TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KONUMLANDIRMA VE HARİTALAMA ALGORİTMALARINDA MAKİNE ÖĞRENMESİ VE PARÇACIK FİLTRELERİNİN KULLANILMASI

Halit ÖRENBAŞ

DOKTORA TEZİ
Kontrol ve Otomasyon Mühendisliği Anabilim Dalı
Kontrol ve Otomasyon Mühendisliği Programı

Danışman
Dr. Öğr. Üyesi Muharrem MERCİMEK

Temmuz, 2019

TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KONUMLANDIRMA VE HARİTALAMA ALGORİTMALARINDA MAKİNE ÖĞRENMESİ VE PARÇACIK FİLTRELERİNİN KULLANILMASI

Halit ÖRENBAŞ tarafından hazırlanan tez çalışması 16.07.2019 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Kontrol ve Otomasyon Mühendisliği Anabilim Dalı Kontrol ve Otomasyon Mühendisliği Programı **DOKTORA TEZİ** olarak kabul edilmiştir.

Dr. Öğr. Üyesi Muharrem MERCİMEK
Yıldız Teknik Üniversitesi
Danışman

Jüri Üyeleri

Dr. Öğr. Üyesi Muharrem MERCİMEK, Danışman
Yıldız Teknik Üniversitesi _____

Prof. Dr. Halit PASTACI, Üye
Haliç Üniversitesi _____

Prof. Dr. Şeref Naci ENGİN, Üye
Yıldız Teknik Üniversitesi _____

Dr. Öğr. Üyesi Claudia Fernanda YAŞAR, Üye
Yıldız Teknik Üniversitesi _____

Dr. Öğr. Üyesi Tarık Veli MUMCU, Üye
İstanbul Üniversitesi - Cerrahpaşa _____

Danışmanım Dr. Öğr. Üyesi Muharrem MERCİMEK sorumluluğunda tarafımca hazırlanan Konumlandırma ve Haritalama Algoritmalarında Makine Öğrenmesi ve Parçacık Filtrelerinin Kullanılması başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Halit ÖRENBAŞ

İmza

Aileme..

TEŞEKKÜR

Akademik hayatımda karşılaştığım problemlerin çözümü konusunda yardım ve desteğini hiç bir zaman esirgemeyen danışman hocam Dr. Öğr. Üyesi Muharrem MERCİMEK'e, desteklerini her zaman hissettiğim ve elde ettiğim başarılarda paylarının çok büyük olduğuna inandığım eşime, aileme ve benim gözümde iş ortamını daha keyifli bir hale getirdikleri için arkadaşlarım Arş. Gör. Ali Nur ÖZ, Arş. Gör. Bilal EROL, Arş. Gör Esra KAYA AYANA ve Arş. Gör. Gülsüm GEZER'e sonsuz teşekkürler..

Halit ÖRENBAŞ

İÇİNDEKİLER

SİMGE LİSTESİ

λ	Adım parametresi
w	Ağırlık
z	Ağırlık toplamı
φ	Aktivasyon fonksiyonu
$\widetilde{p}(.\|.)$	Bayes kuralı
F_n	Belirsizliğin Jacobian matrisi
n	Belirsizlik
$F(n)$	Belirsizlik matrisi
x	Durum vektörü
F_x	Durum vektörünün Jacobian matrisi
F	Geçiş matrisi
A_p	Genlik
x_k^p	Hareket modeli
p	Hedef
$\phi(.,.)$	Homotopi fonksiyonu
K	Kalman kazancı
N_c	Küme sayısı
u	Kontrol işareti
F_u	Kontrol işaretinin Jacobian matrisi
$F(u)$	Kontrol matrisi
P,Q,R	Kovaryans matrisi
d_0	Maksimum ölçülebilir genlik
$\overline{x}_k$	Mevcut tahmin

θ_j	Model parametresi
η	Öğrenme oranı
y_i	Ölçüm
v	Ölçüm gürültüsü
H	Ölçüm matrisi
b	Önyargı vektörü
R	Robot
S	Sensör
z_k	Sensör verisi
m	Veri kümesi örnek sayısı
L_i	Yer işaretleri
κ	Yol kaybı üssü

KISALTMA LİSTESİ

APF	Auxiliary Particle Filter
ASGG	Alınan Sinyal Gücü Göstergesi
CEDH	Clustered Exact Daum-Huang
CPU	Central Processing Unit
ÇKP	Çok Katmanlı Perseptron
DH	Daum-Huang
DLB	Doğrultulmuş Lineer Birim
DSA	Derin Sinir Ağı
EDH	Exact Daum-Huang
EKF	Extended Kalman Filter
EMB	Eşik Mantık Birimi
GPS	Global Positioning System
GPU	Graphic Processing Unit
İHA	İnsansız Hava Aracı
İKA	İnsansız Kara Aracı
KF	Kalman Filtresi
LEDH	Localizaed Exact Daum-Huang
MC	Monte Carlo
OKE	Ortalama Kare Hata
OMAT	Optimal mass transfer
PDF	Probability Density Function
PF	Parçacık Filtresi
RNN	Recurrent Neural Network

ROS	Robot Operating System
RTAB-Map	Real-Time Appearance-Based Mapping
SLAM	Simultaneous Localization and Mapping
SMC	Sequential Monte Carlo
SVM	Support Vector Machine
Tanh	Hyperbolic Tangent
TN	Toplu Normalleştirme
UKF	Unscented Kalman Filter
VW	Visual Words
YSA	Yapay Sinir Ağı

ŞEKİL LİSTESİ

TABLO LİSTESİ

ÖZET

Konumlandırma ve Haritalama Algoritmalarında Makine Öğrenmesi ve Parçacık Filtrelerinin Kullanılması

Halit ÖRENBAŞ

Kontrol ve Otomasyon Mühendisliği Anabilim Dalı
Doktora Tezi

Danışman: Dr. Öğr. Üyesi Muharrem MERCİMEK

Zorlu ortam koşullarına dayanıklı olabilen, gelişmiş ve çeşitli algılayıcılar ile donatılmış robotlar, insanlara oranla daha hızlı karar verebilmeleri ve insanların ulaşamayacağı yerlere ulaşabilmeleri ve benzeri avantajlardan dolayı hem sivil hem de askeri görevlerin temelini oluşturmaktadırlar.

Insansız yapılmak istenen herhangi bir görevin temelini, bilinmeyen çevrenin haritasını çıkarmak ve çıkarılan haritada robotun kendi konumunun tespiti oluşturmaktadır. Kalman filtresi, genişletilmiş Kalman filtresi ve parçacık filtresi haritalama ve konumlandırma algoritmalarında en yaygın olarak kullanılan tahmin algoritmalarıdırlar.

Parçacık filtreleri, herhangi bir durum-uzay modeline uygulanabilen, geleneksel Kalman filtreleme yöntemlerini genelleyen ardışık Monte Carlo yöntemleridirler. Parçacık filtresinin performansı, durum boyutu arttıkça önemli ölçüde azaldığından, bu problemi çözmek adına Exact Daum-Huang filtresi tanıtılmıştır.

EDH filtresinin, algoritmanın içerisinde paralel olarak yürütülen genişletilmiş Kalman filtresine fazla bağımlılığı bazı senaryolarda yetersiz sonuçlar vermesine neden olmaktadır. Localized Exact Daum-Huang filtresi, hesaplama maliyetinden ödün vererek bu problemi çözmüştür.

Tez çalışmasında önerilecek olan sistem, LEDH filtresinin performansından ödün vermeden, makine öğrenmesi teknikleri kullanarak parçacıkların hata oranlarına göre

kümelenmesini sağlayarak performans iyileştirmeleri sunmaktadır. Tez kapsamında kullanılan çok boyutlu konumlandırma problemi ile önerilen sistemin performansı kanıtlanmıştır.

Anahtar Kelimeler: Konumlandırma, haritalama, makine öğrenmesi, derin öğrenme, parçacık filtresi

ABSTRACT

Localization and Mapping Algorithms Using Machine Learning and Particle Filters

Halit ÖRENBAŞ

Department of Control and Automation Engineering
Doctor of Philosophy Thesis

Advisor: Asst. Prof. Dr. Muharrem MERCİMEK

Robots equipped with advanced and various sensors that can withstand harsh environmental conditions can make decisions faster and reach to places that people cannot reach and because of similar advantages, they form the basis of both civilian and military missions.

The basis of any task to be made without an unmanned is to map the unknown environment and to localize its own position. Kalman filter, extended Kalman filter and particle filter are the most widely used estimation algorithm in the mapping and localization algorithms.

Particle filters are consecutive Monte Carlo methods that can be applied to any state-space model, which generalize the traditional Kalman filtering methods. The Exact Daum-Huang filter was introduced because the performance of the particle filter decreased significantly as the size of the dimension increased.

The excessive dependence of the EDH filter on the extended Kalman filter carried out in parallel within the algorithm results in insufficient in some scenarios. Localized Exact Daum-Huang filter solved this problem by compromising the cost of calculation.

The system proposed in this thesis provides a significant improvement in performance without compromising the performance of the LEDH filter by using machine learning techniques to cluster particles according to their error rates. The performance of the proposed system in this thesis has been proven with the multi-dimensional localization problem.

Keywords: Localization, mapping, machine learning, deep learning, particle filter

1
Giriş

1.1 Literatür Özeti

Gelişmiş ve çeşitli algılayıcılara sahip olabilen robotlar insanlara oranla daha hızlı karar verebilmekte, zorlu ortam koşullarına daha dayanıklı olabilmekte ve insanların ulaşamayacağı yerlere ulaşarak zorlu görevleri yerine getirebilmektedirler. Bu avantajlardan dolayı hem askeri hem de sivil görevlerde giderek artan bir oranda tercih edilmektedirler [1].

Başarılarını çevrenin tanınması, hedefin tespit edilmesi ve takibi gibi özel uygulamalarda kanıtlayan insansız hava araçları ve insanız kara araçları önceden insanlarla yapılan çoğu zorlu uygulamada giderek daha fazla tercih edilmektedirler. Bundan dolayı İHA ve İKA'lar savunma sanayisinde, güvenlik problemlerinin çözümünde, doğal afetlerde kurtarma görevlerinde ve ülke sınırlarında sınır ihlallerinin incelenmesi gibi özel konularda önemli yerlere sahiptirler.

Doğal afetleri örnek olarak gösterdiğimizde, bu senaryoda görev alana İHA veya İKA'nın sahip olduğu ön yüklü harita, afet zamanında değişen çevre şartlarında ve kısıtlı imkanlarda yetersiz kalabilmektedir. Bu şartlar altında performansı yüksek, düşük hata oranına sahip haritalama ve konumlandırma algoritmaları büyük önem arz etmektedir.

Doğal afet senaryosu üzerinden verilen örnek, sadece o şartlar altında değil otonom ilerleyen bir araba, ilk yardım, gözetleme, hedef tespiti ve takibi, lojistik, petrol ve doğal gaz boru hatlarının denetlenmesi, ekinlerin ilaçlanması, film ve fotoğraf sektörü ve daha bir çok görevlerde kullanılan robotlar ve makineler için geçerlidir. Kısacası insansız yapılmak istenen herhangi bir görevin temelini, bilinmeyen çevrenin haritasını çıkarmak ve çıkarılan haritada makinenin kendi konumunu tespit etmesi oluşturmaktadır [2].

Otonom çalışan bir makine, ya ön-yüklü bir harita ile ya da üzerinde bulunduruğu algılayıcıların yardımıyla çevresinde bulunan hareketli veya hareketsiz nesneleri veya

canlıları tespit ederek doldurması gereken boş bir harita ile göreve başlar. Verilen hedefe ulaşabilmek adına sürekli kendi konumunu güncel tutmak zorundadır.

Makinelerin kullandığı algılayıcılar, teknolojinin ilerlemesi ile giderek artan doğruluk oranlarına sahip olsalar dahi, gürültülerden ve algılayıcı ölçümlerinden gelen hatalar haritalama ve konumlandırma algoritmalarında büyük problemlere sebebiyet vermektedir. Bunun önlemek adına filtreler ve olasılıksal yöntemler geliştirmiş ve geliştirilmeye devam etmektedir. Bu yöntemlerden en yaygınları, doğrusal tahmin problemleri için Kalman Filtresi [3], doğrusal olmayan tahmin problemleri için düzenlenmiş versiyonu olan Genişletilmiş Kalman Filtresi [4] ve parçacıkların ağırlıklandırılmış toplamı ile değişken kestirimi yapan Parçacık Filtresi'dir [5]. Bu algoritmalar filtrelemenin yanı sıra, haritalama ve konumlandırma algoritmalarının merkezinde çalışan tahmin algoritmalarıdırlar.

Parçacık filtreleri, herhangi bir durum-uzay modeline uygulanabilen, geleneksel Kalman filtreleme yöntemlerini genelleyen ardışık Monte Carlo yöntemleridirler. Parçacık filtreleri, parçacıkları önceki dağılımdan çekerek en son ölçüm olasılığını kullanırlar ve her bir parçacığın ağırlığını sürekli güncellerler [5]. Durum boyutu yüksek olduğu senaryolarda, önceki dağılımdan çekilen parçacıkların çoğunluğu, ihmal edilebilir ağırlıklara yol açan çok düşük olasılıksal bölgelere dönüşecektir. Sonraki dağılımın Monte Carlo yaklaşımı, birkaç parçacık tarafından belirlendiğinden dolayı bu ağırlık dejenerasyonu, dağılımın zayıf bir şekilde temsil edilmesine yol açar [6].

Optimal teklif dağılımı yaklaşımı, önem ağırlıklarının varyansını en aza indirir [7]. Auxiliary Particle Filter, yeni ölçümlerden gelen bilgileri hesaba katması sayesinde, parçacıkların daha etkili bir şekilde örneklenmesini sağlar [8]. Rao-Blackwellised parçacık filtresi, Monte Carlo tahminlerinin varyansını azaltan bir çözüm sunar [9].

Yüksek boyutlu sistemlerde doğrusal olmayan filtreleme, başta çoklu hedef takibi olmak üzere birçok uygulamada ortaya çıkan önemli bir problemdir. Bahsi geçen metotlar, yüksek boyutlu problemlerde yetersiz performans sergilemektedirler [10]. Parçacık filtresinin performansı, durum boyutu arttıkça önemli ölçüde azalmaktadır [11].

Parçacık filtrelerinin yüksek boyutlu sistemlerdeki filtreleme probleminin çözümü için çeşitli yöntemler sunulmuştur. Eş ağırlıklar parçacık filtresi (Equivalent weights particle filter) [12], yüksek boyutlarda ümit verici bir performans göstermesine rağmen, geleneksel parçacık filtresinin istatistiksel tutarlılığından ödün vermektedir. Ümit verici başka yaklaşımlar, durum uzayını çarpanlarına ayırma yoluyla önceki değerlerin uygun bir faktörizasyonunu belirlemeye dayanan [13] ve

[14] yöntemleridir. Ancak bu yöntemlerin de uygulanabilirlikleri kısıtlıdır [15].

Daum ve Huang doğrusal olmayan filtreleme problemine alternatif bir yaklaşım başlatmışlardır [16–18]. DH filtrelerinde, her zaman aralığı için, önsel ve sonsal dağılımlar arasında bir homotopi fonksiyonu uygular. Bu fonksiyon, kısmı diferansiyel denklemin çözümü olarak tarif edilen bir parçacık akışını ifade eder. Parçacık akışı, bir dizi parçacığın, önsel dağılımın büyük değerlere sahip olduğu bölgelere kademeli olarak göç etmesini sağlar [19].

DH filtreleri farklı doğrusal olmayan filtreleme problemleri için etkileyici performans sergilemesine rağmen, EDH filtresi, algoritmanın içerisinde paralel olarak yürütülen EKF/UKF filtresine fazla bağımlılık göstermektedir. Bu bağımlılık EDH'nin bazı senaryolarda yetersiz sonuçlar vermesine neden olmaktadır [20].

[19]'da, sistemin ve gözlem gürültülerinin Gauss bir dağılım çizdiği, sistem haritasının doğrusal olduğu fakat elde edilen sonuçların doğrusal olmadığı durumlar üzerine bir çözüm geliştirilmiş ve LEDH filtresi tanıtılmıştır. Yapılan iyileştirme, sistemin doğrusallaştırılması ve tüm parçacıkların bir seferde göç parametrelerinin hesaplanmasından ziyade, her bir parçacık için parametrelerin tek tek hesaplanmasını önermesidir. Bu iyileştirme karmaşık problemlerde performans iyileştirmesi sağlamasına rağmen algoritmanın hesaplama maliyetini önemli ölçüde arttırmaktadır.

1.2 Tezin Amacı

Tez çalışmasının ana amacı yüksek durum boyutlarına sahip sistemlerin haritalama ve konumlandırma algoritmalarının performansını arttırmak adına, haritalama ve konumlandırma algoritmalarının merkezi tahmin uygulaması olarak kullanabilecek yeni bir parçacık filtresi geliştirmektir.

Makine öğrenmesi tekniklerinden K-Ortalamalar algoritması yardımıyla güçlendirilen bu yeni parçacık filtresi Clustered Exact Daum-Huang Parçacık filtresidir. CEDH filtresi, performansını yüksek durum boyutuna sahip bir sistemin konumlandırma probleminin çözümü üzerinden kanıtlamıştır.

Tezin diğer bir amacı ise, CEDH filtresinin tanıtılmasının yanı sıra, verilerin ulaşılabilirliğinin ve işlemci güçlerinin her geçen gün artmasıyla uygulama alanları gittikçe artan derin öğrenme ağlarının konumlandırma problemlerine yönelik çözümleri hakkında araştırmalar yapmak ve bu araştırmalar neticesinde bir konumlandırma probleminin çözümünü sağlamaktır.

1.3 Orjinal Katkı

CEDH filtresi, LEDH filtresinin performansından ödün vermeden, hatta hata payı yüksek parçacıkların genel sistem üzerindeki etkisini kümeleme ve eleme yöntemiyle azaltılmasını sağlayarak, hesaplama maliyetini düşürmektedir. Parçacıkların, hata değerleri üzerinden K-Ortalama algoritması [21] kullanılarak kümelenmesi sağlanmaktadır.

CEDH filtresinde, parçacıklar için ne EDH filtresinde olduğu gibi tek bir parametre hesaplaması yapılmakta ne de LEDH filtresinde olduğu gibi tek tek parametreler hesaplanmaktadır. Yaklaşık hata değerlerine sahip parçacıklar için ortak göç parametreleri hesaplanarak hesaplama maliyeti önemli ölçüde düşürülmektedir.

2
Eş Zamanlı Konumlandırma ve Haritalama

Eş Zamanlı Konumlandırma ve Haritalama, eşzamanlı olarak makinenin kendini konumlandırması sırasında, algılayıcıları yardımıyla çevredeki dünyanın haritasının çıkarmasıdır. SLAM probleminin olasılıksal problemi için ufuk açıcı çözüm Smith ve Cheeseman tarafından yayınlanmıştır [22]. SLAM probleminde, EKF sıklıkla merkezi tahmin edici olarak kullanılmaktadır [23].

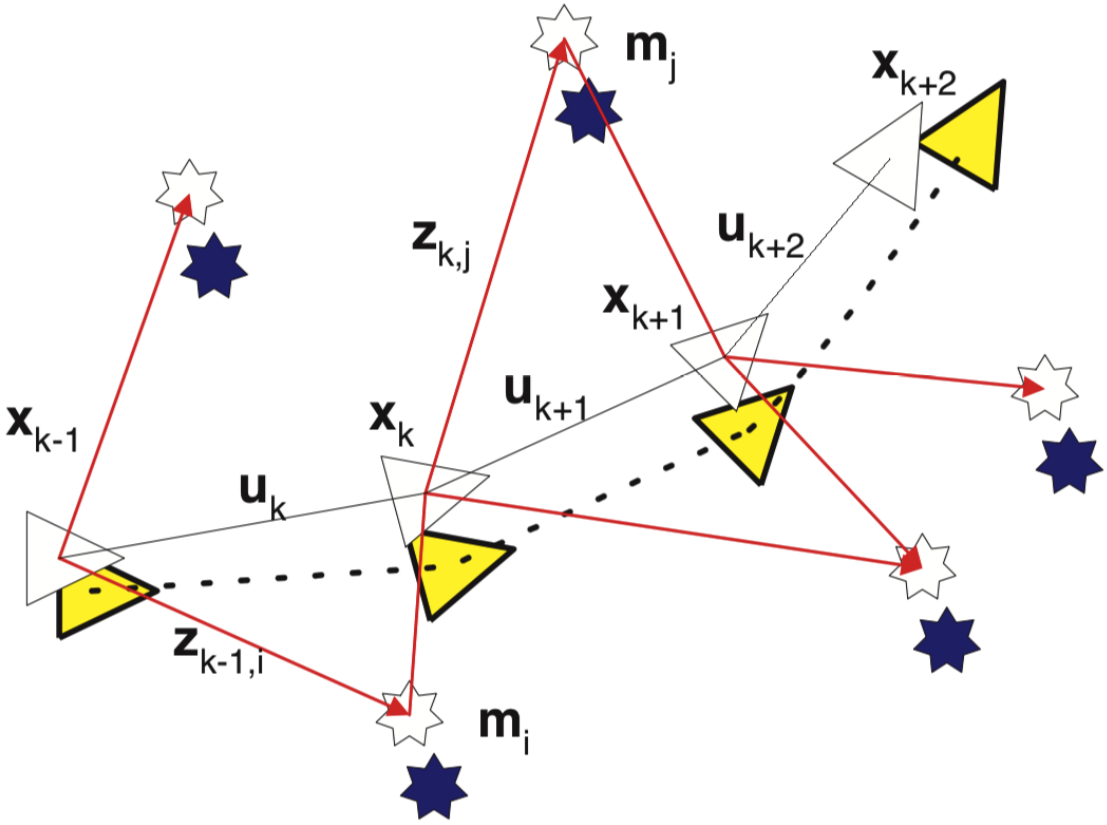
SLAM temelinde, günlük olarak karşılaştığımız en temel problemlerden biridir. Bilinmeyen bir ortama girdiğimizde, gözlemler, çevremizi tanımaya başlar ve hareket ederken kendimizi konumlandırmaya çalışırız. Bunu farkında olmadan her yeni ortama girdiğimizde yapmaktayız. Bu davranış tarzının analizi, SLAM algoritmalarının oluşmasına ışık tutmuştur.

SLAM iki temel sorunun cevabını arar: "ben neredeyim?" ve "çevremde ne var?". Makine, algılayıcılarından gelen ölçümler doğrultusunda, bilinmeyen bir çevrede ilerlerken olasılıksal yöntemler yardımıyla hem kendi konumunu belirler hem de haritalama yapar [24, 25].

SLAM uygulamasında, çevresi hakkında bilgi toplayan en az bir sensöre sahip bir ajan bulunmak zorundadır. Eğer bu sensör kamera, lazer tarayıcı veya bir sonar ise bunlar eksteroseptif sensörler, jeometre, ivmeölçer veya tekerlek kodlayıcısı gibi kendi hareketini ölçen aletler ise proprioseptif sensörler olarak adlandırılır.

SLAM, sürekli yenilenen 3 temel adımdan oluşur [26]:

1. Ajan hareket eder: Bu adım robotun konumu hakkındaki belirsizliği arttırır. Bu adımın çözümü için matematiksel bir modele ihtiyaç vardır: hareket modeli
2. Ajan çevresindeki ilginç özellikleri keşfeder: Bu adımda çevrede bulunan yer işaretleri ile birlikte işbirliği yapılmaktadır. Bir önceki adımda robotun konumu hakkında bir belirsizlik olmasına ilaveten bu adımda sensör ölçümlerinden gelen hata sebebiyle yer işaretleri de belirsizlik içermektedir. Bu adımı çözecek



Şekil 2.1 SLAM genel gösterim [24]

matematiksel modele ters gözlem modeli diyoruz.

3. Ajan daha önce haritaladığı yer işaretlerini gözlemler: Bu adımda ajan hem kendi konumunu hem de çevredeki yer işaretlerinin konumunu düzeltir, yani belirsizlik azalır. Bu adımı çözecek matematiksel modele doğrudan gözlem modeli denir.

Şekil 2.1'de ajan üçgen ile, yer işaretleri yıldız ile gösterilmektedir. Mavi renk, yer işaretlerinin tahmini konumlarını gösterirken, sarı renk ajanın tahmini konumlarını göstermektedir. Beyaz renk ile de ajanın ve yer işaretlerinin gerçek konumları gösterilmektedir.

2.1 Hareket ve Gözlem Modelleri

2.1.1 Hareket Modeli

Robot R için, uygulanan u kontrol işareti ve n belirsizliği altında, robotun durum uzayı

$$R \leftarrow f(R, u, n) \tag{2.1}$$

olarak güncellenir. Burada kontrol işareti, bilgisayardan robotun tekerleğine gelen sinyal olabileceği gibi, sensörden alınan veriyi de temsil edebilir. Ayrıca herhangi bir kontrol işaretinin olmadığı senaryolar da söz konusudur.

2.1.2 Doğrudan Gözlem Modeli

L_i yer işaretleri ve S sensörleri ile elde edilen ölçümleri y_i' dir. Robot R, sensör S'den elde edilen ölçümlerle daha önce haritaladığı yer işaretlerini bu adımda tekrar gözlemler. O halde elde edilen ölçümler,

$$y_i = h(R, S, L_i) \tag{2.2}$$

denklemi ile modellenir. Bu aşama daha önce belirtildiği gibi belirsizliği azaltır.

2.1.3 Ters Gözlem Modeli

Bu aşamada robot, yeni gözlemler yapmaktadır. Robotun konumu hakkında bir belirsizlik olmasına ilaveten bu adımda sensör ölçümlerinden gelen hatalar sebebiyle yer işaretleri de belirsizliği arttırmaktadır.

$$L_i = g(R, S, y_i) \tag{2.3}$$

Robot yeni keşfedilen yer işaretlerini denklem 2.3'e göre hesaplar.

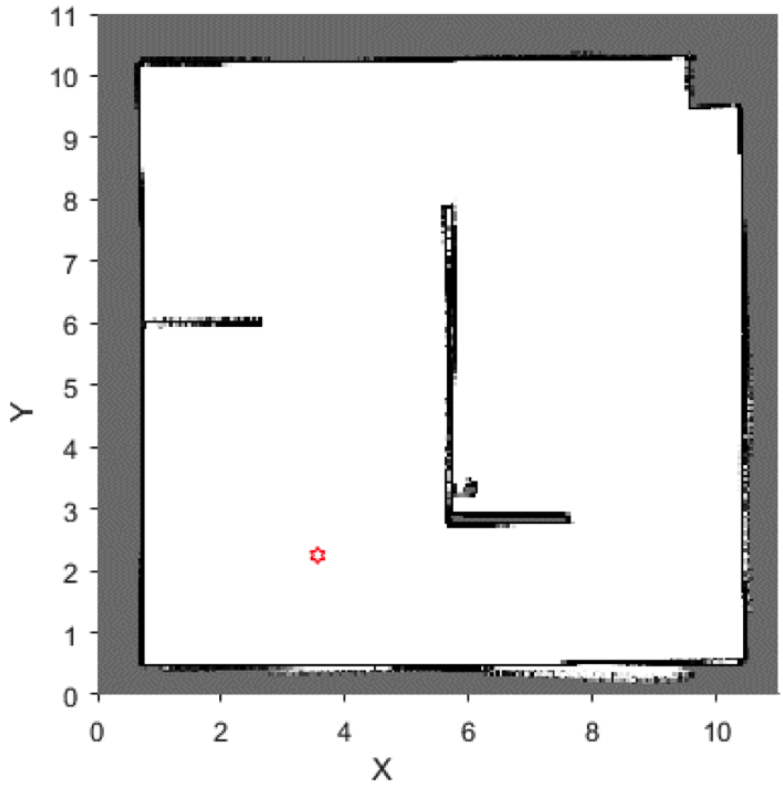
2.2 Harita Modelleri

Yer işaretçisi tabanlı harita, doluluk ızgaraları ve topolojik haritalar SLAM algoritmalarında yaygın olarak kullanılan harita modelleridir.

2.2.1 Doluluk Izgaraları Tabanlı Haritalar

Doluluk ızgaraları tabanlı haritalarda çevre, her birinin dolu olması olasılığının belirtildiği hücrelerden oluşan ayrık ızgaralar ile temsil edilir [27]. Bu yöntem ile oluşturulan bir harita her bir koordinatın 0-1 arasında bir olasılığıa sahip olması mantığında çalışmaktadır.

Şekil 2.2' de doluluk ızgaraları tabanlı haritalar kullanılan yapılan bir SLAM çalışmasından örnek gösterilmektedir.



Şekil 2.2 Doluluk ızgaraları tabanlı haritalama örneği

Izgaraların çözünürlüğü çevrenin haritasının çıkarılmasında belirli bir etkendir. Izgaraların çözünürlükleri azaltılıp arttırılabilmektedir. Çözünürlüğün arttırıldıkça, işlem yükü ve bellek ihtiyacı artar fakat ortalama hata azalır. Doluluk ızgaraları tabanlı haritaların en büyük dezavantajı fazla bellek gereksinimine ihtiyaç duymalarıdır.

2.2.2 Yer İşaretçisi Tabanlı Haritalar

Yer işaretçisi tabanlı haritalar, robotların konum bilgileri ile çevrede bulunan yer işaretlerinin pozisyon bilgilerinin metrik olarak saklanmasından oluşur. Eğer yer işaretçisi tabanlı haritalar iki boyutta ise konum bilgileri Kartezyen koordinat sisteminde saklanır [28].

2.2.3 Topolojik Haritalar

Topolojik haritalar, düğümler ve kenarlar olmak üzere iki alt başlıktan oluşur. Kenarlar düğümler arasındaki geçişleri temsil ederken, düğümler belirli bir bölgeyi temsil ederler.

2.3 Kalman Filtresi Tabanlı Haritalama ve Konumlandırma

Kalman filtresi tabanlı SLAM üç ana varsayıma dayanmaktadır.

1. Hareket modeli doğrusal bir modeldir ve gürültü Gauss dağılımı sergilemektedir.
2. Gözlem modeli doğrusaldır.
3. Başlangıç belirsizliği Gauss dağılımı sergilemektedir.

Kalman filtresi uygulanacak sistem,

$$X = F(x)x + F(u)u + F(n)n \tag{2.4}$$

olarak verilirse, ölçüm fonksiyonu,

$$y = Hx + v \tag{2.5}$$

olarak gösterilir. Burada $F(x)$ geçiş matrisi, $F(u)$ kontrol matrisi, $F(n)$ belirsizlik matrisi , H ölçüm matrisi, P, Q ve R kovaryans matrisi, x durum vektörü, u kontrol vektörü, n belirsizlik vektörü ve v ölçüm gürültüsüdür.

Verilen sistem için öncelikle $F(x)$, $F(u)$, $F(n)$ ve H tanımlanmalıdır. Başlangıç koşulları belirlendikten sonra ilk aşama tahmin aşamasıdır. Bu aşamada x, P, Q ve R tahmin edilmeye çalışılır. Geçici döngü içerisinde her adım için, x'in ortalama değeri ve gelen u kontrol işareti için P,

$$x_i = F(x)x_{i-1} + F(u) * u \tag{2.6}$$

$$P_i = F(x)P_{i-1}F(x)^T + F(n)QF(n)^T \tag{2.7}$$

formülleriyle hesaplanır. İkinci aşama düzeltme aşamasıdır. Ölçüm değeri y göz önünde bulundurularak sistemin hatası ve Kalman kazancı hesaplanır.

Bu aşamayı kendi içerisinde üç kısma ayırabiliriz.

1. Beklenti:

$$e = Hx \tag{2.8}$$

$$E = HPH^T \tag{2.9}$$

2. Yenilik:

$$z = y - e \tag{2.10}$$

$$Z = R + E \tag{2.11}$$

3. Kalman kazancı:

$$K = PH^T Z^{-1} \tag{2.12}$$

Son aşamada durum vektörü x ve kovaryans matrisi P, Kalman kazancı ile tekrar hesaplanır [29].

$$x_i = x_{i-1} + Kz \tag{2.13}$$

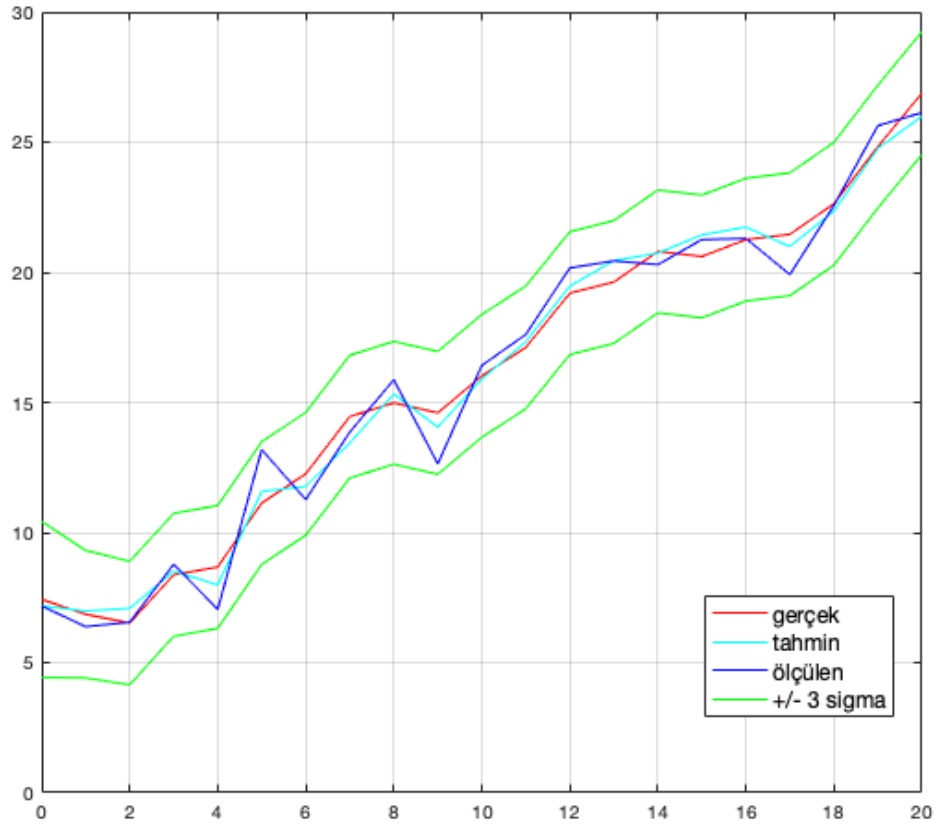
$$P_i = P_{i-1} - KHP \tag{2.14}$$

Şekil 2.3' de Kalman filtresi kullanılarak yapılan bir konumlandırma uygulaması için robotun gerçek konumu, ölçülen konumu ve tahmini konumu verilmiştir.

2.4 Genişletilmiş Kalman Filtresi Tabanlı Haritalama ve Konumlandırma

EKF'nin KF'den en büyük farkı sistemin doğrusallaştırılarak filtre işlemine tabi tutulmasıdır. EKF beş adımdan oluşur.

1. Durum tahmini: Bu aşamada kontrol işareti u ve robotun hareket modeli birlikte kullanılarak konum tahmini yapılır.
2. Gözlem yapılması: Gözlem aşaması yeni bilgilerin sisteme eklendiği aşamadır.



Şekil 2.3 Kalman filtresi ile konumlandırma

3. Ölçüm tahmini: Bir önceki aşamada yapılan gözlemler ile yer işaretçilerinin ve robotun konumu güncellenir.
4. Eşleştirme: Gözlem ve ölçüm tahmini arasındaki hata oranı hesaplanır.
5. Tahmin: Hesaplanan hata oranı doğrultusunda yeni tahmin yapılır.

EKF uygulanacak sistem,

$$X = f(x, u, n) \tag{2.15}$$

olarak verilirse, ölçüm fonksiyonu,

$$y = h(x) + v \tag{2.16}$$

olarak gösterilir. Burada $F(x)$ geçiş matrisi, H ölçüm matrisi, P, Q ve R kovaryans

matrisi, x durum vektörü, u kontrol vektörü, n belirsizlik vektörü ve v ölçüm gürültüsüdür.

Verilen sistem için öncelikle $F(x,u,n)$ ve $h(x)$ tanımlanmalıdır. Başlangıç koşulları belirlendikten sonra ilk aşama tahmin aşamasıdır. Bu aşamada x, P, Q ve R tahmin edilmeye çalışılır. Kalman filtresinden farklı olarak, bu aşamada Jacobian matrislerinin de hesaplanması gerekmektedir.

$$X = f(x,u,0) \tag{2.17}$$

$$P_i = F_x P_{i-1} F_x^T + F_n Q F_n^T \tag{2.18}$$

F_x durum vektörünün, F_u kontrol işaretinin ve F_n belirsizliğin Jacobian matrisini ifade etmektedir.

Sonraki aşama düzeltme aşamasıdır. Ölçüm değeri y göz önünde bulundurularak sistemin hatası ve Kalman kazancı hesaplanır.

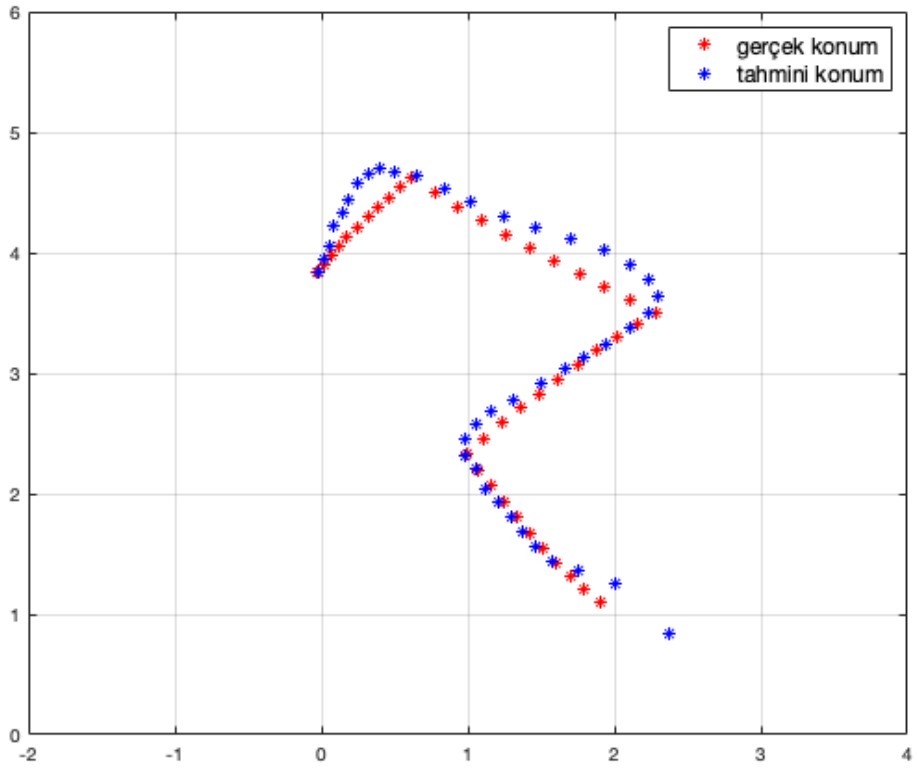
Şekil 2.4' de Genişletilmiş Kalman filtresi kullanılarak yapılan bir konumlandırma uygulaması için robotun gerçek konumu ve tahmini konumu verilmiştir.

2.5 Parçacık Filtresi Tabanlı Haritalama ve Konumlandırma

Parçacık filtreleri, herhangi bir durum uzay modeline rahatlıkla uygulanabilen, geleneksel Kalman filtreleme yöntemlerini genelleyen ve olasılıksal yoğunluklarının parçacık temsillerine dayanan ardışık Monte Carlo yöntemleridir. Parçacık filtresinde değişken birçok parçacığın kullanması ile temsil edilir ve her bir parçacığın temsili bir ağırlığı vardır. Bu ağırlık parçacığın olasılığını belirtmektedir. Parçacık filtresi, parçacıkları öncül dağılımdan çeker ve en son ölçüm olasılığını kullanarak her bir parçacığın ağırlığını günceller. Değişkenin nihai tahmini, parçacıkların ağırlıklandırılmış toplamı kullanılarak elde edilir [30].

Algoritma yenilemeli olarak tahmin ve güncelleme olarak iki adımdan oluşur.

1. Tahmin: Her bir parçacık belirli bir modele göre her hareket sonrasında değiştirilir. Rastgele bir gürültü değeri eklenerek, eklenen gürültünün değişken üzerindeki etkisinin benzetimi yapılır.
2. Güncelleme: Her bir parçacığın ağırlığı, elde edilen son veriler ışığında tekrar



Şekil 2.4 Kalman filtresi ile konumlandırma

değerlendirilir. Hata oranı az olan, yani gerçek olma ihtimali kuvvetli olan parçacıklar, daha fazla ağırlığa sahip olurlar.

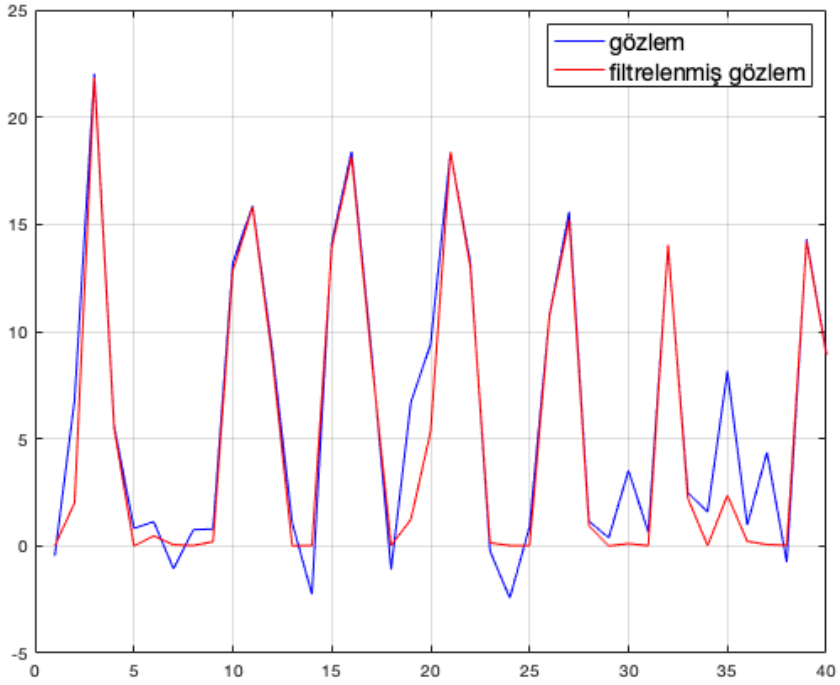
Örnek olarak, filtrelenecek fonksiyonun,

$$x_k = x_{k-1}/2 + 25x_{k-1}/(1 + x_{k-1}^2) + 8\cos 1.2k + v_{k-1} \tag{2.19}$$

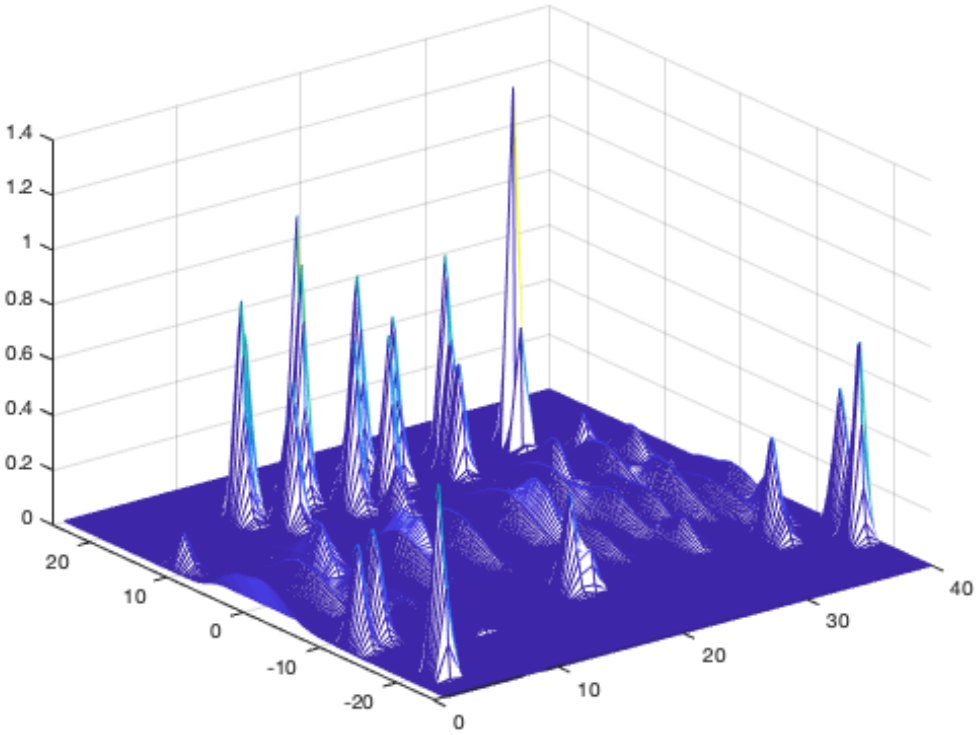
ve ölçüm fonksiyonunun,

$$y_k = x_{k-1}^2/20 + n_k \tag{2.20}$$

olduğu bir sistem ele alalım. v_{k-1} ve n_k burada Gauss dağılımı gösteren gürültüler olsun. Giriş fonksiyonunun filtrelenmiş çıktısı Şekil 2.5' de ve durum yoğunluklarının değişimi Şekil 2.6' da verilmiştir. Bu iki grafik incelendiğinde, doğrusal olmayan sistem üzerinden yapılan filtre tahminlerinin başarılı olduğu gözlenmektedir.



Şekil 2.5 Gözlem ve filtrelenmiş gözlem



Şekil 2.6 Durum yoğunluklarının evrimi

2.5.1 Monte Carlo Yöntemiyle Konumlandırma

Parçacık filtre lokalizasyonu olarak da bilinen Monte Carlo lokalizasyonu, parçacık filtresini kullanan bir algoritmadır [31]. Algoritma bilinen bir çevrede uygulanır. Robotun pozisyonunu ve yönelimini, robotun hareketlerini ve çevreyi algılayarak tahmin etmeye çalışır. Monte Carlo lokalizasyonu, olası konumların dağılımlarını göstermek için parçacık filtresini kullanır. Her bir parçacık robotun konumunu olasılıksal olarak gösterir.

N adet parçacığın olduğunu varsayarsak, örneklere karşılık gelen ağırlık seti

$$X_{k-1} = x_{k-1}^1, x_{k-1}^2, x_{k-1}^3, ..., x_{k-1}^N \tag{2.21}$$

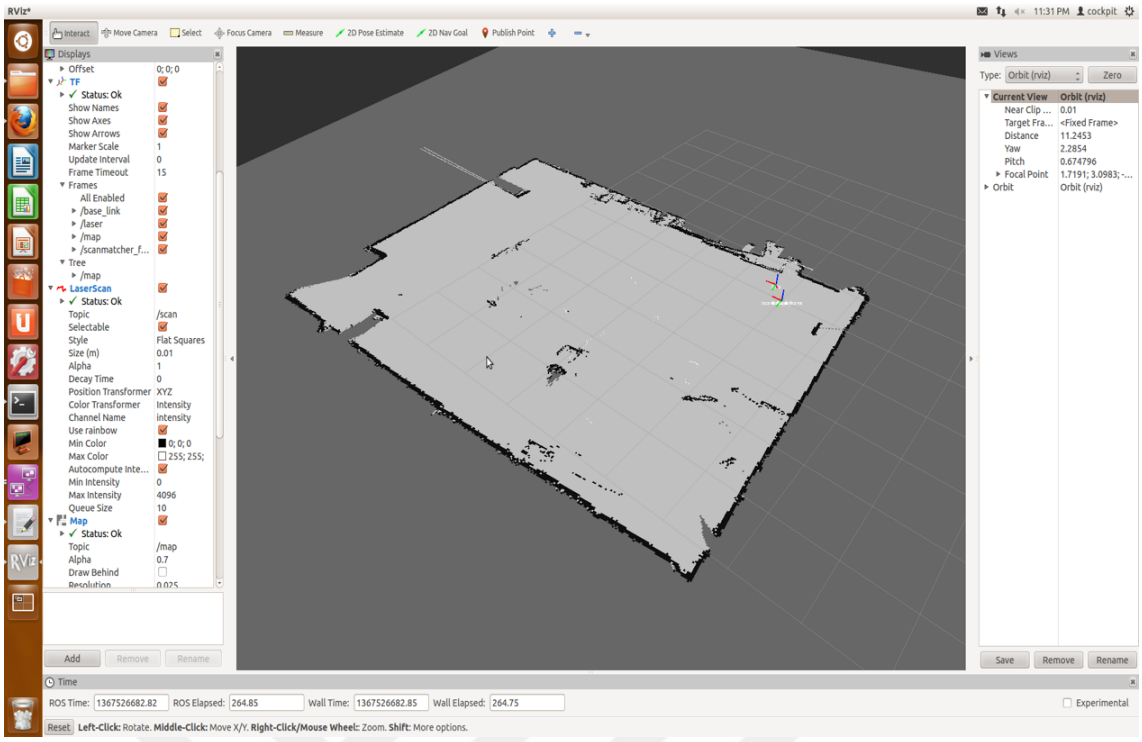
olarak gösterilir. Kontrol işaretini u_k, sensörden alınan verileri z_k ve algoritmanın çıktısını X_k ile gösterelim.

Algoritma 2.1 Monte Carlo konumlandırma algoritması

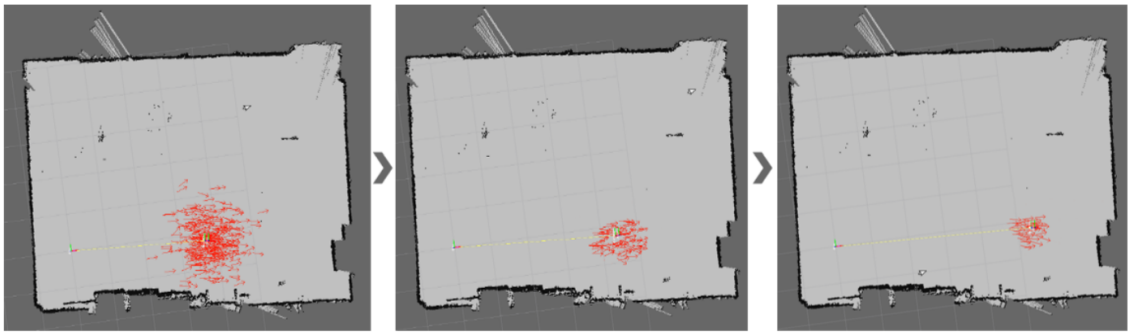
1: $\bar{X}_k = X_k$
2: **for** $m = 1 : M$ **do**
3: $\quad x_k^m =$ hareket-güncelleme(u_k, x_{k-1}^M)
4: $\quad w_k^m =$ sensör-güncelleme(z_k, x_k^M)
5: $\quad \bar{X}_k = \bar{X}_k + (x_k^m, w_k^m)$
6: **for** $m = 1 : M$ **do**
7: $\quad w_k^m$ olasılığı ile X_k' dan x_k^m tahmini
8: $\quad X_k = X_k + x_k^m$

Algoritma başlangıcında parçacıklar ya rastgele tüm uzaya dağılmış şekilde , ya da belirli bir bölge etrafında dağılmış olarak başlatılır. Gerçekleşen her bir hareket ve yönelim değişkenlerin durumlarını günceller. Sonuç olarak değişken kestirimi parçacıkların ağırlıklandırılmış toplamı ile elde edilir.

Şekil 2.7'de A009 araştırma laboratuvarının çıkarılmış harita görüntüsü verilmiştir. Şekil 2.8' de ise Robot Operating System (ROS)' da yapılan Monte Carlo lokalizasyon uygulamasının çıktıları görülmektedir. A009 araştırma laboratuvarında yapılan çalışmada üzerinde lazer sensör bulunan altı-rotorlu bir İHA kullanılmıştır. Başlangıçta belirsizliğin çok olduğu, İHA hareket ettikçe, yeni gelen bilgiler doğrultusunda, parçacıkların bir noktada yoğunlaştığı görülmektedir. Yani başlangıçta fazla olan belirsizlik gittikçe azalmıştır.



Şekil 2.7 A009 Araştırma laboratuvarının çıkarılmış harita görüntüsü



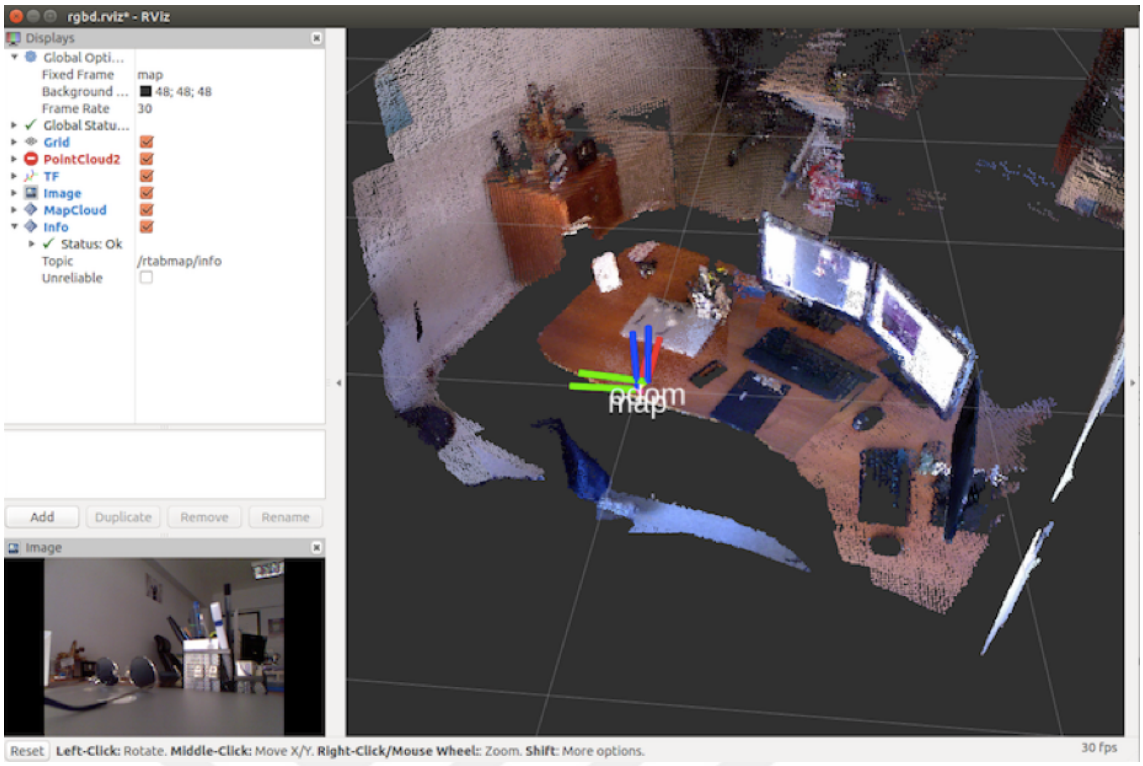
Şekil 2.8 ROS platformunda çalıştırılan Monte Carlo Algoritma çıktısı

2.6 3 Boyutlu Haritalama ve Konumlandırma

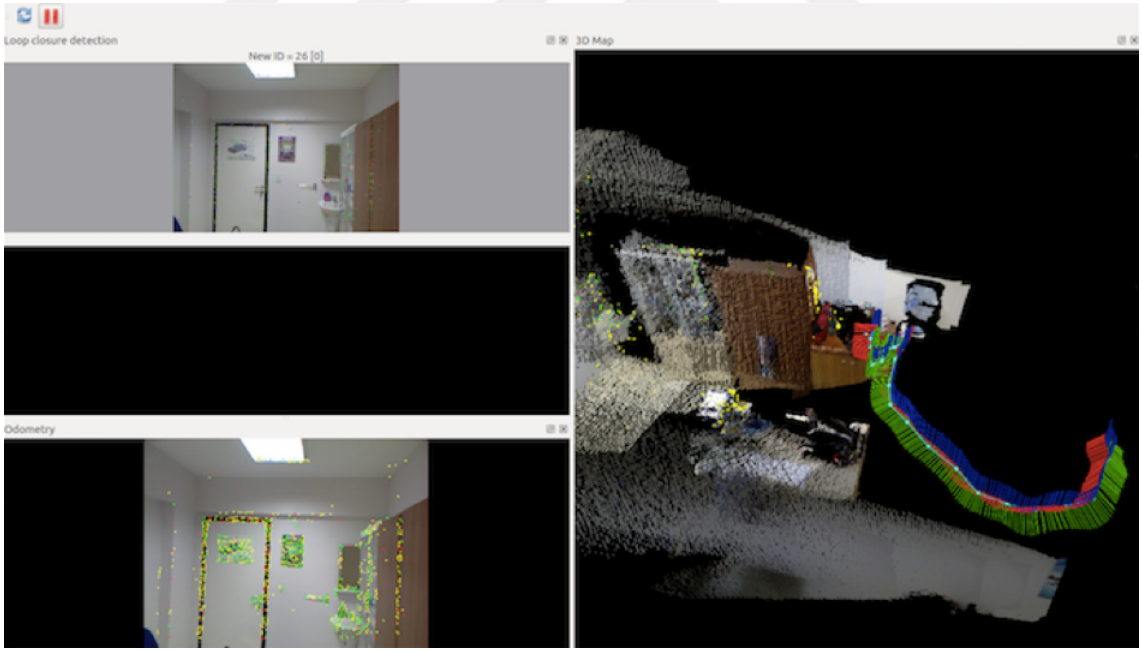
3 boyutlu SLAM, 2 boyutlu SLAM tekniklerine göre daha fazla ortam farkındalığına sahiptir ve daha fazla bilgi içerir. Cisimler 3 boyutlu ortamda konumlandırılabilmektedir. 3D lazer tarayıcılar, stereo kameralar ve RGB-D kameralar 3D SLAM için sıklıkla kullanılır.

Gerçek Zaman Görünüm Tabanlı Haritalama (Real-Time Appearance-Based Mapping) en çok kullanılan 3D SLAM algoritmalarından biridir [32]. RTAB-Map temelinde, SLAM uygulamasının gerçek zamanlı yapılabilmesi için döngü kapatma tespiti (loop closure detection) ve grafik optimizasyonu tekniklerini birleştirir [33].

Algoritma, özellik çıkarımı tekniklerinden biri olan Kelimelerin Torbası (bag-of-words)



Şekil 2.9 Gerçek Zaman Görünüm Tabanlı Haritalama - 1



Şekil 2.10 Gerçek Zaman Görünüm Tabanlı Haritalama - 2

[34] algoritmasının özelliklerini kullanan Görsel Kelimeler (Visual Words) [35] tekniğini kullanır. Bu yöntem sayesinde çevreden çıkarılan bilgilerin konumlandırılması sağlanır ve döngüleri tespit etmek adına ziyaret edilen konumlar sanal kitaplığa kaydedilir. Bir döngü tespit edildiğinde döngüdeki hatalar giderilir ve döngü tamamlanır. Algoritma, karşılaştırma performansının hızını arttırmak için

kd-tree [36] yöntemine başvurur. RTAB-Map ayrıca hafızayı verimli kullanmak için bir hafıza yönetim modeline sahiptir. Şekil 2.9 ve Şekil 2.10' da A114 nolu ofis için yapılan RTMAP-Map uygulaması ekran görüntüleri verilmiştir.

3
Parçacık Akış Filtresi ile Yüksek Boyutlu Nesne Takibi

Dinamik bir sistem hakkında çözümleme ve çıkarım yapabilmek için iki modele ihiyaç vardır. Birincisi sistem modeli, ikincisi ise ölçüm modelidir. Sistem modeli durumların zamanla değişimini açıklayan ve sistemi tanımlamak için tüm bilgileri içeren bir modeldir. Ölçüm modeli ise durum vektörleriyle ilgili gürültülü olabilecek gözlemleri temsil eden bir modeldir. Olasılıksal durum uzay formülasyonu ve yeni ölçümlerin alınmasıyla ilgili bilgilerin güncellemesi gerekliliği göz önünde bulundurulduğunda Bayes yaklaşımı uygun bir çözümdür [5]. Durum uzay yaklaşımı, çok değişkenli verileri ve doğrusal olmayan süreçleri ele almak için elverişli bir yapıya sahip olup, geleneksel zaman serisi tekniklerine göre önemli avantajlar sağlar [37].

Bayes yaklaşımında, dinamik durum tahmini için, mevcut tüm bilgilere dayanarak olasılıksal yoğunluk fonksiyonu oluşturulur. Her ölçüm alındığında yeni bir tahmin yapılması gerektiğinden, çözüm için özyinelemeli bir yaklaşım daha uygundur. Özyinelemeli filtreleme yaklaşımı, alınan verilerin toplu olarak işlenmesi yerine sıralı olarak işlenebileceği anlamına gelir. Böylece tüm veri setinin saklanmasına veya yeni bir ölçüm alınması durumunda mevcut verilerin tamamen yeniden işlenmesine gerek olmaz.

Parçacık filtreleri, geleneksel Kalman filtreleme yöntemlerini genelleyen ardışık Monte Carlo yöntemleridirler ve herhangi bir uzay-durum modeline uygulanabilirler. [5]. Ele alınan sistemin durum boyutu yüksek olduğunda, önsel dağılımdan çekilen parçacıkların çoğunluğu, ihmal edilebilir ağırlıklara yol açan çok düşük olasılıksal bölgelere dönüşecektir. Ağırlık dejenerasyonu problemi, soncul dağılımın Monte Carlo yaklaşımı birkaç parçacık tarafından belirlendiğinden dolayı, soncul dağılımın zayıf bir şekilde temsil edilmesine yol açar [6]. Optimal teklif dağılımı yaklaşımı, önem ağırlıklarının varyansını en aza indiren bir çözüm sunar [7].

Yardımcı parçacık filtresi, yeni ölçümlerden gelen bilgileri hesaba katması sayesinde, parçacıkların daha etkili bir şekilde örneklenmesini sağlar [8]. Rao-Blackwellised parçacık filtresi, Monte Carlo tahminlerinin varyansını azaltan bir çözüm sunar [9].

Yüksek boyutlu sistemlerde doğrusal olmayan filtreleme, başta çoklu hedef takibi olmak üzere birçok uygulamada ortaya çıkan önemli bir problemdir. Bahsi geçen metotlar, yüksek boyutlu problemlerde yetersiz performans sergilemekte [10] ve parçacık filtresinin performansının, durum boyutu arttıkça önemli ölçüde azalmasını engelleyememektedir [11]. Bu problemin çözümü için başka yöntemler de sunulmuştur. Eş ağırlıklar parçacık filtresi [12], yüksek boyutlarda ümit verici bir performans göstermesine rağmen, geleneksel parçacık filtresinin istatistiksel tutarlılığından ödün vermektedir. Çözüm sunan diğer yaklaşımlar ise, durum uzayını çarpanlarına ayırma yoluyla önceki değerlerin uygun bir faktörizasyonunu belirlemeye dayanan [13] ve [14] yöntemleridir. Ancak bu yöntemlerinde uygulanabilirlikleri kısıtlıdır [15].

Doğrusal olmayan filtreleme problemine çözüm olarak alternatif bir yaklaşım Daum ve Huang tarafından gelmiştir [16–18]. Daum-Huang filtrelerinde, her zaman aralığı için, önsel ve sonsal dağılımlar arasında bir homotopi fonksiyonu uygular. Bu fonksiyon, kısmı diferansiyel denklemin çözümü olarak tarif edilen bir parçacık akışını ifade eder. Parçacık akışı, bir dizi parçacığın, önsel dağılımın büyük değerlere sahip olduğu bölgelere kademeli olarak göç etmesini sağlar [19].

Exact Daum-Huang filtresi algoritmanın içerisinde paralel olarak yürütülen genişletişmiş Kalman filtresine fazla bağımlılık göstermektedir. Bu bağımlılık EDH'nin bazı senaryolarda yetersiz sonuçlar vermesine neden olmaktadır [20]. Buna ek olarak [19]'da, sistemin ve gözlem gürültülerinin Gauss bir dağılım çizdiği, sistem haritasının doğrusal olduğu fakat elde edilen sonuçların doğrusal olmadığı durumlar üzerine bir çözüm geliştirilmiş ve Localized Exact Daum-Huang parçacık filtresi tanıtılmıştır. Yapılan iyileştirme, sistemin doğrusallaştırılması ve tüm parçacıkların bir seferde göç parametrelerinin hesaplanmasından ziyade, her bir parçacık için parametrelerin tek tek hesaplanmasını önermesidir. Bu iyileştirme karmaşık problemlerde performans iyileştirmesi sağlamasına rağmen algoritmanın hesaplama maliyetini önemli ölçüde arttırmaktadır.

3.1 Parçacık Akış Filtresi

Parçacık akış filtresi sayesinde, parçacık dejenerasyonu problemi ortadan kaldırılmış ve parçacıkların, belirlenen bir logaritmik homotopi fonksiyonu aracılığıyla, sonsal olasılığı en yüksek olan parçacıklara doğru en hızlı bir şekilde yakınsaması sağlanmıştır. Kullanılan homotopi fonksiyonu parçacık akışı için önsel ve sonsal olasılılık yoğunluklarının dağılımları arasında geçişi tanımlamaktadır [38].

Normalize edilmemiş marjinal önsel yoğunluğu tanımlamayan Bayes kuralı,

$$\widetilde{p}(x_k|y_{1:k}) = p(y_k|x_k)p(x_k|y_{1:k-1}) \tag{3.1}$$

olarak verilsin. Daum ve Huang homotopi fonksiyonunu,

$$\phi(x_k, \lambda) = \log g(x_k) + \lambda \log h(x_k) \tag{3.2}$$

olarak tanımlanmıştır. Burada,

$$h(x_k) = p(y_k|x_k) \tag{3.3}$$

$$g(x_k) = p(x_k|y_{1:k-1}) \tag{3.4}$$

ve λ gerçek değerli olan ve 0-1 arasında tanımlı, parçacık akış miktarını belirleyen bir adım parametresidir.

λ evrildikçe, homotopi fonksiyonunun sabit kalacağı şekilde filtre tasarlanmıştır. Bu gereksinim, kısmı diferansiyel denkleme yol açar ve parçacıkların akışı,

$$\frac{\partial \phi}{\partial x}\frac{dx}{d\lambda} + \frac{\partial \phi}{\partial \lambda} = 0 \tag{3.5}$$

çözülerek hesaplanır. Daum ve Huang, Fokker-Planck denklemini kullanarak,

$$\frac{\partial \phi}{\partial x}\psi(x, \lambda) + \log(h) = -Tr(\frac{\partial \psi}{\partial x}) \tag{3.6}$$

denklemini türetmiş, tam akış DH filtresini [18] tanıtarak filtreyi geliştirmiş ve genelleştirmiş. Burada,

$$\frac{dx}{d\lambda} = \psi(x, \lambda) \tag{3.7}$$

olup, olasılıksal yoğunluk akışı, bu denklemin çözümü ile elde edilir. Bu durumda da parçacıkların iterasyondaki akış miktarları,

$$\frac{dx}{d\lambda} = A(\lambda)x + b(\lambda) \tag{3.8}$$

olarak hesaplanır [18]. Burada,

$$A = -\frac{1}{2}PH^T(\lambda HPH^T + R)^{-1}H \tag{3.9}$$

ve

$$b = (I - 2\lambda A)[(I + \lambda A)PH^T R^{-1}z + A\overline{x}] \tag{3.10}$$

olarak verilmiştir. P tahmin hatasının ve R ölçüm gürültüsünün kovaryans değerlerini tanımlamaktadır. H ölçüm matrisidir. x ,her döngüde göç eden durum değişkenini ifade ederken, $dx/d\lambda$ ise, x'in adım büyüklüğünün λ'ya bağlı değişimini belirlemektedir.

3.2 Parçacık Akış Filtresinin Uygulanışı

Algoritma 3.1 EDH Filtresi algoritması

Başlangıç: $p(x_0)$'dan $(x_0^i)_{i=1}^N$ elde et
$\hat{x}_0$, m_0 *ve* P_0 *tanımla*
for $k = 1 : T$ **do**
    Parçacıkların yayılması: $x_{k-1}^i = f_k(x_{k-1}^i) + v_k$
    $\overline{x}_k$ hesapla
    UKF/EKF tahmini: $(m_{k-1|k-1}, P_{k-1|k-1}) \rightarrow (m_{k|k-1}, P_{k|k-1})$
    for $j = 1 : N_\lambda$ **do**
        $\lambda = j\Delta\lambda$
        $\overline{x}_k$'da $\gamma_k()$ doğrusallaştırılması ile $H_{\overline{x}}$ hesapla
        A ve b hesapla
        for $i = 1 : N$ **do**
            Her bir parçacık için $dx_k^i/d\lambda$ hesapla
            Parçacıkların göçü: $x_k^i = x_k^i + \Delta\lambda(dx_k^i/d\lambda)$
        x_k^i kullanarak $\overline{x}_k$ tekrar hesapla
    UKF/EKF güncelleme: $(m_{k|k-1}, P_{k|k-1}) \rightarrow (m_{k|k}, P_{k|k})$
    $P_{k|k}$ kullanarak x_k^i' dan $\widehat{x}_k$'ı tahmin et
    Parçacıkları yeniden düzenle: $x_k^i \sim N(\widehat{x}_k, P_{k|k})$ (isteğe bağlı)

Algoritma 3.1, EDH filtresinin anlatılan teorik bilgilere dayanarak sözde kodlamasını içerir. Bu algoritmada, N parçacık sayısı ve T zaman aralıklarının sayısını temsil eder.

EKF/UKF durum ve kovaryans matris tahminleri sırasıyla m ve P ile gösterilir. Parçacık göçü 7-16 satırlar arasında gerçekleştirilmektedir.

6. satırda UKF/EKF tahmini, 17.satırda UKF/EKF güncelleme işlemleri yapılmaktadır. Ölçüm matrisi $H_{\overline{x}}$, mevcut tahmin $\overline{x}_k$ değerinde doğrusallaştırılarak 9.satırda hesaplanır. Güncellemeyi hesaplamak adına burada tüm parçacıklar için, aynı A ve b değerleri kullanılır. 17. adımdan yapılan, parçacıkların yeniden düzenlenmesi işlemi isteğe bağlıdır.

Ding ve Coates tarafından yapılan basit ama etkili değişim sonrasında tanıtılan LEDH filtresine ait söz kodlar Algoritma 3.2' de verilmiştir [19]. Algoritma 3.2, Algoritma 3.1'de 7-19 arasındaki satırları değiştiren bir algoritmadır. Diğer kısımlar Algoritma 3.1' de verildiği gibidir.

Burada iki büyük değişiklik vardır. Her bir parçacık için, ölçüm fonksiyonunun ilgili parçacıkla alakalı bölümü doğrusallaştırılmaktadır ve ilgili parçacığın A_i ve b_i değerleri, H_i matrisi kullanılarak hesaplanmaktadır. Böylece göç parametrelerinin hesaplanması, her bir parçacık için tek tek yapılmaktadır. Diğer bir değişiklik ise, ortalama UKF/EKF'den, Daum-Huang filtresinden gelen durum tahminiyle değiştirilmiştir [19].

Algoritma 3.2 LEDH Filtresi algoritması

1: **for** $j = 1 : N_\lambda$ **do**
2: $\lambda = j\Delta\lambda$
3: **for** $i = 1 : N$ **do**
4: x^i'de $\gamma_k()$ doğrusallaştırılması ile $H_{\overline{x}}$ hesapla
5: A^i ve b^i hesapla
6: Her bir parçacık için $dx_k^i/d\lambda$ hesapla
7: Parçacıkların göçü: $x_k^i = x_k^i + \Delta\lambda(dx_k^i/d\lambda)$
8: x_k^i kullanarak $\overline{x}_k$ tekrar hesapla
9: UKF/EKF güncelleme: $(m_{k|k-1}, P_{k|k-1}) \rightarrow (m_{k|k}, P_{k|k})$
10: $P_{k|k}$ kullanarak x_k^i' dan $\widehat{x}_k$'ı tahmin et
11: $m_{k|k} = \widehat{x}_k$
12: Parçacıkları yeniden düzenle: $x_k^i \sim N(\widehat{x}_k, P_{k|k})$ (isteğe bağlı)

3.3 Kümelenmiş Parçacık Akış Filtresi

Parçacık akış filtreleri, geleneksel parçacık filtrelerinin aksine, bünyesinde yoğunluk fonksiyonu ve önem örneklemesi içermeyen, logaritmik homotopi fonksiyonundan türetilmiş filtrelerdir. Parçacık akışı kullanarak, parçacıkların başarılı bir şekilde

göç etmesini sağlamaktadırlar. EDH bu göç için gerekli olan parametreleri tek seferde hesaplarken, onun modifiye edilmiş ve daha performanslı versiyonu olan LEDH filtresi bu göç parametrelerini tek tek hesaplamaktadır. Clustered Exact Daum-Huang filtresi ile, hata payları üzerinden bir kümeleme algoritması çalıştırılarak göç parametrelerinin kümeler için ortak hesaplanması ile LEDH Filtresinin hesaplama maliyetinde iyileştirmeler yapılmıştır. [39]. Ayrıca hata değeri yüksek parçacıkların aynı kümede toplanması ve en büyük hataya sahip kümenin elenmesi sayesinde bu parçacıkların genel sistem üzerindeki olumsuz etkileri azaltılmaktadır [39].

CEDH filtresinde, parçacıklar hata değerleri üzerinden, K-Ortalamalar++ algoritması [21] kullanılarak kümelenmesi sayesinde ne EDH filtresinde olduğu gibi tüm parçacıklar tek bir parametre hesaplamasına tabi tutulmakta ne de LEDH filtresinde olduğu gibi tek tek hesaplanmaktadır.

3.3.1 K-Ortalamalar++ Kümeleme Algoritması

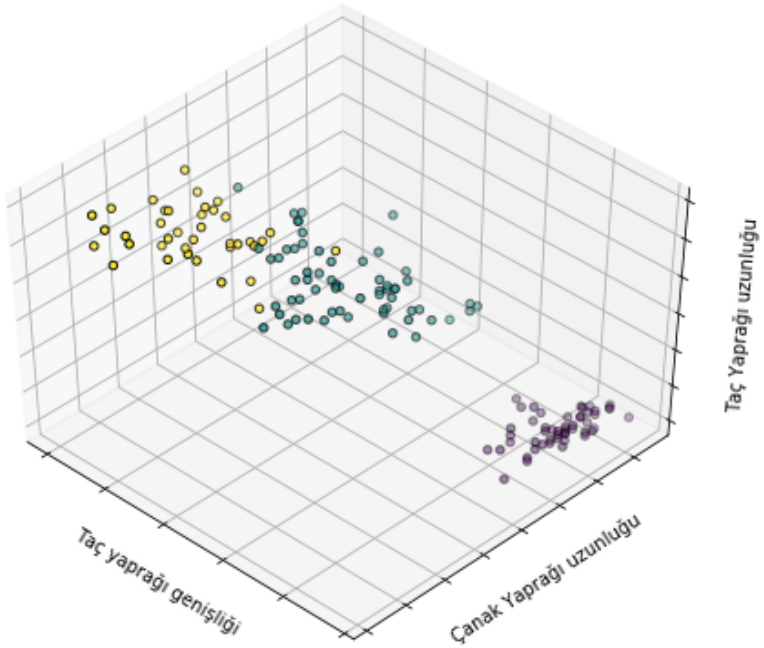
Kümeleme, etiketsiz verilerin küme adı verilen gruplara bölünmesidir. K-Ortalamalar denetimsiz öğrenme yöntemlerinden biri olup, verileri k adet bölümlere ayıran bir kümeleme algoritmasıdır. Her bir küme bir merkez noktaya sahiptir olup burada k kullanıcı tarafından belirlenmektedir.

K-Ortalamalar yöntemi, aynı kümedeki noktalar arasındaki ortalama mesafenin karesini en aza indirmeyi amaçlayan yaygın olarak kullanılan bir kümeleme tekniğidir [40]. Verilen k değeri için, K-Ortalamalar algoritması aşağıda listelendiği şekilde çalışır:

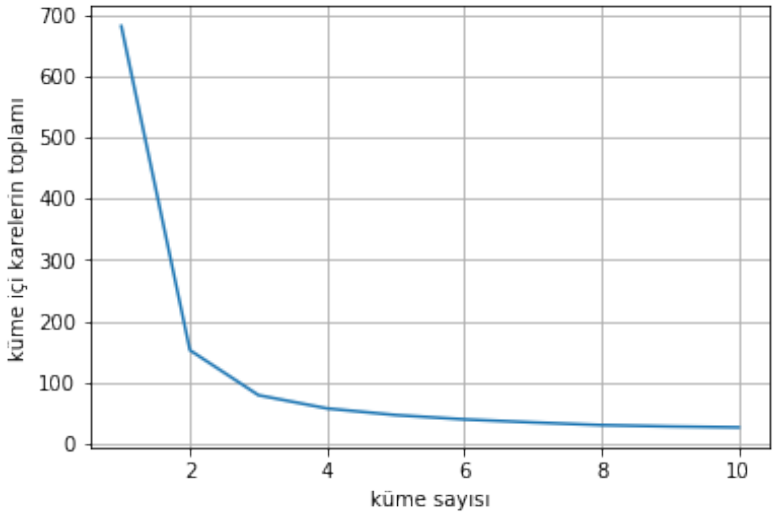
1. k adet veri noktasını başlangıç ağırlık merkezi olarak belirle.
2. Her bir noktayı, en yakın olduğu ağırlık merkezine ait kümeye ata.
3. Ağırlık merkezlerini, yeni küme elemanlarını kullanarak yeniden hesapla.
4. 2. ve 3. adımı, ağırlık merkezleri yer değiştirmeyene kadar tekrarla.

Arthur ve Vassilvitskii, basit bir rastgele tohumlama tekniği ile, K-Ortalamalar algoritmasını güçlendirerek, optimal kümeleme yöntemine sahip K-Ortalamalar++ algoritmasını tanıtmışlardır [21].

Şekil 3.3' de iris veri kümesi için, taç yaprağı genişliğine, çanak yaprağı genişliğine ve taç yaprağı uzunluğuna göre kümelenmiş K-Ortalamalar++ algoritma çıktısı görülmektedir. Burada $k = 3$ seçilmiştir. Uygun k değerini seçmek için dirsek



Şekil 3.1 İris veri kümesi için K-Ortalamalar++ kümeleme algoritmasının çıktısı



Şekil 3.2 Değişen küme sayısına göre küme içi kare toplamları

yöntemi gibi bazı yöntemler mevcuttur. Bu yöntemde, küme içerisindeki elemanların ağırlık merkezlerine olan uzaklıklarının karelerinin toplamı hesaplanmaktadır. Belli bir noktada, küme içi kare toplamının türevi hızlı bir azalmaya maruz kalmaktadır. Dirsek yöntemi, k değeri için o noktanın seçilmesini önermektedir. Şekil 3.2' de aynı iris veri seti için 1'den 10'a kadar olan k küme sayılarının küme içi kare toplamları verilmiştir. Bu sonuca göre $k = 3$ bu veri seti için uygulanabilir gözükmektedir.

Hızlı ve başarılı bir yöntem olması nedeniyle CEDH filtresi için kümeleme yöntemi olarak K-Ortalamalar++ algoritması seçilmiştir.

3.3.2 CEDH Filtresinin Uygulanışı

LEDH filtresinde, her bir sözde zaman aralığında, her bir parçacık için akış parametrelerinin hesaplanmasını gerektiğinden, hesaplama maliyeti yüksek olabilmektedir.

Algoritma 3.3 CEDH Filtresi algoritması

1: *Başlangıç:* $p(x_0)$*'dan* $(x_0^i)_{i=1}^N$ *elde et*
2: $\hat{x}_0$, m_0 *ve* P_0 *tanımla*
3: **for** $k = 1 : T$ **do**
4: Parçacıkların yayılması: $x_{k-1}^i = f_k(x_{k-1}^i) + v_k$
5: $\overline{x}_k$ hesapla
6: UKF/EKF tahmini: $(m_{k-1|k-1}, P_{k-1|k-1}) \rightarrow (m_{k|k-1}, P_{k|k-1})$
7: **for** $j = 1 : N_\lambda$ **do**
8: $\lambda = j\Delta\lambda$
9: **for** $i = 1 : N$ **do**
10: x^i'de $\gamma_k()$ doğrusallaştırılması ile $H_{\overline{x}}$ hesapla
11: Parçacıkların hatalarına oranlarına göre kümelenmesi
12: **for** $c = 1 : N_c$ **do**
13: A^c ve b^c hesapla
14: Her bir parçacık için $dx_k^c/d\lambda$ hesapla
15: Parçacıkların göçü: $x_k^c = x_k^c + \Delta\lambda(dx_k^c/d\lambda)$
16: En fazla hataya sahip en büyük kümenin elenmesi
17: x_k^i kullanarak $\overline{x}_k$ tekrar hesapla
18: UKF/EKF güncelleme: $(m_{k|k-1}, P_{k|k-1}) \rightarrow (m_{k|k}, P_{k|k})$
19: $P_{k|k}$ kullanarak x_k^i' dan $\widehat{x}_k$'ı tahmin et
20: Parçacıkları yeniden düzenle: $x_k^i \sim N(\widehat{x}_k, P_{k|k})$ (isteğe bağlı)

CEDH filtresinin sözde kodları Algoritma 3.3'de verilmiştir. Burada N_c küme sayısını belirtmektedir. Kümeleme 11. satırda parçacıkların hata payları üzerinden yapılmaktadır. 12-15 arasındaki satırlarda yapılan kümeleme sayısı kadar göç

parametreleri hesaplanmaktadır. Yani ne EDH filtresinde olduğu gibi parçacıklar tek bir parametre hesaplamasına tabi tutulmakta ne de LEDH filtresinde olduğu gibi, tek tek parçacıkların parametreleri hesaplanmaktadır. Zaten hataları yakın olan parçacıklar için yaklaşık yakın göç parametreleri hesaplanacağından, CEDH burada kümelere yaparak gereksiz hesaplama maliyetini düşürmektedir. 16. satırda ise en fazla hataya sahip en büyük küme elenmektedir. Bu şekilde de, hesaplama maliyeti düşürülürken, hata oranı fazla parçacıkların genel sistem üzerindeki etkileri azaltılmakta ve bu da performans iyileştirmesini beraberinde getirmektedir.

3.4 Benzetim Çalışması

Yöntem sonuçları, [41]'dan uyarlanmış çok hedefli bir takip problemi ile gösterilmektedir. Burada, Şekil 3.3'de gösterilen $40mx40m$ boyutunda dikdörtgen bir bölge içerisinde, bölgeyi bölen ızgaraların kesişme noktalarına eşit olarak yerleştirilmiş 25 adet akustik genlik sensör düğümlerinden oluşan bir kablosuz sensör ağı modellenmektedir. İki eksende, birbirinden bağımsız hareket eden dört hedef bulunmaktadır. Genel durum vektörü, bu dört hedefin her biri için hız, konum ve türevleri dahil olmak üzere 4 durum içermekte ve dolayısıyla toplamda 16 boyutludur.

p hedefi belirtmekle birlikte, sabit bir hızda hareket eden 4 adet cismin bağımsız hareket modeli,

$$x_k^{(p)} = F x_{k-1}^{(p)} + W_p v_k^{(p)} \tag{3.11}$$

olarak verilmektedir. Burada,

$$x_k^p = [x_k^p, y_k^p, \dot{x}_k^p, \dot{y}_k^p] \tag{3.12}$$

olup, $p.$ hedefin x-y pozisyonu ve x-y hız bileşenlerini içerir. Geçiş matrisi ise,

$$F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

olup, ölçüm gürültüsü,

$$v_k^{(p)} \sim N(0, \sigma_v^2 V) \tag{3.14}$$

olarak verilmiştir. Uygulamada, $\sigma_v^2 = 0.01$ olarak ve filtre için proses gürültüsünün kovaryansı,

$$W_p = \begin{pmatrix} 0.5 & 0 & 0.2 & 0 \\ 0 & 0.5 & 0 & 0.2 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.15}$$

olarak seçilmiştir.

Her bir hedef (p), sabit ve bilinen varsayılan bir A_p genliği ile bir ses yayar. j sensör pozisyonunda, ζ, hedef p nedeniyle oluşan ses büyüklüğü,

$$A_p/(\|p_k^{(p)} - \zeta_k^i\|^\kappa + d_0) \tag{3.16}$$

olarak modellenmiştir. Hedef p'nin pozisyonu,

$$p_k^{(p)} = (x_k^{(p)}, y_k^{(p)})^T \tag{3.17}$$

olup, κ yol kaybı üssü ve d_0 maksimum ölçülebilir genliği belirleyen bir eşiktir. k zamanında j sensörüyle elde edilen ölçüm,

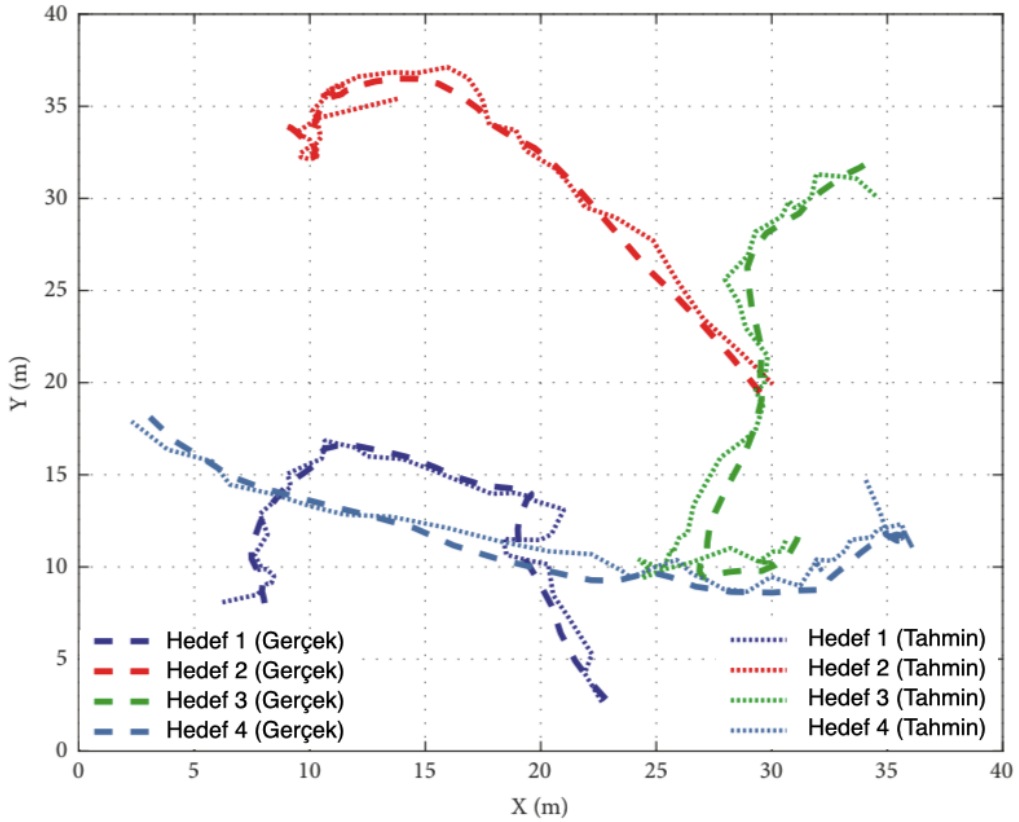
$$y_k^i = \gamma_k^i(x_k) + v_k^j \tag{3.18}$$

olur, burada,

$$\gamma_k^i(x_k) = \sum_{p=1}^{p} \frac{A_p}{\|p_k^p - \zeta_k^j\|^k + d_0} \tag{3.19}$$

eşit olup,

$$v_k^j \sim N(0, \sigma_v^2) \tag{3.20}$$



Şekil 3.3 CEDH filtresi kullanılmış gerçek ve tahmini rota örneği

ortalama varyansın sıfır ortalama Gauss değişkenidir.

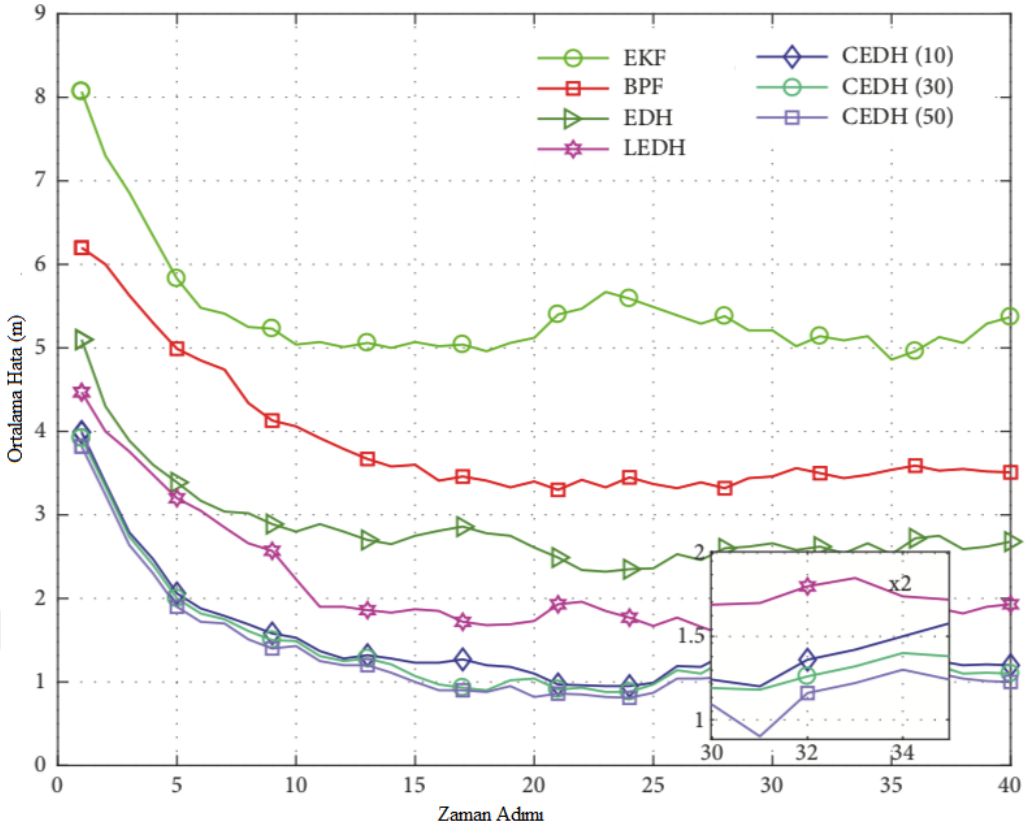
Detayları verilen simülasyon modeli için hedeflerin olasılıksal olarak oluşturulan yörüngeleri ve bu yörüngelere ait CEDH tahmin sonuçları Şekil 3.3'de verilmiştir. Sonuçlar 100 defa çalıştırılan benzetim çalışmasına ait 67. iterasyona denk gelen sonuçların çıktısını göstermektedir.

Simülasyon 40 zaman adımından oluşmaktadır ve her bir algoritma ayrı ayrı 100 defa çalıştırılarak, her bir zaman aralığının kendi içinde ortalaması alınarak OMAT hesaplanmıştır. EKF, BPF, EDH, LEDH ve CEDH için hesaplanan ortalama OMAT hata değerleri metre cinsinden Şekil 3.4'de verilmiştir.

Tablo 3.1 Filtreler hakkında detaylı bilgiler

Algoritma	EKF	PF	EDH	LEDH	CEDH	CEDH	CEDH
Parçacık sayısı	-	10000	500	500	500	500	500
Lambda	-	-	30	30	30	30	30
Küme sayısı	-	-	-	-	10	30	50
Ortalama hata (m)	4.47	3,79	3,24	2,60	1,50	1,41	1,37
Gerçekleme zamanı (s)	0,03	1,57	0,48	44,73	23,05	24,79	28,42

Şekil 3.4' de görüldüğü üzere EKF filtresi hata payı en yüksek olan filtredir. Onu 10000

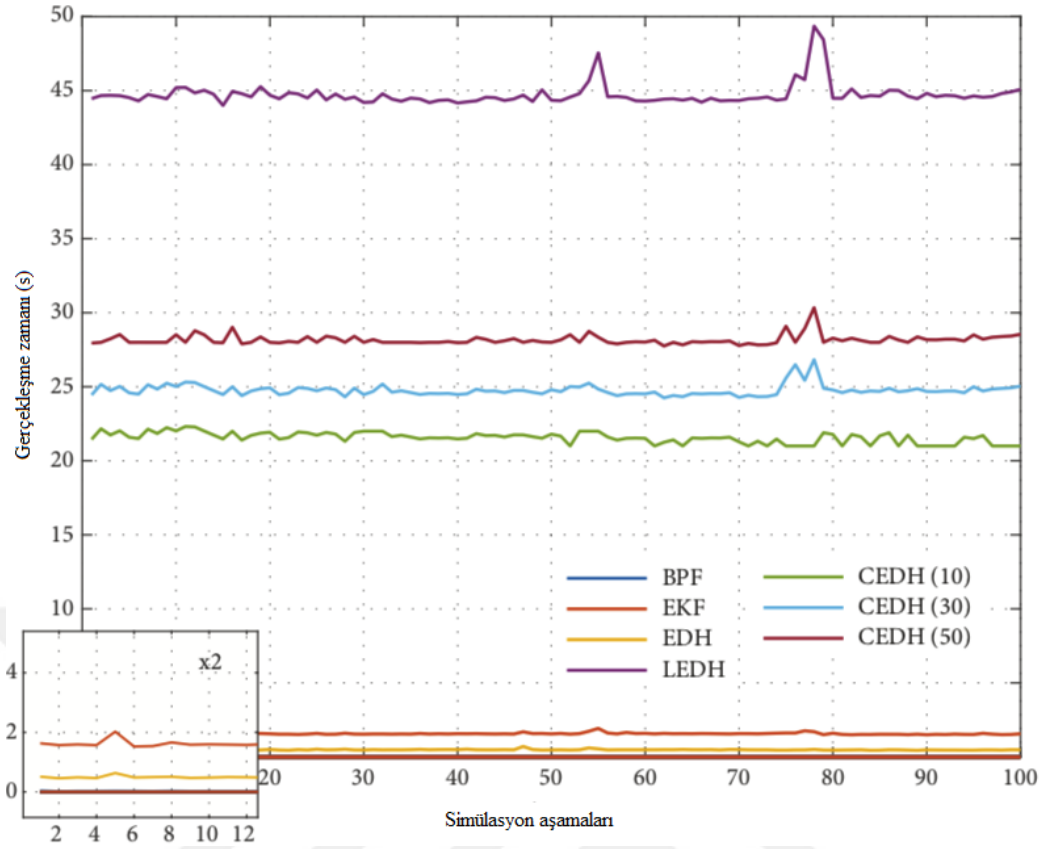


Şekil 3.4 Filtrelerin tüm zaman adımları için ortalama hataları

tane parçacık ile gerçeklenen standart parçacık filtresi izlemektedir. Daha sonrasında sırayla 500 parçacık ve 0'dan 30'a üstsel olarak dağıtılan λ değeri ile gerçeklenen EDH ve LEDH filtreleri gelmektedir. Aynı parçacık ve λ değerine sahip CEDH ise tüm zaman aralıklarında en yüksek performansı göstermektedir. Ayrıca k küme sayısının arttırılmasının performansa olumlu etkisi olduğu görülmektedir.

Algoritmalara ait parçacık sayısı, λ değeri, ortalama gerçekleme süreleri ve hata değerleri Tablo 3.1'de detaylı olarak verilmiştir. Matlab ortamında benzetimi yapılan çalışma için kullanılan bilgisayar Macbook Air (Early 2015) olup, bu bilgisayar 1,6 Ghz Intel Core i5 işlemciye, 4GB 1600 Mhz DDR3 belleğe ve Intel HD Graphics 5000 1536 MB ekran kartına sahiptir.

Şekil 3.6'da görüldüğü gibi CEDH algoritması, LEDH algoritmasının neredeyse yarı zamanda çalışarak hesaplama maliyetine önemli iyileştirmeler getirmiştir. Ancak k küme sayısının artması performansı arttırırken, hesaplama maliyetini de arttırdığı görülmektedir. Burada aynı hata değerlerine sahip parçacıklar için zaten yaklaşık olarak yakın göç parametreleri hesaplanacağından, CEDH algoritması bunu tek tek tüm parçacıklar için hesaplamaktansa, kümeler için ortak olarak hesaplayarak çok daha hızlı bir şekilde gerçeklemektedir. Verilen çok boyutlu sisteme ait hedef konumlandırma problemini CEDH başarılı bir şekilde yapmaktadır.



Şekil 3.5 Filtrelerin tüm zaman adımları için gerçekleme süreleri

4
Derin Öğrenme Ağları ile Konumlandırma

Derin öğrenmenin temelini yapay sinir ağları oluşturduğundan, derin ağları anlamadan önce YSA'yı öğrenmek gerekmektedir. YSA, ilk olarak 1943'de nörofizyolog Warren McCulloch ve matematikçi Walter Pitts tarafından tanıtılmıştır. McCulloch ve Pitts, karmaşık hesaplamaları gerçekleştirmek için hayvan beyinlerindeki biyolojik nöronların birlikte nasıl çalıştığını açıklayan basitleştirilmiş bir hesaplama modeli sunmuşlardır [42].

1980'lerin başında, yapay ağlarda yapılan çalışmaların ve yeni mimarilerin geliştirilmesi bu alandaki umutların artmasını sağlasa da, donanım yönündeki eksiklikler ve bu yüzden meydana gelen uzun işlem süreleri YSA'ların yaygınlaşmasını engellemiştir. 1990'larda SVM gibi güçlü Makine Öğrenmesi tekniklerinin, YSA'lardan daha iyi ve hızlı sonuçlar vermesi, bu alandaki çalışmaların ikinci plana itilmesine sebep olmuştur. 2000'lerde donanım yönündeki gelişmeler, oyun sektörünün grafiksel işlemcilere olan yatırımların artmasına öncülük etmesi ve giderek artan ve ulaşılabilir veriler derin öğrenme ağlarının gelişmesine öncülük etmiştir. Büyük veriye sahip karmaşık problemler için Makine Öğrenmesi tekniklerine göre çok daha fazla tercih edilir hale gelmişlerdir.

4.1 Biyolojik Nöronlar

Çekirdek ve hücrenin karmaşık bileşenlerinin çoğunu içeren bir hücre gövdesinden oluşan biyolojik nöronlar, çoğunlukla hayvanların beyin korteklerinde bulunur. Dentrit adı verilen uzun dallanmalara ve akson adı verilen, dentritlerin bir iki katı uzunluktan, binlerce kat uzunluğuna kadar olabilen bir uzantıya sahiptirler. Aksonun uçunda telondria adı verilen bir çok uzantı vardır ve bu dalların ucunda sinaptik terminaller adı verilen minik yapılar bulunur. Bu yapılar diğer nöronların dentritlerine veya doğrudan hücre gövdelerine bağlıdırlar. Biyolojik nöronlar, sinapslar aracılığıyla diğer nöronlardan kısa elektrik sinyalleri alırlar. Bir nöron yeterli miktarda sinyali aldığında ise milisaniyelere mertebesinde bir sürede kendi sinyalini gönderir.

Korteks, her bir nöronun tipik olarak binlerce diğer nörona bağlandığı, milyarlarca büyük nöron ağından organize edilmiştir. Ağların karmaşıklığı arttıkça, çözebilecekleri problem ve sundukları çözümler giderek artmaktadır. Bu yapı, basit bir YSA'daki gizli katmanların arttırılarak derin ağların oluşturulmasına ilham vermiştir. Literatürde, tam bir rakam olmasa da, bir ağın derin ağ olarak adlandırılması için 2'den fazla gizli katmana sahip olması gerekmektedir.

4.2 Perseptron

Perseptron Frank Rosenblatt tarafından 1957'de tanıtılmış, basit YSA yapılarından biridir [43]. Perseptronlar, eşik mantık birimi adı verilen bir temele dayanan, giriş ve çıkış ünitelerinin ikilik tabanda değerler yerine sayıları sakladığı ve her bir giriş bağlantısının bir ağırlık ile ilişkilendirildiği yapılardır.

EMB'ye gelen bir giriş fonksiyonu,

$$z = w_1 x_1 + w_2 x_2 + ... + w_n x_n = x^T w \tag{4.1}$$

denklemine tabi tutularak ağırlıkların toplamı hesaplanır. Burada x girişi, z ağırlık toplamını ve w ağrılıkları temsil etmektedir. Daha sonrasında z, bir adım fonksiyonuna tabi tutularak çıktı hesaplanır. Perseptronlarda kullanılan en yaygın adım fonksiyonları birim basamak adım fonksiyonu,

$$step(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases} \tag{4.2}$$

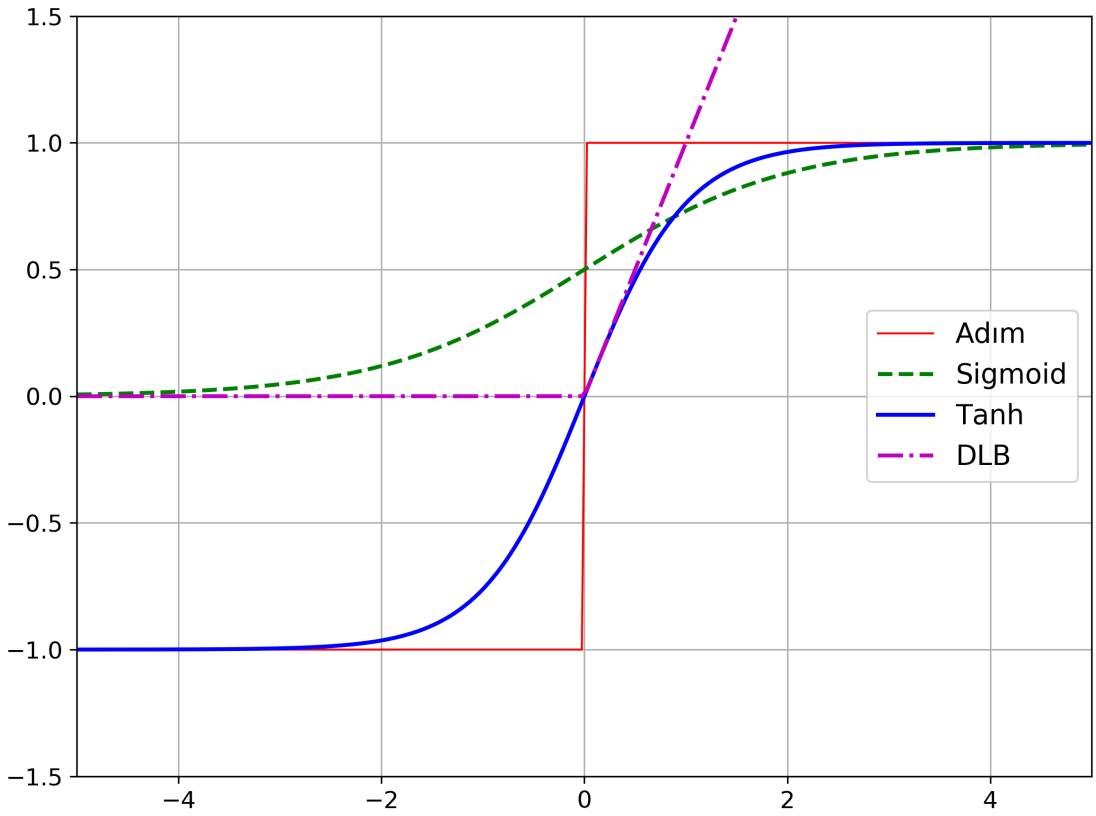
ve işaret fonksiyonudur:

$$sgn(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ +1 & z \geq 0 \end{cases} \tag{4.3}$$

Giriş değerleri eğer eşik değeri aşarsa çıkış pozitif, aşamazsa çıkış negatif olmaktadır. Bu mantık ile EMB basit bir doğrusal ikilik sınıflandırma problemi için kullanılabilir. Sınıflar pozitif sınıf veya negatif sınıf şeklinde, elde edilen çıktıya göre ayrılabilirler. Burada persepton tarafından hesaplanan ve bulunan değerler ağırlıklardır. EMB'yi

eğitmek, verilen modeli en iyi şekilde temsil edecek ağırlıklar bulunmaya çalışmak demektir.

Şekil 4.1'de adım, sigmoid, Tanh ve DLB aktivasyon fonksiyonlarının karşılaştırılması verilmiştir. Şekil 4.2'de ise ilgili aktivasyon fonksiyonlarının türevleri bulunmaktadır.



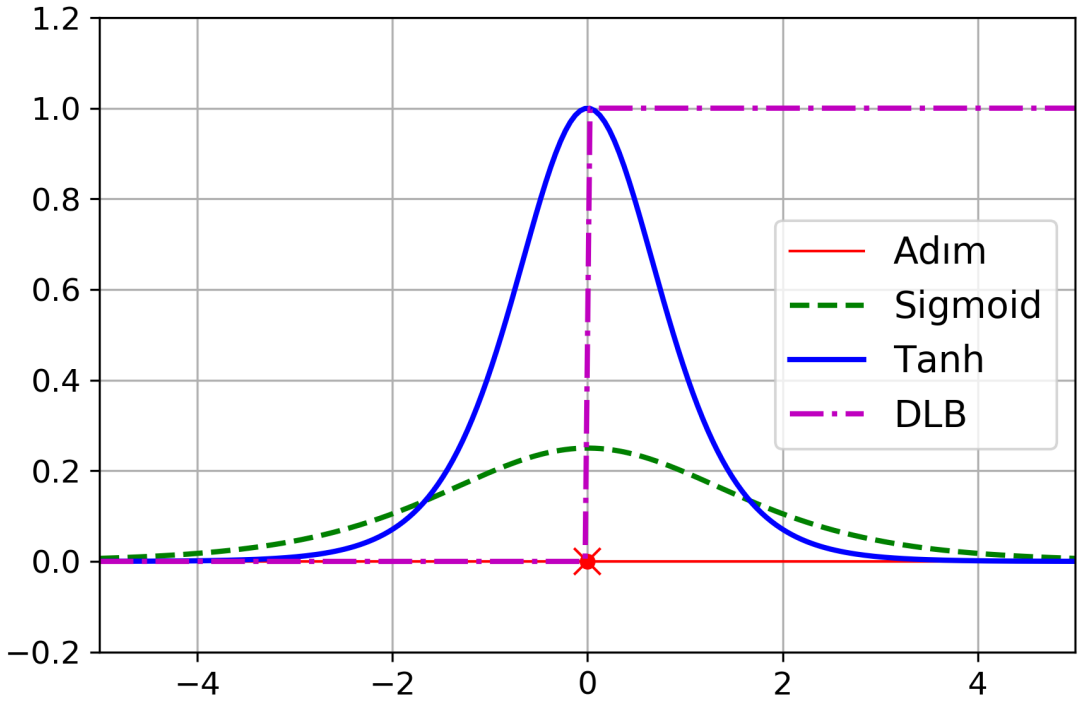
Şekil 4.1 Farklı aktivasyon fonksiyonlarının karşılaştırılması

Birbirine bağlı EMB'lerin birleşiminden bir perseptron oluşmaktadır. Eğer bir katmandaki tüm nöronlar bir önceki katmandaki nöronların hepsine bağlıysa, bu model yoğun katman olarak adlandırılmaktadır. Genellikle önyargı nöronu olarak adlandırılan ve tüm çıkışları 1 olan özel nöronlar sisteme eklenmektedir.

Bir katmandaki tüm ağırlık bulma işlemleri, lineer cebir sayesinde kolaylıkla, tek seferde yapılabilmektedir. Yoğun katmanlı bir perseptron için çıkış,

$$f_{w,b}(X) = \varphi(XW + b) \tag{4.4}$$

denklemi ile hesaplanabilmektedir. X giriş değerlerinin matris gösterimini, W katmandaki tüm ağırlıkları içeren ağırlıkların matris gösterimini, b önyargı vektörünü ve φ aktivasyon fonksiyonunu ifade etmektedir.



Şekil 4.2 Aktivasyon fonksiyonlarının türevleri

Rosenblatt tarafından biyolojik nöronların birbirlerini tetiklediği, aralarındaki bağların nöronları karşılıklı güçlendirdiği fikrinin ortaya atılması, perseptronların eğitim algoritmalarına ilham olmuştur [44]. Daha sonrasına bu fikir Löwel tarafından matematiksel ispatlarıyla güçlendirilmiştir. Löwel'e göre, iki nöron arasındaki benzerlik arttıkça, yani aynı çıkışı verdiği durumlarda, bağlantı ağırlıkları artmaktadır [45]. Bu kural, literatürde Hebbian öğrenme olarak da geçmektedir. Hebbian öğrenme tekniğinde ağlardaki hatalar dikkate alınarak, bağlantılar ağırlık hatalarının azaltılmasına zorlanmaktadır.

Perseptron öğrnem kuralı,

$$w_{i,j}^{k} = w_{i,j}^{k-1} + \eta(y_j - \hat{y}_j)x_i \tag{4.5}$$

olarak verilmektedir. η öğrenme oranını, $w_{i,j}$ $i.$ giriş nöronu ve $j.$ çıkış nöronu arasındaki ağırlıkları, x_i $i.$ giriş değerinin anlık değerini, $\hat{y}_j$ $j.$ çıkış nöronunun anlık tahmin değerini ve y_j ise $j.$ çıkış nöronunun gerçek değerini vermektedir.

4.3 Gradyan İniş Algoritması

Gradyan iniş, bir çok problem için optimal çözümü bulma kapasitesine sahip, genel bir optimizasyon algoritmasıdır ve parametreler uzayında değişimler yaparak, maliyet fonksiyonunu minimize etmeye çalışır.

Gradyan iniş algoritması, parametre vektörünün hata fonksiyonunun bölgesel gradyanını ölçer ve parametre uzayında tam tersi istikamette yönelimi sağlar. Gradyon sıfır olduğu durumda, minimum değere ulaşılmış olur. Başlangıçta rastgele bir nokta seçilir ve her bir adımda öğrenme oranı kadar ilerlenir. Öğrenme oranı eğer çok fazla ise, parametre uzayında zıplamalar yapılarak bir önceki iterasyondan daha yüksekteki bir noktaya gidilmesi mümkündür. Eğer öğrenme oranı çok az ise de, minimum değere ulaşmak için çok fazla iterasyon gerekeceğinden, öğrenme süreci çok fazla uzayabilir.

Maliyet fonksiyonunun bölgesel minim noktasının, evrensel minimum olmadığı durumlar da mevcut olabilir. Bu durumlarda başlangıç değerinin seçimi büyük önem arz etmektedir. Seçilen bir başlangıç bölgesi için öğrenme süresi çok fazla sürebileceği gibi, evrensel minimuma ulaşmadan bölgesel minimumda iterasyonun sonlanması durumu söz konusudur.

4.3.1 Toplu Gradyan İnişi

Toplu gradyan inişi algoritması, her bir gradyan iniş adımında tüm eğitim kümesini kullanarak hesaplamaları yapmaktadır. Bu işlem eğitim süresini dolayısıyla uzatmaktadır.

Gradyan iniş algoritmasını gerçeklemek için, her bir modelin değişimiyle ilişkili olarak maliyet fonksiyonunun gradyanını hesaplamak gerekmektedir. Yani, model parametresi θ_j'nin belirli miktarda değişimleri için maliyet fonksiyonunun ne kadar değişeceğini hesaplamak gerekmektedir. Bunun için maliyet fonksiyonun kısmı türevini almak gerekmektedir. Ortalama kare hata fonksiyonu,

$$OKH(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T x^i - y^i)^2 \tag{4.6}$$

ile gösterilirse, maliyet fonksiyonun kısmı türevi,

$$\frac{\partial}{\partial \theta_j} OKH(\theta) = \frac{2}{m}\sum_{i=1}^{m}(\theta^T x^i - y^i)x_j^i \tag{4.7}$$

olur. Burada θ model parametre vektörünü, θ_j j. modelin parametresini ve m veri kümesindeki örnek sayısını göstermektedir.

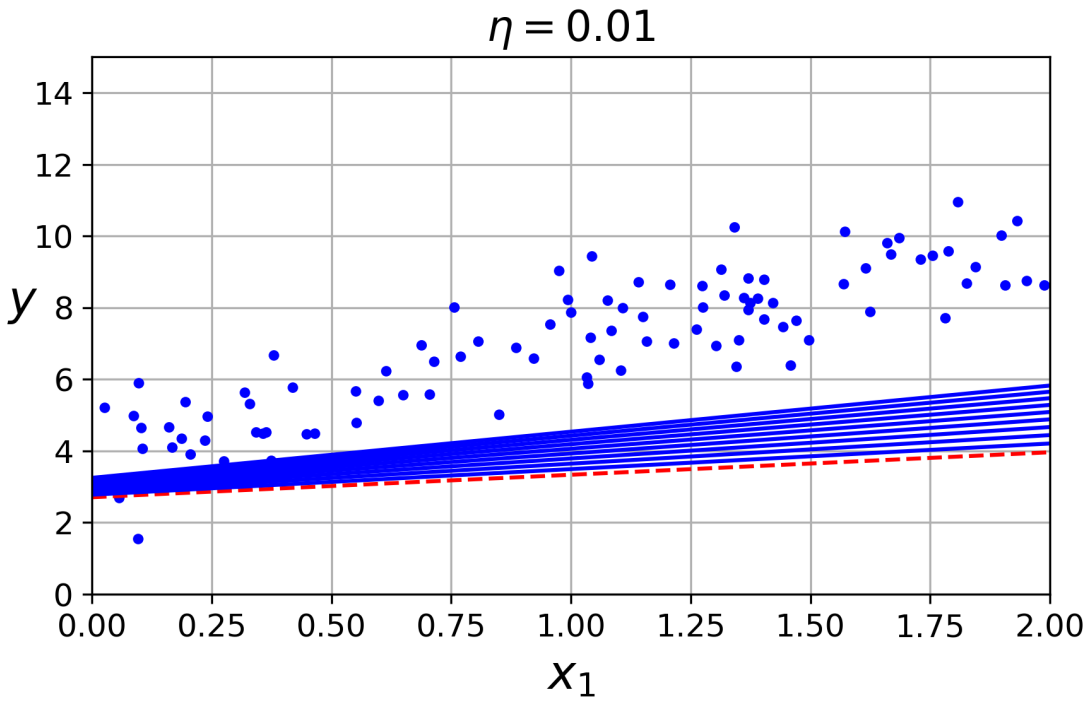
Kısmı türevi denklem 4.7'de gösterildiği gibi bireysel hesaplamak yerine, denklem 4.8 ile hepsi bir seferde hesaplanabilir.

$$\Gamma_\theta OKH(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} OKH(\theta) \\ \frac{\partial}{\partial \theta_1} OKH(\theta) \\ . \\ . \\ \frac{\partial}{\partial \theta_n} OKH(\theta) \end{pmatrix} = \frac{2}{m} X^T (X\theta - y) \tag{4.8}$$

Gradyan vektörü hesaplandıktan sonra, model parametre vektörü θ'dan çıkarılarak, maliyet fonksiyonun minimum noktasına ulaşılmaya çalışılmaktadır.

$$\theta^k = \theta - \eta \Gamma_\theta OKH(\theta) \tag{4.9}$$

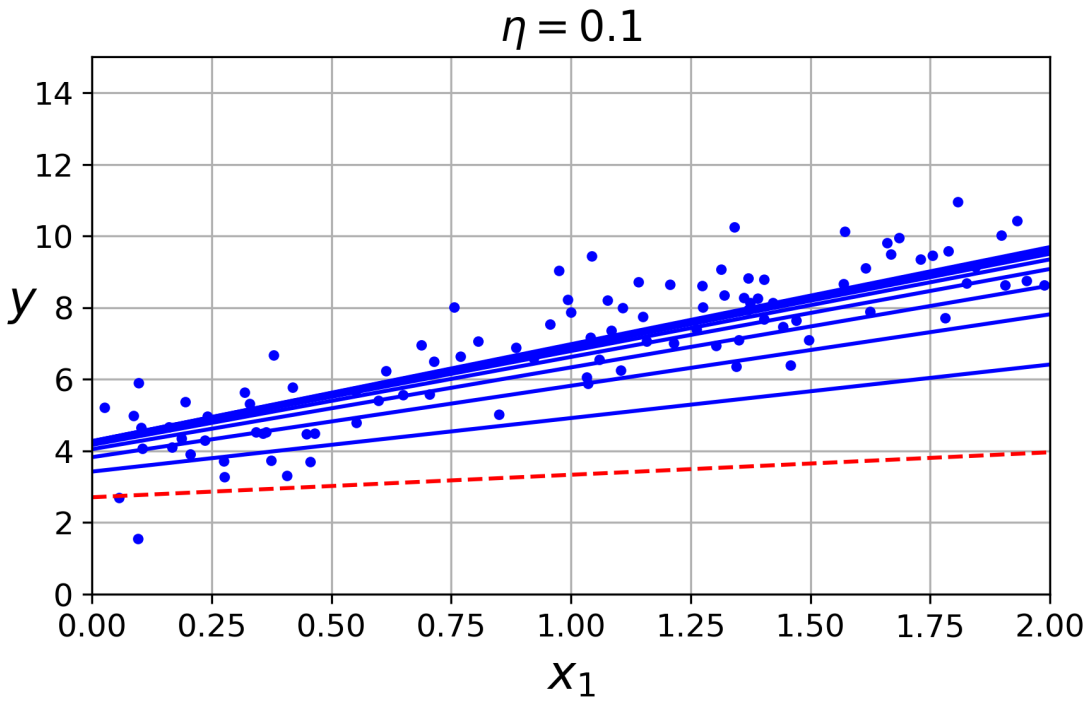
burada η öğrenme oranıdır.



Şekil 4.3 Öğrenme oranının 0.01 olduğu durumda gradyan iniş algoritması

Şekil 4.3' de kullanılan $\eta = 0.01$ öğrenme oranı çok yavaş olduğundan, gradyan iniş algoritmasının çözüme ulaşması çok fazla vakit almaktadır. Kırmızı çizgi ile tahminin

başlangıç noktası gösterilmektedir.



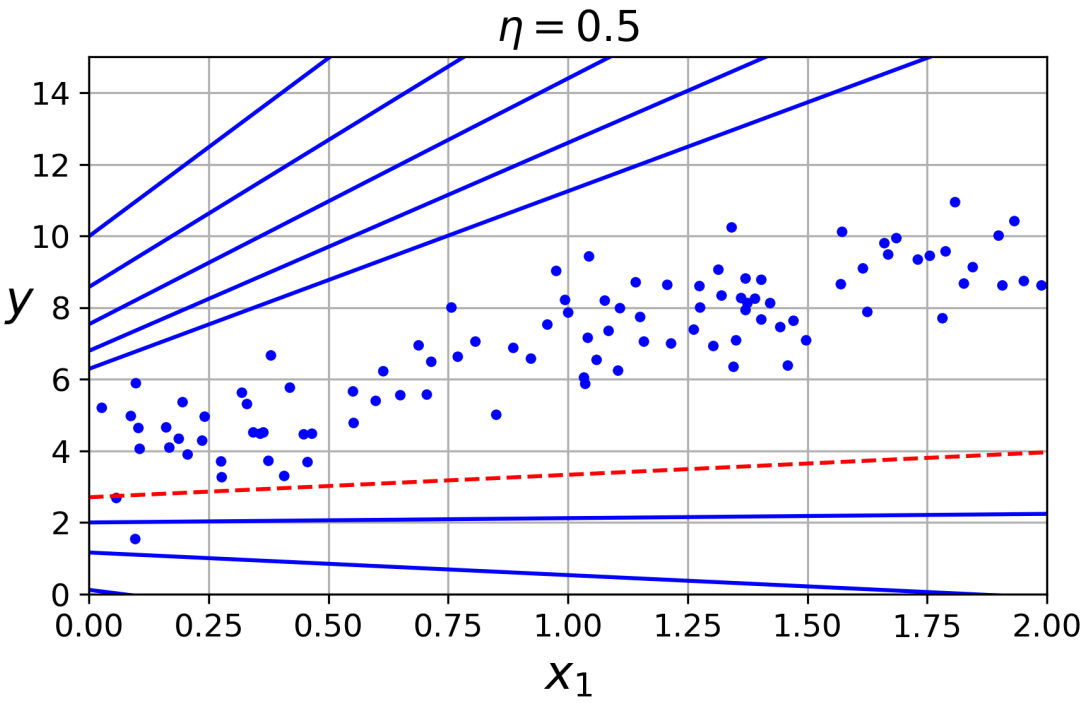
Şekil 4.4 Öğrenme oranının 0.1 olduğu durumda gradyan iniş algoritması

Şekil 4.4'de kullanılan $\eta = 0.5$ öğrenme oranı, verilen doğrusal regresyon problemi için uygun gözükmektedir.

Şekil 4.5'de kullanılan $\eta = 0.1$ öğrenme oranı çok yüksek olduğundan, daha önce bahsedildiği gibi, maliyet fonksiyonunda minimuma ulaşmaya çalışan algoritma attığı büyük adımlar sebebiyle çözüme yaklaşmak yerine uzaklaşmaktadır. Bu görseller öğrenme oranı seçiminin önemi ortaya koymaktadır.

4.3.2 Stokastik Gradyan İniş

Stokastik gradyan iniş algoritması, her bir adımda veri kümesinde rastgele olarak seçtiği örnekler üzerinden gradyan değerlerini hesaplamaktadır. Toplu gradyan algoritması, her adımda tüm veri kümesini kullandığından daha yavaş çalışabilmektedir. Stokastik gradyan algoritması, rastgelelik katarak bu problemi çözmüştür. Ancak bu rastgelelik, algoritmanın maliyet fonksiyonun minimum değeri etrafında dolaşmasına sebebiyet vermektedir. Bu sebeple algoritma, bir limit değerin altında yitim değerine ulaştığında sonlandırılmalıdır. Algoritma sonlandırıldığına, elde edilen parametrelerin optimal değil, kriterleri sağlayan değerler olduğu unutulmamalıdır.

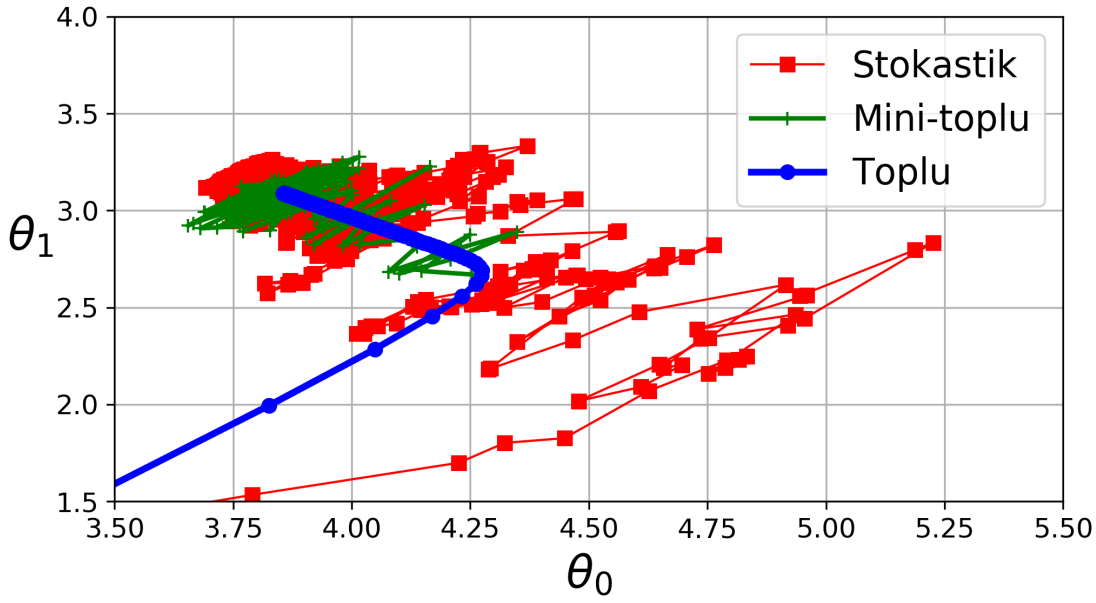


Şekil 4.5 Öğrenme oranının 0.5 olduğu durumda gradyan iniş algoritması

4.3.3 Mini-Toplu Gradyan İniş

Mini-toplu gradyan iniş algoritması toplu gradyan iniş algoritmasının aksine, veri kümesinin tamamını işleme tabi tutmak yerine, onu rastgele küçük kümelere bölerek hesaplamalar yapmaktadır. Algoritmanın, stokastik gradyan iniş algoritmasından farkı ise, her adımda rastgele seçilen bir veri kümesini işlemek yerine, parça parça veri kümelerini işleme tabi tutmasıdır. En büyük avantajı, veri kümesini alt kümelere ayırdığından paralel hesaplamalarda, özellikle GPU kullanılarak yapılan çalışmalarda, matris işlemlerinin donanım optimizasyonunu sağlayarak performans iyileştirmeleri sunmasıdır.

Şekil 4.6 iki parametreye sahip bir doğrusal regresyon probleminde, maliyet fonksiyonun minimumuna inmeyi hedefleyen, stokastik , mini-toplu ve toplu gradyan iniş algoritmalarının parametre uzayındaki değişimleri gösterilmektedir. Stokastik ve mini-toplu algoritmalar minimum noktasında git gel yaparken, toplu gradyan iniş algoritması minimum noktasında durmaktadır. Ancak eğitim sırasında her iterasyonda tüm veri kümesini kullandığından, işlem süresi diğeri iki algoritmadan daha uzun olmaktadır. Stokastik algoritmasının, rastgelelik içermesi parametre uzayındaki zıplamalarına sebebiyet vermektedir.



Şekil 4.6 Farklı gradyan iniş algoritmalarının karşılaştırılması

4.4 Çok Katmanlı Perseptron

Çok katlı perseptron , bir adet giriş katmanından, bir adet çıkış katmanından ve bir veya birden fazla olabilen gizli katmanlardan oluşmaktadır. Eğer gizli katman sayısı 2 den fazla ise bu ağ derin ağ olarak kabul edilmektedir.

4.4.1 Geri Yayılım

Rumelhart, Hinton ve Williams'ın 1986 yayınladığı geri yayılım eğitim algoritması, ÇKP için yeni ufuklar açmıştır [46]. Bu başarılı algoritma hala günümüzde kullanılmaktadır.

Geri yayılım algoritması temelinde, gradyanları otomatik olarak hesaplamak için verimli teknikler kullanan bir gradyan iniş yöntemidir. Ağdaki her bir ağırlığın ilgili hatasının derecesi, ileri ve geri olmak üzere yapılan iki geçiş sayesinde hesaplanabilmektedir. Algoritma öncelikle her bir eğitim kümesi için bir tahminde bulunur, bu ileri yönde olan geçiştir. Ardından hata hesaplanır ve her bir katmandan gelen hata katkısını ölçmek için geri yönde geçiş yapılır. Bu işlem, aynı gradyan iniş algoritmasında olduğu gibi ya bir performans kriteri sağlanana kadar ya da maliyet fonksiyonunun minimum değerine ulaşılana kadar devam eder.

4.5 Kaybolan/Patlayan Eğim Problemi

Gradyanlar, ağ derinleştikçe ve derin ağlara ilerledikçe giderek küçülebilmektedir. Bu küçülme, aşağı katmanlarda gradyan iniş algoritmasının ağırlıkları neredeyse hiç değişmeden bırakmasına ve dolayısıyla eğitimin kabul edilebilir bir çözüme yakınsayamamasına sebebiyet verir. Literatürde bu problem kaybolan eğim problemi olarak bilinmektedir.

Gradyanların, aşağı katmanlara ilerledikçe giderek büyümesi ve bir çok katmanda çok büyük ağırlık güncellemelerinin ortaya çıkması algoritmayı saptırabilmektedir. Bu problem ise literatürde patlayan eğim problemi olarak bilinmektedir. Patlayan eğim problemi, çoğunlukla RNN'de görülebilmektedir.

Glorot ve Bengio tarafından 2010 yılında yayınlanan makalede, problemin sebebinin lojistik sigmoid aktivasyon fonksiyonu gibi popüler aktivasyon fonksiyonlarının ve bazı ağırlık başlatma tekniklerinin olduğu ispatlanmıştır. [47]. Ağda ilerledikçe, her bir katmanda varyans artmasına rağmen, son katmanlarda aktivasyon fonksiyonunun doyuma ulaşması gradyanın neredeyse sıfır olmasına sebebiyet vermektedir. Bu sebeple derin ağlarda lojistik sigmoid aktivasyon fonksiyonunun yerine hiperbolik tanjant aktivasyon fonksiyonu daha başarılı sonuçlar vermektedir.

4.5.1 Glorot Başlangıcı

Glorot ve Bengio, ağda ileri ve geri yönde yayılım yaparken sinyalin hem kaybolmaması hem de patlamaması adına, her bir katmanın giriş varyansının, çıkış varyansına eşit olması gerektiğini savunmaktadırlar. Bunun, katmanın eşit sayıda girişe (fan_g) ve nörona (fan_c) sahip olmadıkça garanti edilmesinin imkansız olmasından dolayı, Glorot Başlangıcı adı verilen ve pratikte iyi sonuçlar veren algoritma tanıtılmıştır [47].

Glorot başlangıcı, her bir katmanın bağlantı ağırlıklarının ya sıfır ortalama ve denklem 4.10'da verilen varyans ile birlikte normal dağılım olmasını ya da denklem 4.11'de verilen r için, $-r$, $+r$ aralığında tek düze dağılım olmasını önermektedir.

$$\sigma^2 = 1/fan_{ort} \tag{4.10}$$

$$r = \sqrt{3/fan_{ort}} \tag{4.11}$$

burada,

$$fan_{ort} = (fan_g + fan_c)/2 \tag{4.12}$$

olmaktadır.

4.5.2 Toplu Normalleştirme

Ioffe ve Szegedy tarafından 2015 yılında kaybolan/patlayan eğim problemi için Toplu Normalleştirme adında bir teknik yayınlanmıştır [48]. TN, her bir gizli katmanın aktivasyon fonksiyonu öncesine ve sonrasına bir operatör eklemektedir. Bu operatör ile her bir giriş ilk olarak sıfır merkezine çekilerek normalize edilmekte ve sonrasında iki yeni parametre ile ölçeklenmekte ve kaydırılmaktadır. Bu iki ölçekleme ve kaydırma parametreleri eğitim sırasında optimize edilerek, tekniğin herhangi bir katmandaki veri üzerinde ne kadar oynama yapacağını kendi karar vermesini sağlamaktadır.

Toplu normalleştirme tekniğinde, giriş vektörünün ortalaması,

$$\mu_B = \frac{1}{m_B}\sum_{i=1}^{m_B} x^{(i)} \tag{4.13}$$

ile hesaplanır. m_B örnek numarasını vermektedir. Giriş vektörünün varyansı,

$$\sigma_B^2 = \frac{1}{m_B}\sum_{i=1}^{m_B}(x^{(i)} - \mu_B)^2 \tag{4.14}$$

ve sıfır merkezli ve normalleştirilmiş girişlerin vektörü,

$$\widehat{x}^{(i)} = \frac{x^{(i)} - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{4.15}$$

ile hesaplanır. Burada ϵ sıfırla bölünme sorununu çözmek için kullanılan çok küçük bir sayıdır ve genellikle 10^{-5} seçilir.

TN işleminin çıktısı,

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \tag{4.16}$$

olarak hesaplanır. Burada γ katmanın çıktısının ölçekleme parametresi ve β kaydırma parametresidir.

4.6 Aşırı Öğrenme

Derin ağlarda optimize edilmesi gereken on binlerce, bazı derin ağlarda hatta milyonlarca parametre bulunmaktadır. Bu kadar fazla parametre beraberinde ağda çok fazla oranda serbestlik miktarı getirmektedir. Aşırı öğrenme, ağın verileri öğrenmesinden ziyade ezberlemesi durumudur. Veriler eğitim ve test olarak ayrıldığında, eğitim kümesinin yitim değeri, test kümesinin yitim değerinin çok üstünde ise aşırı öğrenme durumu söz konusudur.

4.6.1 Seyreltme

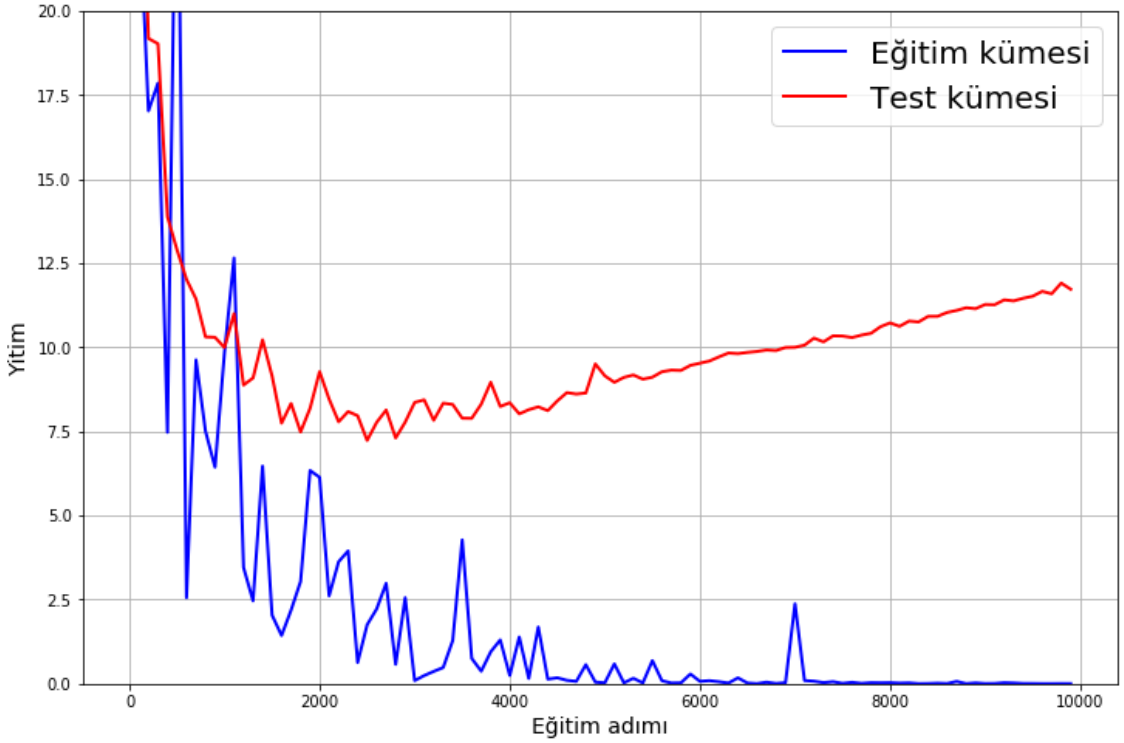
Derin ağlarda aşırı öğrenmenin önün geçmek için kullanılan en yaygın yöntemlerden biri seyreltmedir. Hinton ve diğerleri tarafından 2012 yılında önerilen tekniğin ayrıntıları [49], 2014 yılında Srivasta ve diğerleri tarafından yayınlanarak başarısı kantılanmıştır [50].

Seyreltme her bir eğitim adımında, çıkış nöronları hariç ve giriş nöronları dahil olmak üzere tüm nöronların ρ olasılığıyla yok sayılması işlemidir. Bir eğitim adımında yok sayılan bir nöron bir sonraki aşamada tekrar aktif olabilir. Burada ρ seyreltme oranıdır ve kullanıcı tarafından seçilir. Her bir katman için farklı bir seyreltme oranı seçilebilmektedir. Eğitim bittikten sonra test aşamasında seyreltme işlemi uygulanmaz.

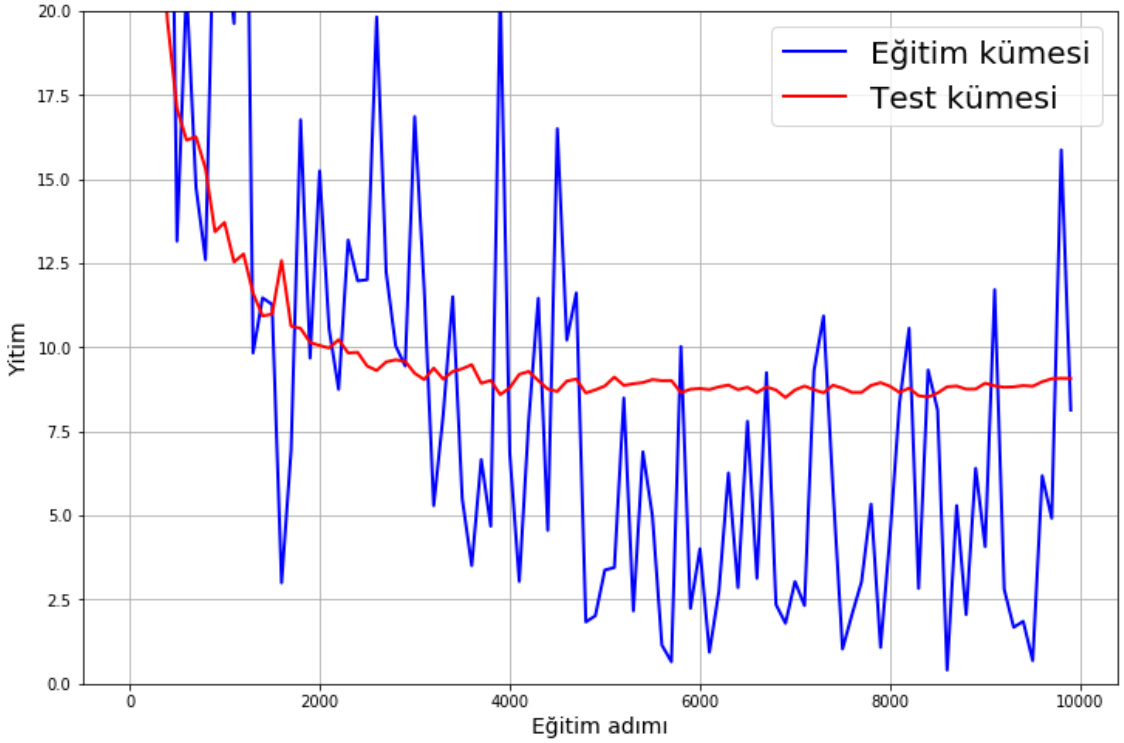
Şekil 4.7'de MNIST veri kümesi uygulanarak el yazısı rakam tanıma uygulaması için seyreltme uygulanmamış derin ağın, yitim fonksiyonları çıktısı eğitim ve test kümesi için ayrı ayrı verilmiştir. Yaklaşık 5000. eğitim adımından sonra aşırı öğrenmenin başladığı açıkça görülmektedir. Şekil 4.8'de verilen seyreltme uygulanmış sistem için ise 10000. eğitim adımında bile aşırı öğrenmeye rastlanmamaktadır. Burada seyreltme uygulanan sistemde, rastgele ihmal edilen nöronlar nedeniyle eğitim kümesindeki dengesizliğin arttığı gözlenmektedir.

4.7 Otokodlayıcı

Otokodlayıcılar, ilk olarak Rumelhart ve diğerleri tarafından 1986'da tanıtılmış olup giriş nöron sayısı ile çıkış nöron sayısına birbirine eşit olan denetimsiz ÇKP ağlarıdırlar [51]. Otokodlayıcıların asıl amacı, PCA'ın yaptığı gibi giriş verisinin daha az boyutta



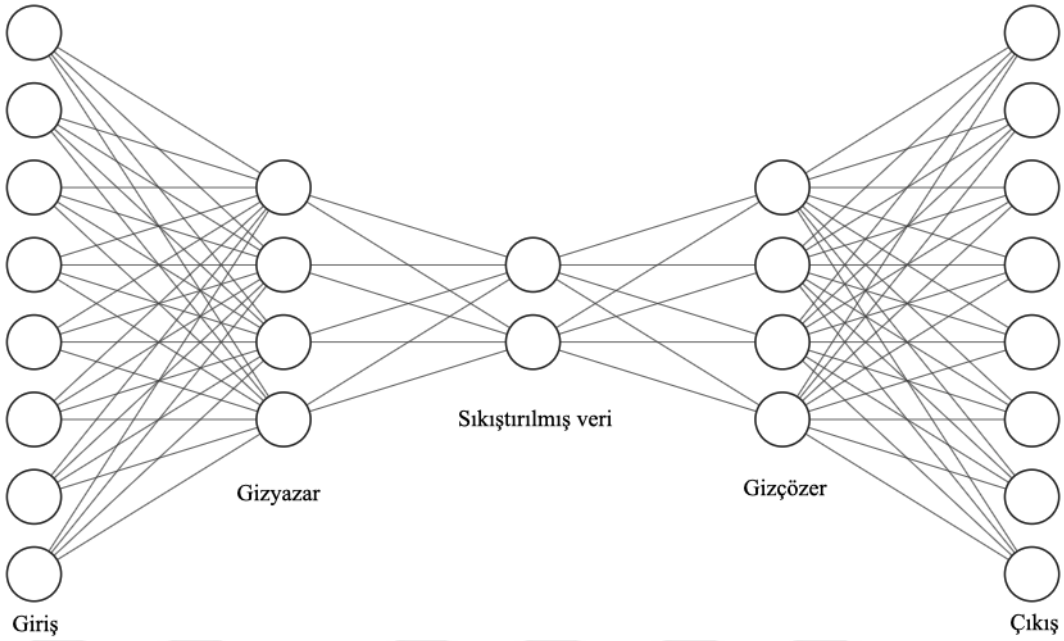
Şekil 4.7 Seyreltme uygulanmamış sistemin eğitim ve test verilerinin yitim değerleri



Şekil 4.8 Seyreltme uygulanmış sistemin eğitim ve test verilerinin yitim değerleri

temsil edilen, mümkün mertebe benzerini çıktı olarak vermektir.

Otokodlayıcılar, gizyazar ve gizçözer adında iki yapıdan oluşurlar. Genel ağ yapılarına



Şekil 4.9 Otokodlayıcıların genel ağ yapısına örnek bir ağ

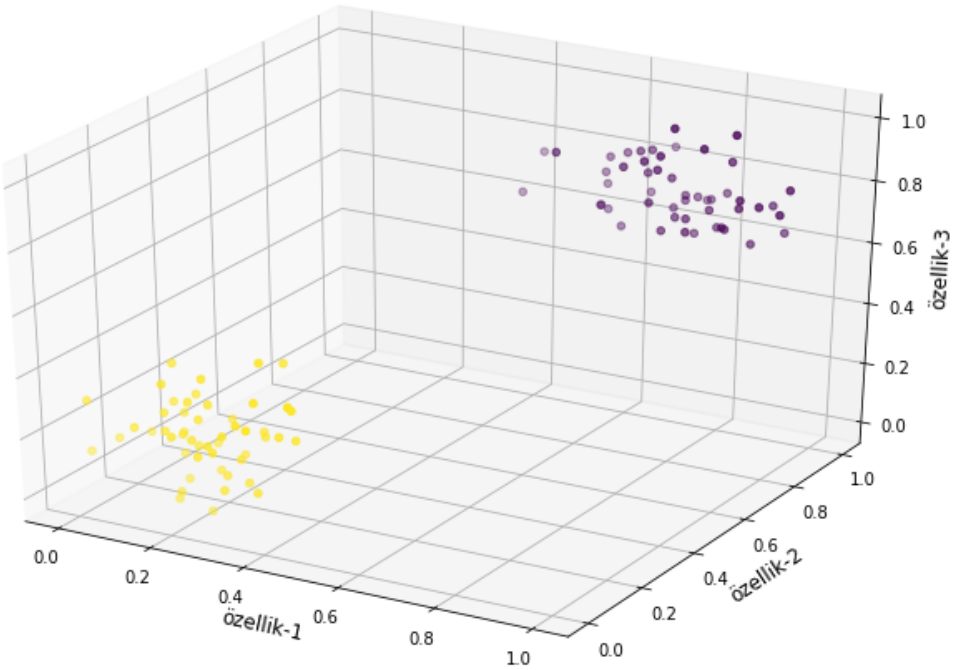
örnek bir ağ şekil 4.9'da verilmiştir. Giriş verisi ilk olarak gizyazar ağlarına gelerek sıkıştırılır ve ardından gizçözer ağlarında ise giriş verisiyle aynı boyutta olmak üzere çıkış verisini oluşturmak adına veriyi yeniden inşa eder. Bu işlem sırasında boyut indirgeme sağlanarak, veri daha az özellik ile temsil edilir [52].

Otokodlayıcılar, boyut indirgeme yaparak verilerin içindeki önemli özellikleri ortaya çıkarırlar. Ses ve görüntü verilerindeki gürültüleri temizlemekte sık sık otokodlayıcılar kullanılmaktadır. Otokodlayıcıların PCA'dan farkı, PCA doğrusal boyut azaltma yöntemi iken, otokodlayıcılar doğrusal olmayan boyut azaltma işlemidirler. Şekil 4.10'da örnek olarak verilen 3 boyutlu bir veri kümesinin, otokodlayıcı uygulanması sonrasında boyut indirgenmiş versiyonunun dağılımı şekil 4.11'de gösterilmiştir.

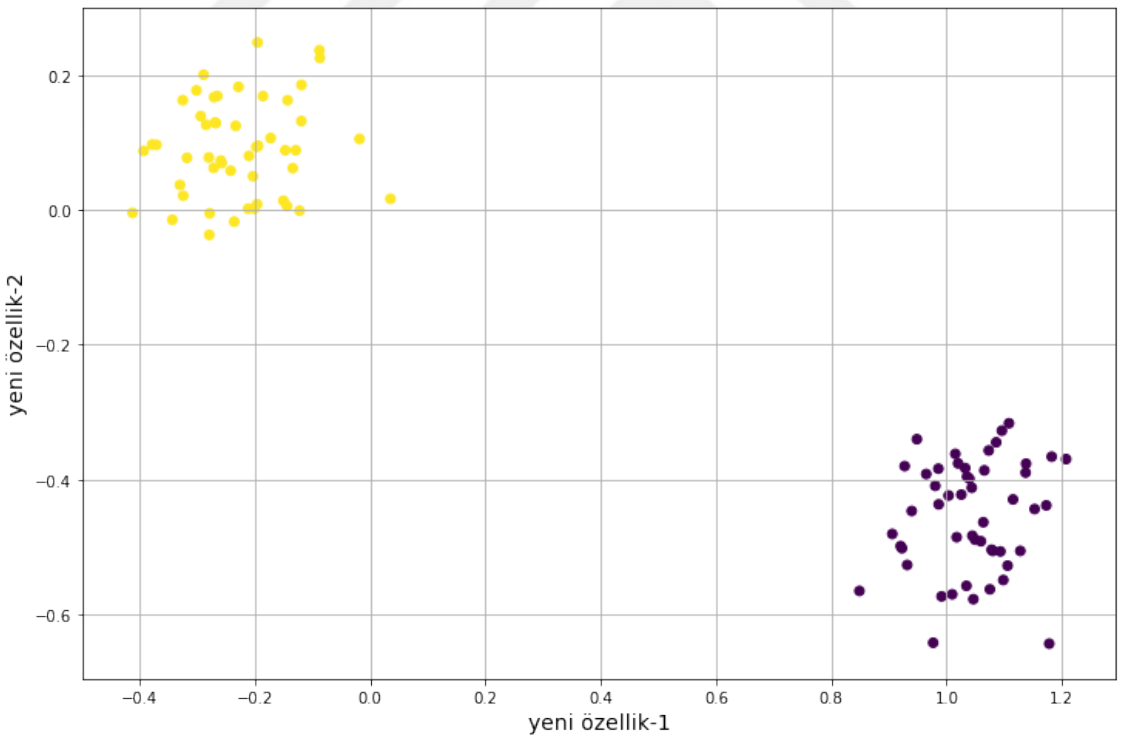
4.8 Derin Ağlar ile Konumlandırma Uygulaması

Dış mekan konumlandırma problemleri çoğunlukla GPS sensörleri sayesinde yüksek doğrulukta çözülebilmektedir. Ancak iç mekan konumlandırmaları, GPS sensörlerinin iç mekanlarda kullanılamaması sebebiyle hala çözüm bekleyen bir problemdir.

Birçok uygulama, servislerinin kullanılması için kullanıcılarının konum bilgilerine ihtiyaç duymaktadır. Bu ihtiyaç, kullanıcı konumlandırma probleminin yoğun araştırılan konular arasında yer almasını sağlamaktadır. İç mekanlarda yaşanan GPS sensörü bilgisinin yetersizliği, araştırmacıları problemin çözümü için kablosuz yerel alan ağlarını ve cep telefonlarını kullanmaya yönlendirmiştir. Günümüzde



Şekil 4.10 Veri kümesinin 3 boyutta gösterimi



Şekil 4.11 Veri kümesinin otokodlayıcı sonrası gösterimi

kablosuz ağların her yerde bulunması, ulaşılabilirliği bunu desteklemektedir. Ayrıca cep telefonlarının günlük hayatımızdaki yeri göz önüne alındığında, cep telefonları ile kullanıcıların aynı konumda olduğu kabulü kolaylıkla yapılabilmektedir.

Kablosuz ağ parmak izi tabanlı konumlandırma sistemleri, Alınan Sinyal Gücü Göstergesi değerine dayanmaktadır. Yöntem yaygın olarak üç aşamadan oluşmaktadır [53]:

1. Kalibrasyon: Kullanıcıların algılanması gereken alanın radyo haritası oluşturulur.
2. Operasyon: Kullanıcı, bulunduğu yerden tespit edilebilecek tüm görünür erişim noktalarının sinyal gücünü alır ve test örnekleri oluşturulur.
3. Karşılaştırma: Bir önceki aşamada oluşturulan test örnekleri, radyo haritasının eğitim örnekleriyle karşılaştırılmak üzere sunucuya gönderilir ve en benzer örnek ile ilişkilendirilen konum, kullanıcı konum tahmini olarak belirlenir.

İç mekanda kablosuz ağlar ile kullanıcı konumlandırma uygulamasının, ekstra donanım gerektirmemesi ve çok yüksek oranda hali hazırda binalarda kurulu bulunması sebebiyle herhangi bir kurulum maliyetinin olmaması, problemin çözümü için kullanılması adına önemli avantajlarındandır. Ancak kullanıcıların hareketleri, cep telefonlarını taşıma şekilleri ASGG değerini etkilemekte, bu da problemin çözümünü zorlaştırmaktadır [54]. Kablosuz ağların tasarımının amaçları arasında, konumlandırma probleminin çözümünün olmadığı da unutulmamalıdır.

4.8.1 Veri Kümesinin Tanıtımı

Bu uygulama için, Sospedra, Montoliu ve diğerleri tarafından 2014 yılında tanıtılan, "UJIIndoorLoc Veri Kümesi" olarak bilinen veriler kullanılmıştır [55]. Bu veri kümesi, ilgili alan için halka açık olarak yayınlanan ilk ve en büyük veri kümesi olma özelliğini taşımaktadır.

Veri kümesinin detayları aşağıda listelenmiştir [55]:

- 4 veya 5 katlı 3 farklı binayı içermekte ve toplam 108703m^2'lik bir alanı kapsamaktadır.
- Farklı 933 referans noktasını içermektedir.
- 933 farklı referans noktası için toplam 21049 tane konum bilgisi bulunmaktadır. Bunlardan 19938 tanesi eğitim kümesi, 1111 tanesi test kümesi olarak ayrılmıştır.
- 520 tane kablosuz erişim noktası mevcuttur.

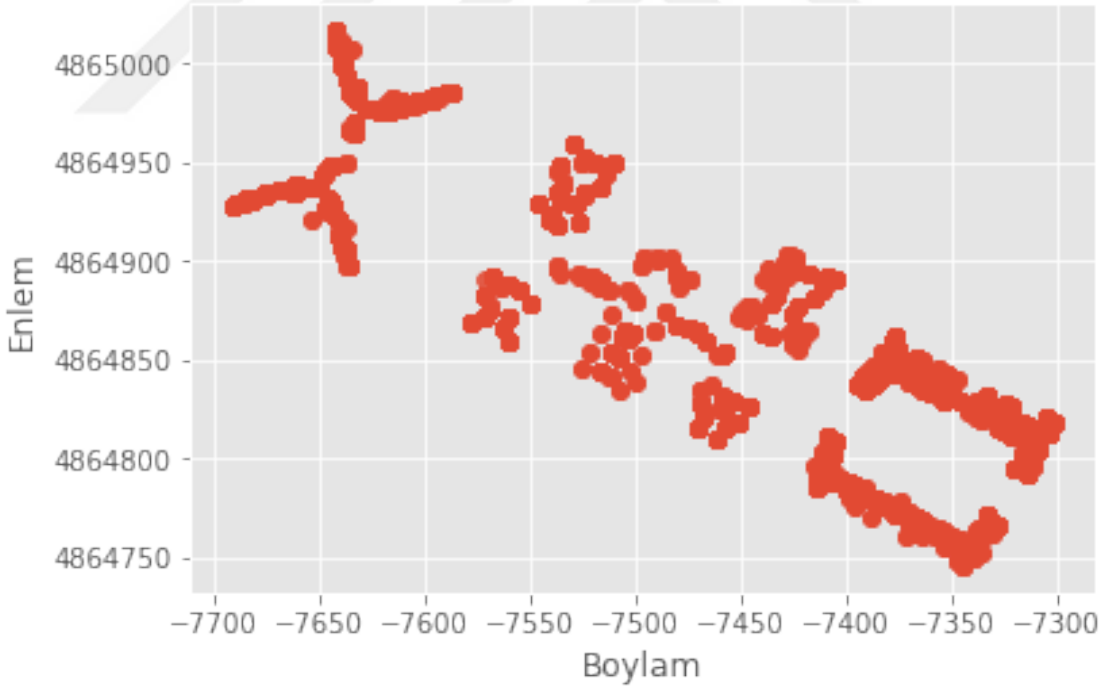
- 20'den fazla kullanıcının taşıdığı toplamda 25 farklı telefon modelinden elde edilen verilerden elde edilmiştir.

Veri kümesinde bulunan her bir 21049 kayıt 529 adet nümerik bilgi içermektedir. Bu nümerik değerler hakkında detaylı bilgiler Tablo 4.1'de verilmiştir.

Tablo 4.1 Veri kümesinin detayları

001 - 520	ASSG değerleri
521 - 523	Örneklerin gerçek dünyadaki koordinatları
524	Apartman numarası
525	Boşluk numarası
526	Boşluk numarası ile ilişkili pozisyonlar
527	Kullanıcı numarası
528	Telefon numarası
529	Zaman damgası

Şekil 4.12' de veri kümesinin gerçek dünyadaki koordinat bilgilerine göre dağılımları verilmiştir.



Şekil 4.12 Veri kümesinin enlem ve boylam dağılımları

4.8.2 Uygulama

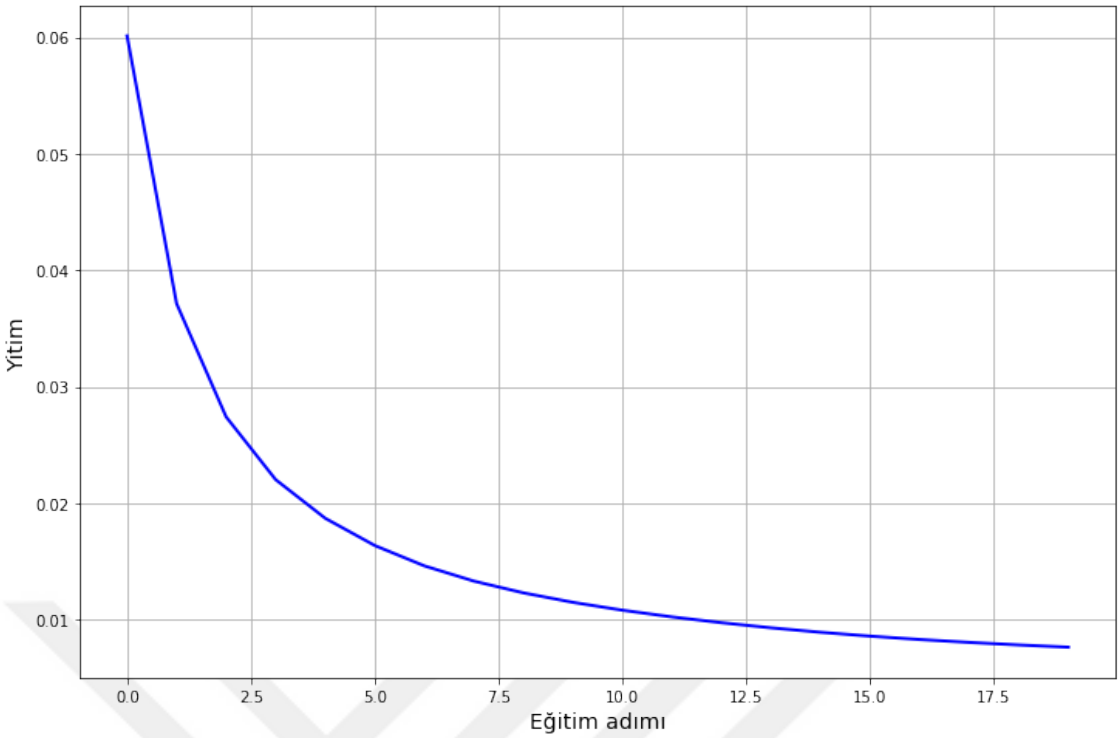
Kablosuz ağlar ile iç mekan konumlandırma probleminin çözümü üzerinde bir çok makine öğrenmesi tekniği çalışılmış ve kullanılmıştır [56–59]. Makine öğrenmesi

teknikleri bu konuda yüksek doğrulukta çözümler vermesine rağmen fazladan ayarlama ve filtreleme işlemini gerektirmektedir. Derin ağlar ile birlikte kullanacak otokodlayıcı bu çabayı azaltacaktır [60]. Otokodlayıcı sonrası sınıflandırma için kullanılacak derin ağ ile birlikte de kullanıcının hangi binada ve katta olacağının tahmini yapılacaktır.

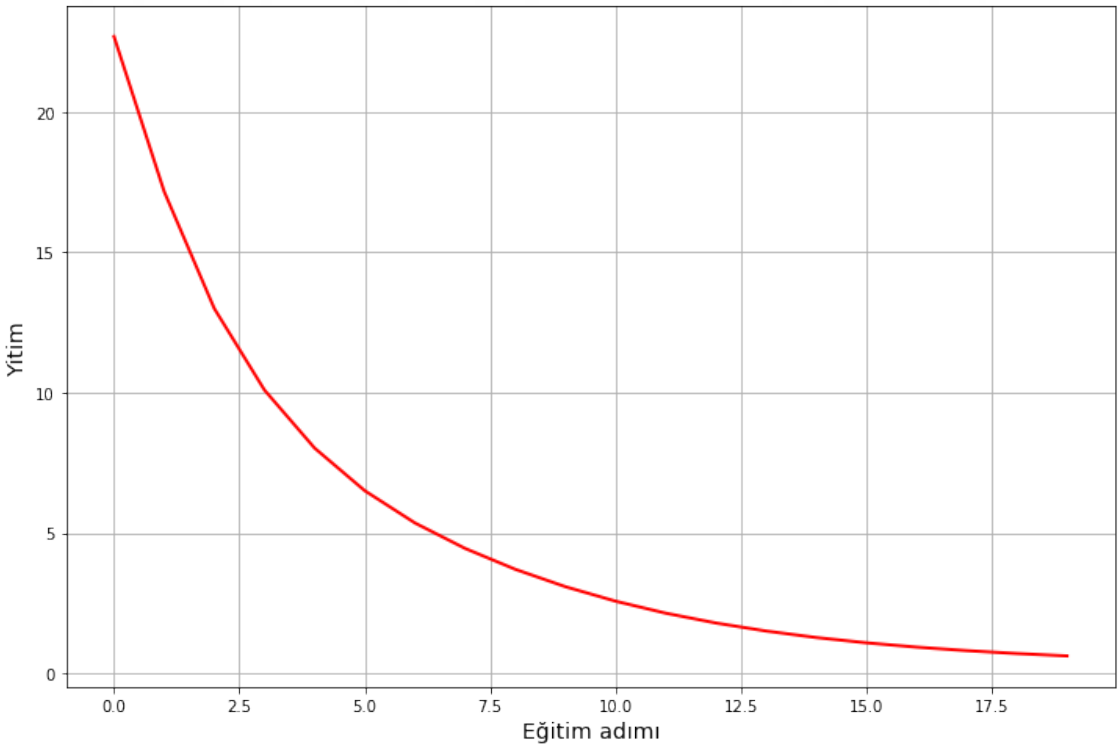
Kablosuz ağlar ile iç mekan konumlandırma uygulamasının derin öğrenme temelli çözümü için Tensorflow kütüphanesi kullanılmıştır. Kullanılan otokodlayıcı içeren sınıflandırma derin öğrenme ağı ile katların ve binaların tahmini yapılmıştır. 520 boyutlu ASGG verisini içeren veri kümesi kodlayıcı yardımıyla kademeli olarak 256,128 ve 64 boyutuna indirilmiş ve ardından gizçözer ile 64,128,256 ve son olarak giriş boyutuyla aynı değere getirilmiştir.

Aktivasyon fonksiyonu çıkış katmanı hariç hiperbolik tanjant ve iyileştirici olarak Adam iyileştirici [61] seçilmiştir. Öğrenme oranı 0.0001 olup veriler ağa 10'lu yığınlar şeklinde verilmiştir. Otokodlayıcı ağı eğitildikten sonra gizçözer ve çıkış katmanı ihmal edilerek, orta katmanda elde edilen sıkıştırılmış veri sınıflandırma ağına giriş olarak verilmiştir. Sınıflandırma ağında 128 boyutlu iki adet ara katman bulunmaktadır ve çıkış katmanının aktivasyon fonksiyonu olarak eşiksiz en büyük fonksiyonu (softmax function) seçilmiştir.

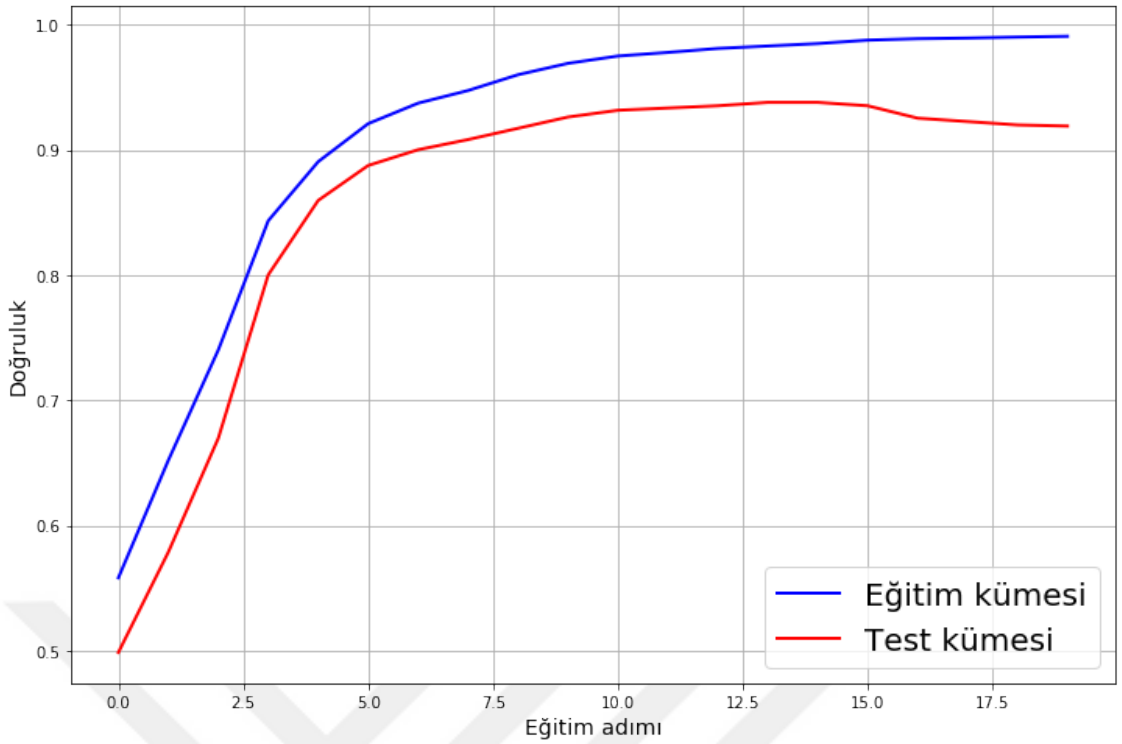
Hem otokodlayıcı hem de sınıflandırma ağında, ağ 20 dönemden (epochs) oluşmaktadır. Ayrıca aşırı öğrenmeyi engellemek adına sınıflandırma ağına seyreltme işlemi uygulanmıştır. Otokodlayıcı ağının yitim değeri Şekil 4.13'te, sınıflandırma ağının yitim değeri ise Şekil 4.14'de verilmiştir. Tüm ağın nihai sonucu için eğitim ve test kümelerinin ayrı ayrı doğruluk oranları da Şekil 4.15'te verilmiştir. Test kümesi üzerinde ağın başarı oranı %92.61'dir.



Şekil 4.13 Otokodlayıcı ağı için yitim değerinin eğitim sırasında değişimi



Şekil 4.14 Sınıflandırma ağı için yitim değerinin eğitim sırasında değişimi



Şekil 4.15 Eğitim ve test kümesi için doğruluk değerlerinin eğitim sırasında değişimi

5
Sonuç ve Öneriler

Gelişmiş ve çeşitli algılayıcılar ile donatılan robotlar, sahip oldukları avantajlardan dolayı hem sivil hem de askeri birçok görevin temelini oluşturmaktadırlar. Robotların gerçekleştirdiği görevlerin temelini ise de, bilinmeyen çevrenin haritasını çıkarmak ve çıkarılan haritada robotun kendi konumunun tespitini yapması oluşturmaktadır.

Eşzamanlı olarak robotun kendini konumlandırması sırasında, algılayıcıları yardımıyla çevredeki dünyanın haritasının çıkarılması olan SLAM probleminde, Kalman Filtresi, Genişletilmiş Kalman Filtresi ve Parçacık Filtresi sıklıkla merkezi tahmin edici olarak kullanılmaktadır.

Parçacık filtreleri, herhangi bir durum-uzay modeline uygulanabilen ardışık Monte Carlo yöntemleridirler. Kalman filtresinin aksine, parçacık filtrelerinde hareket ve gözlem modelinin doğrusal olması gerekmemektedir ve gürültünün Gauss dağılımı yaptığı varsayımına ihtiyaç yoktur. Bu nedenle parçacık filtreleri, kolaylıkla herhangi bir durum-uzay modeline uygulanabilmektedir. Ancak durum boyutu yüksek olduğu senaryolarda, parçacık filtreleri yüksek performans göstermezler. Yüksek boyutlu sistemler için görülen bu problemin çözümü adına EDH ve LEDH filtreleri tanıtılmıştır.

EDH filtresinde tüm parçacıklar için ortak göç parametreleri hesaplanırken, LEDH filtresinde her bir parçacık için tek tek göç parametreleri hesaplanmaktadır. Makine öğrenmesi tekniklerinden K-Ortalamalar algoritması yardımıyla güçlendirilen CEDH filtresi ise parçacıkları hataları üzerinden kümeleyerek, kümeler için ortak göç parametrelerini hesaplamıştır. Bu sayede hesaplama maliyetini önemli ölçüde düşürmektedir. Ayrıca CEDH filtresinde, en çok hataya sahip küme elenerek performans iyileştirmeleri sunulmaktadır.

CEDH filtresinin başarısı, çok boyutlu konum takibi uygulaması ile kanıtlanmıştır. İleriki çalışmalarda, problemin tespiti ve gürültünün tanınması ile CEDH filtresi desteklenerek küme eleme aşaması güçlendirilecektir.

Tezin son bölümünde iç mekan uygulaması için kablosuz ağlar ile konumlandırma

problemi incelenmiştir. Derin öğrenme ağlarının tanıtımı yapılarak, problemlerine ve literatürdeki çözümlerine yer verilmiştir. Derin ağların, ilerleyen veri ulaşılabilirliği ve artan işlemci gücü ile birlikte her geçen gün çözüm olduğu problem sayısı artmaktadır. Söz konusu uygulama derin ağlar ile çözülmüş ve yüksek bir başarı oranı elde edilmiştir. İleriki çalışmalarda, başka ağ yapıları ve kombinasyonları denenerek uygulamanın performansını arttırmak için çalışmalar yapılacaktır.

A
Ek

A.1 Bir robotun düzlemdeki hareketi

$$R = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} t \\ \theta \end{bmatrix} \tag{A.1}$$

A.2 Rotasyon Matrisi

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{A.2}$$

A.3 Polar Koordinat Sistemi

$$\widehat{\rho} = \begin{bmatrix} \rho \\ \phi \end{bmatrix} = polar(\rho) = \begin{bmatrix} \sqrt{x^2 + y^2} \\ \arctan(y, x) \end{bmatrix} \tag{A.3}$$

A.4 Olasılık yoğunluk Fonksiyonu

$$\xi(x) \triangleq \lim_{dx \to 0} \frac{P(x \leq X + dx)}{dx} \tag{A.4}$$

A.5 Gauss dağılımı

$$\aleph(x, \bar{x}, P) = \frac{1}{\sqrt{(2\pi)^n |P|}} \exp(-\frac{1}{2}(x - \bar{x})^T P^{-1}(x - \bar{x})) \tag{A.5}$$

A.5.1 n-sigma Elipsoidi

Tablo A.1 Rastgele bir değişkenin n-sigma elipsoidi içinde olma olasılığı

	1σ	2σ	3σ	4σ
2D	39,4%	86,5%	98,9%	99,97%
3D	19,9%	73,9%	97,1%	99,89%

Referanslar

[1] T. Hashimoto and H. Kobayashi, "Study on natural head motion in waiting state with receptionist robot saya that has human-like appearance," in *2009 IEEE Workshop on Robotic Intelligence in Informationally Structured Space*, 2009, pp. 93–98. DOI: 10.1109/RIISS.2009.4937912.

[2] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001, ISSN: 1042-296X. DOI: 10.1109/70.938381.

[3] P. J. Hargrave, "A tutorial introduction to kalman filtering," in *IEE Colloquium on Kalman Filters: Introduction, Applications and Future Developments*, 1989, pp. 1/1–1/6.

[4] S. Yang and M. Baum, "Extended kalman filter for extended object tracking," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4386–4390. DOI: 10.1109/ICASSP.2017.7952985.

[5] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002, ISSN: 1053-587X. DOI: 10.1109/78.978374.

[6] P. Bunch and S. Godsill, "Approximations of the optimal ımportance density using gaussian particle flow ımportance sampling," *Journal of the American Statistical Association*, vol. 111, no. 514, pp. 748–762, 2016. DOI: 10.1080/01621459.2015.1038387.

[7] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000, ISSN: 1573-1375. DOI: 10.1023/A:1008935410038. [Online]. Available: https://doi.org/10.1023/A:1008935410038.

[8] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, 1999, ISSN: 01621459. [Online]. Available: http://www.jstor.org/stable/2670179.

[9] A. Doucet, N. d. Freitas, K. P. Murphy, and S. J. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, ser. UAI '00, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 176–183, ISBN: 1-55860-709-9. [Online]. Available: http://dl.acm.org/citation.cfm?id=647234.720075.

[10] A. Beskos, D. Crisan, and A. Jasra, "On the stability of sequential monte carlo methods in high dimensions," *Ann. Appl. Probab.*, vol. 24, no. 4, pp. 1396–1445, Aug. 2014. DOI: 10.1214/13-AAP951. [Online]. Available: https://doi.org/10.1214/13-AAP951.

[11] P. Bui Quang, C. Musso, and F. Le Gland, "An insight into the issue of dimensionality in particle filtering," in *2010 13th International Conference on Information Fusion*, Jul. 2010, pp. 1–8. DOI: 10.1109/ICIF.2010.5712050.

[12] M. Ades and P. J. van Leeuwen, "The equivalent-weights particle filter in a high-dimensional system," *Quarterly Journal of the Royal Meteorological Society*, vol. 141, pp. 484–503, Jan. 2015. DOI: 10.1002/qj.2370.

[13] P. M. Djuric, T. Lu, and M. F. Bugallo, "Multiple particle filtering," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, vol. 3, Apr. 2007. DOI: 10.1109/ICASSP.2007.367053.

[14] A. Beskos, D. Crisan, A. Jasra, K. Kamatani, and Y. Zhou, "A stable particle filter for a class of high-dimensional state-space models," *Advances in Applied Probability*, vol. 49, no. 1, pp. 24–48, 2017. DOI: 10.1017/apr.2016.77.

[15] Y. Li and M. J. Coates, "Particle filtering with ınvertible particle flow," *IEEE Transactions on Signal Processing*, vol. 65, pp. 4102–4116, 2017.

[16] F. Daum and J. Huang, "Nonlinear filters with log-homotopy," vol. 6699, 2007. DOI: 10.1117/12.725684. [Online]. Available: https://doi.org/10.1117/12.725684.

[17] F. Daum and J. Huang, "Nonlinear filters with particle flow induced by log-homotopy," vol. 7336, 2009. DOI: 10.1117/12.814241. [Online]. Available: https://doi.org/10.1117/12.814241.

[18] F. Daum, J. Huang, and A. Noushin, "Exact particle flow for nonlinear filters," vol. 7697, 2010. DOI: 10.1117/12.839590. [Online]. Available: https://doi.org/10.1117/12.839590.

[19] T. Ding and M. J. Coates, "Implementation of the daum-huang exact-flow particle filter," in *2012 IEEE Statistical Signal Processing Workshop (SSP)*, Aug. 2012, pp. 257–260. DOI: 10.1109/SSP.2012.6319675.

[20] A. N. Fred Daum Jim Huang, "Coulomb's law particle flow for nonlinear filters," vol. 8137, 2011. DOI: 10.1117/12.887514. [Online]. Available: https://doi.org/10.1117/12.887514.

[21] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07, New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035, ISBN: 978-0-898716-24-5. [Online]. Available: http://dl.acm.org/citation.cfm?id=1283383.1283494.

[22] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986. DOI: 10.1177/027836498600500404.

[23] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun. 2001, ISSN: 1042-296X. DOI: 10.1109/70.938381.

[24] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, Jun. 2006, ISSN: 1070-9932. DOI: 10.1109/MRA.2006.1638022.

[25] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 108–117, Sep. 2006, ISSN: 1070-9932. DOI: 10.1109/MRA.2006.1678144.

[26] J. Sola. (2014). Simulataneous localization and mapping with the extended kalman filter, [Online]. Available: http://www.iri.upc.edu/people/jsola/JoanSola/objectes/curs_SLAM/SLAM2D/SLAM%20course.pdf (visited on 05/14/2019).

[27] S. Thrun, "Probabilistic robotics," *Commun. ACM*, vol. 45, no. 3, pp. 52–57, 2002, ISSN: 0001-0782. DOI: 10.1145/504729.504754. [Online]. Available: http://doi.acm.org/10.1145/504729.504754.

[28] S. Williams, "Efficient solutions to autonomous mapping and navigation problems," *Ph.D. Dissertation*, 2001.

[29] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, vol. 4, Mar. 1987, pp. 850–850. DOI: 10.1109/ROBOT.1987.1087846.

[30] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEE Proceedings F - Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, Apr. 1993, ISSN: 0956-375X. DOI: 10.1049/ip-f-2.1993.0015.

[31] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, May 1999, 1322–1328 vol.2. DOI: 10.1109/ROBOT.1999.772544.

[32] M. Labbé and F. Michaud, "Appearance-based loop closure detection for online large-scale and long-term operation," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 734–745, Jun. 2013, ISSN: 1552-3098. DOI: 10.1109/TRO.2013.2242375.

[33] M. Labbe and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based slam," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 2661–2666. DOI: 10.1109/IROS.2014.6942926.

[34] T. Botterill, S. Mills, and R. Green, "Bag-of-words-driven, single-camera simultaneous localization and mapping," *Journal of Field Robotics*, vol. 28, no. 2, pp. 204–226, 2011. DOI: 10.1002/rob.20368. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.20368. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20368.

[35] Sivic and Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proceedings Ninth IEEE International Conference on Computer Vision*, Oct. 2003, 1470–1477 vol.2. DOI: 10.1109/ICCV.2003.1238663.

[36] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383172.

[37] M. West and J. Harrison, *Bayesian Forecasting and Dynamic Models (2Nd Ed.)* Berlin, Heidelberg: Springer-Verlag, 1997, ISBN: 0-387-94725-6.

[38] E. Duymaz, A. E. Oğuz, and H. Temeltaş, "Eş zamanlı konum belirleme ve haritalama probleminde yeni bir durum tahmin yöntemi olarak parçacık akış filtresi," *Journal of the Faculty of Engineering and Architecture of Gazi University*, vol. 32, no. 4, pp. 1255–1270, Jan. 2017. DOI: 10.17341/gazimmfd.369697.

[39] H. Örenbaş and M. Mercimek, "Clustered exact daum-huang particle flow filter," *Mathematical Problems in Engineering*, vol. 2019, p. 8, 2019. DOI: 110.1155/2019/8369565. [Online]. Available: https://doi.org/10.1155/2019/8369565.

[40] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982, ISSN: 0018-9448. DOI: 10.1109/TIT.1982.1056489.

[41] O. Hlinka, O. Slučiak, F. Hlawatsch, P. M. Djurić, and M. Rupp, "Distributed gaussian particle filtering using likelihood consensus," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 3756–3759. DOI: 10.1109/ICASSP.2011.5947168.

[42] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943, ISSN: 1522-9602. DOI: 10.1007/BF02478259. [Online]. Available: https://doi.org/10.1007/BF02478259.

[43] F. Rosenblatt, *The Perceptron, a Perceiving and Recognizing Automaton Project Para.*: Cornell Aeronautical Laboratory, 1957.

[44] D. Hebb, *The organization of behavior: a neuropsychological theory.*: John Wiley, 1964.

[45] S. Lowel and W. Singer, "Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity," *Science*, vol. 255, no. 5041, pp. 209–212, 1992, ISSN: 0036-8075. DOI: 10.1126/science.1372754. eprint: https://science.sciencemag.org/content/255/5041/209.full.pdf. [Online]. Available: https://science.sciencemag.org/content/255/5041/209.

[46] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," in, D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds., Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362, ISBN: 0-262-68053-X. [Online]. Available: http://dl.acm.org/citation.cfm?id=104279.104293.

[47] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh and M. Titterington, Eds., ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a.html.

[48] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing ınternal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. arXiv: 1502.03167. [Online]. Available: http://arxiv.org/abs/1502.03167.

[49] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. arXiv: 1207.0580. [Online]. Available: http://arxiv.org/abs/1207.0580.

[50] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[51] D. E. Rumelhart and J. L. McClelland, "Learning ınternal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. MITP, 1987, ISBN: 9780262291408. [Online]. Available: https://ieeexplore.ieee.org/document/6302929.

[52] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, Eds., ser. Proceedings of Machine Learning Research, vol. 27, Bellevue, Washington, USA: PMLR, Jul. 2012, pp. 37–49. [Online]. Available: http://proceedings.mlr.press/v27/baldi12a.html.

[53] N. Marques, F. Meneses, and A. Moreira, "Combining similarity functions and majority rules for multi-building, multi-floor, wifi positioning," in *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Nov. 2012, pp. 1–9. DOI: 10.1109/IPIN.2012.6418937.

[54] K. Kaemarungsi and P. Krishnamurthy, "Properties of indoor received signal strength for wlan location fingerprinting," in *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004.*, Aug. 2004, pp. 14–23. DOI: 10.1109/MOBIQ.2004.1331706.

[55] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, J. P. Avariento, T. J. Arnau, M. Benedito-Bordonau, and J. Huerta, "Ujıındoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems," in *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Oct. 2014, pp. 261–270. DOI: 10.1109/IPIN.2014.7275492.

[56] S. Bozkurt, G. Elibol, S. Gunal, and U. Yayan, "A comparative study on machine learning algorithms for indoor positioning," in *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*, Sep. 2015, pp. 1–8. DOI: 10.1109/INISTA.2015.7276725.

[57] R. Elbasiony and W. Gomaa, "Wifi localization for mobile robots based on random forests and gplvm," in *2014 13th International Conference on Machine Learning and Applications*, Dec. 2014, pp. 225–230. DOI: 10.1109/ICMLA.2014.42.

[58] B. A. Akram, A. H. Akbar, and O. Shafiq, "Hybloc: Hybrid ındoor wi-fi localization using soft clustering-based random decision forest ensembles," *IEEE Access*, vol. 6, pp. 38 251–38 272, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2852658.

[59] M. T. Uddin and M. M. Islam, "Extremely randomized trees for wi-fi fingerprint-based indoor positioning," in *2015 18th International Conference on Computer and Information Technology (ICCIT)*, Dec. 2015, pp. 105–110. DOI: 10.1109/ICCITechn.2015.7488051.

[60] M. R. Nowicki and J. Wietrzykowski, "Low-effort place recognition with wifi fingerprints using deep learning," *CoRR*, vol. abs/1611.02049, 2016. arXiv: 1611.02049. [Online]. Available: http://arxiv.org/abs/1611.02049.

[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

Tezden Üretilmiş Yayınlar

İletişim Bilgileri: orenbas@yildiz.edu.tr

Makale

1. H. Örenbaş and M. Mercimek, “Clustered Exact Daum-Huang Particle Flow Filter,” Mathematical Problems in Engineering, vol. 2019, Article ID 8369565, 2019. https://doi.org/10.1155/2019/8369565.