3 tekerlekli bir yapıda önde bir kastor arka da ise 2 adet tahrik tekerleği; Şekil 2.5'de görüldüğü gibi 4 tekerlekli yapıda önde ve arkada iki kastor orta da ise 2 adet tahrik tekerleği; Şekil 2.6'da gösterilen 6 tekerlekli yapıda ise önde ve arkada ikişer adet adet toplam 4 kastor ve ortada 2 adet tahrik tekerleği bulunmaktadır.




**Şekil 2.5:**. Önde ve Arkada Birer Kastor Tekerlekli Diferansiyel Araç Tekerlek Yapısı




**Şekil 2.6:**. Önde ve Arkada Çift Kastor Tekerlekli Diferansiyel Araç Tekerlek Yapısı



**Şekil 2.7:**. Diferansiyel Aracı Bir Noktadan Bir Noktaya Öteleme İşlemi

Diferansiyel aracı bir yerden bir yere ötelediğimizde aracın pozisyonunun bulunması için gerekli olan kinematik denklemler.

$D_n$: nominal tekerlek çapı

$C_e$: encoder çözünürlüğü

n: dişli oranı

$\Delta U_L$: sol tekerlek hız değişimi

$\Delta U_R$: sağ tekerlek hız değişimi

$\Delta\theta$: aracın yönündeki açı değişimi

$N_L$: sol tekerlek encoder değeri farkı

$N_R$: sağ tekerlek encoder değeri farkı

$$c_m = \frac{\pi . D_n}{n . C_e} \tag{2.5}$$

$$\left.\begin{aligned} \Delta V_L &= c_m . N_L \\ \Delta V_R &= c_m . N_R \end{aligned}\right\} \Rightarrow \Delta V = \frac{\Delta V_R + \Delta V_L}{2} ; \Delta\theta = \frac{\Delta V_R - \Delta V_L}{L} \tag{2.6}$$

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

$$\left.\begin{aligned} x_v(k+1) &= x_v(k) + \Delta V.Cos(\theta(k) + \Delta\theta) \\ y_v(k+1) &= y_v(k) + \Delta V.Sin(\theta(k) + \Delta\theta) \\ \theta(k+1) &= \theta(k) + \Delta\theta \end{aligned}\right\} \quad \textbf{(2.8)}$$

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

## 2.3 Sensör Sistemleri

İnsansız kara araçlarının konumlarını belirleme ve yön tespitlerinin yapılabilmesi ve etraftaki engellerden sakınıp onları algılayabilmesi için algılayıcılarda dediğimiz çeşitli sensörlere ihtiyaç duymaktadırlar. Sensörleri dokunmatik, iç, dış, görüntü ve uydu tabalı sensörler olarak beş gruba ayıracağız.

### 2.3.1 Aktif Sensörler

Aktif sensörler etraftaki nesnelere cisimlerden yansıyan ışık, ses gibi dalgalar göndererip geri dönen dalgaların genlik ve frekanslarından faydalanarak mesafe, açı gibi ölçümleri yapan sensörlerdir. En sık karşılaştığımız tipleri lazer, radar, infrared ve ultrasonic tip olanlarıdır.

#### 2.3.1.1. Lazer

Nesneye lazer dalga boylu ışık göndererek ölçüm yapmaktadır. Çeşitli konfigürasyon yapılarında olmakla beraber en çok kullanılanı ışını yayan ve alan elemanların tek bir cihaz içinde olanıdır. İnsansız kara araçlarında en çok kullanılan lazer tipi 2D laser range finder olarak bilinen hem nesnenin lazere olan mesafesini hem de nesnenin lazere göre yönünü açı cinsinden ölçen tip lazerdir. Aşağıdaki şekil ve tabloda SICK firmasının üretmiş olduğu LMS 200 lazer range findera ait bilgiler görülmektedir.[14]


**Şekil 2.8:** SICK LMS200 Lazer Range Finder

SICK LMS200 özellikleri

180° kapsama açısı

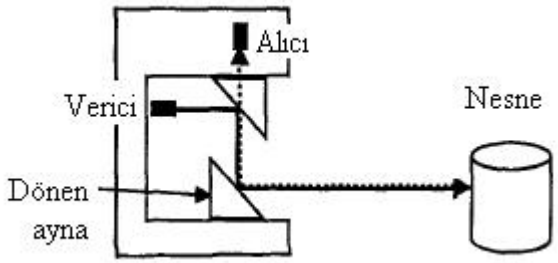
0.5° açı hassasiyeti

10mm uzaklık ölçü hassasiyeti

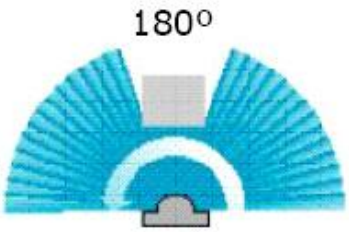
80m mesafeye kadar ölçüm

75 Hz'e kadar tarama hızı

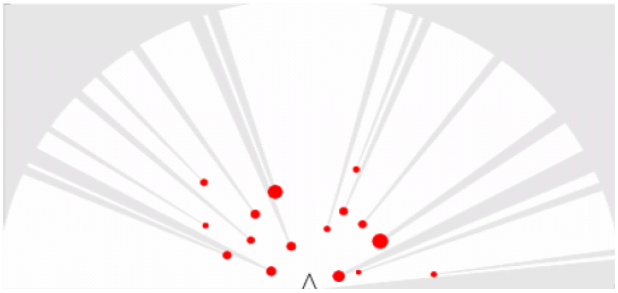
Kapalı alan (Indoor) uygulamalar için




(a)




(b)



$$\text{Ham Veri}: \left\{ (r_1, \phi_1)^T \ldots\ldots (r_m, \phi_m)^T \right\}$$

(c)

**Şekil 2.9:** a) LMS200'in çalışma prensibi b) Lazerin etrafı 180 derece taraması c) Lazerin elde ettiği m adet nesneye ait ölçümler (Tardos)

### 2.3.1.2. Ultrasonik Sensörler

Ultrasonik sensörler ses dalgasının bir nesneye gönderilip geri gelme süresine bağlı olarak mesafeyi algılayan sensörlerdir. Piezoelektrik transducer'ler elektrik enerjisini ses dalgasına dönüştürerek nesneye gönderilmesini sağlarlar. Piezoelektrik transducer'lara AC elektrik verildiğinde oldukça yüksek frekansta ses dalgaları üretmektedirler. Algılama kısmında ise gene piezoelektrik kristaller kullanılmaktadırlar. Piezoelektrik kristaller üzerine bir yük veya ses şiddeti uygulandığında voltaj üreten ekipmanlardır. Dolayısıyla nesneden yansıyan ses dalgaları piezoelektrik kristallerin voltaj üretmesini sağlamaktadırlar.

TOF: Ses sinyalinin gidiş geliş süresi

T: Ortam sıcaklığı

d: Sensörün nesneye olan uzaklığı (m)

c: havadaki ses dalgasının hızı

$$\left.\begin{aligned} d &= c \times TOF/2 \\ c &= 331.4\sqrt{\frac{T+273}{273}}\, m/s \end{aligned}\right\} \qquad \textbf{(2.10)}$$

### 2.3.2 Pasif Sensörler

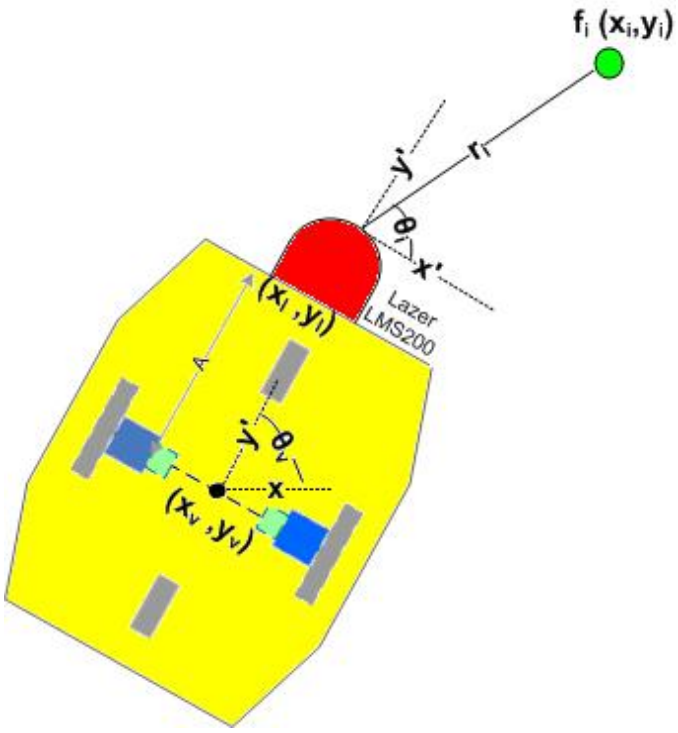
Pasif sensörler nesnelerin yaydığı termal enerjiyi ve elektromanyetik dalgaboyunu algılayan sensörlerdir. Kameralar ve pyroelektik sensörleri birer pasif sensörlerdir. İnsansız kara araçlarında kameralar çok sık kullanılmaktadırlar.

### 2.3.3 Proprioseptif Sensörler

Belirli bir referans eksenine göre aracın konumunu ölçen sensörlere proprioseptif sensörler denir. GPS (Global Positioning System), Şaft enkoderleri, eğim ölçer, pusula, INS (Inertial Navigation System) gibi sensörler bu tip sensörlerdir.

Sensör seçimi yapılacak uygulamaya ve uygulamanın gerçekleşeceği çevreye göre belirlenir. Her sensörün birbirine göre avantaj ve dezavantajları bulunmaktadır. Sonar sensörleri ucuz olmakla beraber hassas alanlar için çok uygun değildirler. Hassas ve kapalı alanlarda lazer en uygun çözümdür. Pusula gibi sensörler manyetik etkileşimlerden dolayı çevrede çok fazla manyetik alan yayan cihazların bulunduğu bölgelerde iyi sonuçlar vermezler.

## 2.4 Gözlem Modeli




**Şekil 2.10:** LMS200 Lazer sensörlü diferansiyel araç için gözlem modeli

$$(x_l, y_l) = (x_v, y_v+A); \quad \textbf{(2.11)}$$

$$\bar{z}(k) \cong \begin{bmatrix} r_i \\ \theta_i \end{bmatrix} \quad \textbf{(2.12)}$$

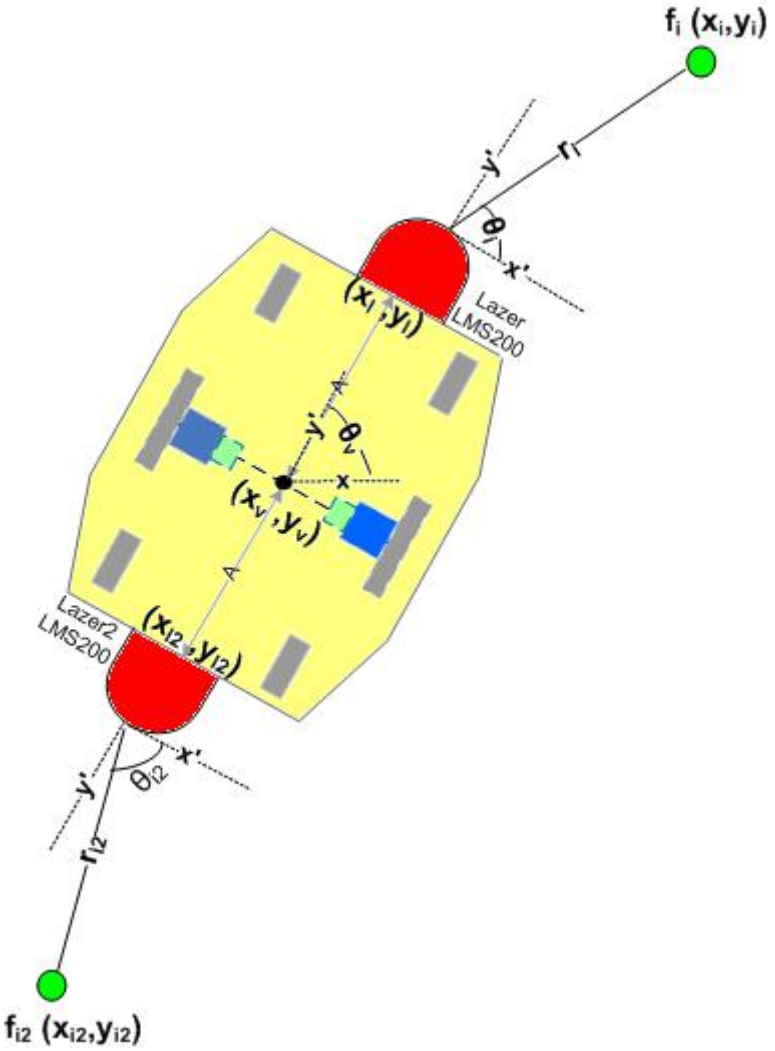
$$\bar{z}(k) = h(\bar{x}(k), \bar{w}(k)) \quad \textbf{(2.13)}$$

Gözlem modeli aşağıdaki gibi çıkarılır.

$$\bar{z}(k) \cong \begin{bmatrix} \sqrt{(x_i - x_l(k))^2 + (y_i - y_l(k))^2} \\ a\tan^2(\frac{y_i - y_l(k)}{x_i - x_l(k)}) - \theta_v \end{bmatrix} \quad \textbf{(2.14)}$$

Burada elde edeceğimiz değişkenler $(x_i,y_i)$'dir. Bunun için yukarıdaki denklemi çözmeye gerek yoktur. Matlab ortamındaki pol2cart($\theta_i$ , $r_i$) komutu bize ($x_i$ ,$y_i$) koordinatlarını verecektir.

Çift taraflı lazer sensör olan aracın gözlem modeli nasıl çıkarılır. Tek taraflıya benzer bir şekilde gözlem modelini oluşturuyoruz. Fakat bu iki gözlem modelinden gelen verilerin veri ilişkilendirme algoritmaları veya öngörü ve güncelleme adımları farklı olabilir.




**Şekil 2.11:** Çift Taraflı LMS200 Lazer sensörlü diferansiyel araç için gözlem modeli

$(x_{l2}, y_{l2}) = (x_v, y_v+A);$ **(2.15)**

$$\bar{z}_2(k) \cong \begin{bmatrix} r_{i2} \\ \theta_{i2} \end{bmatrix} \quad \textbf{(2.16)}$$

$$\bar{z}_2(k) = h(\bar{x}(k), \bar{w}(k)) \quad \textbf{(2.17)}$$

Gözlem modeli aşağıdaki gibi çıkarılır.

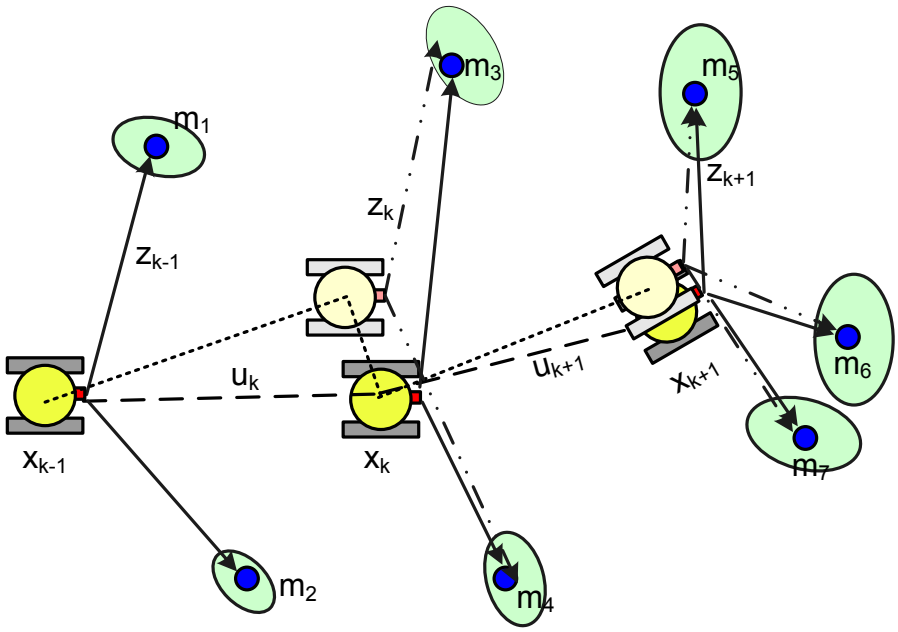
$$\bar{z}_2(k) \cong \begin{bmatrix} \sqrt{(x_{i2} - x_{l2}(k))^2 + (y_{i2} - y_{l2}(k))^2} \\ a\tan^2(\frac{y_{i2} - y_{l2}(k)}{x_{i2} - x_{l2}(k)}) - \theta_v \end{bmatrix} \quad \textbf{(2.18)}$$

Burada elde edeceğimiz değişkenler ($x_{i2}$, $y_{i2}$)'dir. Bunun için yukarıdaki denklemi çözmeye gerek yoktur. Matlab ortamındaki pol2cart($\theta_{i2}$ , $r_{i2}$) komutu bize ($x_{i2}$ , $y_{i2}$) koordinatlarını verecektir. Bu koordinatları elde ettikten sonra önemli olan bu koordinatların lazer 1'in daha önce gördüğü bir işaretçi nesneye mi ait olduğu yoksa yeni bir nesneye mi ait olduğu bilgisidir. Bu bilgi ancak iki lazeride kapsayan özel bir veri ilişkilendirme algoritması ile elde edilebilir. Eğer lazer1 ve lazer2 iki ayrı harita oluşturup sonra bu haritalar birleştirecekse özel bir veri ilişkilendirme algoritmasına gerek kalmaz. Fakat böyle bir durumda da haritaların birleştirilmesi için özel hesaplamalar ve baz alınması gereken ortak noktaların bulunması gerekecek. Bu ortak nokta araca GPS konularak elde edilebilir. Her iki teknikte denenerek doğru haritaya ne kadar yaklaşıldığı incelenebilir.

# 3. NAVİGASYON VE SLAM

## 3.1 NAVİGASYON

Navigasyon, aracın belirlenen varış noktasına en kısa, en optimum yol güzergahını tespit edip planlayarak, hedefe doğru kontrollü bir şekilde ulaşmasını sağlayan sistemdir. Bu işlem GPS gibi uydu tabanlı sensörlerle olabileceği gibi, pusula gibi cihazlarla da yapılabilmektedir. Navigasyon çok genel bir tabirdir. Otomobillerin gibi kara araçlarının otoyollarda gideceği güzargahının belirlenmesi, gemilerin denizde ve uçakların havada veya uzay araçlarının uzayda rotalarının belirlenmesi de birer navigasyondur. Hatta daha ilkel tabiriyle dağdaki çobanın yıldızlara ve aya bakarak yönünü bulması da bir navigasyondur. İnsansız kara araçları da benzer şekilde bu teknikleri kullanarak navigasyon yapabilmektedirler. Otonom insansız kara araçlarının navigasyonunda en çok kullanılan teknik mesafe sayacından lokalizasyon hesabı dediğimiz tekniktir. Mesafe sayacından lokalizasyon hesabı tek başına bir anlam ifade etmez bunun için aracın bir haritaya sahip olması gerekiyor. Mesafe sayacından lokalizasyon hesabı aracın tekerleklerinden alınan encoder bilgilerinden aracın bulunduğu pozisyonunun (x ve y eksenindeki değerleri) ve referans alınan düzleme göre aracın yönünü ve açısının hesaplanması işlemidir. Aracın yönü gyroscope dediğimiz sensörlerle bulunabileceği gibi diferansiyel araçlarda olduğu gibi tekerleklerin farklı dönme hızlarından da hesaplanabilmektedir. Biz bu işleme lokalizasyon yani konum belirleme diyeceğiz. Aracın kendi pozisyonunu hesaplaması da tek başına navigasyon açısından bir anlam ifade etmez. Bunun için aracın nereden başladığını, nerede olduğunu ve nereye gittiğini de bilmesi gerekiyor. Bunun için araca bir harita verilmesi ve başladığı noktanında bu haritada gösterilmesi gerekiyor.



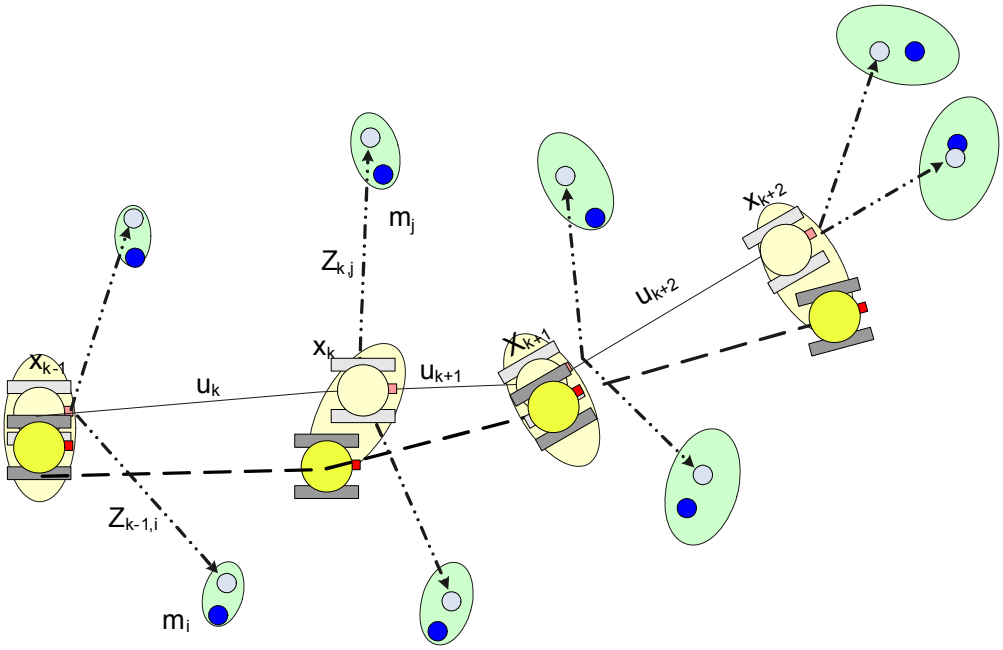

**Şekil 3.1**: Robotun Navigasyonu

Şekil 3.1'de otonom insansız kara aracının navigasyonu gösterilmiştir. $x_{k-1}$ aracın ilk pozisyonu, $z_{k-1}$ aracın (k-1) anında lazer ile algıladığı nesnelere ait ölçümler, $u_k$ ise aracı k anındaki pozisyonuna götürmek için uygulanan kontrol işareti, içi dolu küçük daireler işaretçi nesneler, elipsler ise ölçüm belirsizliğini ifade etmektedir. Araç k-1 anından k anına daha sonra da k+1 anına öteleniyor. Aracın izlenmesi istenilen güzergah içi dolu araç sembolü ile ifade edilmiştir. Araca verilen haritada m ile ifade edilen işaretçi nesnelerin pozisyonları bilinmektedir. Araç $u_k$ kontrol sinyali ile ötelendikten sonra olması gereken pozisyon (içi dolu araç sembolü)'dan saptığı ve içi boş araç sembolünün olduğu pozisyona gittiği görülmektedir. Araç o anda yaptığı $z_k$ ölçüm sonuçlarına göre kendi pozisyonunu düzeltmekte yani güncellemektedir. Araçtaki bu sapma mesafe sayıcı ve araçtaki diğer ölçüm belirsizliklerinden (kayma vb.) kaynaklanmakta ve araç içi boş sembolle belirtildiği yerde olmasına rağmen içi dolu araç sembolüyle belirtilen yerde olduğunu sanmaktadır. İşte lazerden alınan ölçümler aracın kendi pozisyonunu düzeltmesine ve güncellemesine yardımcı olmaktadır. Tabi bu güncelleme mükemmel olamaz. Çünkü lazer ölçümünün de bir gürültüsü ve belirsizliği bulunmaktadır. Bu belirsizlik elips ile ifade edilmiştir. Yapılan ölçüme göre nesne bu elipsin içindeki herhangi bir yere düşmektedir. Araç elipsin merkezine göre kendi konumundaki sapmayı buluyor ve ona göre haritada ki konumunu (**x**) güncelliyor.

## 3.2 SLAM

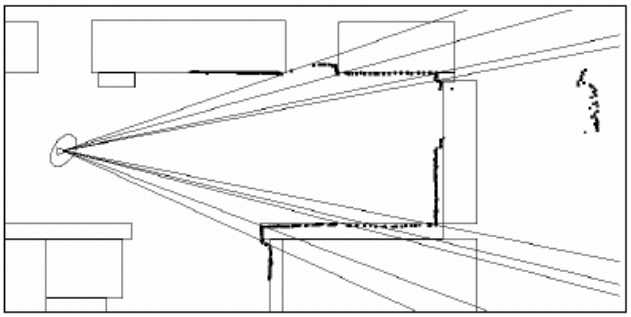
Eş zamanlı konum belirleme ve haritalama (SLAM)'nın konumlama ile ilgili bir örneğini Şekil 3.1'de verilen grafikte görebiliriz. Şekilde araç verilen haritada konumunu eş zamanlı belirlemekte ve güncellemektedir. Bu pratikte mümkün olmakla beraber haritanın daha önce çıkarılıp araca yüklenmesi pratik olmamakla beraber zaman ve nakit israfı getirmektedir. Haritadaki nesnelerin yerleri değiştiğinde ise harita güncelliğini yitirmekte ve araç pozisyonunu yanlış yerdeki nesnelere göre güncellemeye çalışmaktadır. Haritaların daha önce çıkarılmaları bazı durumlarda mümkün de olmamaktadır. Böyle bir durumda aracın hem kendi konumunu belirlemesi hemde ortamın haritasının çıkarılması istenmektedir. Bu işleme eş zamanlı konum belirleme ve harita oluşturma denilmektedir. Araca önceden bir harita verilmez, boş bir haritaya sahiptir. Araç aldığı encoder değerlerinden kendi pozisyonunu kestirmekte, lazer ile yaptığı ölçümlerden etraftan algıladığı nesneleri de haritanın içine koymaktadır. Araç otonom bir şekilde çevrede dolaşırken bir nesneye ait birçok ölçüm yapmakta ve bu ölçümlerin sayısı ve lazer sensörünün hassasiyeti ile orantılı olarak belirsizlikler azalmakta ve elipslerin boyu küçülmektedir. Elips'lerin boyu ne kadar küçük olursa buna bağlı olarak nesnelerin haritadaki yerleri daha kesin bir şekilde belirlenmekte ve araçta kendi konumunu daha doğru bir şekilde güncelleyebilmektedir.

Şekil 3.2 'de otonom bir kara aracının eş zamanlı konum belirleme ve haritalama yapması sembolik olarak gösterilmiştir. Şekildeki içi boş daireler aracın lazer ölçümlerine dayanarak haritaya koyduğu nesneleri, içi dolu daireler nesnelerin gerçek pozisyonlarını, içi boş araç sembolü aracın mesafe sayıcı ve lazer ölçümlerinden belirleyip güncellediği konumu, içi dolu araç sembolü ise araçtan istenen güzergahı simgelemektedir. $x_k$ aracın k anında öngörülen ve güncellenen yani kestirilen pozisyonunu ifade etmektedir. $m_i$ve $m_j$ nesnelere ait pozisyon ve id bilgilerini içermektedir. $Z_{k,j}$ j.nesneye ait k anında alınan ölçümü ifade etmektedir. Şekil 3.2'den de görüldüğü gibi araç hem konumunu belirlemekte ve verilen güzergahı takip etmekte hem de nesnelerin mesafe ve konumlarını ölçüp nesneleri haritaya eklemektedir. Bu işleme SLAM yani eş zamanlı konum belirleme veya lokalizasyon ve haritalama diyoruz.



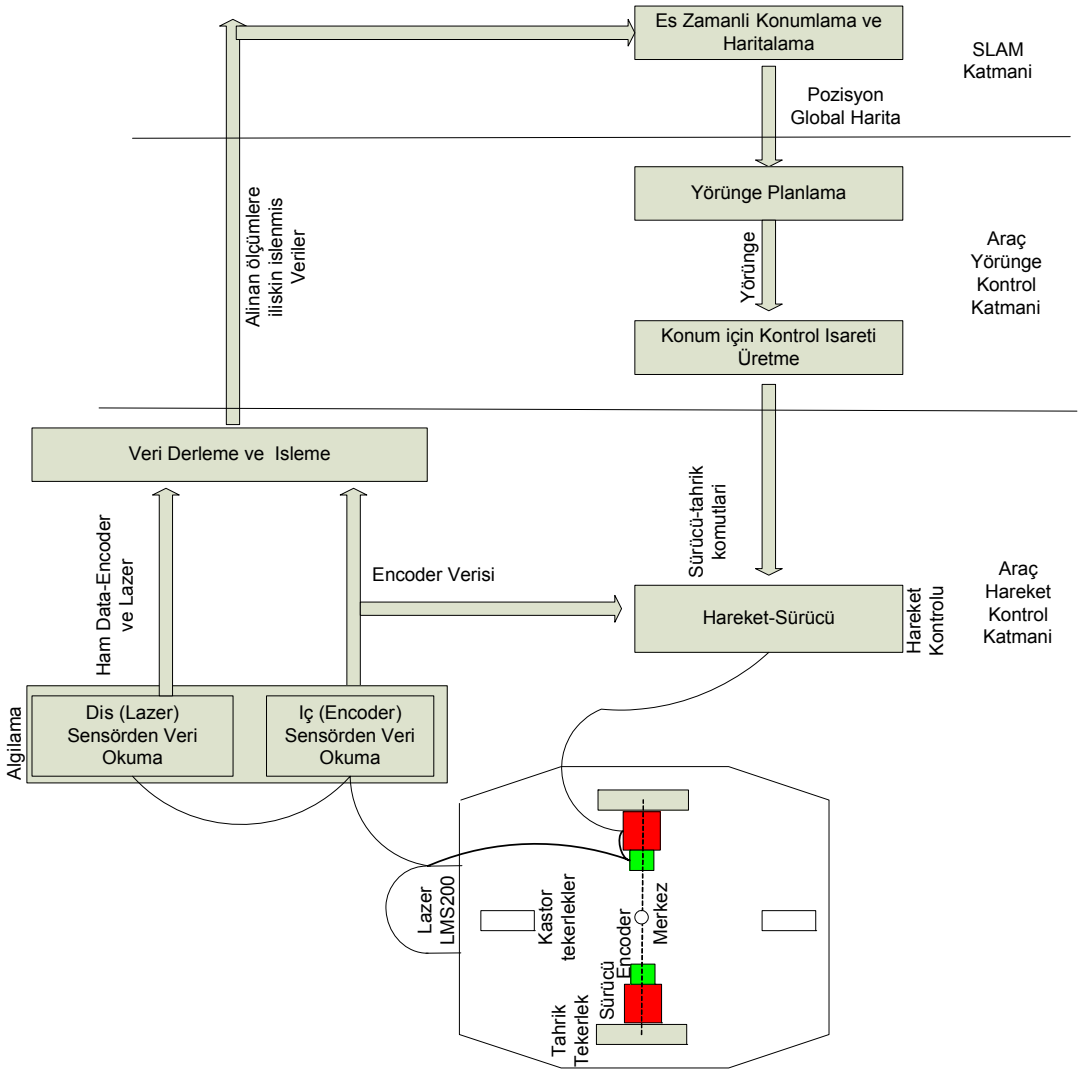

**Şekil 3.2:** Otonom Diferansiyel Aracın SLAM Yapması

SLAM için çok çeşitli algoritmalar ve kestirim filtreleri geliştirilmektedir. Genişletilmiş Kalman Filtresi (GKF), Sıkıştırılmış GKF, Rao-Blackwellized parçacık filtresi, enformasyon filtreleri bu kestirim filtrelerinden başlıcalarıdır. Bu konuda üniversite ve çeşitli araştırma gruplarında SLAM algoritmalarının geliştirilmesi ile ilgili çeşitli araştırmalar yapılmaktadır. Biz GKF ve SGKF tabanlı SLAM ile ilgili yaptığımız bir çalışmayı göstereceğiz. Bundan önce navigasyon ve SLAM ile ilgili ne gibi belirsizlikler ve problemler var onlara değinelim.



**Şekil 3.3:** SLAM ile Oluşturulan Bir Odaya Ait Harita (Tardos)

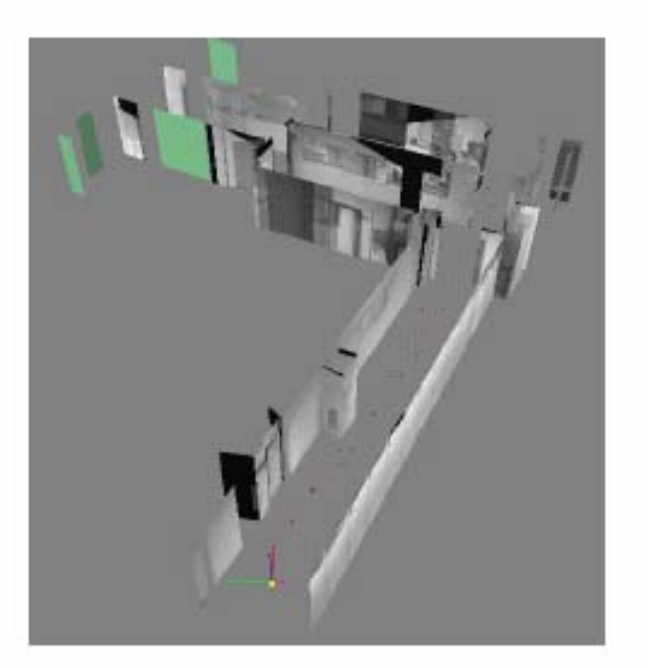
Şekil 3.3'de SLAM ile elde edilmiş odaya ait bir harita görülmektedir. Nokta ile gösterilenler yapılan ölçümler neticesinde oluşturulan haritayı göstermektedir.



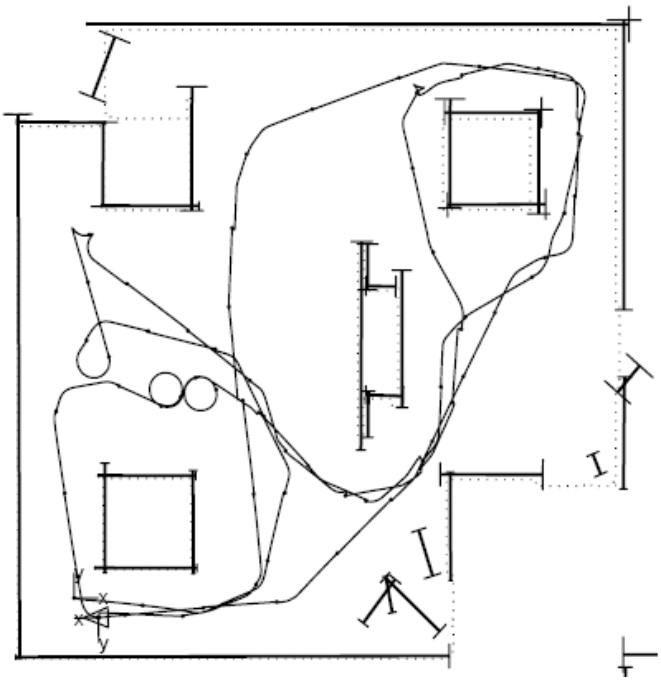

**Şekil 3.4:** İnsansız Bir Araca Ait Algılama Karar Verme ve Hareket Döngüsü

Şekil 3.4'de insansız bir kara aracına ait algılama, karar verme ve hareket döngüsü görülmektedir. Araç dış sensör lazer ve iç sensör mesafe sayıcıdan elde ettiği ham verileri derleyerek SLAM katmanına göndermektedir. Bu katmanda aracın konumu ve lazerden gelen verilerle çevredeki işaretçi nesnelerin konumları kestirilerek çevrenin haritası ve aracın bu haritadaki konumu belirlenir. Aracın haritadaki konum bilgisi yörünge kontrol katmanına gönderilerek aracın belirlenen hedefe ulaşması için gerekli güzergah ve bu güzergaha göre kontrol işareti oluşturulur. Araç hareket kontrol katmanı kontrol işaret sinyalini alarak sürücü yada tahrik sistemine uygular. Mesafe sayıcıdan alınan bilgi ile de güzergahta oluşacak anlık sapmalar hesaplanarak düzeltilir. Araç bu şekilde hem belirlenen hedefe kendi kendine karar vererek gitmekte hemde ortamda bulunan işaretçi nesnelerden faydalanarak ortamın haritasını çıkarmaktadır.

Şekil 3.5'te kameradan alınan verilerle oluşturulan harita görülmektedir. Şekil 3.6'da ise lazer sensörü ile oluşturulan harita görülmektedir. Haritalar arasındaki görünüm farklılıkları haritalamalarındaki temel prensibi ortaya koymaktadır. Kamera ile haritalama da görüntü teknikleri kullanılmaktadır. Lazer ile haritalamada ise stokastik teknikleri kullanılmaktadır.



**Şekil 3.5:** Kamera'dan Elde Edilen Verilerle Oluşturulan Harita (Tardos)



**Şekil 3.6:** Lazerle Elde Edilen Bilgilerle Oluşturulan Bir Harita (Tardos)

### 3.3 SLAM Problemleri

İnsansız kara araçları eş zamanlı konum belirleme ve haritalamayı yaparken bir takım problemlerle karşılaşırlar. Haritaların doğru çıkarılması ve robotun kendi lokasyonunu doğru kestirebilmesi bu problemlerin aşılmasına bağlıdır. En sık karşılaşılan problemlerden ilki kestirim algoritmalarındaki matematiksel işlemlerin, sensörlerden algılanıp haritanın içerisine eklenen çevredeki işaretçi nesnelerin sayısından kaynaklanan haritanın sürekli büyümesinin getirdiği hesaplama problemidir. Eş zamanlı lokalizasyon ve haritalama gerçek zamanlı çalışan bir işlem olduğu için hesaplamalarda harcanacak CPU zamanı çok önemlidir. En önemli ikinci problem ise veri ilişkilendirme problemidir. Araç gezinirken aynı nesneye ait birden fazla ölçüm yapmaktadır. Bu ölçümlerin daha önce gördüğü bir nesneye mi ait olduğu yoksa yeni gördüğü bir nesneye mi ait olduğu çok önemlidir. Eğer bu karar verme mekanizması yanlış çalışacak olursa harita yanlış çıkabiliceği gibi harita da gereğinden daha fazla büyümüş olacak ve problem aynı zamanda hesaplama problemine de dönüşmüş olacaktır. İç ve dış sensör hassasiyetleri ve gürültüleri de üçüncü problem olarak karşımıza çıkmaktadır. SLAM'nin dördüncü problemi çevredeki nesnelerin geometrilerinin doğru algılanma problemidir.

#### 3.3.1 Hesaplama Problemi

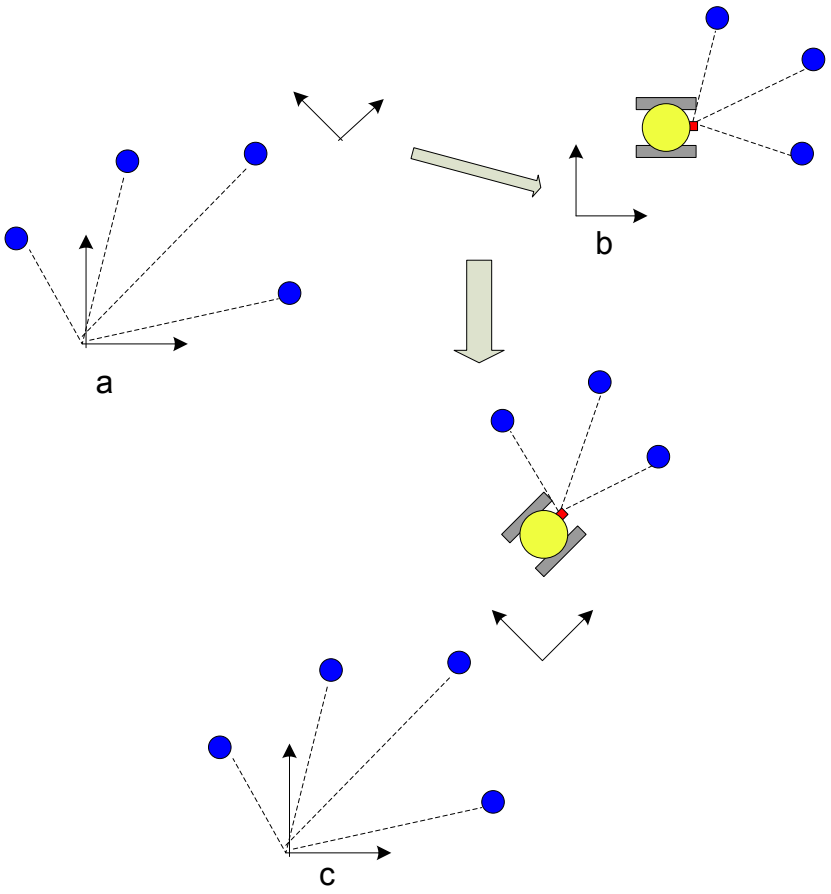
Eş zamanlı lokalizasyon ve haritalamanın en önemli problemlerinden biri kestirim algoritmalarındaki hesaplamaların CPU'daki işlemlerinin çok uzun sürmesidir. SLAM işleminde robot etrafta gördüğü veya algıladığı işaretçi nesneleri haritaya eklemektedir. Haritanın büyümesi algoritma içerisindeki matris ters alma işlemleri gibi matematiksel operasyonların CPU'da çok daha fazla zaman harcanmalarına sebep olmaktadır. CPU bu işlemlerle uğraşırken robotun anlık ihtiyacı olan verileri sağlayamamakta ve haritalarılarında yanlış çıkmalarına sebebiyet vermektedirler. SLAM'ın gerçek zamanlı bir algoritma olduğu yani robot hareket ederken lokalizasyon ve haritalama yaptığı için hesaplamadaki harcanacak zamanın kısaltılması gerekmektedir. Bunun için çok çeşitli algoritma teknikleri sunulmaktadır. Bu tekniklerden bazıları Çözülmüş Stokastik Haritalama (Leonard and Feder 2000), Lokal Haritalama Algoritması (Chong ve Kleeman 1999), Alt-optimal SLAM (Guivant ve Nebot 2001), Dağınık Ağırlık Filtresi (Julier 2001), Ayrıştırılmış Genişletilmiş Bilgi Filtresi (Thrun 2003), Erteleme (Davidson 1998, Knight, Davidson ve Reed 2001), SGKF (Guivant ve Nebot 2001), SLAF (Williams

2001), Harita Ekleme (Tardos 2002). Bu teknikler kullanılırken pozisyon ve kovaryans kestirimlerindeki hataların ne durumda olduklarının da karşılaştırılması gerekmektedir. Bazı teknikler hesaplamadaki harcanan zamanı çok kısaltmalarına rağmen pozisyon ve kovaryans hatalarını da çok artırmaktadırlar. Bunun için en optimum teknik parçalanmış durumların güncellenmesi tekniğidir [8].

#### 3.3.1.1. Parçalanmış Durumların Güncellenmesi

SGKF (Sıkıştırılmış Genişletilmiş Kalman Filtresi) ve SLAF (Sınırlandırılmış Lokal Altharitalama Filtresi).olarak iki temel tipi vardır [8]. SGKF'de durum vektörü ve kovaryans matrisi aktif ve pasif durumlar olmak üzere iki parçaya ayrılıyor. Aktif durumlar o anda lokal olarak görülen işaretçi nesnelere ait pozisyonların ve araç pozisyonunun bilgisini tutmaktadır. Global haritanın geri kalan kısmındaki nesnelere ait pozisyonlar ise pasif durumlar olarak adlandırılmaktadır. Aktif durum vektörü ve kovaryans matrisleri üzerinden GKF kestirim algoritması çalıştırılıyor. Daha sonra geri kalan pasif durum vektörü ve kovaryans matrisleri en son olarak güncelleniyor. Böylece durum vektörü büyüklüğü n adet durum ve matris işlemleri yapılacağı yerde $n_a$ oranında matris işlemleri yapılıyor. Standart GKF algoritmalarında hesaplama kompleksliği N haritadaki işaretçi nesnelere ait x ve y pozisyon sayısı olmak üzere ve M = N+3 olmak üzere $O(M^3)$'dür. Buradaki 3 rakamı durum vektöründeki aracın x,y ve θ konum bilgilerinden gelmektedir. SGKF algoritmalarında ise $N_a$ aktif durum sayısı olmak üzere hesaplama kompleksliği $O(2N_a^2)$'ye kadar azaltılmaktadır.[9]

SLAF'ın ilk avantajı güncellenen işaretçi nesne sayısı lokal altharita koordinat ekseninde tanımlı işaretçi nesnelerle sınırlandırılıyor. Böylece kestirimin güncellemesi esnasındaki hesaplama zamanı toplam harita büyüklüğünden bağımsız hale getiriliyor. Lokal haritalar oluşturulup güncellendikten sonra global haritaya basit vektör ve matris işlemleri ile aktarılıyor. İkinci avantajı, lokal olarak referanslanmış koordinat düzleminde düşük belirsizliklerin olması lineerizasyona bağlı olan yaklaşımlardaki hataları azaltıyor [8]. Şekil 3.7'de SLAF'a ait bir harita görülmektedir.




**Şekil 3.7:** SLAF. SLAM'ın sınırları (b) deki lokal haritaya göre oluşturuluyor ve periyodik olarak (a)'deki global haritaya kayıtlanarak (c)'deki optimal global kestirimi oluşturuyor.

### 3.3.1.2. Seyreltme İşlemi (Sparsification)

Standart GKF-SLAM kestirilen x durum vektörünü ve gerçek x durum vekörüne bağlı Gaussian olasılık yoğunluğunun ilk iki merkez momentiyle tanımlana P kovaryans matrisini üretir. Bu gösterimin alternatifi bilgi vektörü $y_k$ ve üretilen kovaryans matrisinin tersi $Y_k$ informasyon matrisidir.

$$\left.\begin{aligned} Y_k &= P_k^{-1} \\ \hat{y}_k &= Y_k \hat{x}_k \end{aligned}\right\} \qquad \textbf{(3.1)}$$

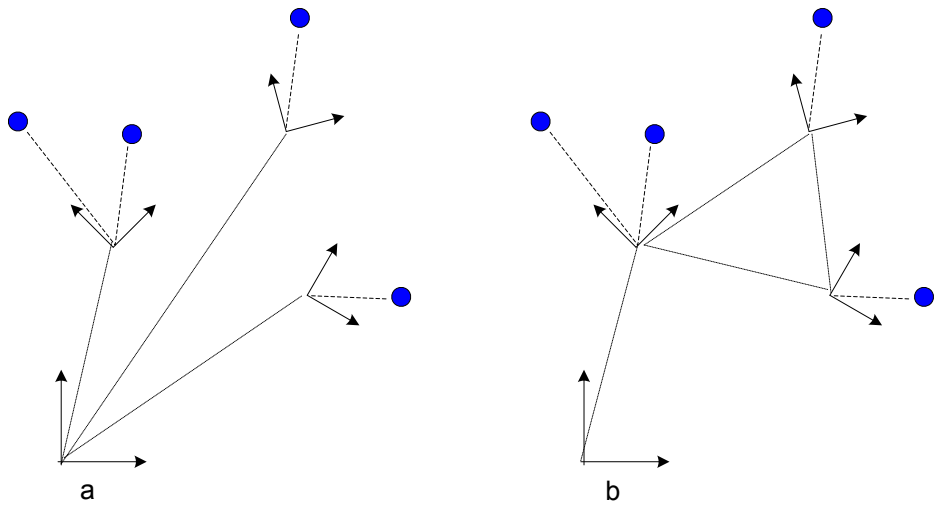
Bu alternatif gösterimin avantajı çok büyük haritalarda normalize edilmiş bilgi matrisinin diagonal elemanlarının sıfıra çok yakın olmasıdır. Thrun bu sıfıra yakın elemanları sıfıra setleyip seyreltme işlemi yapan bir teknik sunmuştur. Bunun dışında çeşitli seyreltme teknikleri mevcuttur. Bu teknikler için Bailey ve Durrant-Whyte'ın ilgili SLAM kaynağından faydalanılabilir [8]. Bu teknikler matristeki sıfırlı elemanları seyrelttikleri için hesaplamada harcanacak zamanı azaltmış oluyorlar.

### 3.3.1.3. Altharitalar

Altharitalama güncellemeler sırasında işaretçi nesne sayısıyla artan hesaplama derecelerini düşüren metodlardandır. Altharitalama Şekil 3.8'de görüldüğü gibi iki teknikle karşımıza çıkmaktadır. Birincisi global altharitalama ikincisi ise relatif altharitalamadır. İki tekniğin ortak noktası altharitalara ilişkin lokal koordinat düzlemleri belirlemeleri ve işaretçi nesne pozisyonlarını bu lokal düzleme göre yerleştirip güncellemeleri ve global haritaya kayıt etmeleridir.

Global altharitalama metodunda kestirimlerin global lokasyondaki yerleri lokal koordinatların ortak bir koordinata göre relatif olarak belirlenmesiyle elde edilmektedir. Bunun için Rölatif Işaretçi Nesne Gösterimi (RIG), Hiyerarşik SLAM ve Sabit Zamanlı SLAM (SZE) gibi teknikler kullanılmaktadır. Global altharitalama hesaplama derecesini iki derece düşürmekle beraber ortak bir koordinat düzlemini referans aldıkları için pozisyon belirsizliklerinden dolayı oluşan lineerizasyon konusuna çözüm getirmemektedir.

Relatif altharitalama da ise global alt haritalamadan farklı olarak ortak bir koordinat düzlemi yerine yanındaki lokal altharitaya ait koordinat düzlemine göre relatif olarak kendi lokal düzlemininin yerini belirlemektedir. Bütün lokal haritalar yanındaki lokal karitaya ait koordinat düzlemine göre relatif olarak kendi düzlemlerini belirleyerek bir ağ oluştururlar ve böylece global harita oluşturulmuş olur. Bu teknikle hem hesaplama derecesi düşürülüyor hem de nonlineerlik problemine çözüm getirilmiş oluyor. Bunun için Sınırlandırılmış Rölatif Altharitalama Filtresi (SRAF)[5], Atlas Taslağı ve Ağ Birleşik Özellik Haritaları (ABÖH) teknikleri kullanılmaktadır. [8]

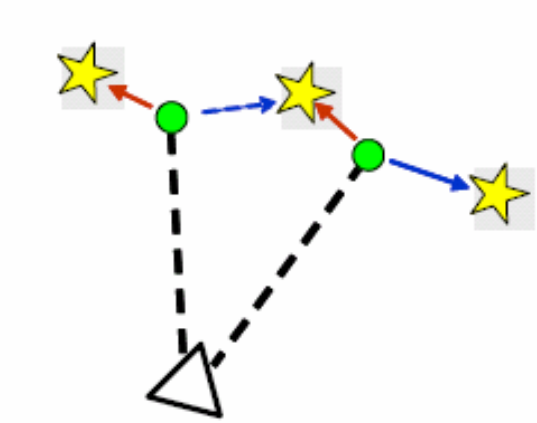



**Şekil 3.8:** Global (a) ve rölatif (b) alt haritalar

### 3.3.2 Veri İlişkilendirme Problemi

Haritalamanın en zor problemi data association adıyla bilinen veri ilişkilendirme problemidir. Veri ilişkilendirme problemi robotun aynı yerden geçtiğini kestirmesi yani aynı nesneleri veya aynı engelleri tanıyabilmesi problemidir. Diğer bir ifade ile robotun, karşına çıkan işaretçi nesnelerin daha önce gözleyip haritaya eklediği işaretçi nesnelere ait birer gözlememi yoksa yeni bir işaretçi nesneye ait gözlememi ait olduğuna karar vermesi veri ilişkilendirme olarak adlandırılır. Eğer robot kapalı bir çevrim şeklinde geziniyorsa başladığı yere tekrar geldiğini anlayabilmesi gerekmektedir. Eğer kapalı çevrim bittiğinde bunu anlayamazsa yörüngesini hatalı oluşturur. Yeni gözlediği verileri haritaya eklemeden önce mevcut haritayla ilişkilendirmesi gerekmektedir. Çünkü haritaya ekledikten sonra bu artık yeni bir engel (işaretçi nesne) olmuştur yani geriye doğru veri ilişkilendirme yapılamaz. Yanlış veri ilişkilendirme harita kestiriminin gerçekten uzaklaşmasına ve bunun neticesinde de lokalizasyon algoritmalarının kötü bir şekilde çakılmalarına sebep olmaktadır. Bu problemi çözmek için çok çeşitli teknikler kullanılmaktadır.

Şekil 3.9'da bir nesneye ait atama problemi görülmektedir. Burada üçgen aracı, yıldız ile gösterilenler işaretçi nesneleri, yeşil ile gösterilenleri ise elde edilen ölçümleri göstermektedir. Mavi oklar yapılan gözlemlerin doğru işaretçi nesnelerine ait eşleşmeleri, kırmızı oklar ise yapılan gözlemlerin yanlış işaretçi nesnelerine ait eşleşmeleri göstermektedir. Robot yaptığı gözlemleri kırmızı oklar ile gösterilen işaretçi nesnelerle eşleştirdiği için yanlış bir veri ilişkilendirme yapmıştır. Veri ilişkilendirme yanlış yapıldığı için ölçülen işaretçi nesne başka bir işaretçi nesne ile ilişkilendirilmiş ve o işaretçi nesneye ait pozisyon ve kovaryans matrisi güncellenmiştir. Eğer bu yanlış eşleşmeler artarsa harita doğru haritadan sapacaktır.



**Şekil 3.9:** Bir Araçta Oluşabilecek Atama Problemi

### 3.3.2.1. Grup Doğrulama

Işaretçi nesnelar arasındaki geometrik ilişki kullanılması ilkesine dayanmaktadır. Gözlenen veriler ve haritaya atılmış işaretçi nesneler arasında bir eşleştirme yapılır. Bu eşleştirmeler arasında bir karşılaştırma yapılarak en uygun olanları seçilir. Testi geçen bu uygun eşleşmelerden faydalanılarak ilgili işaretçi nesne pozisyonlarının güncellenmesi sağlanır. Testi geçemeyenler ise yeni işaretçi nesne olarak atanırlar. Bu tekniğin iki formu bulunmaktadır. İlki Bileşik Uyumlu Dallanma and Bağlanma (BUDB) [10] ikincisi ise Birleşik Sınırlandırılmış Veri İlişkilendirme (BSVİ) [4,6]'dir. İlk teknik ağaç araştırma ikincisi ise grafik araştırma algoritması olarak bilinmektedir. BSVİ aracın pozisyonunu bilmese bile güvenilir veri ilişkilendirmesi yapabilmektedir.

Grup ayırıcı güvenilir bir veri ilişkilendirmesi yapabilmektedir: Eğer ayırıcı yeterli bir şekilde sınırlandırılırsa, ilişkilendirmede ki hatalar önemsiz kalır ve eğer doğru bir işaretçi nesneye fiziksel olarak yakın yanlış bir işaretçi nesne için yanlış ilişkilendirme yapılırsa, tutarsızlık minumum seviyede kalmış olacaktır. Bu daha karmaşık alanlarda tutmayabilir. Onun için daha kompleks [ÇHİ-Çokluhipotez İzleme gibi] veri ilişkilendirme mekanizmaları kullanılmaktadır.[8]

### 3.3.2.2. Görünüm Tanıma

Geometrik kalıplar ve şablonlar güvenli bir ilişkilendirme için tek başına yeterli olmayabilir. Bunun için görüntü ile yapılan algılamalar da şekil, renk ve yapı olarak da iki veri arasındaki ilişkiler gözlenerek desteklenir. SLAM için Görünüm Tanıma kapalı bir çevrimde mümkün olabilecek veri ilişkilendirmelerini tek başına tahmin edebilecek yada klasik ayırıcılara işaretçi nesnelerin farklı özellik ve bilgilerini ekleyerek yardımcı olabilecek kullanışlı bir araçtır.

Daha önceden Görünüm Tanıma ve Görüntü Benzerlik Metodu, görüntü veritabanları için indeks oluşturma ve topolojik haritalar içerisindeki yerleri tanıma amaçlı geliştirilmişlerdir. Çok yakın geçmişte bu teknik üzerinde Newman'ın kapalı çevrimi algılamadaki çalışmaları iki önemli yenilik getirmiştir. Sıralı görüntüler üzerinde benzerlik ölçümü hesaplanır ve özdeğer tekniği ile benzer ortak modlar kaldırılır. Bu yaklaşım sadece ilginç veya ortak olmayan benzerlikleri dikkate alarak hatalı pozitiflerin meydana gelmesini oldukça azaltır.[8]

### 3.3.2.3. Çoklu Hipotez İzleme (ÇHİ) Veri İlişkilendirmesi

Çoklu-hipotez veri ilişkilendirme tekniği karmaşık alanlarda güçlü hedef takibi için temel bir tekniktir. Her bir ilişki hipotezi için ayrı bir yol kestirimi üreterek, gereğinden fazla yolları dallandırarak, veri ilişkilendirmedeki belirsizliği çözümlemeye çalışır. Oluşturulan yolların sayısı sistemin işleyiş hızına bağlı olarak limitlenir ve düşük-ihtimalli yollar hipotez ağacından budanır.

ÇHİ özellikle büyük alanlarda güçlü SLAM uygulamaları için önemlidir. Örneğin, kapalı çevrimlerde, robot yapısal olarak benzer özellikler algıladığı çevrede şüphenilen çevrimler için ayrı hipotezler ve çevrim olmadığı hipotezini üretir. En büyük dezavantajı her bir hipotez için ayrı birer harita kesitirmi yaptığından işlemsel yük getirmesidir. Seyreltme veya altharitalama methodları kullanılarak kolay işlenebilir bir çözüm üretilebilir. Hızlı-SLAM algoritmaları herbir parçacığın kendi haritasını kestirmesinden dolayı birer doğal çoklu-hipotez çözümleridir.[8]

### 3.3.3 Gürültü Problemi

SLAM'daki problemlerden biriside gürültülerdir. Sensör gürültülerinin stokastik olarak yapıları incelenerek bu problem minimize edilebilir. GKF tabanlı SLAM algoritmalarında gürültüler Gauss yapıda olmak zorundadır. Eğer gürültü Gauss yapıda değilse parçacık ve bilgi filtresi gibi değişik kestirim algoritmaları kullanmamız gerekmektedir. Bu yüzden kestirim algoritmasını kullanmadan önce lazer, mesafe sayıcı, giroskop gibi algılayıcılara ait ölçümlerden gelen gürültüler incelenmeli ve ona göre kestirim algoritmaları seçilmelidir. Bu sensörlerden bir çok veri alınıp gerçek değerle stokastik olarak karşılaştırılıp gürültü tipi belirlenmeli, gaussian gürültü ise de kestirimin daha doğru olması için varyans değerleri saptanmalıdır. Böylece kestirim algoritmalarının daha doğru çalışmalarına yardımcı olmuş oluruz. Ölçüm hataları arasındaki korelasyonlarda önemlidir. Ölçümlerdeki hataların biribirine bağımlı olup olmadığıda istatiksel olarak belirlenmelidir. Uygulamalarda bu hata bağımsız olarak düşünülmektedir. Buna bağlı olarak da kovaryans matrisleri oluşturulmaktadır.

### 3.3.4 Çevre Problemi

Çevredeki nesnelerden gözlenen işaretçi nesneler bir duvara, bir masaya, açık bir kapıya veya geometrik olmayan bir şekle ait olabilir. Lazerin algıladığı sadece mesafe ve açı değerleridir. Her algılanan nesneye ait mesafe ve açı bilgileri haritada

bir yer tutar ve bir nokta ile gösterilirler. Bu noktalar birleştirilerek nesneler oluşturulmaya çalışılır. Yanlış noktalar böylece harita dışına atılabilir. Bu işleme birleştirme ve ayrıştırma işlemi diyoruz. X durum vektörü sadece sayılardan oluşmaktadır. Bu sayıların anlamlı hale gelmesi ve harita oluşturabilmesi için bu gösterimlerin düzgün ve doğru yapılması gerekmektedir. Ayrıca çevrenin dinamik olması da ayrı problemi beraberinde getirmektedir. Gerçek dünya statik değildir. Oysa biz uygulamalarımızda hep statik işaretçi nesneları algılamaya çalışıp onların haritalarını oluşturmaya ve güncellemeye çalışmaktayız. Park edilmiş araçlar, masalar, sandalyeler gibi nesneler sürekli yer değiştirirler. Bu tip nesneler algılanıp ayrıştırılarak statik bir işaretçi nesne gibi haritaya eklenmezler. Bunun çözümünü daha üst katmanlarda da yapılabilir. Haritalar bir kez oluşturulduktan sonra güncellenmeye devam edilebilir ve periyodik olarak haritalar arasında görüntü karşılaştırılmaları yapılarak farklılıklar tesbit edilip harita daha yalın hale getirilebilir.

## 4. GKF Tabanlı SLAM

Genişletilmiş Kalman Fitresi (GKF) ile SLAM en çok bilinen ve en iyi geliştirilen metodlardan birisidir. SLAM'ın ilk uygulamaları da bu kestirim metodu ile yapılmıştır. GKF tabanlı SLAM için filtre geliştirmeleri ve problemlerine yönelik tekniklerin bulunması ile ilgili araştırmalar halen devam etmektedir. GKF tabanlı SLAM'a alternatif olarak Parçacık Filtre tabanlı SLAM ve İnformasyon Filtre tabanlı SLAM'lar da ortaya atılmışlardır. 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 değildirler. Bunun için araştırmacılar lineer olmayan modeller için Kalman Filtresinin kullanılabilirliğini araştırmışlar ve lineer olmayan modelleri lineer hale getirerek Kalman Filtresinin lineer olmayan modeller içinde kestirim metodu olarak kullanılabileceğini bulmuşlardır. Kalman filtresi ve Genişletilmiş Kalman Filtresi'nin temeldeki yapısı ve mantığı aynıdır. Kalman filtresi formülleri içerisinde sadece küçük bir değişiklik yapılarak Genişletilmiş Kalman filtresi ortaya atılmıştır. Kalman filtresi ardışıl bayes kestirim metoduna benzer hatta aynı yapıda bir kestirim metodudur. Aşağıdaki şekilde kestirim araçlarının çalışmasına ait bir şekil görülmektedir. [17-19]


**Şekil 4.1:** Kestirim Aracı Yapısı

### 4.1 Ardışıl Bayes Kestirimi

Birçok durumda x'in önceki değerleri hakkında bilgimiz vardır. X p(x) dağılımına sahip bir çokdeğişkenli bir rastgele değişken olsun. Öncül bilgilerden faydalanarak bunun $N(\mu_p, \sigma_p^2)$ değerlerine sahip gaussian bir değişken olduğunu düşünelim. Eğer p(z|x) ile modellenebilen bir özelliğe sahip z değerlerini gözleyebilen bir sensörümüz

varsa, x'e ait son bilginin dağılım fonksiyonu p(x|z)'nin kestirimini Bayes kuralı uygulayarak bulabiliriz. Bu kestirim için sadece gözlenen z değerlerinin değil x'in önceki değerlerininde bilinmesi gerekmektedir.

$$\left.\begin{aligned} p(x \mid z) &= \frac{p(z \mid x).p(x)}{p(z)} \\ p(x \mid z) &= C \times p(z \mid x).p(x) \end{aligned}\right\} \quad \textbf{(4.1)}$$

Maksimum Soncul (MAP) kestirim algoritması $p(z \mid x) \times p(x)$ değerini maksimize eden x değerlerini bulmaya çalışır. Özetleyecek olursak MAP kestirimi verilen z gözlem değerleri, $p(z \mid x)$ olasılık fonksiyonu ve x'in öncül değerlerine ait p(x) dağılımından faydalanarak soncul p(x|z) dağılım fonksiyonunda maksimum x değerlerini bulmaya çalışır. Eğer olasılık fonksiyonu Gaussian ise $p(z \mid x) = \frac{1}{C} e^{-\frac{1}{2}(z-x)^T P^{-1}(z-x)}$ olur ve o dağılımı maksimize eden değer Gaussian standart grafiğinin tepe noktası yani ortalama değeridir.

$$\hat{x}_{map} = \underset{x}{\operatorname{argmax}}\ p(z \mid x).p(x) \quad \textbf{(4.2)}$$

Ardışıl Bayes kestirimi MAP kestiriminden esinlenilerek ortaya atılmıştır. MAP kestiriminde öncül bilgiler ve o anki ölçümler birleştirilerek $\hat{x}_k$ değerleri kestirilmeye çalışılmaktadır.Yeni bir gözlemde ise önceki kestirilen ve artık öncül hale gelen $\hat{x}$ değeri üzerinden yeni $\hat{x}$ değerleri kestirilmeye çalışılacaktır. Bu yaklaşım robot uygulamalarında kullanılmaktadır. Sensör z gözlem değerlerini elde edecektir. Her k. adımda o ana kadar verilen bütün gözlemlerden [$Z^k = (z_1, z_2, ..., z_k)$ $z_k$: o anlık ölçüm değerleri] kestirim yapmaya çalışılacaktır. Şimdi Bayes kuralını tam olarak uygulayalım.

$$p(x, Z^k) = p(x \mid Z^k).p(Z^k) \quad \textbf{(4.3)}$$

diğer bir ifadeyle

$$p(x, Z^k) = p(Z^k \mid x).p(x) \quad \textbf{(4.4)}$$

bu iki ifadeyi eşitleyecek olursak

$$p(x \mid Z^{k}).p(Z^{k}) = p(Z^{k} \mid x).p(x) \quad \textbf{(4.5)}$$

Koşullu gözlemlerin birbirinden bağımsız olduklarını düşünürsek;

$$p(Z^{k} \mid x) = p(Z^{k-1} \mid x).p(z_{k} \mid x) \qquad z_k\text{: k anındaki gözlemler} \quad \textbf{(4.6)}$$

4. 5 ve 4.6 denklemleri birleştirecek olursak

$$p(x \mid Z^{k}).p(Z^{k}) = p(Z^{k-1} \mid x).p(z_{k} \mid x).p(x) \quad \textbf{(4.7)}$$

$p(Z^{k-1} \mid x)$ için Bayes kuralını uygulayalım

$$p(Z^{k-1} \mid x) = \frac{p(x \mid Z^{k-1}).p(Z^{k-1})}{p(x)} \quad \textbf{(4.8)}$$

Denklem 4.7 ve 4.8 birleştirildiğinde,

$$p(x \mid Z^{k}).p(Z^{k}) = p(z_{k} \mid x).\frac{p(x \mid Z^{k-1}).p(Z^{k-1})}{p(x)}.p(x) \quad \textbf{(4.9)}$$

$$= p(z_{k} \mid x).p(x \mid Z^{k-1}).p(Z^{k-1}) \quad \textbf{(4.10)}$$

$$p(x \mid Z^{k}) = \frac{p(z_{k} \mid x).p(x \mid Z^{k-1}).p(Z^{k-1})}{p(Z^{k})} \quad \textbf{(4.11)}$$

$$p(z_{k} \mid Z^{k-1}) = \frac{p(z_{k}, Z^{k-1})}{p(Z^{k-1})} \quad \textbf{(4.12)}$$

$$= \frac{p(Z^{k})}{p(Z^{k-1})} \quad \textbf{(4.13)}$$

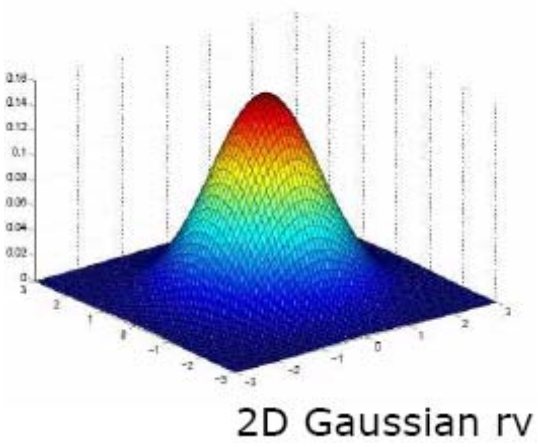
$$\frac{p(Z^{k-1})}{p(Z^{k})} = \frac{1}{p(z_{k} \mid Z^{k-1})} \quad \textbf{(4.14)}$$

sonuç olarak ta istediğimiz sonucu elde ederiz.

$$p(x \mid Z^{k}) = \frac{p(z_{k} \mid x).p(x \mid Z^{k-1})}{p(z_{k} \mid Z^{k-1})} \quad \textbf{(4.15)}$$

Bu sonuç neyi ifade ediyor. Amacımız k adımı dahil bütün adımlardaki gözlemlere bağımlı x'in koşullu dağılımı $p(x|Z^k)$'yi bulmak. $p(x|Z^{k-1})$ (k-1). adımdaki gözlemlerle koşullu en son en iyi kestirilen x'in öncül dağılımını ifade ediyor. Ardışıl Bayes kestirimi, yeni bilgilerin o anki olasılık fonksiyonu ile öncül x'in koşullu dağılımıyla çarpılarak basit bir şekilde eklenebildiği için çok güçlü bir kestirim mekanizması olarak bilinir.

Eğer x çokdeğişkenli rastgele değişkense;




**Şekil 4.2:** İki Boyutlu Gaussian Değişkenin Dağılım Grafiği

$$x \approx N(\hat{x}, P) \tag{4.16}$$

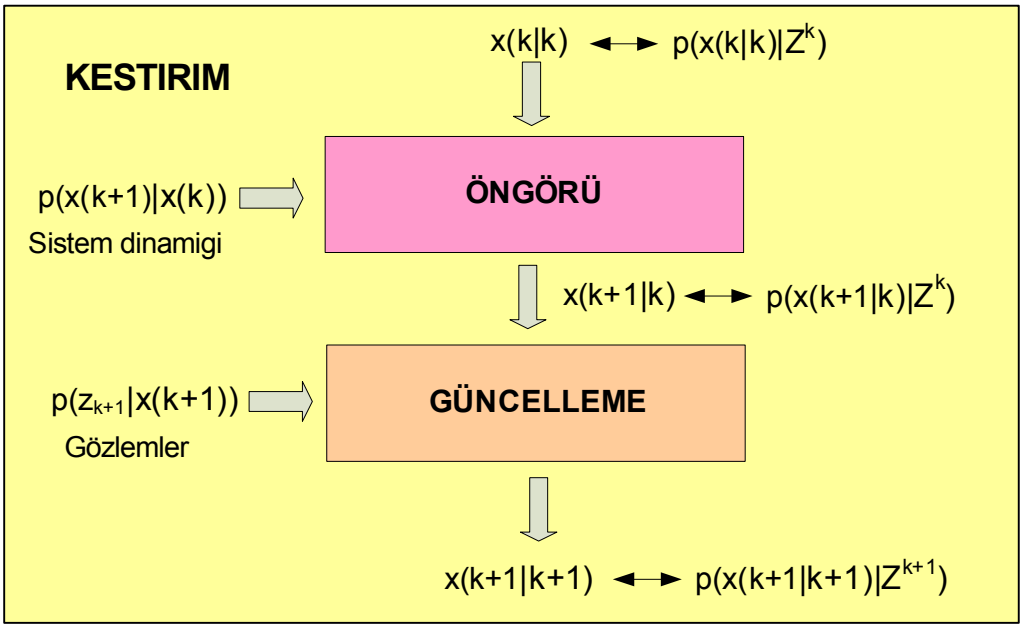
$$\hat{x} = E[x] = \begin{bmatrix} x_1 \\ . \\ . \\ x_n \end{bmatrix} \tag{4.17}$$

Kovaryans matris P

$$P = E[(x-\hat{x})(x-\hat{x})^T] = \begin{bmatrix} P_{11} & P_{12} & . & . & P_{1n} \\ P_{21} & P_{22} & . & . & P_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ P_{n1} & P_{n2} & . & . & P_{nn} \end{bmatrix} \tag{4.18}$$

## 4.2 Genişletilmiş Kalman Filtresi

Kalman Filtre algoritma yapısı aşağıdaki şekilde verilmiştir.




**Şekil 4.3:** Kalman Filtre Yapısı

Bu yapıda sistem (araç) dinamiği ve gözlemler lineer olarak ele alınmıştır. Lineer modellerde yukarıdaki Kalman Filtre kestirimi uygulanabilir. Fakat lineer olmayan modeller -ki gerçek hayatta da sistemler ve gözlemler lineer değillerdir- Kalman Filtre denklemlerine direkt olarak uygulanamazlar. Bunun için bazı dönüşümlere ihtiyaç vardır. Lineer olmayan modellerdeki Kalman Filtre uygulaması Genişletilmiş Kalman Filtre (GKF) olarak adlandırılır.

### 4.2.1 Non-lineer Öngörü

Non-lineer sistem modelinin ifadesi aşağıdaki gibidir. [1,17-19]

$$\left.\begin{aligned} x(k) &= f(x(k-1),u(k),k)+v(k) \\ z(k) &= h(x(k),u(k),k)+w(k) \end{aligned}\right\} \qquad \textbf{(4.19)}$$

GKF'nin ardındaki sır non-lineer modelleri o anki en iyi kestirim [en iyi demek (k|k-1) öngörüsü veya en son (k-1|k-1) kestirimidir] civarında lineerleştirmektir. Lineerleştirme Taylor serisine açılarak yapılır. Bu işlemde önceden bilinen $\hat{x}(k-1\,|\,\mathrm{k}-1)$ kestiriminden faydalanılarak.yapılır.

$$x(k) = f(\hat{x}(k-1 \mid \text{k -1}), \text{u(k)}, \text{k}) + \nabla \text{F}_{\text{x}}[x(k-1) - \hat{x}(k-1 \mid \text{k -1})] + \ldots \quad \textbf{(4.20)}$$

$\nabla F_x$ terimi (f) fonksiyonun x'e göre jacobian ifadesidir.

$$\nabla F_x = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix} \quad \textbf{(4.21)}$$

Daha önceki kestirim metodlarında bildiğimiz gibi x'in öngörü değeri (k-1) gözleme göre koşullu x değişkeninin ortalama değeridir. Yani;

$$\hat{x}(k \mid \text{k -1}) = \text{E}\{\text{x(k)} \mid \text{Z}^{\text{k-1}}\} \quad \textbf{(4.22)}$$

$$\hat{x}(k-1 \mid \text{k -1}) = \text{E}\{\text{x(k -1)} \mid \text{Z}^{\text{k-1}}\} \quad \textbf{(4.23)}$$

$$\hat{x}(k \mid \text{k -1}) = \text{E}\{\text{f}(\hat{\text{x}}(\text{k -1} \mid \text{k -1}), \text{u(k)}, \text{k}) + \nabla \text{F}_{\text{x}}[x(k-1) - \hat{x}(k-1 \mid \text{k -1})] + \ldots + \text{v(k)} \mid \text{Z}^{\text{k-1}}\} \quad \textbf{(4.24)}$$

$$= \text{E}\{\text{f}(\hat{\text{x}}(\text{k -1} \mid \text{k -1}), \text{u(k)}, \text{k} \mid \text{Z}^{\text{k-1}})\} + \nabla \text{F}_{\text{x}}[\hat{x}(k-1 \mid \text{k -1}) - \hat{x}(k-1 \mid \text{k -1})] \quad \textbf{(4.25)}$$

$$= f(\hat{x}(k-1 \mid \text{k -1}), \text{u(k)}, \text{k}) \quad \textbf{(4.26)}$$

u(k) kontrol işaretine bağlı olarak en son kestirilen değerin non-lineer model üzerinden x'in öngörü değeridir. Peki bu durumda kovaryansın durumu nedir.

$$P(k \mid \text{k -1}) = E\{\tilde{x}(k \mid \text{k -1})\tilde{x}(k \mid \text{k -1})^{\text{T}} \mid \text{Z}^{\text{k-1}}\} \quad \textbf{(4.27)}$$

$$\tilde{x}(k \mid \text{k -1}) = \text{x(k)} - \hat{x}(k \mid \text{k -1}) \quad \textbf{(4.28)}$$

$$\approx \text{f}(\hat{\text{x}}(\text{k -1} \mid \text{k -1}), \text{u(k)}, \text{k} + \nabla \text{F}_{\text{x}}[x(k-1) - \hat{x}(k-1 \mid \text{k -1})] + \text{v(k)} - \text{f}(\hat{\text{x}}(\text{k -1} \mid \text{k -1}), \text{u(k)}, \text{k}) \quad \textbf{(4.29)}$$

$$= \nabla \text{F}_{\text{x}}[x(k-1) - \hat{x}(k-1 \mid \text{k -1})] + \text{v(k)} \quad \textbf{(4.30)}$$

$$= \nabla \text{F}_{\text{x}}[\tilde{x}(k-1 \mid \text{k -1})] + \text{v(k)} \quad \textbf{(4.31)}$$

$$= \nabla \text{F}_{\text{x}}.\tilde{x}(k-1 \mid \text{k -1}) + \text{v(k)} \quad \textbf{(4.32)}$$

Buna göre kovaryansın öngörü değeri;

$$P(k \mid \mathrm{k}\text{-}1) = E\{\tilde{x}(k \mid \mathrm{k}\text{-}1)\tilde{x}(k \mid \mathrm{k}\text{-}1)^{\mathrm{T}} \mid Z^{\mathrm{k}\text{-}1}\} \quad \textbf{(4.33)}$$

$$\approx E\{(\nabla \mathrm{F_x}.\tilde{x}(k-1 \mid \mathrm{k}\text{-}1) + \mathrm{v(k)})(\nabla \mathrm{F_x}.\tilde{x}(k-1 \mid \mathrm{k}\text{-}1) + \mathrm{v(k)})^{\mathrm{T}} \mid Z^{\mathrm{k}\text{-}1}\} \quad \textbf{(4.34)}$$

$$= E\{\nabla \mathrm{F_x}.\tilde{x}(k-1 \mid \mathrm{k}\text{-}1).\tilde{x}(k-1 \mid \mathrm{k}\text{-}1)^{\mathrm{T}} \nabla \mathrm{F_x}^{T} \mid Z^{\mathrm{k}\text{-}1}\} + E\{\mathrm{v(k)v(k)}^{\mathrm{T}} \mid Z^{\mathrm{k}\text{-}1}\} \quad \textbf{(4.35)}$$

$$= \nabla \mathrm{F_x}.P(k-1 \mid \mathrm{k}\text{-}1).\nabla \mathrm{F_x}^{T} + Q \quad \textbf{(4.36)}$$

böylece non-lineer modeller için öngörü denklemleri elde edilmiş olur.

Eğer gürültü basit eklenebilir bir gürültü değilse ve modelin fonksiyonunun bir girdisi ise , yani,

$$x(k) = f(x(k-1), u(k), v(k), k) \quad \textbf{(4.37)}$$

Böyle bir durumda çok değişkenli Taylor serisi açılımını kullanmalıyız ki notasyon olarak çok daha komplex ve yorucu bir cebirsel işlemler gerekiyor. Burada böyle bir durum için direkt sonuç denklemini yazalım.

$$P(k \mid \mathrm{k}\text{-}1) = \nabla \mathrm{F_x}.P(k-1 \mid \mathrm{k}\text{-}1).\nabla \mathrm{F_x}^{T} + \nabla F_v.Q.\nabla F_v^{T} \quad \textbf{(4.38)}$$

$\nabla F_v$ f fonksiyonun gürültü vektörüne gire jacobian'ıdır.

Kontrol sinyalini yazacak olursak

$$u(k) = u_n(k) + v(k) \qquad u_n(k): \text{nominal kontrol sinyali} \quad \textbf{(4.39)}$$

$$\nabla F_v = \nabla F_u \quad \textbf{(4.40)}$$

### 4.2.2 Non-lineer Gözlem Modeli

Şimdi yukarıdaki denklemleri non-lineer gözlem modeline (mesafe ve açı sensörü-lazer sensörü) uygulayalım. [1,17-19]

$$z(k) = h(x(k)) + w(k) \quad \textbf{(4.41)}$$

Non-lineer öngörü adımında kullandığımız aynı tekniği gözlem vektörünün öngörüi için kullanalım.

$$\left.\begin{aligned} z(k \mid \mathrm{k}-1) &= \mathrm{E}\{\mathrm{z(k)} \mid \mathrm{Z}^{\mathrm{k}-1}\} \\ z(k \mid \mathrm{k}-1) &= h(\hat{x}(k \mid \mathrm{k}-1)) \end{aligned}\right\} \quad \textbf{(4.42)}$$

Şimdi S inovasyon kovaryans değerini hesaplayalım. İnovasyon gözlenen aktüel değerlerle, öngörü edilen gözlem değerleri arasındaki fark demektir.

$$z(k) \approx h(\hat{x}(k \mid \mathrm{k}-1)) + \nabla \mathrm{H}_{\mathrm{x}}(\hat{x}(k \mid \mathrm{k}-1) - \mathrm{x(k)}) + \ldots + \mathrm{w(k)} \quad \textbf{(4.43)}$$

$$\tilde{z}(k \mid k-1) = z(k) - z(k \mid \mathrm{k}-1) \qquad \text{İnovasyon} \quad \textbf{(4.44)}$$

$$= h(\hat{x}(k \mid \mathrm{k}-1)) + \nabla \mathrm{H}_{\mathrm{x}}(\hat{x}(k \mid \mathrm{k}-1) - \mathrm{x(k)}) + \ldots + \mathrm{w(k)} - h(\hat{x}(k \mid \mathrm{k}-1)) \quad \textbf{(4.45)}$$

$$\left.\begin{aligned} &= \nabla \mathrm{H}_{\mathrm{x}}(\hat{x}(k \mid \mathrm{k}-1) - \mathrm{x(k)}) + \mathrm{w(k)} \\ &= \nabla \mathrm{H}_{\mathrm{x}}(\tilde{x}(k \mid \mathrm{k}-1)) + \mathrm{w(k)} \end{aligned}\right\} \quad \textbf{(4.46)}$$

İnovasyon kovaryans matrisi S formülüze edilirse,

$$S = E\{\tilde{z}(k \mid \mathrm{k}-1).\tilde{z}(k \mid \mathrm{k}-1)^{\mathrm{T}} \mid \mathrm{Z}^{\mathrm{k}-1}\} \quad \textbf{(4.47)}$$

$$\left.\begin{aligned} &= E\{(\nabla H_x.(\tilde{x}(k \mid \mathrm{k}-1)) + \mathrm{w(k)}).(\nabla H_x.(\tilde{x}(k \mid \mathrm{k}-1)) + \mathrm{w(k)})^{\mathrm{T}} \mid \mathrm{Z}^{\mathrm{k}-1}) \\ &= \nabla H_x.E\{\tilde{x}(k \mid \mathrm{k}-1).\tilde{x}(k \mid \mathrm{k}-1)^{\mathrm{T}} \mid \mathrm{Z}^{\mathrm{k}-1}\}\nabla H_x^{\,T} + E\{w(k).\mathrm{w(k)}^{\mathrm{T}} \mid \mathrm{Z}^{\mathrm{k}-1}\} \\ &= \nabla H_x.P(k \mid \mathrm{k}-1).\nabla H_x^{\,T} + R \end{aligned}\right\} \quad \textbf{(4.48)}$$

Yukarıdaki çarpım işleminden gelen $E\{\tilde{x}(k \mid \mathrm{k}-1).\mathrm{w(k)}^{\mathrm{T}} \mid \mathrm{Z}^{\mathrm{k}-1}\}$ terimini sıfır olarak düşünüyoruz. Çünkü gözlemden gelen gürültünün, öngörü hatasından bağımsız olduğunu düşünüyoruz.

Peki yukarıdaki elde etmiş olduğumuz denklemleri güncelleme adımındaki $\hat{x}(k \mid \mathrm{k})$ ve $P(k \mid \mathrm{k})$ kestirimleri için nasıl kullanacağız. $\hat{x}(k \mid \mathrm{k})$ durum vektörünün kestirimi için $\hat{x}(k \mid \mathrm{k}-1)$ öngörüine inovasyonun belli bir kazancıyla çarpımına eşit olduğunu düşünebiliriz. Yani,

$$\hat{x}(k \mid \mathrm{k}) = \underbrace{\hat{x}(k \mid \mathrm{k}-1)}_{tah\min edilen,x} + \underbrace{\mathrm{W}}_{kazanc}\ (\ \underbrace{\mathrm{z(k)}}_{\text{gozlem degerleri}} - \underbrace{\mathrm{h}(\hat{x}(k \mid \mathrm{k}-1)}_{\text{tahmin edilen gozlem degerleri}}\ ) \quad \textbf{(4.49)}$$

İnovasyon değerini yerine koyduğumuzda

$$\begin{aligned} \hat{\mathrm{x}}(\mathrm{k} \mid \mathrm{k}) &= \hat{\mathrm{x}}(\mathrm{k} \mid \mathrm{k}-1) + \mathrm{W}.(\nabla \mathrm{H}_{\mathrm{x}}(\tilde{\mathrm{x}}(\mathrm{k} \mid \mathrm{k}-1)) + \mathrm{w(k)} - \mathrm{x(k)} \\ &= \tilde{\mathrm{x}}(\mathrm{k} \mid \mathrm{k}-1) + \mathrm{W}.\nabla \mathrm{H}_{\mathrm{x}}(\tilde{\mathrm{x}}(\mathrm{k} \mid \mathrm{k}-1) + \mathrm{W}.\mathrm{w(k)} \\ &= [I - \mathrm{W}.\nabla \mathrm{H}_{\mathrm{x}}].\tilde{\mathrm{x}}(\mathrm{k} \mid \mathrm{k}-1) + \mathrm{W}.\mathrm{w(k)} \end{aligned} \quad \textbf{(4.50)}$$

$P(k \mid \mathrm{k})$ kestirimini bulalım.

$$P(k \mid \mathrm{k}) = \mathrm{E}\{\tilde{\mathrm{x}}(\mathrm{k} \mid \mathrm{k}-1).\tilde{\mathrm{x}}(\mathrm{k} \mid \mathrm{k}-1)^{\mathrm{T}} \mid \mathrm{Z}^{\mathrm{k}}\} \quad \textbf{(4.51)}$$

$$\begin{aligned} &= E\{([I - \mathrm{W}.\nabla \mathrm{H}_{\mathrm{x}}].\tilde{\mathrm{x}}(\mathrm{k} \mid \mathrm{k}-1) + \mathrm{W}.\mathrm{w}(\mathrm{k}))([I - \mathrm{W}.\nabla \mathrm{H}_{\mathrm{x}}].\tilde{\mathrm{x}}(\mathrm{k} \mid \mathrm{k}-1) + \mathrm{W}.\mathrm{w}(\mathrm{k}))^{\mathrm{T}} \mid \mathrm{Z}^{\mathrm{k}} \\ &= [I - \mathrm{W}.\nabla \mathrm{H}_{\mathrm{x}}].E\{\tilde{\mathrm{x}}(\mathrm{k} \mid \mathrm{k}-1).\tilde{\mathrm{x}}(\mathrm{k} \mid \mathrm{k}-1)^{\mathrm{T}} \mid \mathrm{Z}^{\mathrm{k}}\}.[I - \mathrm{W}.\nabla \mathrm{H}_{\mathrm{x}}]^{T} + W.E\{w(k).\mathrm{w}(\mathrm{k})^{\mathrm{T}} \mid \mathrm{Z}^{\mathrm{k}}\}.W^{T} \end{aligned} \quad \textbf{(4.52)}$$

$$P(k \mid \mathrm{k}) = [I - \mathrm{W}.\nabla \mathrm{H}_{\mathrm{x}}].P\{\mathrm{k} \mid \mathrm{k}-1).[I - \mathrm{W}.\nabla \mathrm{H}_{\mathrm{x}}]^{T} + W.R.W^{T} \quad \textbf{(4.53)}$$

$$R = E\{w(k).\mathrm{w}(\mathrm{k})^{\mathrm{T}} \mid \mathrm{Z}^{\mathrm{k}}\} \quad \textbf{(4.54)}$$

Denklem 4.50 kovaryans kestirimidir.

Yukarıdaki 4.51 denklemindeki çarpma işleminde $\tilde{x}$'in beklenen değerini sıfır olarak alındı. Kazanç değeri için bazı basit hesaplamalarla yani kestirim hatasının karesinin beklenen değeri minimize edilerek bulunur.

$$J(\hat{x}) = E\{\tilde{x}(k \mid \mathrm{k})^{\mathrm{T}}\, \tilde{x}(k \mid \mathrm{k}) \mid \mathrm{Z}^{\mathrm{k}}\} \quad \textbf{(4.55)}$$

$$= Tr(P(k \mid \mathrm{k})) \quad \textbf{(4.56)}$$

Tr fonksiyonun diferansiyeli için kapalı formu vardır. Eğer herhangi bir A için B simetrik ise

$$\frac{\partial Tr(ABA^{T})}{\partial A} = 2AB \qquad \frac{\partial (AC)}{\partial A} = C^{T} \quad \textbf{(4.57)}$$

Denklem 4.55 $P(k \mid \mathrm{k})$'nın sonuç formülüne uygulandığında

$$\frac{\partial J(\hat{x})}{\partial W} = -2.[I - W\nabla H_{x}].P(k \mid \mathrm{k}-1).\nabla H_{x}^{\,T} + 2WR = 0 \quad \textbf{(4.58)}$$

Denklem 4.56 çözüldüğünde

$$\begin{aligned} W &= P(k \mid \mathrm{k}-1).\nabla \mathrm{H}_{\mathrm{x}}^{\,T}.(\nabla \mathrm{H}_{\mathrm{x}}.P(k \mid \mathrm{k}-1).\nabla \mathrm{H}_{\mathrm{x}}^{\,T} + R)^{-1} \\ W &= P(k \mid \mathrm{k}-1).\nabla \mathrm{H}_{\mathrm{x}}^{\,T}.S^{-1} \end{aligned} \quad \textbf{(4.59)}$$

$$S = \nabla \mathrm{H}_{\mathrm{x}}.P(k \mid \mathrm{k}-1).\nabla \mathrm{H}_{\mathrm{x}}^{\,T} + R \quad \textbf{(4.60)}$$

$$P(k \mid \mathrm{k}-1).\nabla \mathrm{H}_{\mathrm{x}}^{\,T} = W.S \quad \textbf{(4.61)}$$

$$P(k \mid \mathrm{k}) = P(k \mid \mathrm{k}-1) - \mathrm{WSW}^{\mathrm{T}} \quad \textbf{(4.62)}$$

Aşağıdaki tabloda yukarıda hesaplanan Genişletilmiş Kalman Filtre denklemlerinin sonuç denklemleri gösterilmiştir.

**Öngörü adımı:**

$$\hat{x}(k \mid \mathrm{k}\text{-}1) = \overbrace{\mathrm{f}\,(\,\underbrace{\hat{x}(k-1 \mid \mathrm{k}\text{-}1)}_{\text{bir önceki kestirim değeğe}}\,,\underbrace{\mathrm{u(k)}}_{\text{kontrol}},\mathrm{k})}^{\text{sistem modeli}} \tag{4.63}$$

$$\underbrace{P(k \mid \mathrm{k}\text{-}1)}_{\text{tahmini kovaryans}} = \nabla \mathrm{F_x}.\ \underbrace{P(k-1 \mid \mathrm{k}\text{-}1)}_{\text{bir önceki kestirilen kovaryans}}\ .\nabla \mathrm{F_x}^{T} + \underbrace{\nabla F_v.Q.\nabla F_v^{\,T}}_{\text{sistem gürültüsü}} \tag{4.64}$$

$$\underbrace{z(k \mid \mathrm{k}\text{-}1)}_{\text{tahmini gözlemler}} = \overbrace{h(\hat{x}(k \mid \mathrm{k}\text{-}1))}^{\text{gözlem modeli}} \tag{4.65}$$

**Güncelleme adımı:**

$$\underbrace{\hat{x}(k \mid \mathrm{k})}_{yeni-kestirilen-durumlar} = \overbrace{\underbrace{\hat{x}(k \mid \mathrm{k}\text{-}1)}_{tah\,\mathrm{min}\,edilen,x} + W.\underbrace{\upsilon(k)}_{inovasyon}}^{\text{tahmin ve düzeltme}} \tag{4.66}$$

$$\underbrace{P(k \mid \mathrm{k})}_{yeni-kestirilen-ko\,\mathrm{var}\,yans} = P(k \mid \mathrm{k}\text{-}1) - \mathrm{WSW}^{\mathrm{T}} \tag{4.67}$$

$$\begin{aligned} \upsilon(k) &= \overbrace{z(k)}^{gözlemler} - z(k \mid \mathrm{k}\text{-}1) \\ \mathrm{W} &= \underbrace{\mathrm{P(k \mid k\text{-}1).\nabla H_x}^{T}.S^{-1}}_{Kalman-kazancı} \\ S &= \underbrace{\nabla \mathrm{H_x}.P(k \mid \mathrm{k}\text{-}1).\nabla \mathrm{H_x}^{T} + R}_{inovasyon-ko\,\mathrm{var}\,yans} \end{aligned} \tag{4.68}$$

$$\nabla F_x = \frac{\partial f}{\partial x} = \underbrace{\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix}}_{\hat{x}(k-1|k\text{-}1)\text{ de hesaplanan}} \quad \nabla F_U = \frac{\partial f}{\partial u} = \underbrace{\begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \vdots & \cdots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{bmatrix}}_{u(k)'dahesaplanan} \quad \nabla H_x = \frac{\partial h}{\partial x} = \underbrace{\begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_m} \\ \vdots & \cdots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \cdots & \frac{\partial h_n}{\partial x_m} \end{bmatrix}}_{\hat{x}(k|\mathrm{k}\text{-}1)\text{ de hesaplanan}}$$

Yukarıdaki denklemler Genişletilmiş Kalman Filtresi denklemleridir. Lineer Kalman denklemine çok benzemektedir. Lineer Kalman Filtre denklemlerinde jacobian'ın

yerine F ve H geliyor. Aşağıda karşılaştırmak için lineer kalman filtre denklemleri verilmiştir.

**Lineer Kalman Denklemleri**

**Öngörü adımı:**

$$\hat{x}(k \mid \mathrm{k}\text{-}1) = F.\hat{x}(k-1 \mid \mathrm{k}\text{-}1) + \mathrm{B.u(k)} \qquad \textbf{(4.69)}$$

$$P(k \mid \mathrm{k}\text{-}1) = F.P(k-1 \mid \mathrm{k}\text{-}1).\mathrm{F}^T + G.Q.G^T \qquad \textbf{(4.70)}$$

**Güncelleme adımı:**

$$\hat{x}(k \mid \mathrm{k}) = \hat{x}(k \mid \mathrm{k}\text{-}1) + W(k)\upsilon(k) \qquad \textbf{(4.71)}$$

$$P(k \mid \mathrm{k}) = P(k \mid \mathrm{k}\text{-}1) - \mathrm{W(k).S.W(k)}^{\mathrm{T}} \qquad \textbf{(4.72)}$$

$$\begin{aligned} &\upsilon(k) = z(k) - H.\hat{x}(k \mid \mathrm{k}\text{-}1) \\ &\mathrm{W(k)} = \mathrm{P(k \mid k\text{-}1).H}^T.S^{-1} \\ &S = \mathrm{H}.P(k \mid \mathrm{k}\text{-}1).\mathrm{H}^T + R \end{aligned} \qquad \textbf{(4.73)}$$

### 4.2.3 GKF Kestirimi Kararlığı

GKF Kestirim kararlığı testleri durum vektörü ve kovaryans matrisini kestirilmesi için önemlidir. $x_k$ doğru durum vektörü, $\hat{x}_k$ ve $P_k$ ise kestirilen durum vektörü ve kovaryans matrisleri olsun.

$$E[x_k - \hat{x}_k] = 0 \qquad \textbf{(4.74)}$$

$$E[(x_k - \hat{x}_k).(x_k - \hat{x}_k)^T] = P_k \qquad \textbf{(4.75)}$$

Denklem 4.71 ve 4.72'yi sağlayan kestirim metodları kararlıdır denilebilir [15].

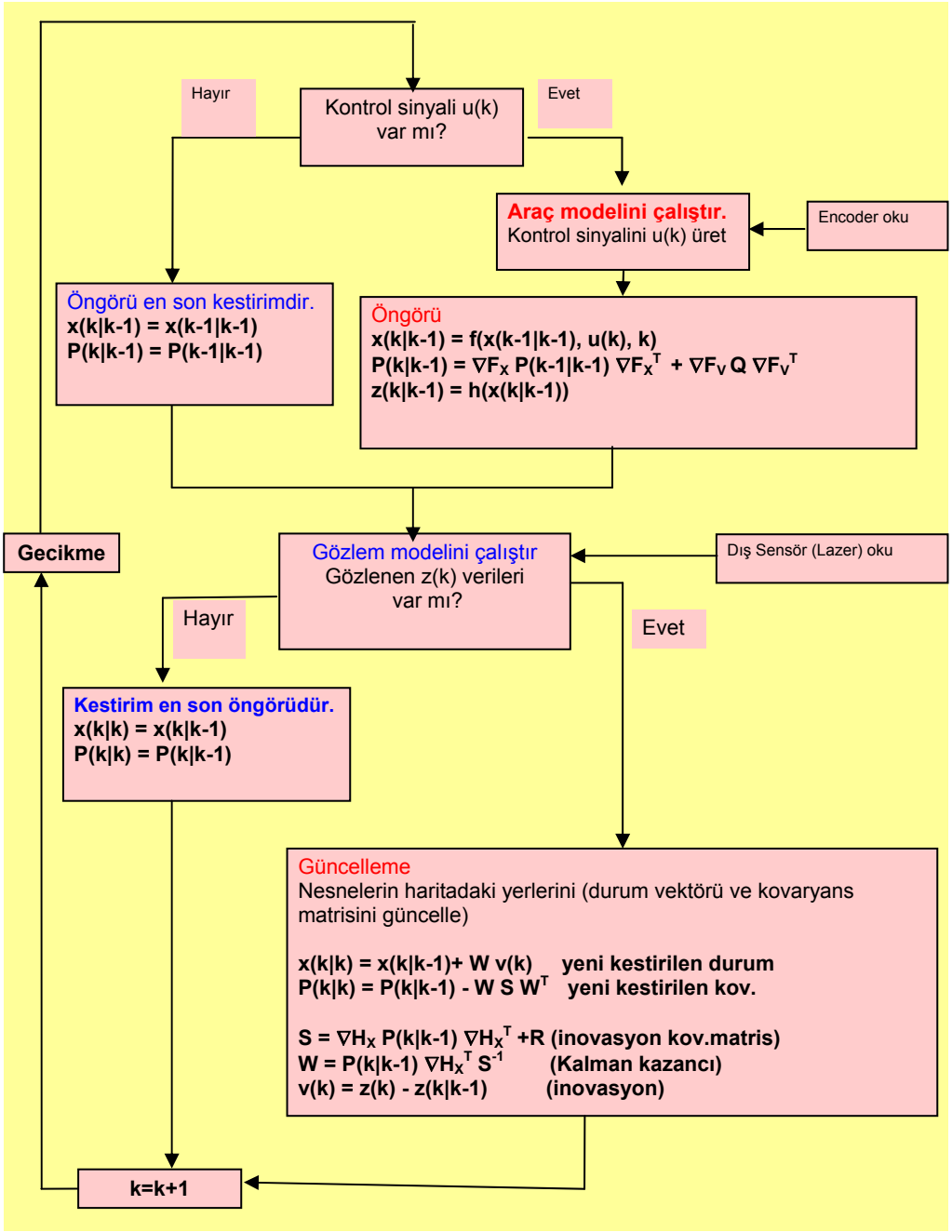
## 4.3 GKF Tabanlı SLAM Algoritmaları

GKF tabanlı SLAM literatürde en iyi bilinen SLAM metodur. GKF tabanlı SLAM uygulamasının akış diyagramı Şekil 4.4'te verilmiştir. Algoritma ilk olarak araca kontrol işaretinin uygulanması ile başlar. Eğer kontrol işareti yoksa en son kestirim değeri öngörü olarak atanır ve lazerden gözlem verilerinin gelip gelmediğine bakılır.

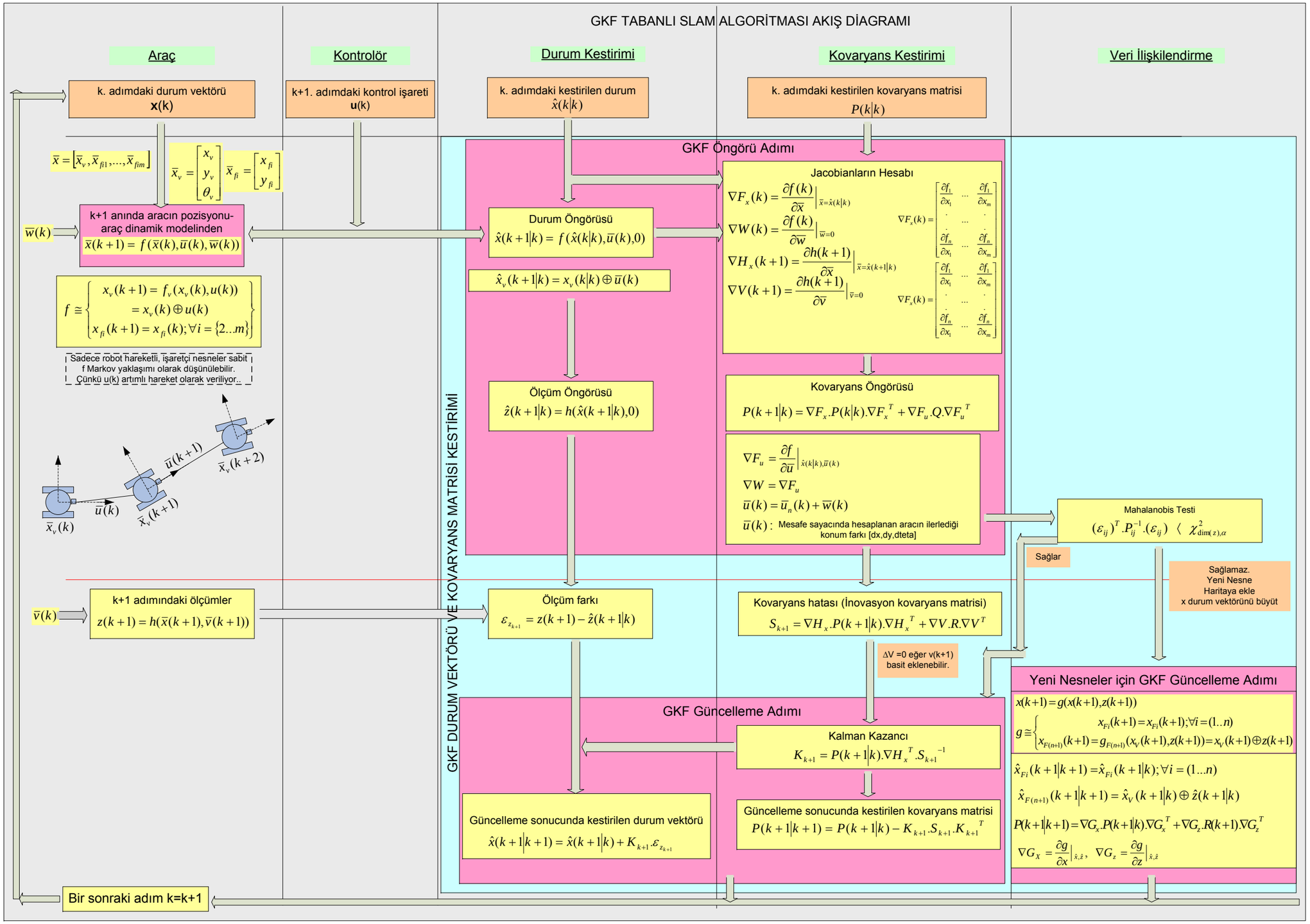
Eğer lazerden veri geliyorsa bu verilere göre araç ve çevrede bulunan işaretçi nesnelerin pozisyonları güncellenir. Araç için pozisyon bilgisi x,y ve aracın yönünü ifade eden θ değerleridir. Işaretçi nesneler için pozisyon bilgisi ise x ve y değerleridir. Kontrol işareti ve aracın bir önceki kestirilen pozisyon değeri, aracın dinamik modeline uygulanarak aracın yeni pozisyonunun öngörüsü elde edilir. Dinamik model lineer olmayan bir fonksiyondur. O yüzden kestirim aracı olarak Genişletilmiş Kalman Filtresi kullanılmıştır. Aracın dinamik modeli için girişler aracın bir önceki kestirilmiş pozisyonu, uygulanan kontrol işareti ve sistem gürültüsüdür. Bu dinamik modelden aracın ve işaretçi nesnelerin öngörülen yeni pozisyonları **x(k|k-1)** elde edilir. Kovaryans matrisi ise aracın ve gözlemlerin belirsizlik elipsini vermektedir. Bu belirsizlik elipsi ne kadar küçük olursa araç ve işaretçi nesne pozisyonlarıda o kadar gerçeğe yakın elde edilirler. Bu elipsleri küçültmek için elde edilen ölçüm verileri ile kovaryans matrisini de güncellememiz gerekmektedir. Güncellenen kovaryans matrisine **P(k|k)** kestirilmiş kovaryans denir. **P(k|k-1)** kovaryans matrisi öngörüsünde (denklem 4.64) bir önceki kestirilen kovaryans matris değeri **P(k-1|k-1)** ve aracın nonlineer modellerinin lineerleştirilmesiyle elde edilen $\nabla \mathbf{F_X}$ ve $\nabla \mathbf{F_v}$ jacobian değerleri kullanılacaktır. Buradaki gürültünün kovaryans matrisi **Q** önemlidir. Sistemden gelen gürültü artımlı mesafe sayıcının hassasiyeti ve gürültüsüne bağlıdır. Artımlı mesafe sayıcı ne kadar hassas ve ne kadar az gürültüyle ölçüm yapıyorsa öngörülerimiz o kadar iyi olacaktır. Peki bu Q hata kovaryans matrisi nasıl elde edilecek. Araç x pozisyonunda hareket ettilerek encoderden gelen değer ile gerçek ölçüm arasında standart sapma hesaplanır. Bu işlem mümkünse birçok kere tekrarlanarak gaussian bir grafik elde edilir. Bu grafiğin standart sapması $\sigma_x^2$'i verir. $\sigma_y^2$ , $\sigma_\theta^2$ değerleride aynı şekilde elde edilerek gürültü kovaryans matrisi Q elde edilir. $\nabla \mathbf{F_X}$ ve $\nabla \mathbf{F_v}$ değerleri aracın kinematik modelinde elde edilirler. Uygulama kısmında bu değerler ayrıntılı olarak verilmiştir. Kovaryans matrisi öngörüinden sonra elde edilen öngörü durum vektörün gözlem modeline uygulanması ile gözlem öngörüi **z(k|k-1)** de hesaplanır. Bu hesaplamalarla durum ve gözlem vektörü, kovaryans matrisi öngörü adımı tamamlanmış olur. Lazerden gelen yeni gözlem verileri varsa güncelleme adımına geçilir. Eğer lazerden herhangi bir veri gelmiyorsa en son öngörü değerleri kestirim değeri olarak kullanılırlar. Lazerden gelen yeni gözlem verileri ile durum vektörü ve kovaryans matrisleri güncellenerek kestirilirler. Kestirme işleminin sonucu aşağıdaki algoritmanın güncelleme kutucuğunda gösterilmiştir. Güncelleme adımı

tamamlandıktan sonra o adımdaki kestirim işlemi biter ve algoritmanın bir çalışma zamanı tamamlanır ve tekrar başa dönülür.

Akış diyagramının gerçek zamanlı uygulaması ve ayrıntılı gösterimi Şekil 4.5'te gösterilmiştir. [17-19]




**Şekil 4.4.** GKF Kestirimi Akış Diyagramı




**Şekil 4.5:** GKF Tabanlı SLAM Akış Diyagramı

### 4.4 GKF Tabanlı SLAM Kararlılığı

GKF tabanlı SLAM'ın kestirdiği durum vektörü ve kovaryans matrisleri kararlılıkları yönünden incelemek için aşağıdaki testler uygulanır.

$x_k$ doğru durum vektörü, $\hat{x}_k$ ve $P_k$ ise kestirilen durum vektörü ve kovaryans matrisleri olsun.

$$D^2 = (x_k - \hat{x}_k)^T . P_k^{-1} . (x_k - \hat{x}_k) \quad \textbf{(4.76)}$$

$$D^2 \leq \chi^2_{r,1-\alpha} \quad \textbf{(4.77)}$$

Denklem 4.75 ve 4.76'yı sağlayan haritalama algoritmasına kararlı diyebiliriz. Bu test Normalize edilmiş kestirim karesel hata olarak adlandırılır. [15]

## 5. SGKF Tabanlı SLAM

SGKF standart GKF'ye benzer bir kestirim yapısı kullanır. SGKF GKF'ye yakın bir kestirim performansını çok düşük bir hesaplama yoğunlugunda sağlayarak işlemci'yi yoran matris işlemlerinden kaynaklanan yükü önemli ölçüde azaltır. Çok büyük hesaplamalar içeren öngörü (prediction) ve güncelleme (update) adımlarını durum vektörlerinin indirgenerek oluşturulan fonksiyonlarıyla daha basit hesaplamalar ile gerçeklemektedir. Tam kestirim problemi böylece geçici olarak indirgenmiş kestirim işlemine dönüşmektedir. İndirgenen durum vektörü ve kovaryans matrisinin öngörü ve güncelleme adımlarından sonra kestirilen değerleri bulunur. Bu işlemlerden sonra genel yada tam güncelleme olarak adlandırılan en son P kovaryans matrisinin ve X durum vektörünün tamamı güncellenerek algoritma tamamlanır. [2,9,17-19]

### 5.1 Filtre Yapısı

Sistemin dinamik modeli

$$X(k+1) = f(X(k), u(k), k) \qquad X \in R^n \tag{5.1}$$

ve gözlem modeli

$$z(k) = h(X(k), k) \qquad z \in R^m \tag{5.2}$$

Bu işlem için önce X durum vektörü iki gruba ayrılır.

$$X = \begin{pmatrix} X_a \\ X_b \end{pmatrix} \qquad X_a \in R^{na}, X_b \in R^{nb}, X \in R^n \tag{5.3}$$

$X_a$ : aktif durumlar
$X_b$ : pasif durumlar
n : toplam durum sayısı

$n_a$ : toplam aktif durum sayısı (Lokal alandaki görünen işaretçi nesnelerin x ve y eksenindeki pozisyonları + Aracın x,y ve θ pozisyonları

$n_b$ : toplam pasif durum sayısı (Lokal alanda olmayıp global alanda bulunan işaretçi nesnelara ait x,y pozisyonları

Bununla ilişkili olarak P kovaryans matrisi de ayrılır.

$$
\begin{aligned}
P &= \begin{pmatrix} P_{aa} & P_{ab} \\ P_{ba} & P_{bb} \end{pmatrix} = E\{(X-\hat{X}).(X-\hat{X})^T\} \in R^{n.n} \\
P_{aa} &= E\{(X_a-\hat{X}_a).(X_a-\hat{X}_a)^T\} \in R^{na.na} \\
P_{bb} &= E\{(X_b-\hat{X}_b).(X_b-\hat{X}_b)^T\} \in R^{nb.nb} \\
P_{ab} &= E\{(X_a-\hat{X}_a).(X_b-\hat{X}_b)^T\} \in R^{na.nb} \\
P_{ba} &= P_{ab}{}^T \in R^{nb.na}
\end{aligned}
\qquad \textbf{(5.4)}
$$

k. adımın başlangıcı $k_1$ olsun bitişi $k_2$ olsun. Yani zaman aralığı $\Omega = \{k \setminus k_1 \le k \le k_2\}$ olsun. Bu durumda model karakteristikleri

$$
\begin{aligned}
\begin{pmatrix} X_a(k+1) \\ X_b(k+1) \end{pmatrix} &= \begin{pmatrix} f_a(X_a(k),u(k),k)+v_a(k) \\ X_b(k)+v_b(k) \end{pmatrix} \\
z(k) &= h(X_a(k),k)+w(k) \\
&\forall (X,u,k)/k \in \Omega
\end{aligned}
\qquad \textbf{(5.5)}
$$

$X_b$ zamanla değişen bir durum değildir ve gözlem fonksiyonu bu durumlardan bağımsızdır.

Model ve gözlem gürültülerinin aşağıdaki karakteristiğe sahip Gaussian birer değişken oldukları varsayılmıştır.

$$
\begin{aligned}
&E\{v_a(k)\} = E\{v_b(k)\} = E\{w(k)\} = \overline{0} \\
&E\{v_a(k).v_b(j)^T\} = E\{v_a(k).w(j)^T\} = E\{v_b(k).w(j)^T\} = \overline{\overline{0}} \\
&E\{v_a(k).v_a(j)^T\} = \delta_{k,j}.Q_{aa}(k) \\
&E\{v_b(k).v_b(j)^T\} = \delta_{k,j}.Q_{bb}(k) \\
&E\{w(k).w(j)^T\} = \delta_{k,j}.R(k)
\end{aligned}
\qquad \textbf{(5.6)}
$$

Ω periyotu boyunca çalışan GKF, aşağıda gösterilen sıkıştırılmış GKF yapısına göre hesaplanır.

$k_1$ başlangıç anında aşağıdaki yardımcı fonksiyonlar oluşturulur.

$$\begin{aligned} &\phi,\psi,\theta,Q_{bb}{}^{*} \\ &\phi,\psi \in R^{na.na} \\ &\theta \in R^{na} \\ &Q_{bb}{}^{*} \in R^{nb.nb} \end{aligned} \qquad \textbf{(5.7)}$$

Bu matrislerin k=$k_1$ anındaki başlangıç değerleri de aşağıdaki gibi olur.

$$\phi(k_1) = I, \psi(k_1) = \overline{\overline{0}}, \theta(k_1) = \overline{\overline{0}}, Q_{bb}{}^{*}(k_1) = \overline{\overline{0}} \qquad \textbf{(5.8)}$$

Her öngörü veya güncelleme adımında , $P_{aa}$ ve $X_a$ üzerinden normal GKF uygulanır. Önce herhangi bir öngörü adımında $P_{aa}$ ve $X_a$ üzerinden standart GKF öngörü adımı $k \setminus k_1 \le k \le k_2$ zaman periyotunda uygulanır ve yardımcı fonksiyonlar aşağıdaki formüllerle güncellenir.

$$\left.\begin{aligned} J = \begin{pmatrix} J_{aa} & J_{ab} \\ J_{ba} & J_{bb} \end{pmatrix}, Q = \begin{pmatrix} Q_{aa} & \overline{\overline{0}} \\ \overline{\overline{0}} & Q_{bb} \end{pmatrix} \\ J_{aa} = (\partial f_a / \partial X_a)\,|_{X_a(k)} \\ \phi(k) = J_{aa}.\phi(k-1) \\ \psi(k) = \psi(k-1) \\ \theta(k) = \theta(k-1) \\ Q^{*}{}_{bb}(k) = Q^{*}{}_{bb}(k-1) + Q_{bb}(k) \end{aligned}\right\} \qquad \textbf{(5.9)}$$

Daha sonra $P_{aa}$ ve $X_a$ üzerinden normal GKF güncelleme adımı (standart GKF güncelleme adımı) uygulanarak aşağıdaki yardımcı fonksiyonlar güncellenir.

$$\left.\begin{aligned} \phi(k) = (I - \mu(k)).\phi(k-1) \\ \psi(k) = \psi(k-1) + \phi^{T}(k-1).\beta(k).\phi(k-1) \\ \theta(k) = \theta(k-1) + \phi^{T}(k-2).H_a{}^{T}(k-1).S(k-1)^{-1}.\upsilon(k-1) \\ H_a(k) = (\partial h / \partial X_a)\,|_{X_a(k)} \\ \beta(k) = H_a{}^{T}(k).S(k)^{-1}.H_a(k) \\ \mu_k = P_{aa}(k).\beta(k) \\ S(k) = H_a(k).P_{aa}(k).H_a{}^{T}(k) + R \\ \upsilon(k) = z(k) - h(X_a(k),k) \end{aligned}\right\} \qquad \textbf{(5.10)}$$

k=$k_2$ anında da genel güncelleme yada tam güncelleme olarak adlandırılan işlem yapılarak durum vektörü ve kovaryans matrisi kestirimi tamamlanmış olur.

$$\left.\begin{array}{c} P_{ab}(k_2) = \phi(k_2).P_{ab}(k_1) \\ P_{bb}(k_2) = P_{bb}(k_1) - P_{ab}(k_1).\psi(k_2).P_{ab}(k_1) + Q_{bb}^{*}(k_2) \\ X_b(k_2) = X_b(k_1) - P_{ab}(k_1).\theta(k_2) \end{array}\right\} \quad \textbf{(5.11)}$$

$X_b(k)$, $P_{bb}(k)$, $P_{ba}(k)$, $P_{ab}(k)$ ait bilgiler sadece $k=k_1$ ve $k=k_2$ zamanlarında belirli oluyor. $k \setminus k_1 < k < k_2$ zaman aralığında ise bu durumlarla ilgili herhangi bir kovaryans ve çapraz kovaryans değerleri bilinmez. Bütün bu bilinmeyen bilgileri yardımcı fonksiyonlar içermektedir. Yani $X_b(k)$ durumlarına ait bütün değerler (kovaryans, çapraz kovaryans ve pozisyon değerleri) bu $\phi(k),\psi(k),\theta(k)$ yardımcı fonksiyonları içerisine sıkıştırılmıştır. [9,17-19]

## 5.2 Gösterim

Herhangi bir güncelleme adımındaki H Jacobian matrisi denklem 5.12'de verilmiştir. [2,9,17-19]

$$\begin{array}{l} H = H(k) = (\partial G / \partial X)|_{X(k)} = (\partial G / \partial (X_a, X_b)|_{X(k)} \\ = \left(\partial G / \partial X_a \quad \partial G / \partial X_b\right) = \left(\partial G / \partial X_a \quad 0\right) = \left(H_a \quad 0\right) \end{array} \quad \textbf{(5.12)}$$

Yukarıdaki denklemden görüldüğü gibi H matrisinin pasif durumlarla ilgili satırı sıfırdır ve $H_a$ hesabıda $X_b$ pasif durumlarından bağımsızdır.

GKF güncelleme adımında kullanılan ara matris hesapları da denklem 5.13'de verilmiştir.

$$\begin{array}{l} P.H^T = P.\left(H_a \quad \overline{\overline{0}}\right)^T = \begin{pmatrix} P_{aa} & P_{ab} \\ P_{ba} & P_{bb} \end{pmatrix}.\begin{pmatrix} H_a{}^T \\ 0 \end{pmatrix} = \begin{pmatrix} P_{aa}.H_a{}^T \\ P_{ba}.H_a{}^T \end{pmatrix} \\ H.P.H^T = \left(H_a \quad \overline{\overline{0}}\right)\begin{pmatrix} P_{aa}.H_a{}^T \\ P_{ba}.H_a{}^T \end{pmatrix} = H_a.P_{aa}.H_a{}^T \end{array} \quad \textbf{(5.13)}$$

$h_k = h(X_a(k), k)$ fonksiyonu $X_b(k)$'dan bağımsızdır ve kovaryans değeri de denklem 5.14'teki gibidir.

$$E\{(h_k - \overline{h}_k).(h_k - \overline{h}_k)^T\} = H(k).P(k).H^T(k) = H_a(k).P_{aa}(k).H_a{}^T(k) \quad \textbf{(5.14)}$$

$\upsilon(k) = z(k) - h(k)$ inovasyonuna ait kovaryans değeri

$$E\{(\upsilon_k - \overline{\upsilon}_k).(\upsilon_k - \overline{\upsilon}_k)^T\} = S(k) = H_a(k).P_{aa}(k).H_a{}^T(k) + R(k) \quad \textbf{(5.15)}$$

$$R(k) = E\{(z(k) - \overline{z}(k)).(z(k) - \overline{z}(k))^T\} \quad \textbf{(5.16)}$$

Kazanç matrisi ise;

$$W(k) = P(k).H^T(k).S(k)^{-1} = \begin{pmatrix} P_{aa}(k).H_a{}^T(k).S(k)^{-1} \\ P_{ba}(k).H_a{}^T(k).S(k)^{-1} \end{pmatrix} = \begin{pmatrix} W_a(k) \\ W_b(k) \end{pmatrix} \quad \textbf{(5.17)}$$

Yukarıdaki denklemlerden;

- Jacobian $H_a$ matrisi $X_b$ pasif durumlarından bağımsızdır.
- S ve $W_a$ matrisleri $H_a$ ve $P_{aa}$'ya bağımlı; $X_b$, $P_{ab}$, $P_{ba}$ ve $P_{bb}$'den bağımsızdır.

sonuçları çıkarılır.

Kovaryans matrisindeki bir güncelleme adımından sonraki değişimi;

$$\left.\begin{aligned} dP = \begin{pmatrix} (P_{aa}.H_a{}^T.S^{-1}.H_a.P_{aa}) & (P_{aa}.H_a{}^T.S^{-1}.H_a.P_{ab}) \\ (P_{aa}.H_a{}^T.S^{-1}.H_a.P_{ab})^T & (P_{ba}.H_a{}^T.S^{-1}.H_a.P_{ab}) \end{pmatrix} = \\ = \begin{pmatrix} P_{aa}.\beta.P_{aa} & \mu.P_{ab} \\ dP_{ab}{}^T & P_{ba}.\beta.P_{ab} \end{pmatrix} \\ \beta = H_a{}^T.S^{-1}.H_a \\ \mu = P_{aa}.\beta \end{aligned}\right\} \quad \textbf{(5.18)}$$

Bir güncelleme adımından sonraki kovaryans matrisi ( basitlik için $k_1$=0 olarak ele alınsın).

$$P(1) = P(0) - dP(0) \quad \textbf{(5.19)}$$

$$P_{aa}(1) = P_{aa}(0) - P_{aa}(0).\beta(0).P_{aa}(0) \quad \textbf{(5.20)}$$

$$P_{ab}(1) = P_{ab}(0) - \mu(0).P_{ab}(0) = (I - \mu(0)).P_{ab}(0) \quad \textbf{(5.21)}$$

$$P_{bb}(1) = P_{bb}(0) - P_{ba}(0).\beta(0).P_{ab}(0) \quad \textbf{(5.22)}$$

Ardışıl k tane güncelleme adımından sonra $P_{ab}$, $P_{bb}$, $P_{ba}$;

$$P_{ab}(k) = \phi(k-1).P_{ab}(0) \quad \textbf{(5.23)}$$

$$P_{bb}(k) = P_{bb}(0) - P_{ba}(0).\psi(k-1).P_{ab}(0) \quad \textbf{(5.24)}$$

$$\begin{aligned}
&\phi(k) = (I - \mu(k)).(I - \mu(k-1))\ldots(I - \mu(0)) \\
&\psi(k) = \sum_{i=1}^{k} (\phi^T(i-1).\beta(i).\phi(i-1)) + \beta(0) \\
&\beta(i) = H_a{}^T(i).S^{-1}(i).H_a(i) \\
&\mu(i) = P_{aa}(i).\beta(i) \\
&H_a(i) = (\partial h(X_a)/\partial X_a)\,|_{X_a(i)}
\end{aligned} \quad \textbf{(5.25)}$$

$\phi(k), \psi(k)$ matrislerinin ardışıl olarak hesaplanması;

$$\begin{aligned}
&\phi(k) = (I - \mu(k)).\phi(k-1) \\
&\psi(k) = \psi(k-1) + \phi^T(k-1).\beta(k).\phi(k-1)
\end{aligned} \quad \textbf{(5.26)}$$

$$\begin{aligned}
&\phi(0) = I - \mu(0) \\
&\psi(0) = \beta(0)
\end{aligned} \quad \text{başlangıç değerleriyle.} \quad \textbf{(5.27)}$$

$X_b$ durumları k güncelleme adımından sonra aşağıdaki şekilde güncellenir.

$$X_b(k) = X_b(0) - P_{ba}(0).\theta(k) \quad \textbf{(5.28)}$$

θ(k) yardımcı fonksiyonu ardışıl olarak aşağıdaki gibi hesaplanır.

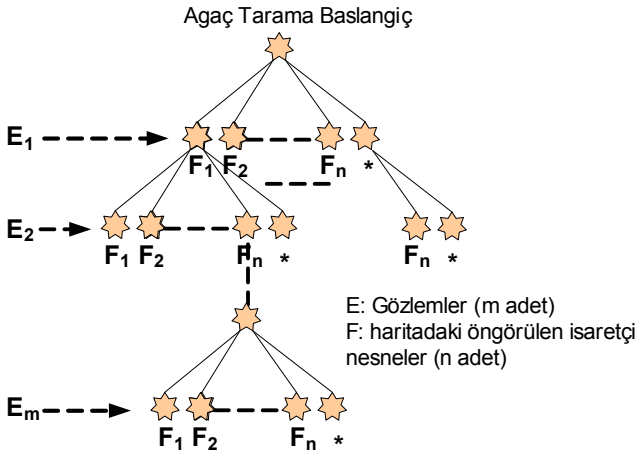
$$\begin{aligned}
&\theta(k) = \theta(k-1) + \phi^T(k-1).S^{-1}(k-1).\upsilon(k-1) \\
&\theta(0) = \overline{0}
\end{aligned} \quad \textbf{(5.29)}$$

# 6. VERİ İLİŞKİLENDİRME

Haritalamanın en zor problemi data association adıyla bilinen veri ilişkilendirme problemidir. Veri ilişkilendirme problemi robotun aynı yerden geçtiğini kestirmesi yani aynı nesneleri ve aynı engelleri tanıyabilmesi problemidir. Eğer robot kapalı bir çevrim şeklinde geziniyorsa başladığı yere tekrar geldiğini anlayabilmesi gerekmektedir.Eğer kapalı çevrim bittiğinde bunu anlayamazsa yörüngesini hatalı oluşturur. Yeni gözlediği verileri haritaya eklemeden önce mevcut haritayla ilişkilendirmesi gerekmektedir. Çünkü haritaya ekledikten sonra bu artık yeni bir engel (işaretçi nesne) olmuştur yani geriye doğru veri ilişkilendirilemez. Yanlış ilişkilendirme harita kestiriminin gerçekten uzaklaşmasına ve bunun neticesinde lokalizasyon algoritmalarının kötü bir şekilde çakılmalarına sebep olur. Bu problem için çok çeşitli teknikler kullanılmaktadır. Biz bu tekniklerden grup doğrulama kısmındaki iki tekniği ele alacağız. Grup doğrulama tekniğinde Bireysel Uyumlu Yakın Komşu (BUYK), Bileşik Uyumlu Dallanma and Bağlanma (BUDB) ve Birleşik Sınırlandırılmış Veri İlişkilendirme (BSVİ) gibi algoritmalar kullanılmaktadır. Görünüm Tanıma tekniği grup doğrulama ile beraber çalıştırılarak güzel veri ilişkilendirmeleri yapılabilir. Görünüm tanıma tekniğine burada girmeyeceğiz. Bu teknik için kamera sensörü gerekmektedir. Görüntü işleme gibi teknik hesaplamalarda bu teknik içerisinde kullanılması gerekebilir. Grup doğrulama tekniğinde BUDB ve BUYK algoritmalarını detaylandıracağız. Her iki teknikte ağaç tarama tekniği olarak bilinmektedir. Her eşlenen gözlem ve işaretçi nesne ikilisi ağacın birer dallarını oluşturmakta ve ağaç en üstten aranarak en uygun çift bulunmaktadır. Gene Grup Doğrulama tekniklerinden olan BSVİ algoritması ise grafik tarama tekniği olarak bilinmektedir. Eşlenen her bir gözlem ve işaretçi nesne ikilisi grafik içerisinde birer noktayı gösterirler. Her bir farklı gözlem ve eşlenen işaretçi nesneler arasında noktalar birleştirilerek geometrik şekiller ortaya konulur ve en uygun çift ataması bu grafik üzerinden taranmaktadır. [8]

Çevredeki $\{E_1 \cdots E_m\}$ işaretçi nesnelere karşılık gelen araçtaki lazer sensörle m tane gözlem $z = \{z_1, \cdots, z_m\}$ yapılsın. Daha önce gözlenip haritaya konulan n tanede

$\{F_1 \cdots F_n\}$ işaretçi nesne olsun. Veri ilişkilendirmekteki amaç her bir $E_i$ ölçümünü haritadaki her bir işaretçi nesnela $F_j$ eşleştiren $H = [j_1 j_2 \cdots j_m]$ hipotezleri üretmektir. Eğer herhangi bir $j_i = 0$ ise $z_i$ haritadaki herhangi bir işaretçi nesne ile eşleşmemiş demektir. Ölçüm-işaretçi nesne eşleştirmesinin geometrik gösterimini aşağıdaki m seviyeli kıyaslama ağacı (interpretation tree) ile gösterebiliriz. İ seviyesindeki her bir nokta i-kıyaslaması (i-interpretation) olarak adlandırılır. Her bir düğümde n+1 adet dal bulunmaktadır. N adet dal, $E_i$ ölçümü için mümkün olan kıyaslamaları ve en son dalda sahte ölçümü ve aynı hipotezde tekrarlanan işaretçi nesneyi ifade etmektedir.




**Şekil 6.1:** $E_1 \ldots E_m$ ölçümleri ile $F_1 \ldots F_n$ işaretçi nesne eşleşmelerinin Kıyaslama Ağacı

Veri ilişkilendirme algoritması sensör ölçümü ile haritadaki işaretçi nesneler arasındaki uygunluğun geçerliliğini hesaplamak için bir şekilde bu kıyaslama ağacından $(n+1)^m$ noktadan m-kıyaslamasını (m-interpretation) doğru hipotez olarak seçmek zorundadır. [2,10,19]

### 6.1 Bireysel Uyumlu Yakın Komşu (BUYK)

Sensör k. adımda m tane işaretçi nesne $E_i$ (i = 1...m) için $\mathbf{z}_{k,i}$ ölçümü yapsın. Veri ilişkilendirmedeki amaç bu ölçümlerle, haritada ki işaretçi nesneları $F_j$ (j=1...n) eşleştirmek ve bu eşleştirmeden en uygun olanların hipotezini $\mathcal{H}_k = [j_1\ j_2\ \ldots\ j_m]$ oluşturmaktır. Eğer $j_i = 0$ ise $\mathbf{z}_{k,i}$ eşleşmesi haritada ki herhangi bir işaretçi nesne'tan gelmemektedir yani yeni bir işaretçi nesne demektir. Veri ilişkilendirmenin temel araçları gözlem öngörüi ve bu öngörü ile sensörden okunan değer arasındaki farktır.

$F_j$'ye ait ölçüm öngörüi araç ve işaretçi nesne lokasyonlarının yer aldığı durum öngörü vektörü $x_{k|k-1}$ kullanılarak nonlineer gözlem fonksiyonu $\mathbf{h}_{k,j}$ ile hesaplanır. Eğer $\mathbf{z}_{k,i}$ gözlemleri $F_j$ işaretçi nesnelarına aitse aşağıdaki eşitliği sağlaması gerekiyor.

$$z_{k,i} = h_{k,j}(x_{k|k-1}) + w_{k,i} \tag{6.1}$$

$\mathbf{w}_{k,i}$ basit eklenebilir , sıfır ortalama değerli, $\mathbf{R}_{k,i}$ kovaryansına sahip ve $\mathbf{v}_k$ sistem gürültüsünden bağımsız gözlem gürültüsüdür. Yukarıdaki denklemin lineerleştirilmiş hali;

$$h_{k,j}(x_{k|k-1}) \cong h_{k,j}(\hat{x}_{k|k-1}) + \mathrm{H}_{k,j}(x_k - \hat{x}_{k|k-1}) \tag{6.2}$$

$$H_{k,j} = \left.\frac{\partial h_{k,j}}{\partial x}\right|_{(\hat{x}_{k|k-1})} \tag{6.3}$$

i.gözlem ile haritadaki j.işaretçi nesnea ait öngörü gözlem değeri arasındaki fark olan inovasyon terimini $\upsilon_{k,ij}$ olarak ifade ediyoruz.(Tabi bu olayların k.adımda olduğu unutulmamalıdır). $S_{k,ij}$'de bu inovasyon terimine ait kovaryans matrisi.

$$\left.\begin{aligned} \upsilon_{k,ij} &= z_{k,i} - h_{k,j}(\hat{x}_{k|k-1}) \\ \mathrm{S}_{k,ij} &= H_{k,j}.P_k.{H_{k,j}}^T + R_{k,i} \end{aligned}\right\} \tag{6.4}$$

Eğer ölçüm ilgili işaretçi nesne'la ilgili ise Mahalanobis uzaklık $D^2_{k,ij}$ testini geçer.

$$D^2{}_{k,ij} = \upsilon^T{}_{k,ij}.S^{-1}{}_{k,ij}.\upsilon_{k,ij} \le \chi^2{}_{d,1-\alpha} \tag{6.5}$$

d = dim($h_{k,j}$) ve 1-α ulaşılması istenen seviye, genellikle %95'tir. Bireysel Uyumlu (BU) olarak adlandırılan ve öngörülen durum vektörüne (predicted state) uygulanan bu test herhangi bir ölçümle uyumlu olan haritadaki işaretçi nesneları belirlemek için kullanılır ve en çok bilinen diğer veri ilişkilendirme algoritmalarınında temelini oluşturur.

Elde edilen gözlemlerden yapılan eşleştirmelerden en uygun olanı seçecek en basit kriter yakın komşuluktur. Yakın komşuluk Mahalanobis uzaklık testini geçenler arasından en küçük Mahalanobis uzaklığa sahip olan gözlem-işaretçi nesne eşleştirmesidir. Aşağıda algoritma akış şeması verilen ve en popüler veri ilişkilendirme algoritması olan BUYK (Bireysel Uyumlu Yakın Komşu) bu temel

üzerine kuruludur. Kavramsal basitliği ve hesaplama verimliliği nedeniyle sık kullanılır. m.n sayısı kadar uygunluk testlerini gerçekler ve haritanın boyuyla lineer olarak artar.

Algoritma : Bireysel Uyumlu Yakın Komşu (BUYK)

for i = 1 to m do {ölçüm Ei}

$D^2_{min}$ = mahalanobis2 ($E_i$, $F_1$)

yakın = 1

for j = 2 to n do {feature $F_j$}

$D^2_{ij}$ = mahalanobis2 ($E_i$, $F_j$)

if $D^2_{ij} < D^2_{min}$ then

yakın = j

$D^2_{min} = D^2_{ij}$

end if

end for

if $D^2_{min} \leq \chi^2_{di,1-\alpha}$ then

$\mathcal{H}_i$ = yakın

else

$\mathcal{H}_i$ = 0

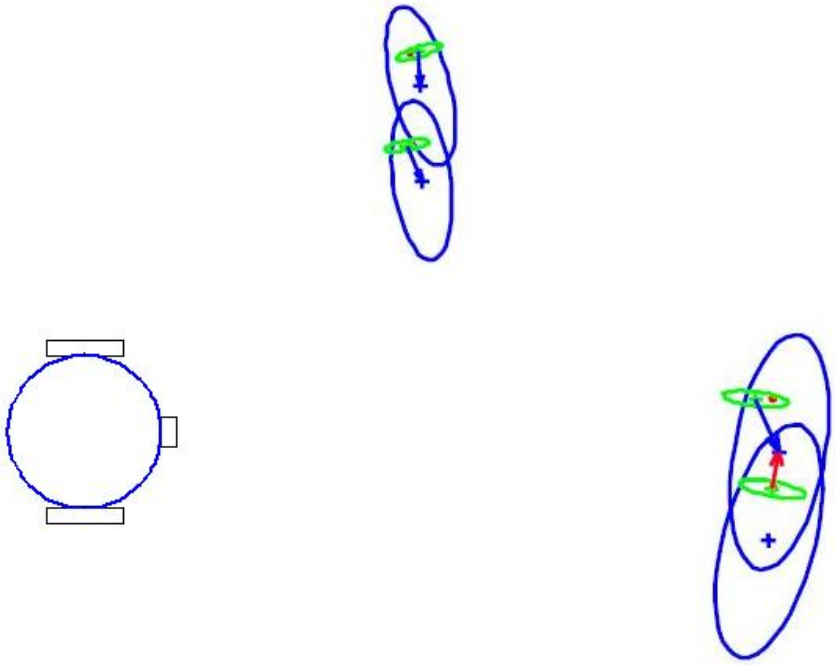
end if

end for

return $\mathcal{H}$

BU ölçüm ve işaretçi nesneler arası bireysel uygunluğa (BU) bakmaktadır. Bununla beraber Bireysel Uyumlu (BU) çiftler tutarlı bir hipotez için Birleşik Uyumluluğa (BRU) bakmaz. Bunun sonucu olarak BUYK tutarsız hipotezler için yüksek bir risk taşıdığı ve bunun sonucu olarak durum vektörü güncellemesi de birbiriyle uyumsuz ölçüm değerleri ile yapıldığı için GKF kestirimleri gerçekten oldukça sapacaktır. Buna da sensör hataları eklendiğinde bu algoritmanın güvenirliği oldukça

düşmektedir. Aşağıdaki örnek resimde bu algoritmanın bir işaretçi nesne ve gözlemi nasıl yanlış bir şekilde hipoteze alındığı görülebilir. [2,10,19]



**Şekil 6.2:** BUYK Algoritmasının Yanlış Bir Eşleştirmesi. Mavi elipsler öngörü durumlarını, yeşiller ise gözlemleri göstermektedir.

### 6.2 Birleşik Uyumlu Dallanma ve Bağlanma (BUDB)

Veri ilişkilendirmenin temelinin ölçüm ve öngörüi işaretçi nesne durumlarının eşleştirilmesi olarak belirtmiştik. Bu eşleşmeler ne kadar doğru ise veri ilişkilendirme algoritmasıda o kadar doğru çalışacaktır. Yanlış ve sahte eşleşmelerin olasılığını limitlenmesi için kurulan eşleşmelerin yeniden tetkik edilmesi gerekmektedir. Hipotezdeki çiftlerin sayısı artarken, sahte çiftlerin hipotezdeki diğer çiftlerle birleşik ortak uyumlu olma olasılığı azalmaktadır. Bütün ölçümlerin diğer uyumlu işaretçi nesnelarla birleşik uyumlu (BRU) olması gerektiği gözden kaçan bir gerçektir. Hipotez $\mathcal{H}_k$ 'nın tutarlı hale getirmek için, ölçümler $\mathbf{h}_{\mathcal{H}k}$ fonksiyonu kullanılarak birleşik öngörü yapılabilir.

$$h_{H_k}(x_{k|k-1}) = \begin{bmatrix} h_{j1}(x_{k|k-1}) \\ \vdots \\ h_{jm}(x_{k|k-1}) \end{bmatrix} \qquad \textbf{(6.6)}$$

o anki kestirim civarında lineerize edilmiş ifadesi

$$h_{H_K}(x_{k|k-1}) \cong h_{H_K}(\hat{x}_{k|k-1}) + \mathrm{H}_{H_K}(x_k - \hat{x}_{k|k-1}); \quad \mathrm{H}_{H_K} = \begin{bmatrix} \mathrm{H}_{j1} \\ \vdots \\ \mathrm{H}_{jm} \end{bmatrix} \tag{6.7}$$

Bileşik inovasyon ve kovaryans matrisi ;

$$\upsilon_{H_K} = z_k - h_{H_K}(\hat{x}_{k|k-1}) \tag{6.8}$$

$$S_{H_K} = \mathrm{H}_{H_K}.P_k.\mathrm{H}_{H_K}{}^T + R_{H_K} \tag{6.9}$$

Aşağıdaki testi geçen ölçümler $z_k$, $H_k$'ya göre ilişkili işaretçi nesnelarla uyumlu denilebilir. Bu teste bileşik uygunluk (JC) testi denir.

$$D^2{}_{H_K} = \upsilon^T{}_{H_K}.S^{-1}{}_{H_K}.\upsilon_{H_K} \le \chi^2{}_{d,1-\alpha}, \quad \mathrm{d} = \dim(\mathrm{h}_{Hk}) \tag{6.10}$$

BRU testi aşağıda algoritması verilen BUDB (Bileşik Compatiblity Dallanma and Bağlanma) veri ilişkilendirme tekniğinin temelini oluşturmaktadır.

Algoritma: Bileşik Uyumlu Dallanma and Bağlanma (BUDB)

function BUDB ($E_1 \ldots E_m$, $F_1 \ldots F_n$):

Best = []

BUDB_R ([],1)

return Best

function BUDB_R (*H*,k) % $E_k$ gözlemi için çiftleri bulur.

if k > m then % boşta gözlem kaldı mı ?.

  if pairings (*H*) > pairings (Best) then

    Best = *H*

  end if

else

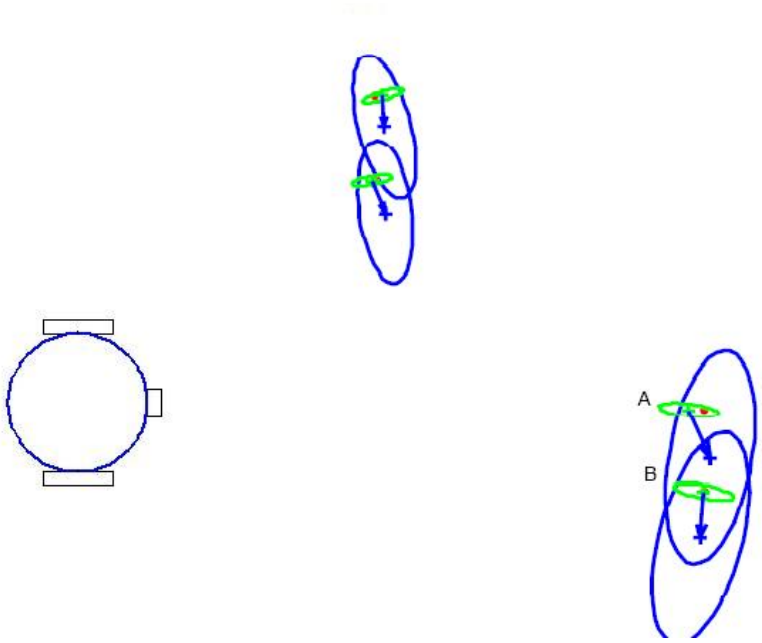
  for j = 1 to n do

    if bireysel_uyumlu (k,j) and then bileşik_uyumlu (*H*,k,j)

```
        then
            BUDB_R ([H j], k +1)   % Ek Fj eşleşmesi kabul edildi.
        end if
    end for
    if pairings(H) + m – k > pairings(Best) then   %daha iyisi varmı ?
        BUDB_R ([H 0], k +1)   % Yıldız düğüm, Ek eşleşemedi.
    end if
end if
```

BUDB, en büyük numaraya sahip bileşik uyumlu (BRU) çiftleri içeren hipotezler için kıyaslama ağacını çapraz tarayarak veri ilişkilendirme yapmaktadır. $H_i$ hipoteziyle ilgili i.seviyedeki bir nodun kalitesi, o noddan elde edilen boş olmayan çiftlerin sayısına bağlıdır. Böylece en iyi (Best) mümkün hipotezden düşük kaliteye sahip nodlar taranmamaktadır. Bu yapıya tarama sınırlaması (Bağlanma Taraması) denilmektedir. Mahalanobis uzaklığı $D^2_{H\ i}$ kullanan yakın komşuluk kuralı için sezgisel olarak kullanılmaktadır. Böylece hipotezle ilişkili nodlardan önce yüksek derecede bileşik uyumluğa sahip nodlardan araştırılmaya başlanmaktadır. $\mathbf{h}_{Hi}$ ve $\mathbf{S}_{Hi}$ 'nin boyutları $H_i$ hipotezinin büyüklüğü ile artmaktadır. Buda ek bir hesaplama yükü getirmektedir. Mahalanobis uzaklığındaki ters alma işlemleri ek hesaplama yükleri getirmektedir. Mahalanobis uzaklığını hesaplamak için bazı teknikler önerilmiştir. Bunlardan birisi Cholesky faktörizasyonunu kullanan tekniktir, uygulamada bu teknik kullanılmıştır.

Veri ilişkilendrime algoritmalarında sensör gürültüleri ve araçların pozisyonlarındaki belirsizlik çok önemlidir. Araç pozisyonundaki belirsizlik arttıkça aracın pozisyonu ve işaretçi nesneler arasındaki geometrik yapı değişecek ve bu da yakın komşu (YK) gibi basit algoritmaların Şekil 6.2'deki çakılmalarına sebep olacaktır. Böyle bir durumda BUDB gibi algoritmalar Şekil 6.3'teki gibi daha doğru veri ilişkilendirme yapmaktadırlar. Bunun nedeni ise korelasyonlar aracılığıyla araç hatasından bağımsız işaretçi nesneların arasındaki rölatif lokasyonuda hesabın içerisinde değerlendirmesinden kaynaklanmaktadır.



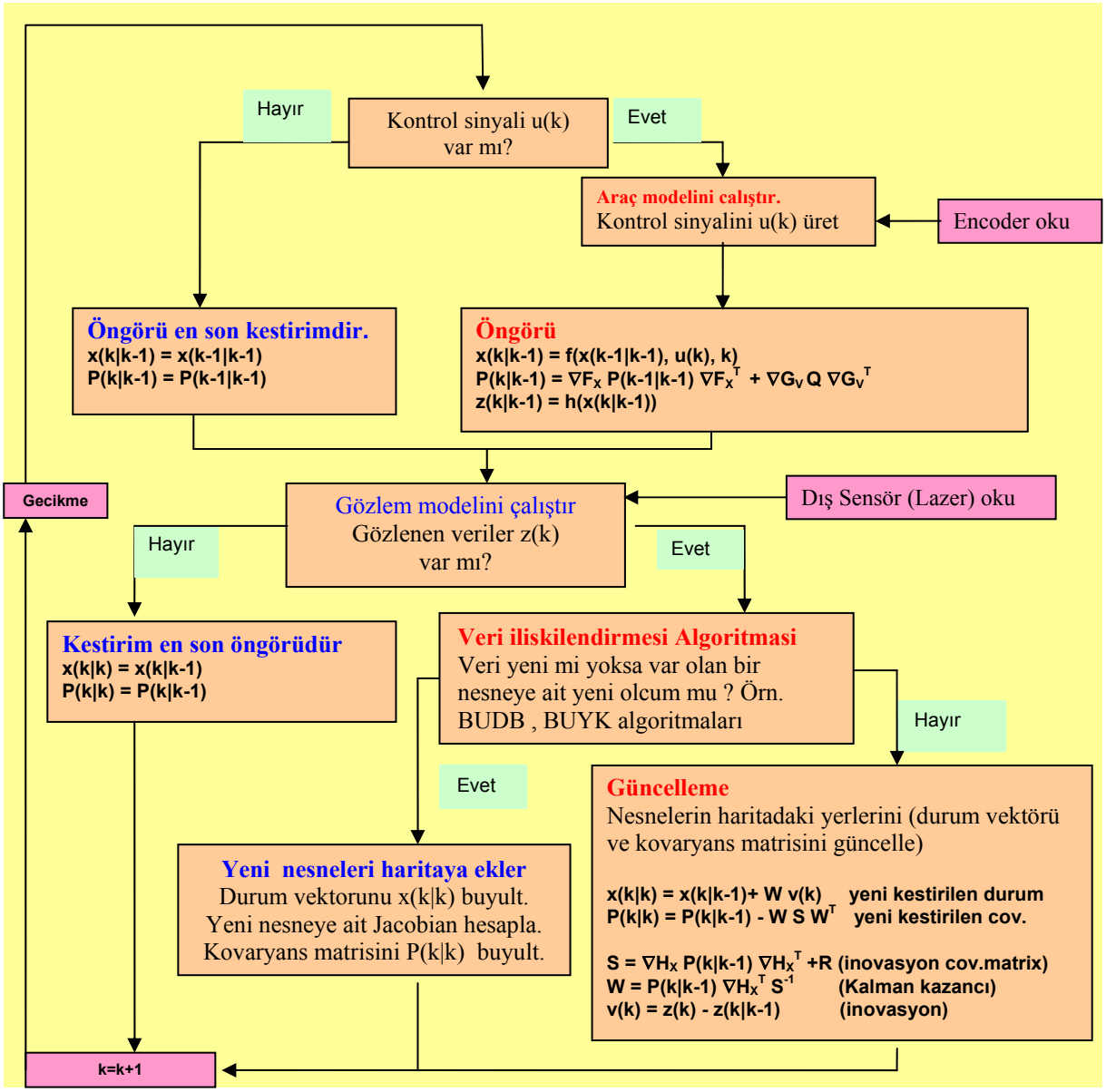

**Şekil 6.3:** BUDB Algoritması ile Bir Eşleme

BUDB algoritmasındaki hesaplama araçtaki hataya göre değişmiyor çünkü ölçümlerin birleşik uyumluluğu (BRU) temelde araç hatasına bağlı olan salt hata yerine sensör ve harita hassasiyetine bağlı olan relatif hataya bağlıdır. BUDB veri ilişkilendirme algoritması eğer lineerizasyon hatası makul seviyede ise sağlam bir veri ilişkilendirmesi yapmaktadır. [2,10,19]

### 6.2.1 Veri İlişkilendirme ve SLAM Algoritmaları

Bölüm 4.3'te anlatılan algoritmaya ek olarak veri ilişkilendirme eklenmiştir. Veri İlişkilendirme algoritmaları GKF öngörü adımı ve gözlem modelinin çalıştırılmasından sonra elde edilen öngörü durum vektörü ve gözlemlere göre ilişkilendirme yapmaktadır. Veri İlişkilendirmeden sonra GKF güncelleme adımına geçilmektedir. Çünkü GKF güncelleme adımı veri ilişkilendirme algoritmalarının ürettiği sonuçlara göre yapılmaktadır. Gözlemler hangi işaretçi nesnelere aitse o işaretçi nesne lokasyonları güncellenmekte ve eğer gözlem haritada olmayan işaretçi nesnea aitse durum vektörü ve kovaryans matrisi büyültülerek yeni duruma göre güncelleme adımı yapılmaktadır. İşte veri ilişkilendirmenin doğru bir haritalama işlemi için önemi buradan kaynaklanmaktadır. Yanlış işaretçi nesne eşleştirmesi veya haritada olan işaretçi nesneye ait bir gözlemi yeni bir işaretçi nesneye aitmiş gibi yapılan eşleştirme ve atamalar haritanın gerçek haritadan uzaklaşmasına ve GKF öngörü ve güncelleme adımlarınında yanlış yapılmasına sebep olmaktadırlar. Şekil

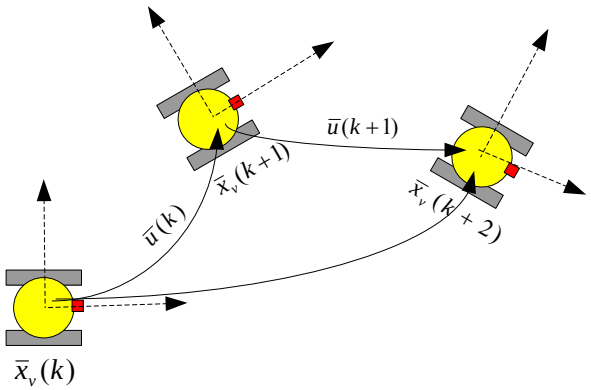
4.5'te veri ilişkilendirmeli GKF kestirimi için real-time uygulama için bir döngülük bir akış şeması gösterilmiştir. Şekil 6.4'te daha yalın bir akış şeması görülmektedir. SGKF içinde veri ilişkilendirmenin SLAM algoritmasındaki yeri değişmemektedir. Çünkü bölüm 5'te anlatıldığı gibi SGKF'nin GKF'den farkı öngörü ve güncelleme adımlarının optimize edilmesi ve bu adımların indirgenmiş fonksiyonlar üzerinden yapılarak en son olarak tam güncelleme yapılması mantığına dayanmaktadır.




**Şekil 6.4:** Veri İlişkilendirme algoritması ile SLAM akış şeması

# 7. UYGULAMA

## 7.1 Diferansiyel Araç Modeli ve Hesaplanması




**Şekil 7.1:** Diferansiyel Araç Hareket Eksenleri Dönüşümü

Aracın 1. pozisyondan 2. pozisyona gidebilmesi oradan da 3. pozisyona gidebilmesi ve bu pozisyonlarda aracın konumunun kestirilmesi için koordinat eksenlerinin transformasyonundan faydalanılacaktır. Burada iki tane $\oplus$ ve $\Theta$ operatörlerini tanımlanacaktır. Yukarıdaki şekli gözönüne alındığında $x_{1,3} = x_{1,2} \oplus x_{2,3}$ olduğu görülür. $x_1=(x_1,y_1,\theta_1)$ ve $x_2=(x_2,y_2,\theta_2)$ için;

$$\bar{X}_1 \oplus \bar{X}_2 = \begin{bmatrix} x_1 + x_2 Cos\theta_1 - y_2 Sin\theta_1 \\ y_1 + x_2 Cos\theta_1 - y_2 Sin\theta_1 \\ \theta_1 + \theta_2 \end{bmatrix} \quad \textbf{(7.1)}$$

$$\Theta x_1 = \begin{bmatrix} -x_1 Cos\theta_1 - y_1 Sin\theta_1 \\ x_1 Sin\theta_1 - y_1 Cos\theta_1 \\ -\theta_1 \end{bmatrix} \quad \textbf{(7.2)}$$

Denklem 7.1 ve 7.2 bir eksendeki pozisyonları diğer bir eksendeki karşılıklarını ifade etmektedir. Araç önceki mesafe sayacından lokalizasyon hesabıyla elde edilen pozisyonundan $x_v(k)$, yeni bir hareketle u(k) yeni mesafe sayacından lokalizasyon hesabıyla (dead-reckoned) elde ettiği pozisyonuna $x_v(k+1)$ gittiğinde yeni mesafe

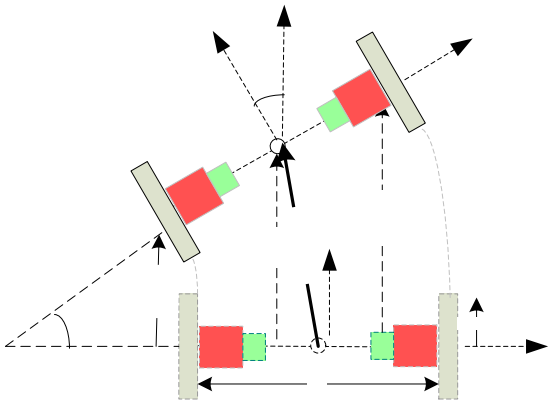
sayacından lokalizasyon hesabıyla elde edeceği pozisyonunu denklem 7.3 ile hesaplanır.

$$x_v(k+1) = x_v(k) \oplus u(k) \qquad \textbf{(7.3)}$$

Buradaki u(k) yeni hareket yani aracı 1 nolu eksenden 2 nolu eksene taşıyan kontrol işaretinin bir sonucu ve verileri mesafe sayacından lokalizasyon hesabından elde edilir. [1]

$$u(k) = \Theta x_v(k) \oplus x_v(k+1) \qquad \textbf{(7.4)}$$

Bu ifadeler araç modeli halinde üretilecek olursa simulasyonlarda kullanılabilecek şekilde denklemler ayrık hale dönüştürülmüş bir non-lineer model haline getirilmiş olacaktır.



**Şekil 7.2:** Diferansiyel Aracın Hareketi

Diferansiyel araç bir yerden bir yere ötelenğinde aracın pozisyonunun bulunması için gerekli olan kinematik denklemler.

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

$$c_m = \frac{\pi . D_n}{n . C_e} \tag{7.5}$$

$$\left.\begin{aligned} \Delta V_L &= c_m . N_L \\ \Delta V_R &= c_m . N_R \\ \Delta V &= \frac{\Delta V_R + \Delta V_L}{2} \\ \Delta\theta &= \frac{\Delta V_R - \Delta V_L}{L} \end{aligned}\right\} \tag{7.6}$$

$$\begin{bmatrix} dx \\ dy \\ d\theta \end{bmatrix} = \begin{bmatrix} \Delta V . Cos(\theta(k+1)) \\ \Delta V . Sin(\theta(k+1)) \\ \Delta\theta \end{bmatrix} \tag{7.7}$$

$$\left.\begin{aligned} x_v(k+1) &= x_v(k) + \Delta V . Cos(\theta(k) + \Delta\theta) \\ y_v(k+1) &= y_v(k) + \Delta V . Sin(\theta(k) + \Delta\theta) \\ \theta(k+1) &= \theta(k) + \Delta\theta \end{aligned}\right\} \tag{7.8}$$

$$\left.\begin{aligned} \Delta V &= r.(\Delta\omega_R + \Delta\omega_L)/2 \\ \Delta\omega &= r.(\Delta\omega_R - \Delta\omega_L)/L \\ \Delta\theta &= \Delta\omega \end{aligned}\right\} \tag{7.9}$$

Denklem 7.10'da gösterilen hareket vektörü aracı 1.eksenden 2.eksene taşıyan kontrol işaretinin ürettiği mesafe sayacından lokalizasyon hesabı ile hesaplanan vektördür. Aracın hareketleri artımlı hareket olarak düşünülebilir. Yani araca uygulanan kontrol işaretinin etkisiyle aracın x yönündeki pozisyonu $d_x$ , y yönündeki pozisyonu $d_y$ ve aracın yönü ise dθ kadar değişir. Bu artırımlı hareketle aracın yeni pozisyonu önceki pozisyonu ile artırımlı hareketin transformasyonel toplanması ile elde edilir. Bu transformasyonel ifade denklem 7.11 ile gösterilmiştir. Aracın dinamik modeli bize aracın eski pozisyonu ve kontrol işareti ile yeni pozisyonunun bilgisini vermektedir. Bu bağlamda denklem 7.11 ile gösterilen formül aracın modelinin bir ifadesi olmaktadır. Aracın önceki pozisyonu ile uygulanan kontrol işaretinin etkisiyle elde edilen x,y ve θ'deki değişimin transformasyonel toplanması aracın yeni pozisyonunu belirleyen bir model olmaktadır.

$$u(k) = \begin{bmatrix} dx \\ dy \\ d\theta \end{bmatrix} \tag{7.10}$$

$$\left.\begin{aligned} \overline{x}_v(k+1) &= f(\overline{x}_v(k), \overline{u}(k)) \\ \overline{x}_v(k+1) &= \overline{x}_v(k) \oplus \overline{u}(k) \end{aligned}\right\} \tag{7.11}$$

Denklem 7.11 ile verilen araç modeline ait Jacobianlar denklem 7.12 ve 7.13' deki formüllerle hesaplanır.

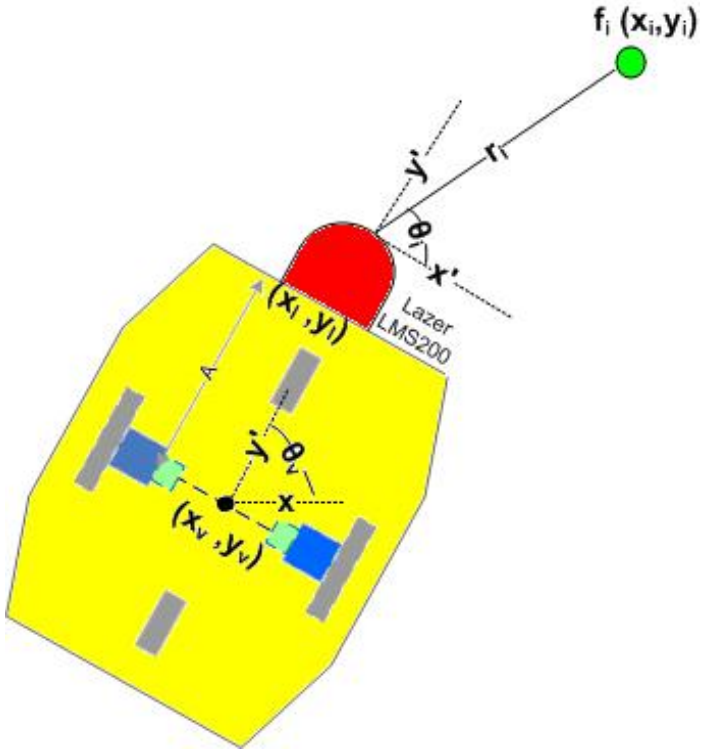
$$JFx(\overline{x}_{vk}, \overline{u}_k) \cong \frac{\partial(\overline{x}_{vk} \oplus \overline{u}_k)}{\partial \overline{x}_{vk}} = \begin{bmatrix} 1 & 0 & -dx.\sin\theta_1 - dy.\cos\theta_1 \\ 0 & 1 & dx.\cos\theta_1 - dy.\sin\theta_1 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.12}$$

$$JFu(\overline{x}_{vk}, \overline{u}) \cong \frac{\partial(\overline{x}_{vk} \oplus \overline{u}_k)}{\partial \overline{u}_k} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.13}$$

Aracın x,y ve θ'daki yer değiştirmeleri mesafe sayıcından alınan bilgiyle yapılmaktadır. Buda belli bir gürültüyü beraberinde getirmektedir. Bu gürültü sistem gürültü kovaryans matrisi Q ile ifade edilecektir. Q x,y ve θ'daki değişimlerin hangi standart sapma ile gerçekleşeceğinin bilgisini taşımaktadır. Burada iki varsayım yapılmıştır. Bu varsayımlardan birincisi x,y ve θ'daki gürültülerin birbirinden bağımsız oldukları, ikincisi ise bu gürültülerin gaussian olduklarıdır. Q sistem gürültü kovaryans matrisi denklem 7.14'de verilmiştir. d-r ifadesi mesafe sayacından lokalizasyon hesabının (dead-reckoned) kısaltmasıdır. [1,17-19]

$$Q = \mathrm{U}_{\text{d-r}} = \begin{bmatrix} \sigma^2_{(d-r)x} & 0 & 0 \\ 0 & \sigma^2_{(d-r)y} & 0 \\ 0 & 0 & \sigma^2_{(d-r)\theta} \end{bmatrix} \tag{7.14}$$

## 7.2 Gözlem Modeli Hesaplamaları




**Şekil 7.3:** Diferansiyel Aracın Lazerle Gözlem Yapması

Araçtaki lazerle yapılan ölçümler lazer ve işaretçi nesne arasındaki mesafe $r_i$ ve lazerle işaretçi nesne arasındaki $x'$ eksenine göre olan açı $\theta_i$ 'yi vermektedir. Ölçüm vektörü $r_i$ ve $\theta_i$ bilgilerini veren iki elemanlı bir vektördür.

$$(x_l, y_l) = (x_v, y_v+A) \quad \textbf{(7.15)}$$

$$\bar{z}(k) \cong \begin{bmatrix} r_i \\ \theta_i \end{bmatrix} \quad \textbf{(7.16)}$$

$$\bar{z}(k) = h(\bar{x}(k), \bar{w}(k)) \quad \textbf{(7.17)}$$

Denklem 7.16'da verilen gözlem modeli işaretçi nesne ve lazer arasındaki geometrik ilişkinin bir ifadesidir. Gözlem modeli elde edilen $r_i$ ve $\theta_i$ ölçümlerinden işaretçi nesne'nin x ve y eksenindeki değerleri $x_i$ ve $y_i$ 'yi vermektedir. $x_i$ ve $y_i$ değerleri ise Denklem 7.17 ile hesaplanmaktadır. Bu sebeple, denklem 7.17 sistemin gözlem modelidir.

$$\bar{z}(k) \cong \begin{bmatrix} \sqrt{(x_i - x_l(k))^2 + (y_i - y_l(k))^2} \\ \arctan(\dfrac{y_i - y_l(k)}{x_i - x_l(k)}) - \theta_v \end{bmatrix} \quad \textbf{(7.18)}$$

7.17'de verilen denklemi cebirsel olarak çözüp $x_i$ ve $y_i$ değerlerinin matematiksel olarak ifadesini bulmak gereksizdir. Matlab bu işlemi pol2cart komutu ile yapmaktadır. İki nokta arasındaki mesafe ve açı değerlerinden x ve y eksenlerindeki değişim bu komutla bulunabilmektedir.

Denklem 7.17 ile verilen gözlem modeline ait jacobian hesabı denklem 7.18 ve 7.22 arasındaki denklemlerle gösterilmiştir.

$$\nabla Hx_l = \frac{\partial h}{\partial x} = \begin{bmatrix} \dfrac{\partial r}{\partial x_l} & \dfrac{\partial r}{\partial y_l} & \dfrac{\partial r}{\partial \theta_l} \\ \dfrac{\partial \theta}{\partial x_l} & \dfrac{\partial \theta}{\partial y_l} & \dfrac{\partial \theta}{\partial \theta_l} \end{bmatrix} \quad \textbf{(7.19)}$$

$$\nabla Hx_l = \frac{\partial h}{\partial x} = \begin{bmatrix} \dfrac{x_i - x_l(k)}{r} & \dfrac{y_i - y_l(k)}{r} & 0 \\ \dfrac{y_i - y_l(k)}{r^2} & -\dfrac{x_i - x_l(k)}{r^2} & -1 \end{bmatrix} \quad \textbf{(7.20)}$$

$$\nabla Hx_f = \frac{\partial h}{\partial xf_{xi,yi}} = \begin{bmatrix} \dfrac{\partial r}{\partial x_i} & \dfrac{\partial r}{\partial y_i} \\ \dfrac{\partial \theta}{\partial x_i} & \dfrac{\partial \theta}{\partial y_i} \end{bmatrix} \quad \textbf{(7.21)}$$

$$\nabla Hx_f = \frac{\partial h}{\partial xf_{xi,yi}} = \begin{bmatrix} -\dfrac{x_i - x_l(k)}{r} & -\dfrac{y_i - y_l(k)}{r} \\ -\dfrac{y_i - y_l(k)}{r^2} & \dfrac{x_i - x_l(k)}{r^2} \end{bmatrix} \quad \textbf{(7.22)}$$

$$\nabla Hx = \begin{bmatrix} \nabla Hx_l & 0 \\ 0 & \nabla Hx_f \end{bmatrix} \quad \textbf{(7.23)}$$

Gözlem modelinin gürültü kovaryans matrisi denklem 7.23'te verilmiştir. Burada lazer mesafe gürültüsü ve açı gürültüsü arasında herhangi bir korelasyon bulunmadığı ve hatalarında gaussian olduğu varsayılmıştır. R matrisinin doğru bulunabilmesi için lazerin doğru modellenmesi gerekmektedir. Simulasyonda bu değerler rastgele seçilmektedir. Gerçek zamanlı uygulamalarda lazerle mesafe ve açı ölçümleri yapılarak gerçek değerlerinden ne kadar saptıkları belirlenmelidir. Bu hatalardan elde edilen dağılımın standart sapmaları hesaplanarak R kovaryans matrisi oluşturulacaktır.

$$R = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} \quad \textbf{(7.24)}$$

### 7.3 Yeni İşaretçi nesnelere ait durumların güncellemesi

Araç yeni bir işaretçi nesne gözlediğinde bu işaretçi nesnenin pozisyonu **X** durum vektörünün son satırına eklenerek buna ait hata kovaryansıda kovaryans matrisine eklenmektedir.

Burada eski durum vektörünü yeni durum vektörüne çeviren bir **Y** işaretçi nesne başlangıç fonksiyonu tanımlansın.

$$\bar{x}(k|k)^* = Y(\bar{x}(k|k), \bar{z}(k), \bar{x}_v(k \mid \mathrm{k})), \quad \text{büyütülmüş X durum vektörü} \quad \textbf{(7.25)}$$

$$\bar{x}(k \mid \mathrm{k})^* = \begin{bmatrix} \bar{x}(k \mid \mathrm{k}) \\ \mathrm{g}(\bar{\mathrm{x}}(\mathrm{k}), \bar{\mathrm{z}}(\mathrm{k}), \bar{\mathrm{x}}_1(k \mid \mathrm{k})) \end{bmatrix} \quad \textbf{(7.26)}$$

$$\bar{x}(k \mid \mathrm{k})^* = \begin{bmatrix} \bar{x}(k \mid \mathrm{k}) \\ x_l + r.\cos(\theta + \theta_v) \\ y_l + r.\sin(\theta + \theta_v) \end{bmatrix} \quad \textbf{(7.27)}$$

Yeni işaretçi nesneye ait koordinat bilgisi **g** fonksiyonu ile ifade edilir.

$$\bar{x}_{fyeni} = \mathrm{g}(\bar{\mathrm{x}}(\mathrm{k}), \bar{\mathrm{z}}(\mathrm{k}), \bar{\mathrm{x}}_1(k \mid \mathrm{k})) \quad \textbf{(7.28)}$$

$$\bar{x}_{fyeni} = \begin{bmatrix} x_l + r.\cos(\theta + \theta_v) \\ y_l + r.\sin(\theta + \theta_v) \end{bmatrix} \quad \textbf{(7.29)}$$

Yeni bir işaretçi nesne eklendiğinde **P** kovaryans matrisi nasıl oluşur.

$$P(k \mid \mathrm{k})^{*} = \nabla Y_{x,z} . \begin{bmatrix} P(k \mid \mathrm{k}) & 0 \\ 0 & R \end{bmatrix} . \nabla Y_{x,z}{}^{T}$$ , büyütülmüş kovaryans matrisi **(7.30)**

$$\bar{x}_{fyeni} = \begin{bmatrix} x_l + r.\cos(\theta + \theta_v) \\ y_l + r.\sin(\theta + \theta_v) \end{bmatrix} \quad \textbf{(7.31)}$$

$$\nabla Y_{x,z} = \begin{bmatrix} I_{nxn} & 0_{nx2} \\ \nabla G_x & \nabla G_z \end{bmatrix}$$, Y fonksiyonuna ait Jacobian **(7.32)**

Yeni işaretçi nesneler durum vektöründeki diğer işaretçi nesnelerden bağımsız oldukları için $\nabla G_x = 0$ olmaktadır.

Bu durumda yeni kovaryans matrisi denklem 7.32'deki hale gelir. [1,17-19]

$$P(k \mid \mathrm{k})^{*} = \begin{bmatrix} P(k \mid \mathrm{k}) & 0 \\ 0 & \nabla G_z . R . \nabla G_z{}^{T} \end{bmatrix} \quad \textbf{(7.33)}$$

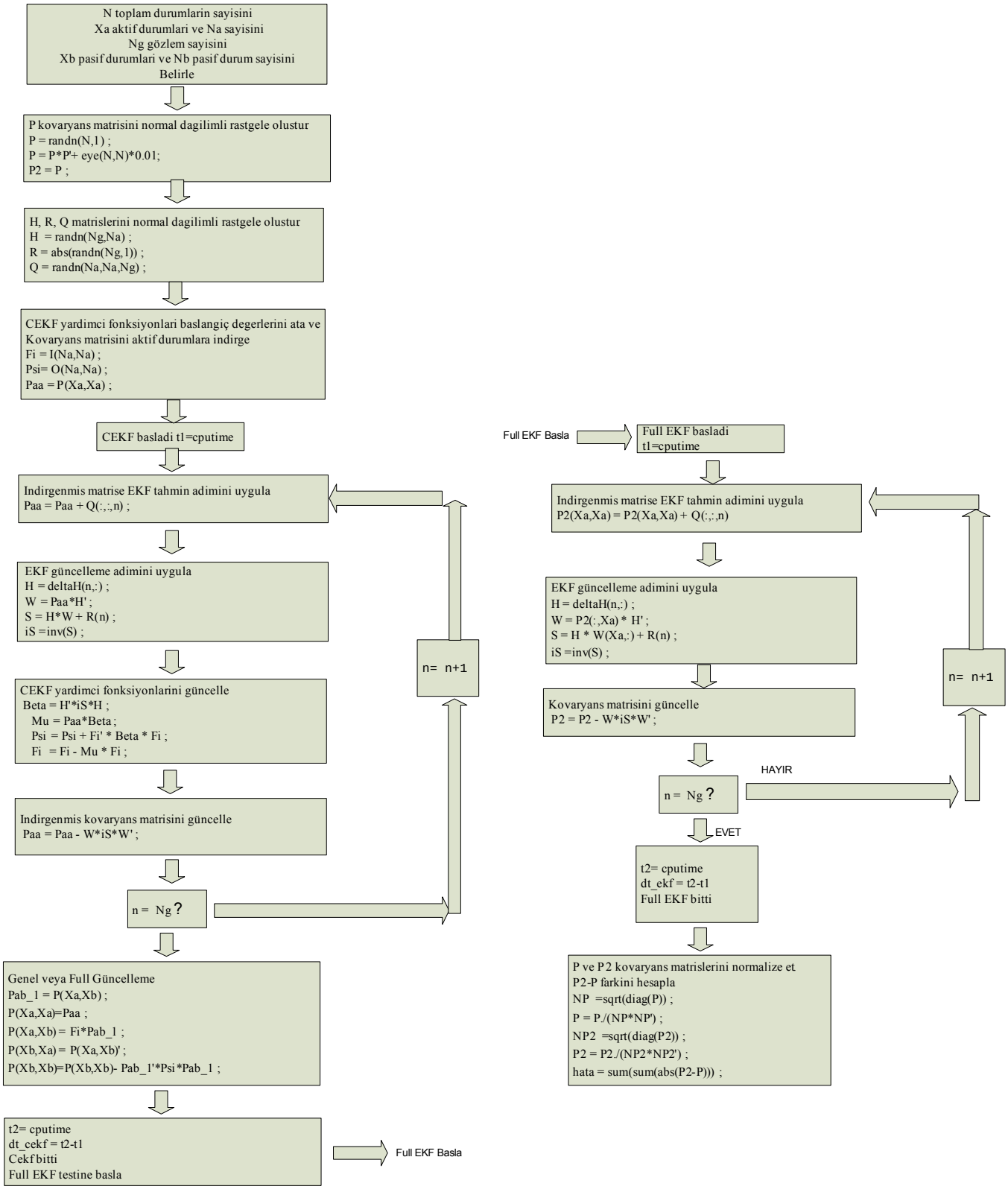
$$\nabla G_z = \frac{\partial g}{\partial z} = \begin{bmatrix} \cos(\theta + \theta_v) & -r.\sin(\theta + \theta_v) \\ \sin(\theta + \theta_v) & r.\cos(\theta + \theta_v) \end{bmatrix} \quad \textbf{(7.34)}$$

### 7.4 SGKF Uygulaması – Program1

SGKF ile GKF filtreleri arasındaki hata oranları ve hesaplama yoğunluğu dereceleri arasındaki farkın ne kadar olduğunu belirlemek için basit bir Matlab uygulaması yapılmıştır. Matlab uygulama kodu çalıştırıldığında aşağıdaki sonuçlar elde edilmiştir. Başlangıçta toplam durum sayısı 200, gözlem sayısı 100 ve aktif durum sayısı da 20 olarak alınmıştır. Matlab uygulama dosyasındaki durum sayıları 50'den başlayarak 1000'e kadar çıkarıldığı durumlarda ve aktif durum sayısı 20 olduğunda SGKF'nin kovaryans kestiriminin GKF kovaryans kestirimine ne kadar yaklaştığı ve kestirim algoritmalarının işlemcide ne kadar zaman harcadıkları yani hesaplama yoğunlukları karşılaştırılmıştır. Sonuçlar bölüm 7.6'daki grafiklerde gösterilmiştir. Hesaplama süresi uygulamanın koştuğu bilgisayarın işlemci zamanından alındığı için farklı bilgisayarlarda veya herhangi bir işlemcide farklılıklar gösterebilir. Fakat bu sonuçlara yakın sonuçlar alınacaktır. Program her durum sayısı için ortalama 10 sefer koşturulmuştur. Algoritmanın çalışma zamanı sonucunda harcanan işlemci zamanının ve SGKF kovaryans kestiriminin GKF kovaryans kestirimine göre

göreceli hatasının bu 10 tekrar için ortalaması alınmıştır. Durum sayısı 50, 100, 200, 300, 400, 500, 600, 700, 800, 900 ve 1000 olarak toplam 11 farklı durum sayısı için program toplamda 110 sefer koşturulmuş olmaktadır. Aynı işlemler ayrı ayrı aktif durum sayısı 40 ve gözlem sayısı 200 yapılarak iki defa daha tekrarlanmıştır. Toplamda program 330 defa koşturularak daha kararlı sonuçlar elde edilmeye çalışılmıştır. Matlab uygulama kodunun program akış şeması şekil 7.4'te gösterilmiştir. Bu akış şemasına göre önce SGKF çalıştırılıyor sonra da tam GKF çalıştırılıyor. Matlab çıktısı aşağıdaki gibi alınmış ve sonuçlar excel dosyasına aktarılarak bölüm 7.6'daki grafikler elde edilmiştir.

```
--------------------------------

Toplam durum sayisi  =[200]

Aktif durum sayisi   =[20]

Gözlem sayisi        =[100]

SGKF basladi

SGKF bitti           , dt=[20]ms

Tam GKF basladi

Tam GKF bitti        , dt=[180]ms

her iki normalize edilmis kovaryans matrisleri arasindaki hata toplami

[hata = sum(sum(abs(P_SGKF-P_tam)))].


SGKF_hatasi,[0.00000000001285]

SGKF_hatasi,[1.285305e-011]

--------------------------------
```



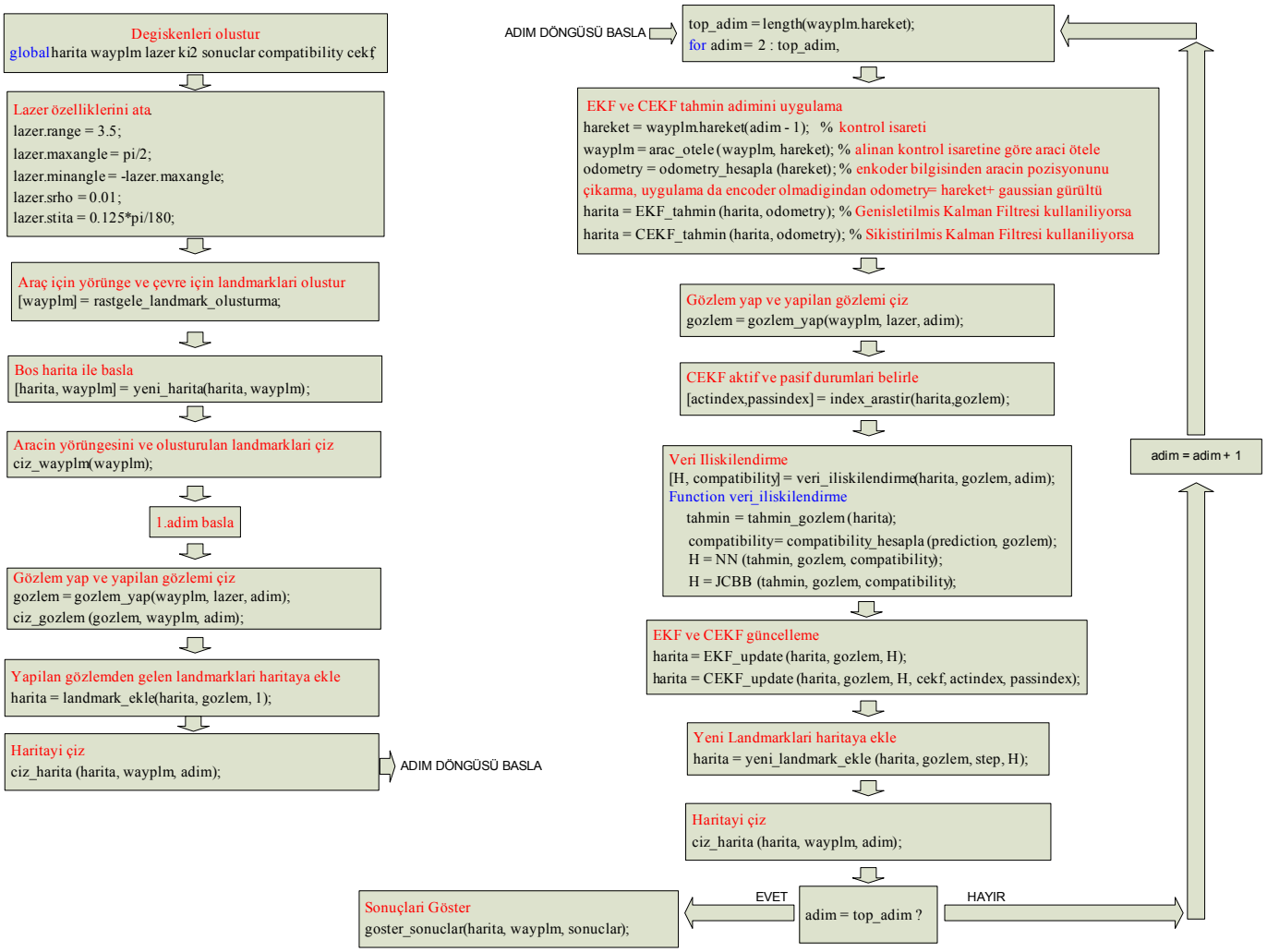

**Şekil 7.4:** SGKF ve GKF Hata ve Hesaplama Yoğunluğu Test Programı Akış Şeması

### 7.5 GKF ve SGKF Tabanlı SLAM Uygulaması – Program 2

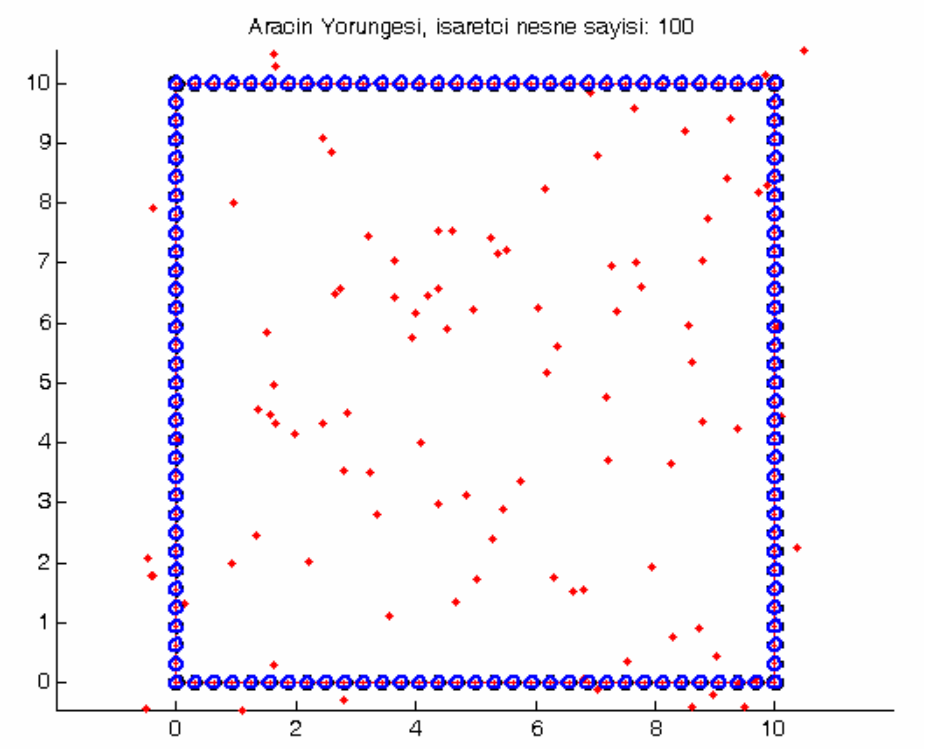
GKF ve SGKF'nin veri ilişkilendirme algoritmaları da kullanılarak incelenmesi SLAM'nin gerçek zamanlı uygulamaları için önemlidir. Çünkü gerçek uygulamalarda filtreler veri ilişkilendirme olmadan doğru harita kestirimi yapamazlar. Bu amaçla ikinci uygulama her iki filtre için Bireysel Uyumlu Yakın Komşuluk (BUYK) ve Bileşik Uyumlu Dallanma ve Bağlanma (BUDB) veri ilişkilendirme algoritmaları kullanılarak çalıştırılmıştır. Matlab uygulama kodu için akış diyagramı şekil 7.5'te verilmiştir.

Uygulama için varsayımlar;

- Diferansiyel araç karesel olarak bir yörünge izleyecek ve başladığı yere geri dönecektir. Karenin her bir kenarını 33 adımda tamamlayacak ve köşelerden kendi etrafında dönme hareketiyle 9 adımda dönecektir. Aracın izleyeceği toplam adım sayısı 168 adım olacaktır. Karenin her bir kenarı 10 metredir.
- Değişken sayıda (50, 100, 200, 300, 400, 500) işaretçi nesne rasgele etrafa yerleştirilmiştir.
- Sadece araç hareketlidir. İşaretçi nesnelerin sabit oldukları varsayılmıştır.
- Lazerin maksimum ölçme mesafesi 3.5 metre, taradığı açı 180 derece, mesafe ölçüm standart sapması 0.01 ve açı ölçüm standart sapması 0.125 olarak alınmıştır. Böylece i.işaretçi nesne için hata kovaryans matrisi $R_i = \begin{bmatrix} (r_i \times 0.01)^2 & 0 \\ 0 & (\theta_i \times 0.125)^2 \end{bmatrix}$ olarak hesaplanmıştır.

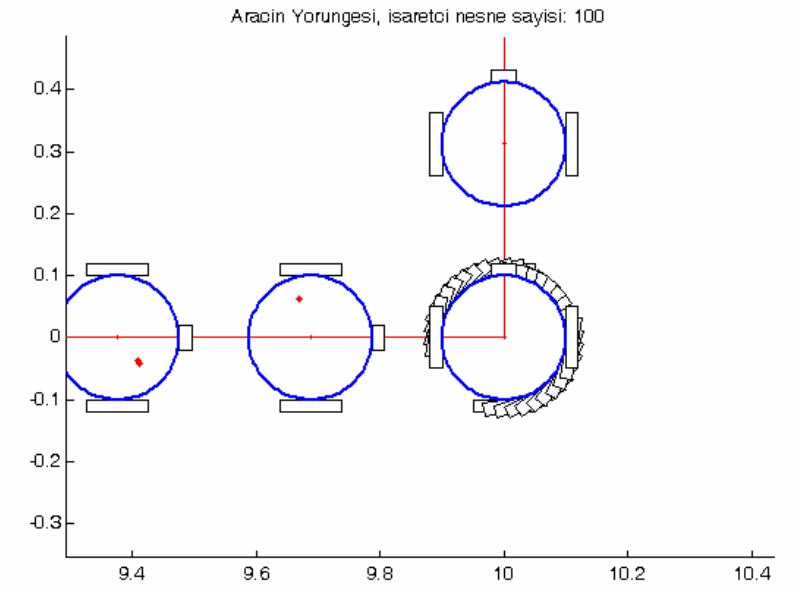



**Şekil 7.5:** Uygulama 2 için Matlab Programı Akış Şeması




**Şekil 7.6:** Bir kenarı 10 m olan karesel bir yörüngede ilerleyen araç ve karesel yörünge içine rasgele serpiştirilmiş işaretçi nesneler

Diferansiyel aracımız 10m'lik karesel bir yörüngeyi 168 adımda tamamlıyor (Şekil7.6). 33 adım karenin kenarlarını, 9 adım ise karenin köşelerini tamamlamak için kullanılıyor (Şekil 7.7). İşaretçi nesne sayısı ise rasgele olarak değişik sayılarda atanıyor. Şekil 7.6'da işaretçi nesne sayısı 100 olarak alınmıştır.



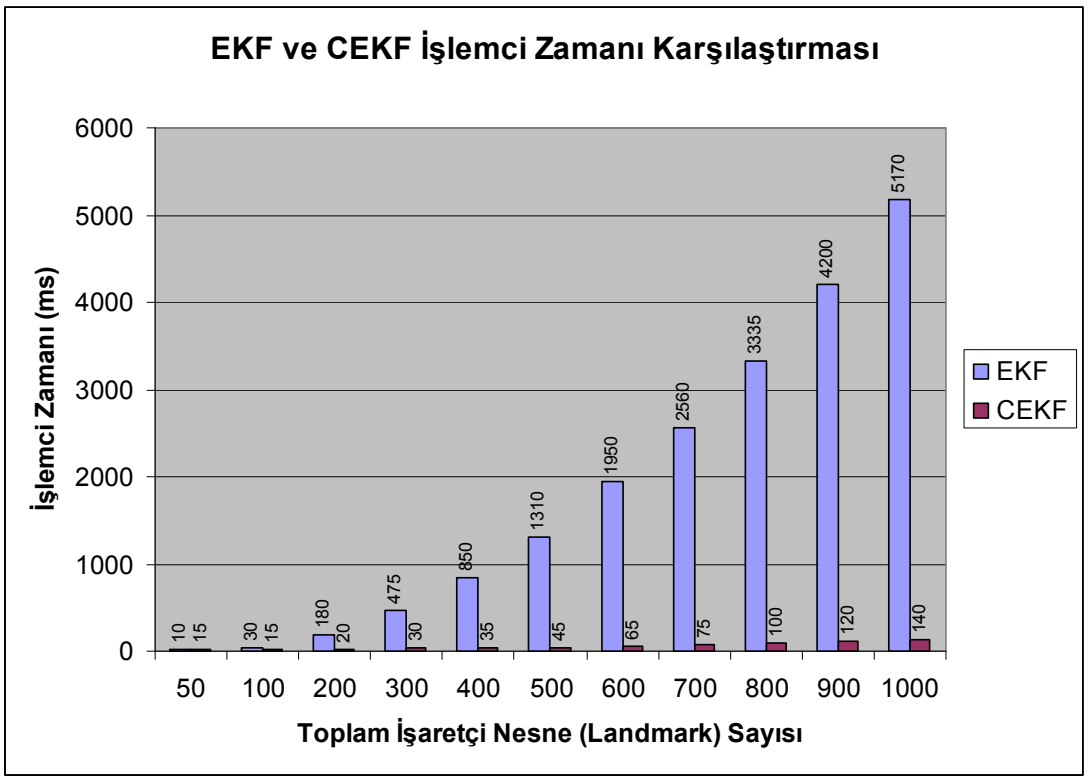

**Şekil 7.7:** Araç için Köşe Dönme Senaryosu

## 7.6 Uygulama Programı 1 Simülasyon Sonuçları

SGKF ve GKF kestirimleri arasındaki hesaplama yoğunluğu ve SGKF kovaryans kestiriminin GKF kovaryans kestiriminden ne kadar hatalı olduğunu görmek için hazırlanan uygulama programı 1 için elde edilen sonuç grafikleri Şekil 7.8 ve Şekil 7.13 arasındaki şekillerde gösterilmiştir. GKF'nin işlemci zamanlarını belirleyen ana unsur haritadaki toplam işaretçi nesne sayısı iken SGKF'nin işlemci zamanlarını belirleyen ana unsur ise lokal bölgedeki aktif durum sayısıdır. Aktif durum sayısı aracın o anda gördüğü veya gezindiği lokal alanda bulunan işaretçi nesneleri ifade etmektedir. Araç toplam 100 adet gözlem yaparak GKF ve SGKF kestirimlerini güncellemektedir. Yani kovaryans matrisi GKF ve SGKF öngörü ve güncelleme adımlarının 100 defa döngüye girmesi neticesinde güncellenmektedir. Aktif durum sayısı 20'den 40'a çıkarıldığında işlemci zamanı ve SGKF kovaryans matrisinin GKF kovaryans matrisine göre göreceli hatası şekil 7.10 ve 7.11'de gösterilmiştir. Buradaki hata hesabı SGKF'nin kestirdiği normalize edilmiş kovaryans matrisinin tam GKF'nin kestirdiği normalize edilmiş kovaryans matrisinden ne kadar saptığının ifadesidir.

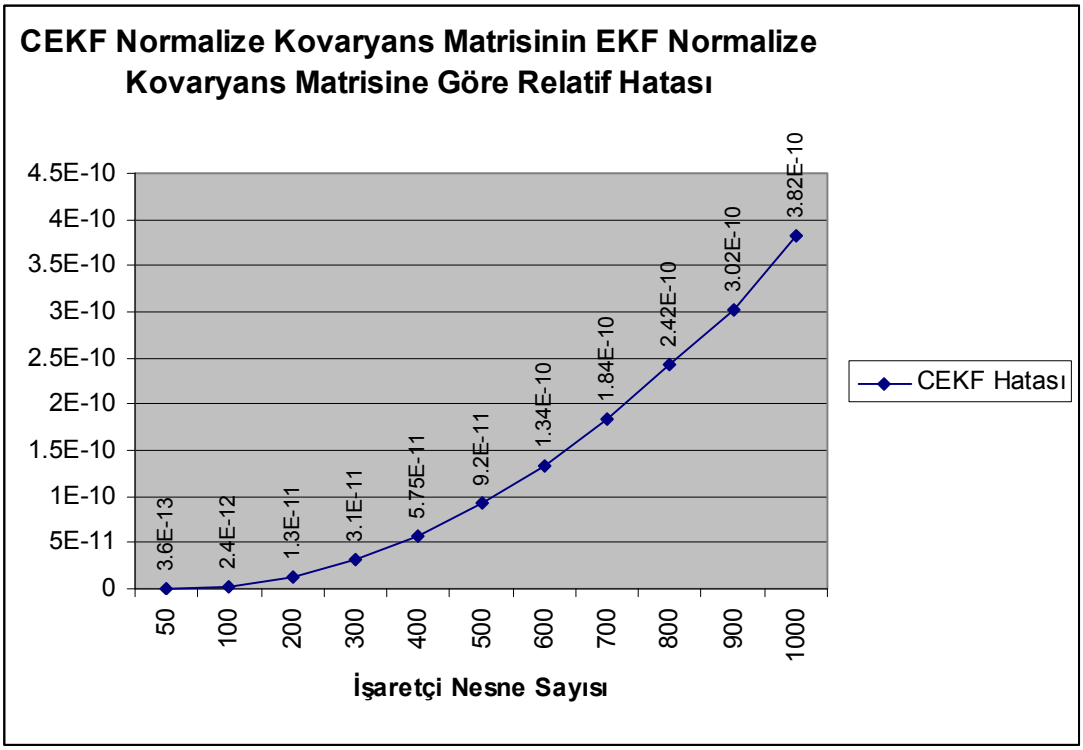
hata = sum(sum(abs(P_SGKF-P_tamGKF))) **(7.35)**

Aracın yaptığı gözlem sayısı 100 yerine 200 yapıldığında yani öngörü ve güncelleme adımları 200 defa tekrarlandığında elde edilen işlemci zamanları ve SGKF kovaryans matrisinin göreceli hatası şekil 7.12 ve 7.13'de gösterilmiştir.




**Şekil 7.8:** GKF (EKF) ve SGKF (CEKF) işlemci zamanı karşılaştırması

Şekil 7.8 aktif işaretçi sayısı 20 yapıldığında, elde edilen 100 gözlem sonucunda GKF ve SGKF işlemci zamanlarını göstermektedir. Şekildende anlaşıldığı gibi işaretçi nesne sayısı az olduğu (50,100) durumlarda GKF ve SGKF zamanları neredeyse birbirine yakın çıkmaktadır. Toplam 100 adet işaretçi sayısı ile kestirim yapıldığında GKF 30 ms, SGKF ise 15ms gibi bir sürede kestirimi tamamlamaktadır. Fakat işaretçi nesne sayısı artırıldığında GKF ve SGKF işleme zamanları arasındaki fark giderek büyümektedir. İşaretçi nesne sayısı 200 olduğu durumda GKF SGKF'ye göre yaklaşık 15 kat fazla bir sürede kestirim işlemini tamamlamaktadır. İşaretçi nesne sayısını 1000 olarak düşündüğümüzde, GKF kestirimi 5170 ms'de tamamlarken SGKF kestirimi 140 ms'de tamamlamaktadır ki gerçek uygulamalarda eğer bütün nesneleri haritaya dahil edecek olursak bu sayıya rahatlıkla ulaşılabileceği gözönüne alınırsa, GKF'nin SGKF'ye göre çok yavaş kaldığını söyleyebiliriz. Böyle bir durum gerçek zamanlı kestirim algoritmalarını GKF ile çalışamıyacağını göstermektedir. EKF kestiriminin işaretçi nesne sayısı maksimum 100 olduğu durumlarda iyi çalıştığı söylenebilir. O yüzden gerçek uygulamalarda ortamın önce fizibilitesi yapılarak işaretçi nesne sayıları tahmin edilmeli ve ona uygun kestirim algoritmaları üzerine yoğunlaşılmalıdır.

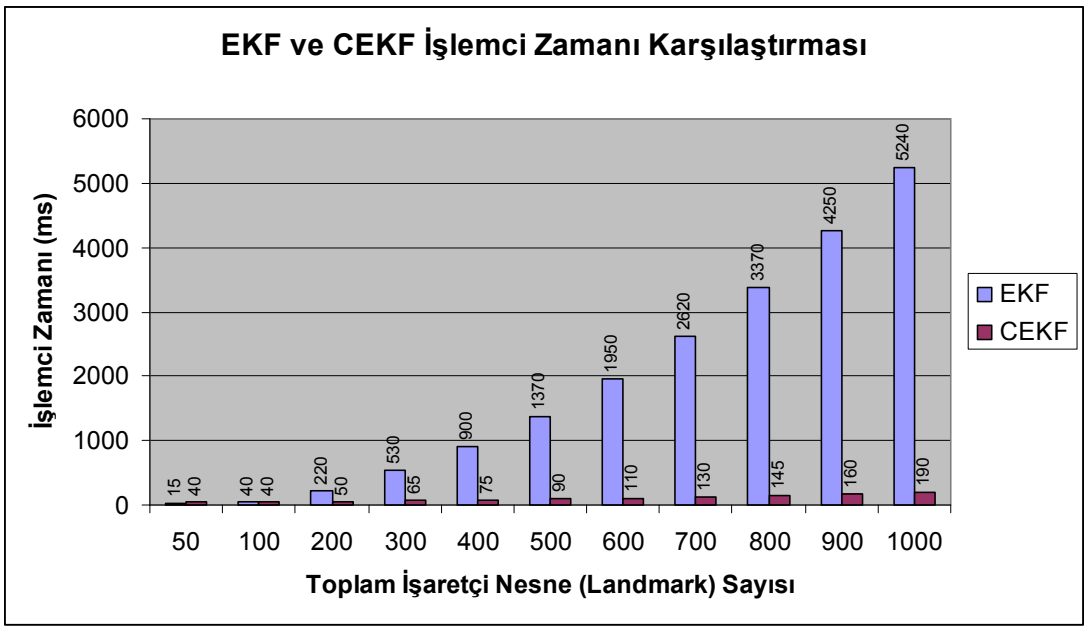



**Şekil 7.9:** SGKF (CEKF)'nin kestirdiği normalize edilmiş kovaryans matrisinin GKF (EKF)'nin kestirdiği normalize edilmiş kovaryans matrisine göre göreceli hatası

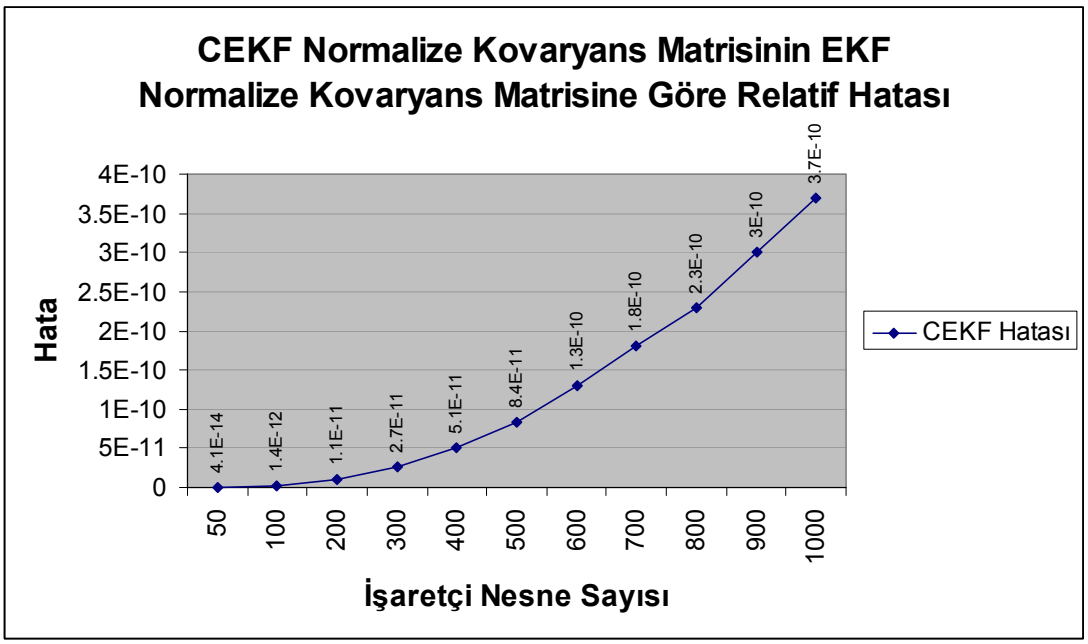
Şekil 7.9 aktif durum sayısı 20 ve gözlem sayısı 100 olmak üzere toplam işaretçi nesne sayısı arttırıldığında SGKF kovaryans matrisinin GKF kovaryans matrisine göre ne kadarlık bir sapma ile kestirimi tamamladığını göstermektedir. Bu hata işaretçi nesne sayısı 100 olduğu durumda $2.1e^{-12}$ iken işaretçi nesne sayısı 1000 yapıldığı durumda $3.82e^{-10}$ gibi bir değere artmaktadır. Hata artmış görünse de şekil 7.8'deki sonuçlar baz alındığında on milyarda 4 oranındaki kovaryans hatası harita kestirimleri için önemli değildir ve kabul edilebilir seviyelerdedir.

Şekil 7.10'da aktif işaretçi sayısı 20'den 40'a çıkarıldığı durumda toplam işaretçi sayısı artarken GKF ve SGKF kestirimleri için harcanan işlemci zamanları değişimleri görülmektedir. Şekil 7.10 Şekil 7.8'e benzer bir özellik göstermektedir. İşaretçi nesne sayısı artarken GKF ve SGKF'nin işlemci zamanlarıda artmakta fakat SGKF'nin işlemci zamanı çok düşük bir şekilde artarken GKF'nin işlemci zamanı neredeyse üstsel bir biçimde artmaktadır. GKF kestirim zamanı işaretçi nesne sayısı 200'den sonra kopmaktadır. Şekil 7.8 ile karşılaştırdığımızda aktif işaretçi sayısının artması GKF kestirim zamanını neredeyse hiç artırmamakta, SGKF kestirim zamanını ise ortalama 30-40 ms civarında artırmaktadır. Bunun sebebi ise GKF'nin

harcadığı işlemci zamanı toplam işaretçi nesne sayısıyla orantılı bir şekilde artarken, SGKF'nin harcadığı işlemci zamanı ise aktif durum sayısıyla orantılı şekilde artmaktadır. Burada aktif durum sayısı değiştirildiği için GKF işlemci zaman grafiği şekil 7.8'deki ile hemen hemen aynı kalırken, SGKF işlemci zaman grafiği ise şekil 7.8'dekine göre yaklaşık 30-40 ms civarında artış göstermiştir.

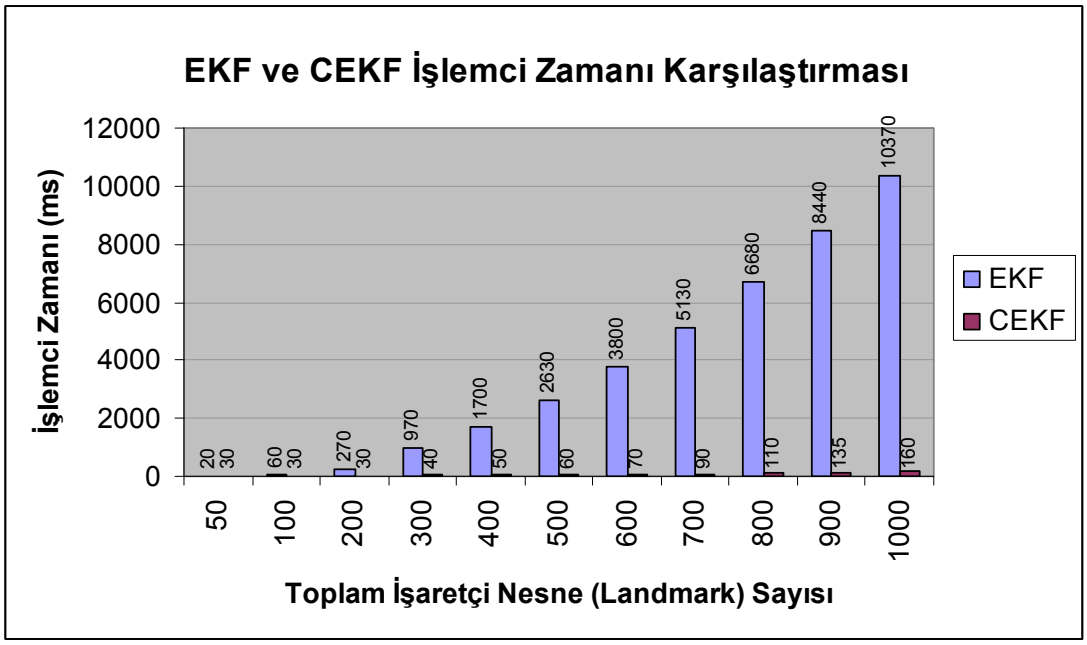



**Şekil 7.10:** Aktif işaretçi nesne sayısı 40 yapıldığında GKF (EKF) ve SGKF (CEKF) işlemci zamanı karşılaştırılması




**Şekil 7.11:** Aktif işaretçi nesne sayısı 40 olduğunda, SGKF (CEKF)'nin kestirdiği normalize edilmiş kovaryans matrisinin GKF (EKF)'nin kestirdiği normalize edilmiş kovaryans matrisine göre göreceli hatası

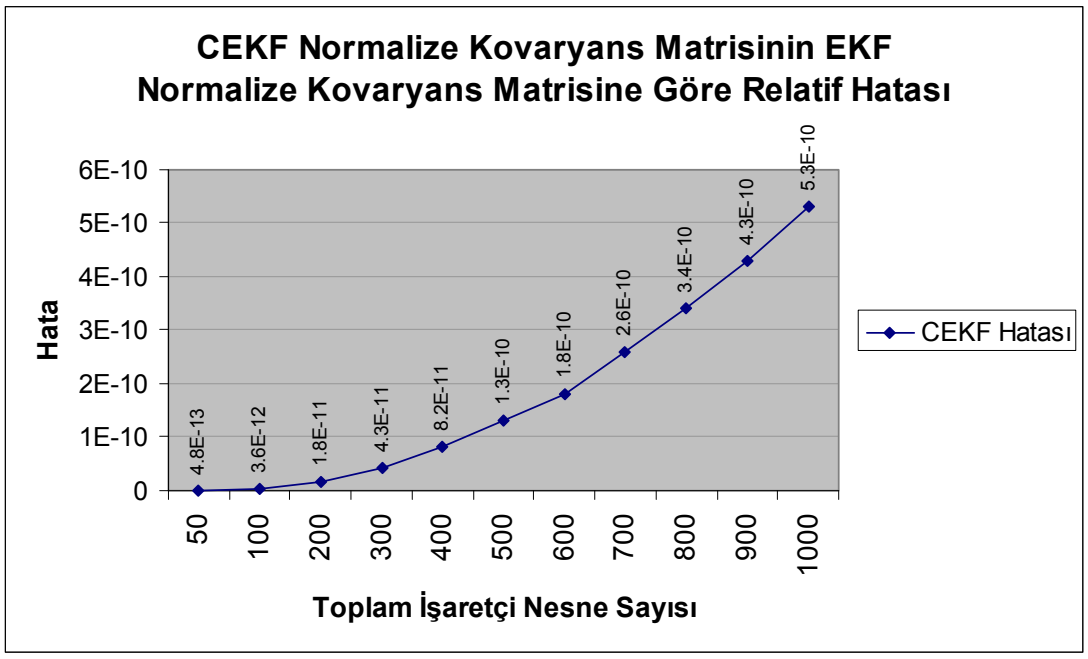
Şekil 7.11'de aktif durum sayısı 40 yapıldığı durumda toplam işaretçi sayısı artarken SGKF'nin kestirdiği kovaryans matrisinin GKF'nin kestirdiği kovaryans matrisine göre göreceli hatasının değişimi görülmektedir. Şekil 7.9'da verilen 20 aktif durum sayısına göre hatada çok fazla bir değişim söz konusu değil. Yani aktif işaretçi nesne sayısının artması kovaryans matrisinin göreceli hatasını etkilememektedir.




**Şekil 7.12:** Gözlem sayısı 200 yapıldığında GKF (EKF) ve SGKF (CEKF) işlemci zamanı karşılaştırılması

Şekil 7.12'de gözlem sayısı yani öngörü ve güncelleme adımlarının ardışıl tekrarlanma sayısı 100'den 200'e çıkarıldığı, aktif durum sayısının 20 kabul edildiği durumda (Şekil 7.8) toplam işaretçi nesne sayısı artarken GKF ve SGKF kestirimleri için işlemcide harcanan zamanın değişimi görülmektedir. Şekil 7.8 ve 7.12 karşılaştırıldığında gözlem sayısının artması GKF kestirimi için harcanan zamanı gözlem sayısı 100 olan duruma göre 2 kat artırmasına rağmen SGKF kestirimi için harcanan zamanı 20 ms gibi çok küçük bir değer kadar artırmıştır. İşaretçi sayısının 1000 olduğu durumda, gözlem sayısı 100 iken GKF kestirimi için harcanan zaman 5200 ms civarında iken gözlem sayısı 200 yapıldığında 10300 ms civarına çıkmaktadır ki bu da işlem zamanının iki kat artması demektir ki buda gözlem sayısı ve işaretçi nesne sayısının fazla olduğu uygulamalarda GKF tabanlı SLAM'ın gerçek zamanlı olarak kullanılamayacağını gösterir.

Şekil 7.13'de gözlem sayısının 200'e çıkarıldığı, aktif durum sayısının 20 kabul edildiği durumda toplam işaretçi nesne sayısı artarken SGKF kovaryans kestiriminin GKF kovaryans kestirimine göre hata değişimi görülmektedir. Şekil 7.9 ve 7.11 ile karşılaştırıldığında gözlem sayısının artması hata oranını çok fazla değiştirmemiştir.




**Şekil 7.13:** Gözlem sayısı 200 yapıldığında, SGKF (CEKF)'nin kestirdiği normalize edilmiş kovaryans matrisinin GKF (EKF)'nin kestirdiği normalize edilmiş kovaryans matrisine göre göreceli hatası

Yukarıdaki sonuçlara bakıldığında işaretçi nesne sayısının ve gözlem sayısının artması SGKF kovaryans kestirim hatasını GKF kovaryans kestirimine göre değiştirmemektedir. GKF'nin harcadığı işlemci zamanı ise işaretçi nesne sayısı artarken üssel şekilde arttığı gibi ve gözlem sayısı arttıkça da artan oran kadar işlemci zamanı artmaktadır. Dolayısıyla SGKF tabanlı kestirim algoritmaları bu açıdan gerçek zamanlı uygulamalar için özellikle büyük alanlarda tercih edilmektedir.

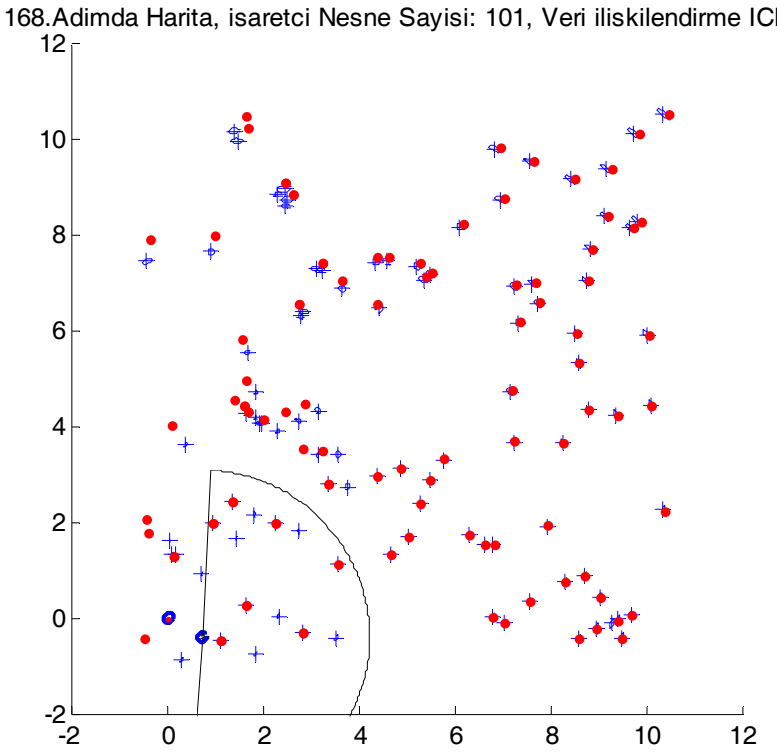
### 7.7 Uygulama Programı 2 Simülasyon Sonuçları

İkinci uygulama GKF (EKF) ve SGKF (CEKF) tabanlı SLAM uygulamasıdır. Bu uygulamada BUYK (ICNN) ve BUDB (JCBB) veri ilişkilendirme algoritmları kullanılmışlardır. Uygulamadan elde edilen simulasyon sonuçları Şekil 7.14 ile 7.33 arasındaki şekillerde gösterilmiştir. Şekil 7.14'de gösterilen kırmızı noktalar ortamdaki işaretçi nesnelerin gerçek pozisyonlarını, mavi noktalar ise kestirilen

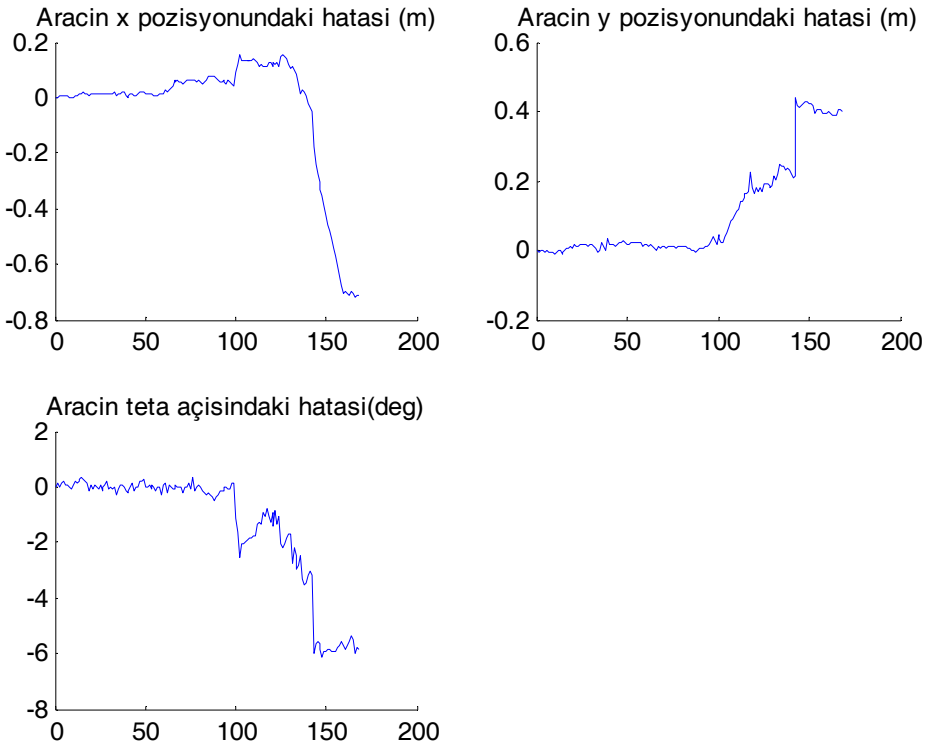
pozisyonlarını ifade etmektedir. Siyah yarım çember lazer sensörün ölçme aralığını göstermektedir. Lazer 180° açıyı tarayarak 3.5 metrelik mesafeyi ölçmektedir. Aracımız ise yarım dairenin merkezinde bulunan çift tekerlekli bir diferansiyel araçtır.

Şekil 7.14'de diferansiyel aracın 10 m'lik karesel yörüngeyi tamamladığı 168. adım sonrasında GKF (EKF) tabanlı SLAM metodu ile BUYK (ICNN) veri ilişkilendirme algoritması kullanarak kestirdiği harita ile doğru harita arasındaki farklar görülmektedir. İlk adımlarda, karesel yörüngenin sağ tarafında kalan nesnelerin kestirildiği adımlar, nesneler doğru pozisyonlarında kestirilmiş fakat son adımlara doğru aracın pozisyonundaki belirsizliğin artması ve aracın pozisyonun kestirimindeki hataların son adımlara doğru artması sonucunda haritadaki nesnelerin kestirimide gerçek pozisyonlarından sapmıştır. Aracın burada sürekli loop yapması durumunda aracın pozisyonundaki hata giderek artacağı için nesnelerin pozisyon kestirimlerindeki hata da giderek artacaktır ve sonuç olarakta haritalama tamamen yanlış bir şekilde ortaya çıkacaktır. Bu durum kapalı çevrim kararsızlığı yada kapalı çevrim tutarsızlığı olarak adlandırılır ki bu da aracın mesafe sayacından lokalizasyon hesabındaki belirsizlikler, gürültüler ve hassasiyetin yeterli seviye olmamasından kaynaklanmaktadır.

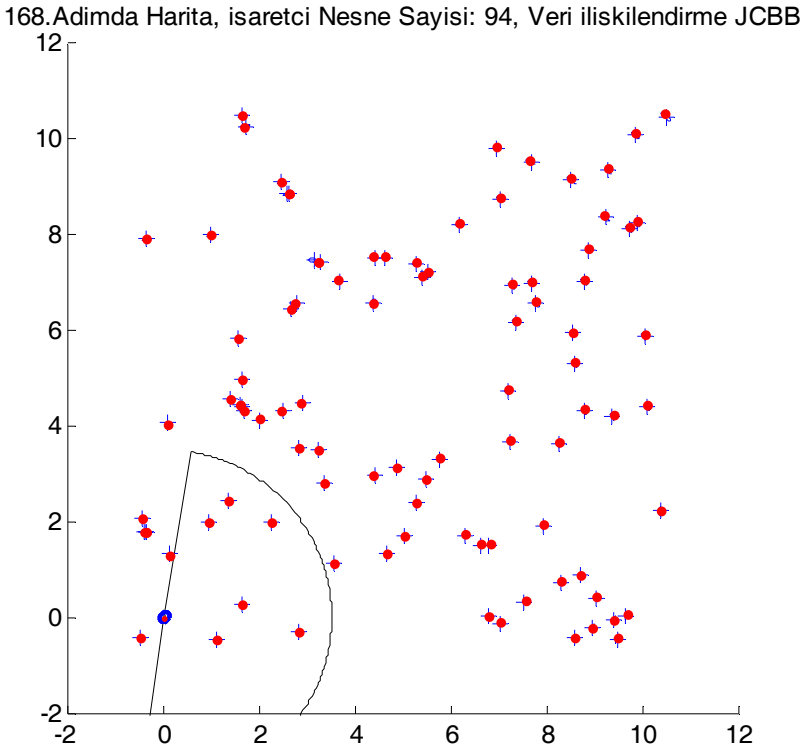
Şekil 7.15'te GKF tabanlı SLAM metodunun BUYK veri ilişkilendirme algoritması ile kullanılması durumundaki araç lokasyonundaki (x, y ve θ) hataları görülmektedir. Dikey sütun aracın x ve y yönündeki hatanın metre olarak, θ açısındaki hatanın da derece olarak ifadesini, yatay sütün ise aracın adım sayısını göstermektedir. Şekilden de anlaşılacağı gibi yukarıdaki sebeplerden dolayı ilk adımlarda araç hataları minimum seviyede iken son adımlara yaklaşıldığında hatalar giderek büyümektedir. 100.adımda 0.1 m civarında olan x eksenindeki hata son adımda -0.7'ye, ye eksenindeki hata 0.01'den 0.4'e, θ açısındaki hata ise -2 derecelerden -6 derecelere kadar büyümüştür. Kapalı çevrim devam ettikçe bu hatada artacağından dolayı araç istenilen konumdan oldukça sapacak ve hedefe ulaştığı düşünülen araç hedefin çok uzağında kalacaktır.




**Şekil 7.14:** BUYK (ICNN) veri ilişkilendirme algoritması kullanılarak GKF (EKF) harita kestirimi

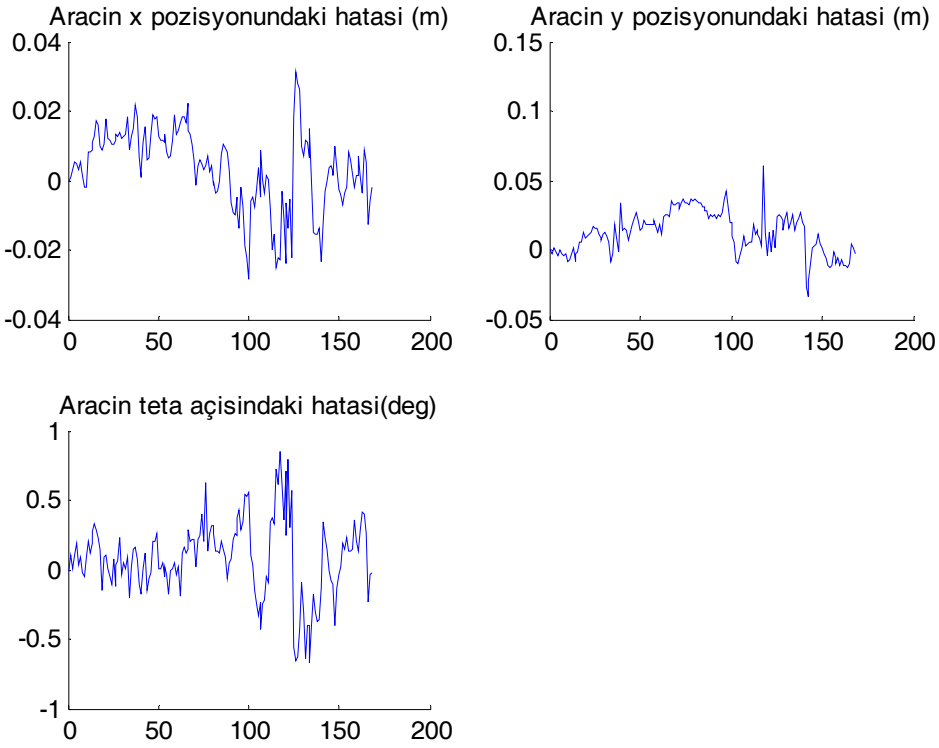



**Şekil 7.15:** BUYK (ICNN) veri ilişkilendirme algoritması kullanılarak GKF (EKF) araç pozisyonu kestirimindeki hatalar



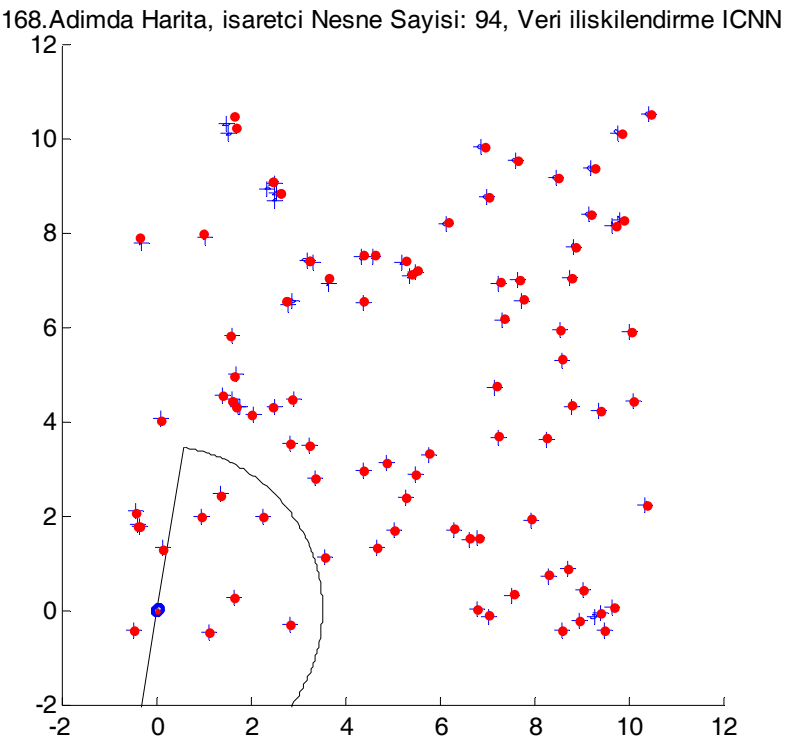

**Şekil 7.16:** BUDB Veri İlişkilendirme Algoritması Kullanılarak GKF Harita Kestirimi

Şekil 7.16'da GKF tabanlı SLAM'ın BUDB veri ilişkilendirme algoritması ile yaptığı harita kestirimini görülmektedir. Kırmızı renkle gösterilenler nesnelerin doğru pozisyonları, mavi renkle gösterilenler ise nesnelerin kestirilen pozisyonlarını ifade etmektedir. BUYK ile karşılaştırıldığında nesnelerin pozisyonları gerçeğe çok yakındır. Bunun sebebi de BUDB algoritmasının çok daha güçlü veri ilişkilendirme yaparak nesneleri doğru gözlemlerle eşleştirmeleri ve dolayısıyla araç pozisyonlarını doğru güncelleyerek gerçeğe yakın kestirebilmektedir. Bununla beraber işaretçi nesneleri doğru gözlemlerle eşlemesi ve aracın pozisyonunu gerçeğe yakın kestirilmesine yardımcı olmasından dolayıda işaretçi nesnelerin konumlarını gerçekten çok az bir sapmayla tesbit edip doğru haritalamanın yapılmasını sağlamaktadır.

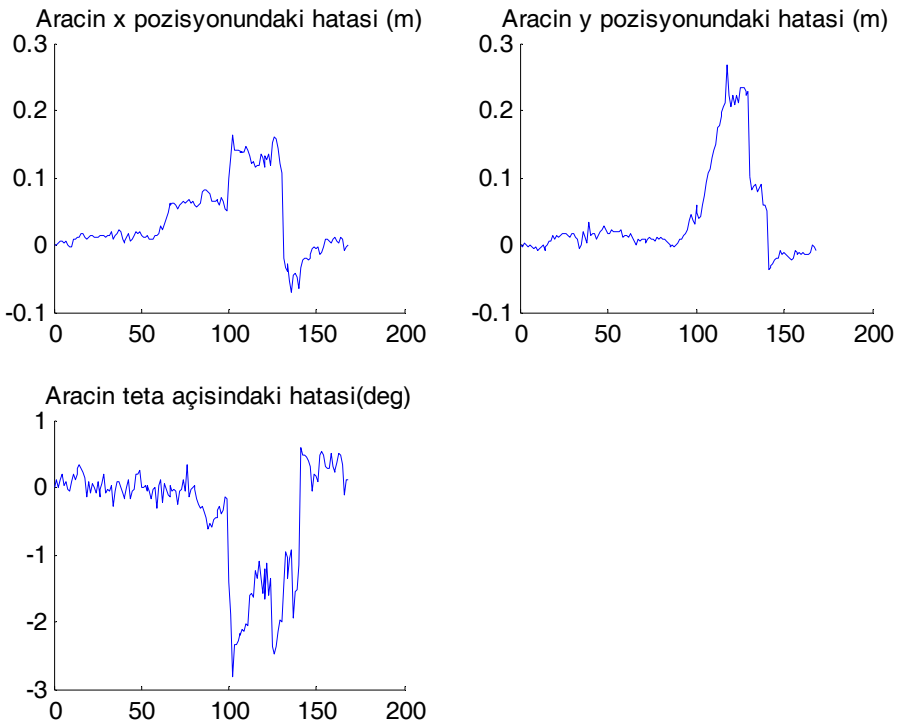



**Şekil 7.17:** BUDB (JCBB) veri ilişkilendirme algoritması kullanılarak GKF (EKF) araç pozisyonu kestirimindeki hatalar

Şekil 7.17'de GKF tabanlı SLAM'ın BUDB algoritması ile çalıştırıldığı durumda aracın her adımdaki lokasyon hataları görülmektedir. Dikey sütun aracın x ve y yönündeki hatanın metre olarak, θ açısındaki hatanın da derece olarak ifadesini, yatay sütün ise aracın adım sayısını göstermektedir. BUYK ile karşılaştırıldığında araç lokasyon hataları yok denecek kadar azdır. Lokasyondaki hatalar her adımda neredeyse sıfıra doğru yakınsamaya çalışmıştır. Aracın x ve y eksenindeki konum hatası -2 ile 2 cm arasında, yön hatası ise -0.5 ile 0.5 derece arasında kalmaktadır.



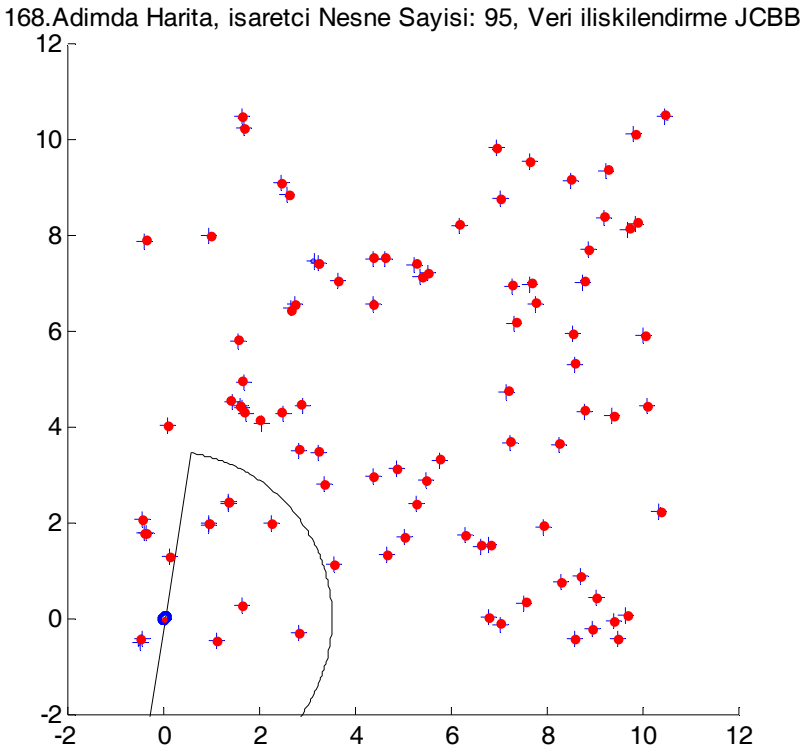

**Şekil 7.18:** BUYK (ICNN) veri ilişkilendirme algoritması kullanılarak SGKF (CEKF) harita kestirimi




**Şekil 7.19:** BUYK (ICNN) veri ilişkilendirme algoritması kullanılarak SGKF (CEKF) araç pozisyonu kestirimindeki hatalar

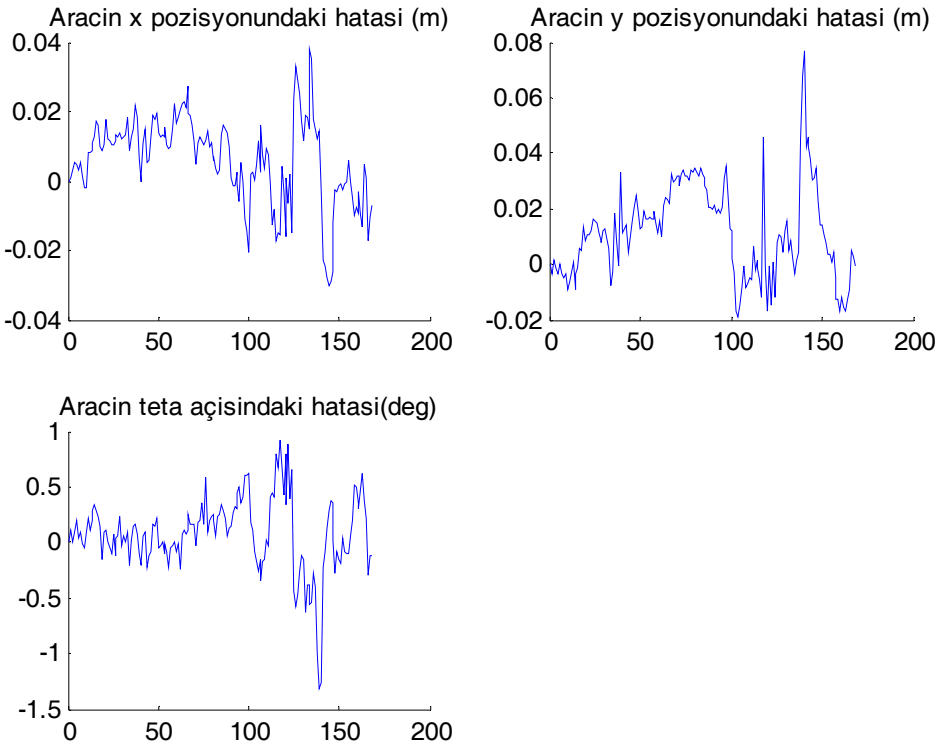
Şekil 7.18'de SGKF (CEKF) tabanlı SLAM'ın BUYK (ICNN) veri ilişkilendirme algoritması kullanarak kestirdiği harita görülmektedir. GKF tabanlı SLAM'ın BUYK veri ilişkilendirme algoritması kullanarak kestirdiği harita ile karşılaştırıldığında SGKF'nin daha başarılı olduğu söylenilebilir. Bunun bir sebebinin aracın pasif durumları dikkate almayarak aktif durumlara göre hareket etmesi olduğunu düşünebiliriz. O anda gördüğü nesnelere göre kendini konumlandırdığı için hatalar artmamış olduğu düşünülebilir.

Şekil 7.19'da SGKF tabanlı SLAM'ın BUYK algoritması ile çalıştırıldığı durumda aracın her adımdaki lokasyon hataları görülmektedir. Dikey sütun aracın x ve y yönündeki hatanın metre olarak, θ açısındaki hatanın da derece olarak ifadesini, yatay sütün ise aracın adım sayısını göstermektedir. Araç 100.adım ile 140. adım arasında yanlış pozisyonda kestirilse de son 30 adımda pozisyonunu düzeltmiş ve lokasyondaki hatalar neredeyse sıfıra doğru yakınsamaya çalışmıştır. GKF ile karşılaştırıldığında yanlış kestirilen adımlardaki hatalar bile çok makul düzeylerde kalmaktadır.



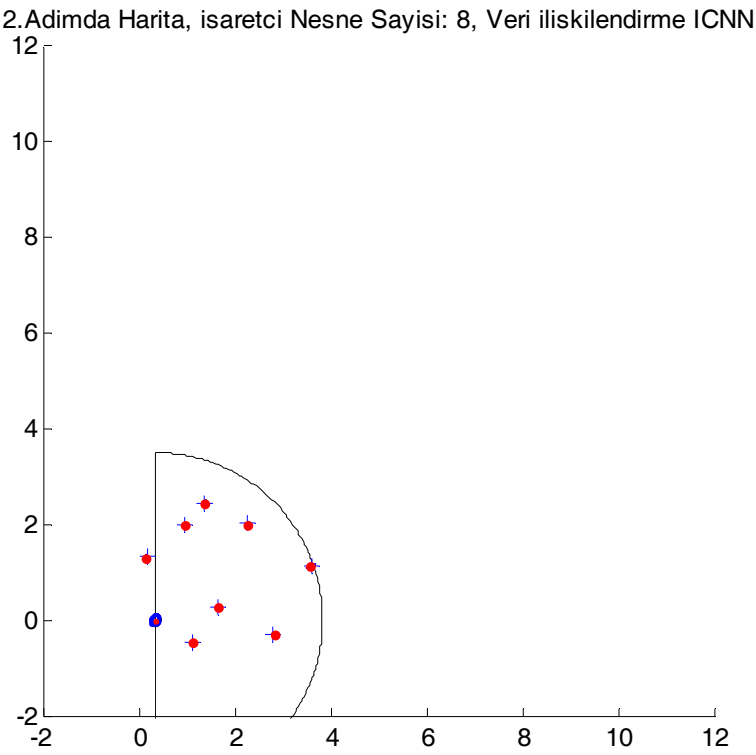

**Şekil 7.20:** BUDB (JCBB) veri ilişkilendirme algoritması kullanılarak SGKF (CEKF) harita kestirimi

Şekil 7.20'de SGKF (CEKF) tabanlı SLAM'ın BUDB (JCBB) veri ilişkilendirme algoritması kullanarak kestirdiği harita görülmektedir. SGKF tabanlı SLAM'ın BUYK veri ilişkilendirme algoritması kullanarak kestirdiği harita ile karşılaştırıldığında 100.adımla 140. adımlar arasındaki lokasyondan sapmalarda ortadan kalkmaktadır. Buda BUDB veri ilişkilendirme algoritmasının güçlü ve kararlı yapısından kaynaklanmaktadır. GKF-BUYK ile karşılaştırıldığında ise çok farklı bir fark olmadığı görülebilir.

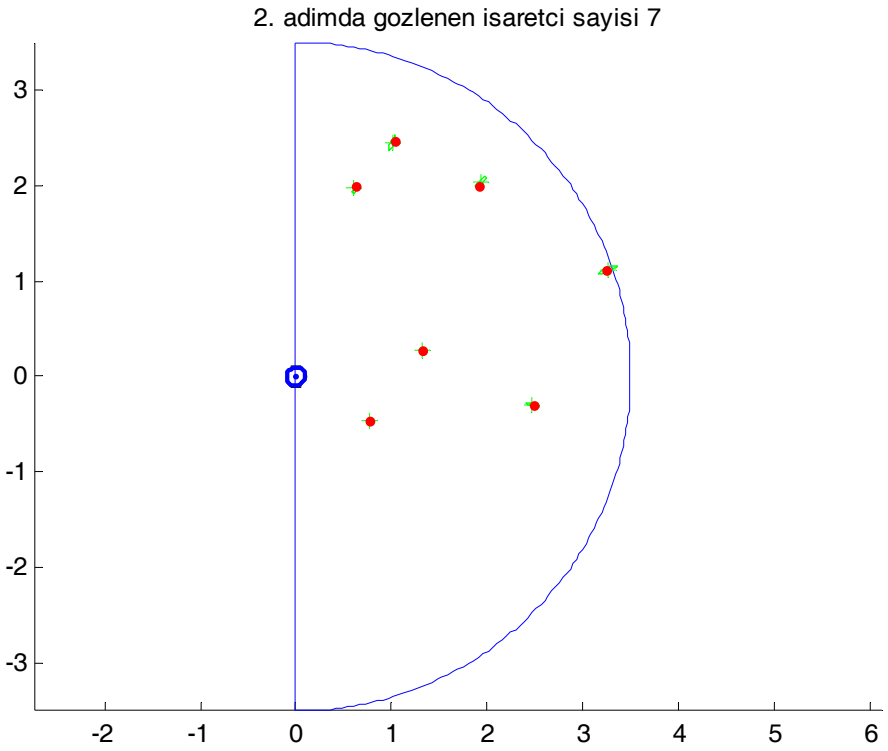



**Şekil 7.21:** BUDB (JCBB) veri ilişkilendirme algoritması kullanılarak SGKF (CEKF) araç pozisyonu kestirimindeki hatalar

Şekil 7.21'de SGKF tabanlı SLAM'ın BUDB algoritması ile çalıştırıldığı durumda aracın her adımdaki lokasyon hataları görülmektedir. Dikey sütun aracın x ve y yönündeki hatanın metre olarak, θ açısındaki hatanın da derece olarak ifadesini, yatay sütün ise aracın adım sayısını göstermektedir. SGKF-BUYK ile karşılaştırıldığında araç lokasyonunun orta adımlarındaki hataların kaybolduğunu görebiliriz. GKF-BUDB ile karşılaştırıldığında ise araç lokasyonu hataları arasında çok fazla bir fark olmadığını söylebiliriz.

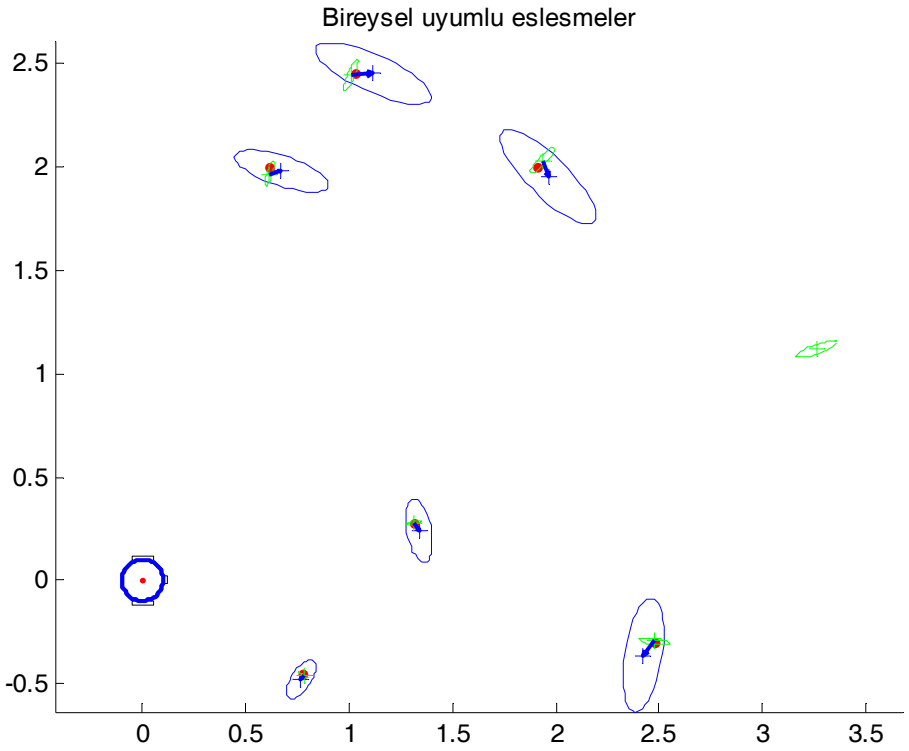



**Şekil 7.22:** 2.adımda harita kestirimi



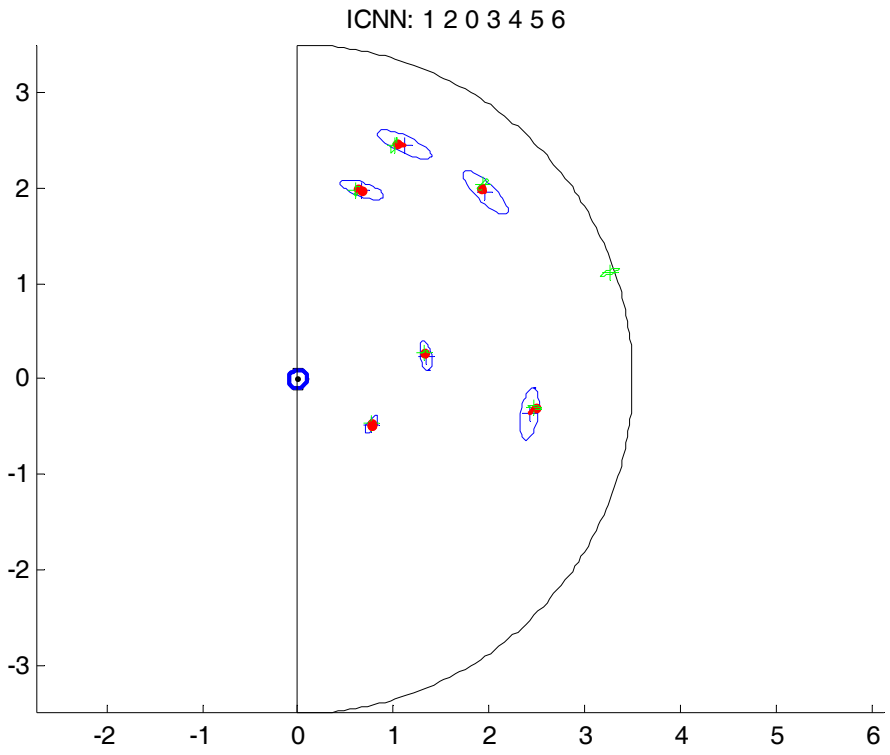

**Şekil 7.23:** 2.adımda gözlenen işaretçi sayısı

Şekil 7.22'de 2. adımda kestirilen durum vektörünün oluşturduğu harita ve araç pozisyonu, şekil 7.23'de ise 2.adım esnasında kestirilen ve o adımda görülen işaretçi nesneleri gösteren bir kesit görülmektedir. Sensörün algıladığı alandaki işaretçi nesne sayısı 7'dir ve daha önce haritaya atılmış ve sönsöründe algılama alanından çıkmış 1 adet işaretçi nesne vardır.



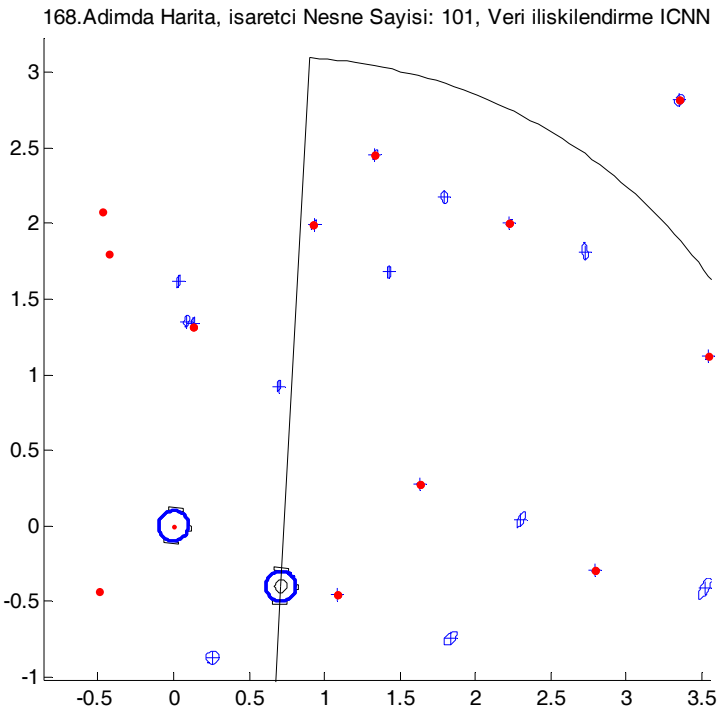

**Şekil 7.24**: 2.adımda yapılan gözlemlerle öngörülen durumların eşleşmesi

Şekil 7.24'te yapılan gözlemlerle haritaya daha önce atılmış işaretçi nesnelerin öngörülen durumlarının eşleşmelerini görmekteyiz. Yeşil renkli elipsler sensörden 2.adımda alınan ölçümleri, mavi renkli elipsler ise 1.adımda haritaya atılan işaretçi nesnelerinin 2.adımda öngörülen durumlarını göstermektedir. Elipsler ölçüm belirsizliklerinin ifadesidir. Araca yakın nesnelerin elipslerin küçük olmasının iki sebebi bulunmaktadır. Birincisi belirsizlik ölçülen mesafenin lazerin r bileşenine ait standart sapmayla çarpılmasıyla oluşturulmaktadır. Dolayısıyla araca yakın nesnelere ait belirsizlikler küçük olacaktır. İkinci ise aynı nesnenin daha çok gözlenmesi neticesinde gözlem modelindeki jacobian ifadelerinin daha çok güncellenmesinden dolayı kestirilen pozisyonlara ait elipsler gerçek pozisyonlarının merkezine doğru küçülecektir. Yani belirsizlikler giderek ortadan kalkacaktır.

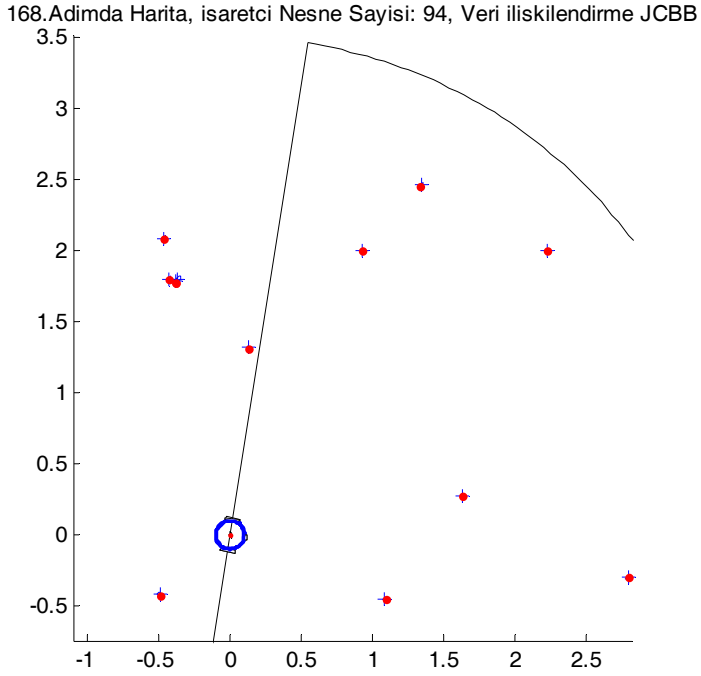



**Şekil 7.25:** 2.adımda BUYK (ICNN) için üretilen hipotez

Şekil 7.25'te 2.adımda BUYK (ICNN) tarafından üretilen hipotez görülmektedir. Bu hipotezin 0 olan elemanının anlamı gözlem veri vektörünün 3.sırasında gelen nesnenin daha önce görülmeyen yeni bir nesneye ait olduğudur. Siyah çemberin en ucundaki yeşil ölçümde bu yeni nesneye ait ölçümdür. Hipotezdeki diğer rakamlar o sıradaki gözlemlerin haritadaki hangi işaretçi nesneleri ile eşleştirildiğini göstermektedir. Bu hipotez sonucunda eşlenen durumlara ait öngörülen pozisyonlar bu gözlemlerle güncellenirken, hipotezde 0 olan elemanda yeni nesne olarak algılanıp durum vektörüne eklenerek haritada yer edinmekte ve durum vektörü büyümektedir. Bununla ilgili kovaryans matrisi büyültmeside yapılmaktadır.

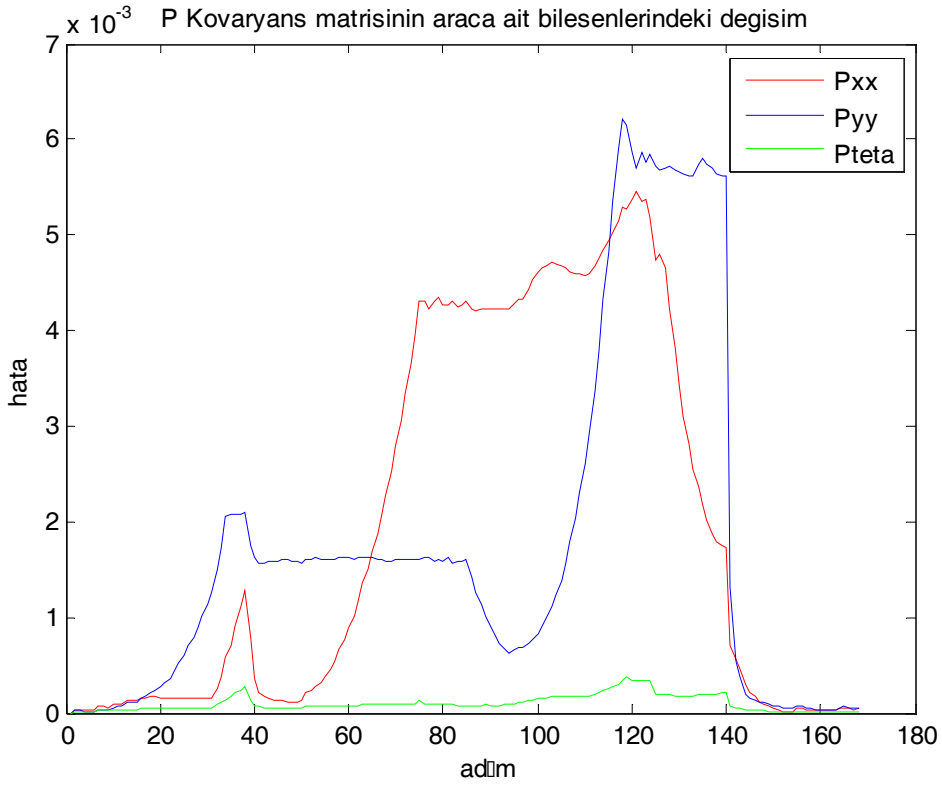



**Şekil 7.26:** BUYK (ICNN) algoritması kullanıldığında 168.adımda aracın pozisyonundaki hatadan dolayı işaretçilerde yanlış yere konuluyor.




**Şekil 7.27:** BUDB (JCBB) algoritması kullanıldığında 168.adımda aracın pozisyonundaki hata neredeyse sıfıra yakın olmasında dolayı işaretçiler haritada doğru yerde.

Şekil 7.26 ve 7.27'de BUYK (ICNN) ve BUDB (JCBB) algoritmalarının son adım tamamlandığındaki araç pozisyonları gösterilmiştir. Şekillerden de anlaşılacağı gibi BUDB algoritması ile araç pozisyon ve işaretçi nesne pozisyon kestirimleri gerçeğe çok yakın çıkmaktadır. BUYK veri ilişkilendirme algoritması ile araç ve işaretçi nesne pozisyonları gerçek pozisyonlarından oldukça sapmaktadır. Bu da BUDB algoritmasının ne kadar güçlü veri ilişkilendirme yaptığını göstermektedir.




**Şekil 7.28:** GKF ve SGKF kestirim yapıldığında kovaryans matrisinin araca ait bileşenleri (x bileşeni kırmızı, y bileşeni mavi, teta bileşeni θ)

Şekil 7.28'de GKF ve SGKF'nin BUYK ve BUDB algoritmaları sonucunda araca ait P kovaryans matrisinin bütün adımlardaki değerleri görülebilir. Şekilden de anlaşılacağı gibi kovaryans matrisi son adımlara doğru sıfıra yaklaşmaktadır. Bu da GKF ve SGKF'nin kararlılığını göstermektedir. Bu şekle göre GKF ve SGKF algoritmaları kararlı birer kestirim algoritmalarıdır diyebiliriz.

Tablo 7.1'de son adımda değişik sayıda (50, 100, 200, 300, 400 ve 500) işaretçi nesne sayısıyla GKF ve SGKF algoritmalarının BUYK ve BUDB veri ilişkilendirme algoritmaları ile kullanıldığında güncelleme, veri ilişkilendirme ve tüm harita kestirimi işlemleri için harcadıkları işlemci zamanları saniye olarak gösterilmiştir.

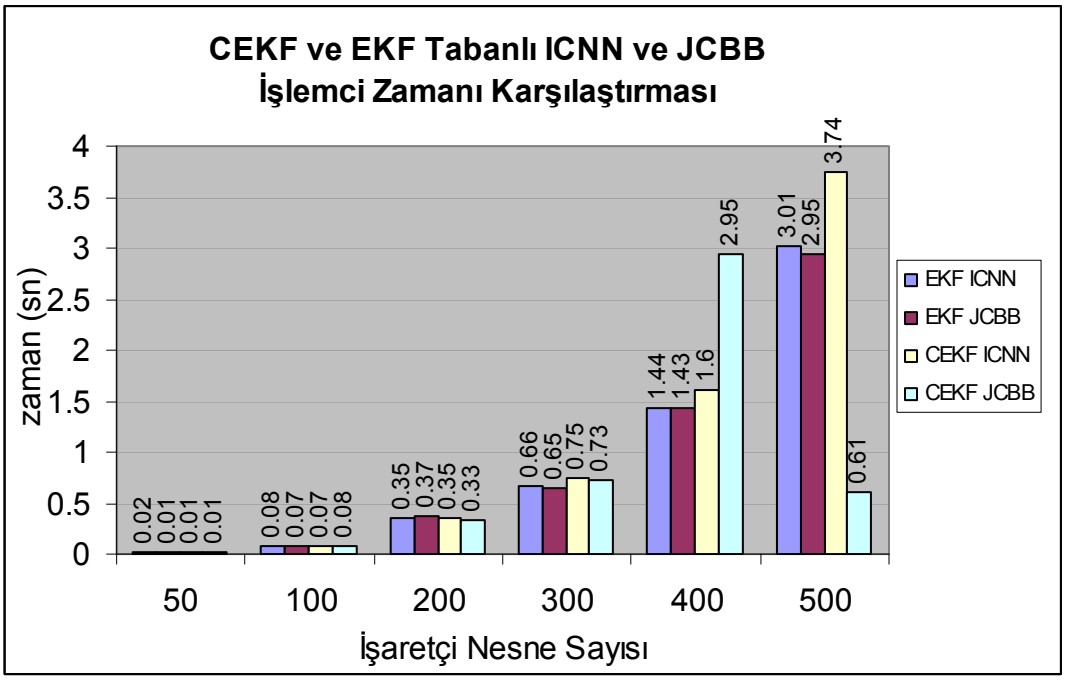
**Tablo 7.1:** Son Adımda Hesaplanan İşlemci Zamanları

| Son Adım | | Veri İlişkilendirme Algoritması | | | Tüm Harita Kestirimi | | Güncelleme İşlemi | |
|---|---|---|---|---|---|---|---|---|
| İşaretçi Nesne Sayısı | Gözlem Sayısı m | t(sn) | BUYK | BUDB | BUYK | BUDB | BUYK | BUDB |
| 50 | 1 | GKF | 0.02 | **0.01** | 6.93 | **7.24** | 0.01 | **0.01** |
| | | SGKF | 0.01 | 0.01 | 7 | 7.44 | 0.01 | 0.01 |
| 100 | 6 | GKF | 0.08 | 0.07 | 19.86 | 20.43 | 0.03 | 0.03 |
| | | SGKF | 0.07 | 0.08 | 19.72 | 20.45 | 0.02 | 0.03 |
| 200 | 13 | GKF | 0.35 | 0.37 | 81.45 | 85.06 | 0.17 | 0.18 |
| | | SGKF | 0.35 | 0.33 | 79.58 | 80.17 | 0.12 | 0.1 |
| 300 | 18 | GKF | 0.66 | 0.65 | 189.6 | 188.6 | 0.33 | 0.31 |
| | | SGKF | 0.75 | 0.73 | 193.3 | 190.8 | 0.26 | 0.24 |
| 400 | 31 | GKF | 1.44 | 1.43 | 356.6 | 409.4 | 0.65 | 0.59 |
| | | SGKF | 1.6 | 2.95 | 349.1 | 495.6 | 0.5 | 0.41 |
| 500 | 45 | GKF | 3.01 | 2.95 | 651 | 1099 | 1.62 | 1.49 |
| | | SGKF | 3.74 | 0.61 | 619.8 | 1022 | 1.59 | 0.79 |

**Tablo 7.2:** Bütün Adımlarda Hesaplanan İşlemci Zamanları Ortalaması

| Ortalama | Veri İlişkilendirme Algoritması | | | Güncelleme İşlemi | |
|---|---|---|---|---|---|
| İşaretçi Nesne Sayısı | t(sn) | BUYK | BUDB | BUYK | BUDB |
| 50 | GKF | **0.015** | **0.017** | **0.006** | **0.005** |
| | SGKF | 0.015 | 0.018 | 0.006 | 0.005 |
| 100 | GKF | 0.047 | 0.05 | 0.014 | 0.014 |
| | SGKF | 0.046 | 0.05 | 0.014 | 0.013 |
| 200 | GKF | 0.2 | 0.21 | 0.07 | 0.063 |
| | SGKF | 0.2 | 0.2 | 0.05 | 0.048 |
| 300 | GKF | 0.45 | 0.48 | 0.15 | 0.15 |
| | SGKF | 0.5 | 0.5 | 0.13 | 0.11 |
| 400 | GKF | 0.85 | 1.22 | 0.32 | 0.29 |
| | SGKF | 0.87 | 1.89 | 0.23 | 0.21 |
| 500 | GKF | 1.5 | 4.37 | 0.67 | 0.6 |
| | SGKF | 1.51 | 59 | 0.45 | 0.36 |

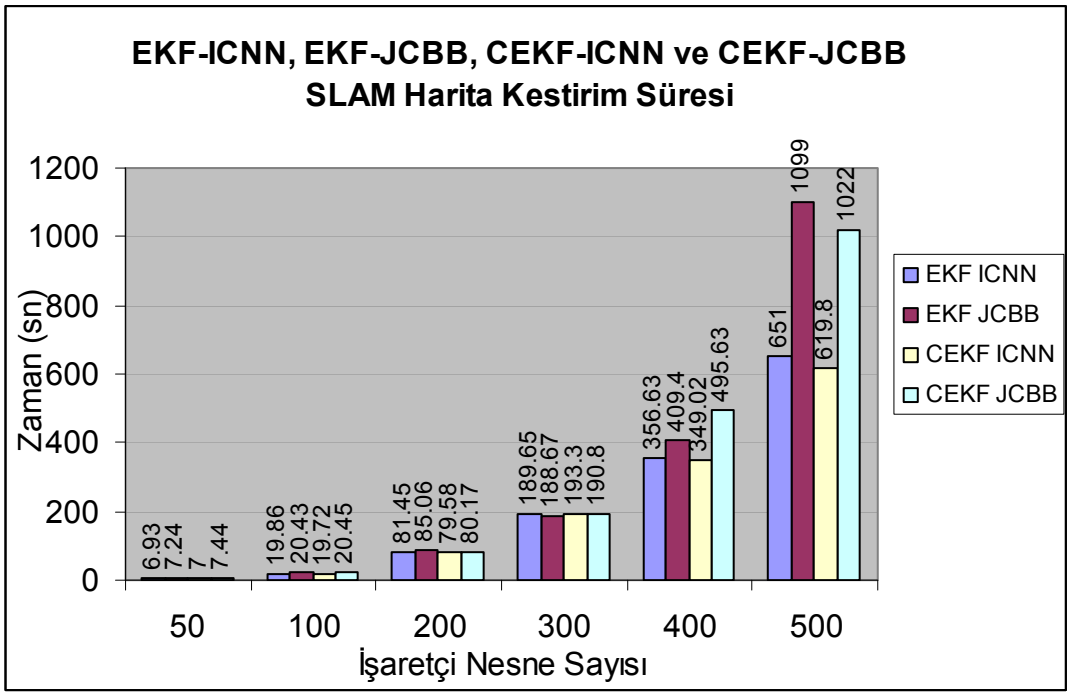
Tablo 7.2'de ise değişik sayıda (50, 100, 200, 300, 400 ve 500) işaretçi nesne sayısıyla GKF ve SGKF algoritmalarının BUYK ve BUDB veri ilişkilendirme algoritmaları ile kullanıldığında güncelleme, veri ilişkilendirme işlemleri için harcadıkları ortalama zaman saniye olarak gösterilmiştir. Bu tablolara ait grafikler şekil 7.29 ve 7.33 arasındaki şekillerde gösterilmiştir.




**Şekil 7.29:** GKF (EKF) ve SGKF (CEKF) tabanlı SLAM'da BUYK (ICNN) ve BUDB (JCBB) algoritmalarının son adımda harcadığı işlemci zamanları

Şekil 7.29 GKF ve SGKF tabanlı SLAM'ın BUYK ve BUDB algoritmaları kullanılarak son adımda harcadıkları işlemci zamanlarını göstermektedir. Bazı farklılıklar simulasyonun çalıştığı bilgisayardan kaynaklanabilir. İşaretçi nesne sayısı arttıkça harcanan zaman artmaktadır. GKF-BUYK, GKF-BUDB, SGKF-BUYK ve SGKF-BUDB algoritmalarının son adımdaki zamanları birbirine yakın çıkmaktadır.

Şekil 7.30'da GKF-BUYK, GKF-BUDB, SGKF-BUYK ve SGKF-BUDB tabanlı SLAM'da harita kestirimi için harcanan toplam zaman görülebilir. İşaretçi nesne sayısı 400'ün altında iken algoritmalar birbirine yakın işlem zamanlarına sahipken işaretçi nesne sayısı 500 olduğunda BUDB veri ilişkilendirmesi kullanan GKF ve SGKF tabanlı SLAM'ın harcadığı işlemci zamanı artmaktadır. Çünkü BUDB veri ilişkilendirme algoritması eşlemeleri sadece gözlem ve öngörü olarak taramamakta aynı zamanda eşlemelerin geçerliliğini de taramaktadır. Bu işlem BUDB'yi veri ilişkilendirme açısından güçlü kılmakta fakat harcanan zamanı da artırmaktadır. Eğer harcanacak zaman makul seviyelerde kalırsa doğru harita için bu zaman farkı feda edilerek BUDB veri ilişkilendirme algoritması tercih edilebilir.

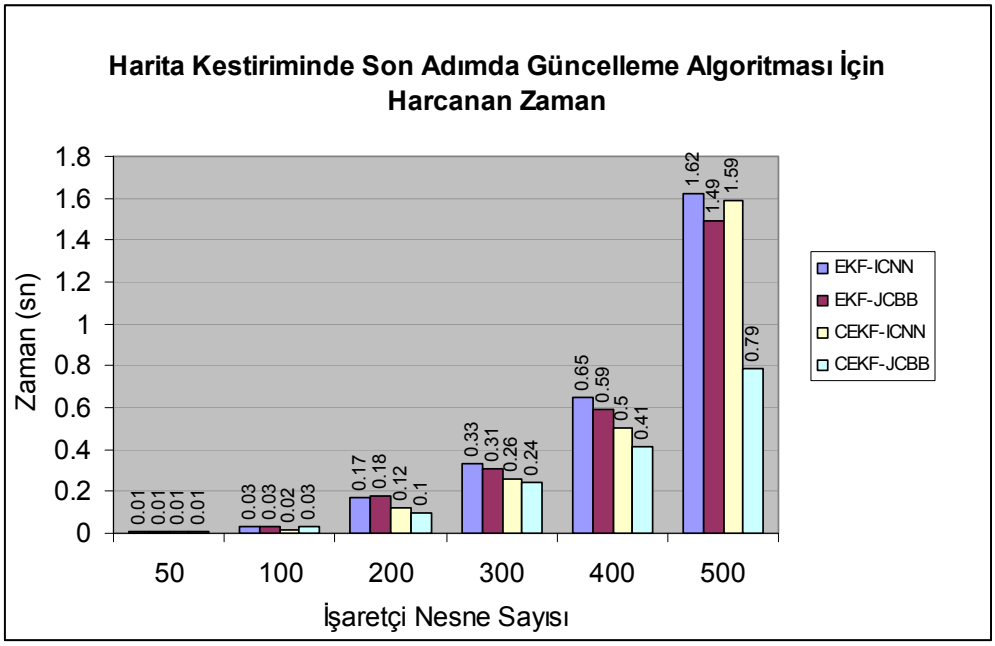



**Şekil 7.30:** GKF-BUYK (EKF-ICNN), GKF-BUDB (EKF-JCBB), SGKF-BUYK (CEKF-ICNN) ve SGKF-BUDB (CEKF-JCBB) tabanlı SLAM'da harita kestirimi için toplam harcadığı işlemci zamanları

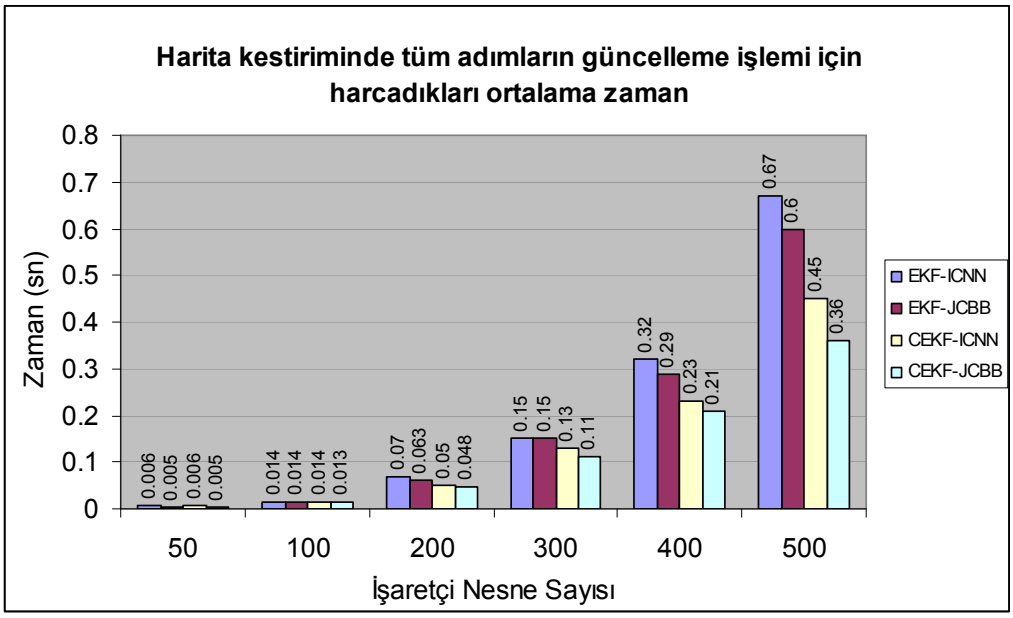
Şekil 7.31'de GKF-BUYK, GKF-BUDB, SGKF-BUYK ve SGKF-BUDB tabanlı SLAM'ın harita kestiriminde son adımda kestirimin güncelleme adımı için harcanan zaman, Şekil 7.32'de ise tüm adımlarda kestirimin güncelleme adımı için harcanan ortalama zaman görülebilir. Grafiklerdende anlaşılacağı üzere işaretçi sayısı arttıkça güncelleme işlemi için hem son adımda harcanan zaman hemde bütün adımlarda harcanan ortalama zaman da artmaktadır. İşaretçi nesne sayısı 100 iken güncelleme işlemi için son adımda harcanan zaman 0.03 saniye, ortalama harcanan zaman ise 0.013 saniye civarlarında iken, işaretçi nesne sayısı 500'e çıkarıldığında son adımda harcanan zaman GKF için 1.6 sn, SGKF için 0.8 sn ve ortalama harcanan zaman ise GKF için 0.6 sn, SGKF içinse 0.4 sn civarlarında olmaktadır.

Şekil 7.33'de GKF-BUYK, GKF-BUDB, SGKF-BUYK ve SGKF-BUDB tabanlı SLAM'nin harita kestiriminde kestirimin veri ilişkilendirme işlemi için bütün adımlarda harcanan ortalama zamanı görmekteyiz. Buna göre veri ilişkilendirme algoritmalarını işlemek için harcanan zaman işaretçi nesne sayısı arttıkça artmaktadır. İşaretçi nesne sayısının 400'ün altında olduğu durumlarda BUDB için harcanan zaman BUYK için harcanan zamanlarla aynı iken, nesne sayısı 500'e çıkarıldığında BUDB için harcanan zaman BUYK için harcanan zamandan daha

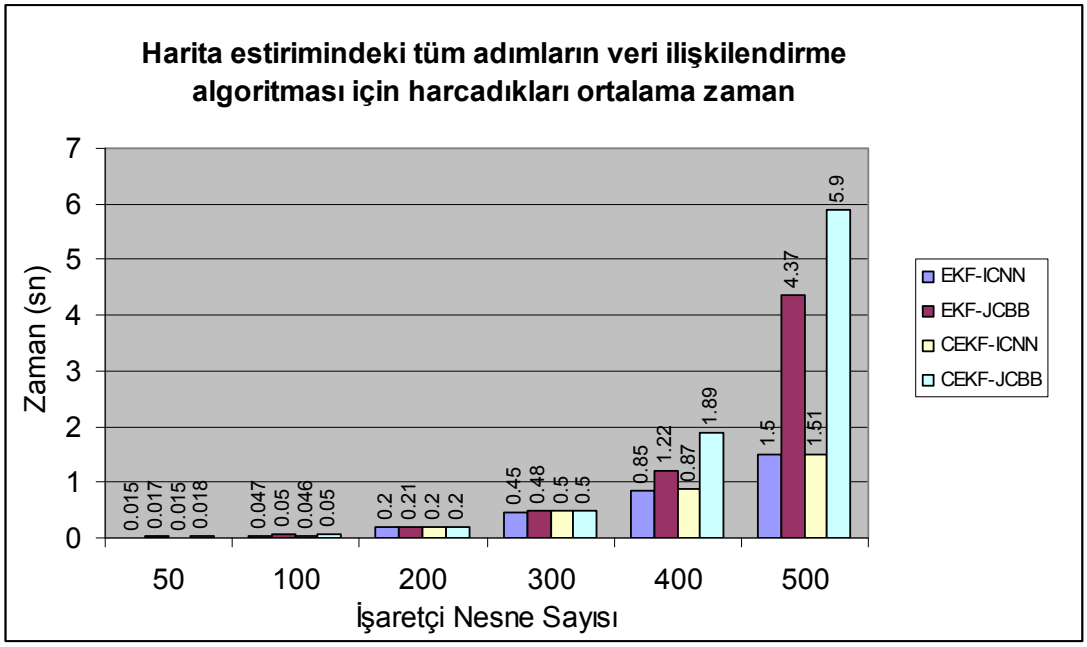
fazla artış göstermektedir. BUDB veri ilişkilendirme algoritması büyük alanlardaki işaretçi nesnelerin sayıca çok fazla olduğu durumlarda çok fazla zaman tükettiği için farklı veri ilişkilendirme algoritmalarına ihtiyaç duyulabilir.




**Şekil 7.31:** Harita kestiriminde son adımda güncelleme algoritması için harcanan işlemci zamanları




**Şekil 7.32:** Harita kestiriminde tüm adımlarda güncelleme algoritması için harcanan ortalama işlemci zamanları




**Şekil 7.33:** Harita kestiriminde tüm adımlarda veri ilişkilendirme algoritması için harcanan ortalama işlemci zamanları

Bu zamanlar gerçek zamanlı SLAM uygulamalarında da test edilerek gerçeklenmeli ve doğru kestirimle beraber düşürülmeye çalışılmalıdır. Bunun için bu yöntemler diğer yöntemlerle de kıyaslanarak en uygun zamanı makul seviyede ki bir hata oranı ile kestirebilen algoritmalar gerçek zamanlı SLAM uygulamalarımızda kullanılmalıdır. Hem diğer kestirim metodları hemde diğer veri ilişkilendirme algoritmaları da bu testlere tabi tutularak en uygun ikili seçilerek kullanılmalıdır.

# 8. SONUÇLAR VE TARTIŞMA

Yapılan iki simulasyon uygulaması neticesinde Bölüm 7'de elde edilen grafik ve tablolar incelendiğinde SGKF tabanlı SLAM'ın GKF tabanlı SLAM'a göre çok daha hızlı bir şekilde çalıştığı ve GKF tabanlı SLAM'a göre de hata oranının yok denecek kadar az olduğu söylenilebilir. GKF ve SGKF tabanlı SLAM BUYK ve BUDB algoritmaları ile çalıştırıldığında elde edilen simulasyon sonuçlarına göre BUYK veri ilişkilendirme algoritmaları için daha az zaman harcansa da araç ve işaretçi nesneleri konumlandırmada oluşturduğu hatalardan dolayı gerçek zamanlı SLAM uygulamalarında tercih edilmezler. BUDB algoritması her iki filtre ile de güzel sonuçlar vermiştir. SLAM yapılırken BUDB algoritmasının, BUYK'ye göre biraz fazla zaman harcasa da gözlemlerle, işaretçi nesneleri doğru bir şekilde ilişkilendirdiği ve araç ile işaretçi nesne konumlandırmalarını gerçeğe yakın bir şekilde kestirilmesine olduğundan dolayı tercih edilmesi kaçınılmazdır. Tablo 8.1'de GKF-BUYK, GKF-BUDB, SGKF-BUYK ve SGKF-BUDB tabanlı SLAM için performans karşılaştırma tablosu gösterilmiştir..

Tablo 8.1 GKF-BUYK, GKF-BUDB, SGKF-BUYK ve SGKF-BUDB algoritmaları performans karşılaştırma tablosu

| Filtre Tipleri | Veri İlişkilendirme Algoritmaları | |
|---|---|---|
| | BUYK | BUDB |
| GKF | İşlemci Zamanı Yüksek<br>Hata Oranı Yüksek<br>SONUÇ: KÖTÜ | İşlemci Zamanı Yüksek<br>Hata Oranı Düşük<br>SONUÇ: ORTA |
| SGKF | İşlemci Zamanı Düşük<br>Hata Oranı Normal<br>SONUÇ: İYİ | İşlemci Zaman Düşük<br>Hata Oranı Düşük<br>SONUÇ: ÇOK İYİ |

SLAM ile ilgili en büyük iki problem olan hesaplama yoğunluğu ve veri ilişkilendirme probleminin çözümüne yönelik bir çok çalışma yapılmış ve yapılmaya devam etmektedir. Daha ileri ki çalışmalarda yeni makalelerde ortaya sunulmuş algoritma ve teknikler incelenip karşılaştırılarak daha iyi sonuçlar elde etmek mümkündür. Hesaplama problemine çözüm getiren Sınırlandırımış Lokal Altharitalama Filtresi (SLAF) ve Sınırlandırılmış Relatif Altharitalama Filtreleri (SRAF) ile veri ilişkilendirme problemine çözüm getiren Birleştirilmiş Sınırlandırılmış Veri İlişkilendirme (BSVİ) algoritmaları da incelenerek SLAM için optimum metod bulunabilir. Burada yapılan çalışmalar stokastik haritalamayı esas alan teknikler üzerinden yapılan çalışmalardır. Görüntülü SLAM, Bilgi Filtreli SLAM, Çokaraçlı SLAM ve Hiyerarşik SLAM metodları da hep doğru haritayı kabul edilen makul bir zamanda gerçek zamanlı olarak çıkarmayı amaçlamış SLAM metodlarıdır. SLAM konusunda son yıllarda giderek artan çalışmalar daha önce ortaya çıkarılmış SLAM tekniklerini geliştirmeye yönelik olarak yapıldığı gibi yeni SLAM tekniklerinin de bulunması için yapılmaktadır. SLAM'nin insansız kara araçları için hem sanayide hemde askeri amaçlı kullanımının önümüzdeki yıllarda yaygınlaşması kaçınılmazdır.

## KAYNAKLAR


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

[2] **Castellanos, J.A., Neira, J. ve Tardos, J.D.**, 2006. Lecture Notes on SLAM and Consensus in Data Association, *Department of Informatics and System Engineering. Univ. Zaragoza*

[3] **Kurt, Z.,** 2007. Eş Zamanlı Konum Belirleme ve Haritalamaya Yönelik Akıllı Algoritmaların Geliştirilmesi, *Yüksek Lisans Tezi,* Y.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

[4] **Newman, P.M.,** 2002. Mobile Robot Localization and Mapping on in Extensive Outdoor Environments, *PhD. Thesis,* Department of Mechanical and Mechatronics Engineering, University of Sydney, Australia.

[5] **Williams, S.B.,** 2001. Efficient Solution to Autonomous Mapping and Navigation Problems, *PhD. Thesis,* Department of Mechanical and Mechatronics Engineering, University of Sydney, Australia.

[6] **Newman, P.M.,** 1999. On the Structure and Solution of SLAM Building Problem, *PhD. Thesis,* Department of Mechanical and Mechatronics Engineering, University of Sydney, Australia.

[7] **Bailey, T. and Durrant-Whyte, H.F.**, 2006. Simultaneous Localization and Mapping (SLAM): Part I, *IEEE Robotics and Automation Magazine,* **June 2006,** 99-108.

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

[9] **Guivant, J.E. and Nebot, E.M.**, 2001. Optimization of the Simultaneous Localization and Map-Building Algorithm for Real-Time Implementation, *IEEE Transcations on Robotics and Automation, June 2001,* **17, 3,** 242-257.

[10] **Neira, J. and Tardos, J.D.**, 2001. Data Association in Stochastic Mapping Using the Joint Compatibility Test, *IEEE Transcations On Robotics and Automation, December 2001,* **17, 6,** 890-897.

[11] **Dissanayake, G.M.W.M., Newman, P., Clark, S., Durrant-Whyte, H.F. and Csorba, M.**, 2001. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem, *IEEE Transcations on Robotics and Automation, June 2001,* **17, 3,** 229-241.

[12] **Castellanos, J.A., Montiel J.M.M., Neira, J., and Tardos, J.D.,** 1999. The SPmap: A Probabilistic Framework for Simultaneous Localization and Map Building, *IEEE Transcations on Robotics and Automation, October 1999,* **15, 5,** 948-952.

[13] **Rodriguez-Losada, D., Matia, F., Pedraza, L., Jimenez, A., and Galan, R.**, 2007. Consistency of SLAM-EKF Algorithms for Indoor Environments, *Intelligent Robotic Systems, Springer Science*.

[14] **Ye, C. and Borenstein, J.,** 2002. Characterization of a 2-D Laser Scanner for Mobile Robot Obstacle Negotiation, *International Conference on Robotics and Automation,* May 2002, 2512-2518.

[15] **Castellanos, J.A., Neira, J., and Tardos, J.D.,** 2004. Limits to the Consistency of EKF-Based SLAM, *5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, July 5-7,*

[16] **Castellanos, J.A., Martinez-Cantin, R., Neira, J., and Tardos, J.D.,** 2004. Robocentric Map Joining: Improving the Consistency of EKF-SLAM, *JRAS Conference 2006,*

[17] **SLAM Summer School,** 2002. Lecture Notes, Presentation, Practical Exercises and Codes

[18] **SLAM Summer School,** 2004. Lecture Notes, Presentation, Practical Exercises and Codes

[19] **SLAM Summer School,** 2006. Lecture Notes, Presentation, Practical Exercises and Codes

## ÖZGEÇMİŞ

**Deniz KAVAK**, 1979 yılında Afyon Karahisar ilinin Şuhut ilçesi Hallaç köyünde doğmuştur. İlk, orta öğrenimini tamamladıktan sonra 1997 yılında Afyon Anadolu Öğretmen Lisesi'nden mezun olmuştur. 2002 yılında Ege Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü'nü başarıyla tamamlamıştır. Lisans öğrenimi boyunca bitirme tezi dahil bir çok projeden birincilik, ikincilik ve üçüncülük dereceleri elde etmiştir. 2002 yılında Balıkesir Üniversitesi Elektrik-Elektronik Mühendisliği bölümünde 20 ay araştırma görevlisi olarak çalışmış ve 2004 yılında İstanbul Teknik Üniversitesi Kontrol ve Otomasyon programında yüksek lisansa 2008 yılında da doktoraya başlamıştır. 2004 yılından itibaren Ereğli Demir Çelik Fabrikaları'nda Elektronik-Otomasyon Mühendisi olarak halen çalışmaktadır.