T.C

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞLARI İLE GPS DESTEKLİ NAVİGASYON SİSTEMİ

Süleyman CANAN

DOKTORA TEZİ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI
Konya, 2006

ÖZET

Doktora Tezi

YAPAY SİNİR AĞLARI İLE GPS DESTEKLİ NAVİGASYON SİSTEMİ

Süleyman CANAN

Selçuk Üniversitesi
Fen Bilimleri Enstitüsü
Elektrik-Elektronik Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Ramazan AKKAYA
2006, 156 sayfa

Jüri: : Prof. Dr. Mehmet BAYRAK
Prof. Dr. Novruz ALLAHVERDİ
Doç. Dr. Semih ERGİNTAV
Yrd. Doç. Dr. Ramazan AKKAYA
Yrd. Doç. Dr. Abdullah ÜRKMEZ

Bu çalışmada; açık ortamlarda, beton ve asfalt zeminlerde hareket edebilen karasal bir mobil robot tasarlanmış ve gerçekleştirilmiştir. Tasarlanan mobil robot, son teknoloji mutlak konumlandırma sistemi olan GPS ve bağıl konumlandırma sistemlerinde kullanılan fiber optik jiroskop ve enkoderler ile donatılmış, navigasyon algoritmaları ve sensör verisi toplayan yazılımları geliştirmeye açık ve esnek olarak tasarlanmıştır.

Geleneksel navigasyon ve sensör entegrasyon algoritması olan Kalman filtresine alternatif bir yöntem olarak yapay sinir ağları (YSA) ile sensör entegrasyonu ve navigasyon yöntemi geliştirilmiştir. YSA öğreneceği sistem hakkında bir ön bilgiye ihtiyaç duymaz ve nonlineer sistem tanılamada çok başarılıdır. YSA ile gerçekleştirilen navigasyonda, literatürde geliştirilen benzer yöntemlerden farklı olarak doğrusal olmayan oto regresiv sistem tanılama metodunun (NARX) YSA' ya uygulanması ile kararlı ve hızlı sistem tanılama sağlanmıştır. Sensör entegrasyonu yönteminde, sistemin kesintisiz konum bilgisi üretmesi, navigasyon sisteminin performansını belirlemektedir. Bu nedenle mutlak sensör bilgilerinin kesintiye uğradığı durumlarda da mobil robotun, konumunu kestirebilmesi gerekir. Yapılan çalışmada, YSA-NARX ile gerçekleştirilen GPS destekli sensör entegrasyonu sayesinde navigasyon sistemi, kabul edilebilir hata seviyelerinde kesintisiz konum bilgisi üretebilmekte ve başarıyla konum kestirebilmektedir. Gerçekleştirilen sistem, iki farklı test güzergahında simule edilmiş ve pratik olarak denenmiştir. Elde edilen simülasyon ve deneysel sonuçlardan mobil robotun konumunun belirlenmesinde YSA-NARX sensör entegrasyonunun Kalman filtresine alternatif bir yöntem olabileceği görülmüştür.

Anahtar Kelimeler: Navigasyon, Mobil Robot, GPS, Yapay Sinir Ağları, NARX, Parakete, Kalman Filtre, Sensör Entegrasyonu

ABSTRACT

PhD Thesis

GPS AIDED NEURAL NETWORK NAVIGATION SYSTEM

Süleyman CANAN

Selçuk University
Graduate School of Natural and Applied Sciences
Department of Electrical and Electronics Engineering

Supervisor: Assist. Prof. Dr. Ramazan AKKAYA
2006, 156 pages

Jury: Prof. Dr. Mehmet BAYRAK
Prof. Dr. NOVRUZ ALLAHVERDİ
Assoc. Prof. Dr. Semih ERGİNTAV
Assist. Prof. Dr. Ramazan AKKAYA
Assist. Prof. Dr. Abdullah ÜRKMEZ

In this study, an outdoor land based mobile robot that can navigate on concrete and asphalt roads has been designed and implemented. The mobile robot is equipped with the latest state of the art GPS absolute navigation system, and relative sensors as fiber optic gyroscope, and encoders. The mobile robot's navigation algorithms and sensor data logging software are flexible and open to improvement.

An alternative navigation and sensor integration method using artificial neural network (ANN) has been proposed versus the traditional Kalman filtering. Neural networks do not need any pre-knowledge of the system involved and they are well suited for nonlinear system identification. Hence, the proposed method differ from similar works in the literature by implementing the nonlinear autoregressive with external input system identification methodology (NARX) having great stability and learning speed by using ANN. The aim of sensor integration in navigation systems is to predict positioning data even where the absolute sensor data is locked or unavailable. This feature gives the predictability performance of the navigation system. In this study the ANN-NARX aided GPS navigation system was simulated and supported with experimental works on two different test trajectories. It has been shown that predicted position errors are acceptable and continuous position prediction process is successfully achieved from graphical plots and error analysis. As a result, navigation using ANN-NARX sensor integration may be used as an alternative to the Kalman filter.

Keywords: Navigation**,** Mobile Robot, GPS, Artificial Neural Network, NARX, Dead Reckoning, Kalman Filter, Sensor Integration

ÖNSÖZ

Mobil robot araştırmaları ve çalışmaları, gelişen sensör teknolojilerinin ucuzlaması ve kolay elde edilebilirliği son on yılda oldukça hız kazanmıştır. Mobil robotlar, insanlar için tehlikeli olan ve sürekli tekrar eden işlerde kullanılmaktadır. Yapılan araştırmalar ışığında mobil robotlar, uzay araştırmaları, nükleer santraller, gözetleme ve savunma, bomba imhası, zararlı kimyasal maddelerin taşınması ve işlenmesi sahalarında, açık hava müzelerinde gezi robotları olarak gün geçtikçe kendilerine daha fazla yer bulmaktadır. Günümüzde navigasyon sistemleri, neredeyse hareket eden bütün araçlara entegre edilmeye çalışılmaktadır. Yapılan birçok araştırmada mobil robotların daha otonom olması için farklı algoritmalar test edilmekte ve geliştirilmektedir. Bu nedenle, navigasyon sistemlerinde akıllı algoritmalar kullanılması ve bunların yeni yöntemler ile birleştirilmesi, mobil robotun bulunduğu ortama göre algoritmasındaki parametreleri uyarlayabilmesi oldukça cazip görülmektedir. Ülkemizde robot araştırmalarında büyük bir boşluğu dolduracağı inancıyla bu tezde öğrenebilen navigasyon sistemi üzerine bir çalışmanın yapılmasına karar verilmiştir. Bu amaçla prototip bir mobil robot platformu, navigasyon sistemi ve yapay sinir ağları modeli geliştirilmiştir. Navigasyon sistemlerinde yeni ufuklar açacağı düşüncesiyle gerçekleştirdiğimiz bu robotun navigasyon sisteminin tasarımının her aşamasında hiçbir yardımını esirgemeyen ve çalışmalarımı büyük bir titizlikle yönlendiren danışmanım Yrd. Doç. Dr. Ramazan AKKAYA' ya, tez izleme komitesi üyeleri Doç Dr. Semih ERGİNTAV'a ve Yrd. Doç. Dr. Abdullah ÜRKMEZ'e teşekkürlerimi sunarım.

Ayrıca TÜBİTAK MAM Yer ve Deniz Bilimleri Enstitüsü yönetimine ve mobil robotun saha çalışmalarında desteklerini esirgemeyen Dr. Erkan DANACI, mesai arkadaşlarım Alpay BELGEN ve Levent KURT' a da teşekkürlerimi sunarım.

Bu tez çalışmasının tamamlanmasında daha çok zaman ayırmam için ortam hazırlayan mutluluk kaynağım olan eşim Ümmühan' a, oğlum Sinan Kerim'e ve maddi manevi tüm desteklerinden dolayı aileme minnettarlığımı ve şükranlarımı sunarım.

İÇİNDEKİLER

ÖZET ..... **i**
ABSTRACT ..... **ii**
ÖNSÖZ ..... **iii**
İÇİNDEKİLER ..... **iv**
SİMGELER ve KISALTMALAR ..... **vii**

1. GİRİŞ **1**
1.1 Çalışmanın Amacı ve Önemi 2
1.2 Tezin Organizasyonu 4

2. KAYNAK ARAŞTIRMASI **5**

3. MOBİL ROBOT NAVİGASYON SİSTEMLERİ **9**
3.1 Navigasyon Koordinat Sistemleri 9
3.1.1 Yer merkezli yer sabit koordinat sistemi 10
3.1.2 UTM koordinat sistemi 12
3.1.3 Yerel koordinat sistemi 12
3.1.4 Mobil robot gövde eksenleri 13
3.2 Navigasyon Çeşitleri 14
3.2.1 Bağıl navigasyon sistemleri 14
3.2.1.1 Parakete 14
3.2.1.2 Odometre ve jiroskop ile navigasyon 25
3.2.1.3 Ataletsel navigasyon sistemi 25
3.2.2 Mutlak navigasyon sistemleri 27
3.2.2.1 GPS ile Navigasyon 27
3.2.2.2 GPS ile konumlandırma 32
3.2.3 Hibrid navigasyon Sistemleri 34
3.2.3.1 Kalman filtresi ile mobil robot navigasyonu 36

4. YAPAY SİNİR AĞLARI **40**
4.1 YSA' nın Tarihçesi 40
4.2 Biyolojik Nöronun Yapısı 41
4.3 YSA' nın Özellikleri 42
4.4 Yapay Sinir Hücresi 43
4.5 Aktivasyon Fonksiyonları 44

4.6 YSA'ların Yapılarına Göre Sınıflandırılması 46
4.6.1 İleri beslemeli yapay sinir ağları 46
4.6.2 Geri beslemeli yapay sinir ağları 46
4.7 YSA' ların Öğrenme Algoritmalarına Göre Sınıflandırılması 47
4.8 Çok Katmanlı Perseptronlar 48
4.8.1 Geriye yayılma algoritması 49
4.9 Öğrenme Faktörleri 53
4.10 Yapay Sinir Ağlarında Genelleme 56
4.10.1 Eğitimi erken sonlandırma metodu 57
4.10.2 Otomatikleştirilmiş ayarlama metodu 58
4.11 YSA ile Nonlineer Sistem Tanılama 58
4.11.1 YSA sistem tanılama modelleri 60
4.11.2 NARX modeller 61
4.11.3 NARMAX modeller 61
4.11.4 NOE modeller 62
4.11.5 Tanılanacak sistemin derecesinin tespiti 63
4.11.6 Korelasyon testleri 66
4.11.7 Varyans karşılaştırması 67

5. MOBİL ROBOT PLATFORMUNUN TASARIMI VE GERÇEKLEŞTİRİLMESİ 68

5.1 Mobil Robotun Mekaniksel Tasarımı 69
5.2 Mobil Robotta Kullanılan Navigasyon Sensörleri 71
5.3 Mobil Robotun Elektronik Donanımı 74
5.4 Mobil Robot Veri Toplama Yazılımı 78
5.5 MATLAB Yazılımları 81
5.5.1 Görsel ara yüz 81

6. MOBİL ROBOT KONUM KESTİRİM SİSTEMİ UYGULAMALARI VE SONUÇLARI 84

6.1 YSA Eğitimi ve Test Güzergahlarının Belirlenmesi 84
6.2 Mobil Robot Sensör Verilerinin İncelenmesi 87
6.3 Yapay YSA Eğitim Güzergahının Oluşturulması 92
6.4 YSA ile Elde Edilen Sonuçlar 95
6.4.1 YSA NARX modelinin oluşturulması 95
6.4.2 YSA-NARX modelinin tanılama performansı 103

6.5 YSA ile Test Güzergahlarının Simülasyonları 107
6.5.1 GPS desteği olmadan YSA ile G_1 test güzergahında konum kestirimi 108
6.5.2 GPS destekli YSA ile G_1 test güzergahında konum kestirimi 111
6.5.3 GPS destekli ve GPS desteği olmadan YSA ile G_2 test güzergahında konum kestirimi 114
6.6 Kalman Filtresi ile Mobil Robot Navigasyonu 119
6.6.1 GPS desteği olmadan ve GPS destekli Kalman filtresi ile G_1 test güzergahında konum kestirimi 119
6.6.2 GPS desteği olmadan ve GPS destekli Kalman filtresi ile G2 test güzergahında konum kestirimi 124

7. SONUÇ VE ÖNERİLER 129
7.1 Sonuç 129
7.2 Öneriler 131

8. KAYNAKLAR 133

EK-A UTM ZON HARİTASI 137
EK-B ISA KART DEVRE ŞEMASI 138
EK-C MOBİL ROBOT VERİ TOPLAMA YAZILIMI 142

SİMGELER ve KISALTMALAR

SİMGELER

Ω	Açısal hız (°/sn)
η	Öğrenme oranı
τ, τ_s	GPS alıcısına ve uydulara ait gecikmeler
ΔD	Örnekleme zaman aralığında alınan yol
δ_j	yapay sinir ağı hata sinyali
$a_{sağ}$, a_{sol}	Sağ ve sol teker ivmeleri
c	Işık hızı
d_n	Yapay sinir ağı istenen çıkışlar
e	Eksantrisite
f	Basıklık
$f_a(.)$	Aktivasyon fonksiyonu
L	Fiber optik yolun uzunluğu
L1, L2	L1 ve L2 frekans bandı
N	Fiber optik sargı sayısı
P_0	Kestirim hatası kovaryans matrisi
Q	Sistem gürültü kovaryans matrisi
$\hat{R}_{u\varepsilon}(\tau)$	Kroskorelasyon fonksiyonu
$\hat{R}_{\varepsilon\varepsilon}(\tau)$	Otokorelasyon fonksiyonu
T	GPS alıcısının sinyal alma zamanı
t_s	Sinyalin uydudan ayrıldığı gerçek zaman
T_s	Sinyalin uydudan ayrılma zamanı
$u(k)$	Model girişi
$U_{çıkış}$	Çok katmanlı bir yapay sinir ağında çıkış katmanının çıkışı
$v_{sağ}$,v_{sol}	Sağ ve sol teker hızları
w_{zi}	Nöron girişleri ile nöronlar arasındaki bağlantı ağırlık katsayısı
$\hat{y}(k)$	Model çıkışı
$Y_{ağ}$	Nöron girişlerine gelen verilerin toplamı
$Y_{çıkış}$	Nöron çıkışı

KISALTMALAR

ADALINE	Adaptive Linear Model
ADC	Analog Digital Converter
ANN	Artificial Neural Network
ARMAX	Auto Regressive Moving Average with eXternal input
ARX	Autoregressive with eXternal input
AS	Anti Spoofing
BP	Back Propagation
C/A	Coarse Acquisition
DC	Direct Current
DGPS	Differential Global Positioning System
ECEF	Eath Centered Earth Fixed
ENU	East North Up
FOJ	Fiber Optik Jiroskop
GPS	Global Positioning System
HDOP	Horizontal Dilution Of Precision
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LKF	Linear Kalman Filter
MADALINE	Multiple Adaptive linear Model
MASER	Microwave Amplification by Stimulated Emission of Radiation
MIMO	Multiple Input Multiple Output
MISO	Multiple Input Single Output
MLP	Multiple Layer Perceptron
MSE	Mean Square Error
MSERE	Mean Square Error Regulated
MSEW	Mean Square Error Weight
NARMAX	Nonlinear Auto Regressive Moving Average with eXternal input
NARX	Nonlinear Auto Regressive with eXternal Input
NED	North East Down
NFIR	Nonlinear Finite Impulse Response

NMEA	National Martine Electronic Association
NNSYSID	Neural Network System Identification
NOE	Nonlinear Output error
OE	Output Error
PPS	Precise Positioning
PRN	Pseudo Random Number
PWM	Pulse Width Modulation
RF	Radio Frequency
RLJ	Ring Lazer Jiroskop
RMS	Root Mean Square
RTK	Real Time Kinematic
SA	Selective Availability
SISO	Single Input Single Output
USB	universal Serial Bus
UTM	Universal Transverse Mercator
VAF	Variance Accounted For
WGS–84	World Geodetic System -84
YSA	Yapay Sinir Ağları
TID	Time Index

1. GİRİŞ

Mobil robotlar endüstriden, sağlığa, eğlenceden askeri sahaya kadar pek çok sektörde günlük hayatımızda daha fazla yer almaktadır. Hızlı, hatasız ve tam zamanında üretim, endüstride robotlaşmayı kaçınılmaz kılmıştır. Robotlar, endüstriyel üretim hatlarında yerleri sabitlenmiş ve önceden programlanmış belli görevleri yerine getirmekteydiler. Bu robotların sensörleri belli sayıda ve neredeyse hepsi aynı özellikleri taşımaktaydı. Gerçek dünyada insanların sürekli yer değiştirmesi ve hareket halinde olmasından dolayı, sabit görevler icra eden robotların pek faydalı olmadığı fark edilmiş ve insanlara yardımcı olabilecek mobil robotların tasarlanması ve üretilmesi yoluna gidilmiştir. Mobilitenin robotlara getirdiği ek mühendislik yükleri ise; mobil robotların çevrelerini tanıması, bulundukları ortamdan bilgi toplaması ve bu bilgiler doğrultusunda karar verme mekanizmalarının geliştirilmesini gerektirmiştir. Bir başka ifadeyle mobil robotların öğrenebilmeleri gereksinimi doğmuştur. Bu gereksinimleri karşılamak için çevresini algılayabilecek farklı sensörlere ihtiyaç duyulmuştur. Bunun yanında; daha hızlı ve küçük hacimli bilgisayarların mobil robotlar ile bütünleştirilmesi kaçınılmaz olmuştur. Gelişmiş elektronik donanımları tamamlayan zeki algoritmaların, programlama dillerinin geliştirilmesi ve araştırılması yoluna gidilmiştir.

Mobil robotlar, kullanım alanlarına göre kapalı ortamlarda ve açık ortamlarda çalışan mobil robotlar olmak üzere iki kısma ayrılabilir; Kapalı ortam mobil robotları daha çok fabrika, laboratuar gibi yerlerde yardımcı olarak kullanılan mobil robotlardır. Açık ortam mobil robotları ise askeri, arama-kurtarma ve turistik amaçlı kullanılan mobil robotlardır. Bu iki farklı alanda kullanılan mobil robotların arasındaki temel fark, açık ortamlarda çalışan mobil robotların çok değişken çevre şartlarında, farklı mekaniksel engellerle karşılaşma olasılığının artmasıdır. Kapalı ortamda, mobil robotun bulunduğu çevrenin nerdeyse hiç değişmemesi ve ortamda bulunan referans noktalarının sabit olması sebebiyle, mobil robot çevresini çok kısa sürede öğrenebilmektedir. Bunun aksine, açık ortamlarda, mobil robotun kendi konumunu bilmesi önem taşımaktadır.

Mobil robotlar, hava, deniz ve yeraltında çalışan mobil robotlar olarak da karşımıza çıkmaktadır. Bu nedenle mobil robotların sadece karasal mobil robotlarla

sınırlandırmamak gerekir. Örneğin karadan karaya seyreden kıtalar arası roketleri havada çalışan mobil robotlar olarak sınıflandırabiliriz. Mobil robotlar kendi kendine hareket ediyor ve mobilitesini kontrol eden algoritmalara sahip ise; insansız mobil robotlar olarak da adlandırılır.

Yukarıda da bahsedildiği gibi mobil robotlar, verilen görevi yapabilmek için nerede bulunduklarını bilmek zorundadırlar. Bu çalışmanın konusu olan mobil robot navigasyon sistemleri günümüzde en çok araştırılan konulardan birisidir ve araştırılmaya da devam edilecektir.

1.1 Çalışmanın Amacı ve Önemi

Kara, hava ve deniz mobil robotlarının konumlarını hesaplamak ve tespit etmek için çok çeşitli algoritmalar geliştirilmiştir. Her farklı yöntemin kendine göre avantajları ve dezavantajları bulunmaktadır. Genelde tüm algoritmalar faklı sensör kombinasyonlarını kullanarak ve sensör bütünleştirmesini gerçekleştirerek konum bilgisini elde etmeye çalışır. Bir mobil robotun navigasyonu üç ana soru başlığı altında toplanabilir. Bunlar;

- Mobil robotun konumu nedir?
- Mobil robot nereye gidecek?
- Mobil robot hedefine nasıl gidecek?

Bu üç problemden ilki bu tez çalışmasının ana konusudur, diğer problemler de kendi başlarına birer araştırma konusudur ve ayrıca ele alınmalıdır.

Mobil robotların konum tespiti için en çok kullanılan ve bilinen yöntem, Kalman filtresi ile konum belirleme tekniğidir. Bu tez çalışmasında önerilen yöntem, farklı sensörlerden elde edilen verilerin bir Yapay Sinir Ağı (YSA) modeli kullanılmak suretiyle bütünleştirilerek konum bilgisinin kestirilmesine dayanmaktadır.

Bu çalışmada amaç; birbirinden farklı sensörlerden elde edilen verilerin YSA modeli kullanılarak mobil robot konumunun kestirilmesidir. Burada YSA, sensör bütünleştirme görevini yapmaktadır. YSA, bütünleştirme işlemini yaparken mobil robotun dinamik modelini ve sisteme etki eden hataları da öğrenir. YSA'nın

öğrenebilmesi için eğitim verilerine ihtiyacı vardır. Bu veriler oluşturulurken minimum boyutta olmaları ve YSA'nın bu veriler arasındaki ilişkiyi hızlı öğrenmesi ve doğru sonuç üretmesi amaçlanır. Bunu sağlamak amacıyla optimal boyutta eğitim veri setini elde etmek için uygun bir deneysel düzenek ve eğitim sistematiği geliştirilmiştir. YSA'nın diğer bütün algoritmalara göre en önemli farkı, sistem hakkında bir ön bilgiye ihtiyaç duymadan sistemi öğrenebilmesidir. Bundan dolayı, mobil robot konum kestirim yöntemi için YSA, mobil robot sistemlerinde kullanılan sensörler, mobil robot dinamiği ve robotun çalışma ortamı hakkında bir bilgiye ihtiyaç duymaz. Otomatik mobil robotlar, sürekli değişen farklı ortamlarda çalıştıklarından dolayı konumlandırma sisteminin kendini sürekli olarak ortama adapte edebilmesi çok önemlidir. Mobil robotların mekaniğinde ve bulunduğu çevrede meydana gelebilecek değişiklikler için YSA tekrar eğitilerek bu değişiklere karşı kendini uyarlayabilir.

Mobil robotların konumlarının kestirimi problemi ile ilgili yapılan birçok çalışma daha çok çevresi ve özellikle iyi bilinen kapalı ortamlarda yapılmıştır. Bu çalışma ile açık ortamda YSA kullanılarak mobil robotun konumunun kestirimi için yeni bir yöntem geliştirilmiştir. Konum kestirimi için mutlak ya da bağıl sensör guruplarından birinin konum verisi üretmediği durumlarda YSA ile gerçekleştirilen sistemin kesintisiz konum üretme performansı araştırılmış ve klasik navigasyon algoritması olan Kalman filtresi ile karşılaştırılmıştır. Mobil robotların günümüzde neredeyse bütün kullanım alanlarında (askeri, sivil, arama kurtarma, endüstriyel, uzay teknolojileri, tıp) yer aldıkları düşünülürse, kesintisiz konum kestirim probleminin başarılı ve hızlı bir şekilde çözümlenmesi büyük bir önem taşımaktadır.

Özet olarak bu tez çalışmasında;

- Açık ortamda hareket edebilen dört tekerli karasal bir mobil robot ve mobil robot navigasyon verilerini toplayacak bir bilgisayar sistemi tasarlanmış ve gerçekleştirilmiştir
- Tasarlanan elektronik bir donanımla sensörlerin entegrasyonu sağlanmıştır
- Gerçek zamanda veri toplama yazılımları gerçekleştirilmiştir
- Farklı mobil robot araştırmaları için bir Ar-Ge platformu oluşturulmuştur

- Son teknoloji navigasyon sensörleri (GPS, Jiroskop) kullanılmış ve bu sensörlerin mobil robota entegrasyonu gerçekleştirilmiştir.
- Çok disiplinli sahalardan oluşan mobil robot ve navigasyon teknolojileri konusunda bilgi birikimi oluşturulmuştur
- Mobil robot navigasyonu için YSA tabanlı yeni bir yaklaşım sunulmuş ve uygulanmıştır
- Güçlü bir simülasyon platformu olan MATLAB programlama dilinde YSA ve Kalman filtresi algoritması geliştirilmiştir

1.2 Tezin Organizasyonu

Bu tez çalışması, mobil robot, navigasyon sensör sistemleri, elektronik tasarım, farklı programlama dillerinde yazılım geliştirme, yapay sinir ağları, Kalman filtresi ile navigasyon algoritması, GPS gibi kendi başlarına birer bilim dalı haline gelmiş konuların bütününü içermektedir. Bu açıdan teorik tabanı çok geniş olan bu konulardan tezde kullanılan kısımlar, özet halinde ayrıntıya girilmeden verilmiştir. Daha detaylı bilgiler referanslarda belirtilen kaynaklardan edinilebilir. Bu bağlamda ikinci bölümde tez konusu ile ilgili kaynak araştırması verilmiş, Bölüm-3'te ise navigasyon sistemleri ve ilgili sensörler, karasal mobil robotlar için navigasyon çeşitleri ve yöntemleri ele alınmıştır. Bölüm-4'de YSA algoritması, YSA çeşitleri ve YSA ile lineer olmayan sistemlerin çözümü ve YSA ile sistem tanılama konuları verilmiştir. Bölüm-5'te tasarlanan ve gerçekleştirilen mobil robotun mekanik kısımları, elektronik, sensör ve yazılımlar tanıtılmıştır. Bölüm-6'da YSA ile NFIR modelinin mobil robot sensörlerinden elde edilen veriler ile navigasyon sistemine uygulanması ve Kalman filtresi ile elde edilen navigasyon sonuçlarının karşılaştırılması verilmiştir. Son olarak Bölüm-7'de çalışmanın sonuçları sunulmuş ve genel bir değerlendirme yapılmıştır. Bu bölümde ayrıca öneriler yer almaktadır

2. KAYNAK ARAŞTIRMASI

Tezde kaynak olarak kullanılan ve son yıllarda Mobil Robotların konumlandırması ile ilgili yapılmış olan çalışmaların literatür özetleri aşağıda verilmiştir.

Kim J. H. ve Cho H. C. (1993), mobil robotlarda konum bilgisi elde etmek için sıkça kullanılan parakete yönteminde, mobil robotların hareket halindeyken, tekerlerinde, zemine bağlı olarak meydana gelen kaymaları ve oluşan konum hatalarını YSA ile kestirmeye çalışmışlardır. Kestirilen hata miktarını parakete ile yapılan konum hesabı ile işleme sokarak daha doğru bir konum bilgisini deneysel ve simülasyon çalışmalarıyla elde etmeyi başarmışlardır.

Rivals ve ark. (1994), mobil robotların ve karasal taşıtların dinamik modellerini, referans model öğrenme yöntemi kullanarak YSA'da eğitmişlerdir. Mobil robotun otomatik kontrol döngüsünde sürekli eğitilen bir YSA kullanmışlardır.

Camapum J.F. ve ark. (1996) bir mobil robotun kapalı mekanlarda konumunu belirlemek için, robota lazer tarayıcı eklemişlerdir. Lazer tarayıcı belirli yerlere yerleştirilmiş olan işaretlerden yansıyan lazer ışınlarından, işaretlere olan mesafeleri ölçtükten sonra geri beslemeli bir yapay sinir ağına girdi olarak uygulamışlardır. Mobil robotun mutlak konumu YSA'nın çıkışına uygulayarak, YSA'yı eğitmişlerdir. YSA ile lazer mesafelerini kullanarak mobil robotun konumunu kestirmeyi başarmışlardır.

Rintanen K. ve ark. (1996), tank tipi paletler yardımıyla hareket edebilen bir mobil robotun, dış ortamlarda konumunu ve belirlenen bir güzergahın takibini yapmışlardır. Paletli karasal araçlar için dinamik bir model önermişler ve bu modele dayanarak, fiber optik jiroskop ve RTK (Real Time Kinematic) GPS yardımıyla konum tespiti gerçekleştirmişlerdir. Mobil robotun mevcut konumundan bir sonraki konuma nasıl gideceğini, kübik spline eğrileri yardımıyla optimal güzergah elde ederek, robotun motorlarına uygun kontrol işaretleri üreterek sağlamışlardır.

Bonnifait ve Garcia (1998), odometre metodu ile navigasyon ve çevreye yerleştirdikleri konumları belli yapay işaretleri kullanarak araştırma amaçlı mobil robotları için bir konum kestirme metodu geliştirmişlerdir. Mobil robota monte ettikleri optik sensör yardımı ile işaretler tespit edilerek robotun duruş açısı ile işaret arasındaki açıyı hesapladıktan sonra, odometre bilgisini de kalman filtresi yardımıyla bütünleştirerek mobil robotun optimal konumu bulmasını sağlamışlardır.

Shoval ve ark. (1998), deneysel çalışmalarında kullandıkları mobil robotun konumunu, odometre ve lazer konumlandırma metotlarını kullanarak kapalı bir mekanda hesaplamışlardır. Her iki konumlandırma sistemi arasında oluşan farkı, farklı zeminler için, önceden eğitilen YSA'nın girişine uygulamışlardır. YSA algılayıcı konum farklarından mobil robotun hangi zemin üzerinde hareket ettiğini sınıflandırarak bulmuştur. YSA'nın çıkışına göre, sistemin hangi konumlandırma metoduna göre konum kestireceğine karar vermesini sağlamışlardır.

Dorobantu R. ve Zebhauser B. (1999), üç eksenli ivmeölçer ve tek eksenli jiroskop ile tasarladığı ataletsel ölçüm biriminden üretilen bağıl navigasyon verilerini ve DGPS (Differential GPS) 'ten üretilen mutlak konum bilgisini lineer Kalman filtresi (LKF) yardımıyla bütünleştirerek bir otomobilin hızını, konumunu ve doğrultu açısını kestirmeyi başarmışlardır.

Nebot ve Durrant-Whyte (1999), geliştirmiş oldukları ataletsel konumlandırma sistemini karasal bir araçta kullanmışlardır. Çalışmalarında karasal aracın ilk konum ve ilk açısal durumunu doğru kestiren algoritma sunmuşlardır. Ataletsel konumlandırma sisteminin oluşturduğu gürültüleri minimize eden bu algoritmanın çıkışı ve GPS' den elde edilen konum bilgisini Kalman filtresi yardımıyla bütünleştirerek, karasal aracın konumunu yüksek bir doğrulukla elde etmişlerdir.

Townsend ve Tarassenko (1999), yaptıkları deneysel çalışmalarda, kapalı bir ortamda, Mobil Robota monte ettikleri kızıl ötesi sensörlerden elde edilen uzaklık verileri ile aynı anda Mobil Robotun ortamdaki mutlak konumunu kaydetmişlerdir. Elde ettikleri veri seti ile radyal tabanlı bir YSA modelini eğitmek için kullanmışlardır. Eğitilmiş olan YSA'ya sadece kızıl ötesi sensörlerden elde ettikleri uzaklık verilerini uygulayarak mobil robotun konumunu kestirmişlerdir.

Faruqi ve Turner (2000), yaptıkları çalışmada ataletsel sensörler ile GPS verileri için ayrı ayrı genişletilmiş kalman filtresi kullanarak gevşek bağlantılı bütünleştirme tekniğini uyguladıkları bir konumlandırma sistemi yapmışlardır.

Lee J.H. ve ark. (2000), tasarladıkları navigasyon sisteminde çok antenli GPS alıcı kullanarak odometre sensörlerini, aracın hangi hız aralığında hareket ettiğine bakmaksızın kalibre etmişlerdir. Çok antenli GPS alıcısı ile çok düşük hızlarda hareket eden bir mobil aracın hızının ve doğrultusunun doğru bir şekilde ölçülebileceğini göstermişlerdir. Navigasyon sistemlerinde çift kalman filtresi kullanmışlardır. Kalman filtrelerinin biri ile parakete yöntemi kullanılarak hesaplanan aracın hız ve yön bilgileri kalibre edilmiş, diğer kalman filtresi ile parakete hesabından kaynaklanan aracın konum hataları düzeltmiştir. Bu çalışmaları ile uyguladıkları bu metodun klasik tek antenli GPS alıcı ve parakete entegrasyonundan daha yüksek performanslı sonuç aldıklarını kanıtlamaya çalışmışlardır..

Ojeda ve Borenstein (2000), mobil robotların yön bilgisini elde etmek için elektronik pusula kullanmışlardır. Elektronik pusulanın yapısından ve devrelerinden kaynaklanan ve mobil robotun duruş konumundan oluşan gürültülerin etkisini azaltacak yöntemler geliştirmişlerdir. Bu yöntemlerle güvenilir bir doğrulukta yön bilgisi elde edebileceklerini deneysel çalışmalar ile göstermişlerdir. Fakat dış ortamlarda oluşan manyetik alan etkilerinin çok zor azaltılabileceğini ortaya koymuşlardır. Ayrıca mobil robotların konumlarının kestirimi için yön bilgisinin önemine değinmişlerdir.

Torii T. (2000), tarımsal amaçlı bir aracın dikim alanlarında otomatik olarak hareket edebileceği bir sistem geliştirmiştir. Genetik algoritmalar, bulanık mantık, yapay sinir ağları ve kamera ile görüntü analizi gibi farklı teknikler kullanarak ve bu tekniklerin entegrasyonunu yapmış, ekili arazilerde aracın uygun dümen açı kontrolünü sağlayarak, belirlenen güzergahın takibini uygulamalı ve simülasyon çalışmaları ile göstermiştir.

Ryu Kyung B. (2001), yerleri belli farklı frekanslarda yayın yapan sesli işaretler kullanarak, kapalı mekanlarda deneysel amaçlı bir mobil robot konumu tespit etmeye çalışmıştır. Kullandığı metot, mobil robotta bulunan ses algılayıcılarına

ulaşan ses sinyallerinin faz kaymalarından, ses kaynaklarının yönlerini ve uzaklıklarını tespit ederek, triyangülasyon algoritması ile mobil robotun konumunu deneysel ve simülasyon çalışmaları ile kestirmeyi başarmıştır.

Gu ve Hu (2002), yaptıkları deneysel çalışmalarında dalgacık YSA modelini eğiterek mobil robotun kinematik modelini oluşturmuşlardır. Oluşan YSA' lı modeli, mobil robotu belli bir güzergah üzerinde hareket etmesi için sistemin kontrol döngüsüne entegre etmişlerdir..

Canan ve ark. (2004), tasarladıkları mobil robot için genişletilmiş Kalman filtresi ile konumlandırma sistemi tasarlamışlardır. Fiber optik jiroskop ve enkoder sensörlerinden elde ettikleri konumu, GPS konumu ile bütünleştirerek optimal konum bilgisi elde etmeyi deneysel çalışmalar ile göstermişlerdir.

Yim ve Oh (2004), bir otomobilin matematiksel dinamik modelini referans alarak, geri beslemeli bir YSA'yı bu modeli öğrenmesi için eğitmişlerdir. Eğitilen YSA'yı kapalı bir kontrol döngüsü içinde kullanarak otomobilin otomatik kontrolünü sağlamışlardır.

Canan ve Akkaya (2005), mobil robot konum kestirim sistemi için YSA modeli kullanmışlardır. Hızlı YSA eğitimi için özel referans eğitim güzergahı yöntemini geliştirerek deneysel çalışmalar ile göstermişlerdir.

3. MOBİL ROBOT NAVİGASYON SİSTEMLERİ

İlk navigasyon metotlarında, çevrede bulunan ve bilinen yerlere örneğin; ağaçlar, binalar, deniz kıyısı, adalar, dağlar gibi referans noktalarına göre konum belirlemesi yapılıyordu. Zaman içinde pusulanın keşfedilmesi, astronominin ve hassas mekanik saatlerin gelişmesi ile denizde ve karada daha doğru navigasyon yapılmıştır

Yirminci yüzyılda hava ve uzay araçlarının hızla gelişmesi, karmaşık ve doğruluğu yüksek navigasyon sistemlerinin araştırılması ve geliştirilmesi kaçınılmaz olmuştur. Günümüzde özellikle otomatik hareket edebilen araçların karmaşıklığı nedeniyle navigasyon işlemi ileri konum hesaplama yöntemleri ve sensör teknolojileri ile gerçekleştirilebilmektedir. Bu teknolojiler aracın, konumunu, hızını ve yönünü belirlemek için gerçek zamanda ve yüksek işlem hızlarında navigasyon imkanı sağlamaktadır. Üretim teknolojilerinde otomasyonun gelişimi ile bu tür konumlama sistemleri, artık daha ucuza elde edilebilmektedir.

Kısaca tanımlanacak olursa: **navigasyon**; hareket halindeki kara, hava, deniz ve uzay araçlarının konumunu, yönünü ve hızını tespit etmektir. Mobil robot sistemlerinde de kullanılabilecek ve navigasyonun kısa tanımına uyan sistem elemanları ve yöntemleri bu bölümde anlatılacaktır.

3.1 Navigasyon Koordinat Sistemleri

Herhangi bir kara, deniz veya hava aracının hangi koordinat sisteminde navigasyon yapacağı sistemi tasarlayanın seçimine kalmıştır. Fakat bu seçimi sınırlayan bazı faktörlerin göz önünde bulundurulması gerekir. Bu faktörlerden biri ve en önemlisi, navigasyonun yapılacağı mesafenin uzun veya kısa olmasıdır. Bu faktör, navigasyon koordinat sisteminin belirlenmesini etkiler. Örneğin; kısa mesafedeki navigasyonlar için tasarlanan mobil robotlarda yerel bir koordinat sistemi kullanılması iyi bir seçimdir. Uzun mesafelerde uçuş yapan uçaklar ve seyir eden gemiler, yeryüzüne bağlı olan koordinat sistemlerini kullanırlar. Gezegenler arası navigasyon yapan uzay araçları için, ataletsel olarak dönmeyen güneş merkezli bir koordinat sistemi seçilir. Bu koordinat sisteminin eksenlerini, uygun olarak seçilen yıldızların doğrultusu oluşturur.

3.1.1 Yer merkezli yer sabit koordinat sistemi

GPS sisteminde kullanılan referans kartezyen sistemi, yer merkezli yer sabit eksenler ECEF (Earth Centered Earth Fixed) 'dir. Bu eksenlerin merkezi yerin merkezi ile çakışıktır. İki ana ECEF koordinat sistemi vardır. Bunlar, kartezyen ECEF ve jeodezik ECEF koordinat sistemleridir.

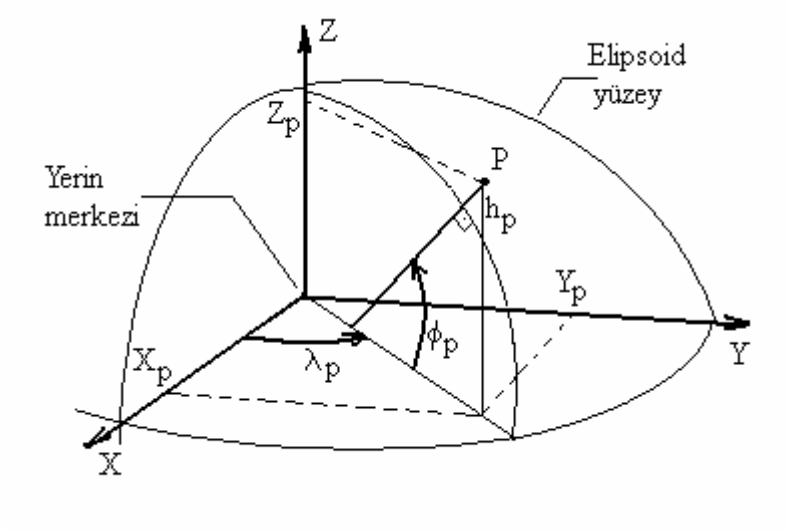
i) Kartezyen ECEF koordinat sistemi

Merkezi, dünyanın merkezi ile çakışık standart bir kartezyen koordinat sistemini ECEF koordinat sistemi olarak tanımlayabiliriz. X ekseni, başlangıç Greenwich meridyeni ve ekvator ile çakışık olup doğrultusu 0° boylamıdır. Y ekseni, 90° doğu boylamı doğrultusundadır. Z ekseni coğrafi kuzey kutbu doğrultusunda ekvator düzlemine diktir ve X,Y eksenleri ekvator düzlemini oluşturur. Şekil 3.1'de ECEF koordinat sistemi verilmiştir. X ve Y ekseni dünya ile birlikte dönmektedir ve bu koordinat sisteminde uzayda sabit doğrultular tanımlanamamaktadır. Yeryüzünde ECEF koordinat sisteminde tanımlanan bir nokta sürekli sabit olacaktır.

ii) Jeodezik ECEF koordinat sistemi

Jeodezik ECEF koordinat sistemini tanımlayan üç parametre vardır. Bunlar; λ, ϕ ve h' dır. Burada, λ boylamı, ϕ enlemi ve h referans yer elipsoidinden bağıl yüksekliği gösterir. Bir P noktasının boylamı, ekvator düzlemi ve yerin elipsoidal normali arasındaki açıdır. Enlemi, ekvator düzleminde başlangıç meridyeni ve noktanın ekvator düzlemine projeksiyonu ile oluşturduğu açı ile tanımlanır. Yükseklik, P noktasında elipsoit yüzeyine dik doğrultudaki ve elipsoidin yüzeyi arasındaki mesafedir. Bu tanımlanan değerlerin geometrik gösterimi Şekil 3.1 'de görülmektedir.

ECEF koordinat sistemi WGS-84 (World Geodedic System 1984) referans elipsoidi ile birlikte kullanılmaktadır (Wellenhof ve ark 1997). Yerkürenin şeklinin tam küre olmamasından dolayı en uygun modelle gösterilmesi gerekmektedir ve bu model de elipsoit şekli olarak seçilmiştir.



Şekil 3.1 ECEF koordinat sistemi

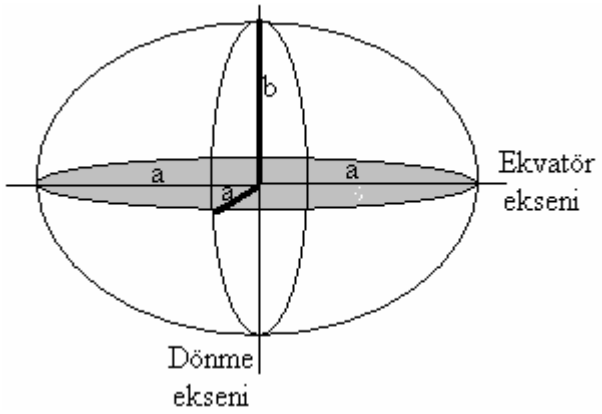
GPS sistemi için kullanılan jeodezik referans (datum) sistemi, WGS–84' dür. Şekil 3.2'de referans elipsoidin şekli verilmiştir. Elipsoid, *a* (büyük yarı eksen), *b* (kısa yarı eksen), *f* (basıklık) ve *e* (eksantrisite) gibi parametreler ile tanımlanır. Bu parametreler, aşağıdaki denklemler yardımıyla hesaplanır. Bu elipsoidin merkezi ECEF koordinat sisteminin merkezi ile çakışıktır.

$$f = \frac{1}{298.257223563}$$

$$a = 6378137\,m$$

$$b = a(1-f) = 6356752.31424518\,m$$

$$e = \sqrt{\frac{a^2 - b^2}{a^2}}$$



Şekil 3.2 Referans elipsoidi

3.1.2 UTM koordinat sistemi

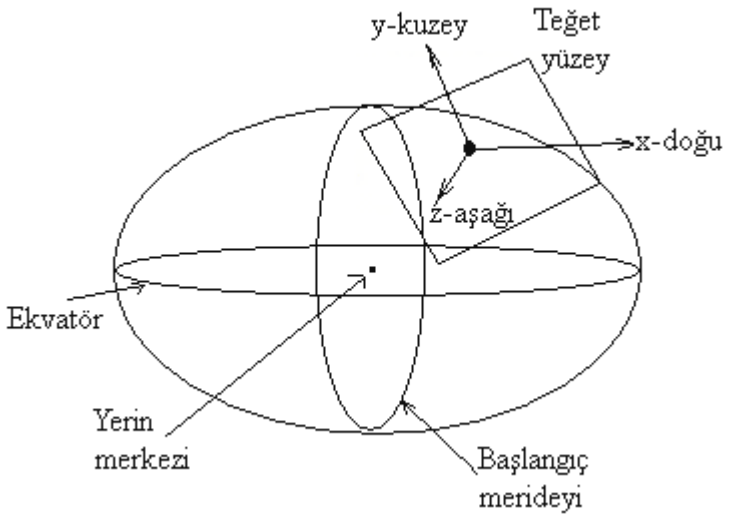
Bilinen ve çok yaygın olarak kullanılan bir diğer koordinat sistemi de, kısaca Universal Transverse Mercator (UTM) olarak tanımlanan koordinat sistemidir. Bu koordinat sisteminde dünya bir kutuptan diğerine her birinde 6 meridyen bulunan ve zon olarak adlandırılan dilimlere bölünmüştür. 180° batı meridyeninden başlayarak doğuya doğru yapılan bu bölmelerde toplam 60 zon mevcuttur. UTM koordinat sisteminde, ekvatordan kutuplara doğru ilerledikçe bozulmalar artar. Bu nedenle bozulmanın kabul edilebileceği üst sınırlar olarak 84° kuzey ve 80° güney paralellerinin ötesinde UTM koordinatlar kullanılmamaktadır. Yine negatif sayılar kullanılmaması için başlangıç meridyeni sıfır noktası 500.000 metre doğu ve ekvator başlangıç noktası 0 metre kuzey veya 10.000.000 metre güney değeri ile tanımlanmaktadır.

Her zon' da ortada yer alan meridyene, dilim orta meridyeni denir. Bu meridyenden sola ve sağa zon sınırlarına doğru gidildikçe bozulmalar başlar. Zon sınırları, bozulmanın kabul edilebilir boyutta olduğu uç noktalardır. UTM projeksiyon sisteminde Türkiye, dilim orta meridyenleri 27°, 33°, 39° ve 45° olan dört dilimde yer almaktadır. Yeryüzünün UTM projeksiyonu zon haritası EK-A verilmiştir.

3.1.3 Yerel koordinat sistemi

Karasal mobil robotların bağıl navigasyon sensörleri yerel koordinat sisteminde navigasyon verileri üretirler. Bu nedenle ECEF koordinat sisteminde çalışmak yerine yerel koordinat sisteminde çalışmak daha uygundur. GPS gibi mutlak konum bilgisi üreten sensörlerin verilerinin yerel koordinat bilgilerine dönüştürülmesi gerekir. Yerel koordinat sistemi, günlük hayatımızda sürekli kullandığımız kuzey, doğu ve aşağı yönleri olan bir kartezyen koordinat sistemidir.

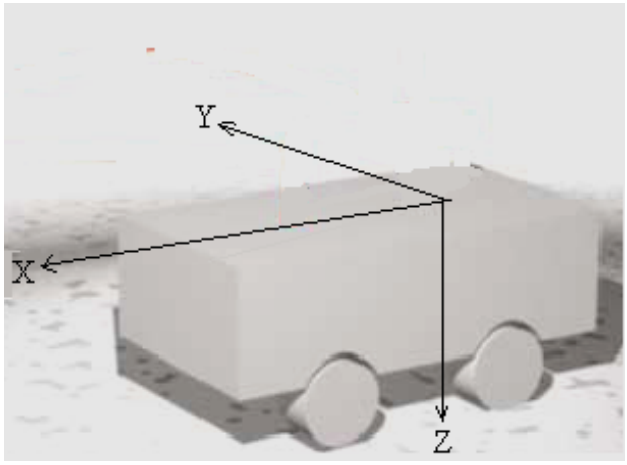
Yerel koordinat sistemini tanımlamak için yeryüzünde ilgilendiğimiz noktaya teğet bir yüzey geçirilir ve bu nokta yerel koordinat sisteminin orijini olur. Şekil 3.3' de görüldüğü gibi y ekseninin yönü gerçek kuzeyi, x ekseninin yönü doğuyu ve z ekseninin yönü yerin merkezini gösterir. Bu koordinat sistemi genellikle NED (North East Down) veya ENU (East North Up) olarak tanımlanır (Farrel 1999).



Şekil 3.3 Yerel koordinat sistemi

3.1.4 Mobil robot gövde eksenleri

Mobil robotun üzerindeki sensörlerin ölçtüğü veriler genelde mobil robotun gövde eksenlerine göredir. Buna göre mobil robotun gövdesi merkez alınarak bir kartezyen koordinat sistemi tanımlanabilir. Mobil robotun burun doğrultusu yönü x ekseni, sağ tarafı y ekseni, aşağı yöndeki eksen z olarak tanımlanır. Mobil robotun gövde eksenleri Şekil 3.4' de verilmiştir. Mobil robotun hızı x yönünde ölçülür, y ve z ekseninde ise hızın sıfır olduğu kabul edilir.



Şekil 3.4 Mobil robot gövde eksenleri

3.2 Navigasyon Çeşitleri

Mobil robot navigasyon sistemleri, üzerlerinde kullanılan sensörlere göre sınıflandırılabilir. Bağıl navigasyon için odometre ve ivmeölçerler, mutlak navigasyon için GPS veya radyo işaretler kullanılır. Mutlak navigasyon, bir dış referans noktasına göre, bağıl navigasyon ise bir koordinat sisteminde koordinatı bilinen bir noktaya göre gerçekleştirilir.

Gelişmekte olan ve yüksek teknoloji gerektiren haritalamaya dayalı navigasyon sistemi; çevresinden kamera yardımıyla görüntüler alıp işleyen, tanıyan ve önceden yüklenmiş bir veri tabanı ile karşılaştırarak mobil robotun navigasyon bilgilerini elde eder (Katyon 2000). Bağıl ve mutlak konum bulma sensörlerinin bir karışımı olan hibrid konumlandırma sistemleri, her iki navigasyon sisteminin artı ve eksilerini tamamlayan sistemlerdir.

3.2.1 Bağıl navigasyon sistemleri

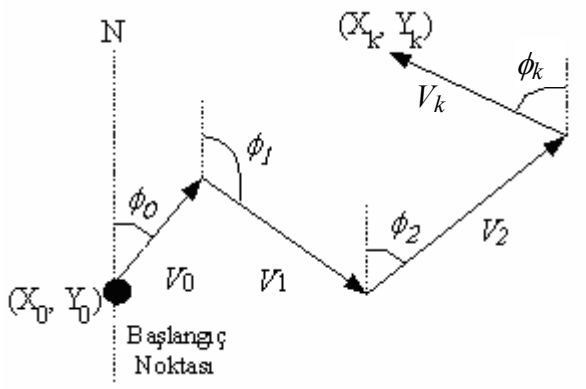
3.2.1.1 Parakete

Parakete, seyir halindeki bir aracın genel bir koordinat sisteminde, koordinatları bilinen referans işaretleri veya noktaları kullanmadan yapılan navigasyon çeşitlerinin ortak adıdır (Borenstein 1998). Parakete ile navigasyonun mobil robotlarda en çok kullanılan formu odometredir. Parakete hesabı, hız ve yön bilgisinden konum hesaplamaktır. Hız ve yön bilgisi seyir eksenine aktarılır ve mobil robotun ilk konum ve yön bilgisine eklenir. Parakete için gerekli olan bilgiler; hız ve yöndür. Bu bilgiler uygulama sahasına göre farklı sensörlerden elde edilebilir. Yön bilgisi kuzey eksen referans alınarak ölçülür.

Bu tez çalışmasında karasal bir mobil robot ele alınmaktadır. Karasal araçlar için kullanılabilecek sensörler; hız ölçümü için ivmeölçer ve odometre, yön için jiroskop ve pusuladır. Her şeyden önce parakete ile konumlandırma yöntemi bağıl konumlandırma kategorisine girer. Konumu tespit edilecek mobil robotun ilk konumu ve yönü önceden bilinmelidir. Böylece odometre donanımından sürekli veri gelmesi ile ilk konum ve yön bilgisi üzerine eklenerek yeni konum üretilir ve bilinen bir koordinat sisteminde konumlandırma yapılmış olur. Fakat zaman içinde sürekli

toplama işleminden kaynaklanan hatalar ortaya çıkar ve konumlandırma hatası sürekli artar. Zamanla artan hatalar sistemde mutlak konum ve yön güncelleme ihtiyacı doğurur. Fakat hata ne kadar artarsa artsın, sensörler veri ürettiği sürece konum ve yön bilgisi hesaplanabilmektedir. Parakete hesabı geometrik olarak Şekil 3.5' de gösterilmiş ve matematiksel olarak denklem (3.1) ile ifade edilmiştir.

$$\begin{aligned} X_k &= X_0 + V_0 \sin\phi_0 + V_1 \sin\phi_1 + V_2 \sin\phi_2 + ... + V_k \sin\phi_k \\ Y_k &= Y_0 + V_0 \cos\phi_0 + V_1 \cos\phi_1 + V_2 \cos\phi_2 + ... + V_k \cos\phi_k \end{aligned} \tag{3.1}$$



Şekil 3.5 Temel parakete hesabının geometrik gösterimi

a) Bağıl navigasyon sistemlerinde kullanılan sensörler

i) Odometre sensörü: Odometre, karasal araçlarda hız ölçmek için kullanılan bir sensördür. Aracın teker hareketini sağlayan dişli kutusuna veya aracın tekerine takılır. Aracın hareket etmesiyle odometre teker dönüş miktarı ile orantılı sayıda tetiklenmiş bir sinyal üretir. Bu tetiklenmiş sinyaldeki tetiklenme sayılarak aracın hızı ölçülür. Kara araçları normal şartlarda yani kaymadıkları durumda burunları doğrultusunda hareket ederler. Bu kriter dikkate alındığında odometrenin aracın burnu doğrultusundaki hızı ölçtüğü söylenebilir. Odometre çok kullanılan bir sensördür, bunun sebebi odometre kısa zamanlar için duyarlı sonuçlar üretmektedir, yüksek hızlarda örnekler sağlamakta ve maliyet olarak ucuzdur. Buna karşın odometre aracın hareket ettiği arazi koşullarına ve zemin yapısına çok duyarlıdır. Örneğin; tekerlerdeki kaymalar kat edilen mesafenin ve yön hesaplamalarında ciddi

hatalar oluşturur. Odometre sensörü zemindeki düzensizlikleri algılayamaz, bunun için yardımcı sensörlere ihtiyaç duyulur.

Bu tez çalışmasında odometre ve enkoder kelimeleri eşanlamlı olarak kullanılacaktır

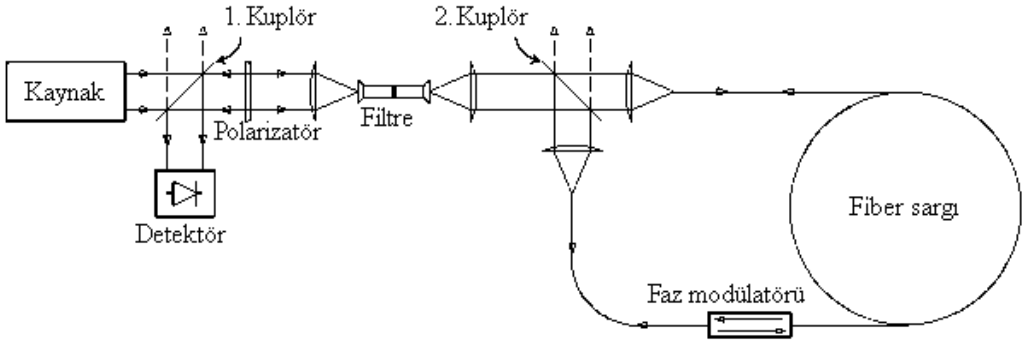
ii) Jiroskop sensörü: Jiroskop monte edildiği bir platformun açısal hızını ölçmeye yarayan bir sensördür. Tek bir jiroskop, tek bir geometrik eksende ölçüm yapabilir. Üç boyutlu bir eksende ölçüm yapabilmek, üç adet birbirine dik olarak yerleştirilen jiroskoplarla mümkündür. Birkaç çeşit jiroskop tipi vardır, bunlardan; mekaniksel donanımı ağırlıklı olan jirokararlaştırılmış jiroskoplar en pahalı olanlarıdır ve fiyatları 10000$ üzerindedir. Jirokararlaştırılmış jiroskoplar, en hassas ve duyarlı ölçüm yapabilen jiroskop tipleridir. Diğer çeşit jiroskoplar ise gövdeye bağlı jiroskoplardır ve fiyatları 50$ ile 2000$ arasındadır. Jiroskopun fiyatını en çok etkileyen faktör jiroskopun ölçüm kararlılığı ve yapısıdır. Gövdeye bağlı jiroskop tipleri ise;

- Piezo elektrik jiroskoplar
- Fiber optik jiroskoplar (FOJ)
- Ring lazer jiroskoplar (RLJ)

Bu tez çalışmasında da FOJ kullanılmıştır ve jiroskop yapısı olarak sadece FOJ incelenecektir, diğer tip jiroskoplara yer verilmeyecektir.

iii) Fiber optik jiroskop: FOJ' ların popülaritesi son on yılda fiber optik teknolojisindeki hızlı gelişimi sayesinde olmuştur. Sonuç olarak daha ucuz, daha küçük boyutlarda ve daha duyarlı FOJ' ların üretilmesini tetiklemiştir. Bu sayede FOJ'lar mobil robot uygulamalarında kendilerine kolayca yer bulmuşlardır ve başarılı birçok navigasyon sistemleri tasarlanmıştır (Barshan ve ark. 1995). FOJ'lar da RLJ'ler gibi interferometre prensibine göre çalışırlar. RLG' lerde dönme açısını ölçmek için yüksek gerilimde çalışan lazer ışını, özel aynalar ve optik yol olarak boşluk kullanılır. FOJ' larda çok daha düşük güçte ve yüksek ışıma sağlayan lazer diyot veya SLD (Super Luminescent Diode) ve optik yol olarak fiber optik kablo kullanılır. Şekil 3.6' da lazer ışını birinci kuplör, polarizatör ve filtreden geçtikten sonra fiber optik sargıya girmeden önce ikinci kuplörden iki yöne ayrılır. İki yöne

ayrılan lazer ışınları birbirlerine zıt yönde fiber optik kabloda yol alırlar. Her iki ışın, fiber optik yolunu tamamladıktan sonra tekrar ikinci kuplörden, filtreden, polarizatörden ve birinci kuplörden detektöre yönlendirilir.



Şekil 3.6 Fiber optik jiroskopların yapısı

Detektör, aynı anda giden ikiye ayrılmış ışınların geri dönüş zamanlarını ölçer. İki ışının geri dönüş zaman farkları, fiber optik sargının açısal dönme hızı ile orantılıdır. Lazer ışınını algılayan foto detektör küçük açısal hızlardaki dönme hareketini algılayamaz ve foto detektör nonlineer davranış sergiler. Bu olay Sagnac etkisi olarak bilinir. Bu olumsuz etkiyi gidermek için Şekil 3.6'da da görüldüğü gibi faz modülatörü kullanılmıştır. (Matthews 1990)

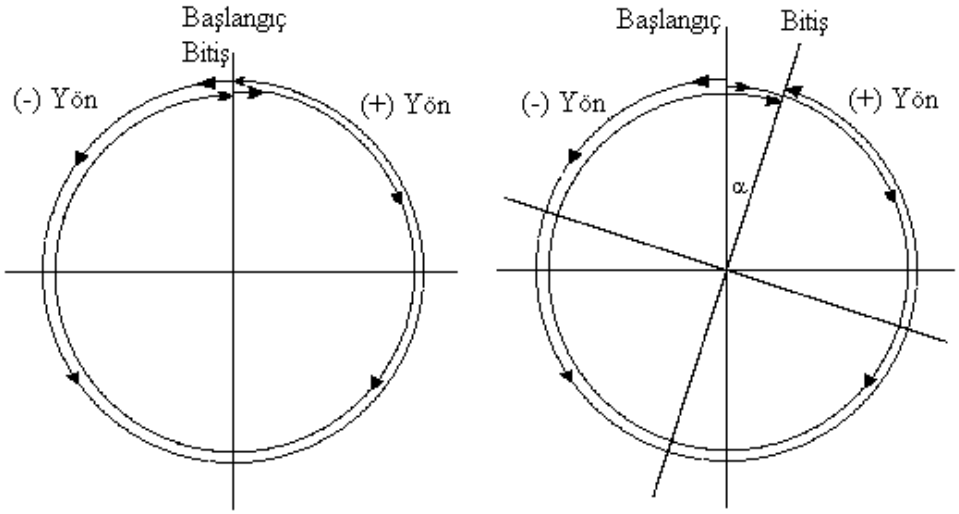
Sagnac etkisi detaylı bir şekilde Şekil 3.7' de verilmiştir. Şekli 3.7a' daki şekil durağan olan fiber optik sargıyı göstermektedir. Şekil 3.7b' deki şekil ise fiber optik sargı α kadar açıyı Ω açısal hızıyla döndüğünde, pozitif yönde ilerleyen ışın, başlangıç noktasına negatif yönde ilerleyen ışına göre daha uzun sürede varacaktır. Böylece sargı sayısı N olan ve lazer ışını tarafından kat edilecek fiber optik yolun uzunluğu

$$L = 2N\pi \tag{3.2}$$

olur.

Dönen fiber optik sargının optik yol farkı yani Sagnac etkisiyle ΔT zaman farkı kayması, (3.3) denklemi ile hesaplanabilir

$$\Delta T = \frac{L}{c - \Omega r} - \frac{L}{c + \Omega r} \tag{3.3}$$



Şekil 3.7 FOJ sargısının durağan ve hareketli durumları

Yukarıdaki (3.2) denklemi (3.3) denkleminde yerine konur ve sadeleştirilirse;

$$\Delta T = \frac{4N\pi r^2\Omega}{c^2 - r^2\Omega^2} \tag{3.4}$$

elde edilir.

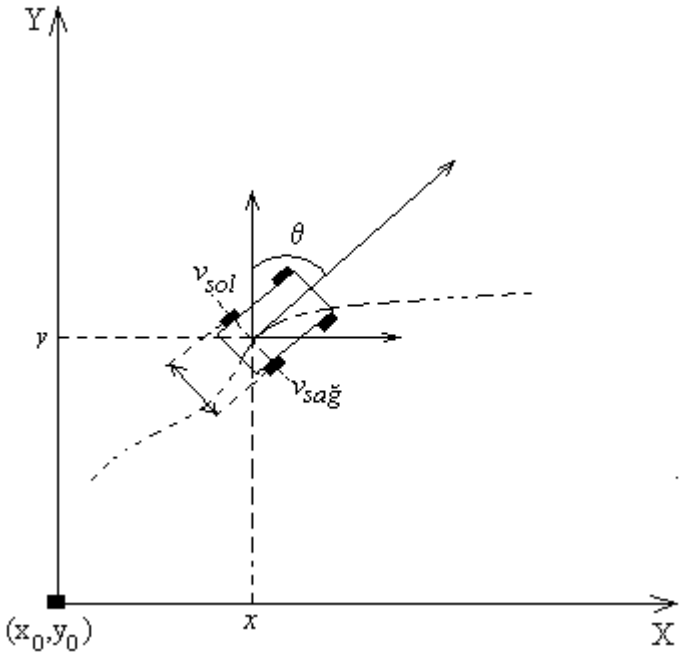
Burada r fiber optik sargının ortalama yarıçapı ve c ışık hızıdır. (3.4) denkleminde fiber optik sargının küçük açısal dönme hızları için $c^2 >> r^2\Omega^2$' dır ve (3.4) denklemi tekrar yazılırsa

$$\Delta T = \frac{4N\pi r^2\Omega}{c^2} \tag{3.5}$$

elde edilir.

b) Mobil robot için parakete hesabı

Şekil 3.8 deki gibi (X,Y) genel navigasyon koordinat sisteminde bulunan bir mobil robotun tekerleri birbirinden bağımsız dönebilmektedir. Bu yapıya göre dönebilen tekerlere diferansiyel teker sistemi denir.



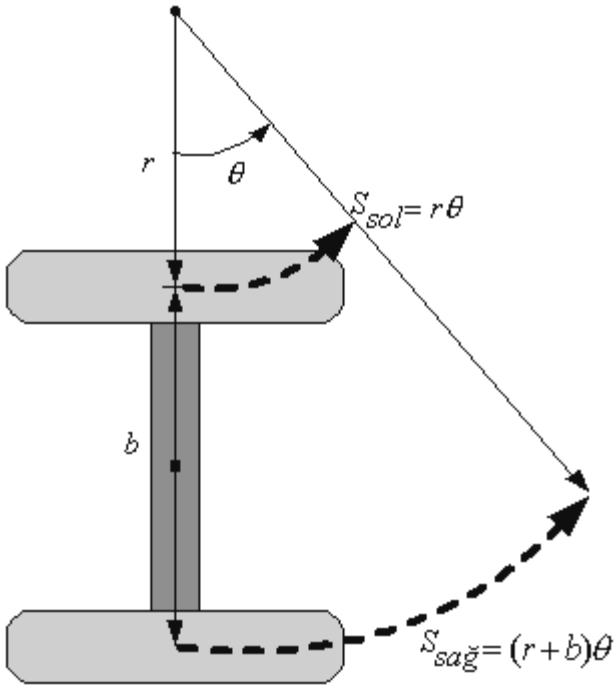
Şekil 3.8 Mobil robot navigasyon koordinat sistemi

Şekil 3.9' da mobil robot θ rad kadar açısal bir hareket yaptığında tekerlerin kat ettiği mesafe;

$$\begin{aligned} S_{sol} &= r\theta \\ S_{sağ} &= (r+b)\theta \\ S_{ort} &= (r+\frac{b}{2})\theta \end{aligned} \tag{3.6}$$

olarak yazılabilir. Burada $S_{sağ}$ ve S_{sol} sırasıyla sağ ve sol tekerlerin aldığı mesafe, S_{ort}, sağ ve sol tekerler arasındaki orta noktanın kat ettiği mesafe, b teker merkezleri arasındaki uzunluk, r dönme çapıdır.

Eğer tekerlerin ivmelenmesine izin verilirse robotun oluşturacağı hareket daha karmaşık olacaktır. Birçok uygulamada mobil robotun kütlesi küçüktür ve sistemin ivmelenmediği varsayılır. Bu tür küçük kütleli robotlarda teker hızları ani olarak değişir. Böylece robotun hareketi dairesele yakın kabul edilir. Yüksek kütleli robotlarda mutlaka ivme dikkate alınmalıdır. Denklem (3.7)' de tekerlerin dönme hızı, sabit ve sıfır ivme şartları altında oluşturulur



Şekil 3.9 Tekerlerin dönme esnasındaki oluşturdukları yol

Mobil robotun konumu ve yönü, arka tekerlerin arasındaki uzunluğun merkezine göre bulunur ve bu nokta, mobil robotun referans noktası olur. Bu noktaya göre mobil robotun x, *y* ve θ durum değişkenleri için aşağıdaki ifadeler yazılabilir*;*

$$v_{ort} = \frac{v_{sağ} + v_{sol}}{2} \tag{3.7}$$

$$\frac{dx}{dt} = v_{ort} \cos\theta \tag{3.8}$$

$$\frac{dy}{dt} = v_{ort} \sin\theta \tag{3.9}$$

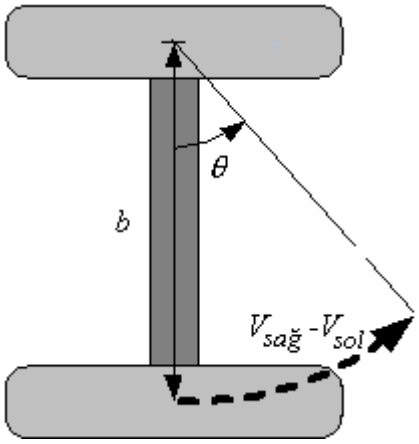
Burada $v_{sağ}$ ve v_{sol} sırasıyla sağ ve sol tekerlerin dönme hızlarıdır. Mobil robotun yön bilgisini ifade eden denklemi yazarken mobil robotun sol tekeri referans alınır ve aşağıdaki diferansiyel denklem elde edilir;

$$\frac{d\theta}{dt} = \frac{(v_{sağ} - v_{sol})}{b} \tag{3.10}$$

Şekil 3.10' da denklem (3.10)' un geometrik gösterimi verilmiştir. Denklem (3.10)' un t ye göre integrali ve ilk yön değeri $\theta(0) = \theta_0$ alınırsa

$$\theta = \frac{(v_{sağ} - v_{sol})}{b} t + \theta_0 \tag{3.11}$$

denklemi elde edilir. (3.11) denkleminde mobil robotun yönü, tekerlerin hızı ve zamanın fonksiyonu şeklinde elde edilir.



Şekil 3.10 Tekerlerin farklı hızlardaki durumu

(3.7) ve (3.11)' deki ifadedeler denklem (3.8) ve (3.9)' da yerine yazılır ve (3.8), (3.9) denklemlerinin integrali ve mobil robotun ilk konumu $x(0)=x_0$, $y(0)=y_0$ alınırsa;

$$x(t) = x_0 + \frac{b(v_{sağ} + v_{sol})}{(v_{sağ} - v_{sol})} \left[\sin\left(\frac{(v_{sağ} - v_{sol})t}{b} + \theta_0 \right) - \sin\theta_0 \right] \tag{3.12}$$

$$y(t) = y_0 + \frac{b(v_{sağ} + v_{sol})}{(v_{sağ} - v_{sol})} \left[\cos\left(\frac{(v_{sağ} - v_{sol})t}{b} + \theta_0 \right) - \cos\theta_0 \right] \tag{3.13}$$

yazılabilir. (3.12) ve (3.13) denklemlerinde teker hızları sabit tutulursa $b(v_{sağ} + v_{sol})/(v_{sağ} - v_{sol})$ ifadesi mobil robotun R dönme çapını verir ve mobil robot dairesel hareket yapar. (3.12) ve (3.13) denklemleri ile uygulama algoritması yazılırken $v_{sağ}=v_{sol}$ olması durumu da özel olarak dikkate alınmalıdır gerekir. Çünkü bu durumda payda sıfıra gidecektir ve bu ifadelerin limitleri alınarak L'Hospital türev kuralı uygulanması gerekir.

Birçok mobil robot uygulamasında, sistemi denetleyen işlemci gücü düşük olur ve yukarda bahsedilen denklem takımları sadeleştirilerek kullanılır. Mobil robot lineer bir güzergah takip ettiğinde ortaya çıkan matematiksel belirsizliklerden kurtulmuş olunur. Buna göre sadeleştirilmiş parakete denklemleri;

$$x(t) = v_{ort}\cos\theta + x_0$$

$$y(t) = v_{ort}\sin\theta + y_0 \tag{3.14}$$

$$\theta(t) = \frac{(v_{sağ} - v_{sol})}{2} + \theta_0$$

şeklinde yazılabilir.

Eğer geliştirilmiş olan modele ivme de eklenirse;

$$v_{sağ} = a_{sağ}t + v_{sağ}(0) \tag{3.15}$$

$$v_{sol} = a_{sol}t + v_{sol}(0) \tag{3.16}$$

$a_{sağ}$ ve a_{sol} sırasıyla sağ ve sol tekerlerin ivmesini, $v_{sağ}(0)$ ve $v_{sol}(0)$ sırasıyla sağ ve sol tekerlerin ilk hızlarını ifade eder.

(3.15) ve (3.16) ifadeleri denklem (3.8) ve (3.9)' da yerine konur; ve aşağıdaki tanımlamalar da yapılırsa;

$$\theta(t) = Ct^2 + Dt + \theta_0 \tag{3.17}$$

$$\frac{dx}{dt} = (At + B)\cos(Ct^2 + Dt + \theta_0) \tag{3.18}$$

$$\frac{dy}{dt} = (At + B)\sin(Ct^2 + Dt + \theta_0) \tag{3.19}$$

elde edilir.

Burada A,B,C ve D;

$$A = \frac{(a_{sağ} + a_{sol})}{2}, \qquad B = \frac{(v_{sağ} + v_{sol})}{2}$$
$$C = \frac{(a_{sağ} - a_{sol})}{2b}, \qquad D = \frac{(v_{sağ} - v_{sol})}{b} \tag{3.20}$$

olur.

Mobil robotun yönünü ifade eden diferansiyel denklem çözülmüştür, fakat denklem (3.18) ve (3.19) ile gösterilen diferansiyel denklemlerin nümerik metotlarla çözülmesi gerekir. Görüldüğü gibi ivme modelini mobil robotun hareket denklemlerine katmak fazladan nümerik işlemler gerektirmektedir. Bu tür işlemleri gerçekleştirmek, sistemin donanım maliyetinin artması demektir.

Sonuç olarak mobil robotu detaylı bir şekilde modellemek fazladan işlem ve donanım maliyeti getirmektedir. Basit parakete modelini kullanmak birçok uygulamada kolaylık sağladığı gibi konum hesaplamada da yeterli olmaktadır. Fakat mobil robotun, basit parakete modeli kullanıldığında, açısal dönme hareketlerinde daha fazla hata katacaktır. Model seçimi yapılacak uygulamaya ve istenilen konum duyarlılığına göre tespit edilmelidir. Türetilen matematiksel modellerde sadece odometre kullanılarak konum ve yön tespiti yapılmıştır. Bu modelde düzlemsel hareketler yeterince duyarlı algılanmasına karşın, yön bilgisi üretmede yeterince verimli değildir, çünkü tekerlerde oluşan mekaniksel gürültüler odometre donanımını etkiler. Bu hız bilgileri denklem (3.11)' e uygulandığında mobil robotun yönünde çok büyük hatalar doğurur ve (x,y) konum bilgilerini doğrudan etkiler.

Yön ve konum bilgilerinin aynı sensörden elde edilmesinden dolayı yön ve konum değerleri arasında yüksek bir korelasyon vardır. Bu korelasyonu gidermek için yön bilgisini daha hassas ölçebilen ve odometre sisteminden bağımsız farklı bir sensör kullanmak gerekir. Yön bulmada kullanılan iki tür sensör vardır, bunlardan biri dünyanın manyetik alanı yardımıyla kuzeyden sapmaları gösteren pusuladır, aynı zamanda pusula mutlak yön ölçen bir sensörüdür ve çevrenin manyetik şartlarına duyarlıdır. Diğer sensör ise, ataletsel bir referans koordinat sisteminde ölçüm yapan jiroskoptur. Jiroskop açısal hızölçer ve açı değerlerini elde etmek için jiroskopun ürettiği açısal hızların integrali alınır.

c) Parakete hesabı hataları

Bilinen bir koordinat sisteminde, parakete ile hesaplanan mobil robotun konumu ile mutlak konum sensörleri tarafından daha doğru hesaplanan mobil robot konumu arasındaki fark, konum hatası olarak tanımlanır. Konum hataları; sistematik olan ve sistematik olmayan hatalar olarak ikiye ayrılır.

i) **Sistematik hatalar:** Sistematik hataları, genelde mobil robotun mekanik aksamının mükemmel olmaması, tekerler arasındaki çap farkı, teker ekseninin kayıklığı, enkoderlerin çözünürlüğünün düşük olması veya örnekleme zamanının yetersiz olmasından kaynaklanan hatalardır. Sistematik hatalar uzun süreli ve aracın kendine özgü teknik özelliklerine bağlı olarak sürekli tekrarlanır (Borenstein 1996). Bu tür hatalar uzun süreli gözlemler sonucu tespit edilir ve istatistiksel modellerle giderilebilir (Martinelli 2002) veya hatalara sebebiyet veren kısımlar iyileştirilerek hatanın etkisi azaltılabilir. Sistematik hataların bazı bileşenleri mobil robotun takip ettiği güzergahtan bağımsızdır. Bundan dolayı kapalı güzergahlarda, başlangıç noktasına geri dönünce bu bileşenlerin etkilerinin kaybolduğu görülmüştür (Kelly 2001).

ii) **Sistematik olmayan hatalar:** Mobil robotun seyir halindeyken bulunduğu ve hareket ettiği zeminin düzensiz olması veya robot tekerlerinin belirsiz zamanlarda kaymasından kaynaklanan hatalardır. Sistematik olmayan hatalar aynı zamanda belirsiz ve kestirilemeyen hata türüdür. Buna rağmen yapılan bazı çalışmalar mobil robotta çok sayıda enkoder kullanılarak sistematik olmayan hataların azaltılabileceğini göstermiştir (Borenstein 1996). Fakat, çok sayıda kullanılan enkoder metodunun herhangi bir mobil robotta uygulanamaması ve özel olarak mobil robotun tasarlanması gerekmektedir.

Sistematik olan ve olmayan hatalara sebep olan etkenler Çizelge 3.1' de özetlenmiştir (Gaudin 2000).

Çizelge 3.1 Sistematik ve sistematik olmayan hataların karşılaştırılması

Sistematik hatalar	Sistematik olmayan hatalar
—Eşit olmayan teker çapları —Teker eksenlerinin kayıklığı —Tekerler arasındaki mesafenin belirsizliği —Enkoder çözünürlüğünün yetersiz olması —Enkoder örnekleme oranının sınırlı olması	—Hareket edilen zeminin sürekli değişmesi —Hareket güzergahında bilinmeyen objelerin üzerinden geçilmesi —Teker kaymaları

3.2.1.2 Odometre ve jiroskop ile navigasyon

Odometre ve jiroskop farklı özelliklerde iki sensör oldukları için, bu navigasyon modeli bir bakıma iki sensörün bütünleştirmesi olarak görülebilir. Odometre ve jiroskop ile mobil robotun konumu aşağıdaki verilen temel parakete denklemler yardımı ile bulunur;

$$x_n = x_0 + \sum_{i=0}^{n-1} v_{ort_i} \cos\theta_{FOJ_i} \Delta t \tag{3.21}$$

$$y_n = y_0 + \sum_{i=0}^{n-1} v_{ort_i} \sin\theta_{FOJ_i} \Delta t \tag{3.22}$$

$$\phi_{FOJ_n} = \sum_{i=0}^{n-1} \omega_i \tag{3.23}$$

Burada, ϕ_{FOJ} radyan cinsinden jiroskop ile elde edilen yön açısı, ω jiroskopun ürettiği açısal hız bilgisidir.

3.2.1.3 Ataletsel navigasyon sistemi

Ataletsel navigasyon sisteminde (INS-Inertial Navigation System) jiroskop ve ivmeölçerler kullanılır. Jiroskop ile dönme hızı, ivmeölçer ile ataletsel koordinat sisteminde aracın (x,y) eksenlerinde ivmesi ölçülür. Bu sensörlerle ölçülen değerler vektörel büyüklüklerdir. INS' lerde bir adet ivmeölçer ile bir jiroskop kullanılabildiği gibi, üç boyutlu bir koordinat sisteminde seyir eden bir mobil robotta bu sensörlerden her eksen için birer tane kullanılır.

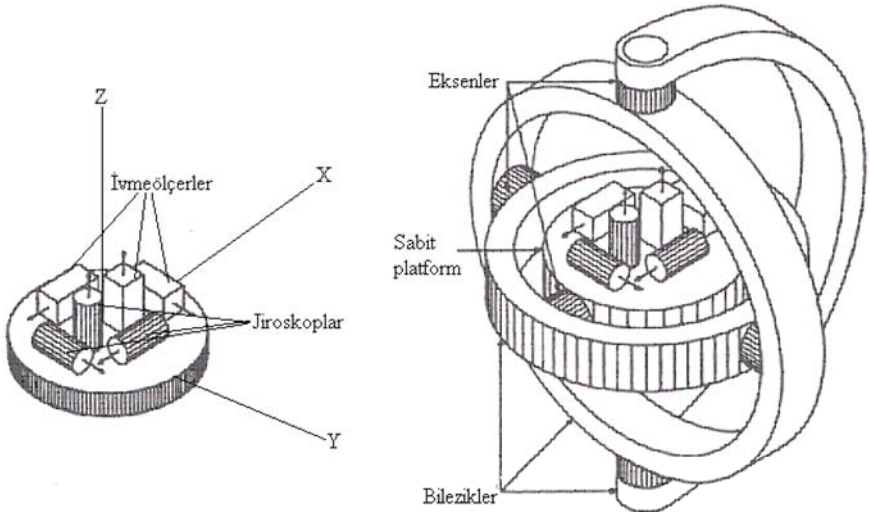
İvmeölçerlerden ölçülen ivme büyüklüklerinin bir kez integrali alınarak hız, iki kez integral alınarak da bir referans noktasına göre kat edilen mesafe bulunabilir. Jiroskoptan ölçülen dönme hızının bir kez integrali alınarak dönme miktarı hesaplanabilir. Ataletsel sensörlerin bir platformda toplanmış haline ataletsel ölçüm birimi (IMU-Inertial Measurment Unit) denir. IMU genelde, sabit bir gövdeye ve aracın (x,y,z) dikey eksenleri yönünde monte edilmiş üç ivmeölçer ve üç jiroskoptan oluşur. Parakete hesabında olduğu gibi ivmeölçerler ve jiroskoplar kullanılarak iki veya üç boyunlu bir koordinat sisteminde bir mobil robotun konumu, hızı ve yönü

bulunabilir. İntegral hesaplamalarından kaynaklanan sınırsız hata artışı, bu navigasyon sisteminin en belirgin hata kaynağıdır. İvmeölçer sensörlerinde oluşabilecek hatalardan biri, mobil robotun düzensiz zeminde seyrederken ivmeölçer tarafından yerçekimi ivmesini de algılaması ve bunun integral hesabına katılmasıdır.

Kullanılan birçok karakteristik performanslarda, değişik ataletsel navigasyon tasarımları vardır. Bunlar Şekil 3.11' de görüldüğü gibi aşağıdaki gibi iki farklı yapıya ayrılırlar;

i) Gövdeye bağlı: Şekil 3.11.a' da görüldüğü gibi mobil robotun gövdesine doğrudan monte edilir ve mobil robotun bütün hareketini takip eder. Bu tür IMU' ler kullanılan mobil robotların hız, konum ve yön bilgilerini ataletsel koordinat sisteminde elde etmek için dönüşüm hesaplarının yapılması gerekir. Gelişen mikroişlemci teknolojisine paralel olarak bu tür IMU' lar yaygınlaşmaya başlamıştır (Grewal ve ark.2001)

ii) Jirokararlaştırılmış platform: Şekil 3.11.b' de görüldüğü gibi bir kafese yerleştirilmiş olan IMU sistemi, belirlenen ilk yönünü sürekli koruyacak şekilde tasarlanmıştır. Bunun için kafesi oluşturan bileziklerin eksenlerine hassas kontrollü servo motorlar ve manyetik sargılar (konum değişimini algılamak için) yerleştirilmiştir. IMU sistemindeki jiroskoplardan alınan açı değişimleri, geri besleme olarak motor sürücülerine girilir ve oluşan açısal değişim kadar bilezikler hareket ettirilir. Böylece mobil robotun yönü değişse de IMU sistemi için belirlenen ilk durumu koruyacaktır.



(a) (b)

Şekil 3.11 (a) Gövdeye bağlı , (b) jirokararlaştırılmış IMU sistemleri

3.2.2 Mutlak navigasyon sistemleri

Mutlak navigasyon, mobil robotun bulunduğu çevrede genelde fark edilebilir, yerleri bilinen, doğal veya yapay elemanların kullanımına dayanır. Mobil robotun çevresine bağlı ve global referans sistemi içerisinde konumu tespit edilir. Mutlak navigasyon yönteminin avantajları, bağıl navigasyonda da olduğu gibi, navigasyon bilgileri hesaplanırken integral ve türevsel işlemlere ihtiyaç duyulmamasıdır. Fakat, mutlak navigasyon yönteminin temelini oluşturan fark edilebilir elemanların mobil robotun görüş alanında bulunması gerekir. Bu durum, çevre ve arazi şartlarından dolayı her zaman mümkün olmayabilir.

Mutlak navigasyon yöntemini verimli kullanabilmek için, mobil robotun bulunduğu çevrenin uygun bir şekilde düzenlenmesi gerekebilir. Genelde mutlak navigasyon sensörleri, bağıl navigasyon sensörlerinden daha ağır ve pahalıdır.

Mutlak navigasyon yöntemi üç ana başlık altında incelenebilir;

i) Yapay işaretler (ultrasonik, kızıl ötesi, ses gibi) ve uydular kullanarak konumlandırma yapmak
ii) Çevrenin haritasını kullanarak konumlandırma yapmak
iii) Çevrede bulunan karakteristik özellikleri fark edilebilir doğal işaretleri kullanarak konumlandırma yapmak

3.2.2.1 GPS ile Navigasyon

Son teknoloji ürünü olan GPS konum belirleme amacıyla, uydulara dayalı yapay referans noktaları oluşturur. 1960’ lı yıllarda, uydu teknolojisinin hızlı bir şekilde gelişmesiyle GPS sistemlerinin tasarımına adım atılmıştır. 1964’ den beri tüm dünya çapında GPS sistemleri kullanıcılara hizmet etmektedir. 1970 ‘ten sonra Navstar/GPS projelendirilmesine hız kazandırılmıştır.

GPS, daha tasarım aşamasında askeri kullanıcılar göz önünde tutularak planlanmıştır ve Amerikan ordusunun yönetimi ve sevki düşünülmüştür. Fakat GPS, birçok değişik bilim dallarında kolay çözümler getirmesinden dolayı sivil kullanıcılar da GPS’den nasibini almışlardır. Bu büyük proje ABD Savunma Bakanlığının denetiminde ve finansmanında gerçekleştirilmiştir. GPS’in tasarım felsefesinde,

dünyanın neresinde olunursa olunsun 24 saat belli bir duyarlılıkta bulunulan yerin konumunun ölçmesi hedeflenmiştir. Sivil kullanıcılar için GPS radyo sinyalleriyle belirlenen konumun hassasiyeti ve doğruluğu özel bozucu bir kodlama yöntemiyle azaltılmıştır. Bu hassasiyet azaltma işlemi seçimli yararlanabilirlik (SA-Selective Availability) olarak adlandırılmıştır. SA, tamamen ABD savunma bakanlığının denetimindedir ve istediği gibi değişikler yapma hakkını elinde tutmaktadır. Fakat 2000 yılında Amerikan başkanının aldığı bir kararla GPS sinyallerinden SA kaldırılmıştır.

Uzay kısmı: GPS uydularının yörüngeleri daireye yakındır ve yeryüzünden yükseklikleri yaklaşık 20200 km kadardır. Dünya etrafındaki GPS uydu sayısı 24 adet olarak planlanmıştır ve her biri ekvatora 63 derecelik açı ile yerleştirilmiştir. Daha sonraları finansman sorunlarından dolayı uydu sayısı 18'e düşürülmek istenmesine rağmen, dünya üzerinde 18 uydunun istenilen 24 saatlik kapsama alanı sağlamamasından dolayı bu fikirden vazgeçilmiştir. Bugün sistem, çalışmakta olan 24 uydudan oluşmaktadır ve altı farklı yörüngeye yerleştirilmiştir. Her bir yörüngede üzerinde dört uydu bulunmaktadır. Uzay kısmı küresel kapsama alanı gün içinde her hangi bir saatte 15 derecelik yüksekliğin üzerinde dört ile 8 uydu aynı anda görülmektedir. Eğer yükseklik maskesi 10 derecenin altında ise 10 uydu 5 derecenin altında ise 12 uydu gözlenebilmektedir.

Uydular: GPS uyduları radyo vericileri, atom saatleri, bilgisayar ve sistemin çalışmasında yardımcı olacak diğer donanımlardan oluşmaktadır. Her bir uydunun elektronik donanımı, kullanıcının uyduya kadar olan bir sahte mesafe (pseudorange) ölçmesini sağlamaktadır ve her bir uydu, kullanıcının rasgele bir zaman diliminde uydunun konumunu bildiren bilgileri yaymaktadır. Bu veriler çerçevesinde kullanıcı kendi konumunu bu sinyallerin kesişimi ile tespit edebilmektedir. Uydunun diğer donanımları ise; güneş panelleri, itme sistemleri ve elektronik kontrol birimleridir. Her bir uydunun kendini tanıtmada kullandığı değişik sistemler mevcuttur. Bunlar; uzaya fırlatma sırası, sözde rasgele sayılar (PRN-Pseudo Random Number), yörünge konum sayısı, NASA katalog kodu. Karmaşayı önlemek için PRN kod kullanılmasına karar verilmiştir.

GPS kullanıcıları:

Askeri Kullanıcılar: GPS'in devreye girdiği ilk yıllarda ana savunma sistemlerinin hepsine GPS alıcısı yerleştirilmesi planlanmıştı. Her uçak, gemi, kara araçları ve piyade grupların uygun GPS alıcısı ile donatarak, askeri harekatların gözlenmesi istenmiştir. Birçok GPS alıcısı 1991 körfez harekatı esnasında kullanılmıştır. Bu harekatta SA kapatılmış ve askerler ucuz sivil GPS alıcıları ile donatılarak çölde konumlarını belli bir doğrulukta belirlemişlerdir.

Sivil Kullanıcılar: GPS' in sivil kullanıcılar tarafından kullanımı, GPS sistemi planlayıcıları tarafından beklenmedik bir şekilde geliştirilmiştir. Sistemin geliştirilme amacı daha çok askeri araçların ve mühimmatların navigasyonu doğrultusundaydı. Bilim adamlarının farklı alanlarda kullanım sahaları oluşturmaları ve geçmişte büyük emeklerle yapılan konum ölçümleri GPS ile daha kolay ve ucuza yapılabiliyordu. Bilim dünyasının bu konuya çok hızlı girmesi ve bu konuda ölçüm metodolojileri geliştirmeleri ve özel sektörü de peşlerinden sürüklemeleri GPS'in sivil alana taşınmasında büyük rol oynamıştır. Gün geçtikçe sivil kullanıcı gruplarının sayısı artmaktadır. Artık GPS günlük hayatımızın her alanına girmeye başlamıştır.

GPS'in sivil kullanım alanları olarak: gemi filolarının denetimi ve kontrolü, kentlerde acil durum araçlarının GPS ile organize edilmesi, gerçek zamanda TIR filosu denetimi, trenlerde, şehir içi ve şehirlerarası otobüslerde, otomobillerde GPS ve şehir haritalarının yüklendiği bilgisayar sistemleri ile günlük yolculuklarda kullanımı, tarımsal otomasyonda ve hassas tarım uygulamalında GPS' in yer bulması gibi uygulamalar sayılabilir ve bunlara her gün bir yenisi eklenmektedir

Uydu sinyalleri:

Mevcut uydu taşıyıcı yayını, frekans karmaşası dışında ya da kasıtlı frekans etkileşimlerine duyarlılığı düşük olacak şekilde sinyalin spektrum dağılımı değiştirilmiştir. Spektrum dağılım tekniği günümüzde sıkça kullanılan bir yöntemdir. Sistemin hassasiyeti ve doğruluğunun altında yatan sır ise radyo sinyallerinin bileşenlerinin atom saatleri tarafından doğrudan ya da dolaylı olarak üretilmesidir. Block II uydularında 2 rubidyum ve 2 sezyum saat olmak üzere dört adet atom saati

bulunmaktadır. Bu saatlerin frekans kararlılıkları ise bir gün içinde 10^{-13} ve 10^{-14} saniye mertebesindedir. Block IIR uyduları için tasarlanan hidrojen MASER' lerin (Microwave Amplification by Stimulated Emission of Radiation)' lerin bir gün içindeki kararlılığı 10^{-14} ile 10^{-15} saniye mertebesindedir.

GPS uyduların kalbini, L bandındaki yüksek doğruluklu, temel frekansı 10.23 MHz olan osilatör oluşturur. Bu temel frekanstan üretilen L1 ve L2 taşıyıcı dalgaları sırasıyla temel frekansın 154 ve 120 tam katlarından oluşurlar.

L1=1575.42 MHz

L2=1227.60 MHz

İyonosferik kırılmalardan kaynaklanan hataların giderilmesinde L1 ve L2 frekansları aynı anda kullanılır

İki temel taşıyıcıyı dalgalarını modüle eden PRN kodları her bir uydu ve alıcı arasındaki seyir zamanını ölçmede kullanılır ve buradan sahte mesafeler hesaplanır.

İlki C/A kodudur. C/A kodu (Coarse/ Acquisition code) sivillerin kullanımına açık olan koddur. Standart konumlandırma servisi SPS (Standard Positioning Servis) olarak da adlandırılır. C/A kodunun yaklaşık dalga boyu 300 metre kadardır. C/A kodu L1 taşıyıcısını modüle eder, L2 ise bu modülasyon işleminin dışında tutulur. L2 taşıyıcısın C/A kodu ile modüle edilmemesinin sebebi ise Birleşmiş Programlar Ofisinin sivil kullanıcılar üzerindeki kontrolü elinde tutmak istemesidir. İkinci kod ise P kodudur (Precision Code). Bu kodun sınırsız kullanımı sadece izinli kullanıcılara verilmektedir. P kod duyarlı konumlandırma servisi PPS (Precision Positioning Servis) olarak da adlandırılır. PPS' in dalga boyu 30 m kadardır. P kodu L1 ve L2 taşıyıcılarını aynı anda modülasyona tabi tutar.

Seçimli yararlanabilirlik: C/A kodunun konum bulmadaki doğruluğunun, tasarımcıları tarafından 400 m olması bekleniyordu. Fakat ilk saha testlerinde konumlandırma doğruluğunun 14 ile 40 m aralığında olduğu gözlenmiştir. Bu beklenen bir sonuç değildi ve bu doğruluğun 40 m üzerine çıkarılması, C/A kod sinyalinde rasgele saat gecikmeleri ve GPS uydu koordinat parametrelerine eklenen rasgele gürültülerle gerçekleştirilmiştir. SA' nın amacı oluşabilecek potansiyel düşmanlara karşı oluşturulan bir doğruluk azaltması işlemidir. SA sadece Block II

uydularında mevcuttur. SA aktif iken elde edilen konumlandırma doğruluğu, yatay düzlemde 100 ile 150 m, yükseklikte 156 m olarak ölçülmüştür. GPS alıcısı hareket halindeyken hızın da ölçülmesine imkan tanımaktadır. SA' nın hız üzerindeki hata etkisinin 0.3m/s mertebelerinde olduğu gözlenmiştir.

Yanıltıcı şifreleme (Anti Spoofing): P-kodu şifrelenmiştir ve şifresi gizli tutulmaktadır. Bu yanıltıcı şifrelemeye (AS-Anti Spoofing) denir ve askeri GPS alıcılarının yanıltıcı GPS sinyallerinden korumak için uygulanmaktadır. Sivil GPS alıcılar da yanıltıcı GPS sinyallerini almaya açıktır, fakat sadece askeri amaçlarla kullanılmaktadır.

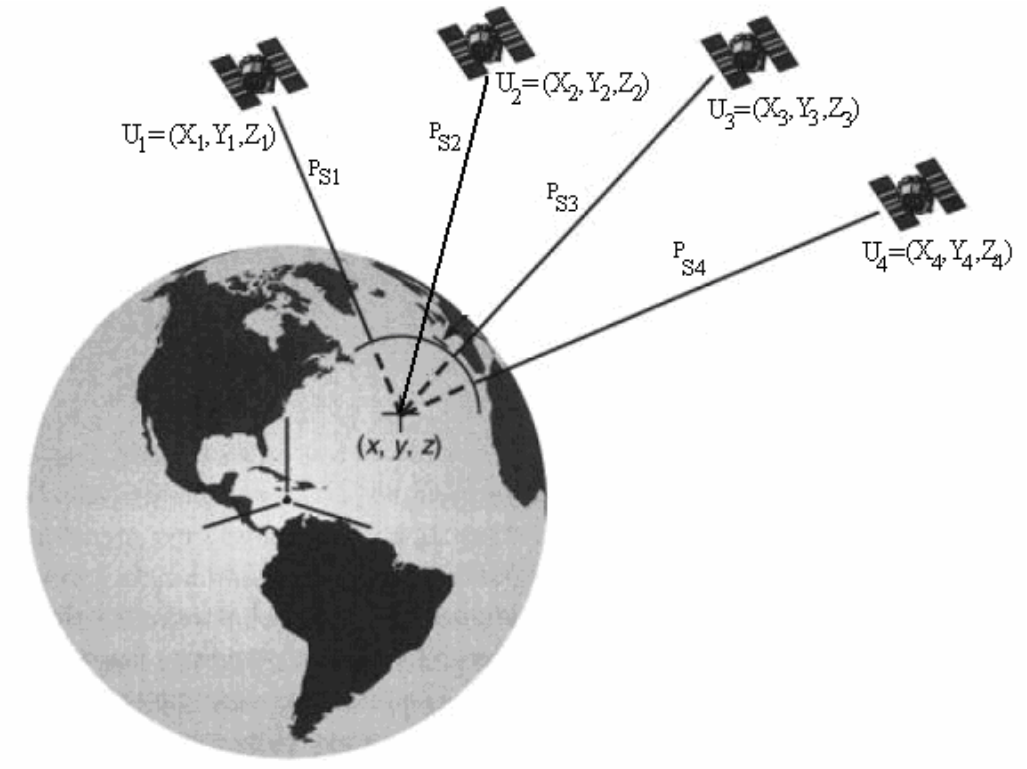
C/A-kod sahte mesafe alıcıları: Bu tip alıcılar ile, C/A kodu kullanılarak sahte mesafe ölçülür. Alıcı genelde elde taşınabilir veya modül şeklinde olur ve doğru akım kaynağı ile çalışır. Cihazın tipik özellikleri ise; bağımsız 12 farklı kanalı bulunur, yani aynı anda 12 GPS uydusundan sinyali takip edebilir. Çıktı olarak GPS, üç boyutlu konum bilgileri olan; enlem, boylam ve yüksekliklerini hesaplayarak bir dijital göstergeye yansıtır veya belirlenen dijital bir çıkışa yönlendirir.

C/A-kod taşıyıcı alıcıları: GPS sisteminin ilk zamanlarında birçok alıcı L1 taşıyıcısına kilitlenmek için C/A kodunu kullanmaktaydı. Birçok cihaz, minimum dört bağımsız kanala sahipti. Fakat yeni tür alıcılar, on iki bağımsız kanala sahiptir yani aynı anda on iki uyduyu izleyebilmektedirler. Yeni model GPS alıcılar, eski alıcılara göre daha fazla bellek ve veri depolama ünitelerine sahipler, aynı zamanda L2 taşıyıcısının fazını kodsuz faz ölçme teknikleri ile ölçebilecek şekilde geliştirilmişlerdir. Kodsuz ölçme tekniklerini kullanan alıcılar kinematik statik konum ölçümleri için en ideal olanlarıdır.

P-kod alıcılar: P-kod alıcıları P-kod' unu kullanarak L1 ve L2 taşıyıcılarına kilitlenir ve taşıyıcı fazlarını ölçebilirler. P-kod alıcıları, özel şifre çözücüler sayesinde hassas konumlandırma yapabilirler fakat bu şifre çözücüler sadece askeri kullanıcılara ve NATO müttefik güçlerine verilmektedir.

3.2.2.2 GPS ile konumlandırma

GPS ile üç boyutta konum belirlenebilmesi için dört uyduya ihtiyaç duyulmaktadır. En az dört GPS uydu ile konumlandırmayı gösteren yöntem Şekil 3.12'de verilmiştir. Her bir uydu "1" ve "0" 'lardan oluşan dijital PRN kodları yayınlarlar. GPS alıcısı bu PRN kodun bir kopyasını oluşturur. Uydudan gelen ve alıcıda üretilen sinyaller arasındaki faz farkının kıyaslanması, sinyalin iletimi ile alınması arasındaki iletim gecikme zamanı ΔT" yi verir. ΔT ile ışık hızının (c) çarpımı uyduya olan mesafeyi belirler. Bu mesafeye sahte mesafe adı verilir. Karşılaştırılacak PRN kodunun üretildiği GPS tarafındaki saat hatalıdır. Bu hata τ olarak adlandırılır. Bu sahte mesafe, aynı zamanda GPS uydu sinyallerinin çevreden yansıyarak alıcıya gecikmeli olarak ulaşması, iyonosfer gecikmeleri ve uyduya ait saat sapmaları gibi hataları da içermektedir.



Şekil 3.12 Dört uydu ile yeryüzünde konum bulma yöntemi

P uyduya olan gerçek mesafe, T alıcının sinyali alma zamanı, T_S sinyalin uydudan ayrılma zamanı, t ölçümün yapıldığı ve t_S sinyalin uydudan ayrıldığı gerçek zaman olsun.

Sahte mesafe uydu ile alıcı arasındaki zaman farkı ve ışık hızına bağlı olarak aşağıdaki gibi verilir,

$$P_S = \Delta T.c \tag{3.24}$$

Burada Δt, sinyalin uydudan alıcıya ulaşma zamanıdır ve

$$\Delta T = (T - T_S) \tag{3.25}$$

olarak ifade edilir. Burada T ve T_s zamanları;

$$\begin{aligned} T &= t + \tau \\ T_S &= t_S + \tau_S \end{aligned} \tag{3.26}$$

denklemleriyle tanımlanır.

(3.25) ve (3.26) denklemleri kullanılarak (3.24) denklemi yeniden düzenlenirse sahte mesafe;

$$P_S = (t - t_S).c + (\tau - \tau_S).c \tag{3.27}$$

olarak elde edilir. Burada;

$$P = (t - t_S).c \tag{3.28}$$

gerçek mesafeyi verir. Böylece sahte mesafe;

$$P_S = P + \tau.c - \tau_S.c \tag{3.29}$$

şeklinde ifade edilir. Buradaki τ' lar GPS alıcısına ve τ_s' ler uydulara ait gecikmelerdir.

WGS–84' e göre dört uydunun üç boyutlu koordinatları: $U_1=(X_1,Y_1,Z_1)$, $U_2=(X_2,Y_2,Z_2)$, $U_3=(X_3,Y_3,Z_3)$, $U_4=(X_4,Y_4,Z_4)$ ve GPS alıcı koordinatları (X,Y,Z) olsun. GPS ile uydular arasındaki gerçek mesafeler aşağıdaki denklemlerle hesaplanır;

$$P_1 = \sqrt{(X_1 - X)^2 + (Y_1 - Y)^2 + (Z_1 - Z)^2}$$

$$P_2 = \sqrt{(X_2 - X)^2 + (Y_2 - Y)^2 + (Z_2 - Z)^2}$$

$$P_3 = \sqrt{(X_3 - X)^2 + (Y_3 - Y)^2 + (Z_3 - Z)^2} \qquad (3.30)$$

$$P_4 = \sqrt{(X_4 - X)^2 + (Y_4 - Y)^2 + (Z_4 - Z)^2}$$

(3.30)'daki gerçek mesafeler (3.29)' daki denklemde yerine yazılır ve düzenlenirse;

$$\sqrt{(X_1 - X)^2 + (Y_1 - Y)^2 + (Z_1 - Z)^2} = P_{S1} + (\tau_{S1} - \tau).c$$

$$\sqrt{(X_2 - X)^2 + (Y_2 - Y)^2 + (Z_2 - Z)^2} = P_{S2} + (\tau_{S2} - \tau).c \qquad (3.31)$$

$$\sqrt{(X_3 - X)^2 + (Y_3 - Y)^2 + (Z_3 - Z)^2} = P_{S3} + (\tau_{S3} - \tau).c$$

$$\sqrt{(X_4 - X)^2 + (Y_4 - Y)^2 + (Z_4 - Z)^2} = P_{S4} + (\tau_{S4} - \tau).c$$

şeklinde olur.

Denklem (3.31)'de P_S'ler ölçüldükten sonra c ve uydu koordinatları bilindiğinde geriye dört bilinmeyen kalır. Burada $(\tau_S - \tau)$ 'lar tek bir değişken olarak kabul edilir. Denklem (3.31), nonlineer dört bilinmeyenli denklem takımıdır ve nümerik metotlarla çözülür. Bu denklem takımı çözüldükten sonra (X,Y,Z) alıcı koordinat bilgileri elde edilir. Burada hesaplanan koordinatlar C/A kodu kullanılarak yapılan en temel konum hesaplama yöntemidir ve bu çalışmada da kullanılan GPS, bu yöntemi kullanarak konum hesaplamaktadır. Bu yöntemle hesaplanan koordinatların doğruluğu ±20m aralığındadır.

3.2.3 Hibrid navigasyon Sistemleri

Hibrid navigasyon sistemleri birden fazla sensörün bütünleştirilmesi ile gerçekleştirilir. Bütünleştirilecek sensörlerin tamamı ataletsel olabileceği gibi, bağıl sensörler ile mutlak sensörlerin bütünleştirilmesi ile de gerçekleştirilir. Borenstein

(1996) çalışmasında odometre ile jiroskopu bütünleştirerek odometredeki yön hesaplamalarını en aza indirmeye çalışmıştır ve uyguladığı bu yönteme (Gyrodometry) adını vermiştir. Bağıl sensörlerle yapılan bütünleştirilmiş sistemlerde mobil robotun konumu, hızı ve yönü sürekli hesaplanabilmektedir. Böylece hesaplamalardaki hatalar artacak ve belli bir zaman sonra navigasyon bilgileri anlamsız olacaktır. Bu hataları azaltmak için mutlak bir sensörün desteğine ihtiyaç vardır. Bağıl sensörlerdeki hataların artması, belli bir eşik seviyesinden sonra mutlak sensörün bilgileri ile düzeltilir ve tekrar bağıl sensörlerle navigasyona devam edilir. Bu iki sensör grubunun avantaj ve dezavantajları Çizelge 3.2' de özetlenmiştir.

Mobil robotlarda, bütünleştirilmiş navigasyon sistemleri tasarımında, bağıl sensörlerin bütünleştirilmesi genel navigasyon sisteminin bir alt birimini oluşturur. Bunun için navigasyon sistemi mutlak ve bağıl sensörler olmak üzere iki kısımdan oluşmaktadır. Kapalı mekanlarda mutlak sensörler; kamera, koordinatları bilinen kodlanmış işaretler, kapalı mekanlar için tasarlanmış GPS sistemleri, lazer ölçüm sistemleri, ultrasonik ve sayısal haritalar yardımıyla melez sistemler gerçekleştirilir. Kapalı mekanlarda kullanılan mutlak konumlandırma sistemleri, mobil robotun üzerine monte edilir ve bu mekan içinde koordinatları bilinen pasif işaretlerin, aktif işaretlerin veya sinyallerin algılanmasına bağlı olarak mobil robotun konumu, yön bilgileri ölçülerek hesaplanır. Açık alanlarda mutlak sensör olarak genelde GPS kullanılır. Çizelge 3.2'den de görüldüğü gibi bağıl konumlandırma sensörleri ile mutlak konumlandırma sensörleri birbirlerini tamamlayan özeliktedirler.

Çizelge 3.2 Bağıl konumlandırma ile mutlak konumlandırmanın karşılaştırılması

Bağıl Konumlandırma	Mutlak Konumlandırma
(+) Çevreye karşı otonom olma	(-) Çevreye bağlı olması ve çevrenin özel olarak düzenlenmesi gerekir
(+) Sürekli ve yüksek frekansta çalışma	(-) Süreksiz ve düşük frekansta çalışma
(-) Kat edilen mesafe ve geçen zaman ile hataların birikmesi	(+) Hata, mesafe ve zamana bağlı değil
(-) Aracın konumunun ilk konum değerlerine ve referans noktasına bağlı olması	(+) İlk durum değerlerine ve referans noktasına ihtiyaç olmaması

Bu iki sensör grubunun avantajları kullanılarak daha kararlı navigasyon sitemleri tasarlanabilir. Bağıl konum sensörleri bir önceki ölçümlerine bağlı olarak navigasyon bilgileri üretirler, fakat mutlak konum sensörleri o an ölçtüklerine bağlıdırlar. Bundan dolayı mutlak konum sensörlerinde hata belli sınırlar içinde kalmaktadır.

Önceki bölümlerde de bahsedildiği gibi hibrid navigasyon sistemlerinin en büyük özelliği mutlak veya bağıl sensörlerden birinin herhangi bir nedenden dolayı veri üretmemesi durumunda navigasyona sağlam kalan sensörlerin yardımıyla devam edilebilmesidir. Burada temel amaç iki farklı sensör tipini birleştirip en optimal hata ile kesintisiz navigasyon yapmaktır.

3.2.3.1 Kalman filtresi ile mobil robot navigasyonu

Kalman filtresi navigasyon sistemlerinde ve sensör bütünleştirme problemlerinde geniş şekilde uygulanmaktadır. Kalman filtresi ilk olarak 1960 yılında R. Kalman tarafından Apollo uzay programı çerçevesinde geliştirilmiştir ve bugüne kadar bu filtre algoritması kendine binlerce uygulamada yer bulmuştur. (Kalman R.E. 1960) (Simon 2001). Kalman filtresi lineer modellerde kullanılarak başarılı optimal çözümler elde edilmiştir. Fakat gerçek hayatta sistemlerin birçoğu lineer değildir. Nonlineer sistemler için genişletilmiş Kalman filtresi geliştirilmiştir (Welsh ve Bishop 2001). Kalman filtresi, lineer sistemlerde optimal çözümü garanti eder. Buna karşılık nonlineer sistemler için kullanılan genişletilmiş Kalman filtresi nonlineer sistemlerde optimal çözümü garanti etmez (Brown ve Hwang 1999).

Kalman filtresi bilinen bir algoritma olarak klasik çözüm yöntemleri arasında yerini almıştır. Araştırılan ve geliştirilen diğer algoritmalara ve yöntemlere referans olmuştur. Bu tez çalışmasında da geliştirilen YSA yöntemi Kalman filtresi sonuçları ile karşılaştırılmıştır. Kalman filtresi uygulama sahaları ve teorisi kaynaklarda belirtilen referanslardan elde edilebilir ve ayrıca burada ayrıntılara girilmemiştir

Kalman filtresi günümüze kadar yaygın olarak araştırılmış ve klasik navigasyon algoritmaları içinde yerini almıştır. Mobil robot navigasyonu alanında sıkça kullanılan yöntemlerden biridir. Mobil robot üzerinde mutlak ve bağıl

sensörlerin bütünleştirilmesinde ve bu bütünleştirmeden elde edilecek optimal mobil robot konumu için Kalman filtresi uygun algoritma olarak görülmektedir. Kalman filtresini verimli bir şekilde kullanmak için gerekli olan ilk basamak uygulanacak sistemin matematiksel modelinin elde edilmesidir. İkinci basamak ise filtreye uygulanacak ilk değerlerin ölçülmesi ve hesaplanmasıdır. Bu değer filtrenin sağlıklı çalışması için gereklidir.

a) Mobil robotun kinematik modeli

Bu model, mutlak konum bilgisi üreten GPS verileri mevcut olmadığı durumlarda, konumlandırma sisteminin performansını doğrudan etkiler ve kritik önem taşır. Çünkü mobil robot bu durumda sadece kinematik modele dayanarak konum bilgisini hesaplayacaktır. Mobil robotun kinematik modeli ayrık zaman formunda aşağıdaki denklemler ile ifade edilir.

$$x_k = \begin{bmatrix} x_{x,k} \\ x_{y,k} \\ x_{\phi,k} \end{bmatrix}, \quad w_k = \begin{bmatrix} w_{x,k} \\ w_{y,k} \\ w_{\phi,k} \end{bmatrix}, \quad v_k = \begin{bmatrix} v_{x,k} \\ v_{y,k} \\ v_{\phi,k} \end{bmatrix} \tag{3.32}$$

$$x_{k+1} = f(x_k, u_k) = \begin{bmatrix} f_x(x_k, u_k) \\ f_y(x_k, u_k) \\ f_\phi(x_k, u_k) \end{bmatrix} + w_k \tag{3.33}$$

Burada;

$$\begin{aligned} f_x(k) &= x(k+1) = x(k) + \Delta D(k).\cos(\phi(k-1) + \frac{\Delta\phi(k)}{2}) + w_x(k) \\ f_y(k) &= y(k+1) = y(k) + \Delta D(k).\sin(\phi(k-1) + \frac{\Delta\phi(k)}{2}) + w_y(k) \\ f_\phi(k) &= \phi(k+1) = \phi(k) + \Delta\phi(k) + w_\phi(k) \end{aligned} \tag{3.34}$$

denklemleri ile ifade edilir. Burada; w_x, w_y w_ϕ, sırasıyla (x,y) yönünde ve mobil robotun yön bilgisine eklenen gürültülerdir, $\Delta\phi$: örnekleme aralığındaki açı farkı, ΔD : örnekleme zaman aralığında kat edilen mesafe, yani hızdır. Ölçüm modeli ise;

$$z_k = h(x_k) = \begin{bmatrix} h_x(x_k) \\ h_y(x_k) \\ h_\phi(x_k) \end{bmatrix} \tag{3.35}$$

şeklinde olur.

b) Lineerleştirilmiş mobil robot modeli

Kalman filtresi, lineer stokastik sistemlerin durum değişkenlerini kestirmek için kullanılır. (3.34) denklemleri nonlineer yapıdadırlar. Bu modele doğrudan Kalman filtresi uygulanamaz. Bunun için genişletilmiş Kalman filtresine ihtiyaç vardır. Mobil robot sisteminin modeli, kestirilecek durum değişkenlerinin *k* anındaki değerlerinde lineerleştirilir. Denklem (3.36) nonlineer stokastik sistemin bağıntısıdır

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) + w_k \\ z_{k+1} &= h(x_k, u_k) + v_k \end{aligned} \tag{3.36}$$

Yukarıdaki ifade lineerleştirildikten sonra;

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k \\ z_{k+1} &= H.x(k) + v(k) \end{aligned} \tag{3.37}$$

haline dönüşür. Burada w_k ve v_k birbiri ile korele olmayan sistem ve ölçüm gürültü matrisidir ve *u(k)* sistem giriş matrisidir.

Sistem matrisi *A* ve girdi matrisi *B*' nin Jacobien'i (3.38) ve (3.39) denklemleri ile ifade edilir

$$A = \begin{bmatrix} \frac{\partial fx}{\partial x(k)} & \frac{\partial fx}{\partial y(k)} & \frac{\partial fx}{\partial \phi(k)} \\ \frac{\partial fy}{\partial x(k)} & \frac{\partial fy}{\partial y(k)} & \frac{\partial fy}{\partial \phi(k)} \\ \frac{\partial f\phi}{\partial x(k)} & \frac{\partial f\phi}{\partial y(k)} & \frac{\partial f\phi}{\partial \phi(k)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta D(k).\sin(\phi(k-1) + \frac{\Delta\phi(k-1)}{2}) \\ 0 & 1 & \Delta D(k).\cos(\phi(k-1) + \frac{\Delta\phi(k-1)}{2}) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.38}$$

$$B = \begin{bmatrix} \frac{\partial fx}{\partial \Delta D(k)} & \frac{\partial fx}{\partial \Delta\phi(k)} \\ \frac{\partial fy}{\partial \Delta D(k)} & \frac{\partial fy}{\partial \Delta\phi(k)} \\ \frac{\partial f\phi}{\partial \Delta D(k)} & \frac{\partial f\phi}{\partial \Delta\phi(k)} \end{bmatrix} = \begin{bmatrix} \cos(\phi(k) + \frac{\Delta\phi(k)}{2}) & -\frac{\Delta D(k)}{2}.\sin(\phi(k) + \frac{\Delta\phi(k)}{2}) \\ \sin(\phi(k) + \frac{\Delta\phi(k)}{2}) & \frac{\Delta D(k)}{2}.\cos(\phi(k) + \frac{\Delta\phi(k)}{2}) \\ 0 & 1 \end{bmatrix} \tag{3.39}$$

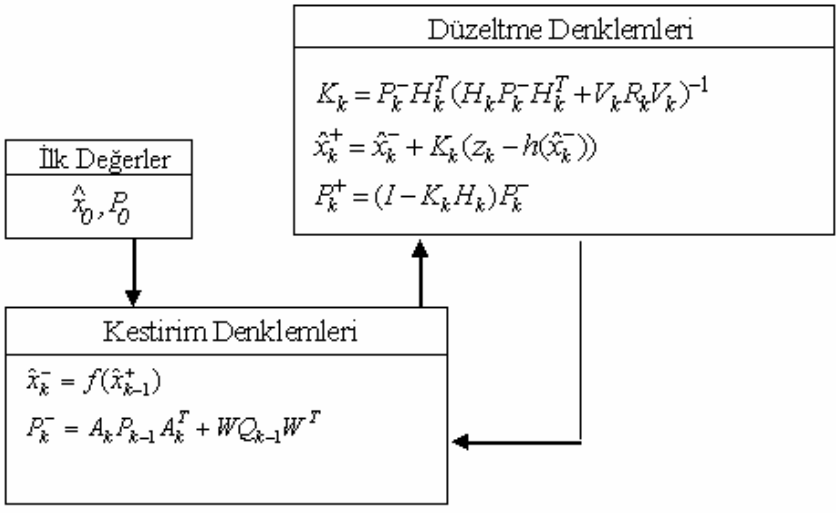
z_{k+1} durum ölçüm vektörünü x_k durum değişkenleri vektörüne bağlayan *H* matrisinin Jacobien'i denklem (3.40) 'daki gibidir

$$H = \begin{bmatrix} \frac{\partial hx}{\partial hx(k)} & \frac{\partial hx}{\partial hy(k)} & \frac{\partial hx}{\partial h\phi(k)} \\ \frac{\partial hy}{\partial hx(k)} & \frac{\partial hy}{\partial hy(k)} & \frac{\partial hy}{\partial h\phi(k)} \\ \frac{\partial h\phi}{\partial hx(k)} & \frac{\partial h\phi}{\partial hy(k)} & \frac{\partial h\phi}{\partial h\phi(k)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.40}$$

z_{k+1} vektörü bütün x_k durum değişkenlerinin gözlenmesi ile elde edilir. Bu tür ölçüm modeli, tam durum algılayıcı ölçüm modeli olarak bilinir.

Yukarda hesaplanan parametreler Şekil 3.13'de verilmiş olan genişletilmiş Kalman filtresi algoritmasında yerine yerleştirildikten sonra filtre döngüsü hesaplanır. Eğer ölçüm verileri mevcut değilse Kalman filtresindeki düzeltme algoritması işletilmez ve model temelli kestirim denklemleri hesaplanır (Peyret ve ark. 2004).

Kalman filtresini başlatabilmek için x_0 ve P_0 değerlerine ihtiyaç vardır. Bu değerlerden; x_0, durum vektörü olup mobil robotun ölçülen ilk konum ve yön değeridir. P_0, kestirim hatası kovaryans matrisidir ve ilk değeri genellikle sıfır ya da sistem gürültü kovaryans matrisi Q' ya eşitlenir. Kalman filtresinde en önemli parametre Kalman kazancı *(K)'* dır. Bu katsayı modelin veya ölçümlerin güvenilirliğine göre her Kalman filtre döngüsünde yeniden hesaplanır. *K* aynı zamanda düzeltme denklemlerinde ne kadar düzeltme yapılacağını belirleyen ağırlık katsayısıdır. Ölçümlere güvenilecek ise *K* katsayısı büyük olur, değilse küçük değerde olur.



Şekil 3.13 Genişletilmiş Kalma filtresi algoritması

4. YAPAY SİNİR AĞLARI

Yapay sinir ağları (YSA) ile ilgili çalışmalar insan beynini modellemek ve nasıl çalıştığını anlamak amacıyla başlamıştır. İnsan beynini çalışma fonksiyonuna benzetme isteği YSA modellerine esin kaynağı olmuştur. Beyin, birbirine değişik etki seviyeleri ile bağlantılı binlerce nörondan oluşmaktadır. YSA modelleri de temel nöron modelleri oluşturmaktadır birbirine ağırlık bağlantıları ile bağlanır. YSA sonuçta bir bilgi işleme modeli olarak karşımıza çıkar.

YSA sistemlerine ilginin bu denli yoğun olmasının ve mühendislik uygulamalarında geniş bir yer bulmasının sebebi, klasik yöntemlerle çözülemeyen veya çözümü çok karmaşık olan problemlere başarıyla uygulanmış olması ve klasik metotlara bir alternatif oluşturmasıdır.

4.1 YSA' nın Tarihçesi

1940'lı yılların başında, nörofizikçi olan McCulloh ve matematikçi olan Pitts ilk olarak tek bir nöronun mantıksal işlemini sağlayan temel parametrik nonlineer işlem modelini geliştirmişlerdir (McCulloch ve ark. 1943). 1958'de Rosenblatt, çalışmalarında çok katmanlı perseptron ve geliştirdiği modelin parametrelerini ayarlayabilen bir algoritma geliştirmiştir. Aynı tarihte Widrow ve Hoff (Widrow ve ark.1960) ADALINE (Adaptive Linear Model) ve MADALINE (Multiple ADALINE) çoklu adaptif lineer nöron yapay sinir ağı modellerini geliştirerek ve ilk olarak YSA' yı sayısal işaret işleme, hava durumu tahmini ve adaptif kontrol problemlerini çözmek için kullanmışlardır. 1969 yılında Misky ve Papert "Perceptron" adlı kitaplarında, tek katmanlı perseptron ağının çok basit işlemleri çözemediğini göstererek (örneğin: XOR problemi) ve hesaplamalarda aşılması zor engeller olduğunu iddia etmiştir. Minsky ve Papert buna dayanarak YSA' ların ilgi çekici bir konu olmadığını vurgulamışlar ve YSA ile ilgili yapılan çalışmaları neredeyse durdurma noktasına getirmişlerdir (Minsky ve Papert 1969).

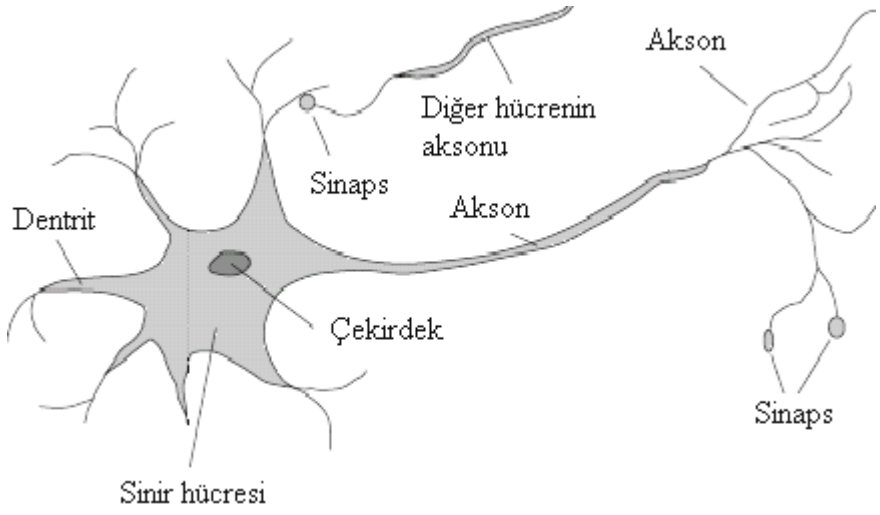
1980'li yıllarda bilgisayar teknolojisinin hızlı gelişmesi ve bireysel bilgisayarların artması sonucu YSA ile ilgili çalışmalara tekrar geri dönülmüştür ve bu alanda çok hızlı bir gelişme kaydedilmiştir. 1982'de Hopfield tarafından yapılan "Neural Networks and Physical systems" adlı çalışması ile modern YSA devri başlar

(Hopfield 1982). 1986'da Rumelhart, ileri beslemeli YSA modellerinde yeni öğrenme modeli olan hatanın geri yayılması algoritmasını geliştirmiştir. Bu algoritma ile daha önce yapılmış olan çalışmalarda ortaya çıkan sorunların çözülebileceğini göstermiştir (Rumelhart ve ark. 1986).

Günümüzde YSA teorisinin iyice bilinmesi ve tamamlanma aşamasına gelmesi sebebiyle, YSA ile ilgili uygulamalı çalışmalar bilgisayar sistemlerindeki gelişmelere paralel olarak büyük bir yoğunluk kazanmıştır.

4.2 Biyolojik Nöronun Yapısı

İnsan beyni, 10^{12} kadar nörondan oluşur ve her bir nöron yaklaşık 10000 farklı nöronla bağlantılıdır. Nöron, bir çekirdekten kendisine bağlı dentrit ve aksonlardan oluşur. Şekil 4.1'de biyolojik bir nöronun yapısı verilmiştir. Her nöron, dentritleri aracılığıyla diğer nöronlardan elektrokimyasal girişler alır. Bu girişlerin elektriksel toplamı belli bir eşik değerini aşarsa, nöron aktive olur ve aksonlarından çıkış sinyalleri üretir. Böylece aksonlara aktarılan elektrokimyasal sinyaller, bağlantı halinde olan diğer nöronların dentritlerine giriş olarak aktarılmış olur. Bütün beyin buna benzer birbiri ile bağlantılı nöronlardan oluşur ve bu nöronlar elektrokimyasal sinyaller yardımıyla iletişim halindedir. Biyolojik nöron basit bir işlem yapıyor gibi görünse de beyin, bu işlemler sayesinde oldukça karmaşık görünen problemleri kolaylıkla çözebilmektedir.



Şekil 4.1 Biyolojik nöronun yapısı

Biyolojik nöronun çalışma prensibi, yapay sinir ağları çalışma modeline temel oluşturmuştur. Fakat yapay sinir ağları modelinde yapılan sürekli araştırma ve geliştirmeler, beynin oluşturduğu karmaşık yapıyı modellemekten henüz çok uzaktır. Buna rağmen YSA' lardan, görüntü tanıma, geçmiş bilgilere dayanarak ileriye yönelik kestirim gibi problemlerde oldukça başarılı sonuçlar alınmıştır. Bu tür işlevler, insan beyninin günlük, sıradan ve sürekli çözdüğü fakat klasik bilgisayar mantığı çerçevesinde çözümü oldukça zor problemlerdir.

Literatürde 100'den fazla YSA modeli mevcuttur. Bu modelleri geliştiren araştırmacılar, insan beyninin; hatırlama, güçlü düşünme ve problem çözme yeteneklerini YSA modellerine aktararak bilgisayarlarda uygulamaya çalışmışlardır.

YSA' lar, herhangi bir problemin girdi ve çıktılar arasındaki ilişkiler lineer olsun veya olmasın öğrenerek çözümler üretebilme özelliğine sahiptirler.

4.3 YSA' nın Özellikleri

YSA, beynin bir işlevini yerine getirmek için geliştirilmiş paralel dağıtılmış işlem modelidir. Beyin hücrelerinde olduğu gibi YSA da yapay sinir hücrelerinin birbirine çeşitli şekillerde bağlanmasından oluşur ve katmanlar şeklinde düzenlenir. YSA' lar tek girişli, tek çıkışlı, çok girişli, çok çıkışlı yapılar da oluşturulabilir. Giriş ve çıkış sayıları, problemin karakteristiğine bağlı olarak belirlenir. YSA ile insan beyni fonksiyonları arasında bazı benzerlikler vardır. Bu benzerlikler Çizelge 4.1' de verilmiştir.

Çizelge 4.1 Biyolojik sinir sistemi ile YSA' nın benzerlikleri

Biyolojik sinir sistemi	Yapay sinir sistemi
Nöron	İşlem elemanı
Dendrit	Toplama fonksiyonu
Hücre gövdesi	Transfer fonksiyonu
Aksonlar	Eleman çıkışı
Sinapslar	Ağırlıklar

YSA' nın bahsedilen buna benzer özellikleri, karmaşık problemleri çözebilme yeteneğini göstermektedir. YSA, yapısında var olan dağıtılmış paralel yapıdan dolayı, hataya karşı toleranslı ve genelleme yeteneğine sahiptir. YSA' yı oluşturan temel işlem elemanı nonlineerdir. Bu nedenle temel işlem elemanlarından oluşan ağ yapısı da nonlineerdir. YSA' nın bu karakteristiği nonlineer karmaşık problemlerin çözümünde büyük başarı gösterir.

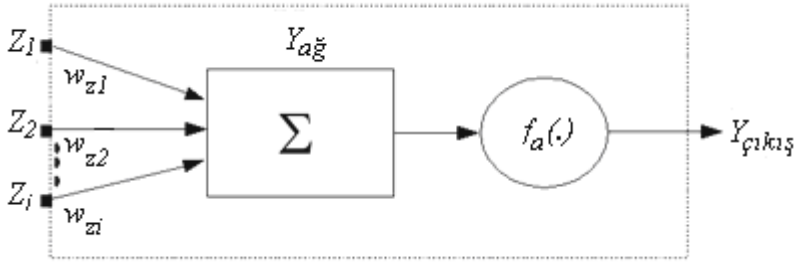
YSA' nın giriş bilgilerine göre çıkış bilgileri üretebilmesi için işlem elemanları arasındaki bağlantı ağırlıklarının ayarlanması gerekir. Bu ayarlama, YSA' nın ilgili problemin giriş-çıkış verileri arasındaki ilişkiyi öğrenerek istenilen davranışı göstermesini sağlar.

YSA, ortam şartlarına göre davranışlarını değiştirebilir. Yani, problemde meydana gelecek değişiklikler karşısında YSA tekrar eğitilerek ağırlıklarını tekrar ayarlayabilir. Eğer problemin parametreleri sürekli değişir ise, YSA sürekli eğitilebilir. YSA' nın bu özelliği uyarlanabilir kontrol sitemlerinde etkin kullanım alanı bulmuştur (Simpson 1990).

Geleneksel hesaplama teknikleri ile yapılamayan işler, YSA ile de yapılamaz. Fakat YSA klasik metotlarla çok karmaşık ve zor çözülebilecek problemleri daha kolay çözebilir. YSA, özellikle giriş-çıkış ilişkileri karmaşık olan sistemlerde, sistemi öğrenerek model oluşturabilir. YSA' nın bu model oluşturma özelliği ile farklı sensör verilerinden oluşan sistemlerde, sürecin hiçbir parametresi bilinmemesine karşın YSA' ya gösterilecek örneklerle süreci tanımlayan karmaşık matematiksel ilişkiyi kolaylıkla öğrenebilir.

4.4 Yapay Sinir Hücresi

YSA' ları oluşturan en küçük bilgi işlem birimi, yapay sinir hücresidir. Şekil 4.2' de i adet girişi ve tek çıkışı olan bir yapay sinir hücresi verilmiştir. Şekil 4.2' deki yapay sinir hücresinde Z_i sinir girişlerini, w_{zi} girişler ile sinir hücresi arasındaki ağırlık katsayılarını ve $f_a(.)$ aktivasyon fonksiyonunu ifade etmektedir. Aktivasyon fonksiyonu, sinir hücresinin toplamdan sonraki çıkışını bir işlemden geçirdikten sonra hücre çıkışını $Y_{çıkış}$ oluşturur. Genellikle aktivasyon fonksiyonu nonlineer bir fonksiyondan oluşur (Karna ve ark. 1989)



Şekil 4.2 Yapay sinir hücre yapısı

Sinir girişlerinin toplamını ifade eden $Y_{ağ}$ fonksiyonu;

$$Y_{\text{ağ}} = \sum_{i=1}^{N} Z_i w_{zi} - \theta_i \tag{4.1}$$

şeklinde hesaplanır. Burada θ_i eşik değeridir. Sinir çıkışını oluşturan $Y_{çıkış}$ fonksiyonu ise,

$$Y_{çıkış} = f_a(Y_{ağ}) \tag{4. 2}$$

denklemi ile hesaplanabilir.

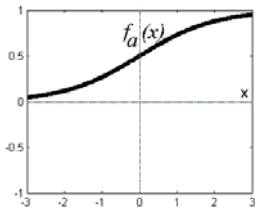
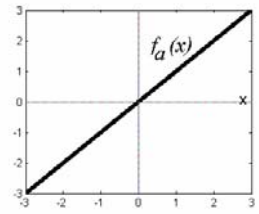
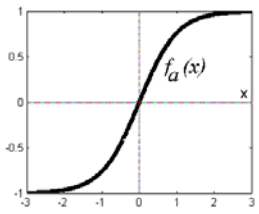
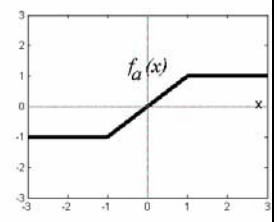
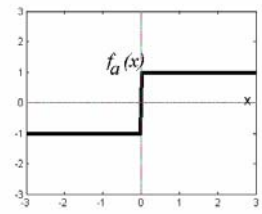
4.5 Aktivasyon Fonksiyonları

YSA, nonlineer aktivasyon fonksiyonları kullanmadan kompleks ve nonlineer problemleri çözemez. Aktivasyon fonksiyonları, girişlerin veya problemde istenilen çıkışların özelliklerine göre seçilebilir. Çizelge 4.2' de YSA' larda en sık kullanılan aktivasyon fonksiyonları verilmiştir. Eğer YSA, lineer aktivasyon fonksiyonu kullanıyorsa ağ, lineer işlem özellikleri gösterir ve lineer problemleri çözebilir.

Sigmoid fonksiyonu, sadece pozitif değerler üretir, tanjant hiperbolik fonksiyon ise hem pozitif hem de negatif değerler üretir. Genellikle çok katmanlı YSA'larda logaritmik sigmoid veya tanjant hiperbolik fonksiyonları kullanılmaktadır. Doyum ve keskin sınırlayıcı fonksiyonlar, mantıksal işlemler veya mantıksal sınıflandırma gerektiren problemlerde kullanılmaktadır. Hangi

problemlerde hangi fonksiyonun kullanılacağı, YSA'nın çözmeye çalışacağı problemin şartlarına bağlıdır ve etkin bir şekilde deneme yanılma yoluyla tespit edilebilir

Çizelge 4.2 Aktivasyon fonksiyonları

Aktivasyon fonksiyonları	Matematiksel yapı	Fonksiyon grafiği
Sigmoid *logsig(x)*	$f_a(\mathrm{x}) = \dfrac{1}{1+e^{-\mathrm{x}}}$	
Lineer *lin(x)*	$f_a(\mathrm{x}) = \mathrm{x}$	
Tanjant hiperbolik *tansig(x)*	$f_a(\mathrm{x}) = \dfrac{1-e^{-2\mathrm{x}}}{1+e^{2\mathrm{x}}}$	
Doyun fonksiyonu *sat(x)*	$f_a = \begin{cases} +1 & \mathrm{x} \geq 1 \\ \mathrm{x} & 1 \leq \mathrm{x} \leq +1 \\ -1 & \mathrm{x} \leq -1 \end{cases}$	
Keskin Sınırlayıcı *signum(x)*	$f_a(\mathrm{x}) = \begin{cases} +1 & \mathrm{x} \geq 0 \\ -1 & \mathrm{x} \leq 0 \end{cases}$	

4.6 YSA'ların Yapılarına Göre Sınıflandırılması

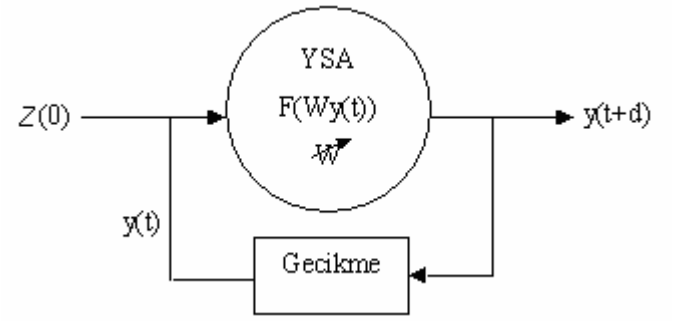
YSA' lar, içyapısındaki bağlantılar açısından incelendiğinde, ikiye ayrılabilir. Bunlar; ileri beslemeli ve geri beslemeli YSA' larıdır.

4.6.1 İleri beslemeli yapay sinir ağları

YSA' lar çok katmanlı işlemsel birimlerden oluşur. Dış ortamdan alınan veriler girişe uygulanır ve giriş katmanında işleme girer, bilgi akış yönünde hiç bir değişikliğe uğratılmadan orta katmanlara iletilir. Bu katmanlarda işleme giren bilgi ileri yönde çıkış katmanına iletilir. Bilginin girişten çıkışa doğru ileri yönde iletilmesi çalışma prensibine dayalı YSA' lara ileri beslemeli YSA denir.

4.6.2 Geri beslemeli yapay sinir ağları

İleri beslemeli YSA' ların tersine geri beslemeli YSA' nın, ağ yapısında bir veya daha fazla işlem elemanında geri besleme bağlantısı bulunur. Geri besleme işlemi, bilgiyi zaman gecikmesine uğratarak geri besler. Geri besleme bağlantıları işlem elemanları arasında olabileceği gibi katmanlar arasında da olabilir. Bir katmanın çıkışı, bir önceki katmanın girişine veya ağın çıkış katmanı, giriş katmanına geri beslenir. Bu yapısı ile geri beslemeli YSA' nın anlık çıkışı, hem o anki hem de önceki girişlerin etkisini yansıtır. Geri beslemeli YSA' lar, zaman serileri analizlerinde, tahmin problemlerinde sıkça kullanılan bir ağ modelidir ve geri besleme bağlantı yapısından kaynaklanan kararlılık sorunları üretebilir. Geri beslemeli YSA' lar, ileri beslemeli ağlara göre daha yavaş işlerler ve çalıştıkları donanımlarda daha fazla bellek gerektirirler. En çok bilinen tipleri Hopfield ağı, hücresel yapay sinir ağı, SOM, Elman ve Jordan ağlarıdır. Şekil 4.3' de basit bir geri beslemeli YSA modeli verilmiştir.



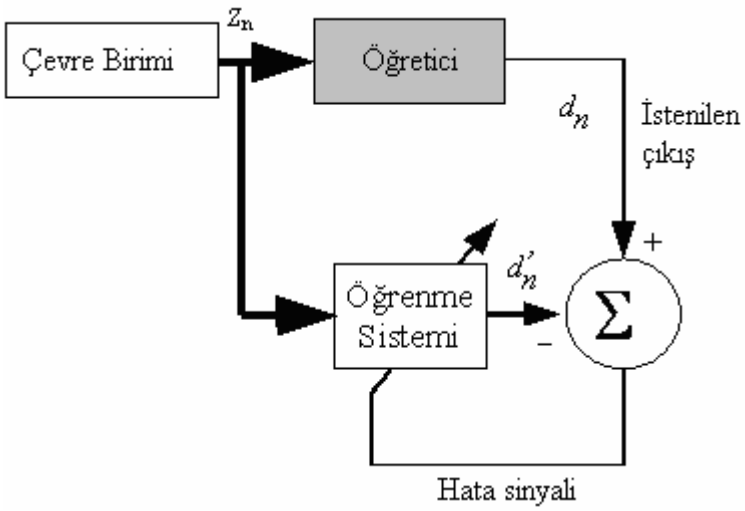
Şekil 4.3 Geri beslemeli ağ

4.7 YSA' ların Öğrenme Algoritmalarına Göre Sınıflandırılması

Öğrenebilmek, zekanın temel özelliklerinden birisidir. Öğrenmeyi kesin olarak tanımlamak, formüle etmek oldukça karmaşık ve zordur. Öğrenme olayı, YSA bağlamında ele alınacak olursa, kesin olarak belirlenmiş bir problemin, YSA tarafından belli kurallar yardımıyla verimli bir şekilde geçekleştirebilmesi için, yapısını veya ağın işlem elemanlarını bağlayan ağırlıkların değiştirilmesi olarak görülebilir.

Ağ, genelde mevcut eğitim setindeki verileri kullanarak işlem elemanlarını bağlayan ağırlık katsayılarını öğrenir. Ağın öğrenme performansı, tekrarlanan ağırlık katsayıları güncellemesi ile artar. Bilginin kurallar şeklinde tanımlandığı klasik uzman sistemlerinin tersine YSA, girişlerine ve çıkışlarına gösterilen örnekten kendi kurallarını oluşturur. YSA, ağırlık katsayılarını belli bir kurala göre düzeltir. Bu kurala öğrenme algoritması denir. Eğiticili, eğiticisiz ve takviyeli öğrenme şekilleri olmak üzere üç çeşit öğrenme çeşidi vardır.

i) Eğiticili öğrenme: YSA' nın eğiticili öğrenme metodunda Şekil 4.4' de görüldüğü gibi harici bir öğretici bulunmaktadır.



Şekil 4.4 Öğreticili öğrenme modeli blok şeması

Bu yöntemde öğretici, çevre biriminden gelen her giriş için bir çıkış üretir. Giriş ve çıkış çiftlerinin $[(z_1,d_1),(z_2,d_2),...,(z_n,d_n)]$ eğitimi için örnek veri setini

oluşturur. Öğretici, eğitim sırasında her bir z_n giriş vektörü için istenilen $d_n^{'}$ çıkış vektörünü YSA' ya gönderir. İstenilen çıkış ile anlık ağ çıkışı d_n arasındaki fark alınır. Bu farka hata sinyali denir. Öğrenme sistemi bu hata sinyalinden faydalanarak ağ parametrelerini düzeltir. Bu düzeltme tekrarlanarak adım adım gerçekleştirilir ve ağ parametreleri güncellenir. Böylece öğreticideki bilgi, YSA'ya eğitim yoluyla en fazla olacak şekilde aktartılmış olur. En fazla bilginin aktarıldığı yani hata sinyalinin en küçük olduğu noktada eğitme işlemi durdurulur ve öğretici devre dışı bırakılır.

ii) Eğiticisiz Öğrenme: Eğiticisiz veya kendi kendini organize eden (self-organized) öğrenme metodunda harici bir öğretici bulunmaz. Girişe verilen bilgilerden elde edilen çıkış bilgisine göre ağ, öğrenme kurallarını kendi kendine belirler. Bu öğrenme yapısında, istenilen çıkış bilgisinin bilinmesine ihtiyaç yoktur. Bu yapıyı gösteren blok şema Şekil 4.5' de verilmiştir. Eğiticisiz öğrenme modeline gruplandırma, alt kümelendirme gibi uygulamalarda rastlamak mümkündür (Haykin 1994).

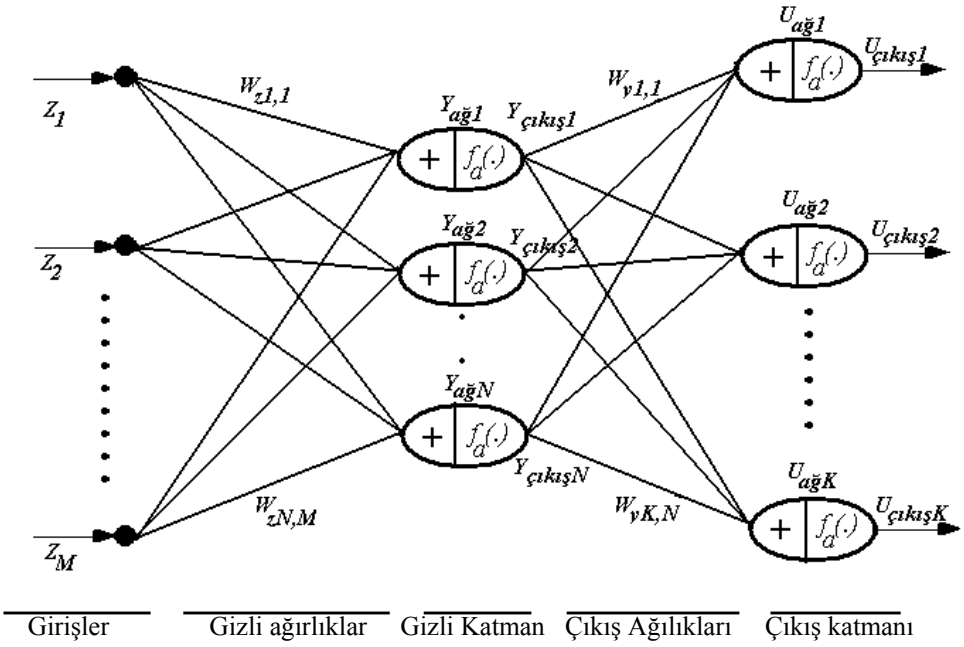


Şekil 4.5 Eğiticisiz öğrenme blok şeması

iii)Takviyeli Öğrenme: Takviyeli öğrenmede, eğiticisiz öğrenmede olduğu gibi istenilen çıkış bilgisine gerek yoktur. İstenilen çıkışı elde etmek için giriş bilgileri ile çıkış bilgilerini değerlendiren bir uygunluk kriteri kullanılır.

4.8 Çok Katmanlı Perseptronlar

Çok katmanlı perseptronlar (MLP-Multi Layer Perceptron), giriş-çıkış katmanları arasında bir ya da daha fazla katmanın kullanıldığı YSA modelleridir. Bu katmanlar, gizli katman olarak isimlendirilir. Şekil 4.6' da iki katmanlı bir ileri beslemeli YSA yapısı verilmiştir.



Şekil 4.6 Çok katmanlı ağın yapısı

Giriş katmanındaki işlemci elemanlar, giriş sinyallerini diğer ara katmandaki işlemci elemanlara dağıtır. Bu sinyaller dağıtılmadan önce, giriş katmanı ile ara katman arasındaki bağlantı katsayıları ile ağırlıklandırıldıktan sonra toplanır ve aktivasyon fonksiyonundan geçirilerek çıkış katmanına aktarılır. Çıkış katmanındaki işlem elemanları ara katmandaki elemanlar gibi davranır ve ağ çıkış değerlerini hesaplar. Giriş ve çıkış katmanındaki hücre sayısı çözülecek problemi tanımlayan giriş/çıkış çiftinin sayısına bağlıdır. Gizli katman sayısı ve gizli katmandaki hücre sayıları deneme yanılma ile belirlenir (Rumelhart 1986). Ancak genelde giriş katmanındaki hücre sayısı ile gizli katmandaki hücre sayısı eşit seçilir. Gizli katman sayısı ve hücre sayısı, çözülecek problemin yapısı ve karmaşıklığı ile değişiklik gösterebilir.

4.8.1 Geriye yayılma algoritması

YSA' nın tarihsel gelişimi içinde çok katmanlı YSA' ların eğitiminde en başarılı olmuş algoritma, geriye yayılma (BP-Back Propagation*)* algoritmasıdır. Bu öğrenme algoritması çok katmanlı ileri beslemeli ağlar ile birleştirildiğinde elde edilen YSA' ya "geriye yayılma ağı" adı verilir. Bu algoritmanın bu isimle anılması YSA'nın gelişiminde geriye yayılma algoritmasının ne kadar önemli olduğunu

göstermektedir. Eğitim kümesindeki (Z_n,d_n) giriş-çıkış çiftleri verildiğinde geriye yayılma ağının giriş ile çıkış arasındaki ilişkiyi doğru bir şekilde öğrenebilmesi için öğrenme kuralı yardımıyla ağırlık matrisini değiştirir. Ağırlıkların güncellenmesi perseptron ağlarında kullanılan eğim azaltma (gradient descent) algoritmasına dayanır. Geriye yayılma ağında bilgi akışı iki yönlüdür. İlk olarak giriş verileri Z girişlerden ağ çıkışına doğru gönderilir. Bu girişlere karşılık ağın çıkışında oluşan $U_{çıkış}$ ile çıkışta görülmesi beklenen $U^d_{çıkış}$ arasındaki fark, yani hata sinyali, çıkıştan geriye doğru giriş katmanına iletilir. Ağın parametreleri güncellemesi, bu hata sinyaline göre yapılır.

Şekil 4.6' da gösterildiği gibi bir ileri beslemeli YSA' da M giriş katmanı, N gizli katmanı ve K çıkış katmanı hücre sayısını göstersin. Burada çıkış katmanında, Çizelge 4.2' de verilmiş olan aktivasyon fonksiyonlarından biri kullanılabileceği gibi, genelde ileri beslemeli YSA' larda sıkça kullanılan geriye yayılma algoritmasında yapılacak işlemlerde kolaylık sağlaması açısından *lin(x)* (f_a(x)=x) aktivasyon fonksiyonu kullanılmıştır. Gizli katman için aktivasyon fonksiyonu $f_a(.)$ genel fonksiyon yapısı ile gösterilmiştir. Geriye yayılma algoritması için iki katmanlı bir ağın matematiksel ifadeleri türetilecektir. Burada gösterilecek olan matematiksel yöntem ikiden fazla katmana sahip olan ağlar için de genişletilebilir.

İki katmanlı bir ağda sinyalin girişten çıkışa ileri doğru yayılmasını gösteren matematiksel ifadeler (4.3), (4.4), (4.5) ve (4.6)' daki denklemlerdeki gibidir;

$$Y_{ağ_j} = \sum_{i=1}^{M} w_{z_{j,i}} .\ Z_i \tag{4.3}$$

$$Y_{çıkış_j} = f_a\left(Y_{ağ_j}\right) \qquad j = 1...N \tag{4.4}$$

$$U_{ağ_j} = \sum_{i=1}^{N} w_{y_{j,i}} .\ Y_{çıkış_j} \tag{4.5}$$

$$U_{çıkış_j} = f_a\left(U_{ağ_j}\right) \qquad j = 1...K \tag{4.6}$$

Burada; Z_i giriş hücrelerinin girişlerini, $w_{z_{j,i}}$ *j.* gizli katman hücresi ile *i.* giriş değeri arasındaki bağlantı ağırlığını, $w_{y_{j,i}}$ *j.* çıkış hücresi ile i. gizli katman hücresi arasındaki bağlantı ağırlığını, $Y_{çıkış_j}$ *j.* gizli katman çıkışını, $U_{çıkış_j}$ *j.* çıkış

hücresinin çıkışını, $f_a(.)$ aktivasyon fonksiyonunu ifade eder. Ağın hata fonksiyonu aşağıdaki gibi yazılabilir;

$$J = \frac{1}{2}\sum_{j=1}^{K}\left(U^{d}{}_{çıkış_j} - U_{çıkış_j}\right)^2 \tag{4.7}$$

YSA' nın çıkışı, o anki çıkışı $U_{çıkış}$ ile istenilen $U^{d}{}_{cıkış}$ çıkışına yakınsadığında hata fonksiyonu J sıfıra gider. Yani ağın ağırlık katsayılarının güncellenmesinin amacı hata fonksiyonunu sıfıra götürmektir.

İki katmanlı YSA' da iki defa ağırlık güncellemesi yapılır. Birincisinde; çıkış katmanı ile gizli katman arasındaki ağırlıklar, ikincisinde gizli katman ile giriş katmanı arasındaki ağırlıklar güncellenir.

Eğim düşüm algoritmasına göre, gizli katmanlardan çıkış katmanına kadar bütün ağırlık katsayılarında yapılacak değişim miktarı;

$$\Delta w_{y_{j,i}} = -\eta \frac{\partial J}{\partial w_{y_{j,i}}} \qquad j = 1...K \;\; ve \;\; i = 1...N \tag{4.8}$$

ifadesinden hesaplanır. Burada η öğrenme oranıdır. (4.8) denkleminde kısmi türeve zincirleme türev kuralını uygulanırsa;

$$\Delta w_{y_{j,i}} = -\eta \left(\frac{\partial J}{\partial U_j} \frac{\partial U_j}{\partial U_{çıkış_j}} \frac{\partial U_{çıkış_j}}{\partial U_{ağ_j}} \right) \left(\frac{\partial U_{ağ_j}}{\partial w_{y_{j,i}}} \right) \tag{4.9}$$

denklemi elde edilir.

Çıkış katmanı için ağırlık adaptasyon ifadesi;

$$\Delta w_{y_{j,i}} = -\eta \delta_{y_j} Y_{çıkış_j} \tag{4.10}$$

ile gösterilir.

(4.10) denkleminde δ_j j. hata sinyalini gösterir. Hata sinyali aşağıdaki denklem ile tanımlanır;

$$\delta_{y_j} = \frac{\partial J}{\partial U_j} \frac{\partial U_j}{\partial U_{çıkış_j}} \frac{\partial U_{çıkış_j}}{\partial U_{ağ_j}} \tag{4.11}$$

Gizli katman ile giriş katmanı arasındaki ağırlıkların güncellenmesi

$$\Delta w_{z_{j,i}} = -\eta \frac{\partial J}{\partial w_{z_{j,i}}} \qquad j = 1...N,\ i = 1...M \tag{4.12}$$

denkleminden hesaplanabilir.

(4.12) denkleminin sağ tarafındaki ifade kısmi türevler cinsinden açılırsa;

$$\Delta w_{z_{j,i}} = -\eta \left(\sum_{k=1}^{K} \frac{\partial J}{\partial U_k} \frac{\partial U_k}{\partial U_{\text{çıkış}_k}} \frac{\partial U_{\text{çıkış}_k}}{\partial U_{a\breve{g}_k}} \frac{\partial U_{a\breve{g}_k}}{\partial Y_{\text{çıkış}_j}} \right) \frac{\partial Y_{\text{çıkış}_j}}{\partial Y_{a\breve{g}_j}} \frac{\partial Y_{a\breve{g}_j}}{\partial w_{z_{j,i}}} \tag{4.13}$$

sonucu elde edilir. Yine kısmi türevleri açılır ve yeniden yazılırsa;

$$\delta_{y_k} = -\frac{\partial J}{\partial U_k} \frac{\partial U_k}{\partial U_{\text{çıkış}_k}} \frac{\partial U_{\text{çıkış}_k}}{\partial U_{a\breve{g}_k}} \tag{4.14}$$

$$w_{y_{k,j}} = \frac{\partial U_{a\breve{g}_k}}{\partial Y_{\text{çıkış}_j}} \tag{4.15}$$

$$Z_i = \frac{\partial Y_{a\breve{g}_k}}{\partial z_{j,i}} \tag{4.16}$$

elde edilir.

Giriş ve gizli katman arasındaki ağırlıkların adaptasyonu için kısaca aşağıdaki denklem yazılabilir;

$$\Delta w_{z_{j,i}} = \eta \delta_{z_{j,i}} Z_i \tag{4.17}$$

(4.17) denklemindeki δ_{zj} çarpanı;

$$\delta_{z_j} = \left(\sum_{k=1}^{K} \delta_{y_k} w_{y_{k,j}} \right) \frac{\partial f_a\left(Y_{a\breve{g}_j}\right)}{\partial Y_{a\breve{g}_j}} \tag{4.18}$$

ifadesinden hesaplanabilir. İki katmanlı geriye yayılım algoritmasının adım adım işleyişi Çizelge 4.3' de verilmiştir.

Çizelge 4.3 İki katmanlı YSA' nın uygulama algoritması

1. Adım	YSA' nın tüm bağlantı ağırlık katsayılarına rasgele değerler ata ve öğrenme oranına $0<\eta<1$ aralığında bir değer ata.
2. Adım	YSA' nın Z girişlerini hazırla.
3. Adım	Z girişlerine göre YSA gizli katman girişlerini $Y_{ağ}$ ve çıkışlarını $Y_{çıkış}$ hesapla.
4. Adım	YSA' nın $U_{ağ}$, $U_{çıkış}$ ve U çıkışlarını hesapla.
5. Adım	$U^d_{çıkış}$ (veya U^d) değerlerini hesapla
6. Adım	Çıkış katmanı için çıkış bağlantı ağırlıklarını güncelle $w_{y_{j,i}}(k+1) = w_{y_{j,i}}(k) + \eta\delta_{y_j} Y_{çıkış_j}$ $\delta_{y_j} = (U_j^d - U_j)\dfrac{\partial f_a(U_{ağ_j})}{\partial U_{ağ_j}}$
7. Adım	Giriş ve gizli katman bağlantı ağırlıklarını güncelle. $w_{z_{j,i}}(k+1) = w_{z_{j,i}}(k) + \eta\delta_{z_j} \qquad j=1...N,\ i=1...M$ $\delta_{z_j} = \left(\sum_{k=1}^{K} \delta_{y_k} w_{y_{k,j}}\right)\dfrac{\partial f_a\left(Y_{ağ_j}\right)}{\partial Y_{ağ_j}} \qquad j=1...N$
8. Adım	İstenen ve YSA çıkışları arasındaki hesaplanan hata, istenilen seviyeye gelinceye kadar 2. adım'dan sonraki adımları tekrar et.

4.9 Öğrenme Faktörleri

Geriye yayılma algoritmalı YSA' larında ağın öğrenme performansını etkileyen birçok faktör vardır. Bunlardan bazıları ilk atanan ağırlık katsayıları, öğrenme oranı, performans fonksiyonu, ağırlık uyarlama kuralı, ağ mimarisi, eğitim süresinin uzunluğu gibi faktörlerdir. Geri yayılma algoritması ile eğitilen bir YSA' nın eğitim süresi çok uzundur. Hata hesaplarında yerel minimumda kalma durumu ve benzeri problemlerin giderilmesi için, öğrenmeyi etkileyen faktörlerin doğru seçilmesi gerekir. Bu problemler geriye yayılma algoritmasının temel dezavantajlarıdır.

Başlangıç ağırlık değerleri: YSA eğitime başlamadan önce, işlem elemanları arasındaki ilişkiyi sağlayan bağlantı ağırlık katsayılarına ilk olarak, rasgele küçük değerler atanır. Atanan ağırlık katsayılarının değerlerinin büyük olması durumunda, eğitim işleminin başında aktivasyon fonksiyonunu doyuma ulaştırabileceğinden geri yayılma algoritması ilk yerel minimum noktasında tıkanır. İlk ağırlık değerleri, ağın performansını, öğrenmedeki hızını ve istenilen çıkışlara yakınsamasını etkilemektedir. Ağırlık katsayılarının seçilmesinde önerilen bir yöntem, bağlantı ağırlıklarının $[-3/\sqrt{k_i}, +3/\sqrt{k_i}]$ aralığında olmasıdır. Burada k_i değeri *i.* hücre girişine gelen bağlantı sayısıdır.

Öğrenme oranı: Eğitim sürecini ve çözümün yakınsamasını doğrudan etkileyen η öğrenme oranıdır. Eğitim sürecinin başlangıcında uygun bir şekilde ayarlanmalıdır. Farklı eğitim aşamaları için uygun bir η değeri belirlenmemiştir. Yani öğrenme oranını belirleyen belli bir kriter yoktur. Bu oran genellikle deneysel olarak belirlenir. Bununla birlikte öğrenme oranı için bazı genellemeler yapılabilir. Büyük değerli öğrenme oranı için öğrenme algoritmasının çözüme hızlı yakınsamasını sağlayabildiği gibi, ağın davranış bozuklularına da sebep olabilir. Küçük değerli öğrenme oranı tanımlanması durumunda algoritmanın çözüme yakınsaması uzun zaman alır. Çözülecek problem için optimum bir öğrenme oranı bulunması gerekir. Genelde bu oran $0.001<\eta<1$ aralığında seçilir.

Performans fonksiyonu: Öğrenme performansını etkileyen önemli faktörlerden biri olan performans fonksiyonu veya hata fonksiyonu genelde L_p normu ve $1<p<\infty$ şeklinde ifade edilir. Genel tanımı aşağıdaki gibidir.

$$E = \frac{1}{n}\sum_j (d_j - d_j')^n \qquad 1 \le n \le \infty \qquad (4.19)$$

Bunlar normlar arasında, L_2 normu en küçük kareler kriteri olarak bilinir ve karesel hata fonksiyonlarında en çok kullanılan kriterdir. Bunun yanında L_1 ve L_∞ normlarını esas alan hata fonksiyonları da yaygın olarak kullanılır. Özellikle L_∞ normu Chebyshev normu olarak bilinir.

Eğitim setinin boyutu: Birçok veri işleme alanlarında olduğu gibi YSA'larında da doğru çözüme ulaşmak için doğru verilere ihtiyaç vardır. Eğer giriş veri kümesi, problemi yeterince temsil etmiyorsa, herhangi bir algoritma veya

yöntemin kullanılması anlam taşımayacaktır ve problemi çözmede yetersiz kalacaktır. YSA, giriş uzayından çıkışları oluşturan veri uzayını haritalayan bir sistem olarak düşünülebilir. Bütün giriş verilerini oluşturan uzayın her bir bölgesini kapsayan giriş verileri ile ancak çıkış verileri uzayı sağlıklı bir şekilde elde edilebilir. Bütün giriş uzayını kapsayan verileri elde etmek büyük kaynak ve maliyet gerektirir. Burada kaynak boyutu; veri toplama yöntemi, veri toplama donanımının karmaşıklığı ve zaman olarak görülebilir. Genel olarak, kaynağın boyutu, giriş uzayını oluşturan hiper uzayın hacmi ile doğru orantılıdır. Giriş uzayının her bölgesinden veri bulunması da yeterli değildir. Yüksek hacimli giriş verileri, ağın öğrenme zamanını uzatır ve ağın giriş uzayının istenilmeyen bölgelerini öğrenmesine sebep olur. Bundan dolayı giriş verilerinin uygun yöntemlerle seçilmesi gerekir. Bu yöntemlerden birisi giriş uzayını oluşturan verilerin kullanılması yerine, bu verileri kullanarak çıkarılan öznitelik vektörlerinin kullanılmasıdır. Bu yöntem, eğitim sürecinin daha verimli olmasını sağlayabilir. Bir başka yöntem de giriş verilerini oluşturan ölçümlerin birbirine yakın veya aynı olanların, ölçüm sınırları dışında olan çok gürültülü verilerin giriş veri uzayından ayıklanarak veri hacminin azaltılmasına gidilmesidir.

Verilerin ön işlenmesi: Giriş ve çıkış verilerinin YSA' ya normalize edilerek uygulanması, YSA' nın performansını doğrudan etkileyecektir. Çünkü normalizasyon işlemi, verilerin aktivasyon fonksiyonunun aktif bölgesinden aktarılmasını sağlar. Normalizasyon işlemi, veri kümesinin bir bölgede yoğunlaşmasını engellemek için gereklidir. Örneğin; *logsig*(x) aktivasyon fonksiyonu [0,+1] aralığında aktiftir. Bu bölgenin dışındaki veriler sıfırdan küçükse sıfıra, birden büyükse bire eşitlenir ve anlamlı verilerin etkisi azalır ve bu da YSA' nın öğrenmesini olumsuz etkiler. Aslında bu bölgenin dışındaki veriler ayrı verilerdir fakat YSA tarafından bir fark yokmuş gibi değerlendirilir. Genellikle verinin [0,1] veya [-1;+1] aralıklarından birine ölçeklendirilmesi önerilir. Eğer aktivasyon fonksiyonu bipolar ise ölçeklendirme [-1,+1] aralığında yapılmalıdır.

Bir X veri kümesinin [0,1] arasında normalize işlemine tabi tutulması için o kümenin x_{min}, x_{max} değerleri bulunur ve (4.20) eşitliği yardımıyla normalize edilir. Buna göre X_{norm} normalize edilmiş değer aralığı;

$$X_{norm} = \frac{X - x_{\min}}{x_{\max} - x_{\min}} \tag{4.20}$$

olarak elde edilir. Eğer ölçeklendirme aralığı [-1,+1] arasına çekilmek isteniliyorsa (4.21) eşitliği yardımıyla gerçekleştirilebilir.

$$X_{norm} = 2\frac{X - x_{\min}}{x_{\max} - x_{\min}} - 1 \tag{4.21}$$

Eğitim esasında, bu işlemlerden ele edilen normalize giriş ve çıkış verileri kullanılır. Eğitim işlemi tamamlandıktan sonra elde edilen YSA sonuçlarına ters normalize işlemi uygulanır ve gerçek değerler elde edilir

YSA' larda sıkça uygulanan bir başka yöntemde giriş ve çıkış verilerinin sıfır ortalama ve birim varyansa göre normalize edilmesidir. Bu işlemi gerçekleştirmek için (4.20)' deki eşitlikten faydalanılır. Buna göre;

$$X_{norm} = \frac{X - ort(X)}{std(X)} \tag{4.22}$$

şeklinde elde edilir. Burada *ort(),* ortalama ve *std(),* standart sapma hesaplayan fonksiyonlardır.

4.10 Yapay Sinir Ağlarında Genelleme

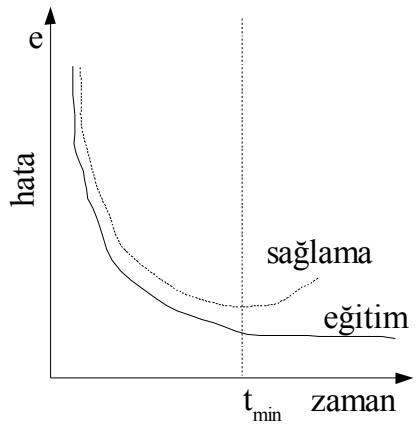
YSA' lar ve benzeri yapay zeka sistemleri, öğrenme süreçlerinde aşırı öğrenme problemi ile karşı karşıyadır (Lawrence ve ark. 1997). Ağ, eğitim verisindeki gürültülere karşı az tepki verip, eğitim kümesinden olmayan verilere aşırı tepki gösterebilir. Burada ağ, eğitim verilerini öğrenmiştir. Ancak aynı sınıftan farklı girişlere karşı genelleştirme yapamamaktadır. Aşırı öğrenme, ağın büyüklüğü ile doğrudan orantılıdır. Yani ağ parametrelerinin toplam sayısı, giriş veri kümesinden büyükse aşırı öğrenme durumu ile karşı karşıya kalınır. Ağın sağlıklı bir genelleştirme yapması için yeterince giriş verisi bulunması ve ağ parametreleri sayısından fazla olması gerekir. YSA' nın genelleştirme kabiliyetini artırmak için değişik metotlar vardır. Bunlardan bazıları; eğitimi erken durdurma, otomatik ayarlama, model seçme, ağırlık azaltma, ayıklama metotlarıdır.

4.10.1 Eğitimi erken sonlandırma metodu

Erken sonlandırma metodunda eldeki mevcut veri üç kısma bölünür. Birinci veri kümesi, YSA' nın eğitimi için kullanılır. İkinci veri kümesi, ağ sağlaması (validation) için kullanılır. Eğitim sırasında sağlama için kullanılan veriler, ürettiği çıkışlar ve çıkışlara göre hatalar denetlenir. Eğitim başlangıcında bu hata, eğitim verisinin ürettiği hata ile eşzamanlı azalır.

Hatanın zamana göre değişimi Şekil 4.7' de verilmiştir. Hata değişim eğrisinden görüleceği gibi ağın aşırı öğrenmeye başladığı t_{min} anında, sağlama verisinden oluşan hata, artma eğilimi gösterir. Bu artış, belirlenmiş olan belli bir tekrarlama sayısı kadar devam ederse eğitim süreci sonlandırılır. Bu noktada ağın minimum sağlama hatası üreten ağ parametreleri korunur. Üçüncü veri kümesi ise test verilerini oluşturur.

Test verileri eğitim sürecinde kullanılmaz, sadece farklı ağ modellerinin performansını karşılaştırmak için kullanılır. Bu testte sağlama verilerinin ürettiği hatalar aynı iterasyon sayısında minimuma gitmiyorsa, veri bölümlemesi yeterince iyi yapılmamıştır. Genelde YSA uygulamalarında toplam verinin %75-%80' i eğitim, %10-%15 sağlama, %10' u test için kullanılır. Bu oranlar, problemin karmaşıklığı ile değişebilmektedir.



Şekil 4.7 Eğitimi erken sonlandırma hata grafiği

4.10.2 Otomatikleştirilmiş ayarlama metodu

Bu metot, performans fonksiyonunun değiştirilmesine dayanan ve YSA' nın genelleştirme yeteneğini iyileştiren metottur (Sağıroğlu 2003). İleri beslemeli ağlarda kullanılan tipik performans fonksiyonu ağ hatalarının karesel toplamının ortalaması olan MSE' (Mean Squared Error) dir. MSE;

$$MSE = \frac{1}{n}\sum_{j=1}^{n} e_j^2 = \frac{1}{n}\sum_{j=1}^{n}(d_j - d_j')^2 \tag{4.23}$$

şeklinde tanımlanır. Burada d_j istenilen çıkış, $d_j^{'}$ ağ çıkışıdır. Ağ sapmaları ve ağırıkları olan w_j' nin kareleri toplamının ortalaması olan bir terim eklenerek genelleştirme iyileştirilebilir. Bu amaçla;

$$MSEREG = \gamma MSE + (1-\gamma)MSEW \tag{4.24}$$

kullanılır, burada γ performans oranıdır ve

$$MSEW = \frac{1}{n}\sum_{j=1}^{n} w_j^2 \tag{4.25}$$

ile ifade edilir. MSEREG performans fonksiyonunun kullanılması, ağın sapması ve ağırlıkların küçük olmasına sebep olacak ağın, yumuşak davranmasının sağlayacak ve aşırı öğrenmesini önleyecektir (Foresee 1997). Optimum ayarlama parametresinin otomatik olarak tespit edilmesi arzulanır. Bu amaçla kullanılan yöntemlerden birisi Bayesian ayarlama yöntemidir (MacKay 1992). Bayesian yönteminde ağın ağırlıklarının ve sapmalarının belli dağılımlara sahip rasgele değişkenler olduğu varsayılır (Foresee 1997).

4.11 YSA ile Nonlineer Sistem Tanılama

Sistem tanılama işlemi, bir dinamik sistemin giriş ve çıkışlarından ölçülen değerlerden, sistemin matematiksel modelinin oluşturulmasıdır. Sistemin matematiksel modeli, sistemin özelliklerini tanımlar ve belli bir amaca yönelik kullanılır (Abulafya 1995). Model, sistemi mutlak doğrulukta tanılamak zorunda değildir ve bu beklenmemelidir. Sistem modelinin kullanım amaçlarını şöyle sıralanabilir;

- Sistemin simülasyonu
- Gelecek adımların kestirimi
- Hata bulma
- Kontrol sistemi tasarımı

Sistem tanılama amacıyla lineer sistemler için bugüne kadar geliştirilmiş birçok teknik mevcuttur. Nonlineer sistem tanılama, lineer sistemlere göre daha zor ve daha karmaşıktır. Nonlineer sistemlerin tanılanması ve denetimi alanında, geri besleme ile lineerleştirme ve tanımlama fonksiyonu gibi yöntemler incelenmiş ancak, sistemlere özgü teoriler geliştirilememiştir (Pham ve ark.1997).

Genel olarak nonlineer sistem teorisi ve kararlılık kriterleri yeterince geliştirilemediğinden karmaşık, belirlenemeyen dinamikleri olan ve nonlineer sistemler kullanım sahalarında yerini aldıkça bu sistemlerin davranışlarını ve dinamik yeteneklerini iyileştiren ileri kontrol yöntemlerine ihtiyaç duyulmaktadır. Tek girişli tek çıkışlı (SISO) lineer sistemlerin teorisi iyi bilinmektedir. Bunun yanında çok girişli ve çok çıkışlı (MIMO) lineer sistemlerin analizi ve tanılanması bile oldukça karmaşık olmaya başlar. Tanılanma teorisinin yeterince gelişmemiş olması, nonlineer MIMO sistemlerinin analizini daha zor ve karmaşık yapmaktadır.

Bir MIMO sistemin tanılanmasını kolaylaştırmak için, sistem ayrıştırılarak çok girişli tek çıkışlı (MISO) veya SISO sistemlerine dönüştürülür. Nonlineer MIMO sistemlerin belirsizlikleri ve dinamik davranışlarındaki değişiklikleri tanılamak için akıllı, uyarlanabilir yöntemlerin en etkin olduğu görülmüş ve son 10-15 yıldır bu alanda yoğun çalışmalar yürütülmektedir (Tuncay S. ve ark. 1999).

Akıllı sistemlere en güzel örnek ve nonlineer sistem tanılama alanında en sık görülen metot, YSA ile nonlineer sistem tanılamasıdır. YSA, nonlineer sistemlerin tanılanmasında en etkin yöntem olarak görülmekte ve gerçek dünyadaki sistemler için YSA' nın modelleme ve denetim yeteneklerinin incelenmesine ihtiyaç duyulmaktadır. YSA' nın tanılama metotları arasında en çok tercih edilmesinin bir sebebi de nonlineer MIMO sistemlerine en yüksek doğrulukta yaklaşabilme yeteneğidir. Akıllı sistemler zamanla değişen ve iyi tanımlanmamış ortamlarda çalışan, çevresel ve sistem dinamiklerindeki değişimlere kendini uyarlayan, bilgileri kararlı bir yöntemle öğrenen ve sistemin üzerinde çok az kısıtlamalar getiren

algoritmalara ihtiyaç duyar. Bu bağlamda bilinen klasik lineer sistem tanılama metotları, YSA ile sistem tanılama çerçevesine genişletilerek gerçekleştirilir. Daha öncede bahsedildiği gibi, YSA bir sistemin giriş ve çıkış verilerinden o sistemi öğrenebilme kabiliyetine sahiptir. Bu bölümde lineer sistem tanılama tekniklerinin nonlineer sistemlere genişletilmesi ve YSA yardımıyla nonlineer sistem tanılama gerçekleştirilmesi konusu üzerinde durulacaktır.

4.11.1 YSA sistem tanılama modelleri

Otoregresif dışardan girişli (ARX-AutoRegressive eXternal input), otoregresif kayan ortalamalı dışardan girişli (ARMAX- AutoRegressive Moving Average eXternal input), çıkış hata modeli (OE-Output Error) gibi lineer model yapıları temel alınarak nonlineer modeller de giriş vektörlerine göre lineer modellere benzer isimlerle söylenir. Bunlar NARX, NARMAX ve NOE modeller olarak isimlendirilir. y(k) sistem çıkışını, $\hat{y}(k)$ model çıkışını u(k) model giriş vektörünü göstermek üzere literatürde geçen bazı nonlineer model yapıları verilmiştir. Aşağıdaki ifadelerde YSA ile nonlineer bir sistem modeli tanımlanabilir;

$$y(t) = g\left[\varphi(t,\theta),\theta\right] + e(t) \tag{4.26}$$

veya kestirimci formunda ifade edilirse;

$$\hat{y}(t \mid \theta) = g\left[\varphi(t,\theta),\theta\right] \tag{4.27}$$

şeklinde olur. $\varphi(t)$ regresyon vektörünü yani giriş ve çıkışların geçmiş değerlerini, θ parametresi YSA' nın ağırlık katsayılarını içeren vektörü, g[] YSA tarafından oluşturulan işlevsel fonksiyonu gösterir.

Bu çalışmada bütün nonlineer YSA modelleri ele alınmamıştır. Sistem tanılamada en sık rastlanan modeller üzerinde durulmuş ve tanıtılmıştır. Önceki bölümlerde, YSA tanıtılırken, sisteme uygulanan giriş ve ölçülen çıkış değerlerinden sistemi öğrenebilme yeteneğinde olduğundan bahsedilmişti. YSA kullanılarak sistem tanılama problemi iki temel başlıkta toplanabilir;

i) YSA' ya uygulanacak girişlerin seçilmesi

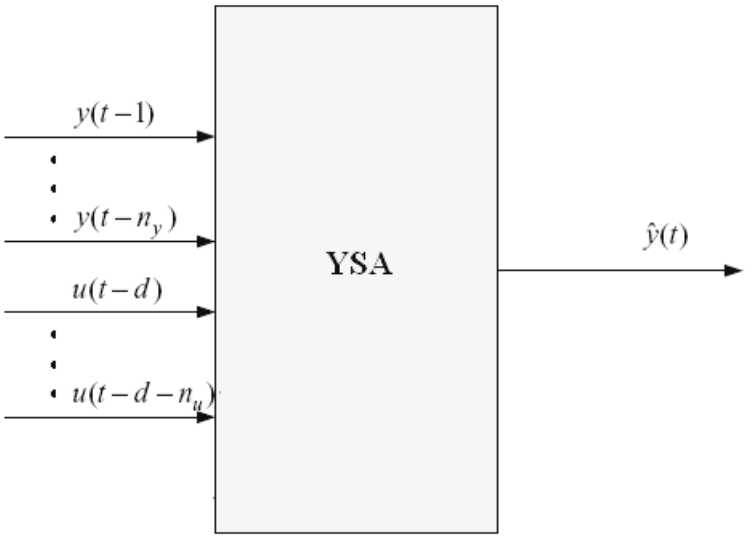
ii) YSA' nın iç mimarisinin belirlenmesi

4.11.2 NARX modeller

YSA ile oluşturulan NARX modeli Şekil 4.8' de görülmektedir. NARX modeli aşağıdaki gibi ifade edilebilir;

$$\hat{y}(t) = g\left[\varphi(t,\theta),\theta\right]$$
$$\varphi(t) = \left[y(t-1)...y(t-n)\,,u(t-d)...u(t-m-d+1)\right]^T \tag{4.28}$$

burada *n* çıkışın geçmiş değer sayısını, *m* girişin geçmiş değer sayısını, *d* gecikmeyi ifade eder. NARX modeli, tanılanacak sistemin sadece giriş ve çıkış verileri kullanır ve YSA modeline uygulaması ile gerçekleşir.



Şekil 4.8 YSA-NARX yapısı

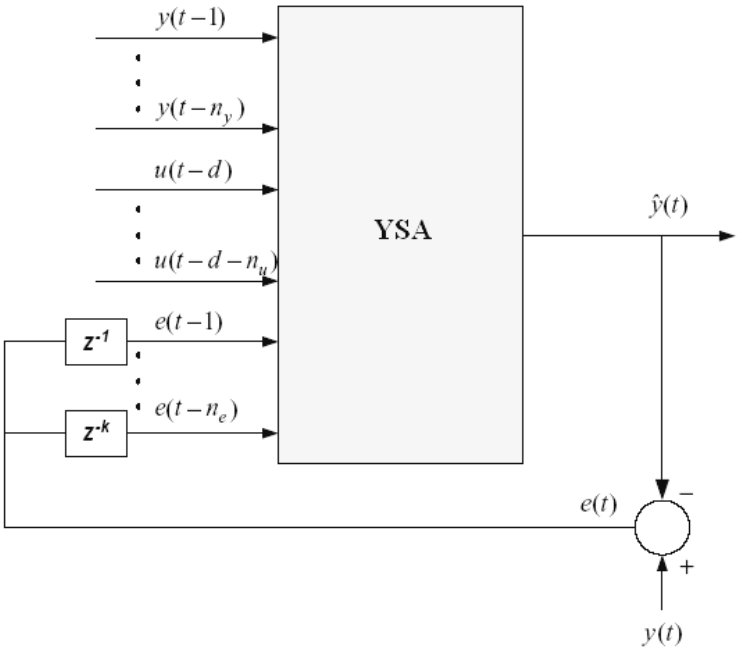
YSA ile kurulan NARX modeller, girişlerine geri besleme yapılmadığında ve giriş ile çıkış arasında sadece cebirsel işlemler bulunduğundan hep kararlıdırlar (Norgaard 2001). NARX modeller giriş verilerinde gürültü seviyesi düşük olduğu zaman daha doğru sistem tanılama gerçekleştirmektedir.

4.11.3 NARMAX modeller

Şekil 4.9' da verilmiş olan NARMAX modelde, kestirilen çıkış ile istenilen çıkış arasındaki fark yani; hata sinyali, geri besleme ile tekrar YSA modelinin girişine uygulanır.

NARMAX modele ilişkin matematiksel ifade aşağıda verilmiştir;

$$\hat{y}(t) = g\left[\varphi(t,\theta),\theta\right]$$
$$\varphi(t) = \left[y(t-1)...y(t-n)\,,u(t-d)...u(t-m-d+1),\varepsilon(t-1)...\varepsilon(t-p)\right]^T \quad (4.29)$$



Şekil 4.9 YSA-NARMAX yapısı

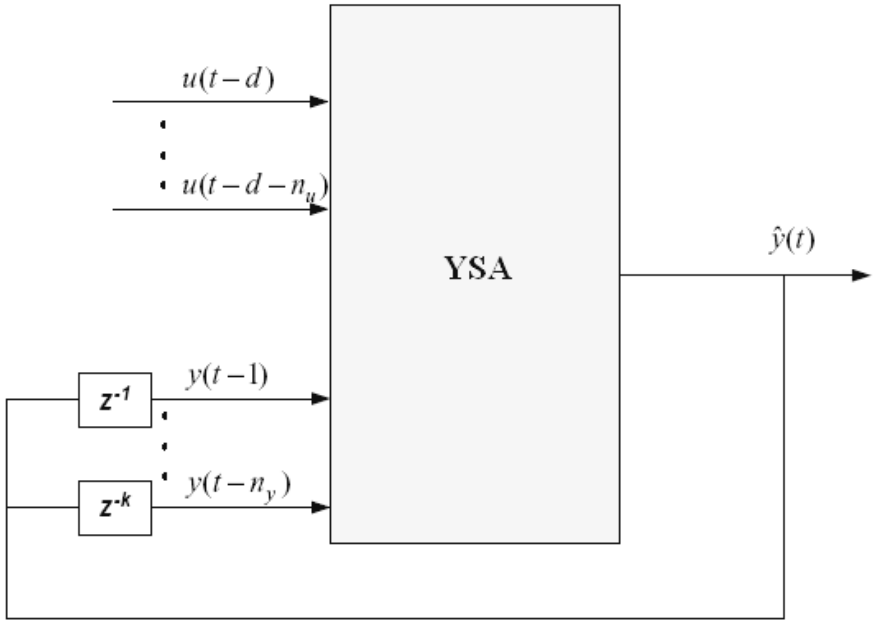
NARMAX modelinde hata sinyali girişe uygulandığından, bu tür bir kestirimci kararsız durumlar sergileyebilir ve bu kararsızlık durumların incelenmesi oldukça zordur. NARMAX modelin yerel olarak karasız olduğu düşünülebilir fakat farklı yapılardaki NARMAX modelleri kararlı olabilir.

4.11.4 NOE modeller

NOE modeli, kestirilen çıkışların girişlere geri beslenmesi ile gerçekleştirilir. Şekil 4.10' da NOE modeli verilmiştir ve bu model matematiksel olarak aşağıdaki gibi ifade eldir;

$$\hat{y}(t) = g\left[\varphi(t,\theta),\theta\right]$$
$$\varphi(t) = \left[\hat{y}(t-1)...\hat{y}(t-n)\ ,u(t-d)...u(t-m-d+1)\right]^T \tag{4.30}$$

NOE sistemleri geri beslemeli yapılar olduklarından kararsızlık sorunları ile karşı karşıya kalabilirler.



Şekil 4.10 YSA-NOE yapısı

NOE yapısı hariç yukarda bahsedilen NARX ve NARMAX yapıları, zaman serilerinin analizinde ve kestirim problemlerinde de kullanılabilir. Bunun için, YSA' ya giriş olarak uygulanan sistem çıkışları sıfırlanır. Böylece bu tür NARX modeller, NFIR (Nonlinear Finite Inpulse Response) modellere dönüşürler. Bu çalışmada da NARX modelinin bu hali kullanılmıştır.

4.11.5 Tanılanacak sistemin derecesinin tespiti

Tanılanacak sistemin derecesini belirlemek için literatürde birkaç değişik metoda rastlamak mümkündür. Krishnaswami (1995), koherans fonksiyonu hesaplamalarına dayanan bir metot geliştirmiştir. Bir diğer metot da He ve Asada (1993), Lipschitz oranlarını kullanan bir algoritma önermişlerdir. He ve Asada' nın önerdiği metot, en çok bilinenidir ve bu bölümde bu algoritma incelenecektir**.** Bu

algoritma optimal regresyon yapısını tespit etmeye yaramaktadır. Bu metot özellikle sistemde gürültü oranı çok yüksekse mutlak sonuç vermemektedir. Ancak sistemin derecesi hakkında bir ön bilgiye sahip olmak için kullanılabilir.

Bir sistemin gürültüden arındırılmış NARX modeli (4.31) denklemleri ile tanımlanmış olsun;

$$\begin{aligned} y(t) &= g_0\left[\varphi(t),\theta\right] \\ \varphi^T(t) &= \left[\varphi_1,\varphi_2,...,\varphi_z\right] \\ &= \left[y(t-1)...y(t-n),u(t-d)...u(t-d-m)\right] \end{aligned} \tag{4.31}$$

Z sisteminin, girişi φ ve çıkışı y ikilisi ile gösterilsin;

$$Z^N = \{[\varphi(t), y(t)], \qquad t = 1,.....,N\} \tag{4.32}$$

Sistemi temsil eden g fonksiyonunun türevinin, pozitif bir sayı olan B ile sınılırlı olduğu varsayılır. Bu durumda;

$$\left|g_l\right| = \left|\frac{\partial g_0}{\partial \varphi_l}\right| \le B \quad l = 1,2,...,z \tag{4.33}$$

ifadesi yazılabilir.

Bütün giriş ve çıkış ikilisinin oluşturacağı kombinasyonlar, (4.34) bağıntısı yardımı ile q_{ij} Lipschitz oranları hesaplanır.

$$q_{ij} = \left|\frac{y(t_i)-y(t_j)}{\varphi(t_i)-\varphi(t_j)}\right| \quad i \neq j \tag{4.34}$$

burada $\left|.\right|$ işlemi Euclide normudur (Euclide mesafesi) .

$\delta y = y(t_i) - y(t_j)$, $\delta\varphi = \varphi(t_i) - \varphi(t_j)$

olsun. Eğer $\delta\varphi$ farkları küçük ise;

$$\delta y = \frac{\partial g}{\partial \varphi_1}\delta\varphi_1 + \frac{\partial g}{\partial \varphi_2}\delta\varphi_2 + ... + \frac{\partial g}{\partial \varphi_z}\delta\varphi_z \tag{4.35}$$

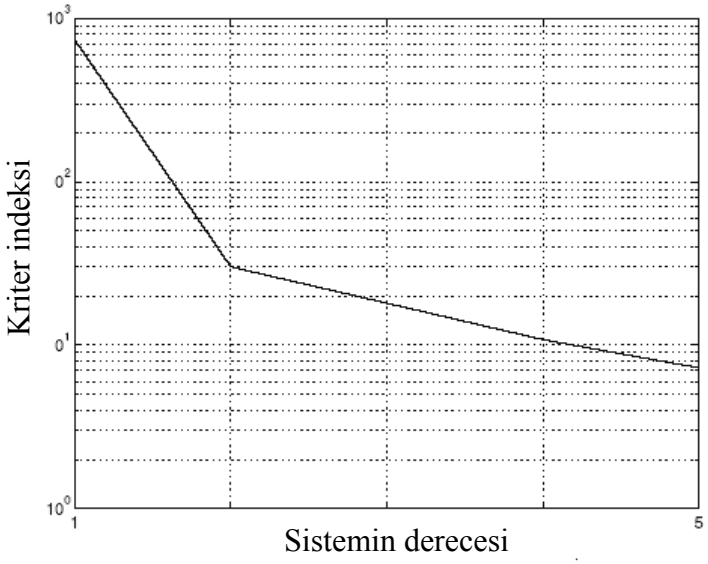
ifadesi yazılabilir.

He ve Asada' nın önerdiği Lipschitz oranlarına dayanan algoritmanın işleyişi şöyledir;

1. Algoritmada, tanılanacak sistemin taranacak derece aralığı seçilir ve bütün Z^N' ler için Lipschitz oranları hesaplanır
2. $\delta\varphi$ farkının en küçük olduğu değerde en büyük q oranı seçilir ve p=0.01N ~ 0.02N arasında seçilir
3. (4.36) bağıntısında verilen kriter indeksi hesaplanır

$$q^{-(n)} = \left(\prod_{k=1}^{p} \sqrt{n} q^{(n)}(k) \right)^{\frac{1}{p}} \tag{4.36}$$

4. Adım 1,2,3' deki işlemler farklı sistem dereceleri için tekrarlanır
5. Seçilen farklı sistem derecelerine karşılık kriter indeks grafiği çizilir ve optimum sistem derecesi bulunur. Optimum sistem derecesi eğrinin kırılma noktası olarak alınır. Şekil 4.11' de çizilmiş olan kriter indeksine karşılık sistem derecesi verilmiştir. Bu grafikten optimal sistem derecesi 2 olarak görülmektedir.



Şekil 4.11 Sistem derecesine göre kriter indeks grafiği

4.11.6 Korelasyon testleri

Kestirim hataları içinde tanılanan sistemin dinamiği ile ilgili bilgi bulunmuyorsa, tanılama işleminin, eğitim setinden (sistemin giriş-çıkış verilerinden) sistemin bütün dinamik bilgilerini yeterli derecede iyi modellediğini gösterir. Sistemin yeterince tanılandığını anlamak için, sistemin lineer ve nonlineer bütün geçmiş giriş ve çıkış değerlerinin kombinasyonunun, kestirim hataların kalıntıları ile korele olmaması gerekir. Pratikte bunu gerçekleştirmek anlamsızdır, bunun yerine otokorelasyon ve kroskorelasyon fonksiyonlarını kullanarak, sistemin yeterince tanılandığı anlaşılabilir. Sistemin iyi modellenmesi durumunda, korelasyon işlemleri sonucu ortaya çıkan değerlerin Gauss dağılımı göstermesi ve *1/n* varyansında olması beklenir (Billings ve ark. 1986). Kestirilen çıkış ile ölçülen çıkış arasındaki fark olan rezidü;

$$\varepsilon(t) = \varepsilon(t, \hat{\theta}_N) = y(t) - y(t, \hat{\theta}_N) \tag{4.37}$$

şeklinde ifade edilir.

Otokorelasyon testi: Otokorelasyon fonksiyonu, iki amaç için kullanılabilir;

i) Bir veri serisin rasgele dağılımlı olup olmadığını anlamak
ii) Rasgele olmayan bir veri serisini modellemek

Sistem tanılama alanında otokorelasyon fonksiyonu, kestirim hatalarının rasgele dağılımlı olup olmadıklarını test etmek için kullanılır. Hatanın otokorelasyon fonksiyonu denklem (4.38)' de verilmiştir.

$$\hat{R}_{\varepsilon\varepsilon}(\tau) = \frac{\sum_{t=1}^{N-\tau}\left(\varepsilon(t,\hat{\theta}) - \overline{\varepsilon}\right)\left(\varepsilon(t-\tau,\hat{\theta}) - \overline{\varepsilon}\right)}{\sum_{t=1}^{N-\tau}\left(\varepsilon(t,\hat{\theta}) - \overline{\varepsilon}\right)^2} = \begin{cases} 1, \tau = 0 \\ 0, \tau \neq 0 \end{cases} \tag{4.38}$$

Kroskorelasyon testi: Eğer oluşturulan model, sistemi doğru bir şekilde temsil ediyorsa giriş verileri ile rezidüler korele değildir, yani giriş verilerinin rezidülerde izinin olmadığını gösterir.

$$\hat{R}_{u\varepsilon}(\tau)=\frac{\sum_{t=1}^{N-\tau}\left(u(t)-\bar{u}\right)\left(\varepsilon(t-\tau,\hat{\theta})-\bar{\varepsilon}\right)}{\sqrt{\sum_{t=1}^{N}\left(u(t)-\bar{u}\right)^2\sum_{t=1}^{N}\left(\varepsilon(t,\hat{\theta})-\bar{\varepsilon}\right)^2}}=0\,,\ \forall\,\tau \tag{4.39}$$

Genelde $\tau\in$[-20,20] aralığında denklem (4.38) ve denklem (4.39) için sonuçların asimptotik olarak sıfır olduğu veya başka bir ifadeyle %95 güvenilirlik sınırları içinde bulunup bulunmadığı test edilir. Bunun için;

$$\frac{-1.96}{\sqrt{N}}<\hat{R}<\frac{1.96}{\sqrt{N}} \tag{4.40}$$

aralığı %95 güvenilirlik sınırı olarak tanımlanabilir.

Denklem (4.39)' daki ifade ile modelin kabul edilebilir olup olmadığı test edilebilir. Aksi takdirde modelde iyileştirilmeye gidilir ve kroskorelasyon testi tekrarlanır. Otokorelasyon ve kroskorelasyon değerlerinin (4.40) 'da belirtilen aralıkta olması istenir. Burada belirtilen testler MISO, MIMO sistemleri için de geçerlidir, bu sistemlerde bütün girişlerin ve çıkışların kombinasyonunun korelasyon fonksiyonları hesaplanır.

4.11.7 Varyans karşılaştırması

Sistemden ölçülen çıkış değerleri y_1 ile modelin ürettiği çıkış değerleri y_2'nin birbirine ne kadar benzediğini yüzde olarak gösteren fonksiyon (VAF-Variance Accounted For) olarak adlandırılır. VAF denklem (4.41)'de tanımlanmıştır;

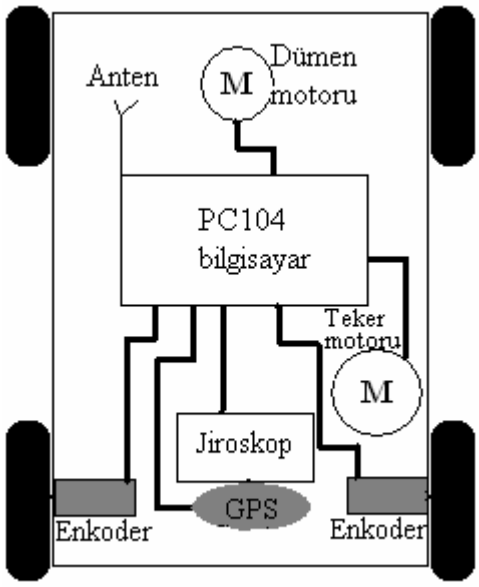
$$VAF=100\left[1-\frac{var(y_1-y_2)}{var(y_1)}\right] \tag{4.41}$$

(4.41) ifadesindeki *var()* varyans fonksiyonudur.

y_1 ve y_2 birbirinin aynısı ise VAF değeri %100' dür. Eğer farklı iseler VAF değeri daha küçük değerde olur. VAF fonksiyonu, bir sistem için oluşturulan modelin cevabını değerlendiren bir performans indeksi olarak da görülebilir (Trabelsi ve ark. 2004).

5. MOBİL ROBOT PLATFORMUNUN TASARIMI VE GERÇEKLEŞTİRİLMESİ

Gerçekleştirilen mobil robot platformu, birçok navigasyon ve sensör uygulamalarına temel oluşturmak amacıyla, karasal bir mobil robot yapısında tasarlanmıştır. Şekil 5.1' de blok diyagramı verilmiş olan mobil robotun, üzerinde bulunan mini bir PC sayesinde istenilen elektronik ve sensör donanımı rahatlıkla bağlanabilir ve geliştirilecek navigasyon amaçlı veri toplama yazılımlarını da koşturabilecek kapasitededir.



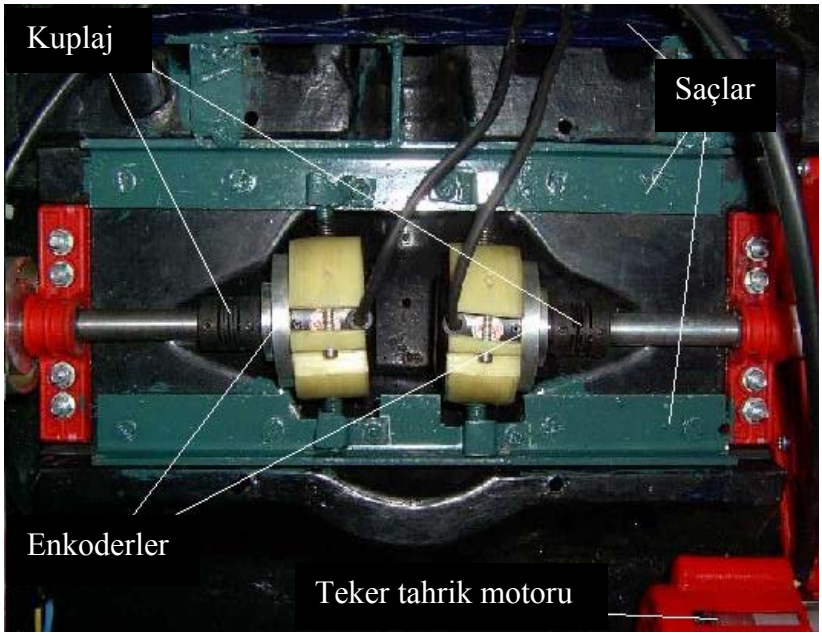
Şekil 5.1 Mobil robot blok diyagramı

Maliyeti düşük tutmak ve tasarım kolaylığı sağlamak amacı ile mobil robotun ana gövdesi olarak piyasada bulunan motorlu bir araç kullanılmıştır. Aracın ana gövdesi ve şasesi tamamen plastik malzemeden imal edilmiştir. Bu çalışmanın amacı doğrultusunda aracın yapısı üzerinde bazı değişiklikler yapılmıştır.

Deneysel çalışmalar sırasında mobil robot kontrolünü sağlayabilmek amacıyla bir RF tele-kontrol sistemi kurulmuştur. Farklı uygulamalar için sistem tamamen otomatik olarak da kendi kendini kontrol edebilecek donanıma sahiptir. Mobil robot sistemi, mekanik, sensör, elektronik donanım ve C dilinde geliştirilmiş yazılımdan oluşmaktadır. Sensör verilerinin işlendiği ve YSA algoritmalarının yürütüldüğü yazılımlar ise MATLAB programlama dilinde gerçekleştirilmiştir.

5.1 Mobil Robotun Mekaniksel Tasarımı

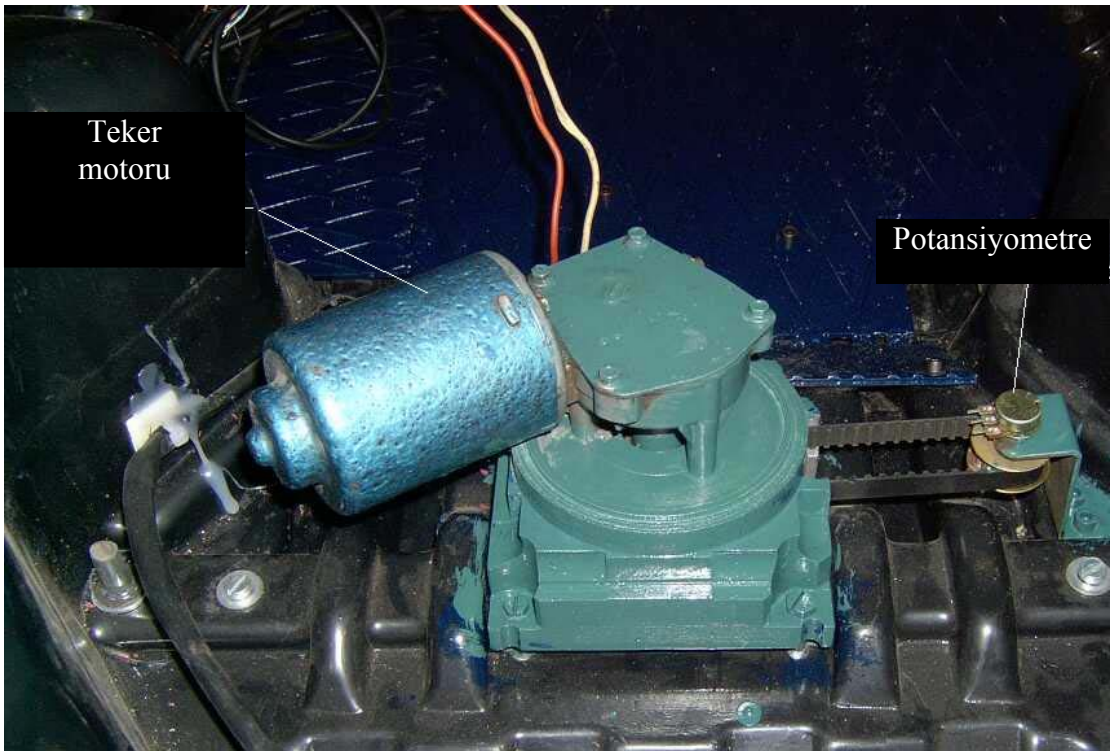
Mobil robot, hazır bir araç üzerinde bazı değişiklikler yapılarak gerçekleştirilmiş ve plastik olan şasenin orta kısımları metal saclarla güçlendirilmiştir. Yük taşıyacak olan arka tekerlerin alüminyum yataklar yardımıyla sabitlenerek esnemeleri önlenmiştir. Önceden birleşik olan arka teker ekseni ortadan kesilmiş, sağ ve sol teker eksenlerine birer enkoder, kuplaj bağlantı elemanları yardımıyla monte edilmiştir. Bu durum Şekil 5.2'de ayrıntılı bir şekilde görülmektedir. Arka tekerlerin tam ortasına ve enkoderleri içine alacak şekilde metal bir kafes ile bu kısımlar kapatılmıştır. Bu kafesin orta kısmına, GPS anten direği monte edilerek sabitlenmiştir. Bütün navigasyon ölçümlerinin referans noktası bu orta nokta alınmıştır. Mobil robotun tahrik motoru, 6V gerilim ile çalışan bir DC motor ve plastik redüktör dişli yardımıyla sağ arka tekere bağlanmıştır. Bu motor, mobil robotu burun doğrultusunda düz zeminde 0.9 m/sn, virajlarda 0.1 m/sn hızlara ulaştırabilmektedir.



Şekil 5.2 Enkoderlerin monte edildiği arka tekerlerin eksenleri

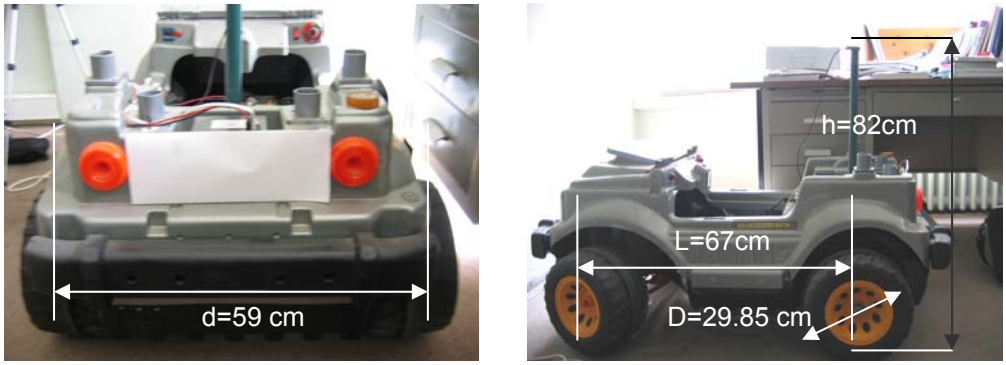
Mobil robotun hız profili, hareket ettiği zeminin yapısına doğrudan bağlıdır. Burada belirtilen hızlar asfalt, yüzeylerde yapılan deneme sürüşleri sonucu elde edilmiştir.

Mobil robota yön veren ön tekerleri kontrol etmek için otomotiv sektöründe kullanılan 12V gerilimle çalışan ve yüksek torka sahip bir motor, orijinal dümen düzeneğinin yerine monte edilmiştir. Mobil robotun ön tekerleri hareket düzleminde ± 45° hareket edebilmektedir. Bu da mobil robotun hareket halindeyken ve ön tekerleri sağa veya sola 45° kırıldıktan sonra 3 metre çapında daireler çizmesini sağlamaktadır. Dümen açısını ölçebilmek için dümen motoruna monte edilen bir dişli kayış yardımıyla lineer bir potansiyometreyi çevirmektedir. Dümen motorunun mekanizması Şekil 5.3' de görülmektedir.



Şekil 5.3 Dümen mekanizması

Mobil robotun fiziki boyutları, enine teker merkezleri arası 59 cm, ön ve arka teker merkezleri arası uzunluğu 67 cm, GPS anteninin yerleştirildiği yerden direk yüksekli 82 cm ve dört tekerin çapı da 29.85 cm' dir. Mobil robotun son monte edilmiş hali ve boyutları Şekil 5.4' deki fotoğraflarda görülmektedir. Fiber optik jiroskop (FOJ) mobil robotun arka tekerlerinin orta ekseninin üzerine yakın bir yere sabitlenmiştir. Burada FOJ, gövdeye bağlı olarak çalışmaktadır



Şekil 5.4 Mobil robotun boyutları

5.2 Mobil Robotta Kullanılan Navigasyon Sensörleri

Fiber optik jiroskop

Mobil robot üzerinde kullanılan FOJ, robot araştırmacılarının sıkça kullandıkları KVH firmasın ürettiği dijital E-Core 1000 jiroskopudur. Bu jiroskop DC 9V ile 18V gerilimleri arasında çalışabilmektedir ve 2W güç tüketmektedir. Jiroskop, ölçtüğü açısal hız verilerini 9600 baud hızında üzerinde bulunan RS232 portu yardımıyla 8 byte veri paketleri şeklinde aktarır ve saniyede maksimum 10 ölçüm gerçekleştirebilmektedir. Her byte' ın en anlamlı biti, veri senkronizasyonu için kullanılmaktadır. Veri paketinin ilk byte' ının en anlamlı biti "1" olur ve veri paketinin başlangıcı bu bit yardımı ile tespit edilir. -100°/sn ile +99.999 °/sn arasındaki açısal hız skalası 16 bit duyarlılıkta kodlanır ve en anlamsız bitin ağırlığı 0.00305 °/saniye' dir. Veri paketinin içinde aynı zamanda jiroskop içinde ölçülen ısı bilgisi de 12 bit duyarlılığında veri paketi içinde kodlanmıştır (KVH 1999). Veri paketi içinde geriye kalan bitlerde ise jiroskopun seri numarası ve paket bilgisinin hatalı iletilip iletilmediğini çözmek için hata düzeltme kodu (CHECKSUM byte' ları) bulunur. Jiroskop' un bazı teknik özellikleri şöyledir; durağan halde 0.2°/sn aralığında bir gürültü ölçer, açısal rasgele yürüyüş değeri ise $20^{o}/\text{saat}/\sqrt{\text{Hz}}$ dir.

Enkoder

Tasarlanan mobil robotta her teker turunda 500 darbe üreten iki adet enkoder kullanılmıştır. Enkoderlerin aynı zamanda her turda bir darbe üreten çıkışları da bulunmaktadır. İleri ve geri yönde hareketlerin algılanabilmesi için aralarında 90° faz

farkı bulunan A ve B darbe çıkışları vardır. Enkoderler 12 V DC gerilimle beslenmektedir ve çıkışta ürettikleri darbe seviyesi de 12 V' tur. Tasarlanan elektronik kartta bu seviyeler 5V TTL seviyesine düşürülmektedir. Enkoder eksenlerini mobil robotun arka teker eksenleri ile birleştirmek için esnek plastikten üretilmiş kuplaj elemanları kullanılmıştır. Yarıçapı $r = 14.925\,cm$ olan bu enkoderlerle elde edilebilecek en küçük teorik mesafe ölçümü için teker çevresi;

$$C = 2\pi r = 2\pi 14.925 = 93.776\,cm$$

olur. Teker çevresi toplam enkoder darbe sayısına bölünerek ölçülebilecek minimum mesafe, yani çözünürlük aşağıdaki gibi bulunur;

$$d = \frac{C}{\text{enkoder darbe sayısı}} = \frac{93.776}{500} cm$$

$$d = 0.1875 cm$$

Mobil robotta kullanılan enkoderler ve zemin ile tekerler arasında oluşabilecek gürültülerin, enkoderlerin darbe sayısının ortalaması alınarak en aza indirilmesi, enkoderin bağlı olduğu tekerde meydana gelebilecek arızalarda ve yanlış ölçümlerde diğer enkoder bilgisi kullanılarak mesafe ölçmede sürekliliğin sağlanması gerekmektedir.

Enkoderli sistemlerde meydana gelen bir başka sorun ise, enkoderin içinde bulunan kızıl ötesi LED'in yaydığı ışığın yarıklı diskten geçerken kırılma meydana getirerek, enkoder çıkışında gürültülü darbe sinyalleri oluşturmasıdır. Bu sinyaller, dijital sayıcı devrelerine uygulandığında sayılması gereken darbelerden çok daha fazlasını saymaktadır. Bu sorun, dijital sayıcılara enkoder sinyalini uygulamadan önce sinyal alçak geçiren bir filtre ile filtrelenerek ya da yazılım yardımıyla bu filtreleme işlemi gerçekleştirilerek giderilir.

GPS

Mobil robota monte edilmiş olan GPS alıcısı, L1 frekansında çalışan SIRFstarIIe alıcısıdır (Sirf Technologie 2000). GPS alıcı, 9V DC gerilim ile beslenmektedir. Bunun için 12V DC' den alınan 9V DC PWM sürücülü güç kaynağı kullanılmaktadır. GPS üzerinde 2 adet RS232 seri port, bir anten konnektörü, bir

reset butonu ve diferansiyel GPS uygulamaları için kısa dalga frekans anten girişi bulunmaktadır.

GPS' in seri portu istenilen baud hızında veri iletişimi sağlayabilmektedir. GPS ASCII formatında istenilen NMEA 0183 v2.01 (National Marine Electronic Association) protokolü üretebildiği gibi, istenirse ham dijital GPS bilgilerini de seri porttan verebilmektedir Bu tez çalışmasında 1 Hz frekansında NMEA 0183 mesajlarında olan GGA formatında veri kullanılmıştır. GGA veri formatının yapısı aşağıda verilmiş olan örnek baz alınarak Çizelge 5.1 de ayrıntılı olarak gösterilmektedir. Diğer NMEA 0183 mesajları ise şöyledir: RMC, GLL, GSA, GSV, MSS, VTG. Bu mesajlar hakkında daha ayrıntılı bilgi NMEA 0183 protokolünde bulunabilir (NMEA 2005). Aşağıdaki örnekte görülen mesajdaki $GP ön eki farklı marka GPS' lerde farklılık gösterebilir.

Örnek GGA mesajı;

$GPGGA,161229.487,3723.2475,N,12158.3416,E,1,07,1.0,9.0,M,,,,0000*18<CR><LF>

Çizelge 5.1 GGA mesaj formatı

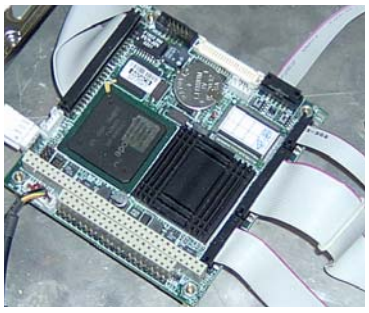
	Kısım	Örnek	Birim	Açıklama
1	Mesaj ID	$GPGGA		GGA mesaj başlığı
2	UTC Zaman	161229.487		hhmmss.sss
3	Enlem	3723.2475		ddmm.mmmm
4	N/S belirteci	N		N=kuzey S=güney
6	Boylam	12158.3416		dddmm.mmmm
7	E/W Belirteci	E		E=doğu W=batı
8	Konum sabitlendi	1		
9	Kullanılan Uydu sayısı	07		0 ile 12 arasında
10	HDOP	1.0		Yatay duyarlılık belirsizliği
11	Yükseklik	9.0	metre	
12	Birim	M	metre	
13	Jeoid ayırımı		metre	
14	Birim	M	metre	
15	Diferansiyel GPS veri zamanı		saniye	DGPS kullanım dışı ise "0" dır
16	Diferansiyel GPS referans istasyon ID	0000		
17	Checksum	*18		Hata düzeltme
18	<CR><LF>			Mesaj sonu

5.3 Mobil Robotun Elektronik Donanımı

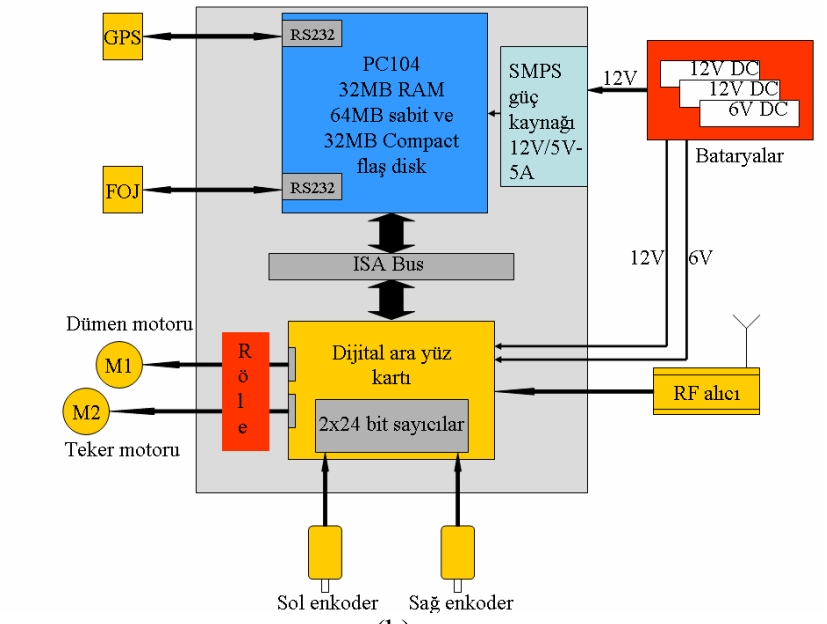
Mobil robotun elektronik donanımı üç kısma ayrılır;

- PC104 bilgisayar kısmı.
- ISA Bus üzerinde çalışan elektronik kart kısmı.
- RF radyo tele kontrol kısmı.

Mobil robotun işlemsel donanımı, yüksek performanslı ve yalnızca 5V DC gerilim ile çalışabilen fansız bir PII 233 MHz PC 104 mini bilgisayardan oluşmaktadır. PC 104' ün üzerinde standart ISA bus giriş/çıkışı, flaş disk yuvası, USB, ağ bağlantı konektörü, 2 adet RS232 seri bağlantı portu, bir paralel port, disket sürücü konnektörü, gömülü VGA grafik kartı, IDE sabit disk portu, klavye ve mouse girişleri mevcuttur. Kullanılan PC 104 kartının fotoğrafı Şekil 5.5.a'da ve sistemin temel şematik yapısı Şekil 5.5.b'de görülmektedir.



(a)



(b)

Şekil 5.5 (a) PC104 ana kart (b) Bilgisayar kontrol sisteminin şematik yapısı

Mobil robot için tasarlanan PC 104 sistemi 32MB RAM bellekten ve düşük güç tüketimi nedeniyle tercih edilen 64MB flaş ve 32 MB Compact marka sabit flaş diskten oluşmaktadır. Sistemde besleme kaynağı olarak 12V DC'den SMPS ile elde edilen 5V/5A DC güç kaynağı kullanılmaktadır.

Şekil 5.5.b' de görüldüğü gibi sistemde güç kaynağı olarak iki adet 12V/7Ah ve bir adet 6V/7Ah batarya bulunmaktadır. 12V bataryanın biri elektronik sistemi beslemek için, diğeri dümen motorunu sürmek için, 6V' luk batarya ise mobil robotu tahrik eden DC motoru sürmek için kullanılmıştır. Dümen ve tahrik motorunun ileri ya da yönde dönmesi, motorun bağlantı uçlarının yönünün röle kombinasyonundan oluşan ayrı anahtarlama devrelerinin kontrolü ile gerçekleştirilmiştir.

PC104, üzerinde bulunan RS232 seri port üzerinden GPS ve FOJ'un ürettiği verileri almaktadır. Enkoder ve mobil robotun durum verilerini almak için PC104 ISA bus'ına bağlanabilecek genel amaçlı dijital bir kart tasarlanmıştır. PC104 standardının normal PC bilgisayarların birebir bütün özelliklerini taşıdığı göz önüne alınırsa, kullanıcılar için boş bırakılan 0x300H ile 0x31F arasındaki 0x300H ve 0x30A arasındaki adres bölgeler tasarlanan kartın erişim bellek bölgesini oluşturacak şekilde ayarlanmıştır.

Mobil robotun arka tekerlerine bağlanan enkoderler tarafından üretilen darbeleri saymak üzere 24 bitlik iki sayıcı tasarlanmıştır. Bunun için, 12V gerilim seviyesinde olan enkoder sinyalleri sayıcılara uygulanmadan önce 5V TTL seviyesine düşürülmüştür. Bu işlemi gerçekleştirmek için enkoder sinyalleri buffer olarak görev yapan ortak kolektörlü NPN transistörlerin bazına uygulanmıştır. Transistörün kolektöründe fazı 180° kaydırılan sinyal, bir NOT kapısından geçirilerek seviyesi düzenlenmiş sinyal ile 12V seviyesinde olan enkoder sinyali aynı faza getirilmiştir. NOT kapısının çıkışı kaskat bağlanmış iki adet 12 bit sayıcıya uygulanmaktadır.

Enkoder üzerinde bulunan A, B ve X çıkışları için yukarda belirtilen seviye ve faz düzeltme işlemleri gerçekleştirilmiştir, fakat X sinyalleri yazılımda dikkate alınmamıştır. Enkoderlerin A ve B sinyalleri arasında 90° faz farkı bulunmaktadır. Bu iki sinyal D tipi bir flip-flop kullanılarak tekerin dönme yönü tespit edilmektedir. Flip-flopun çıkışı "0" ise geri, "1" ise ileri yönü belirtmektedir. Sayıcıların çıkışları,

Çizelge 5.2'den görüldüğü gibi 0x300H ile 0x305H arası adreslerinden okunur. 0x301H ve 0x303H adreslerinden, 24 bit sayıcı gruplarının üst ve alt 12 bitlerinin üst 4 bitleri okunur. Okunan bu veriler daha sonra PC104 üzerinde koşan yazılım tarafında 24 bitlik tam sayılara dönüştürülür ve gerekli ölçeklendirmeler yapılır.

Çizelge 5.2 Tasarlanan ISA bus kartının port adres haritası

Adres Alanı	Açıklama							
0x300H Giriş	Sağ enkoder 24 bit sayıcının alt 12 bitin alt 8 biti							
	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
0x301H Giriş	Üst 12 bitin üst 4 biti				Alt 12 bitin üst 4 biti			
	Q23	Q22	Q21	Q20	Q11	Q10	Q9	Q8
0x302H Giriş	Sağ enkoder 24 bit sayıcının üst 12 bitin alt 8 biti							
	Q19	Q18	Q17	Q16	Q15	Q14	Q13	Q12
0x303H Giriş	Sol enkoder 24 bit sayıcının alt 12 bitin alt 8 biti							
	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
0x304H Giriş	Üst 12 bitin üst 4 biti				Alt 12 bitin üst 4 biti			
	Q23	Q22	Q21	Q20	Q11	Q10	Q9	Q8
0x305H Giriş	Sol enkoder 24 bit sayıcının üst 12 bitin alt 8 biti							
	Q19	Q18	Q17	Q16	Q15	Q14	Q13	Q12
0x306H Giriş	8 bit genel amaçlı giriş							
	~~D7~~	~~D6~~	D5	D4	D3	D2	D1	D0
0x307H Giriş	8 bit genel amaçlı giriş							
	~~D7~~	~~D6~~	~~D5~~	~~D4~~	~~D3~~	~~D2~~	~~D1~~	~~D0~~
0x308H Giriş	ADC808 8 bit giriş							
	D7	D6	D5	D4	D3	D2	D1	D0
0x309H Çıkış	Genel amaçlı 8 bit çıkış							
	~~D7~~	~~D6~~	D5	D4	D3	D2	~~D1~~	~~D0~~
0x30AH Çıkış	ADC808 kanal kontrol 8 bit çıkış							
	~~D7~~	~~D6~~	~~D5~~	~~D4~~	~~D3~~	A2	A1	A0

Mobil robotun tahrik ve dümen motorlarını, enkoderlerini, tele kontrol sistemini test etmek ve bu sistemlerin sağlıklı çalıştığını algılamak için 0x306H adresi 8 bit giriş olarak tanımlanmıştır. 0x306H adresinin tele-kontrol alıcı sisteminin, vericiden gelen emirleri algılayıp algılamadığını kontrol etmek için D0-D3 bitleri kullanılmıştır. D0 ve D1 bitleri tele-kontrol vericisinde sırasıyla ileri ve geri anahtarlara basılmasını algılar. D2 ve D3 bitleri'de sırasıyla sağ ve sol anahtarlarını algılar. Bu bitlerde "1" basıldı, "0" ise basılmadı anlamına gelir.

D4 ve D5 bitleri sırasıyla arka sağ ve sol tekerlerin ileri geri dönüşünü algılar, bu bitler yukarda bahsedilmiş olan flip-flopların çıkışlarıdır. 0x307H adresi boş bırakılmıştır. 0x308H ve 0x30A adresleri 8 kanal analog giriş destekleyen ADC808 için ayrılmıştır. 0x308 adresi ADC' nin analogdan dijitale dönüştürdüğü veriyi okumak ve çıkış olarak tanımlanan 0x30A adresin ilk 3 biti ADC'nin analog giriş kanallarını seçmek için kullanılmıştır. Diğer kalan bitlere hiç bir fonksiyon verilmemiştir.

Önceden de bahsedildiği gibi PC104 için tasarlanan kart, mobil robotun yazılımla da bazı fonksiyonlarını kontrol edebilmektedir. Bunun için çıkış olarak tanımlanan 0x309H adresi kullanılmaktadır. Bu adresin bitlerine verilen fonksiyonlar şöyle sıralanabilir;

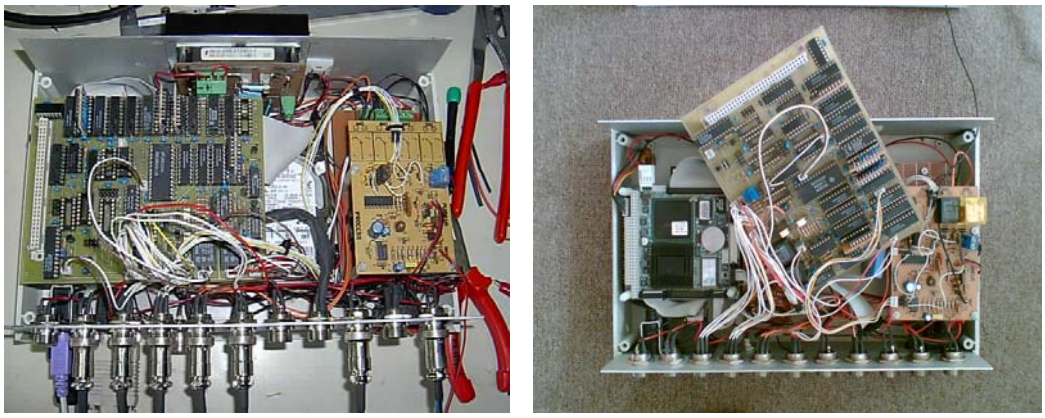
D0-D1: boş
D2: "1" ise teker motoru ileri çevrilir, "0" ise durdurulur
D3: "1" ise dümen motoru sağa çevrilir, "0" ise durdurulur
D4: "1" ise dümen motoru sola çevrilir, "0" ise durdurulur
D5: "1" ise teker motoru geri çevrilir, "0" ise durdurulur
D6-D7: boş

Bu port adresi, mobil robotun motor sistemlerinin yazılımla kontrolü için deneysel çalışmalar başlamadan önce ve sonra test amaçlı kullanılmıştır. Şekil 5.6.a' daki fotoğraflarda dijital ISA kart ve PC104 bilgisayar kartı görülmektedir. Tasarlanan ISA kartın açık devre şeması EK-B verilmiştir.

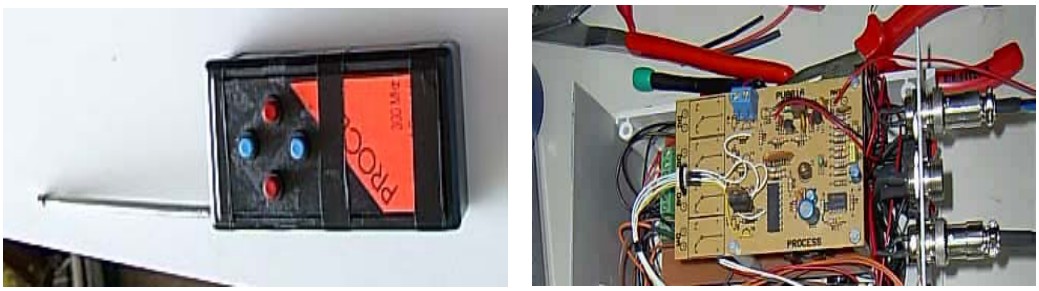
Mobil robot sisteminden sensör verilerini toplama aşamasında, önceden belirlenmiş veya çalışma anında karar verilen güzergahları takip etmek için mobil robot tele-kontrol verici ve alıcısı kullanılmıştır. Şekil 5.6.b' de görülen RF alıcı-verici devreleri 300 MHz frekansında (verici kısmı elde taşınabilen) ve 12V DC gerilim ile çalışmaktadır. RF vericinin üzerinde, dört adet anahtar vardır ve bu anahtarlar ile mobil robotun ileri, geri, sağa ve sola hareketi sağlanır. Her anahtar ayrı ayrı kodlanarak modüle edilmektedir. RF sinyali havada yayılım sırasında oluşabilecek gürültü/sinyal oranını artırmak için kodlanmaktadır.

Elektronik donanım, plastik kutu içine yerleştirilmiş ve dış dünya ile olan bağlantılar kutuya monte edilen konnektörler aracılığı ile gerçekleştirilmiştir.

Elektrik beslemesi, seri iletişim bağlantıları, klavye, sensör beslemeleri ve veri hatları için birer konnektör kullanılmıştır. RF alıcı, kutunun üzerine yerleştirilen anten ile iletişim kurmaktadır.



(a)

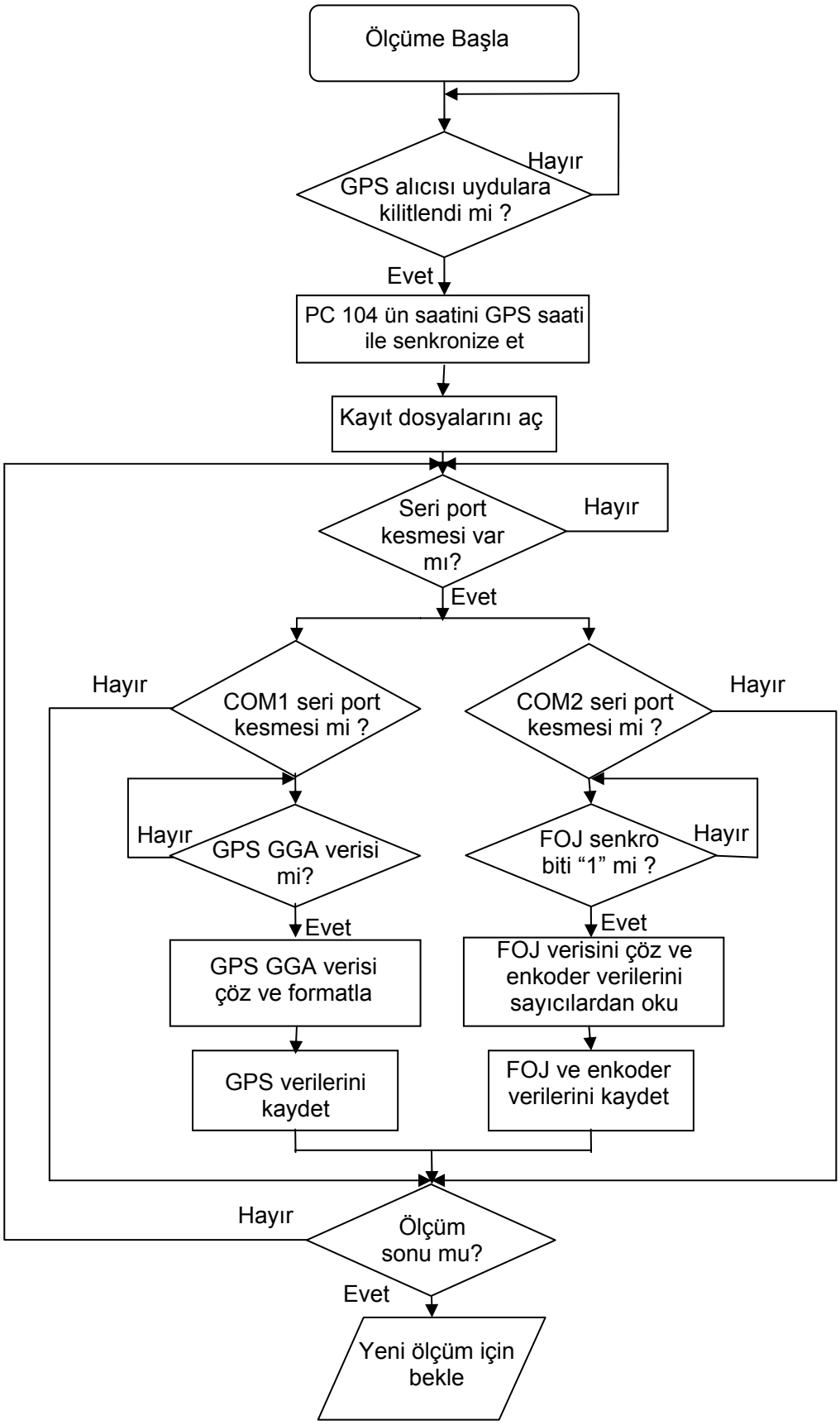


(b)

Şekil 5.6 Elektronik donanım (a) ISA G/Ç kartı (b) ISA ve PC104 kartı (c) RF verici (d) RF alıcı

5.4 Mobil Robot Veri Toplama Yazılımı

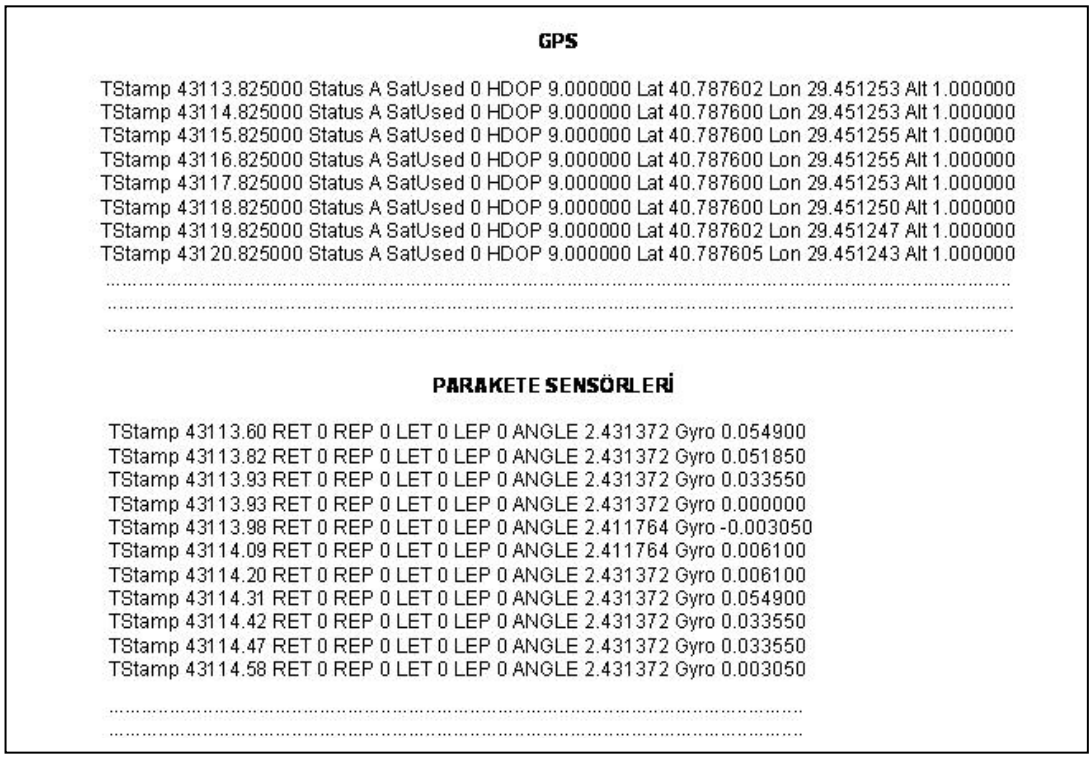
Mobil robot yazılımı sırasıyla; GPS, FOJ, 24 bit'lik sayıcıların saydığı enkoder darbeleri ve PC104 saat kayıtlarını gerçek zamanda kaydeder. Şekil 5.7 mobil robot yazılımının temel akış diyagramını göstermektedir.



Şekil 5.7 Mobil robot PC104 yazılımı akış diyagramı

Yazılım DOS ortamında ve Borland C++ 3.1 programlama dilinde yazılmıştır. Ana yazılım robot.c tarafından çağrılan kütüphanelerden: driver.c, navlib.c ve RS232 kütüphanesi olan PCL4C.lib' dir. ISA bus' ın kontrolünü ve ISA kart üzerindeki çiplerin ilk durum değerlerinin atanmasını driver.c kütüphanesi yapmaktadır. Bu kütüphane aynı zamanda geliştirilmiş olan genel amaçlı ISA bus kartının sürücü kütüphanesidir. Navigasyon verilerinin formatlanması, verilerin senkronizasyonu ve sabit diske saklama işlevini yürüten navlib.c' dir. GPS ve FOJ verilerinin RS232 bağlantı portundan okunmasını ve RS232 bağlantı portlarının gerçek zamanda kesme yöntemi ile çalışmasını yürüten PCL4C.lib aynı zamanda PC104 ortamına veri aktarım sırasında hiç bir veri kaybı olmamasından sorumludur. Mobil robot yazılımının bütün C kod dökümü EK-C' de verilmiştir.

Navlib.c iki adet kayıt dosyası oluşturur. Bunlardan biri, 1 Hz'de üretilen GPS verilerinin kayıtları, diğeri 10 Hz' de üretilen FOJ ve enkoder veri kayıtlarıdır. Bu dosyadaki veriler parakete sensör kayıtları olarak da adlandırılabilir. Kayıt anında her ölçüm için farklı isimlerde yeni dosyalar açılmaktadır. Örnek GPS ve parakete sensör kayıtları Şekil 5.8' de görülmektedir.



Şekil 5.8 GPS ve parakete sensör kayıtları

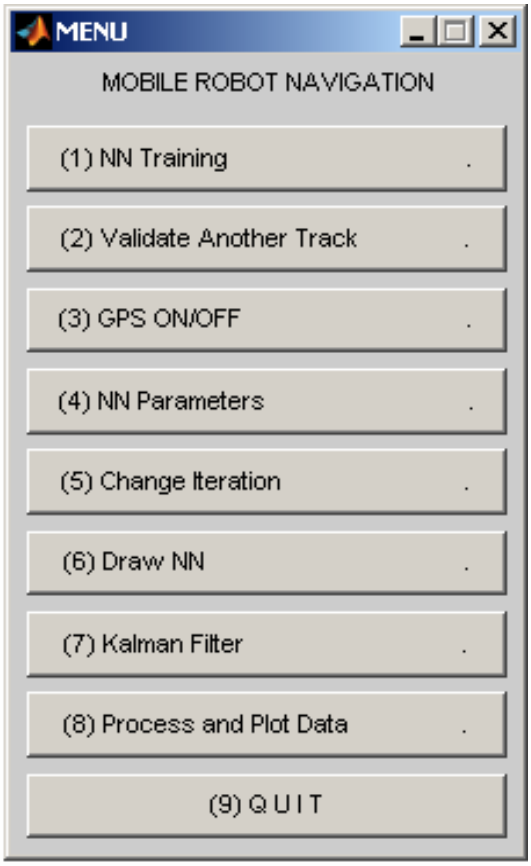
TStamp, GPS ve parakete sensör kayıtlarının o satırda bulunan verilerinin kayıt zamanını gösterir. Buradaki zaman birimi saniyedir. Her iki dosyada kayıt zamanları arasındaki fark bir saniyeyi aşmayacak şekilde tutulur. GPS kayıtlarında Status, GPS uydularının yeterli sayıda olduğunu ve bir konum verisi üretebildiğini gösteren bir bayraktır. Bu bayrak "A" durumunda ise: konum verisi üretilmektedir, "V" durumunda ise: konum verisi üretilmemektedir. HDOP, GPS tarafından üretilen konum bilgisinin kalitesi hakkında fikir veren bir değerdir. Lat ve Lon; GPS alıcısının hesapladığı enlem ve boylam değeridir. Yazılımın navlib.c kütüphanesi tarafından, GPS' ten alınan veriler, derece-desimal formatına dönüştürüldükten sonra dosyaya kaydedilir. Parakete sensör kayıtlarında RET ve REP sırasıyla; sağ arka tekerin tam teker tur sayısını ve sağ arka teker enkoderinin örnekleme zaman aralığında ürettiği darbe sayısını belirtmektedir. LET ve LEP sol arka teker için tur ve darbe sayılarıdır. ANGLE kalibre edilmemiş dümen açısı bilgisidir. Gyro; FOJ açısal hız değerleridir. PC104 tarafından kaydedilen dosyalardaki veriler tamamen ham verilerdir. Herhangi bir filtreleme ve ölçeklendirme işlemine tabi tutulmamaktadır. Bu işlemler MATLAB ortamında off-line olarak gerçekleştirilmiştir.

5.5 MATLAB Yazılımları

Mobil robot navigasyon verilerini işleyen ve öngörülen algoritmaların simülasyonu, PC ortamında çalışan esnek ve kullanışlı olan MATLAB platformunda yazılmış olan programlarla gerçekleştirilmiştir. MATLAB yazılımları, veri önişleme ve ham verileri hızlı gösterme, Kalman filtresi ve YSA modelini oluşturan algoritmalardan oluşmaktadır. Bunların yanında MATLAB DSP, NNSYSID ve Map alet kutularından da yaralanılmıştır. Burada sayılan tüm yazılım kısımlarının kullanımını ve verilerin tekrarlı simülasyonlarını gerçekleştirirken kolaylık sağlaması için tüm sistem düğmelerden oluşan görsel bir ara yüz altında toplanmıştır.

5.5.1 Görsel ara yüz

Yazılmış olan bütün MATLAB program parçaları Şekil 5.9' da verilmiş olan görsel mönü ara yüzü altında toplanmıştır. Mönüdeki düğmelerin bazıları doğrudan ilgili alt programı yürütür, geriye kalan düğmeler ise YSA parametrelerinin ayarlanması için kullanılır.



Şekil 5.9 MATLAB programlarını toplayan görsel ara yüz

Mönü fonksiyonları sırasıyla aşağıdaki özelliklere sahiptir;

i. **NN training:** YSA algoritması için eğitim güzergah dosyası seçilir ve YSA eğitim döngüsü başlatılır. Eğitimin sonunda grafikler ekrana çizilir ve simülasyonun sonucu bir rapor dosyasına kaydedilir.

ii. **Validate Another Track:** Eğitilmiş olan YSA, seçilebilir farklı güzergah verileri ile test edilir, Sonuçların grafikleri çizilir ve raporlanır. Farklı parametreler ile tekrarlanan simülasyonlar için ayrı ayrı dizinler oluşturulur, grafikler ve sonuç raporu buraya kaydedilir.

iii. **GPS ON/OFF:** Kalman filtresinde ve eğitilmiş YSA' da GPS veri desteğini aktif veya pasif duruma getirir.

iv. **NN Parameters:** YSA-NARX modelin giriş-çıkış sayısı, gecikme sayısı ve gizli katmandaki nöron sayısı ayarlanır

v. **Change Iteration:** YSA eğitim iterasyon sayısı değiştirilir.

vi. **Draw NN:** Oluşturulan YSA yapısı ekrana çizilir.

vii. **Kalman Filter:** Kalman filtresi algoritmasını koşturarak konum kestirimi yapar. Filtre ile yapılan simülasyon sonuçları grafiklerle gösterilir, sonuçlar raporlanır ve saklanır.

viii. **Process and Plot Data:** Mobil robot bilgisayarı tarafından toplanan ham GPS ve parakete sensör verilerini okur, GPS ile parakete sensör verilerinin eş zamanlı olmasını sağlar ve ikinci dereceden Butterworth filtreden geçirdikten sonra sonuçları grafikler halinde gösterir ve her bir işlem ayrı ayrı dizinlerde saklanır. Bu veriler GPS ve parakete sensör verileri ile birlikte *.data uzantılı dosyalar haline dönüştürülür. Bu dosyalar daha sonra Kalman filtresi ve YSA simülasyonlarında kullanılır.

ix. **Quit:** Programdan çıkış

6. MOBİL ROBOT KONUM KESTİRİM SİSTEMİ UYGULAMALARI VE SONUÇLARI

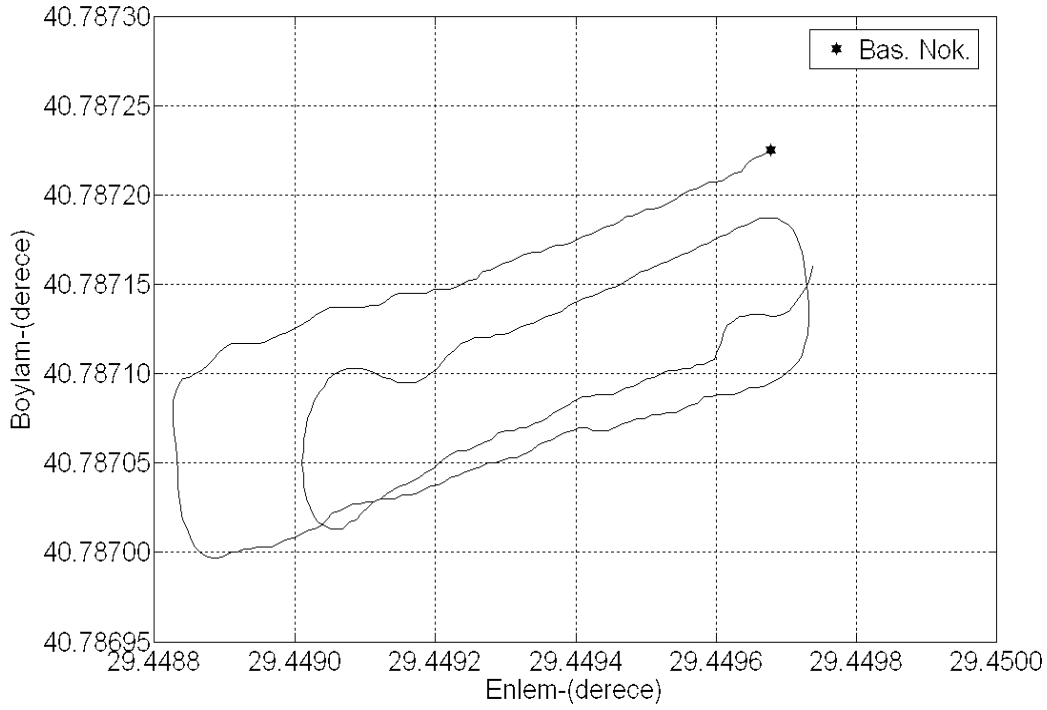
6.1 YSA Eğitimi ve Test Güzergahlarının Belirlenmesi

YSA ve Kalman filtresi navigasyon algoritmaları kullanılmadan önce, mobil robot üzerindeki PC104 bilgisayar birimi tarafından toplanan ham sensör verilerinin bir ön işleme tabi tutulması gerekir. Bu nedenle ham veriler filtreleme, ölçeklendirme gibi ön işlemlerden geçirildikten sonra navigasyon algoritmalarına uygulanmıştır. Çizelge 3.2' deki kıstaslar temel alınarak GPS' ten elde edilen bilgilerin en doğru konum bilgilerini verdiği kabul edilmiş ve bütün karşılaştırmalar GPS konum bilgisine göre yapılmıştır. GPS verilerinin, mobil robot hareket halindeyken ±2 m ve durağan halde ±10 m doğruluk seviyelerinde olduğu görülmüştür. Kullanılan GPS' in, kendi içinde bulunan özel bir filtreleme algoritması yardımıyla hareket halinde iken konum hatası azaltılmaktadır.

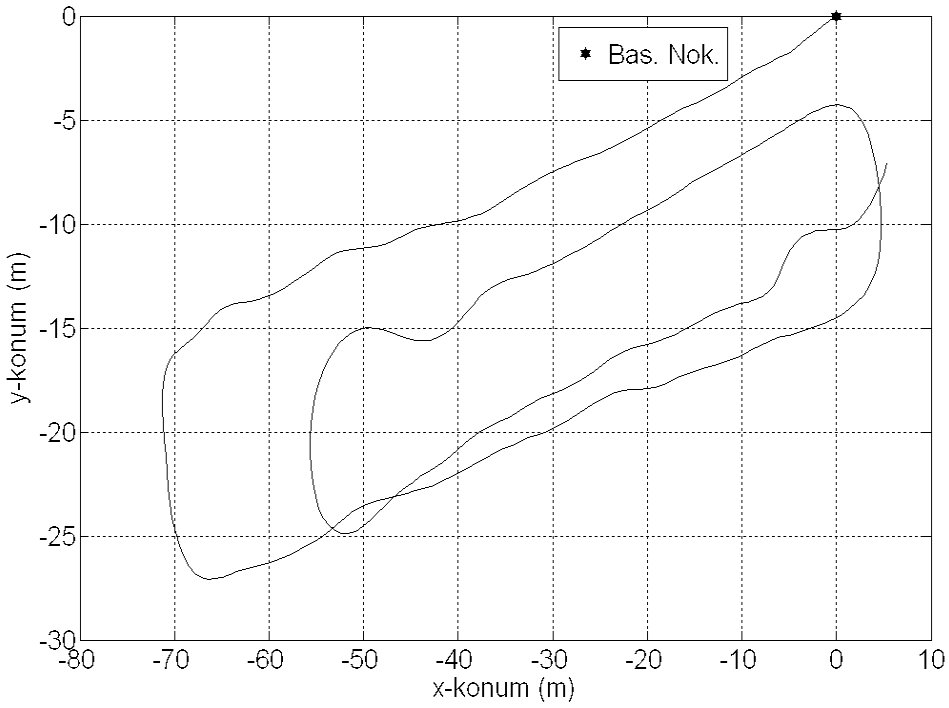
YSA' yı eğitmek için uygun bir eğitim setine ihtiyaç vardır. Eğitim setinin verilerini oluşturmak için, özel olarak oluşturulan bir eğitim güzergahı tespit edilmiştir. Bu güzergahla ilgili ayrıntılı tartışma daha sonraki bölümlerde verilmiştir.

Gerçekleştirilen Kalman ve YSA algoritmalarını test etmek için Şekil 6.1 ve Şekil 6.2' de görüldüğü gibi G_1 ve G_2 olarak adlandırılan iki test güzergahı oluşturulmuştur. Bu güzergahlar Şekil 6.1.a ve Şekil 6.2.a' da coğrafi (enlem, boylam) , Şekil 6.1.b ve Şekil 6.2.b' de UTM koordinat sistemlerinde verilmiştir.

G_1 güzergahı, düz asfalt zeminden ve G_2 güzergahı da pürüzlü beton zeminden oluşmaktadır. Mobil robotun güzergahlardaki seyir süreleri, G_1 için 366 sn, G_2 için 437 sn' dir. YSA ile test güzergahlarının simülasyonlarında, UTM koordinat sistemine dönüştürülmüş GPS konum verileri kullanılmıştır. G_1 güzergahı düz hareket, 90 derecelik dönüş ve bir yarı dairesel dönüşten oluşmaktadır. G_2 güzergahı ise biraz daha karmaşık geometrilerden oluşmaktadır. Oluşturulan bu iki güzergahla, eğitilen YSA' nın genelleştirme ve eğitim seti verilerinden farklı verilere karşı davranışı gözlenmiştir.

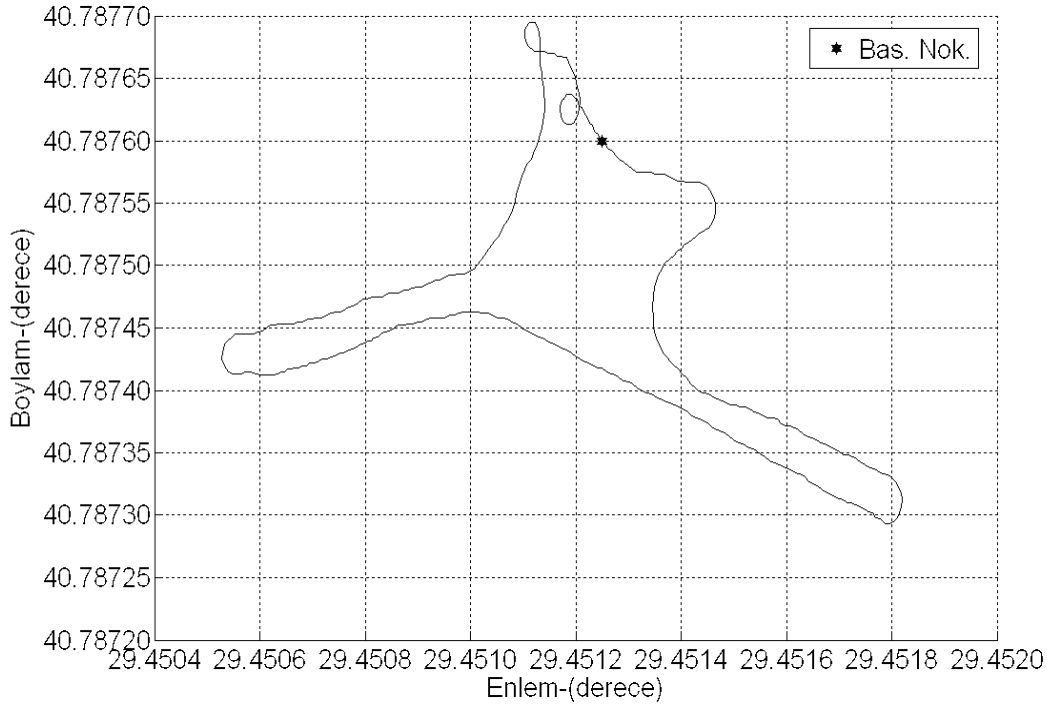


(a)

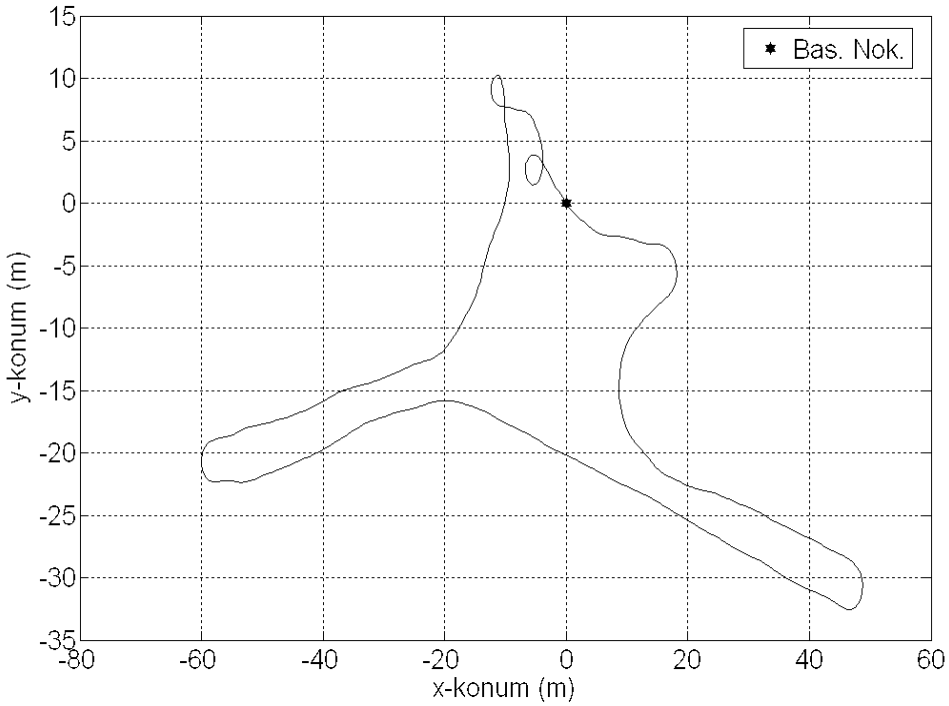


(b)

Şekil 6.1 (a) GPS ile elde edilen coğrafik G_1 test güzergahı ve (b) UTM koordinat sistemine dönüştürülmüş hali



(a)



(b)

Şekil 6.2 (a) GPS ile elde edilen coğrafik G_2 test güzergahı ve (b) UTM koordinat sistemine dönüştürülmüş hali

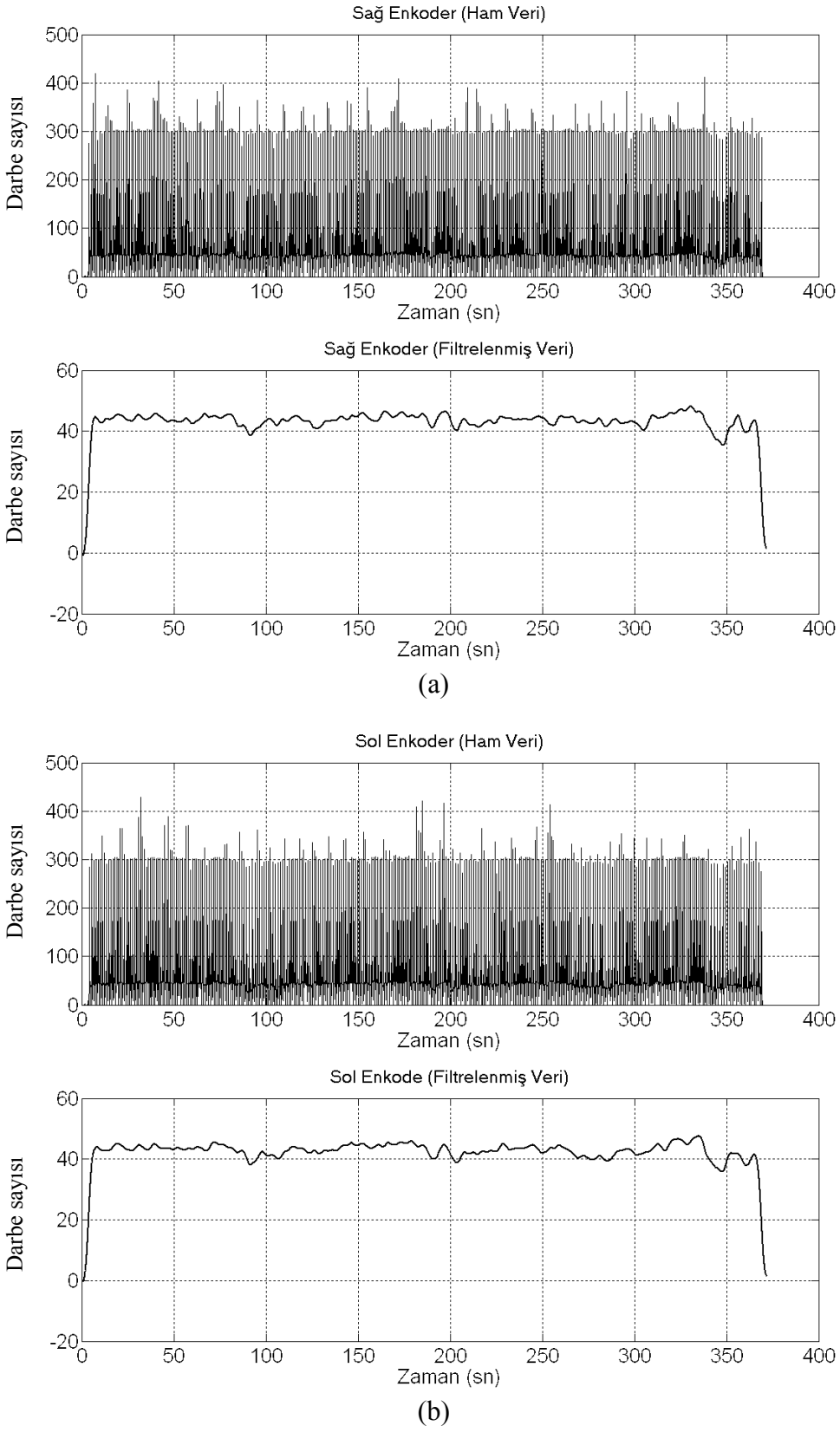
6.2 Mobil Robot Sensör Verilerinin İncelenmesi

Mobil robotun, G_1 ve G_2 güzergahlarına ait enkoder ve jiroskop verileri Şekil 6.3, Şekil 6.4 ve Şekil 6.5'de grafikler şeklinde verilmiştir. Oluşturulan bu iki güzergahta sensör verilerinde oluşan gürültü karakteristiklerinin birbirinden faklı olduğu gözlenmiştir. Şekli 6.3 ve Şekil 6.4' de görülen G_1 ve G_2 güzergahına ait sağ ve sol teker ham enkoder verilerinin gürültünün içinde boğulmuş olduğu görülmektedir. Bu gürültüler, mobil robot hareket halindeyken bulunduğu zeminin arka tekerlerde meydana getirdiği titreşimlerden, enkoder içinde bulunan kızıl ötesi optik elemanın karasızlığından ve mobil robotu hareket ettiren motorun elektriksel gürültüsünden kaynaklanmaktadır.

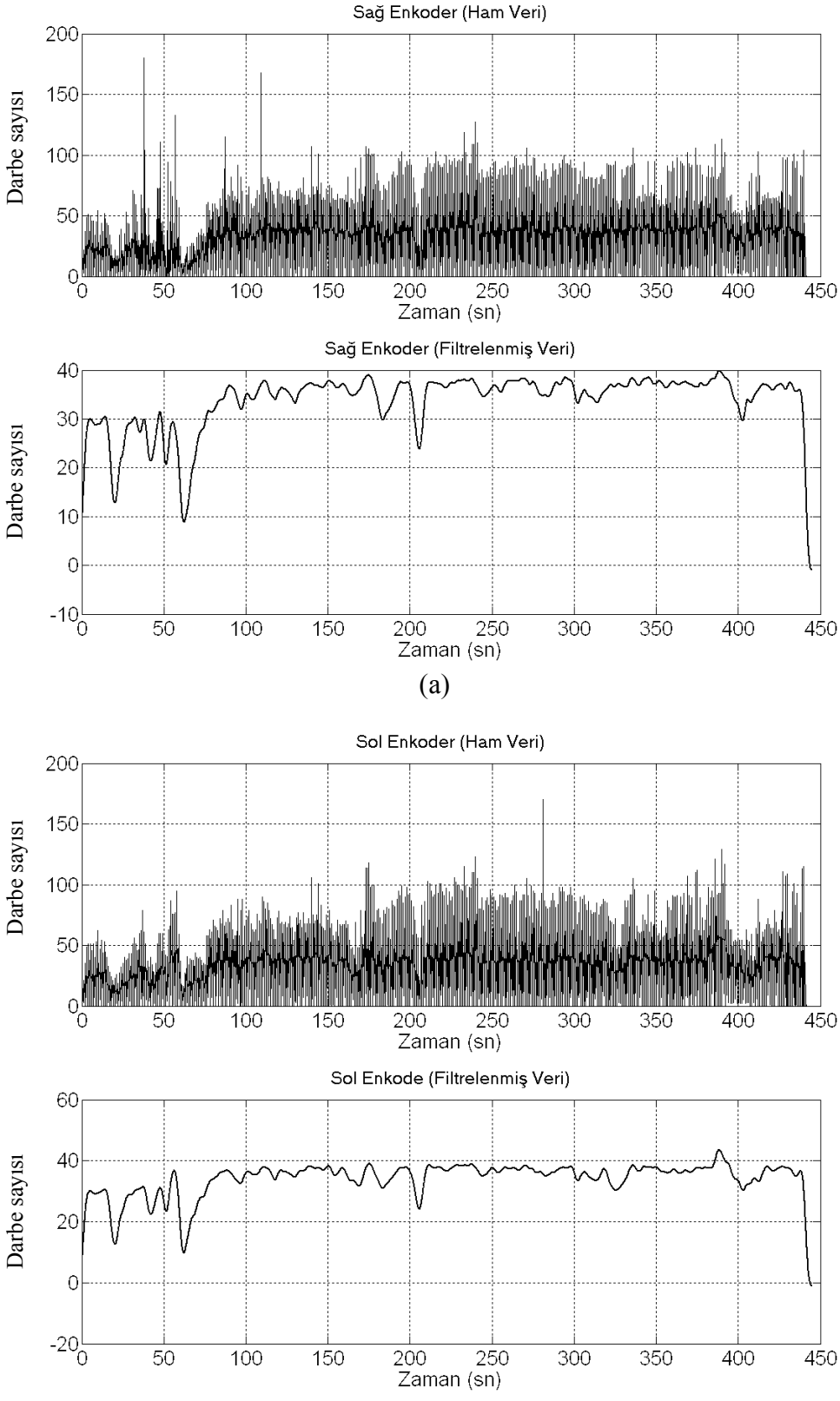
Şekil 6.3'de enkoderlere etki eden gürültünün daha çok elektriksel sebeplerden kaynaklandığı söylenebilir. Enkoder darbe sayıcılarının 0 ile 300 sayılarında neredeyse sabit belirli aralıklarla doyuma ulaştığı görülmektedir. Şekil 6.4'de enkodere etki eden gürültülerin zeminden ve enkoder sensörlerinin opto-elektronik karasızlığından kaynaklandığı söylenebilir. Gürültülü ham enkoder verileri ikinci dereceden Butterworth filtreye uygulandıktan sonra yüksek frekanslardan oluşan gürültüler giderilmiştir. Filtrelenmiş enkoder verilerine ilişkin değişimler Şekil 6.3 ve Şekil 6.4' de görülmektedir.

FOJ sensör verilerinin, enkoder sensörlerine göre daha kararlı, elektriksel gürültüye daha dayanıklı ve gürültü sinyal oranının yüksek olduğu Şekil 6.5' de görülmektedir.

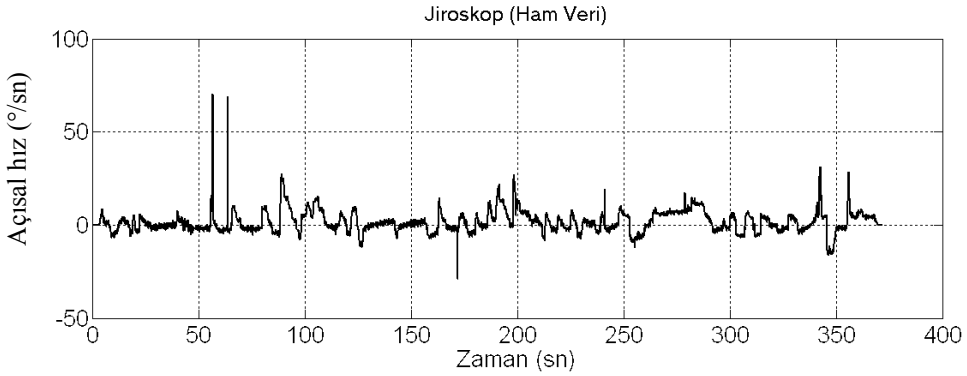
G_1 ve G_2 güzergahlarına ait FOJ' daki gürültü, zemin yapısından, mobil robotun seyri sırasındaki ani dümen manevralarından, geçici mekanik rejimlerden ve mobil robotun ön dümen aksamında bulunan dümen boşluğundan kaynaklanmaktadır. Şekil 6.5.a' da ham veride oluşan ani sıçramalarla yer yer -30°/sn' ile +75 °/sn' lik açısal hız değişimleri görülmektedir. Bu ani açısal hız değişimleri filtrelendikten sonra giderilmiştir.

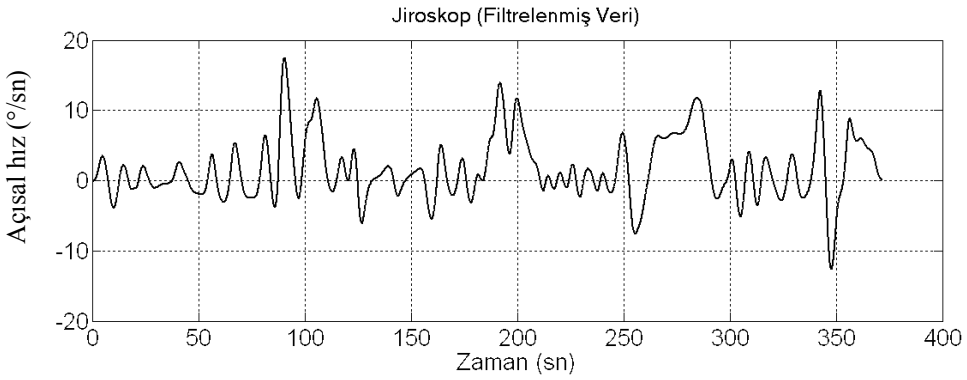


Şekil 6.3 G_1 güzergahın dan elde edilen (a) ham ve (b) filtrelenmiş sağ ve sol teker enkoder verileri

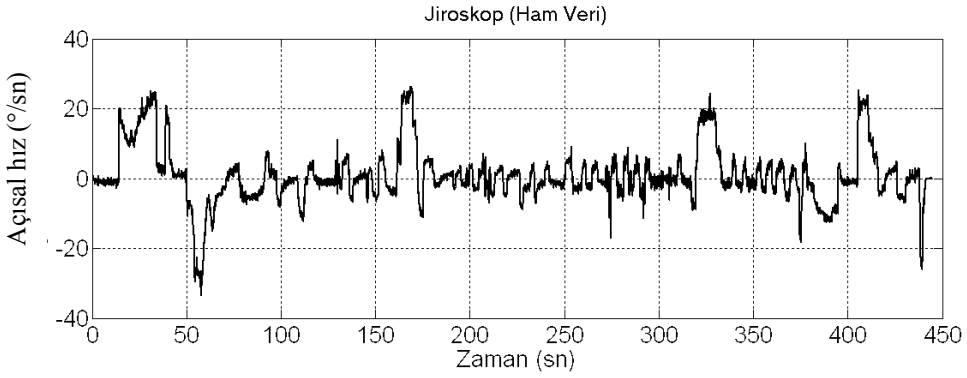


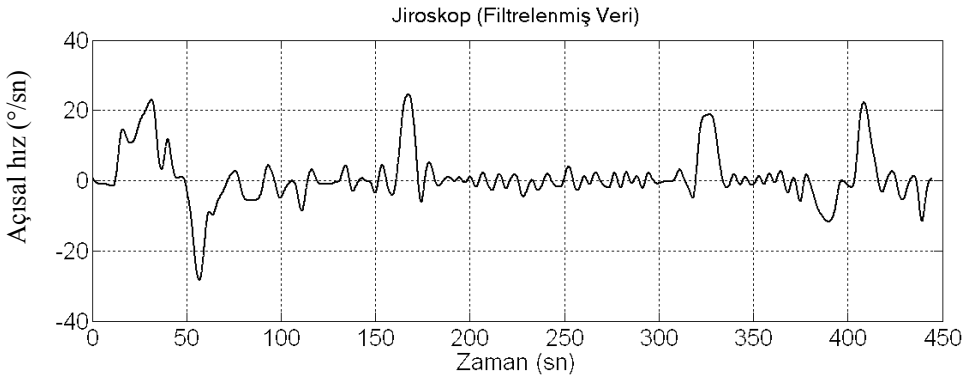
Şekil 6.4 G_2 güzergahın dan elde edilen (a) ham ve (b) filtrelenmiş sağ ve sol teker enkoder verileri





(a)





(b)

Şekil 6.5 (a) G_1 ve (b) G_2 güzergahına ait ham ve filtrelenmiş FOJ verileri

Mobil robotun, verilmiş olan güzergahlardaki GPS ve enkoder sensörlerinden elde edilen hız, FOJ ve GPS' ten elde edilen açısal hız ve açı değerleri için hesaplanmış bazı istatistiksel sonuçlar Çizelge 6.1 ve Çizelge 6.2' de verilmiştir. G_1 güzergahı için, mobil robotun enkoderlerden ölçülen hız aralığı 0 m/sn ile 0.90 m/sn' dir ve Çizelge 6.1'de de görüldüğü gibi bu hız aralığı G_2 güzergahında elde edilen hız aralığını kapsamaktadır.

Çizelge 6.1 aynı zamanda GPS ve enkoder hızlarının karşılaştırılmasına imkan vermektedir. Burada, enkoder ile yapılan hız ölçümlerinin başarılı olduğu sonucuna varılabilir.

Çizelge 6.1 Mobil robotun farklı güzergahlardaki istatistiksel hız verileri

	G1	G2
Enkoder ortalama hız (m/sn)	0.80	0.64
GPS ortalama hız (m/sn)	0.80	0.63
Enkoder hız standart sapma (m/sn)	0.03	0.09
GPS hız standart sapma (m/sn)	0.10	0.14
Enkoder minimum hız (m/sn)	0.00	0.21
GPS minimum hız (m/sn)	0.00	0.20
Enkoder maksimum hız (m/sn)	0.90	0.76
GPS maksimum hız (m/sn)	0.92	0.83

Çizelge 6.2'de, FOJ ile G_1 ve G_2 güzergahında -27.78 °/sn ile +26.96 °/sn aralığında oluşan açısal hız değişimleri ölçülmüştür. Bu ölçülen değerler aralığını kapsayacak şekilde YSA eğitim güzergahı oluşturma yoluna gidilmiştir. YSA'nın eğitimi ile ilgili güzergahtan elde edilen değerler Bölüm 4.9' da bahsedilen eğitim setinin boyut şartını sağlayacak şekilde oluşturulmuştur. Minimum ve maksimum GPS ve Parakete sensör değerleri, mobil robotun farklı güzergahlardaki genel hız ve dönme performansını belirlemektedir

Yapılan gözlemler sonucu, hareket durumundaki mobil robotun hızı, dümen açısı, zemin yapısı, zemin eğimi ve batarya şarj seviyesine göre değişmektedir. Mobil robot, düz bir çizgi takip etmesi durumunda ulaşabileceği maksimum hızına

yaklaşmaktadır. Bu bilgiler ışığında ve Çizelge 6.1 ve Çizelge 6.2' deki maksimum açı ve hız değerleri yardımıyla YSA' nın eğitiminde kullanılmak üzere güzergah modeli oluşturulmuştur.

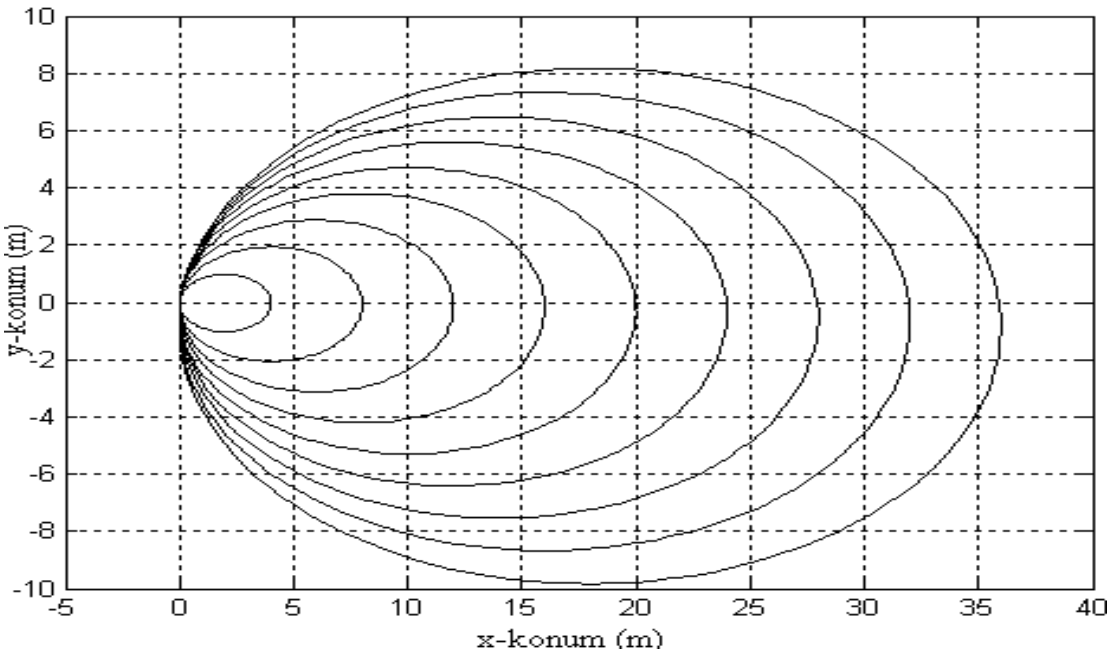
Çizelge 6.2 Mobil robotun farklı güzergahlardaki istatistiksel FOJ ve GPS verileri

	G1	G2
FOJ ortalama açısal hız (°/sn)	1.61	0.76
GPS ortalama açısal hız (°/sn)	-0.06	0.02
FOJ açısal hız standart sapması (°/sn)	4.31	7.19
GPS açısal hız standart sapması (°/sn)	17.01	8.92
FOJ ortalama açı (°)	-1.02	0.17
GPS ortalama açı (°)	-1.06	0.20
FOJ açı standart sapması (°)	1.69	1.96
GPS açı standart sapması (°)	1.69	1.94
FOJ minimum açısal hız (°/sn)	-11.80	-27.78
FOJ maksimum açısal hız (°/sn)	17.32	26.96
GPS minimum açısal hız (°/sn)	-163.65	-64.86
GPS maksimum açısal hız (°/sn)	71.61	19.73

6.3 Yapay YSA Eğitim Güzergahının Oluşturulması

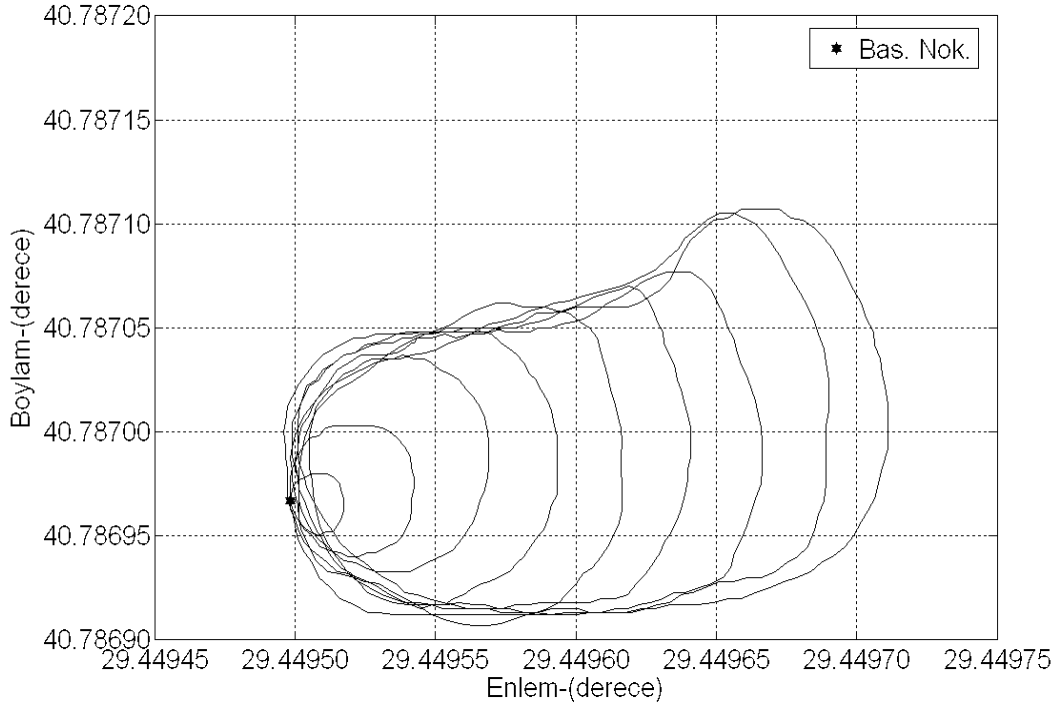
YSA' nın verimli eğitilebilmesi için toplanacak verinin boyutunun optimal olduğu ve kolayca tekrarlanabilen belirli şablona sahip bir güzergaha ihtiyaç vardır. Bu güzergahı oluşturmak için (3.21), (3.22) ve (3.23)' de verilmiş olan temel parakete denklemleri kullanılmıştır. Bu denklem takımında ϕ *[-π,+π] rad* ve V_{ort} *[0,0.9]* m/sn arasında değişecek şekilde x ve y konum değerleri elde edilir. Bunun için parakete denkleminde hız değeri, 0.1 m/sn adımlarla artırılır. Her hız adımı için açı değeri, 0.05 rad adımlarla $-\pi$' den $+\pi$'ye ulaşıncaya kadar değiştirilir ve her adımda x ve y konum değerleri hesaplanır. Bu iterasyon işlemi hız değeri 0.90 m/sn oluncaya kadar devam eder. Bu işlemler sonucu ortaya çıkan güzergah, Şekil 6.6' da verilmiştir. Mobil robot için çalışma alanı olarak gerçekte 40 m x 16 m' lik bir hareket alanı belirlenmiş ve bu alanın büyüklüğüne göre yapay güzergah ölçeklendirilmiştir.

Şekil 6.6'da iç içe dokuz elips bulunmaktadır ve bu sayı tespit edilen hız adım değişim aralığından kaynaklanmaktadır. Önceden de belirtildiği gibi dairesel yörüngeler üzerinde iç dairesel yörüngeden dıştaki dairesel yörüngeye doğru mobil robotun hızı artmaktadır. Şekil 6.6' da belirlenen açı/hız değişimine göre uygun bir model güzergah gerçekleştirilmiştir.

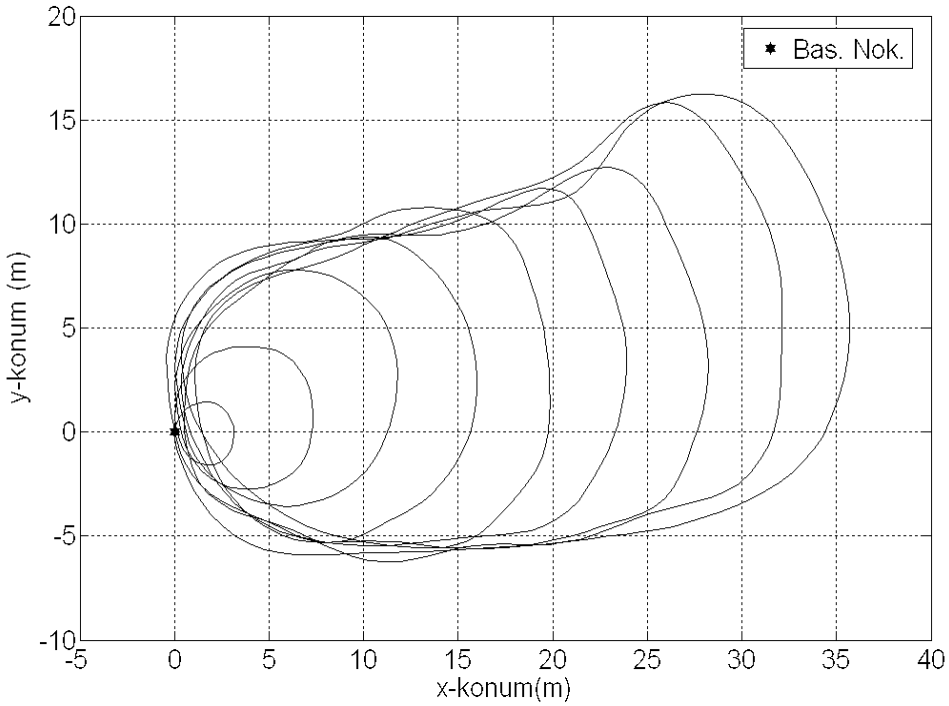


Şekil 6.6 Parakete algoritması ile oluşturulan eğitim güzergahı

Şekil 6.6' daki şablon referans alınarak çalışma sahasındaki zemine iç içe dokuz adet elips yörünge çizilmiştir ve mobil robotun çizilen hatları takip etmesi sağlanmıştır. Fakat ideal olarak hesaplanan güzergah gerçekte biraz farklı oluşmuştur. Bunun sebebi, mobil robotun manevra kabiliyetinden, mekaniksel hatalarından ve çalışma alanının istenilen büyüklükte olmamasından kaynaklanmaktadır. Çalışma alanında GPS ile kaydedilen G_E eğitim güzergahı, coğrafik ve UTM koordinat sisteminde sırasıyla Şekil 6.7' de görülmektedir. Bu güzergah birkaç kez tekrar edilmiş ve elde edilen kayıtlara bakılarak en uygun olanı seçilmiştir. Şekil 6.7' deki yıldız işareti ile belirlenen nokta başlangıç noktasıdır ve hareket yönü olarak saat yönü seçilmiştir. Mobil robotun GPS' ten elde edilen konum değerleri, UTM dönüşümüne tabi tutulduktan sonra, ilk hareket noktası referans alınarak mobil robotun konumu grafikler ile verilmiştir.



(a)



(b)

Şekil 6.7 Çalışma alanında GPS ile elde edilen (a) Coğrafik (b) UTM koordinat sistemindeki G_E eğitim güzergahı

G_E eğitim güzergahında enkoderler ile ölçülen minimum ve maksimum hızlar sırasıyla 0 m/sn ve 1 m/sn' dir. Bu hız aralığı, Çizelge 6.1' de görüldüğü gibi G_1 ve G_2 güzergahlarında elde edilen hız değişim aralığını kapsamaktadır. Açısal hız aralığı, -39.38 °/sn ve 30.54°/sn arasında değişmekte ve Çizelge 6.2' deki G_1 ve G_2 güzergahlarının açısal hız aralıklarını kapsamaktadır. Sonuç olarak belirlenen eğitim güzergahı ve ölçülen sensör aralık değerleri, YSA için eğitime uygundur ve güzergah oluşturulurken belirlenen kriterlere uymaktadır

6.4 YSA ile Elde Edilen Sonuçlar

6.4.1 YSA NARX modelinin oluşturulması

Mobil robot sensörlerinden elde edilen verilerin filtrelendikten ve zaman senkronizasyonu yapıldıktan sonra, YSA' nın eğitim süresini de kısaltacak şekilde düzenlenmesi gerekir. Bunun için GPS verileri önce, UTM koordinat sistemine dönüştürülmüş ve sonra bir önceki örneklenmiş GPS verisi ile farkı alınmıştır. Bu işlem x ve y yönündeki konum verileri için yapılmıştır. Yön verisi ise aynen korunarak uygulanmıştır. Yön verilerinde fark almanın, farkı alınmış verilere türevsel gürültü kattığı ve bu gürültülerin YSA' nın eğitimini olumsuz etkilediği görülmüştür. YSA' ya uygulanan giriş ve çıkış değerleri, sıfır ortalama ve birim varyans normalizasyonuna tabi tutulmuştur.

Mobil robotun YSA-NARX modelini oluşturmadan önce, modelin derecesi hakkında bir ön bilgiye sahip olmak için, sistemin giriş ve çıkışlarını teşkil eden veriler ve Lipschitz algoritması kullanılmıştır. NNSYSID araç kutusundaki Lipschitz algoritması yardımıyla bu işlem geçekleştirilmiştir. Algoritmanın işletilmesi sonucu elde edilen giriş ve çıkış derecesine göre kriter indeksi grafiği Şekil 6.8' de verilmiştir. Bu grafikte eğimin kırılmasının yatay eksende 6. dereceden olduğu görülmektedir. Ancak yapılan denemeler sonucundan 8. dereceden bir NARX modelinin daha iyi sonuç verdiği tespit edilmiştir.

Şekil 6.9' da, tasarlanan YSA-NARX modeli görülmektedir. Ancak bu modelin özel bir hali olan NFIR yapısı kullanılmıştır. Bu amaçla NARX modeli için belirlenen beş çeşit girişten sadece ikisi kullanılmıştır ve her birinin geçmiş 8 değeri

YSA'ya uygulanmıştır YSA' da sistemin çıkışlarını temsil eden girişler kullanılmamıştır. Böylece YSA NARX model yapısındaki dereceler;

Çıkış verileri için;

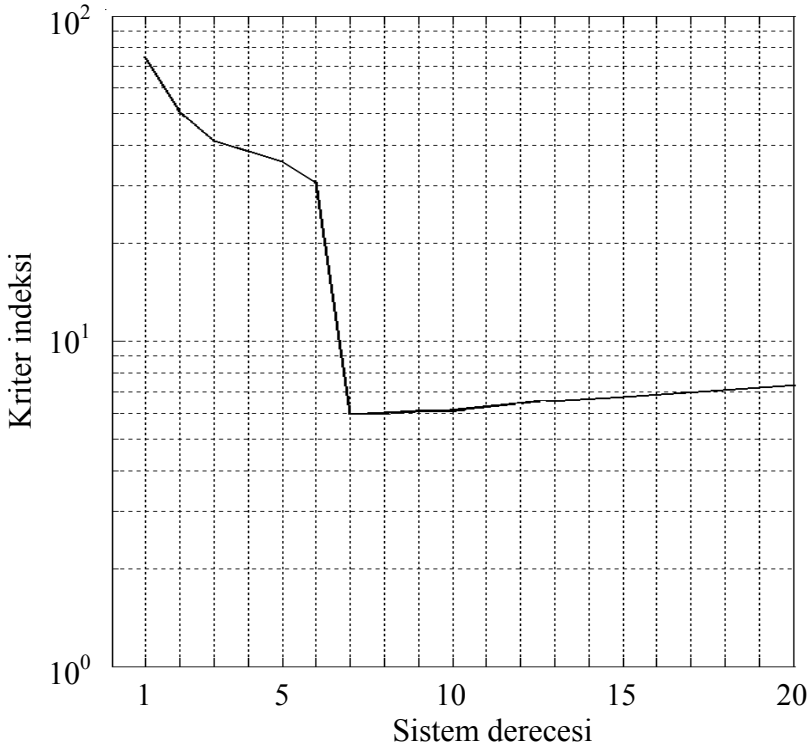
$n_x = n_y = n_\phi = 0$;

Giriş verileri için;

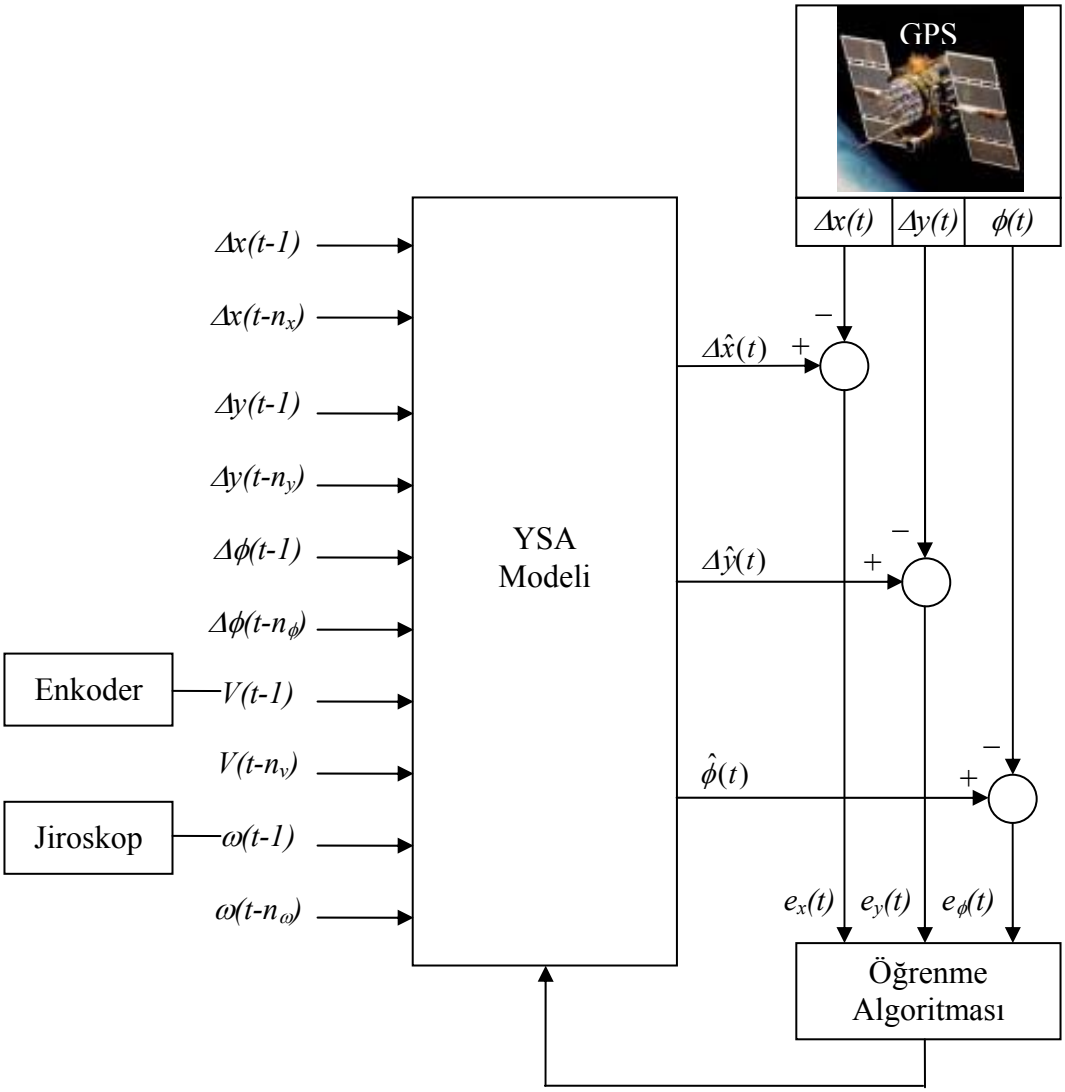
$n_\omega = n_V = 8$.

olur.

Şekli 6.9' da görülen YSA-NARX modeli üç katmandan oluşmaktadır. Bunlar; giriş ve çıkış katmanlarında doğrusal *lin()* ve gizli katmanda *tansig()* aktivasyon fonksiyonları kullanılmıştır. Giriş katmanı, 16 veri girişi ve 3 veri çıkışından oluşmaktadır. Yapılan YSA eğitim çalışmaları sonucunda gizli katman nöron sayısı 16 olduğunda çıkış hataları en düşük seviyede elde edilmiştir.



Şekil 6.8 Sistem derecesine göre Lipschitz algoritması ile elde edilen kriter indeksi grafiği



Şekil 6.9 YSA-NARX modeli

YSA çıkışlarından elde edilen kestirilmiş konum fark değerleri ve aşağıdaki ifadeler kullanılarak mobil robotun yerel koordinat sistemindeki konumu hesaplanır.

$$\begin{aligned}
&\hat{x}(1) = x_0\,; \quad \hat{y}(1) = y_0\,; \quad \hat{\phi}(1) = \phi_0 \\
&\hat{x}(t+1) = \hat{x}(t) + \Delta\hat{x}(t) \\
&\hat{y}(t+1) = \hat{y}(t) + \Delta\hat{y}(t) \\
&\hat{\phi}(t+1) = \hat{\phi}(t)
\end{aligned} \tag{6.1}$$

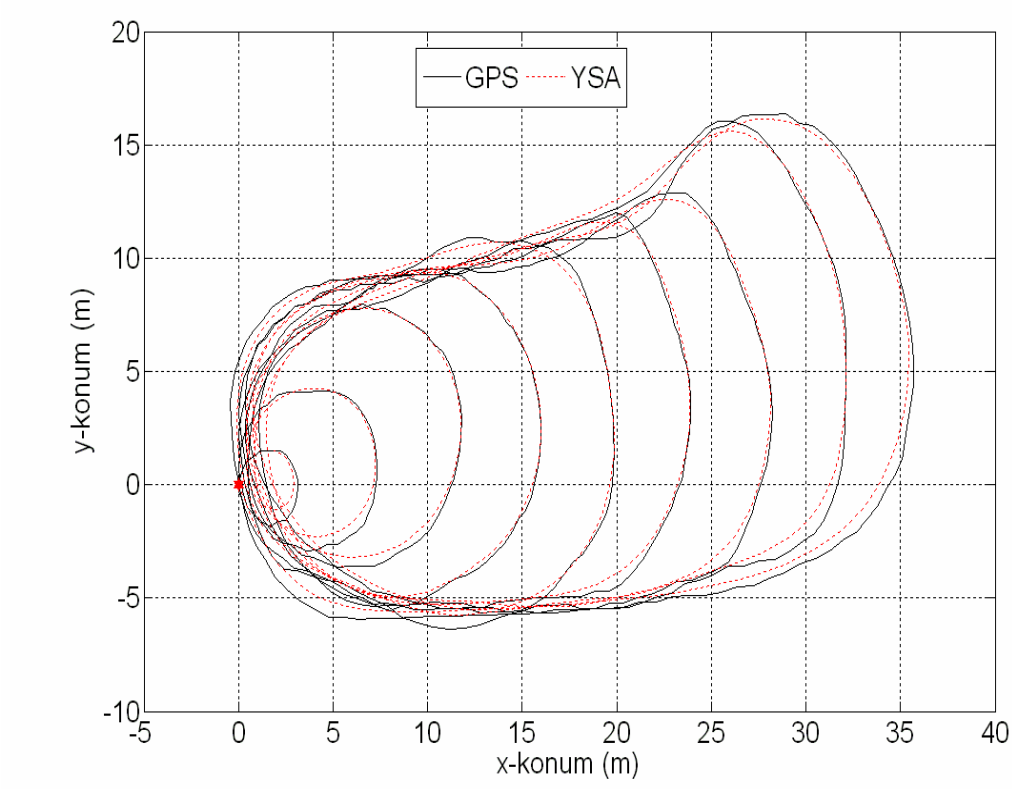
Burada (x_0, y_0, ϕ_0), GPS ile elde edilen başlangıç değerleri ve ($\hat{x}, \hat{y}, \hat{\phi}$), mobil robotun kestirilen navigasyon değerleridir. Çizelge 6.3' de görülen, farklı iterasyon

sayıları kullanılarak eğitim güzergahı YSA' ya öğretilmiştir. Her iterasyon sayısına karşılık mobil robotun RMS konum hatası hesaplanmıştır. Bu çizelgeden, RMS konum hatasının en düşük olduğu iterasyon sayısı olan 500 seçilmiştir. Daha ileri iterasyon sayılarında algoritma, arda arda birkaç kez YSA' daki toplam hatayı düşüremediğinden otomatik olarak eğitimi durdurmaktadır. Elde edilen bütün test güzergahları YSA' yı 500 iterasyonla eğittikten sonra konum kestirimi yapılmıştır.

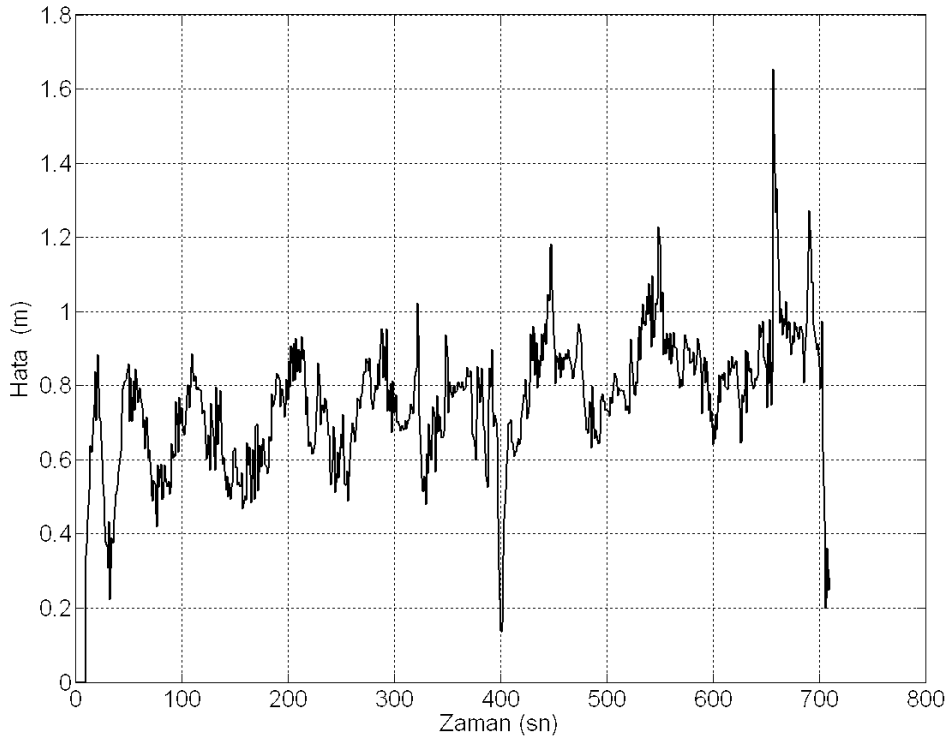
Çizelge 6.3 İterasyon sayısına karşılık RMS konum hatası

İterasyon sayısı	1	5	10	20	40	50	100	200	300	400	500	1000
RMS konum hatası (m)	7.11	2.107	0.912	0.866	0.885	0.88	0.862	0.856	0.863	0.862	0.86	0.86

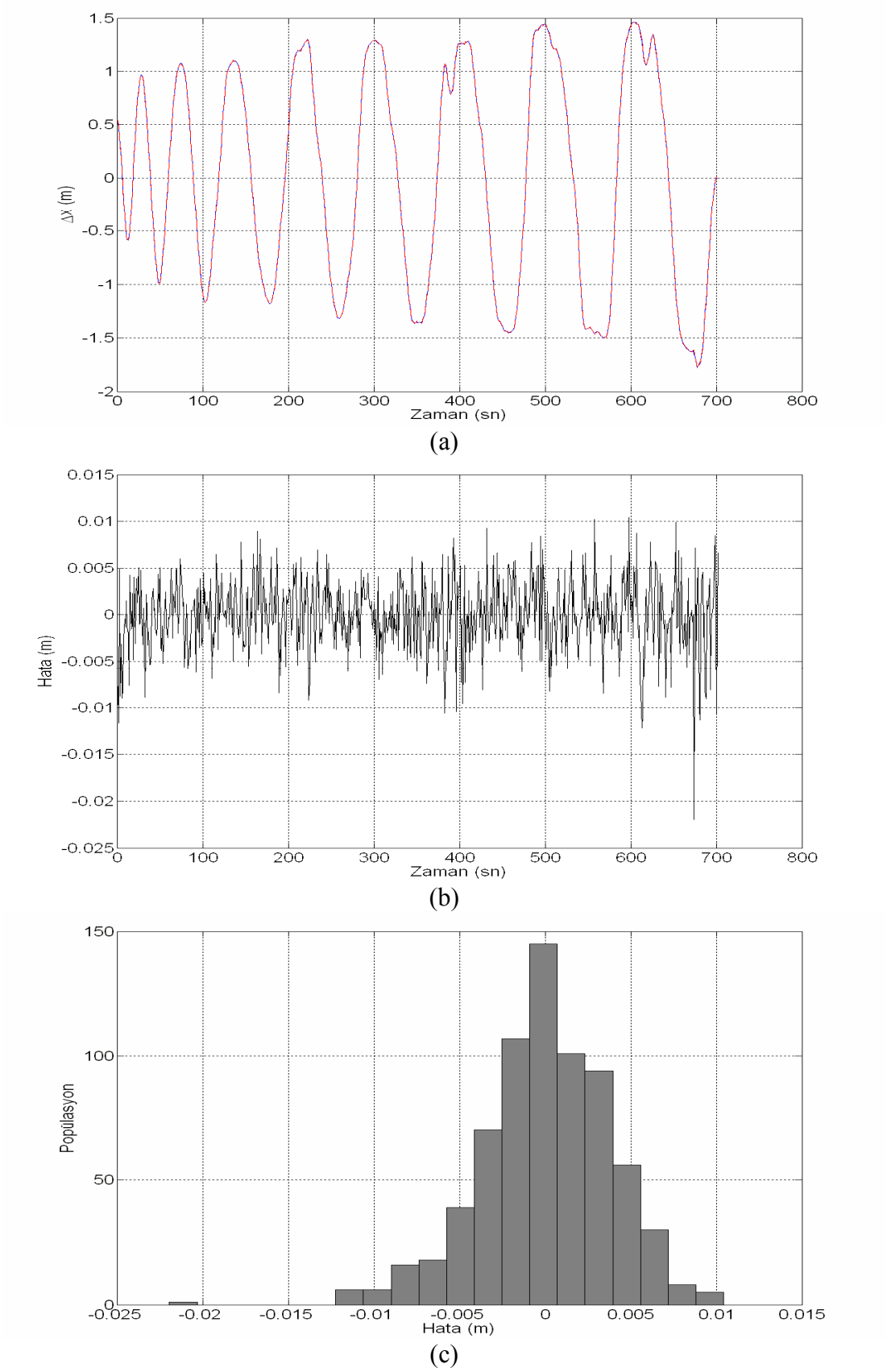
Mobil robotun Şekil 6.7' deki güzergahından elde edilen parakete sensör ve GPS verileri, YSA ile gerçekleştirilen Şekil 6.9' daki NARX modeline uygulanmıştır. YSA' nın çıkışlarından elde edilen değerler ve denklem (6.1) yardımıyla hesaplanan mobil robotun kestirilmiş konumu Şekil 6.10' da GPS ile elde edilen gerçek konumla beraber görülmektedir. Burada YSA ile elde edilen kırmızı renkte kesikli çizgiyle gösterilen mobil robot konumlarının GPS ile elde edilen güzergahı takip ettiği söylenebilir. Şekil 6.11' de verilmiş olan RMS konum hatası değişimi zaman ilerledikçe artmaktadır ve 1 m' ye ulaşmaktadır. Bunun sebebi ise mobil robotun sadece parakete sensörleri ile yapılan konum kestiriminden ve zaman içinde hataların integrasyonundan kaynaklanmaktadır. YSA ile kestirilen konum farkları ve yön bilgisinin, Şekil 6.12.a, Şekil 6.13.a ve Şekil 6.14.a' da GPS konum farkları ve yön bilgisi ile örtüştüğü, Şekil 6.12.b, Şekil 6.13.b ve Şekil 6.14.b'de hataların Δx için ±0.005 m, Δy için ±0.02 m ve ϕ ±0.1° aralığında değiştiği görülmektedir. Şekil 6.12.c, Şekil 6.13.c ve Şekil 6.14.c' de YSA çıkışlarının kestirim hatalarının histogramlarının gaussian olarak dağıldığı görülmektedir. Hata dağılımlarının gaussian olması YSA tarafından yapılan kestirimlerin iyi olduğunu gösterir, ancak mobil robot sisteminin yeterince tanılandığı anlamına gelmez. Tanılanma performansını daha iyi görebilmek için korelasyon testlerinin yapılması gerekir.



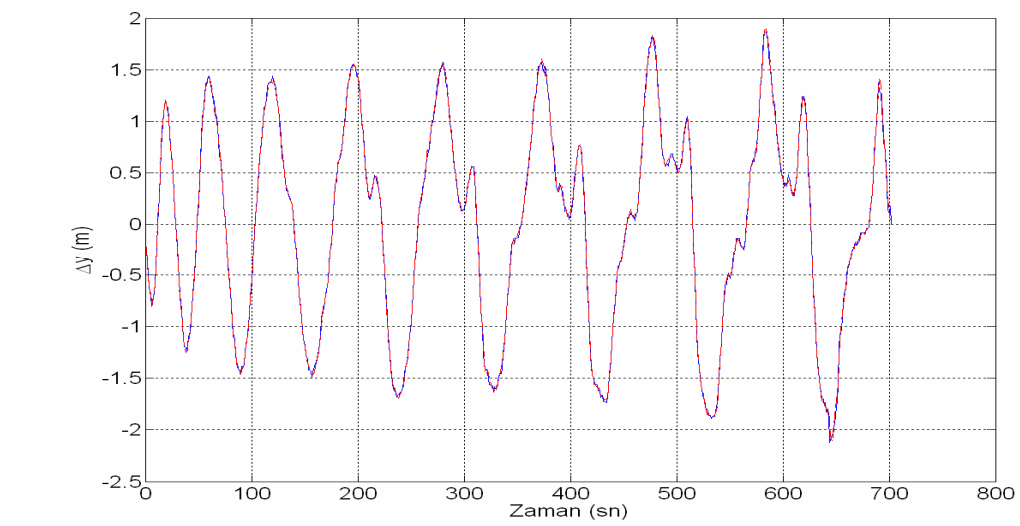
Şekil 6.10 GPS ve YSA ile elde edilen eğitim güzergahı



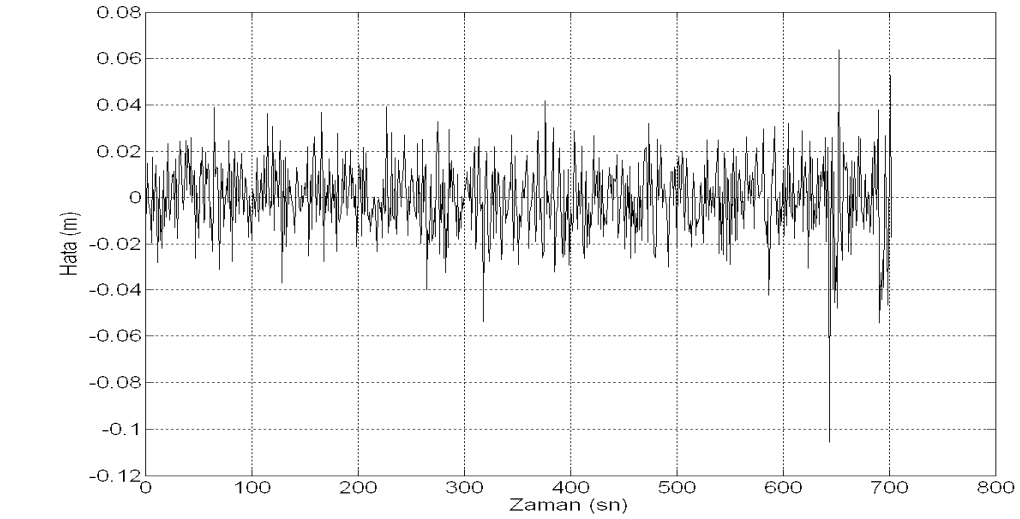
Şekil 6.11 YSA ile kestirilen RMS konum hatası



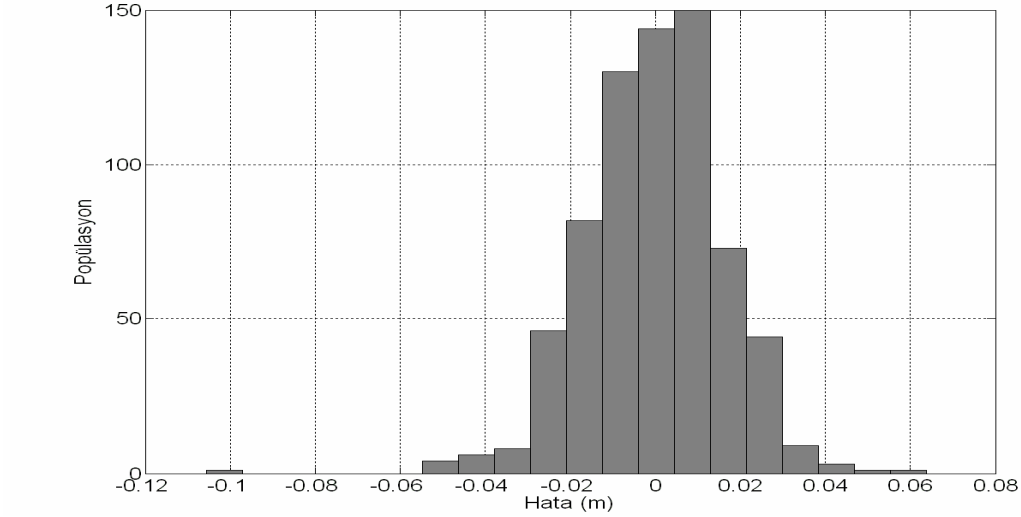
Şekil 6.12 YSA ile Δx 'in (a) kestirimi, (b) hatası ve (c) histogramı



(a)

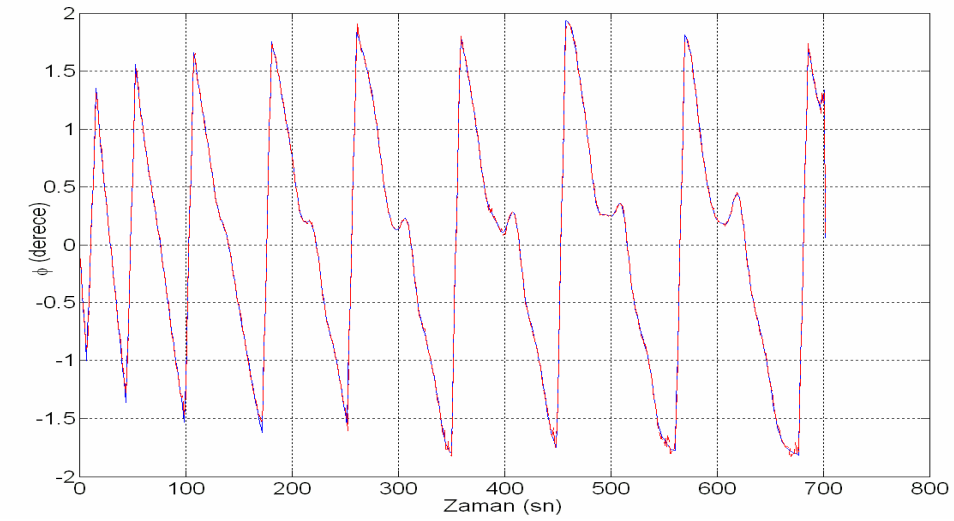


(b)

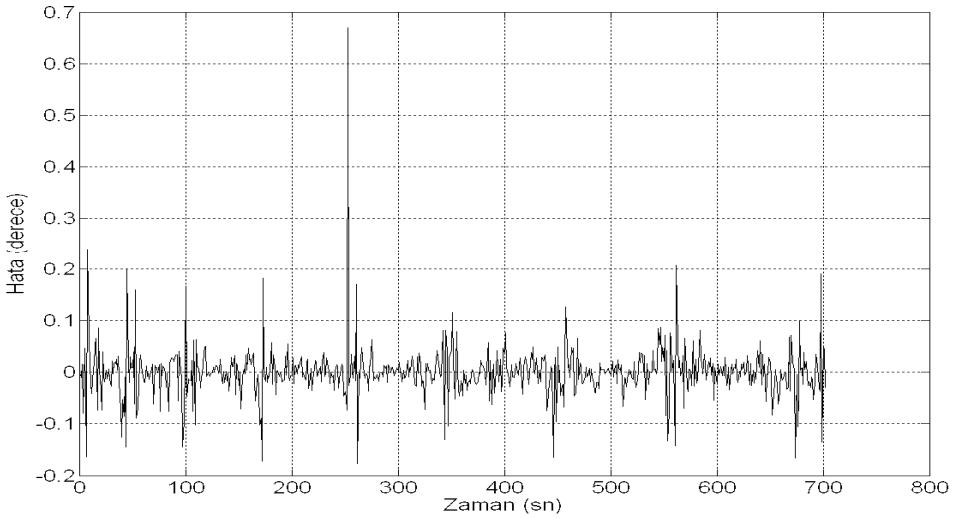


(c)

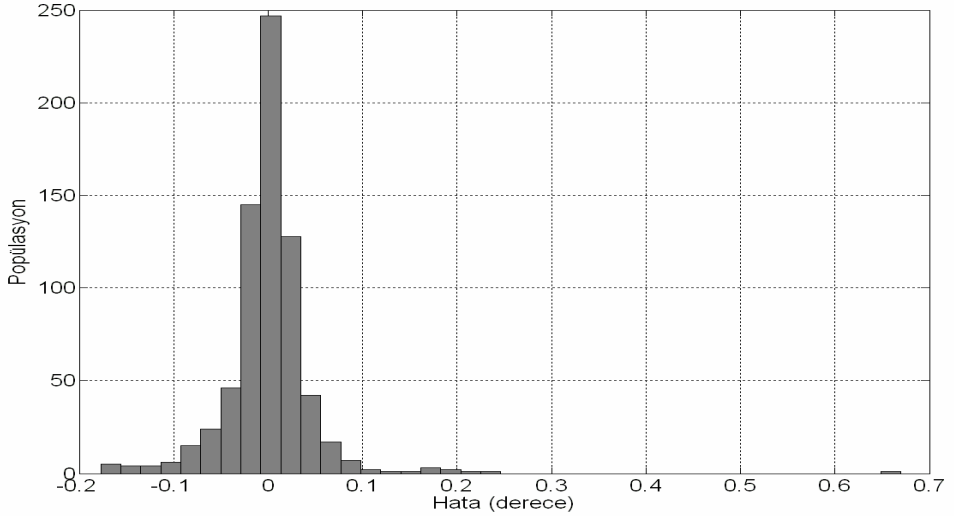
Şekil 6.13 YSA ile Δy 'nin (a) kestirimi,(b) hatası ve (c) histogramı



(a)



(b)



(c)

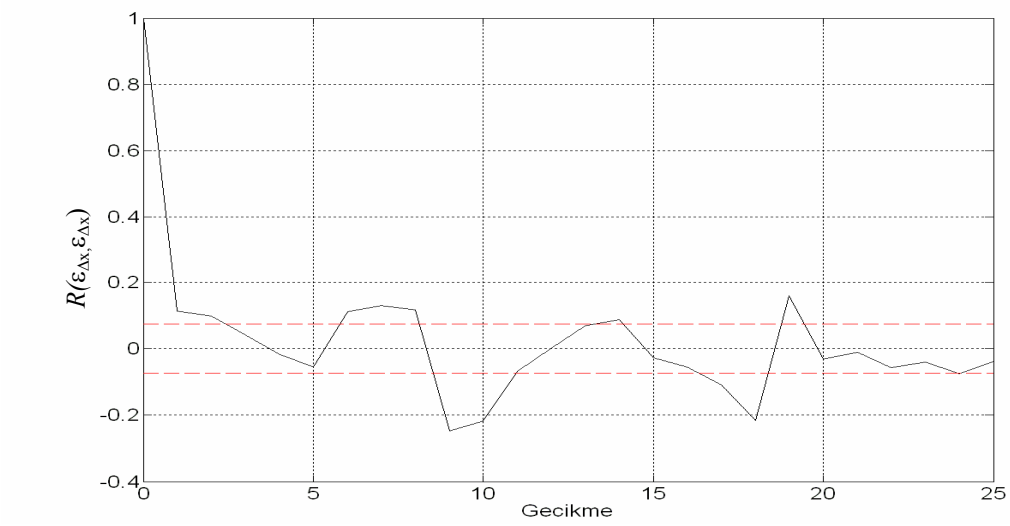
Şekil 6. 14 YSA ile ϕ'nin (a) kestirimi (b) hatası ve (c) histogramı

6.4.2 YSA-NARX modelinin tanılama performansı

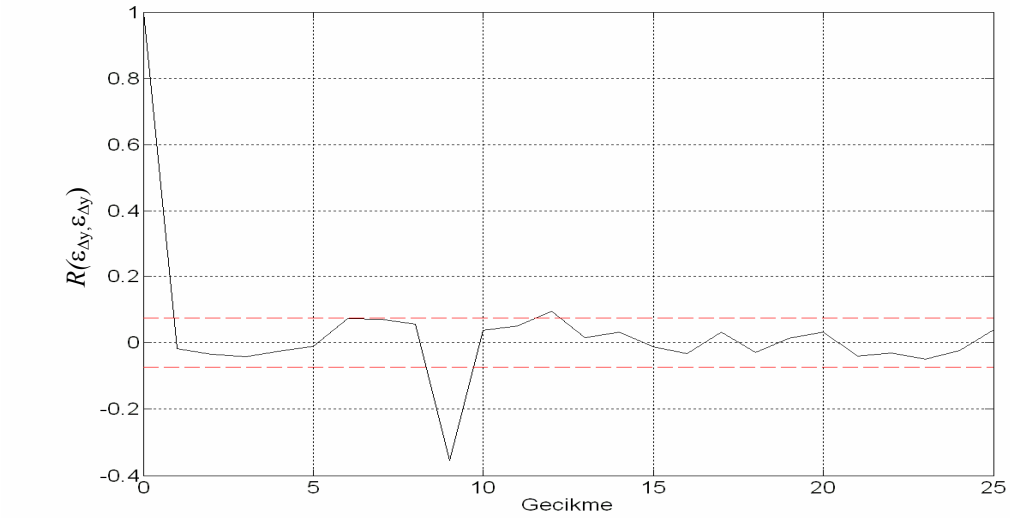
YSA-NARX modelinin sistemi yeterli düzeyde tanılayıp tanılamadığını anlamak için hataların otokorelasyonunu ve YSA' nın girişleri ile kestirim hatalarının kroskorelasyonlarını incelemek gerekir. Şekil 6.12.b, Şekil 6.13.b ve Şekil 6.14.b' deki kestirim hatalarının otokorelasyon sonuçları Şekil 6.15' de görülmektedir. Şekil 6.15.a' da Δx kestirim hatasının otokoreslasyonunun %95' lik güvenilirlik sınırlarını 5 ile 20 gecikme noktaları arasında aştığı gözlenmektedir. Δx kestirim hatası, bu güvenilirlik sınırlarında %100 rasgele bir dağılım göstermez. Bu durum YSA modelinin Δx çıkışını yeterli seviyede modelleyemediği anlamına gelir ve mobil robotun konumunun belirlenmesinde x yönünde hatalar oluşturabilir. Şekil 6.15 (b) ve (c)' de Δy ve ϕ hatalarının otokorelasyonu 5 ile 10 gecikme aralığı haricinde %95' lik güvenilirlik sınırları arasındadır. Burada da Δy ve ϕ kestirim hata dağılımının rasgele olduğu söylenebilir. YSA' nın V girişi ile Δx, Δy ve ϕ kestirim hatalarının kroskorelasyon sonuçları Şekil 6.16' da verilmiştir. Bu grafiklerden görüldüğü gibi otokorelasyon sonuçları %95 güvenilirlik sınırları içinde kalmaktadır ve V girişinin YSA kestirim hatalarında etkisinin bulunmadığını gösterir. Şekil 6.17' de YSA' nın ω girişinin Δx, Δy ve ϕ kestirim hatalarının kroskorelasyon sonuçları verilmiştir. Burada Şekil 6.17.a' da 15 ile 20 gecikme aralığında ve Şekil 6.17.b' de -25 ile -20 gecikme aralığında %95 güvenilirlik sınırını aştığı görülmektedir. ω girişinin Δx ve Δy kestirim hatalarında etkisi bulunmaktadır. ϕ kestirim hatasında ise ω girişinin etkisinin bulunmadığı Şekil 6.17.c' den anlaşılmaktadır. Çizelge 6.4' de, YSA ile kestirilen çıkışlardan oluşturulan eğitim güzergahı ile ilgili RMS ortalama konum hatası, ortalama x, y ve ϕ kestirim hatalarının 1 m'nin altında olduğu ve kestirilen x, y ve ϕ navigasyon değerlerinin GPS' den elde edilenler ile benzerliğinin VAF değerlerlerinden %99' un üzerinde olduğu görülmektedir.

Çizelge 6.4 YSA ile oluşturulan G_E eğitim güzergahı için istatistiksel sonuçlar

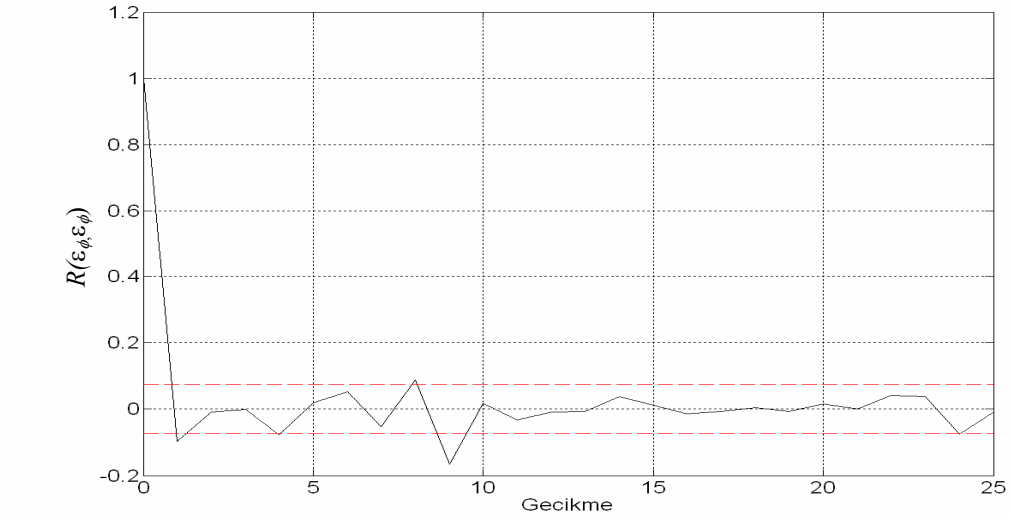
Ort. konum hatası	Ort. x konum hatası	Ort. y konum. hatası	Ort. ϕ yön hatası	RMS konum hatası	VAF(x)	VAF(y)	VAF(ϕ)
0.74 m	-0.05 m	-0.07 m	0.26 °	0.86 m	%99.67	%99.33	%99.41



(a)

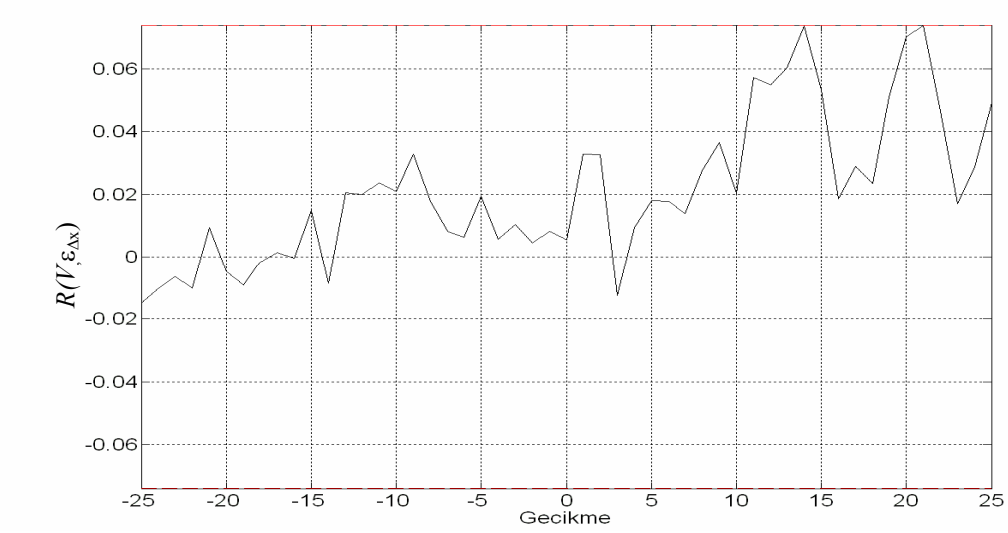


(b)

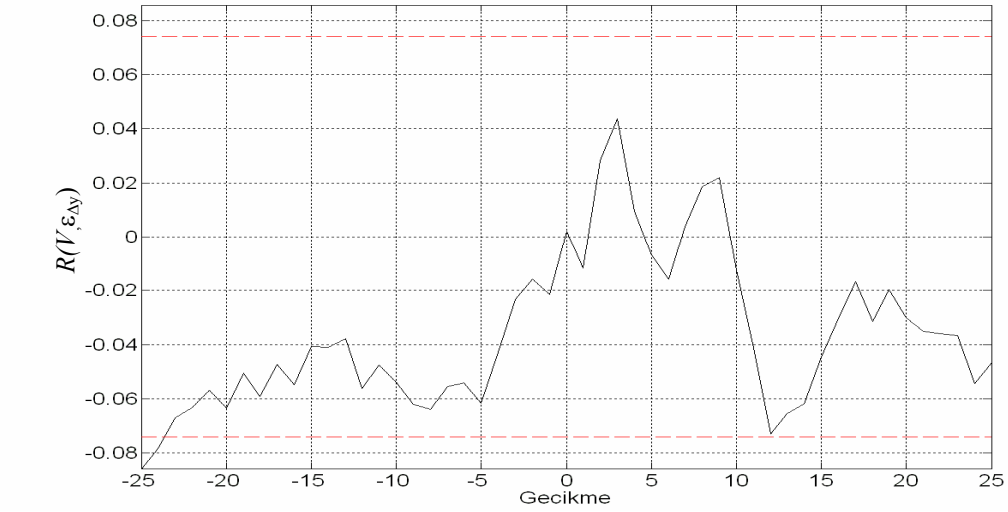


(c)

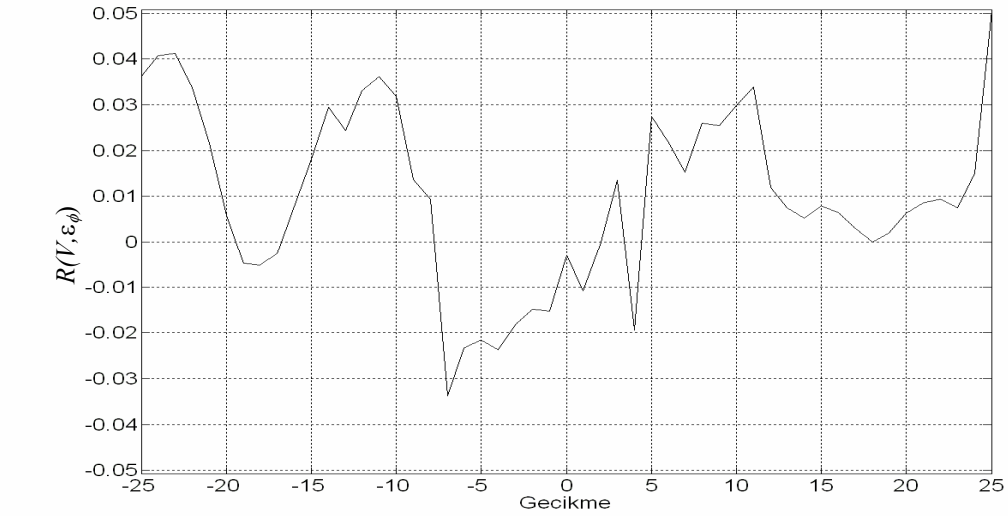
Şekil 6.15 (a) Δx , (b) Δy ve (c) ϕ kestirim hatalarının otokorelasyonu



(a)

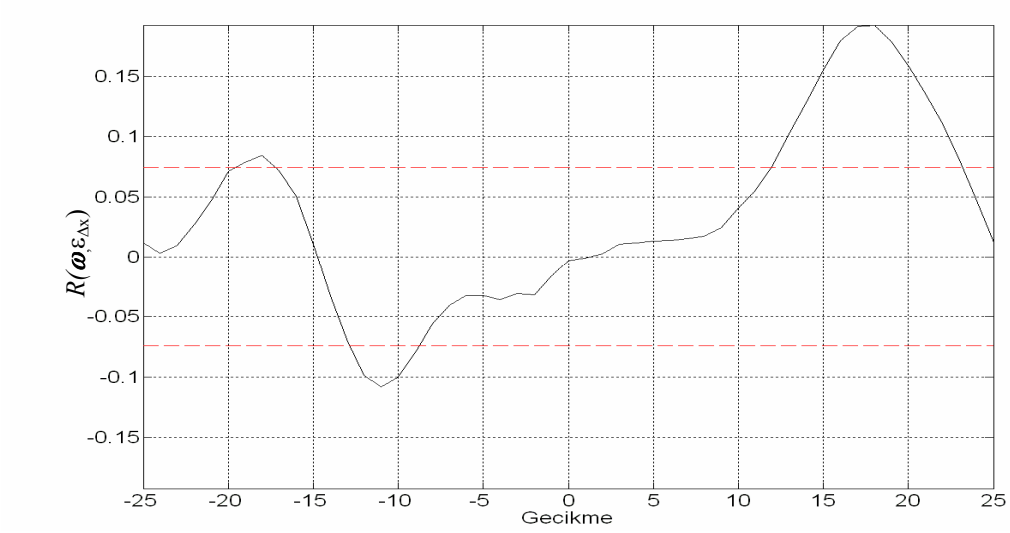


(b)

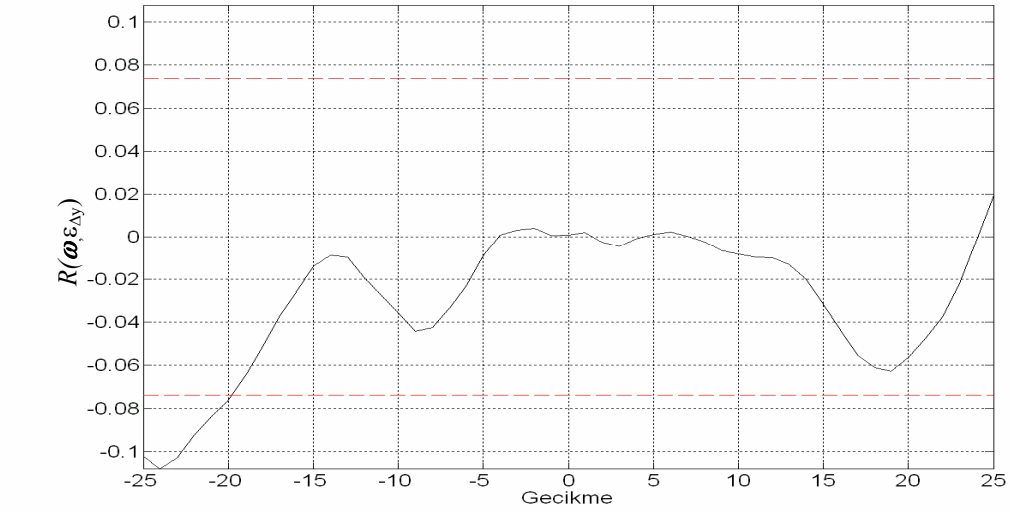


(c)

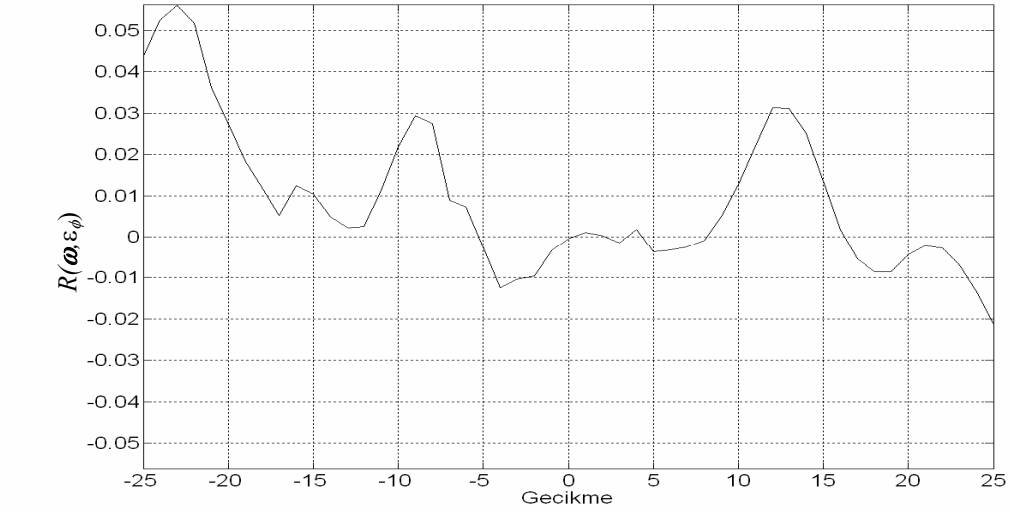
Şekil 6.16 YSA V girişi ile (a) Δx (b) Δy ve (c) ϕ hatalarının kroskorelasyonu



(a)



(b)



(c)

Şekil 6.17 YSA ω girişi ile (a) Δx (b) Δy ve (c) ϕ hatalarının kroskorelasyonu

6.5 YSA ile Test Güzergahlarının Simülasyonları

Elde edilen test güzergahlarının verileri YSA ile iki farklı şekilde değerlendirilmiştir. İlkinde, GPS verisi yardımı olmadan, tamamen eğitilen YSA' nın mevcut ağırlık katsayıları ile navigasyon yapılmış ve mobil robotun konumu kestirilmeye çalışılmıştır. Ortaya çıkan sonuç, GPS ile elde edilen güzergahla karşılaştırılmıştır. İkincisinde ise GPS verisinin kısmen yardımıyla navigasyon yapılmıştır. GPS verisinin rasgele zamanlarda mevcut olduğu durumlarda parakete sensörleri ile bu verilerin entegrasyonu gerçekleştirilmiştir.

Mobil robotun gerçek çalışma şartlarında navigasyonu durumunda oluşabilecek GPS sinyal kaybı, yazılımla incelenmiştir. Bunun için MATLAB programında rasgele zamanlarda GPS "var" ya da "yok" olarak işaretlenmiş ve pratikte olabilecek bir durum böylece simüle edilmiştir. Rasgele belirlenen zamanlar MATLAB programında bulunan TID (Time InDex) dizisine atanmıştır, TID dizisi aşağıdaki gibidir;

TID=(1,16,23,31,35,36,43,97,112,122,135,137,157,165,168,181,192,197,200,213,214,217,222,227,229,239,243,247,262,269,288,290,291,294,301,304,307,331,342,350, 358,366,372, 396,412,414,418,428,446) sn

TID dizisinde verilen zamanların dışında GPS verisi kesintiye uğramış kabul edilmiştir. Bütün güzergahlar ve kalman filtresi ile navigasyonda da dizideki değerler kullanılmıştır.

Bu dizideki maksimum zaman, en uzun güzergaha göre belirlenmiştir. Daha kısa seyir sürelerine sahip olan güzergahlar için o güzergahın maksimum seyir süresi dikkate alınarak dizide ilgili olan kısma kadar olan zamanlar kullanılmıştır. Yukarıda belirtilen YSA ile tasarlanan navigasyon sisteminin performansı, Kalman filtresi ile karşılaştırılmıştır. Grafiksel değişimlerin yanında hata değerleri de çizelgeler halinde verilmiştir. Grafiklerde düz çizgi GPS verilerini, kesikli çizgiler simülasyon sonuçlarını ve yıldız işareti hareketin başlangıç noktasını göstermektedir.

6.5.1 GPS desteği olmadan YSA ile G_1 test güzergahında konum kestirimi

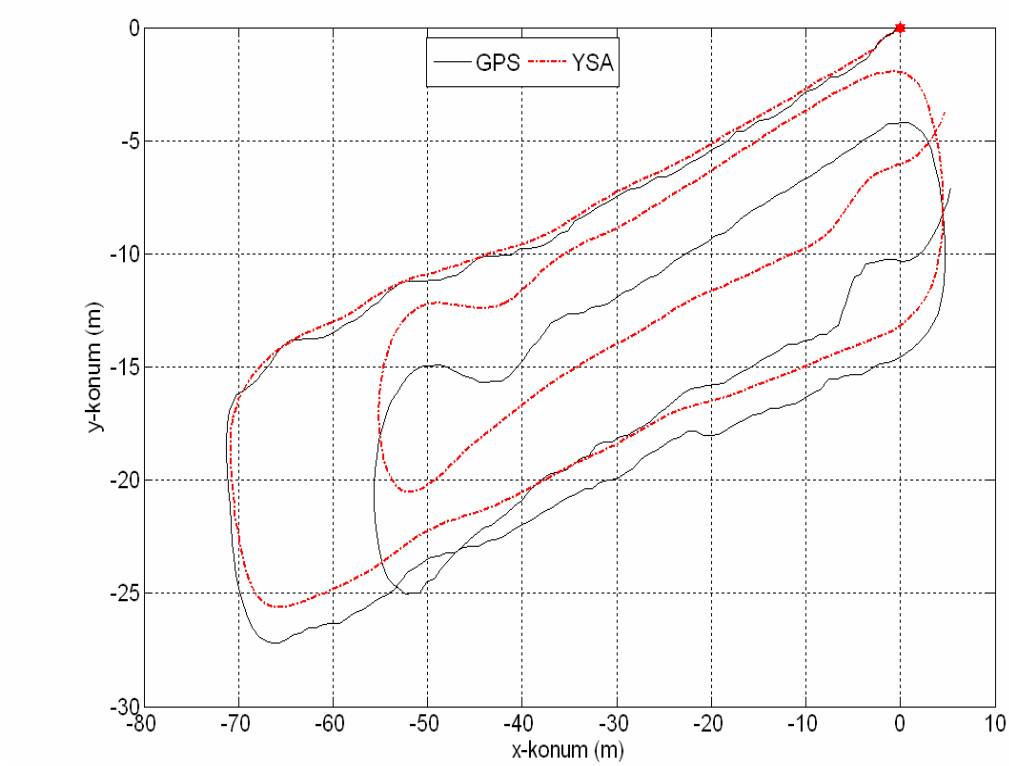
G_1 test güzergahından elde edilen enkoder ve jiroskop verileri, G_E güzergahı ile eğitilen YSA' ya uygulanmıştır. YSA' dan elde edilen kestirilmiş G_1 güzergahı, GPS ile elde edilen güzergahla beraber kırmızı renkte kesikli çizgiyle Şekil 6.18' de verilmiştir. Mobil robotun harekete başlama noktası, Şekil 6.18' de yıldız işareti ile gösterilmiştir ve hareketin yönü saat yönünün tersinedir. Mobil robot, 200 sn sonra yaptığı ikinci 90°' lik dönüşten sonra RMS konum hatası Şekil 6.19'dan görüldüğü gibi 2 m 'nin üzerine çıkmıştır ve geri kalan seyir süresinde RMS hata 3.5 m ile 5 m arasında kalmıştır. Bu hata mobil robotun yön değiştirmesinden kaynaklanmaktadır ve daha sonra hataların integrasyonu ile RMS konum hatası artmaktadır. Şekil 6.20.a' da *x* yönündeki hata ± 1 m arasında salınım yapmaktadır.

Mobil robotun dönüşlerde, jiroskop verilerinde meydana getirdiği ani değişimler, kestirilen Δx ve $\Delta y'$ lere hatalar katmaktadır. G_1 güzergahında, YSA'dan kaynaklanan kestirim hatalarının y yönündeki konum kestirimlerinde baskın olduğu Şekil 6.20.b'de görülmektedir. *y* yönündeki hata 200 sn' den sonra artmaktadır. Bu hata artışını, Şekil 6.20.c' de verilmiş olan yön hataları grafiğine bakarak görmek mümkündür. Yön hataları grafiğinde meydana gelen ani pik değişimleri Şekil 6.20 (a) ve (b)' de sırasıyla *x* ve *y* yönünde ani hata değişimlerinden oluşmaktadır. Bu güzergahta, yön verisinde meydana gelen değişimlerin, *y* yönündeki konum hatalarını daha fazla etkilediği görülmektedir.

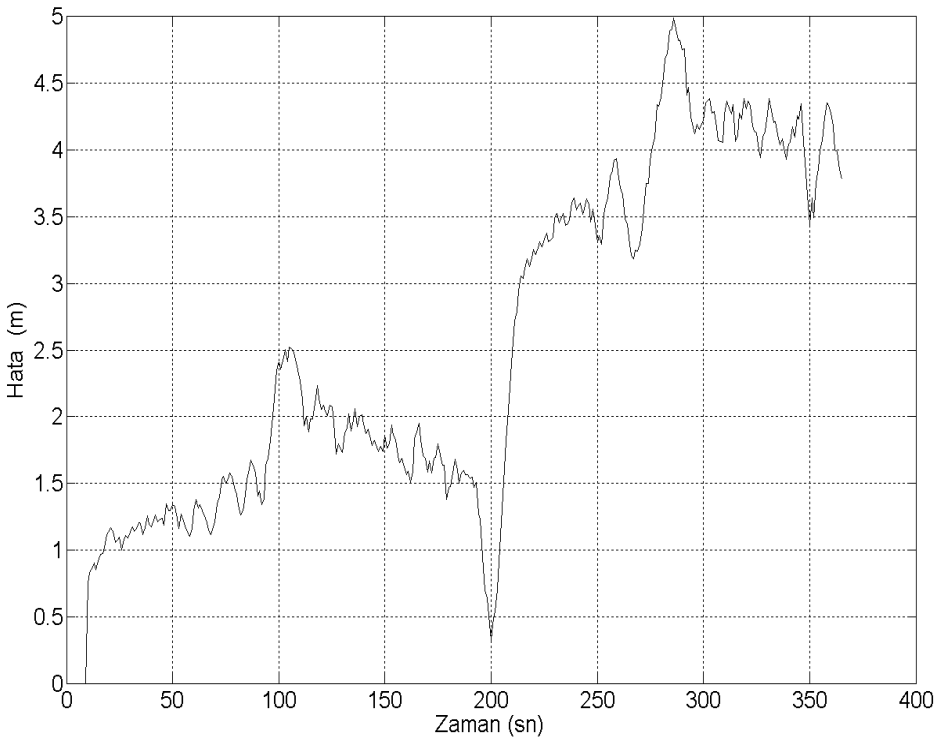
Çizelge 6.5' de RMS hatanın 1.58 m, yön hatasının 1.98° ve *y* yönündeki ortalama konum hatasının -2.30 m olduğu görülmektedir. *y* ve ϕ için GPS ile elde edilen değerler ile VAF benzerlik değerleri %99' un altındadır

Çizelge 6.5 GPS desteği olmadan G_1 güzergahı için istatistiksel sonuçlar.

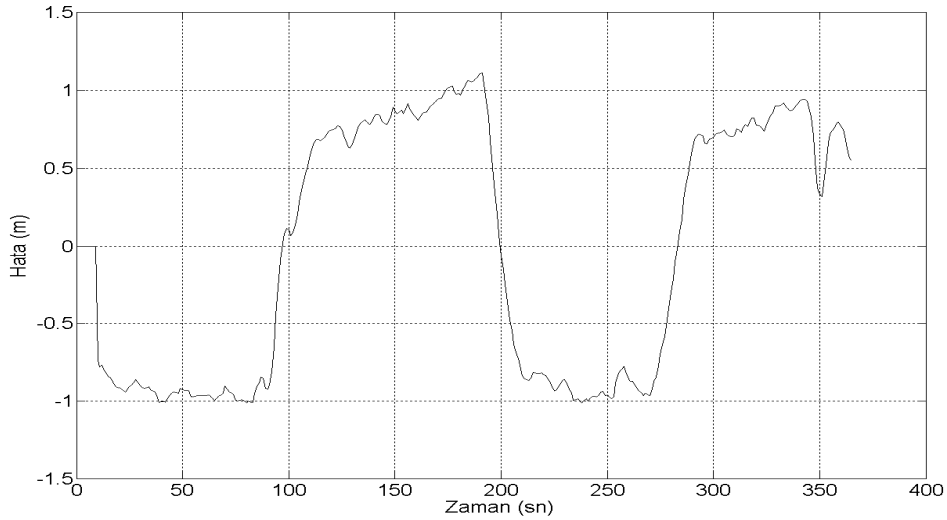
Ort. konum hatası	Ort. x konum hatası	Ort. y konum. hatası	Ort. ϕ yön hatası	RMS konum hatası	VAF(x)	VAF(y)	VAF(ϕ)
2.50 m	-0.03 m	-2.30 m	1.98 °	1.58 m	%99.86	%95.66	%89.71



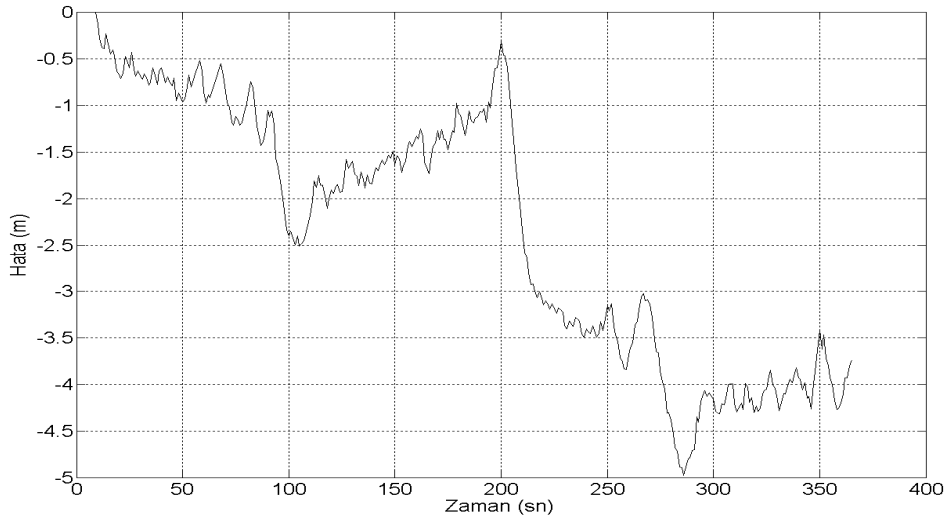
Şekil 6.18 GPS desteği olmadan YSA ile G_1 güzergahının kestirilmesi



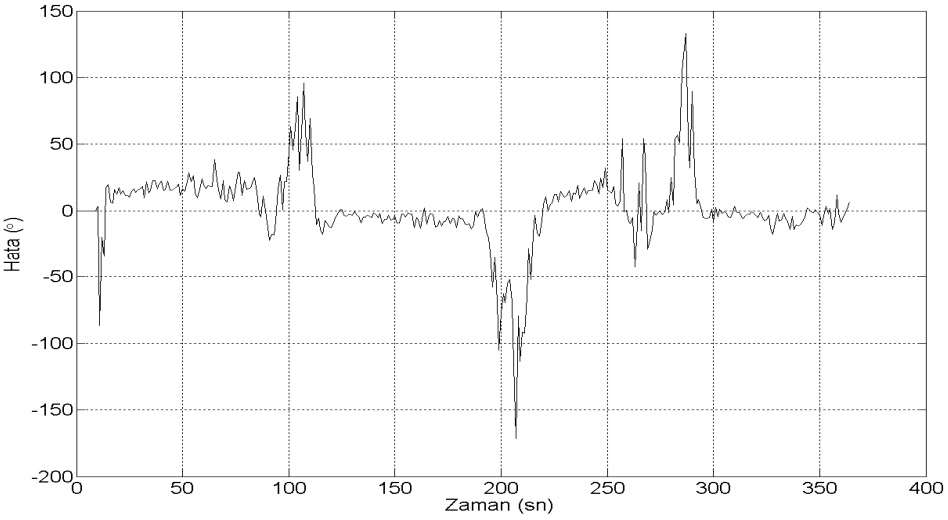
Şekil 6.19 G_1 güzergahı için RMS konum kestirim hatası



(a)



(b)



(c)

Şekil 6.20 (a) x yönündeki (b) y yönündeki ve (c) ϕ hataları

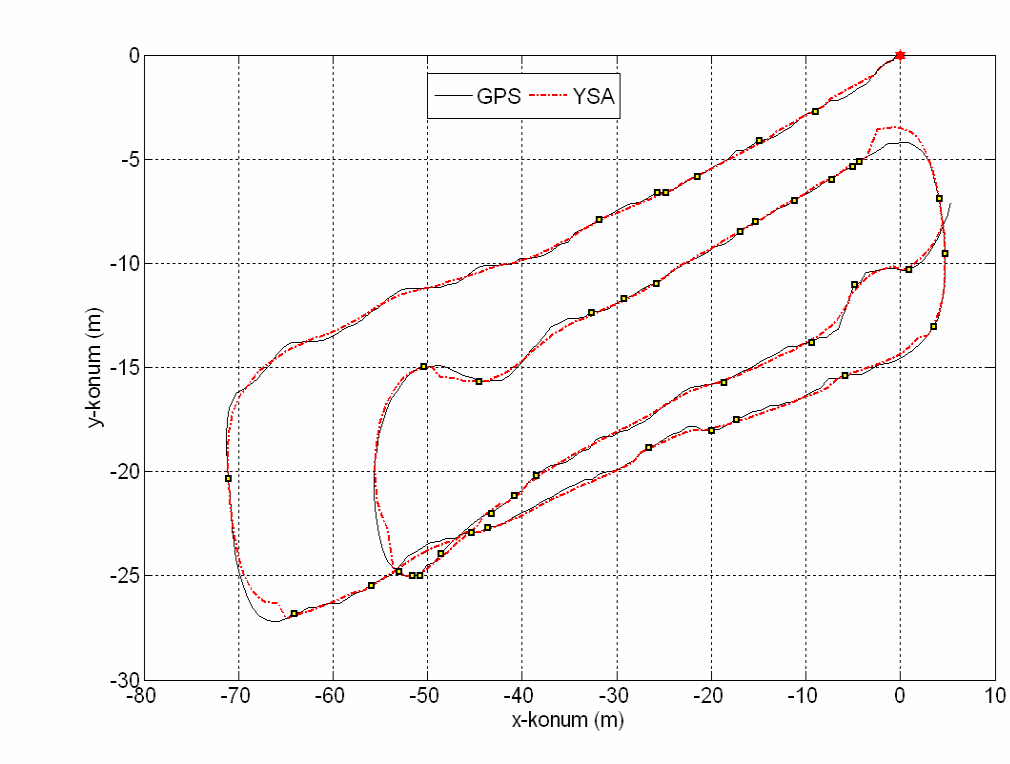
6.5.2 GPS destekli YSA ile G_1 test güzergahında konum kestirimi

YSA-NARX modelinde, parakete sensör verilerinden kestirilen çıktılar yardımı ile mobil robotun yerel koodinat sisteminde konumu hesaplanabilmektedir. GPS desteği olmadan G_1 güzergahının konum kestirimi ile ilgili sonuçlar önceki bölümde verilmiştir. Zamanla artan anlık RMS konum hatasını azaltmak için mutlak konum bilgisi üreten GPS verisine ihtiyaç vardır. Pratikte GPS verisi, kesintiye uğrayabilmektedir ve bu olay yazılan MATLAB programında simüle edilmiştir. G_1 güzergahında, TID dizisindeki rasgele zamanlarda GPS verisinin desteği olduğunda, GPS konumu, başlangıç noktası olarak alınır ve YSA modelinin çıkışları bu yeni başlangıç noktasına eklenerek mobil robotun yeni konumu hesaplanır. Bu işlem, yeni bir GPS verisi elde edilene kadar devam etmektedir. Bu işlemin sonucunda oluşan GPS destekli G_1 güzergahı kırmızı renkte kesikli çizgilerle Şekil 6.21' de verilmiştir. Şekil 6.21' de güzergah üzerinde bulunan kare işaretler, GPS verisinin mevcut olduğu noktalardır. Çizelge 6.6' dan ortalama konum ve toplam RMS konum hatalarının 1 m' nin altında düştüğü görülmektedir. Şekil 6.22' de görülen 1.2 m'nin üzerindeki ani pik hata değişimleri mobil robotun 90°'lik yön değişimlerinde meydana gelmektedir.

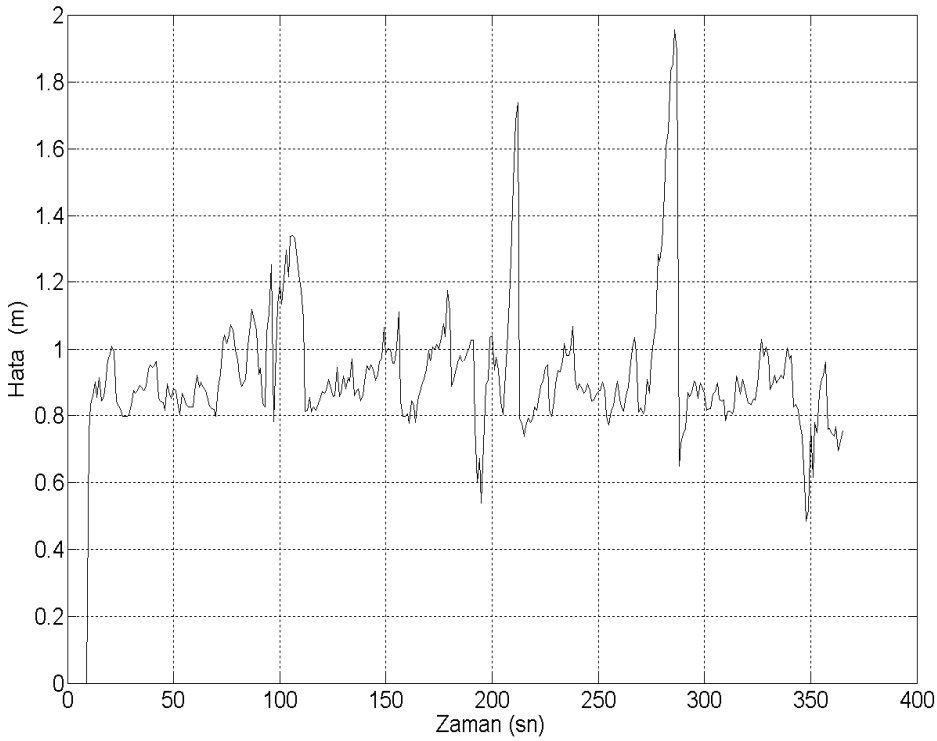
Çizelge 6.6 GPS destekli G_1 güzergahı için istatistiksel sonuçlar

Ort. konum hatası	Ort. x konum hatası	Ort. y konum. hatası	Ort. ϕ yön hatası	RMS konum hatası	VAF(x)	VAF(y)	VAF(ϕ)
0.90 m	0.01 m	-0.08 m	1.58 °	0.94 m	%99.87	%99.44	%91.33

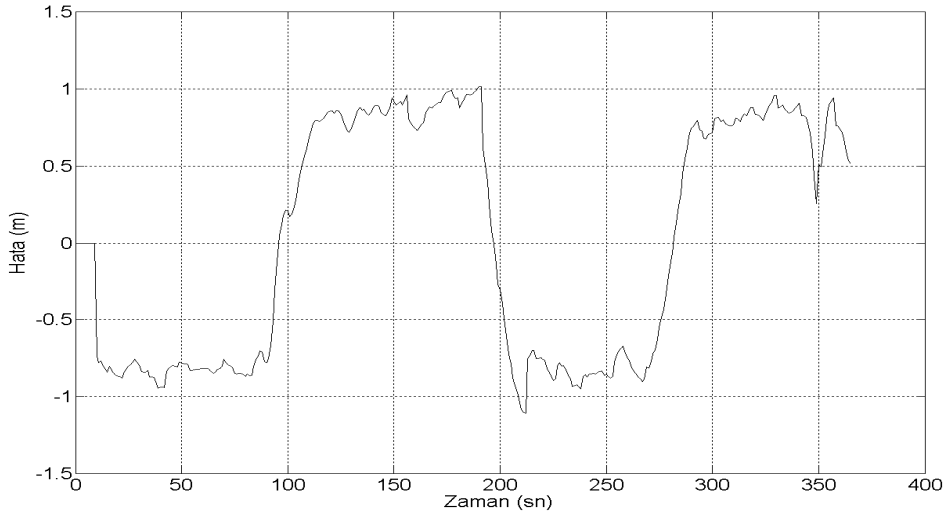
Şekil 6.23.a' da *x* yönündeki konum hatası verilmiştir ve bu hata Şekil 6.20.a' daki gibi ±1 m aralığında salınmaktadır ve çok fazla bir değişme görülmemektedir. Şekil 6.23.b ve Şekil 6.20.b birbiri ile karşılaştırılırsa, RMS hatanın büyük oranda düştüğü aynı zamanda Çizelge 6.5' de verilen y yönündeki ortalama konum hatasının mutlak değer olarak 2.30' m den 0.08 m düzeylerine düştüğü de Çizelge 6.6' da görülmektedir. Şekil 6.23.c' deki ortalama yön hatasında Şekil 6.20.c' dekine göre sadece 0.4°' lik bir iyileşme olduğu görülmektedir. VAF benzerlik değerlerinde *x* konum değerleri için bir değişiklik olmazken *y* ve ϕ değerleri iyileşmiştir



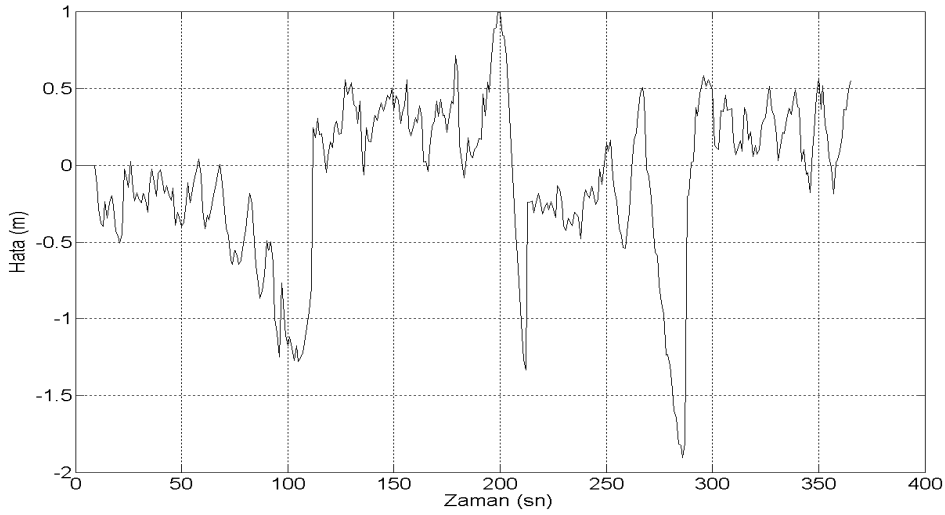
Şekil 6.21 GPS destekli YSA ile G_1 güzergahının kestirilmesi



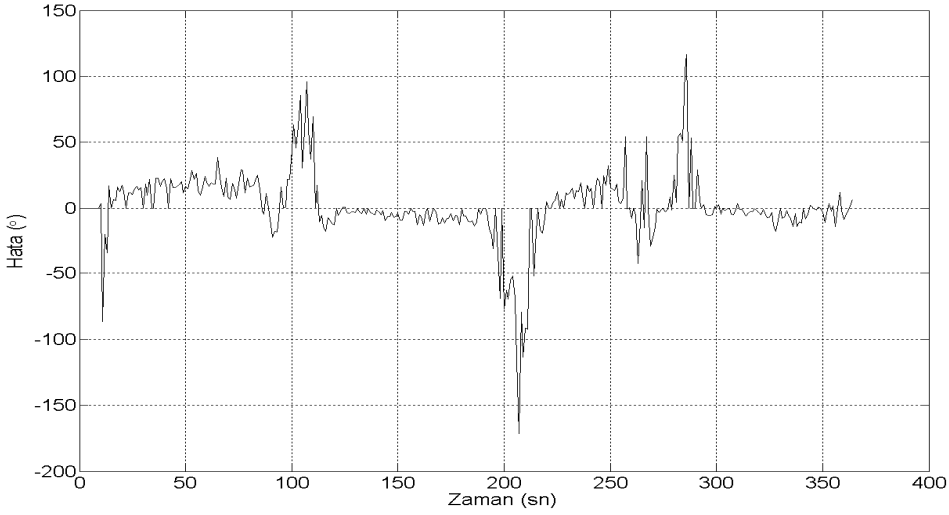
Şekil 6.22 G_1 güzergahı için RMS konum kestirim hatası



(a)



(b)



(c)

Şekil 6.23 (a) x yönündeki (b) y yönündeki ve (c) ϕ hataları

6.5.3 GPS destekli ve GPS desteği olmadan YSA ile G_2 test güzergahında konum kestirimi

Eğitilen YSA-NARX modelinin genelleştirme özelliğini test etmek için G_1 güzergahından biraz daha karmaşık olan G_2 güzergahı oluşturulmuştur. Bu güzergahda, iki tam ve üç yarım dönüş bulunmaktadır. Mobil robotun harekete başlama noktası, yıldız işareti ile gösterilmiştir ve hareket yönü saat yönünün tersidir. Şekil 6.24' de GPS desteği olmadan YSA modeli yardımı ile oluşturulan güzergah ve GPS güzergahı görülmektedir. YSA modelinin yardımı ile oluşturulan G_2 güzergahı, GPS verileri ile elde edilen şablona benzemektedir. Şekil 6.25' de görüldüğü gibi RMS hata zamanla artmaktadır ve 6 m' nin üzerine kadar çıkmaktadır. Şekil 6.26'da kestirilen G_2 güzergahı için GPS desteği olmadan x, y ve ϕ kestirim hataları görülmektedir.

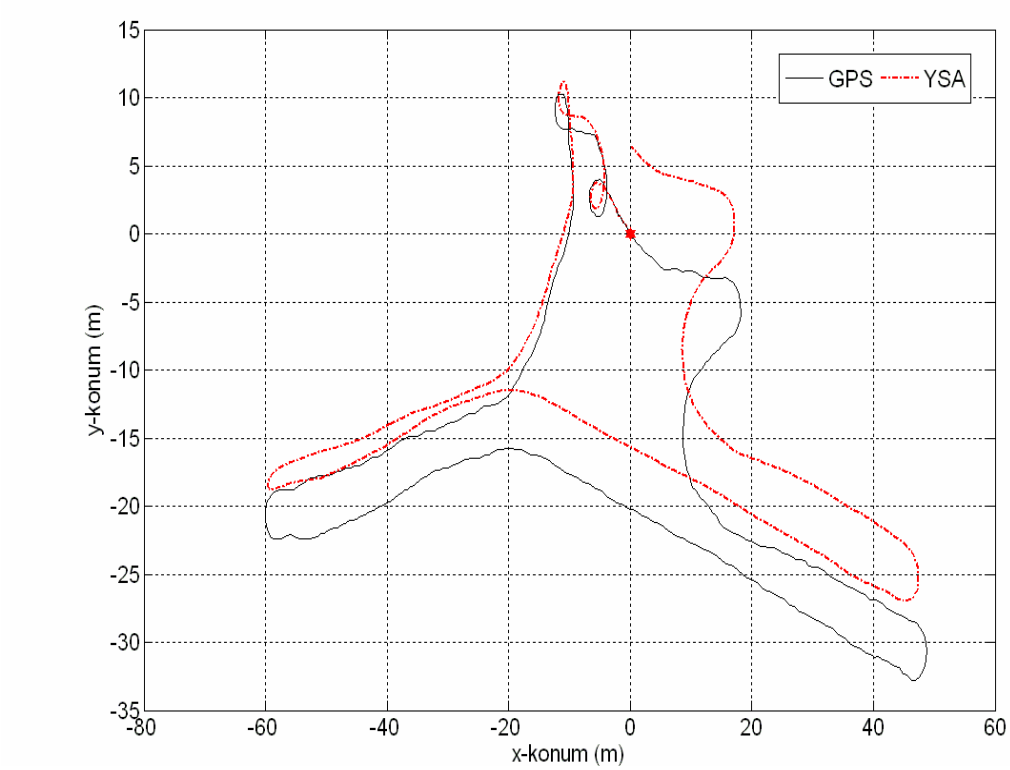
Şekil 6.27' de görüldüğü gibi GPS desteği, kare işaretleri bulunan noktalarda konumu güncellemektedir. Şekil 6.28' de GPS desteği ile RMS hatasının 0.2 m ile 1.8 m aralığına düştüğü görülmektedir. Şekil 6.26.a ve Şekil 6.29.a' da x yönündeki hata değişiminin ±1 m aralığında değiştiği görülmektedir. RMS konum hatasının artmasına doğrudan etki eden hata, Şekil 6.26.b' de görülen y yönündeki zamanla artan mutlak hatadır. GPS desteği ile bu hatanın ±1.5 m aralığına düştüğü Şekil 6.29.b'de görülmektedir. x yönündeki ani hata değişimlerinin kaynağı, Şekil 6.26.c ve Şekil 6.29.c' de görüldüğü gibi ϕ hatasında ani değişimlerin olduğu yerlerdedir. Çizelge 6.7' de GPS desteği olmadan ve Çizelge 6.8' de GPS destekli YSA yardımı ile G_2 güzergahında mobil robot konum kestirim hataları verilmiştir. Çizelge 6.8' deki VAF (Variance Accounted For) değerlerine bakılarak YSA-NARX modeli ile elde edilen G_2 güzergahı şablonunun GPS' den elde edilen ile aralarında benzerlik olduğu söylenebilir.

Çizelge 6.7 GPS desteği olmadan G_2 güzergahı için istatistiksel sonuçlar

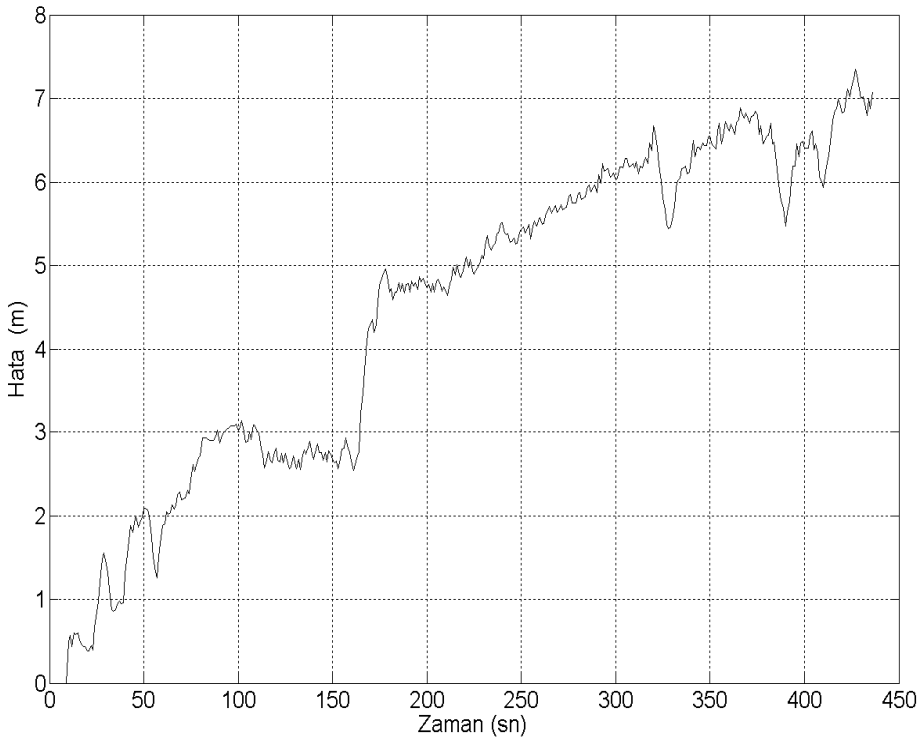
Ort. konum hatası	Ort. x konum hatası	Ort. y konum. hatası	Ort. ϕ yön hatası	RMS konum hatası	VAF(x)	VAF(y)	VAF(ϕ)
4.4 m	0.68 m	-4.3 m	1.64°	2.1 m	%99.94	%97.08	%96,42

Çizelge 6.8 GPS destekli G_2 güzergahı için istatistiksel sonuçlar

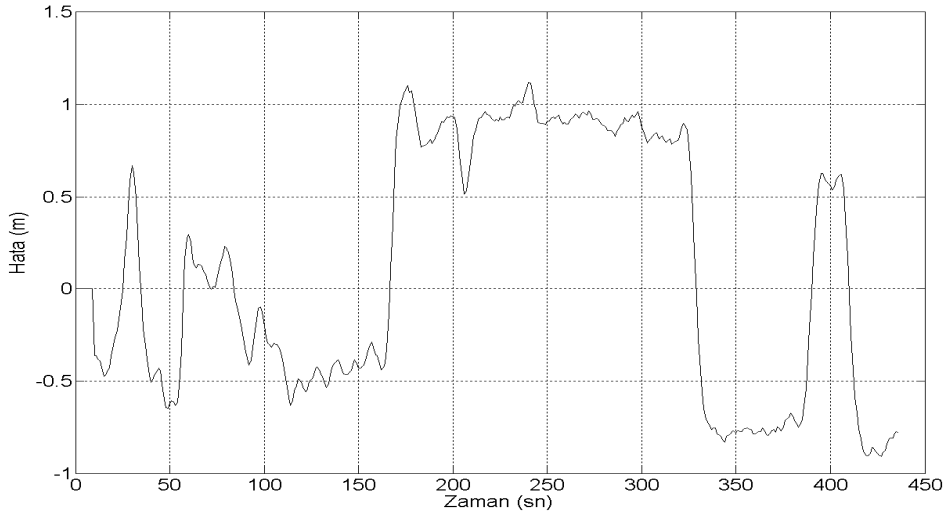
Ort. konum hatası	Ort. x konum hatası	Ort. y konum. hatası	Ort. ϕ yön hatası	RMS konum hatası	VAF(x)	VAF(y)	VAF(ϕ)
0.73 m	0.01 m	-0.1 m	1.3°	0.85 m	%99.94	%99.86	%96.91



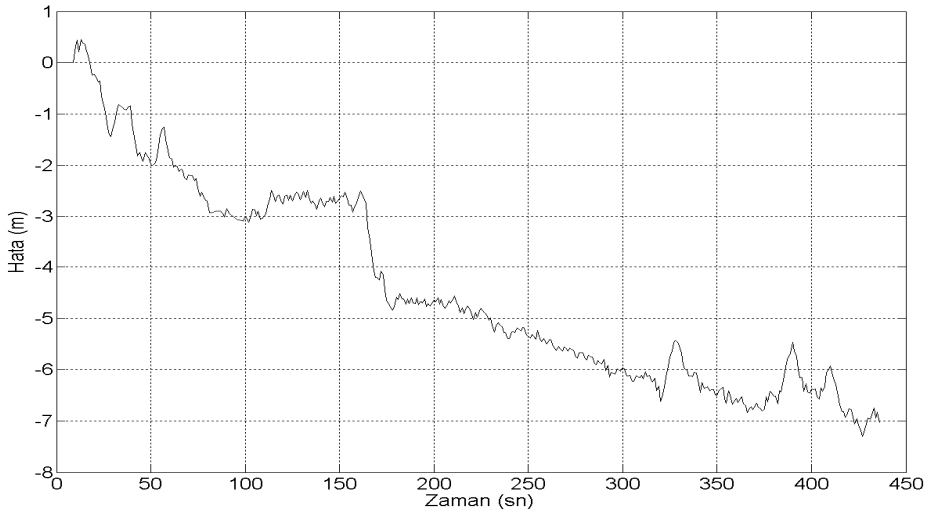
Şekil 6.24 GPS desteği olmadan YSA ile G_2 güzergahının kestirilmesi



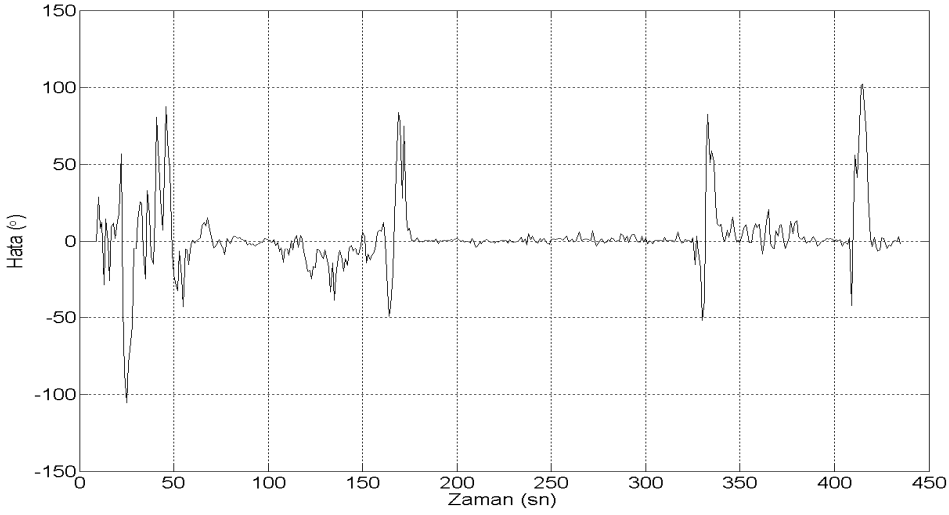
Şekil 6.25 G_2 güzergahı için RMS konum kestirim hatası



(a)

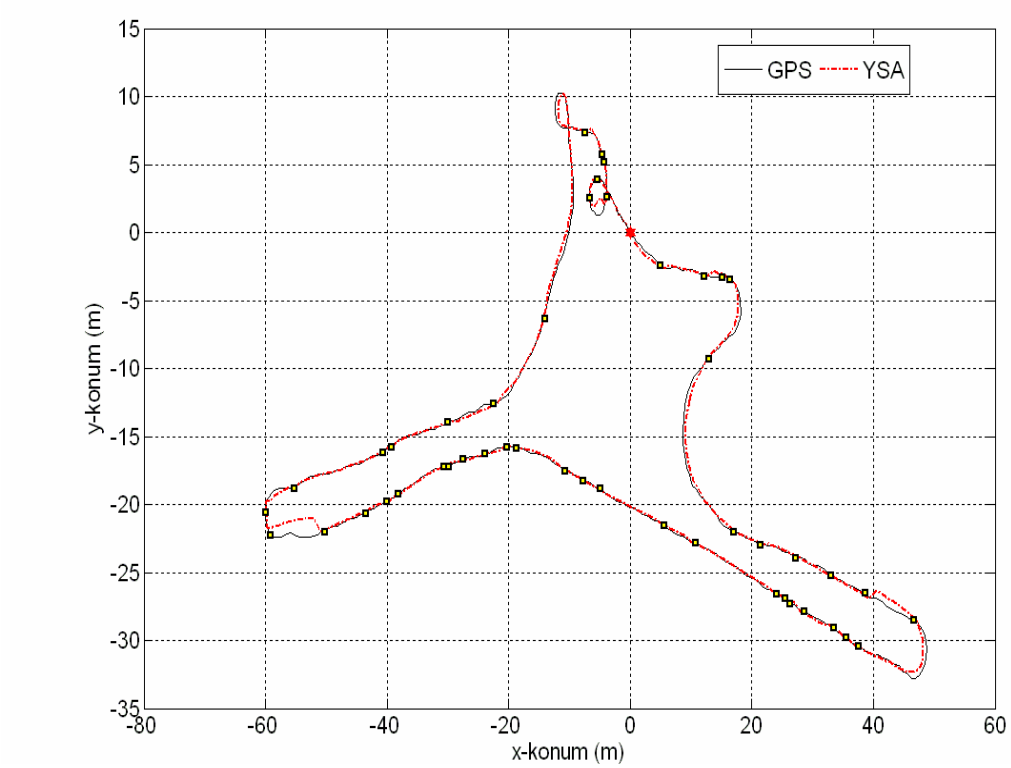


(b)

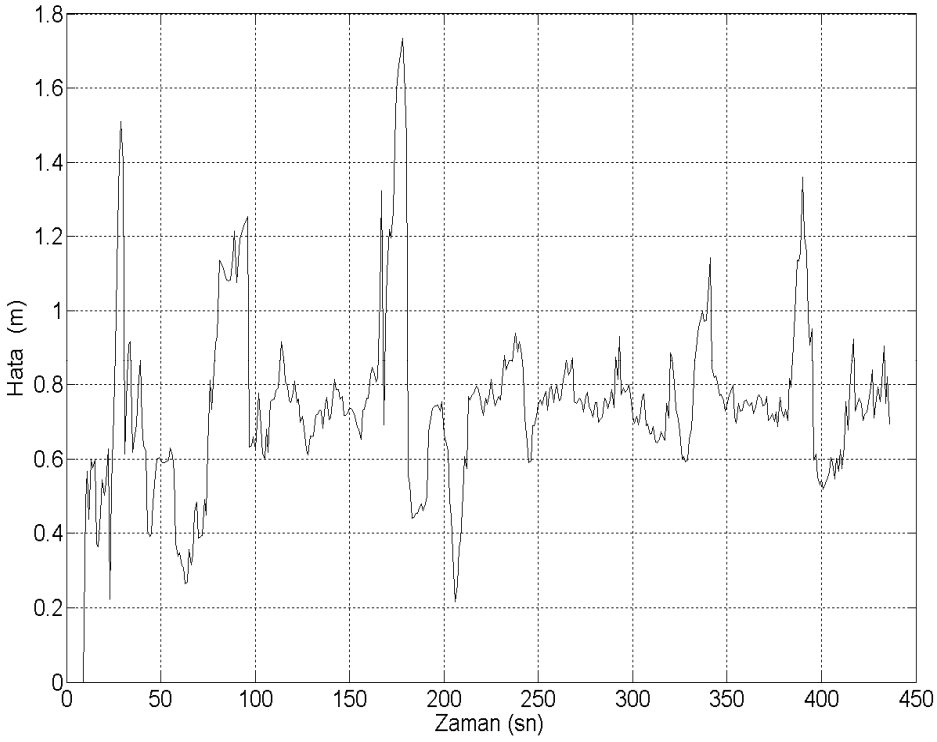


(c)

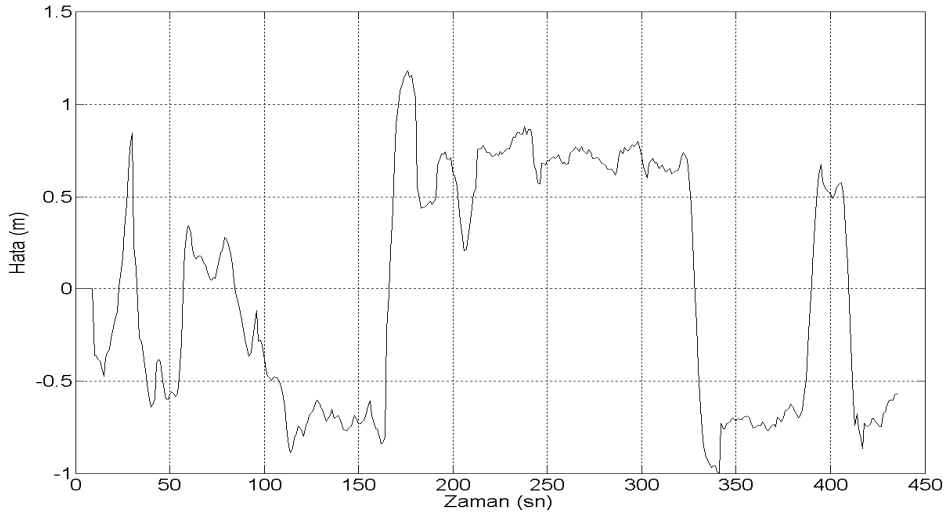
Şekil 6.26 (a) x yönündeki (b) y yönündeki ve (c) ϕ hataları



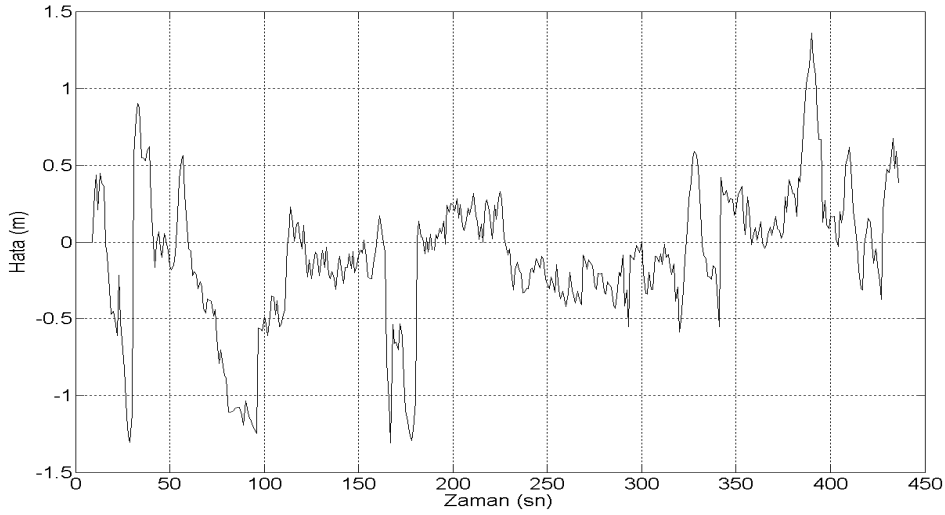
Şekil 6.27 GPS destekli YSA ile G_2 güzergahının kestirilmesi



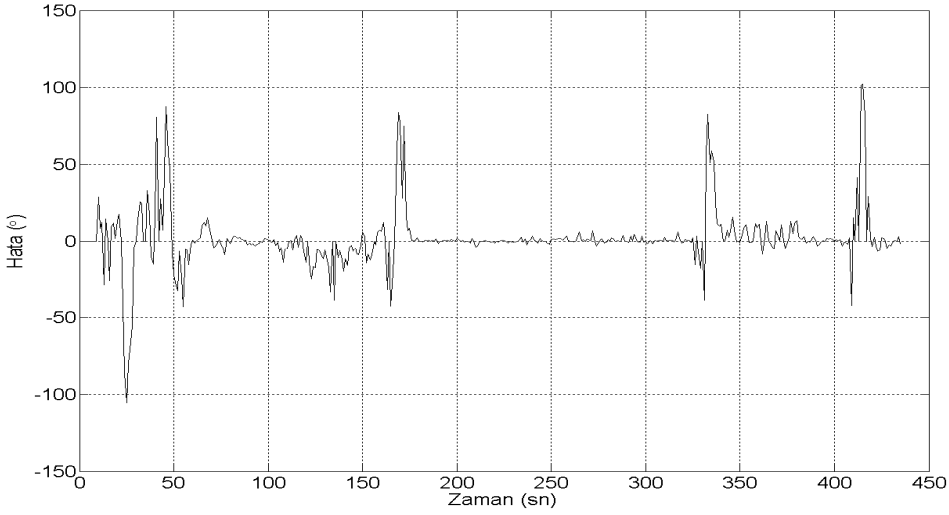
Şekil 6.28 G_2 güzergahı için RMS konum kestirim hatası



(a)



(b)



(c)

Şekil 6.29 (a) x yönündeki (b) y yönündeki ve (c) ϕ hataları

6.6 Kalman Filtresi ile Mobil Robot Navigasyonu

6.6.1 GPS desteği olmadan ve GPS destekli Kalman filtresi ile G_1 test güzergahında konum kestirimi

Mobil robotun GPS' den elde edilen başlangıç konumu belirlendikten sonra GPS desteği olmadan Kalman filtresi algoritması çalıştırılmıştır. Şekil 6.30' da, G_1 güzergahı için Kalman filtresi algoritmasından elde edilen konum kestirimi kırmızı renkte kesikli çizgiyle gösterilmiştir. G_1 güzergahı için anlık RMS hata Şekil 6.31' de görüldüğü gibi 25 m' ye kadar çıkmaktadır. Yön bilgisi hatası zaman içinde toplanarak G_1 güzergahını kaydırmaktadır. Bu kaymanın neden olduğu ortalama hata, Çizelge 6.9'da x ve y yönünde sırasıyla -6.85 m ve 11.1 m olarak görülmektedir ve toplam RMS konum hatası 3.66 m' dir. y ve ϕ için VAF değerlerinin düşük olduğu gözlenmiştir. Şekil 6.32'de kestirilen G_1 güzergahı için x, y ve ϕ kestirim hataları görülmektedir.

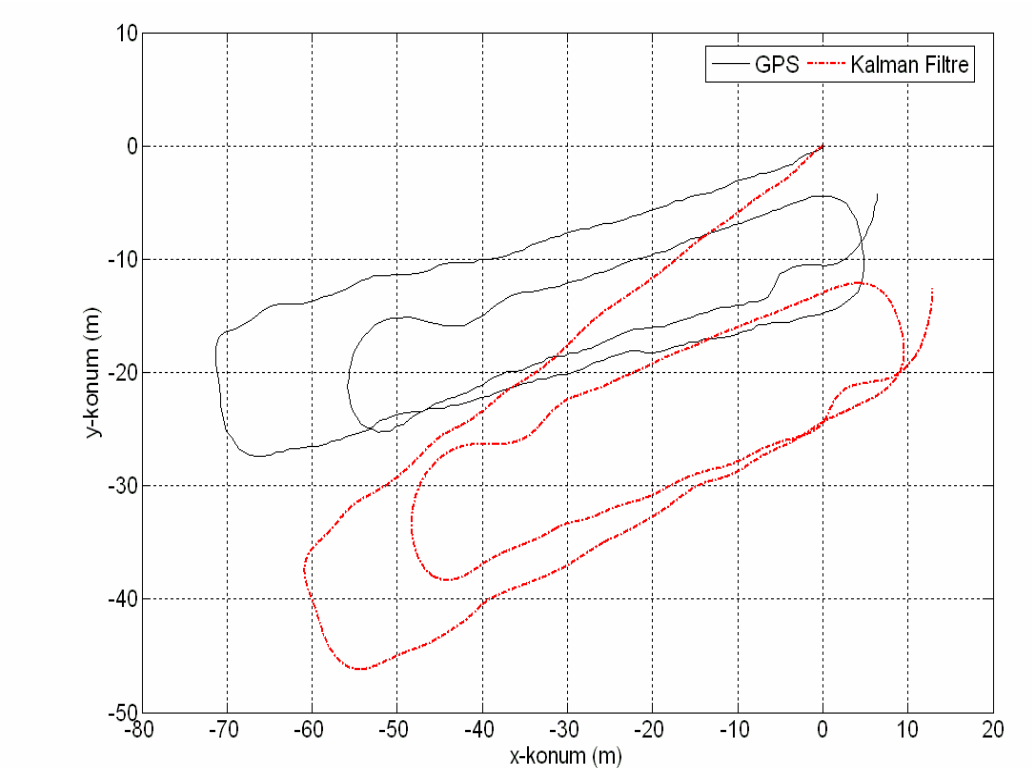
Kalman filtresine GPS desteği uygulandığında G_1 için kestirilmiş güzergahın değişimi Şekil 6.33' de verilmiştir. GPS desteği MATLAB programında bulunan TID dizisindeki rasgele zamanlara göre uygulanmıştır. Bu zamanlar kare noktalar halinde güzergahın üzerinde görülmektedir. Şekil 6.34'de G_1 için anlık RMS konum hata grafiği verilmiştir. Burada anlık RMS hataların 0 m ile 4 m arasında değiştiği görülmektedir. Çizelge 6.10' da toplam RMS hatasının 1 m' nin altına, kestirilen güzergahtaki x ve y yönündeki kaymanın sırasıyla -0.06 m ve -0.24 m' ye düştüğü gözlenmektedir. x ve y için VAF değerlerinin %90 üzerine çıktığı ve ϕ için VAF değerinin düştüğü görülmektedir. ϕ için VAF değerinin düşük çıkmasının sebebi, GPS verisi mevcut olduğunda Kalman filtresinin ani bir şekilde yön değerini düzeltmeye gitmesi ve bu değişim de VAF değerini etkilemesidir. Şekil 6.35'de kestirilen G_1 güzergahı için x, y ve ϕ kestirim hataları görülmektedir

Çizelge 6.9 Kalman filtresi ile GPS desteği olmadan G_1 güzergahı için hatalar

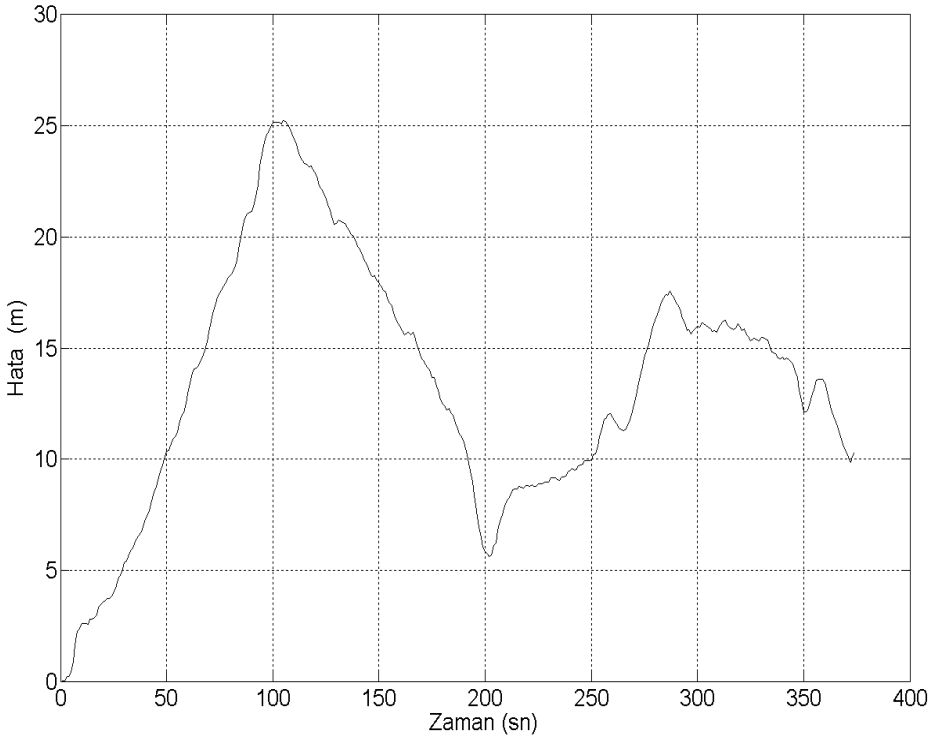
Ort. konum hatası	Ort. x konum hatası	Ort. y konum. hatası	Ort. ϕ yön hatası	RMS konum hatası	VAF(x)	VAF(y)	VAF(ϕ)
13.44 m	-6.85 m	11.1 m	-4.84°	3.66 m	%95.83	%55.17	%63.48

Çizelge 6.10 Kalman filtresi ile GPS destekli G_1 güzergahı için hatalar

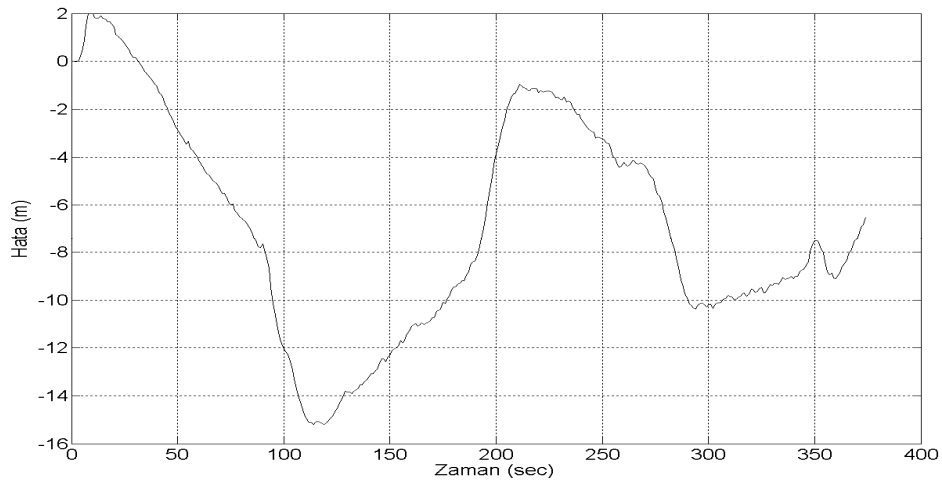
Ort. konum hatası	Ort. x konum hatası	Ort. y konum. hatası	Ort. ϕ yön hatası	RMS konum hatası	VAF(x)	VAF(y)	VAF(ϕ)
0.92 m	-0.06 m	-0.24 m	-3.47°	0.96 m	%99.88	%97.79	%46.05



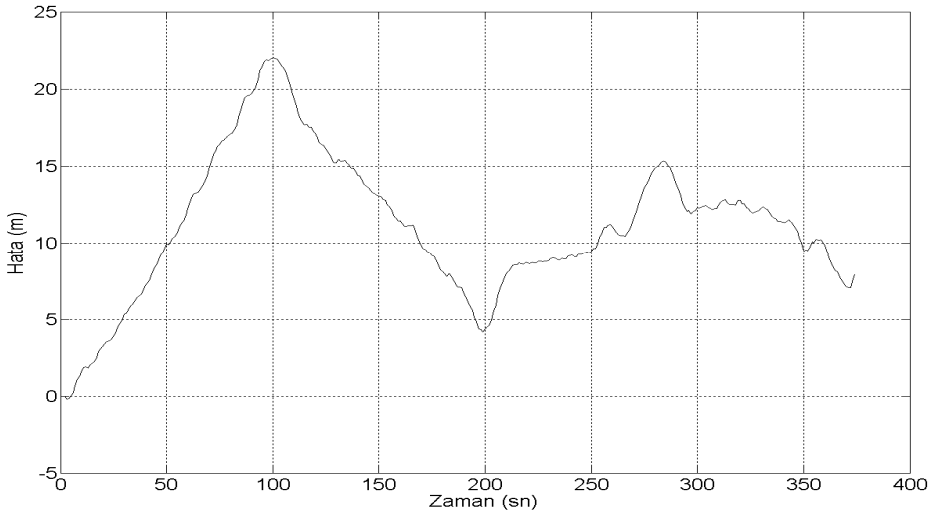
Şekil 6.30 GPS desteği olmadan Kalman filtresi ile G_1 güzergahının kestirilmesi



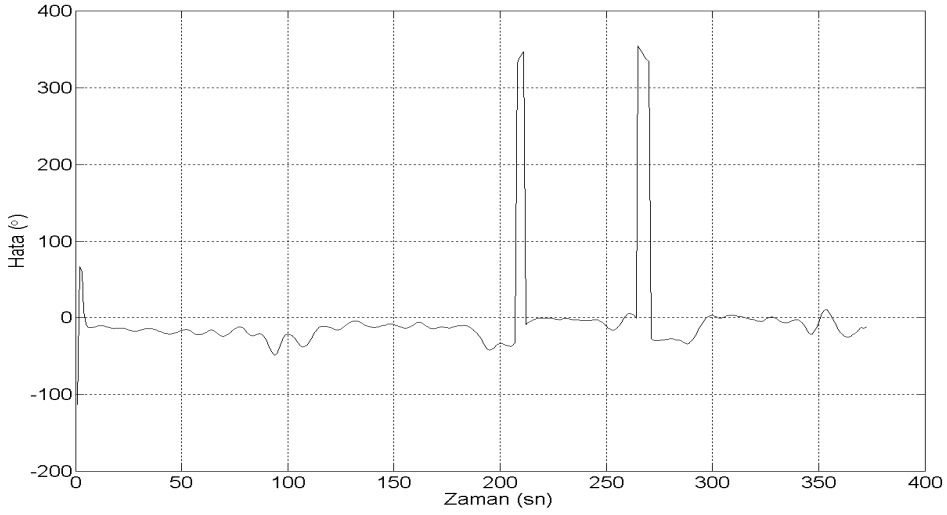
Şekil 6.31 G_1 güzergahı için RMS konum kestirim hatası



(a)

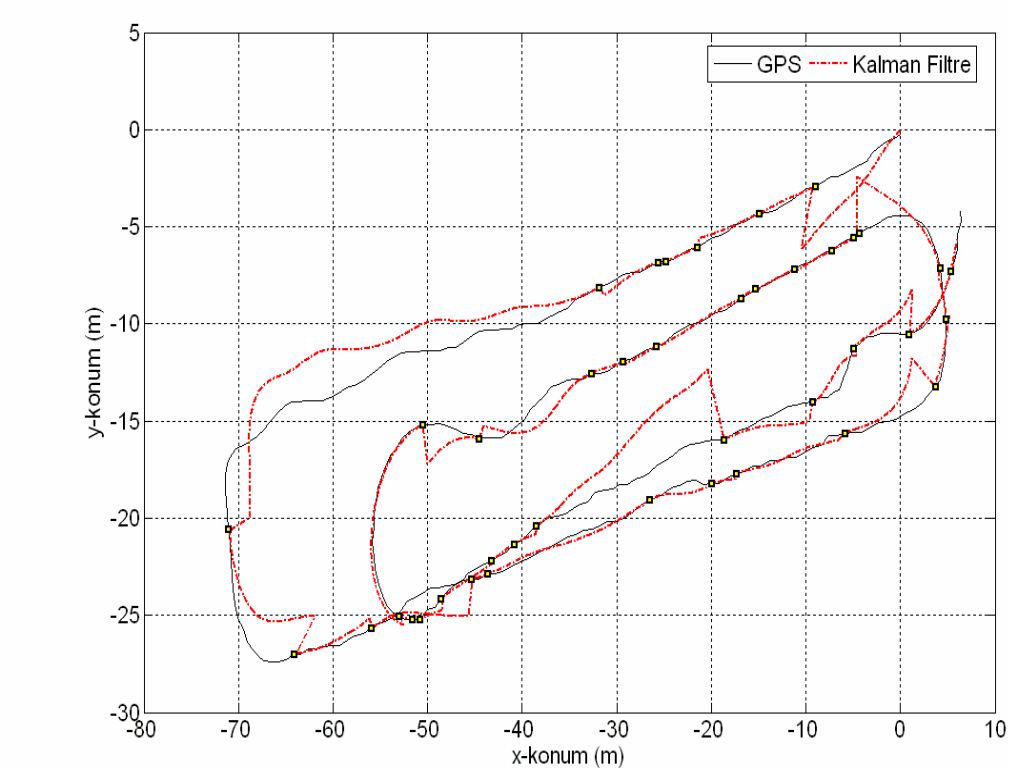


(b)

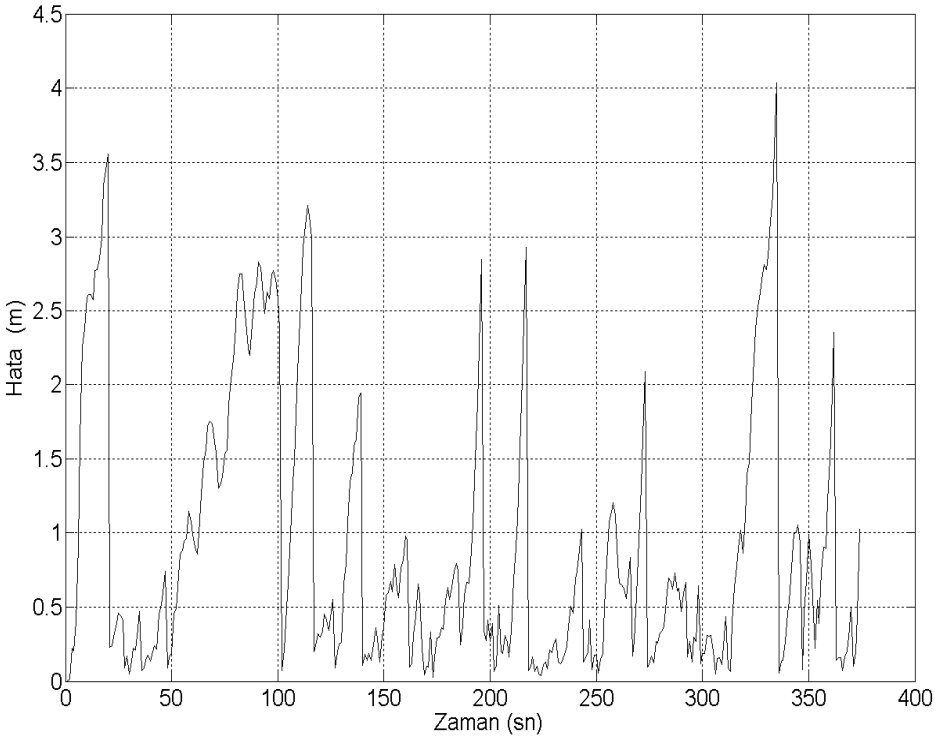


(c)

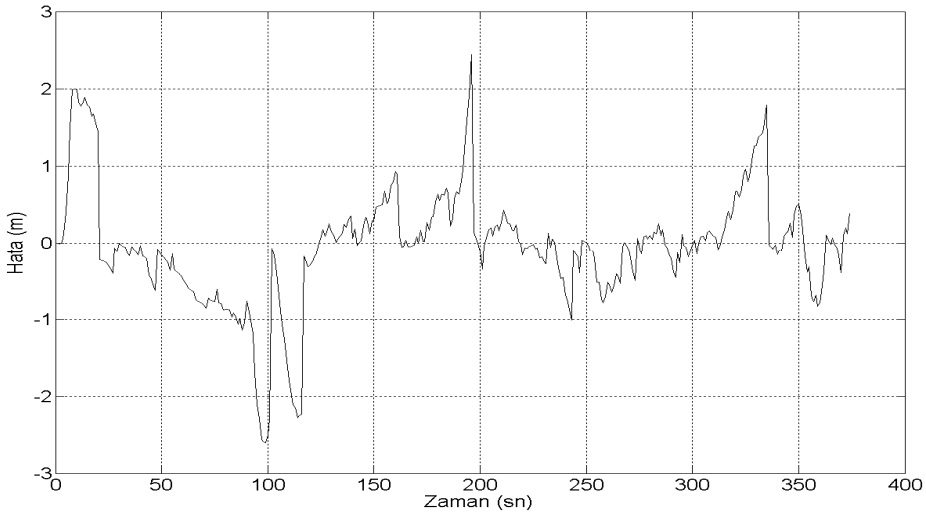
Şekil 6.32 (a) x yönündeki (b) y yönündeki ve (c) ϕ hataları



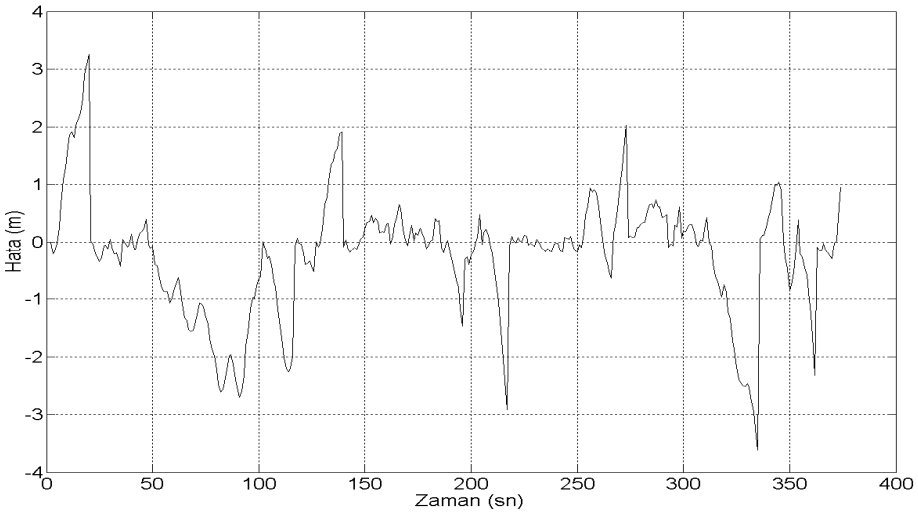
Şekil 6.33 GPS destekli Kalman filtresi ile G_1 güzergahının kestirilmesi,



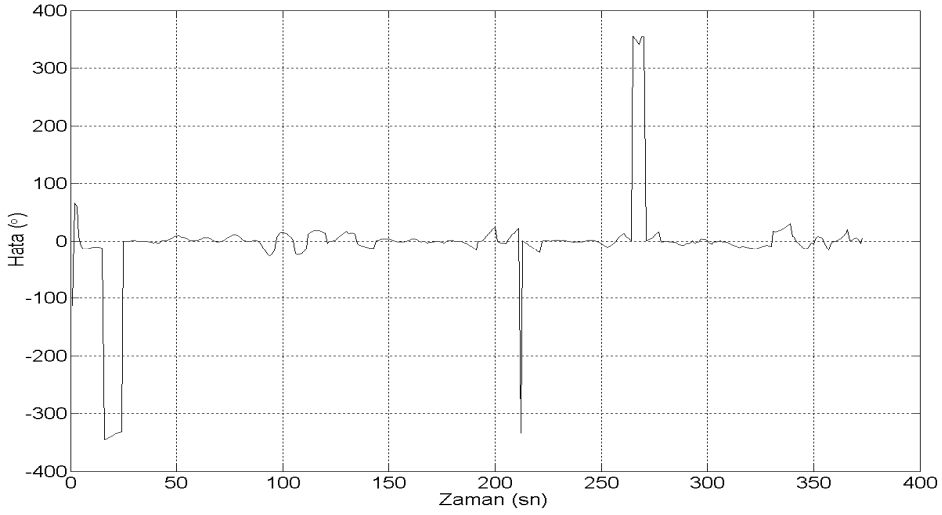
Şekil 6.34 G_1 güzergahı için RMS konum kestirim hatası



(a)



(b)



(c)

Şekil 6.35 (a) x yönündeki (b) y yönündeki ve (c) ϕ hataları

6.6.2 GPS desteği olmadan ve GPS destekli Kalman filtresi ile G2 test güzergahında konum kestirimi

Mobil robotun GPS desteği olmadan Kalman filtresi ile kestirilmiş G_2 güzergahının değişimi Şekil 6.36' da ve anlık RMS konum hata grafiği Şekil 6.37' de verilmiştir. Değişimden, hatanın 6 m' nin üzerine kadar çıktığı görülmektedir. GPS desteği olmadan Kalman filtresi ile kestirilen G_2 güzergahı, GPS ile elde edilen gerçek güzergaha göre fazla kaymamıştır. Çizelge 6.11' den x ve y yönündeki kaymadan dolayı oluşan ortalama x ve y yönündeki konum hatalarının sırasıyla 0.36 m ve 0.02 m olduğu söylenebilir. Toplam RMS konum hatasının 1.63 m, VAF değerlerinin de x ve y için %90 üzerinde ve ϕ için VAF değerinin %73.61 olduğu görülmektedir. Şekil 6.38' de ise x, y ve ϕ hatalarının değişimi görülmektedir.

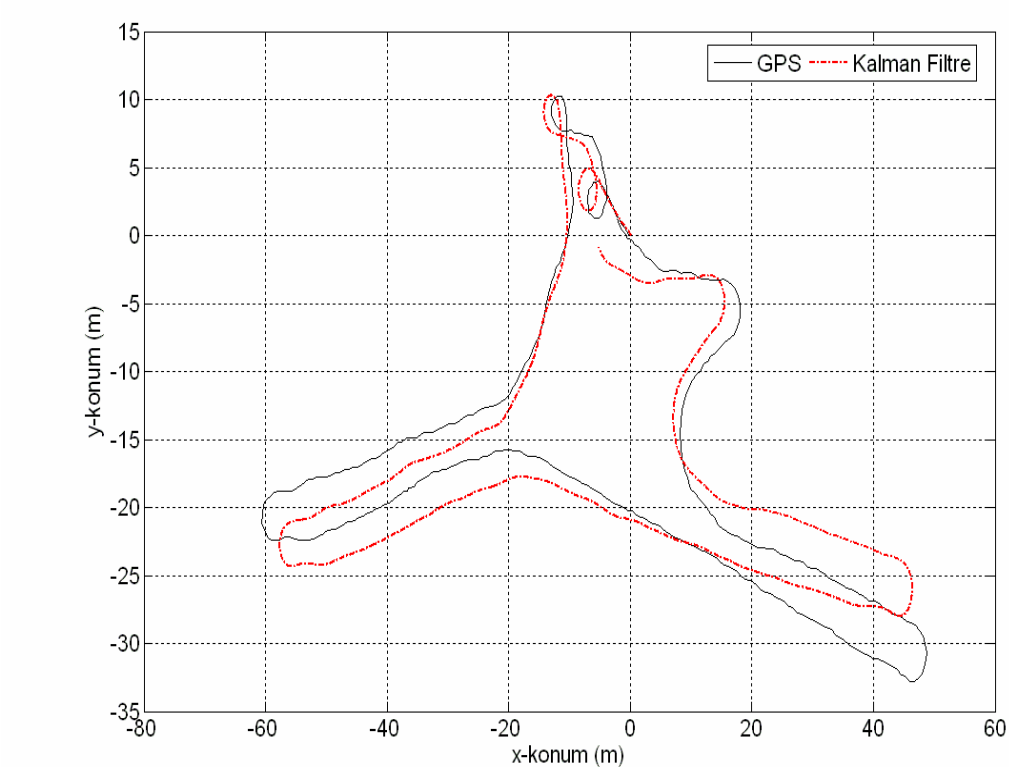
Mobil robotun GPS destekli Kalman filtresi ile kestirilmiş G_2 güzergahının değişimi Şekil 6.39'da ve anlık RMS konum hata grafiği Şekil 6.40' da verilmiştir. Bu grafikten RMS hatanın 0 m ile 2.5 m arasında değiştiği görülmektedir. Toplam RMS konum hatasının 1 m' nin altında, x ve y yönünde güzergahtaki kaymanın -0.08 ve 0.09 m düzeyinde olduğu Çizelge 6.12' de görülmektedir. VAF değerlerinin de %90' ın üzerine çıktığı bunun sonucunda da GPS ve kestirilen konum bilgilerinin birbirine benzerliğinin arttığı söylenebilir. Şekil 6.41'd e kestirilen G_1 güzergahı için x, y ve ϕ kestirim hataları verilmiştir. GPS desteğinin kestirilen konum bilgisinin doğruluğunu artırdığı görülmektedir.

Çizelge 6.11 Kalman filtresi ile GPS desteği olmadan G_2 güzergahı için hatalar

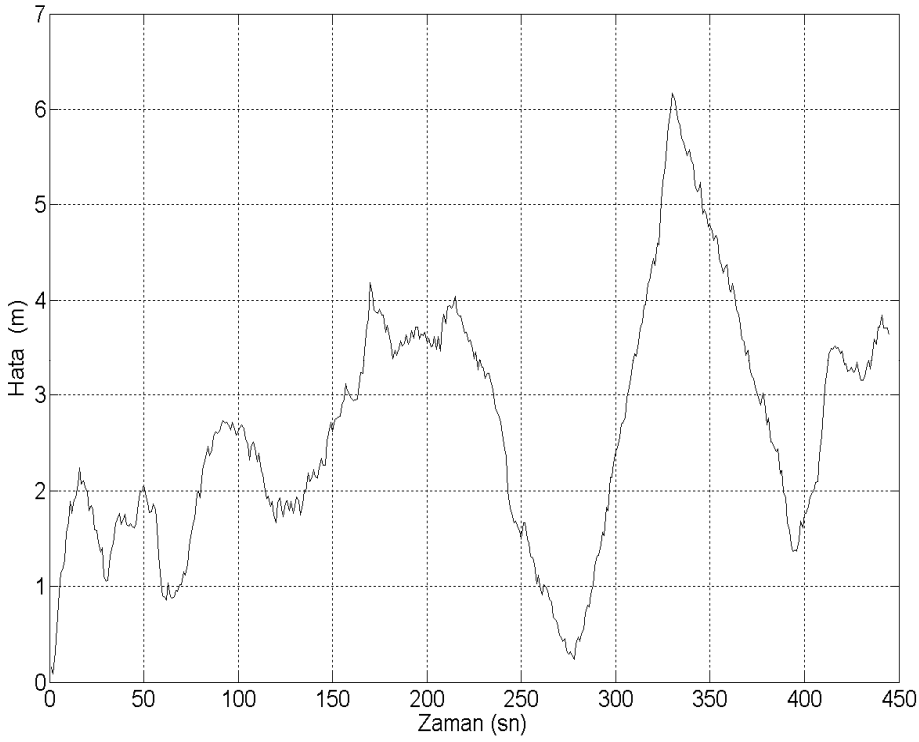
Ort. konum hatası	Ort. x konum hatası	Ort. y konum. hatası	Ort. ϕ yön hatası	RMS konum hatası	VAF(x)	VAF(y)	VAF(ϕ)
2.65 m	0.36 m	0.02 m	5.81°	1.63 m	%99.50	%96.70	%73.61

Çizelge 6.12 Kalman filtresi ile GPS destekli G_2 güzergahı için hatalar

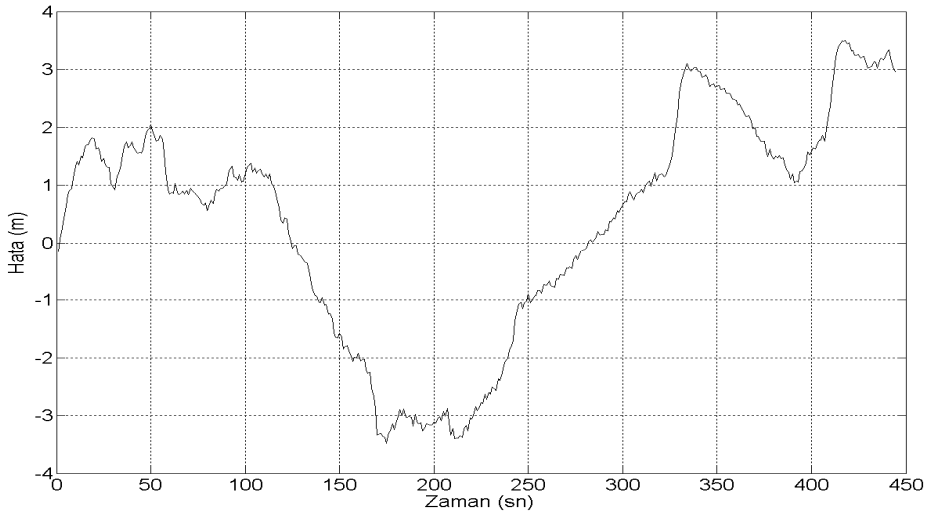
Ort. konum hatası	Ort. x konum hatası	Ort. y konum. hatası	Ort. ϕ yön hatası	RMS konum hatası	VAF(x)	VAF(y)	VAF(ϕ)
0.68 m	-0.08 m	0.09 m	1.97°	0.82 m	%99.95	%99.65	%90.16



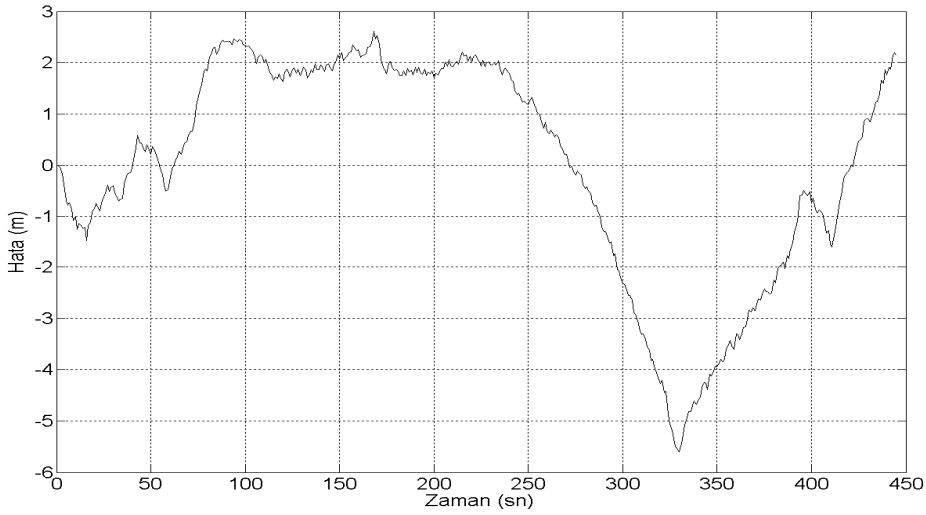
Şekil 6.36 GPS desteği olmadan Kalman filtresi G_2 güzergahının kestirilmesi



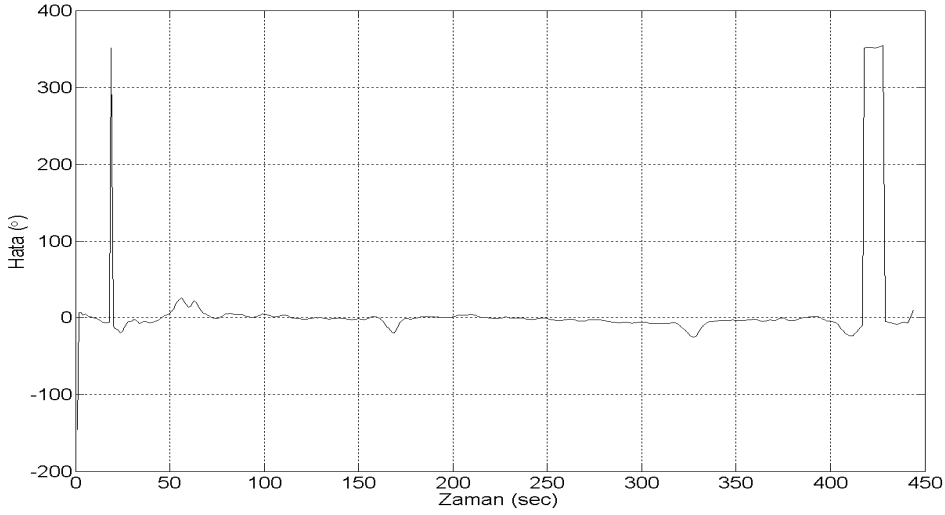
Şekil 6.37 G_2 güzergahı için RMS konum kestirim hatası



(a)

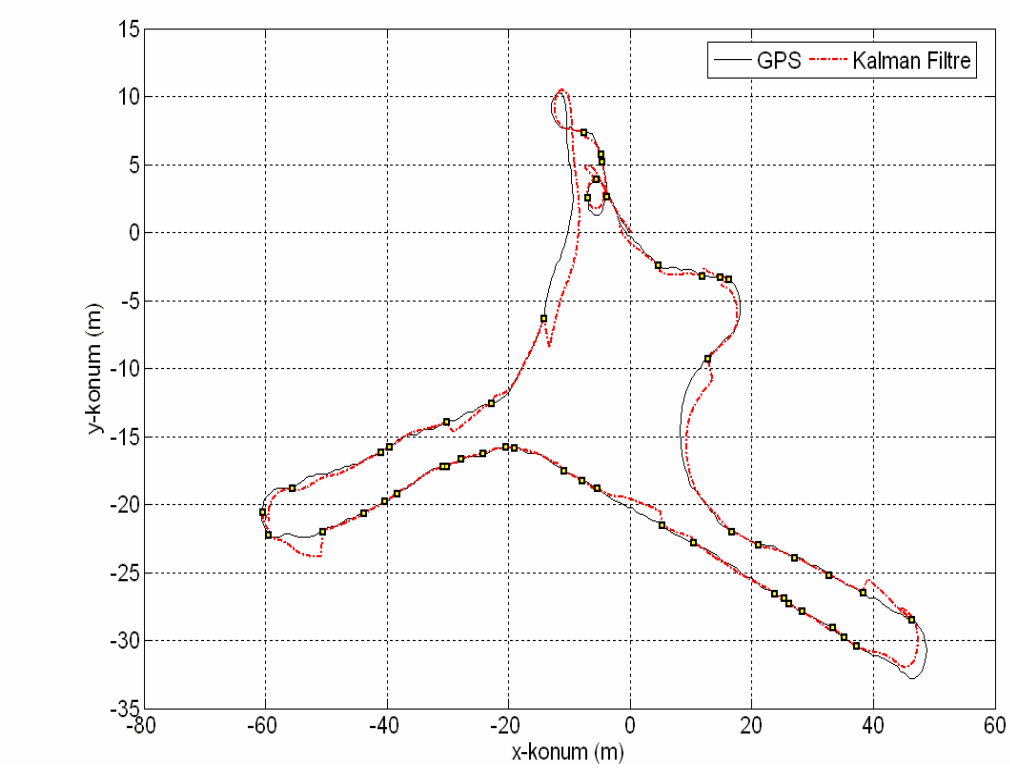


(b)

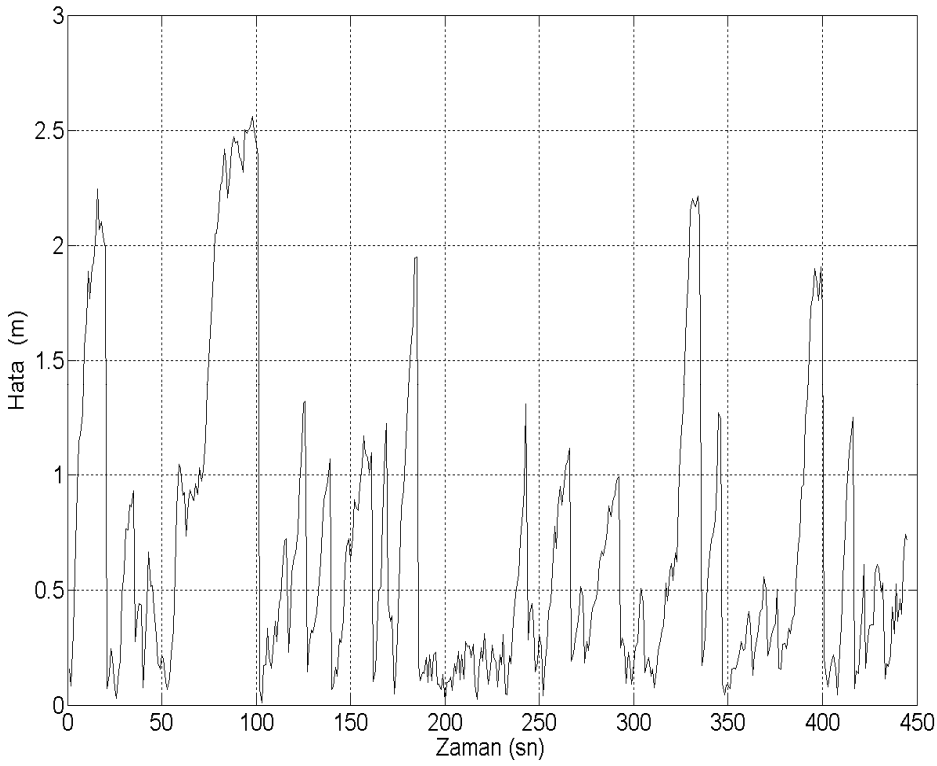


(c)

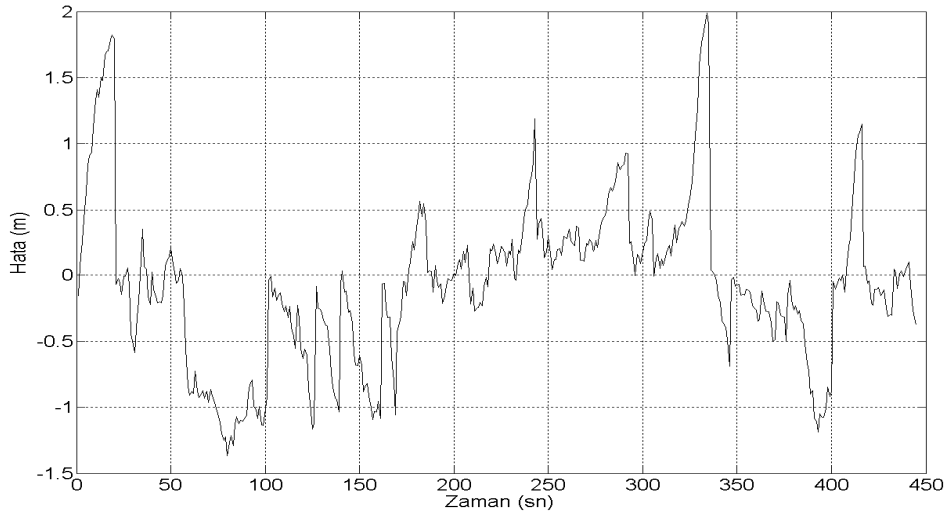
Şekil 6.38 (a) x yönündeki (b) y yönündeki ve (c) ϕ hataları



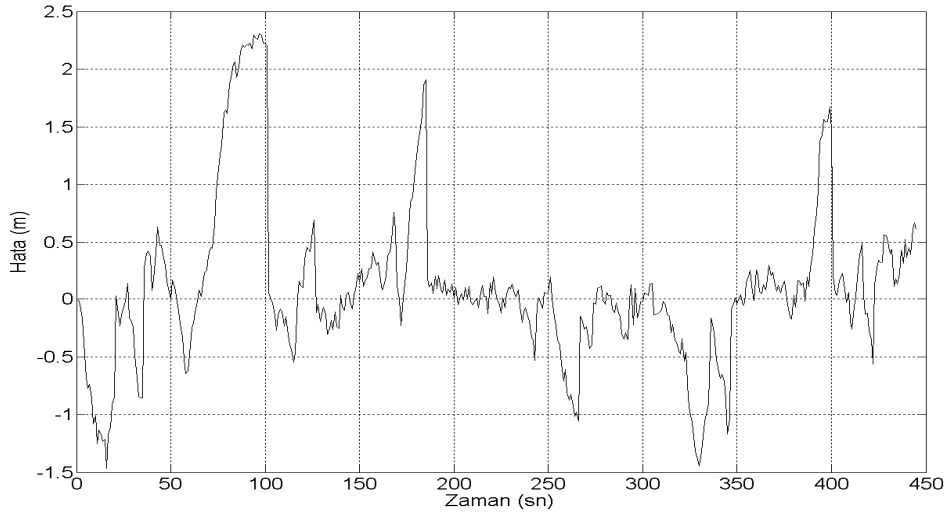
Şekil 6.39 GPS destekli Kalman filtresi ile G_2 güzergahının kestirilmesi



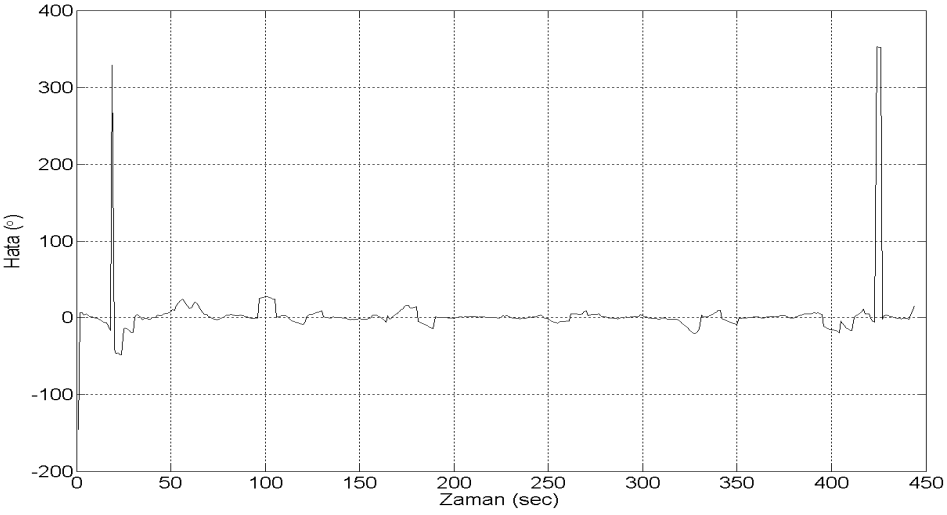
Şekil 6.40 G_2 güzergahında RMS konum kestirim hatası



(a)



(b)



(c)

Şekil 6.41 (a) x yönündeki (b) y yönündeki ve (c) ϕ hataları

7. SONUÇ VE ÖNERİLER

7.1 Sonuç

Bu tez çalışmasında, navigasyon algoritmalarını denemek, GPS ve jiroskop gibi navigasyon sensörlerinin uygun biçimde entegrasyonunu sağlamak amacıyla bir mobil robot platformu tasarlanmış ve gerçekleştirilmiştir. Navigasyon algoritmaları arasında kırk yılı aşkın bir zaman boyunca kullanılan ve geliştirilen Kalman filtresi algoritması ele alınmış ve tasarlanan mobil robotun test güzergahlarında seyri sırasında sensörlerden elde edilen veriler kullanılarak mobil robotun konumu kestirilmiştir. Klasik navigasyon yöntemi olarak da bilinen Kalman filtresine bir alternatif olarak düşünülen ve literatürde, yapay zeka alanında güncelliğini koruyan ve sürekli araştırma ve geliştirme konusu olan Yapay Sinir Ağları (YSA) yardımıyla mobil robot üzerinde bulunan sensörler kullanılarak navigasyon bilgileri elde edilmiştir. Farklı güncel teknolojilerden oluşan sensörlerin, tasarlanan veri toplama bilgisayarı, donanımı ve yazılımı yardımıyla entegrasyonu sağlanmıştır

Bu çalışmada, tasarımı gerçekleştirilen mobil robotun seyir halindeyken minimum hata ile konumunun kestirilmesi üzerinde durulmuştur. Bir mobil robotun, kendisine verilen görevi başarıyla gerçekleştirebilmesi için önceden belirlenen bir koordinat sisteminde nerede olduğunu bilmesi gerekmektedir. Bu problemin çözümüne, mobil robotun kinematik modelinin çıkarılması ile başlanabilir, ancak oluşturulan model yeteri kadar sistemi temsil etmeyebilir. Klasik metodlara alternatif olarak YSA ile mobil robotun modellenmesi yoluna gidilebilir. YSA'lar, bilinmeyen lineer ya da nonlineer bir sistemin giriş ve çıkışlarından sistemi eğitim yoluyla öğrenebilir. YSA ile oluşturulan sistem modeli, analitik yollardan oluşturulan modele girilmeyen parametreleri de içerir ve sistemi daha iyi temsil eder. Klasik metodlarla modellenen nonlineer sistem denklemlerinin, pratikte yazılım ve elektronik olarak ucuz bir işlemci ile hesaplamalarının ve sensör entegrasyonunun gerçekleştirilmesi mümkün olmayabilir. YSA ile modellenen sistemlerde ise, eğitildikten sonra ağırlık katsayılarının işlemciye kodlanması ile girişler ve çıkışlar arasındaki ilişki, çarpma ve toplama işlemleri ile gerçekleştirilir.

Bugüne kadar literatürde yapılmış çalışmalar incelendiğinde, mobil robotlar için kullanılan YSA ile konum kestirme yöntemlerinde mobil robot sisteminin modeli, statik olarak adlandırılabilecek YSA modelleri ile oluşturulmuştur. Oluşturulan bu modellerin genelleştirme performansı incelenmemiş ve eğitim güzergahı dışında, farklı güzergahlarda test edilmemiştir. Ayrıca YSA'nın kolay eğitimi için bir yöntem de önerilmemiştir. Birkaç çalışma ise, YSA'nın öğrenme algoritmaları üzerinde yapılan değişiklikler ile sınırlıdır.

Bu çalışmada;

- AR-GE amaçlı genel bir mobil robot, tasarlanmış ve gerçekleştirilmiştir.
- Mobil robotun jiroskop, enkoder ve GPS sensörleri ile entegrasyonu sağlanmıştır.
- Mobil robotun üzerinde sensör verilerini toplaması için PC104 tabanlı bir bilgisayar donanımı gerçekleştirilmiştir.
- C dilinde PC104 ortamında çalışan ve sensör verilerini eksiksiz toplayan bir program yazılmıştır.
- YSA-NARX modeli kullanılarak tasarlanan mobil robotun kinematik modeli oluşturulmuştur.
- YSA-NARX modelinin hızlı eğitimi amacıyla bir eğitim güzergahı oluşturulmuş ve bu yapay güzergah iki ayrı test güzergahında deneysel çalışmalarla desteklenmiştir.
- Gerçekleştirilen mobil robotun YSA-NARX modeli, oluşturulan test güzergah verileri kullanılarak MATLAB ortamında simüle edilmiştir.
- YSA-NARX modeli kullanılarak belirlenen G_1 ve G_2 test güzergahlarında GPS desteği olmadan ve GPS destekli mobil robot konum kestirimi yapılmıştır
- Kalman filtresi yöntemi kullanılarak belirlenen test güzergahlarında, GPS desteği olmadan ve GPS destekli mobil robot konum kestirimi yapılmıştır.

- GPS desteği olmadan ve GPS destekli olarak YSA-NARX modelinin genelleştirme özelliği, sistemin performansı, konum kestiriminde sağladığı kolaylık ve iyileştirmeler incelenmiştir.

Yapılan simülasyon ve elde edilen deneysel sonuçlardan, YSA'nın öğreneceği sistemin giriş ve çıkış verilerinden tümü hakkında eğitim yoluyla bilgi edindiği görülmüştür. Ayrıca, bir mobil robotun konum kestiriminde, GPS destekli ve GPS desteği olmadan YSA ve Kalman filtresi ile elde edilen sonuçların birbiriyle uyumlu olduğu görülmüştür. Kalman filtresinin performansını artırmak için mobil robot modelinin, daha kapsamlı oluşturulması gerekmektedir. YSA-NARX, mobil robotun modelini sadece sistemin giriş ve çıkışlarından öğrenebilmektedir. Kalman filtresinde ise mutlaka bir matematiksel model kullanmak gerekir.

YSA ve Kalman filtresinde görülen en temel ortak özellik ise mobil robotun yaptığı ani dönüşlerde konum kestiriminin doğrudan etkilenmesidir. Bu da YSA ve Kalman filtresinin yüksek frekans gürültülerine duyarlı olduğu sonucunu vermektedir.

YSA ile sistem tanılama teorisinde verilen NARMAX ve NOE YSA modellerinin yapılarında bulunan çıkıştan girişe doğru geri besleme bağlantısı, bu modelleri kararsızlığa sürükleyebilmektedir. Bu çalışmada elde edilen veriler kullanılarak farklı yapılarda NARMAX ve NOE modelleri ile konum kestirim sistemi tasarlanmaya çalışılmıştır. Ancak YSA eğitimi sırasında, YSA modeli kararsız duruma girmiş ve elde edilen simülasyon sonuçlarının anlamsız olduğu görülmüştür. Bu nedenle bu tez çalışmasında, NARMAX ve NOE modelleri ile yapılan çalışmalara yer verilmemiştir.

7.2 Öneriler

Yapılan bu çalışmadan elde edilen olumlu sonuçlar göz önüne alındığında, YSA ve yapay zeka tekniklerinin, mobil robot navigasyon sistemlerinde daha yaygın olarak kullanılması sonucunda daha ileri navigasyon sistemlerinin geliştirilmesi mümkündür. Mobil robot konusu; yazılım, sensör, mekanik ve elektronik donanım tasarımı gibi birçok disiplini içinde barındıran bir bilim dalıdır. Bu çalışmanın, çok yönlü olması ve güncel teknolojileri içermesi açısından, ülkemizde bu alanda bir

açığın kapatılmasına katkıda bulunacağı ve oluşturulan bilgi birikiminin gelecekte yeni çalışmalara da ışık tutacağı düşünülmektedir.

Bu tez çalışmasının bir uzantısı olarak;

- YSA algoritmasının doğrudan mobil robotun bilgisayarına yüklenmesi ve gerçek zamanda sistem tanılama yapılması,
- Son teknoloji olan YSA mikroçipleri (neurochip) kullanılarak ve YSA algoritmasını sistemin elektronik donanımına programlayarak mobil robotta gömülü donanım oluşturulması,
- MATLAB programlama dilinde geliştirilen ve geliştirilecek olan YSA veya farklı navigasyon algoritmalarını, MATLAB ortamında PC104 kütüphaneleri yardımıyla gerçek zamanda C diline çevirecek altyapının oluşturulması ve etkileşimli bir araştırma geliştirme platformunun gerçekleştirilmesi,
- Değişik bağıl sensörler kullanılarak (ivmeölçer gibi) veya üç eksenli ataletsel ölçüm birimi yardımıyla YSA ile sensör entegrasyonunun gerçekleştirilmesi,
- Hava ve deniz robotlarına mevcut yöntemin uyarlanması,
- Yeni algoritmaların ve tekniklerin sistem için oluşturulan yazılımlara kütüphane olarak eklenmesi suretiyle sistemin çalışma kapasitesinin geliştirilmesi,
- Günümüzde kullanılan araç takip sistemlerine uyarlanması,
- YSA, fuzzy ve genetik algoritma gibi yapay zeka tekniklerinin hibrid olarak kullanılması,

gibi ileriye yönelik bazı önemli çalışmalar yapılabilir.

8. KAYNAKLAR

Abulafya N., 1995, Neural Network for System Identification and Control, Imperial College of Science, Technology and Medicine, London University

Barshan B., Whyte D., 1995, A Low Cost Fiber Optic Gyro for Land Navigation, IEEE Trans. on Robotics and Automation, Vol. 11, No., 3 June 1995, pp:328-342

Billings, S.A., Voon, W.S.F., 1986, Structure Detection and Model Validity Tests in the Identification of Non-Linear Systems, Proc. IEE, Part D, Vol. 127, pp: 272-285

Bonnifait Ph., Garcia G., 1998, Design and Experimental Validation of an Odometric and Goniometric Localization System for Outdoor Robot Vehicles, IEEE Trans. on Robotics and Automation, Vol. 14, No. 4, pp: 541-548

Borenstein J., 1998, Experimental Evaluation of Fiber Optics Gyroscope for Improving Dead-Reckoning Accuracy in Mobile Robots, Proc. of the IEEE Int. Conf. on Robotics and Automation, Leuven, Belgium, pp:3456-3461

Borenstein J., Feng L., 1996, Gyrodometry: A new Method for Combining Data from Gyros and Odometry in Mobile Robots, Proc. of the 1996 IEEE Int. Conf. on Robotics and Automation, Minneapolis, pp: 423-428

Borenstein J., Feng L., 1996, Measurement and Correction of Systematic Odometry Errors in Mobile Robots, IEEE Trans. on Robotics and Automation, Vol. 12, No. 6, pp: 869-880

Brown R. G, Hwang P. Y. C., 1999, Introduction to Random Signals and Applied Kalman Filtering, Third Edition, John Wiley &Sons

Camapum, J. F., Turner K.J., 1996, Possible Back Propagation Neural Network Structures for AGV Localization, Proc. of the 5th Int. Conf. on Optimization of Electric and Electronic Equipment OPTIM'96, The Transylvania University of Brasov, pp:1731-1739.

Canan, S., Akkaya, R., 2005, An Artificial Neural Network Dead Reckoning /GPS Integrated Navigation System for Mobile Robot, Int. Symp. on Innovations in Intelligent Systems and Applications, 15-18 June, İstanbul, Turkey, pp: 330-333

Canan, S., Akkaya, R., Ergintav, S., 2004, Extended Kalman Filter Sensor Fusion and Application to Mobile Robot, 12th SIU2004 meeting, Kuşadası/İzmir, Turkey, pp 771-774

Dorobantu R., Zebhauser B., 1999, Field Evaluation of a Low-Cost Strapdown IMU by Means GPS, Ortung and Navigation, DGON, Bonn, pp: 51-65

Farrel J., Barth M., 1999, "The Global Positioning System and Inertial Navigation", McGraw Hill

Faruqi F.A at al., 2000, Extended Kalman Filter Synthesis for Integrated Global/Inertial navigation systems, Elsevier, Applied Mathematics and Computations, Vol. 115, pp: 213-237

Foresee F. D., Hagan M. T., 1997, Gauss-Newton Approximation to Bayesian Regularization, Proc. of the 1997 Int. Joint Conf. on Neural Networks, pp:1930-1935

Gaudin J. M., 2000, Conseption d'un System Hybrid GPS-Estime pour la Localisation en Condition de Visibilite Satellitaire Reduite, PhD Thesis, Université Montpelier de Science et Technique du Languedoc

Grewal M.S., Weill L. R, Andrews A. P., 2001, Global Positioning Systems, Inertial Navigation, and Integration, Wiley Inter-Science

Gu D., Hu H. 2002, Neural Predictive Control for Car-Like Mobile Robot, Int. Journal of Robotics Autonomous Systems, Vol. 39, No. 2, pp:73-86

Haykin S., 1994, Neural Networks: A Comprehensive Foundation, Macmillan College Publishing Comp. Inc.

He X., Asada H., 1993, A New Method for Identifying Model Orders of Nonlinear Input-Output Models, Proc. of American Control Conf., San Francisco, California, pp: 2520-2523

Hopfield J. J., 1982, Neural Networks and Physical Systems with Emergent Collective Computational Abilities, Proc. National Academy of Sciences, USA, April, Vol.79, pp: 2554-2558

Kalman R.E., 1960, A New Approach to Linear Filtering and Prediction Problems, Transaction of the Journal of Basic Engineering, Vol. 82, pp: 35-45

Karna K. N., Breen D. M., 1989, An Artificial Neural Networks Tutorial: Basics, Neural Networks, Vol.1, No:1, pp: 4 -23

Kayton M., 2000, Navigation Systems, CRC press LLC

Kelly A., 2001, General Solution for Linearized Systematic Error Propagation in Vehicle Odometry, Proc. of the 2001 IEEE/RSJ, Int. Conf. on Intelligent Robots and Systems, Maui, Hawaii, October 29-November 3,USA, pp: 1938-1945

Kim J. H, Cho H. S 1993, An improved Dead Reckoning Scheme for a Mobile Robot Using Neural Networks, Pergamon Press, Mechatronics, Vol. 3, No. 5, pp: 625–645

Krishnaswami V., Kim Y. W., Rizzoni G., 1995, A New Model Order Identification Algorithm with Application to Automobile Oxygen Sensor Modeling, Proc. of the American Control Conf., Seattle, Washington, pp: 2113-2117

KVH, 1999, KVH E-Core1000 Fiber Optic Gyro Technical Manual

Lawrence S. at al, 1997, Lessons in Neural Network Training: Overfitting May be Harder Expected, Proc. of Fourteenth National Conf. on Artificial Intelligence, AAAI-97, AAAI Pres, Menlo Park, California, pp: 540-545

Lee J.H., at al., 2000, A Two-antenna GPS Receiver Integrated with Dead Reckoning Sensors, IAIN World Congress in Association with the U.S. ION 56th Annual Meeting, June 26-28, San Diego, California, pp: 699

MacKay D. J. C., 1992, Bayesian Interpolation, Neural Computation, Vol. 4, no. 3, pp: 415–447.

Martinelli A., 2002, Evaluating the Odometry Error of a Mobile Robot, Proc. of the IEEE/RSJ, Int. Conf. on Intelligent Robots and Systems, EPFL, Lausanne, Switzerland, pp:853-858

Matthews, A., 1990, Utilization of Fiber Optic Gyros in Inertial Measurement Units, Navigation: Journal of the Institute of Navigation, Vol. 37, No. 1. pp: 17-38

McCulloh W., Pitts W., 1943, A Logical Calculus of The Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics., Vol.5, pp:115-133

Minsky M. L., Papert S. A., 1969, Perceptron, Expanded Edition, Cambridge, MA: MIT Press, London, UK

Nebot E., Hugh Durrant-Whyte H., 1999, Initial calibration and low-cost inertial navigation units for land vehicle applications, John Wiley & Sons Inc, Journal of Robotic Systems, Vol. 16, pp: 91–92

NMEA, 2005, http://www.nmea.org/

Norgaard M., Ravn O., 2001, NNSYS and NNCTRL Tools for System Identification and Control with Neural Networks, Computing & Control Engineering Journal, February, pp: 29-36

Ojeda L, Borenstein J., 2000, Experimental Results with the KVH C-100 Fluxgate Compass in Mobile Robots, Proc. of IASTED Int. Conf. Robotics and Applications, Honolulu, Hawaii, USA, pp:318-323

Peyret F., Betailler D, Garcia G., Carrel A., 2004, Precision on a Roll DGPS Solutions for Heavy Machinery, GPS World, January, pp: 20-29

Pham D. T., Liu X., 1997, Neural Networks for Identification, Prediction and Control, Springer Verlag

Rintanen K. et al., 1996, Development of an Autonomous Navigation System for an Outdoor Vehicle, Control Engineering Practice, Pergamon Elsevier Science,Vol. 4, pp: 499-505

Rivals at al., 1994, Modeling and Control of Mobile Robots and Intelligent Vehicles by Neural Networks, Proc. of the IEEE Conf. on Intelligent Vehicles, October 24-26, Paris, France, pp: 137-142

Rosenblat F., 1958, The Perceptron: A Probabilistic Model for Information Storage and Organization in h Brain, Psychological Review Vol. 65, pp: 386-408

Rumelhart D. E., McClelland J. L., 1986, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, (Vol.1), MIT Press, Cambridge, Massachusetts

Ryu K. B., 2001, Application of the Sound Localization Algorithm to the Global Localization of a Robot, Undergraduated Report, University of Maryland

Sağıroğlu Ş., Beşdok E., Erler M., 2003, Mühendislikte Yapay Zeka Uygulamaları-1: Yapay Sinir Ağları, Ufuk Yayınevi, Kayseri.

Shoval S., Mishan A., Dayan J., 1998, Odometry and Triangulation data Fusion for Mobile-Robot Environment Recognition, Control Engineering Practice, Pergamon Elsevier Science, Vol.6, pp:1383-1388

Simon D., 2001, Kalman Filtering, Embedded System Programming, June, pp:72-79

Simpson P. S., 1990, Artificial Neural Networks, A Foundation, Paradigm, Applications and Implementation, Pergamon Press

Sirf, 2001, SiRFstrIIe Evaluation Kit User's Guide

Torii T., 2000, Research in Autonomous Agriculture Vehicles in Japan, Elsevier, Computers and Electronics in Agriculture, pp:133-153

Townsed N.W, Tarassenko L., 1999, Neural networks for mobile robot localization using infra-red range sensing, Springer-Verlag, Neural Computing and Applications, Vol 8, pp: 114–134

Trabelsi A., Lafont F., Kamoun M., ENEA G., 2004, Identification of Nonlinear Multivariable Systems by Adative Fuzzy Takagı-Sugena Model, Int. Journal of Computational Cognition, Vol. 2, N. 3, pp: 137-153

Tuncay S., Leblebicioğlu K., Özgen C., Halıcı U., 1999, A Rule Based Neuro-Optimal Controller for Nonlinear MIMO Systems, Proc. of the 7th Mediterranean Conf. and Automation (MED99), June 28-30, Haifa, Israel, pp:511-523

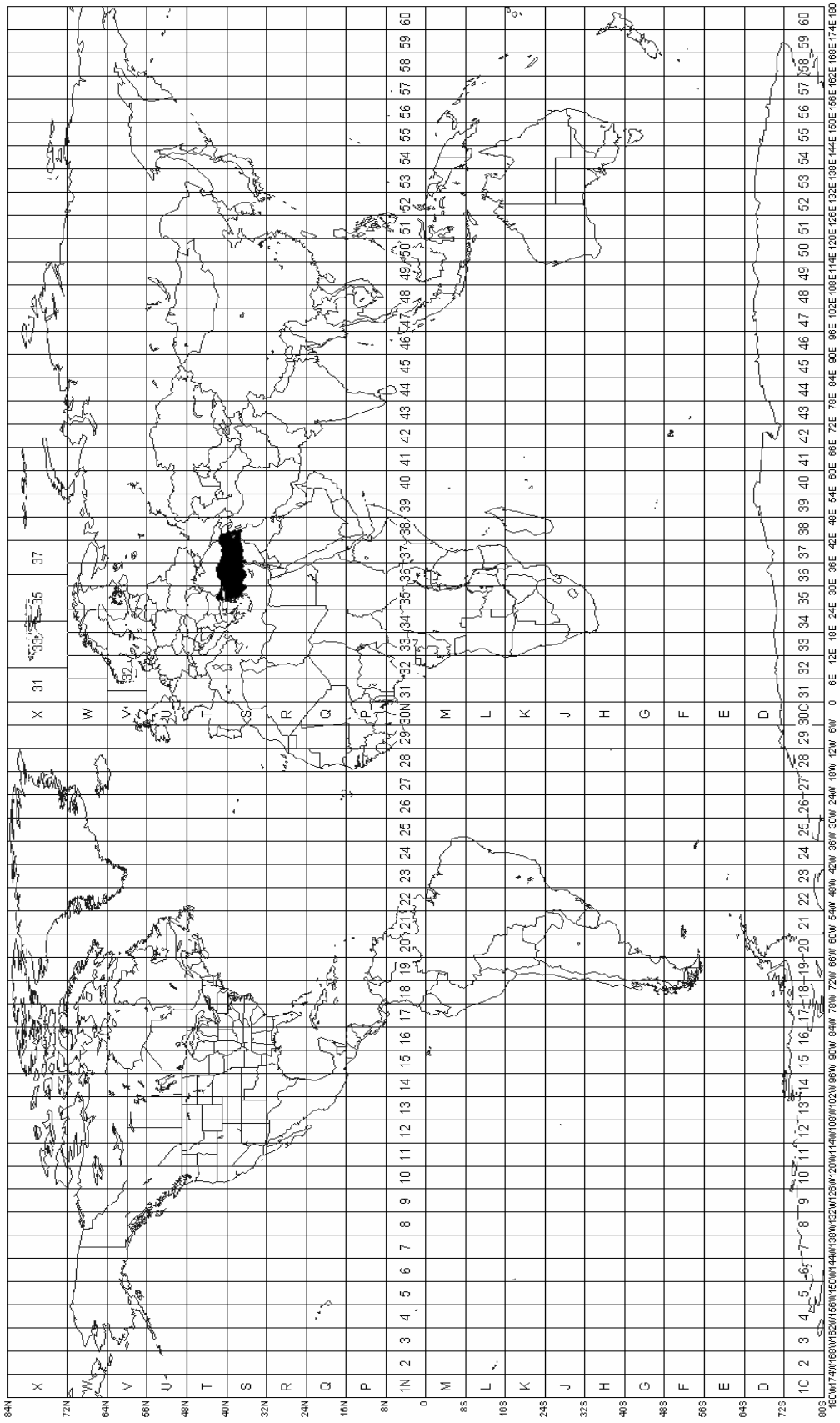
Welch, G. , Bishop, G., 2001, An introduction to the kalman filter., University of North Carolina at Chapel Hill, http://www.cs.unc.edu/~welch/kalman/

Wellenhof B. H., Lichtenegger H., Collins J., 1997, Global Positionning System Theory and Practice, Fourth Edition, Springer- Verlag, Wien

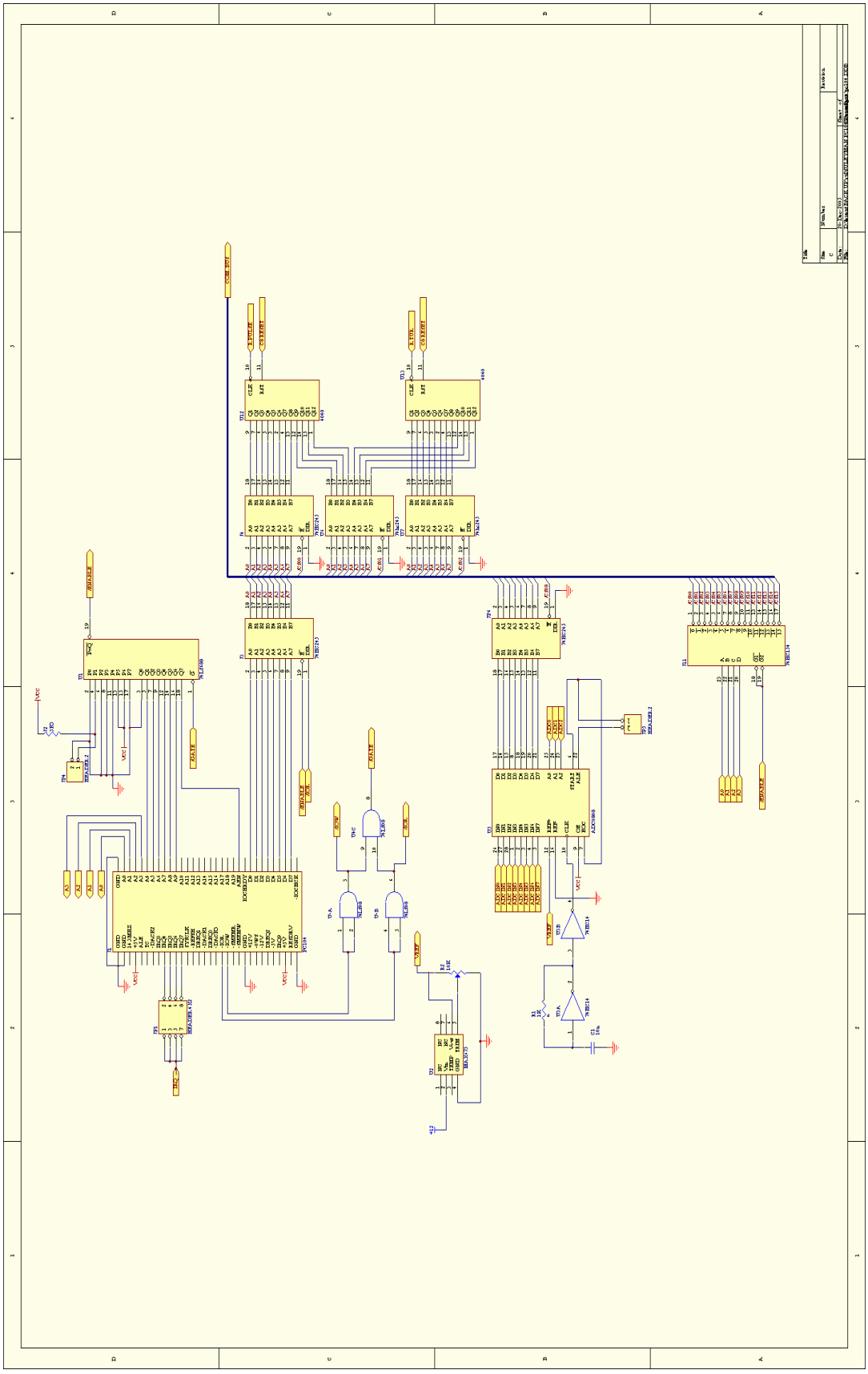
Widrow B., Hoff M. E., 1960., Adaptive Switching Circuits, Proc. IRE WESCON Convention Record, Part 4, IRE, New York, pp: 96-104

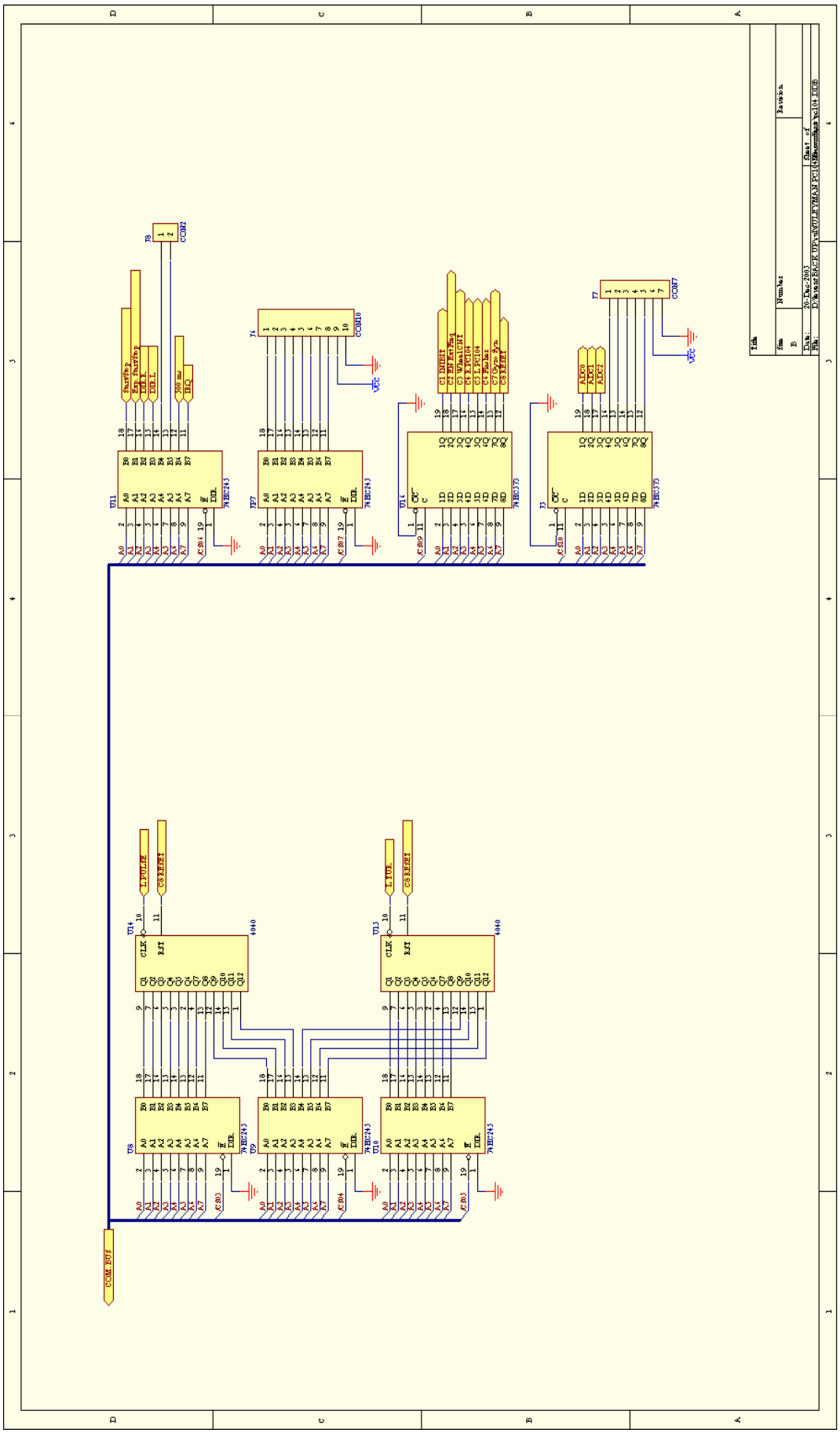
Yim Y.U., Oh S.Y., 2004, Modeling of vehicle dynamics from real vehicle measurements using a neural network with two-stage hybrid learning for accurate long term prediction, IEEE Trans. On Vehicular Technology, Vol. 53, No. 4, pp: 1076-1084

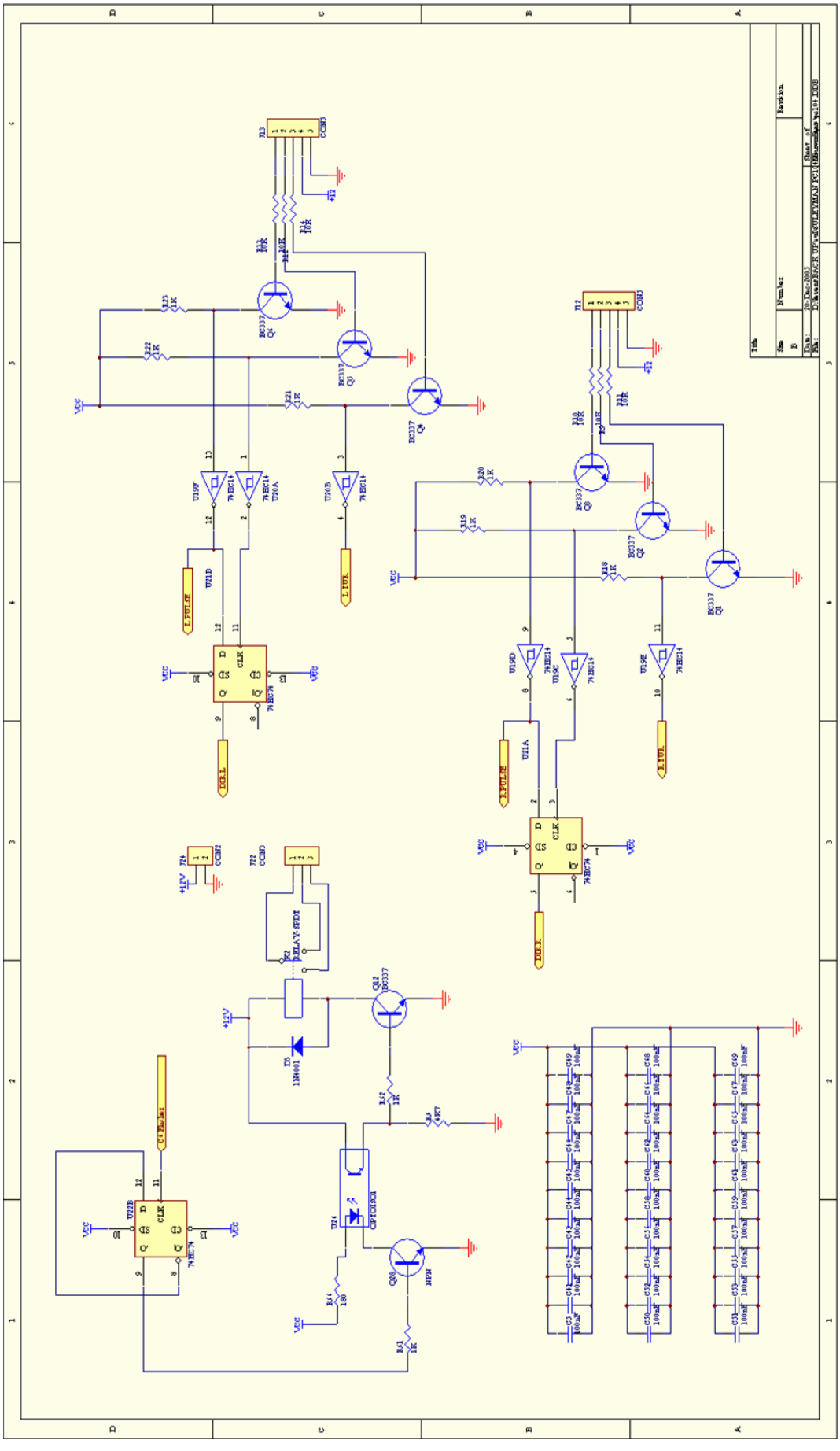
EK-A UTM ZON HARİTASI

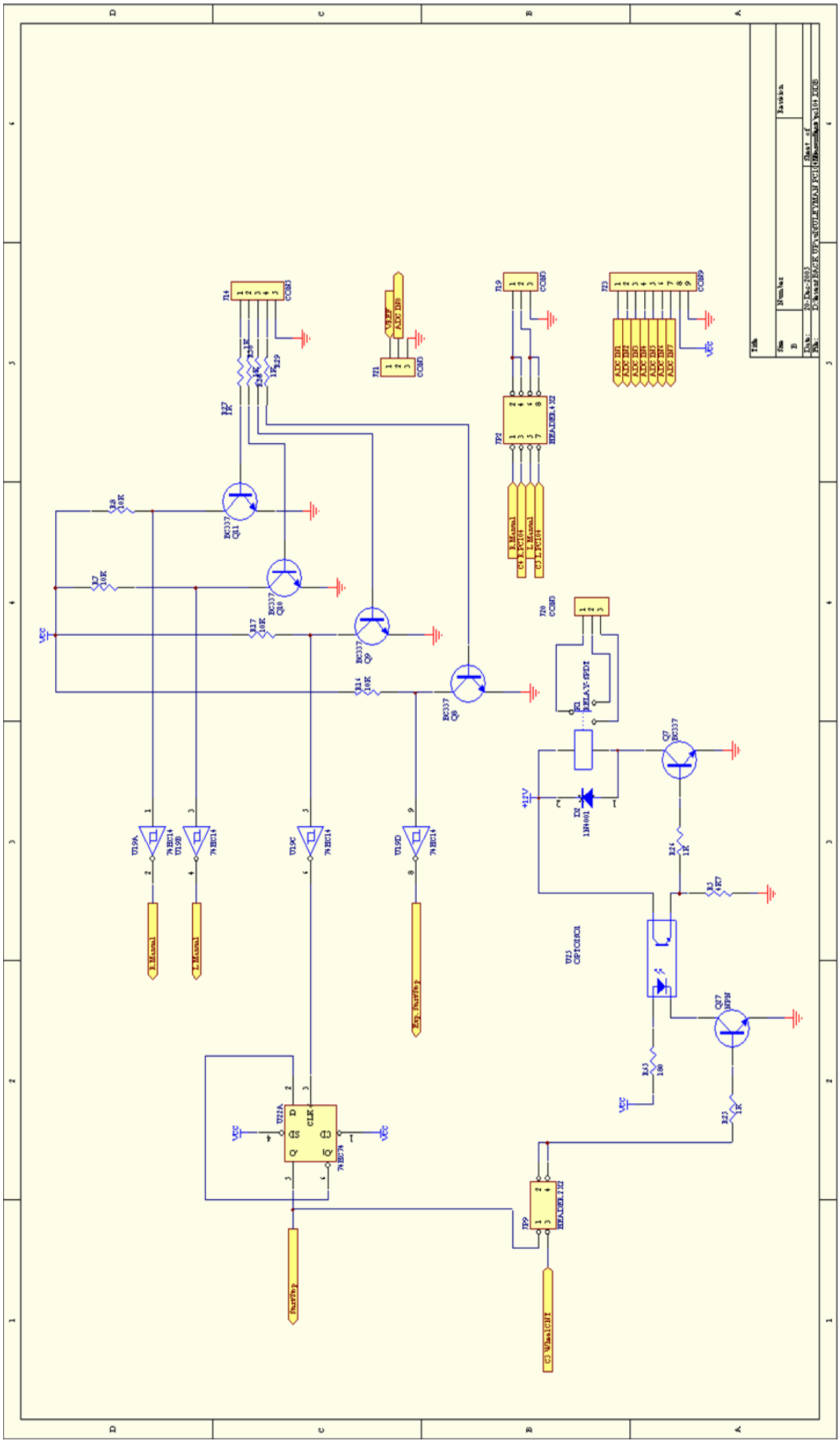


EK-B ISA KART DEVRE ŞEMASI









EK-C MOBİL ROBOT VERİ TOPLAMA YAZILIMI

ROBOT.C

```
/*   MAİN
**  Navigator
**  Started:  18/08/2003
** rev 11/04/2004
** rev 23/04/2004
** rev 07/05/2004
** rev 26/05/2005
**   Author: Suleyman CANAN
*/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <string.h>
#include <math.h>
#include "pcl4c.h"
#include "navlib.h"
#include "navlib.c"
#include "driver.c"


int main(void)
{


int dosyanum;
char *gpsfilename="gpsf";
char *gyrofilename="gyrof";
char *deneyfilename="deneyno.num";

outportb(PORT+10,0); //reset port 10 everything

dosyaDENEY=fopen(deneyfilename,"r+");
fscanf(dosyaDENEY,"%d",&dosyanum);
fclose(dosyaDENEY);


        clrscr();
        printf("Navigator Yazilimi V1.1\n");
        printf("Author: Suleyman Canan\n");
        printf("GYRO Baud Rate 9600 GPS Baud Rate 4800\n");


 Serial_Init();
 init_Robot();
 clrscr();
  sound(440);
  delay(2500);
  nosound();

/* while(1)
```

```
  {
        if(kbhit())
         {
                i = getch();
                switch((char)i)
                {
                case 'm' : flasher_ONOFF();
                break;
                case 'w' :
                outportb(PORT+9,4);   // Motor ileri
                outportb(PORT+9,0);
                delay(500);
                outportb(PORT+9,4);   // Motor ileri
                outportb(PORT+9,0);
                break;
                case 'r' : //dumen saga
                outportb(PORT+10,8);
                delay(50);
                outportb(PORT+10,0);
                break;
                case 'l' : // dumen sola
                outportb(PORT+10,16);
                delay(50);
                outportb(PORT+10,0);
                break;
                case 'b' :
                outportb(PORT+10, 32);// Motor geri
                delay(500);
                outportb(PORT+10,0);
                break;
                case 'x' :
                reset_Counters();
                outportb(PORT+10,0); //reset port 10 everything
                break;
                case CTLZ:
                 // restore COM port status & exit
                  SioDone(COM1);
                  SioDone(COM2);
                  fclose(dosyaGPS);
                  fclose(dosyaGYRO);
                  exit(0);
                }
          }*/
while(1)
  {
        if(kbhit())
         {
                i = getch();
                switch((char)i)
                {
                case 'e' :
                Echo=1;
                sound(440);
                delay(250);
                nosound();
                break;

                case 'f' :
```

```
Echo=0;
sound(440);
delay(250);
nosound();
break;

case 's' :
if(!dosyaflag)
{
itoa(dosyanum, dosyacount, 10);

strcpy(dosya_adi_GPS,gpsfilename);
strcat(dosya_adi_GPS,dosyacount);
strcat(dosya_adi_GPS,uzanti);
dosyaGPS=fopen(dosya_adi_GPS,"w+");

strcpy(dosya_adi_GYRO,gyrofilename);
strcat(dosya_adi_GYRO,dosyacount);
strcat(dosya_adi_GYRO,uzanti);
dosyaGYRO=fopen(dosya_adi_GYRO,"w+");
++dosyanum;
dosyaflag=1;
}

break;
case 'q' :
if(dosyaflag)
{
fclose(dosyaGPS);
fclose(dosyaGYRO);
dosyaflag=0;
}
break;

case 'b' :
outportb(PORT+10, 32);
delay(1000);
outportb(PORT+10,0);
break;

case 'x' :
reset_Counters();
outportb(PORT+10,0); //reset port 10 everything
break;

case 'z' :
GyroAngle=0;
break;

case CTLZ:
 // restore COM port status & exit
if(dosyaflag)
{
sound(440);
delay(10);
nosound();
}
            else
```

```
                        {
                        SioDone(COM1);
                        SioDone(COM2);

                        dosyaDENEY=fopen(deneyfilename,"r+");
                        fprintf(dosyaDENEY,"%d",dosyanum);
                        fclose(dosyaDENEY);

                        //fclose(dosyaGYRO);
                        exit(0);
                    }
                   }
              }


if(Echo)
        {

        gotoxy(2,4);
 //     printf("%s\n",gpsMsg_RMC);
 //     printf("%s\n",gpsMsg_GGA);
        printf("Right wheel %d\n",sensors.Pulse_Right);
        printf("Left wheel %d\n",sensors.Pulse_Left);
        printf("Gyro Angle %f\n",GyroAngle/10.0);
        printf("Gyro Rate  %f\n",GyroRate);
        printf("Dosya No: %d\n",dosyanum);
        printf("*%s* *%s* *%c* *%f* *%c* *%f*
*%c*\n",TmsgRMC.Protocol,TmsgRMC.UTC,TmsgRMC.Status,TmsgRMC.Lat,TmsgRMC.NS,Tms
gRMC.Lon,TmsgRMC.EW);
        printf("speed*%f* course*%f* date*%s* seconds
**%f**\n\n",TmsgRMC.Speed,TmsgRMC.Course,TmsgRMC.Date,TmsgRMC.SecStamp);

        }
/***************************************/
/************** GYRO/GPS DATA **********/
/***************************************/
getData_GYRO();
getData_GPS();
/***************************************/
/***************************************/
/***************************************/
  } /* end while */
} /* end main */
```

NAVLIB.C

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <string.h>
#include <math.h>
#include "pcl4c.h"
char Echo=0;
double GyroRate,GyroAngle;

/*******************************/
/**************GPS***************/
/*******************************/

void getData_GPS(void)
{

int veri_GPS;
//char gpsMsgs[100]="",gpsMsg_RMC[100]="",gpsMsg_GGA[100]="",QBuff[16]="";
char *nmeaMsgGGA="GPGGA",*nmeaMsgRMC="GPRMC";
int veri_COM2,j;
unsigned char ch_veri_GPS,i,ch,ch_count,Qcount;

veri_GPS= SioGetc(COM2,0);
            if (veri_GPS>-1)
            {

                  ch_veri_GPS=(char)veri_GPS; // int veri_GPS donustur char'a
                  if(ch_veri_GPS=='$')
                        {
                        ch_count=0;
                              while(ch_veri_GPS !=',')
                              {
                              veri_GPS= SioGetc(COM2,0);
                              if (veri_GPS>-1)
                                    {
                                    ch_veri_GPS=(char)veri_GPS;
                                    QBuff[ch_count++]=ch_veri_GPS;
                                    } //end if >-1
                               } //end while  ','

                        // GGA veya RMC mesaji ise GPS mesajini al yoksa gec
                  if(strncmp(QBuff,nmeaMsgRMC,4)==0||strncmp(QBuff,nmeaMsgGGA,4)==0)
                  {
                        strcpy(gpsMsgs,QBuff);

                        while(ch_veri_GPS !=0x0D) //until <LF>
                        {
                        veri_GPS= SioGetc(COM2,0);
                              if (veri_GPS>-1)
                              {
                                    ch_veri_GPS=(char)veri_GPS;
                                    gpsMsgs[ch_count++]=ch_veri_GPS;
                              } //end if
                         }  //end while
                                    gpsMsgs[ch_count]='\0';
```

```
                                        if(strncmp(QBuff,nmeaMsgRMC,4)==0)
                                        {
                                                strcpy(gpsMsg_RMC,gpsMsgs);
                                                parse_RMC(gpsMsg_RMC);
//                                              fprintf(dosyaGPS,"%s",gpsMsg_RMC);

                                        }

                                        if(strncmp(QBuff,nmeaMsgGGA,4)==0)
                                        {
                                                strcpy(gpsMsg_GGA,gpsMsgs);
//                                              if(Echo) printf("\n orij %s\n",gpsMsg_GGA);
                                                parse_GGA(gpsMsg_GGA);
                                        if (dosyaflag)
fprintf(dosyaGPS,"TStamp %f Status %c SatUsed %c HDOP %f Lat %f Lon %f Alt
%f\n",TmsgRMC.SecStamp,TmsgRMC.Status,TmsgGGA.SatUsed,TmsgGGA.HDOP,TmsgRMC.Lat
,TmsgRMC.Lon,TmsgGGA.Altitude);

                                        }

                        } //end if QBuff Nmea compare
                                } //end if $
                } //end if GPS*/

}
/*********************************/
/************GYRO***************/
/*********************************/

void getData_GYRO(void)
{
int veri_GYRO,veri_COM1,i=0;
unsigned char ChVeri_GYRO,veri,raw_GYRO[8];
//double GyroRate;
//static double GyroAngle;

        veri_GYRO= SioGetc(COM1,0);

                if (veri_GYRO>-1)
                {
                ChVeri_GYRO=(char)veri_GYRO;
/*
                                                printf("%02x \n ",ChVeri_GYRO);
                                                if ((ChVeri_GYRO&0x80)&&1) printf("****");
                                                for(veri=0x80;veri>1;veri>>=1)
                                                {
                                                printf("%d ",(ChVeri_GYRO&veri)&&1);
                                                }*/


                        if ((ChVeri_GYRO&0x80)&&1) //GYRO SYNCHRO Bit Test
                        {
                                raw_GYRO[0]=ChVeri_GYRO;
                                get_Sensor_Data();
                                        reset_Counters();
```

```
                                        for(i=1;i<8;i++)
                                        {
                                                veri_GYRO= SioGetc(COM1,0);
                                                if (veri_GYRO>-1)
                                                {
                                                        ChVeri_GYRO=(char)veri_GYRO;
                                                        raw_GYRO[i]=ChVeri_GYRO;
                                                }
                                        }

                                        KVH.Bits.Part0=raw_GYRO[7];
                                        KVH.Bits.Part1=raw_GYRO[6];
                                        KVH.Bits.Part2=raw_GYRO[5];
                                        GyroRate=KVH.GyroRate*0.00305; //bundan sonra test icin
integral al
                                        sensors.GyroRate=GyroRate;
                                        GyroAngle+=GyroRate;
//                                      printf("Gyro Angle %lf\n",GyroAngle/10.0);
//                                      printf("Gyro Rate %f\n",GyroRate);
if(dosyaflag)
fprintf(dosyaGYRO,"TStamp %5.2f RET %d REP %d LET %d LEP %d ANGLE %f Gyro
%f\n",TStamp.TStampInSec,sensors.Tur_Right,sensors.Pulse_Right,sensors.Tur_Left,sensors.Pulse_
Left,sensors.Steer_Angle,GyroRate);
        }
                }

/* buraya process_GYRO(ilk veri yolla) funksiyon
yap geri donus GYO Strucuru yap struct
GYRO{ long int anglerate; double Temperature}*/
}

void get_TStamp(void)
{
gettime(&zaman);
TStamp.TStampInSec=zaman.ti_hour*3600.0+zaman.ti_min*60.0+zaman.ti_sec+zaman.ti_hund/100.
0;
}


void parse_RMC(char TempMsgBuff[100])
{
        char *index,comma[20]="";
        int i=0,co=0;

        while(TempMsgBuff[i]!=NULL)
        {
                if (TempMsgBuff[i]==',')
                {
                        co++;
                        comma[co]=i;
                }
                i++;
         }

        index=TempMsgBuff;
                if ((comma[8]-comma[7])>1)
                {
                        strncpy(TmsgRMC.ChSpeed,index+comma[7]+1,comma[8]-comma[7]-1);
```

```
                TmsgRMC.Speed=atof(TmsgRMC.ChSpeed);
        }
                else TmsgRMC.Speed=0;

        if ((comma[8]-comma[7])>1)
        {
                strncpy(TmsgRMC.ChCourse,index+comma[8]+1,comma[9]-comma[8]-
1);
                TmsgRMC.Course=atof(TmsgRMC.ChCourse);
        }
                else TmsgRMC.Course=0;

        strncpy(TmsgRMC.Date,index+comma[9]+1,comma[10]-comma[9]-1);

        for(i=1;i<12;i++) TempMsgBuff[comma[i]]=32;

        sscanf(TempMsgBuff,"%5s %10s %c %lf %c %lf
%c",TmsgRMC.Protocol,TmsgRMC.UTC,&TmsgRMC.Status,&TmsgRMC.Lat,&TmsgRMC.NS,&
TmsgRMC.Lon,&TmsgRMC.EW);

        degmin2degdec(1);
        UTC2sec(1);

    //  if(Echo)
        //{
        //printf("*%s* *%s* *%c* *%f* *%c* *%f*
*%c*\n",TmsgRMC.Protocol,TmsgRMC.UTC,TmsgRMC.Status,TmsgRMC.Lat,TmsgRMC.NS,Tms
gRMC.Lon,TmsgRMC.EW);
        //printf("speed*%f* course*%f* date*%s* seconds
**%f**\n\n",TmsgRMC.Speed,TmsgRMC.Course,TmsgRMC.Date,TmsgRMC.SecStamp);
        //}
}

void parse_GGA(char TempMsgBuff[100])
{
int i=0;

while(TempMsgBuff[i]!=NULL)
    {
        if (TempMsgBuff[i]==',') TempMsgBuff[i]=32;
        i++;
     }

//if (Echo) printf("\nne varGGA  %s\n",TempMsgBuff);
sscanf(TempMsgBuff,"%5s %10s %lf %c %lf %c %c %c %lf
%lf",TmsgGGA.Protocol,TmsgGGA.UTC,&TmsgGGA.Lat,&TmsgGGA.NS,&TmsgGGA.Lon,&Tm
sgGGA.EW,&TmsgGGA.Status,&TmsgGGA.SatUsed,&TmsgGGA.HDOP,&TmsgGGA.Altitude);

degmin2degdec(2);
UTC2sec(2);

//if (Echo) printf("GGA parsed %s %s %lf %c %lf %c %c %c %lf %lf secs
%lf\n",TmsgGGA.Protocol,TmsgGGA.UTC,TmsgGGA.Lat,TmsgGGA.NS,TmsgGGA.Lon,TmsgGG
A.EW,TmsgGGA.Status,TmsgGGA.SatUsed,TmsgGGA.HDOP,TmsgGGA.Altitude,TmsgGGA.SecS
tamp);
}

void UTC2sec(int ca) //if ca=1 then RMC if ca=2 then GGA
```

```
{
static char firsttimeset=0;
double secs,hours,minutes,hunds,hmsec,hmsechund,hm;

        switch(ca){
        case 1:
        hmsechund=atof(TmsgRMC.UTC);
        hunds=modf(hmsechund,&hmsec);
        secs=100*modf(hmsec/100,&hm);
        minutes=100*modf(hm/100,&hours);
        TmsgRMC.SecStamp=(hours+TOffset)*3600+minutes*60+secs+hunds;
        break;
        case 2:
        hmsechund=atof(TmsgGGA.UTC);
        hunds=modf(hmsechund,&hmsec);
        secs=100*modf(hmsec/100,&hm);
        minutes=100*modf(hm/100,&hours);
        TmsgGGA.SecStamp=(hours+TOffset)*3600+minutes*60+secs+hunds;
        break;
        }

        TmsgRMC.Hours=(unsigned char) hours;
        TmsgRMC.Minutes=(unsigned char) minutes;
        TmsgRMC.Secondes=(unsigned char) secs;
        TmsgRMC.Hunds=(unsigned char) (hunds*100.0);

//set PC system time to GPS Local time
//if(TmsgRMC.Status=='A'&& firsttimeset==0)
if(firsttimeset==0)
        {
        zaman.ti_hour=TmsgRMC.Hours+3;
        zaman.ti_min=TmsgRMC.Minutes;
        zaman.ti_sec=TmsgRMC.Secondes;
        zaman.ti_hund=TmsgRMC.Hunds;
        settime(&zaman);
        firsttimeset=1;
        printf("\n\n\n\nok\n\n\n\n\a");
        }
}


void  degmin2degdec(int ca) //if ca=1 then RMC if ca=2 then GGA
{
double dec,deg;

        switch(ca){
        case 1:
        dec=100*modf(TmsgRMC.Lat/100,&deg);
        TmsgRMC.Lat=deg+dec/60.0;

        dec=100*modf(TmsgRMC.Lon/100,&deg);

                TmsgRMC.Lon=deg+dec/60.0;
        break;
        case 2:
                dec=100*modf(TmsgGGA.Lat/100,&deg);
        TmsgGGA.Lat=deg+dec/60.0;
```

```
            dec=100*modf(TmsgGGA.Lon/100,&deg);

                    TmsgGGA.Lon=deg+dec/60.0;
            break;
            }

}



void Serial_Init(void)
{
 char far *Ptr1,*Ptr2;
 int Seg1,Seg2,er1,er2,er3,er4;
outportb(PORT1 + 1 , 0);
outportb(PORT1 + 3 , 0x80);
outportb(PORT1 + 0 , 0x18);

// COM2
outportb(PORT2 + 1,  0);
outportb(PORT2 + 3,  0x80);
outportb(PORT2 + 0,  0x0C);

// COM1
outportb(PORT1 + 1 , 0x00);
outportb(PORT1 + 3 , 0x03);
outportb(PORT1 + 2 , 0xC7);
outportb(PORT1 + 4 , 0x0B);

// COM2
outportb(PORT2 + 1 , 0x00);
outportb(PORT2 + 3 , 0x03);
outportb(PORT2 + 2 , 0xC7);
outportb(PORT2 + 4 , 0x0B);

 SioPorts(4,4,0,PC_PORTS);
 SioUART(COM1,0x03f8);
 SioUART(COM2,0x02f8);
 SioIRQ(COM1,4);
 SioIRQ(COM2,3);

 /* setup 128 byte receive buffer */
 // COM1  GYRO
 Ptr1 = (char far *)RxBuffer1;
 Seg1 = FP_SEG(Ptr1) + ((FP_OFF(Ptr1)+15)>>4);
 SioRxBuf(COM1,Seg1,Size512);
 Ptr1 = (char far *)TxBuffer1;
 Seg1 = FP_SEG(Ptr1) + ((FP_OFF(Ptr1)+15)>>4);
 SioTxBuf(COM1,Seg1,Size128);
 SioParms(COM1,NoParity,OneStopBit,WordLength8);
 SioFIFO(COM1,1);
 SioReset(COM1,Baud9600);

 // COM2  GPS
 Ptr2 = (char far *)RxBuffer2;
 Seg2 = FP_SEG(Ptr2) + ((FP_OFF(Ptr2)+15)>>4);
 er1=SioRxBuf(COM2,Seg2,Size256);
 Ptr2 = (char far *)TxBuffer1;
```

```
Seg2 = FP_SEG(Ptr2) + ((FP_OFF(Ptr2)+15)>>4);
er2=SioTxBuf(COM2,Seg2,Size128);
er3=SioParms(COM2,NoParity,OneStopBit,WordLength8);
er4=SioReset(COM2,Baud4800);
SioFIFO(COM2,3);

if(er1<0||er2<0||er3<0||er4<0)
{
SioError(SioRxBuf(COM2,Seg2,Size128));
SioError(SioTxBuf(COM2,Seg2,Size128));
SioError(SioParms(COM2,NoParity,OneStopBit,WordLength8));
SioError( SioReset(COM2,Baud4800));
SioDone(COM2);
SioDone(COM1);
fclose(dosyaGPS);
fclose(dosyaGYRO);
exit(0);
}
SioRxClear(COM1);
SioRxClear(COM2);
printf("\n");
}
```

DRIVER.C

```
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define PORT 0x300
#define ANGLE 1

void init_Robot(void)
{
char flasher,state,motor;
state=inportb(PORT+6);

// If flasher is ON on powerON then RESET flasher
flasher=(state&16)&&1;
                                        if (flasher)
                                        {
                                        P9_Status&=223;
                                        outportb(PORT+9,P9_Status);
                                        P9_Status|=32;
                                        outportb(PORT+9,P9_Status);
                                        P9_Status&=223;
                                        outportb(PORT+9,P9_Status);
                                        }
// If Motor ON then  Reset Wheel Motor (Port 6 IN0)
motor=(state&1);
                                        if (motor)
                                        {
                                        P9_Status&=251;
                                        outportb(PORT+9,P9_Status);
                                        P9_Status|=4;
                                        outportb(PORT+9,P9_Status);
                                        P9_Status&=251;
                                        outportb(PORT+9,P9_Status);
                                        }
// Select ADC channel 0 ADC IN0 and Reset L & R steer Relays
outportb(PORT+10,0);
// Reset All 74HC4040 Encoders Counters
reset_Counters();
flasher_ONOFF();// reset ADC

}


void flasher_ONOFF(void)
{
P9_Status&=223;
outportb(PORT+9,P9_Status);
P9_Status|=32;
outportb(PORT+9,P9_Status);
P9_Status&=223;
outportb(PORT+9,P9_Status);
delay(1000);
P9_Status&=223;
outportb(PORT+9,P9_Status);
P9_Status|=32;
outportb(PORT+9,P9_Status);
```

```
P9_Status&=223;
outportb(PORT+9,P9_Status);

}



void reset_Counters(void)
{
P9_Status&=127;
outportb(PORT+9,P9_Status);
P9_Status|=128;
outportb(PORT+9,P9_Status);
P9_Status&=127;
outportb(PORT+9,P9_Status);

}


void get_Sensor_Data(void)
{
unsigned char a,b,c,bl,bh;
unsigned char steer;

                        get_TStamp();

a=inportb(PORT);
b=inportb(PORT+1);
c=inportb(PORT+2);
bl=b&15;
bh=b&240;
sensors.Pulse_Right=bl*256+a;
sensors.Tur_Right=bh*16+c;

a=inportb(PORT+3);
b=inportb(PORT+4);
c=inportb(PORT+5);
bl=b&0x0f;
bh=b&240;

sensors.Pulse_Left=bl*256+a;
sensors.Tur_Left=bh*16+c;

steer=inportb(PORT+8);
sensors.Steer_Angle=steer*0.019607840137*ANGLE;
}
```

NAVLIB.H

```
#define CTLZ 0x1a
#define STRING10 11
#define STRING7  8
#define STRING5  6
#define PORT 0x300
#define ANGLE 1
#define TOffset 3

#define PORT1 0x3F8
#define PORT2 0x2F8


unsigned char P9_Status=0;
char gpsMsgs[100]="",gpsMsg_RMC[100]="",gpsMsg_GGA[100]="",QBuff[16]="";

unsigned char i,dosyaflag=0,dosyacount[3];
FILE *dosyaGPS,*dosyaGYRO,*dosyaDENEY;

void getData_GYRO(void);
void parse_RMC(char TempMsgBuff[100]);
void parse_GGA(char TempMsgBuff[100]);
void  degmin2degdec(int);
void UTC2sec(int);
void Serial_Init(void);
void get_TStamp(void);

void init_Robot(void);
void flasher_ONOFF(void);
void reset_Counters(void);
void get_Sensor_Data(void);

static char RxBuffer1[256+16];
static char TxBuffer1[128+16];
static char RxBuffer2[256+16];
static char TxBuffer2[128+16];

char dosya_adi_GPS[8],dosya_adi_GYRO[8],*uzanti=".txt";

struct  time zaman;
struct Stamp
        {
                unsigned char Hours;
                unsigned char Minutes;
                unsigned char seconds;
                unsigned char Hunds;
                double TStampInSec;
        };
struct RMC
        {
                char Protocol[STRING5];
                char UTC[STRING10];
                char Hours;
                char Minutes;
                char Secondes;
                unsigned char Hunds;
                double SecStamp;
```

```
		char Status;
		double Lat;
		char NS;
		double Lon;
		char EW;
		char ChSpeed[STRING10];
		double Speed;
		char ChCourse[STRING10];
		double Course;
		char Date[STRING7];
	};

	struct GGA
	{
		char Protocol[STRING5];
		char UTC[STRING10];
		char Hours;
		char Minutes;
		double Secondes;
		double SecStamp;
		double Lat;
		char NS;
		double Lon;
		char EW;
		char Status;
		char SatUsed;
		double HDOP;
		double Altitude;
	};
struct SENSORS
	{
		unsigned int Pulse_Right;
		unsigned int Pulse_Left;
		unsigned int Tur_Right;
		unsigned int Tur_Left;
		double  Steer_Angle;
		double GyroRate;
		double GyroTemperature;
	};

union Rate
{
	struct bitfields
	{
	char Part0 :7;
	char Part1 :7;
	char Part2 :2;
	} Bits;
	int GyroRate;
} KVH;



struct RMC TmsgRMC;
struct GGA TmsgGGA;
struct Stamp TStamp;
struct SENSORS sensors;
```