T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DAĞITIK MOBİL ROBOTLAR İÇİN YENİ BİR OTONOM YOL PLANLAMA VE ENGEL TESPİT SİSTEMİNİN TASARIMI

DOKTORA TEZİ

Gökhan ATALI

Enstitü Anabilim Dalı	**:**	**MEKATRONİK MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Prof. Dr. Sinan Serdar ÖZKAN**

Aralık 2018

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DAĞITIK MOBİL ROBOTLAR İÇİN YENİ BİR OTONOM YOL PLANLAMA VE ENGEL TESPİT SİSTEMİNİN TASARIMI

DOKTORA TEZİ

Gökhan ATALI

Enstitü Anabilim Dalı : MEKATRONİK MÜHENDİSLİĞİ

Bu tez 27 / 12 / 2018 tarihinde aşağıdaki jüri tarafından oybirliği ile kabul edilmiştir.

Prof. Dr. **Sinan Serdar ÖZKAN** **Jüri Başkanı**	**Prof. Dr.** **Durmuş KARAYEL** **Üye**	**Prof. Dr.** **Recep KOZAN** **Üye**
Doç. Dr. **İhsan PEHLİVAN** **Üye**	**Doç. Dr.** **Ahmet ZENGİN** **Üye**	

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Gökhan ATALI

05.11.2018

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde değerli katkılarını esirgemeyen danışman hocam Sayın Prof. Dr. Sinan Serdar ÖZKAN'a, tezimin her aşamasında yardımcı olan değerli hocam Prof. Dr. Durmuş KARAYEL'e ve tez izlemelerimde değerli fikirleri ile bana yol gösteren Doç. Dr. İhsan PEHLİVAN hocama teşekkür ederim. Manevi desteğini hiçbir zaman esirgemeyen değerli eşim Ayşegül ATALI'ya, en zor anlarımda bana varlıkları ile destek olan kızlarım Elif ve Betül'e ve hayatım boyunca beni destekleyip bugünlere getiren annem Semihe ATALI ve babam Ercan ATALI'ya teşekkürü bir borç bilirim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

E_lc , E_rc : Güncel enkoder değerleri

E_lp , E_rp : Önceki enkoder değerleri

$S_i(\xi)$: Lazer tarama uç noktalarının gerçek ortam koordinatları

V_a : Robotun engelle çapmadan durmasını sağlayan hız kümesi

V_d : Dinamik pencere hız kümesi

V_l : Sol tekerlek ilerleme hızı

V_r : Dinamik Pencere Algoritmasında ortaya çıkan arama alanı

V_r : Sağ tekerlek ilerleme hızı

V_s : Robotun hareketi için aday hızlar kümesi

a_x^i : Aracın, aralıklı olarak dairesel bir yörüngedeki x konumu

a_y^i : Aracın, aralıklı olarak dairesel bir yörüngedeki y konumu

v_k : Mobil robotun doğrusal hızı

v_l : Sol tekerlek dönüş hızı

v_r : Sağ tekerlek dönüş hızı

w_k^i : k zamanında parçacık pozisyonu verilen gerçek sensör ölçümünün olasılıksal ağırlığı

x_k^i : k zamanındaki parçacık pozisyonu

z_k : k zamanındaki sensör ölçümü

ω_k : Mobil Robotun açısal hızı

AMCL : Adaptif Monte Carlo Lokalizasyonu (Adaptive Monte Carlo Localization)

bel(x_t) : Zamana göre x değerinin inanç fonksiyonu

D : Tahrik tekerleri arasındaki mesafe

DGPS : Diferansiyel Global Konumlandırma Sistemi (Differential Global Positioning System)

DWA : Dinamik Pencere Yaklaşımı (Dynamic Window Approach)

f(n) : Hedefe ulaşmak için gereken toplam maliyet tahmini

g(n) : n'ye ulaşmak başlangıç düğümünden itibaren biriken maliyet

GPS : Global Konumlandırma Sistemi (Global Positioning System)

H(n) : n'den hedefe giden en uygun yolun buluşsal tahmini

$hedef_x$: Hedef noktanın x koordinatı

$hedef_y$: Hedef noktanın y koordinatı

LDS : Laser Direct Structuring

LIDAR : Laser Imaging Detection and Ranging

LRF : Laser Rangefinder

M : Harita üzerindeki durum örneklerinin sayısı

MCL : Monte Carlo Lokalizasyonu (Monte Carlo Localization)

r : Tahrik tekerlerinin yarıçapı

PRM : Olasılıksal Yol Haritası (Probabilistic Roadmaps)

ROS : Robot İşletim Sistemi (Robot Operating System)

SLAM : Eş Zamanlı Konum Belirleme ve Haritalama (Simultaneous Localization and Mapping)

URDF : Unified Robot Description Format

x(t) : Aracın zamana göre x eksenindeki pozisyonu

y(t) : Aracın zamana göre y eksenindeki pozisyonu

θ : Mobil robotun yönelim açısı

ξ : Araç pozisyon vektörü

ψ : Tahmini pozisyon

φ : Tekerleğin açısal yer değiştirme hızı

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Otonom robot, Adaptif Monte Carlo Lokalizasyonu, yol planlama, engel algılama, robot işletim sistemi.

Endüstride mobil robotlar genellikle çalışma sahası içerisindeki iş istasyonları arasında yük taşıma amacıyla kullanılmaktadır. Bu yüklerin geleneksel metodlar kullanılarak forkliftler ile taşınması durumunda operatöre bağlı hata ve kazalar oluşabilmektedir. Ayrıca forkliftlerin hareket kısıtlılıkları da yüklerin transferleri sırasında sınırlı çalışma alanı oluşturmaktadır. Otonom mobil robotların endüstriyel sahalarda yer alması bu hata ve kazaların azalması konusunda önem arz etmektedir. Mobil robotların yaygın kullanım alanlarından birisi de stoklama hizmetinin verildiği depolardır. Genellikle robot filoları şeklinde kullanılan bu robotlar için ticari gelişmelerle beraber birçok akademik çalışmalarda gerçekleştirilmektedir. Bu çalışmalardan lokalizasyon ve pozisyonlama problemlerinin çözülmesi önemli bir yer almaktadır. Robotların belirlenen görevleri yerine getirirken en az güç tüketimi ile rota boyunca güvenli bir biçimde ilerlemesi, akademik ve ticari çalışmaların temelini oluşturmaktadır.

Bu tez çalışmasında, dağıtık hareket kabiliyetine sahip mobil robotların yol planlaması üzerine olasılıksal parçacık filtre tabanlı algoritmalara dayanan bir çalışma yürütülmüştür. Çalışmada Eş Zamanlı Konum Belirleme ve Haritalama (SLAM) ile Doluluk Kafesi metotları kullanılarak mobil robotların bulunduğu sahanın 2 boyutlu haritası oluşturulmuştur. Bu methotlar kullanılarak çıkarılan harita içerisindeki mobil robotların belirlenen bir başlangıç noktasından hedef noktaya Adaptif Monte Carlo Lokalizasyon (AMCL) algoritması kullanarak ulaşması sağlanmıştır. Robotlar tarafından bilinen harita üzerinde hedef noktaya ulaşma aşamaları ve navigasyon parametreleri gerçek zamanlı olarak bir sunucu istasyon üzerinden kontrol edilmektedir. Bu işlemler oluşturulan bir kablosuz ağ üzerinden SSH (Secure Shell) protokolü ile gerçekleştirilmektedir. Harita üzerinde oluşturulan dinamik rota, robotlar tarafından takip edildiği esnada oluşabilecek hareketli engellerden robotların kaçınımı kinect kameradan alınan anlık görüntüler ile sağlanmaktadır. Gerçekleştirilen bu lokal yol planlaması için ise Dinamik Pencere Algoritmasından yararlanılmıştır. Tez çalışmasında, modern algoritma ve donanımları kullanarak haritalama, pozisyonlama ve konumlama gibi birçok fonksiyonu yerine getirebilen bir sistem tasarımı ve prototipi gerçekleştirilmiştir. Bu yönüyle çalışma, Endüstri 4.0 uygulamalarının arttığı günümüzde birçok sektörde ihtiyaç hissedilen farklı yapı ve özellikteki mobil robotların ARGE çalışmaları için faydalı olabilecek niteliktedir.

DESIGN OF A NEW PATH PLANNING AND OBSTACLE DETECTION SYSTEM FOR DISTRIBUTED MOBILE ROBOTS

SUMMARY

Keywords: Autonomous robot, Adaptive Monte Carlo Localization, path planning, obstacle avoidance, robot operating system

In industry, mobile robots are generally used for load transportation between workstations in the factory. If these loads are carried by forklifts using conventional methods, errors and accidents may occur due to the operator. Furthermore, the movement limitations of the forklifts also create a limited working area during the transfer of loads. The use of autonomous mobile robots in industrial areas is important in reducing these errors and accidents. One of the common uses of mobile robots that are sensitive to environmental factors and can work autonomously is the warehouses where the stocking service is provided. For these robots, which are usually used as robot fleets is carried out in many academic studies along with commercial developments. Solving localization and positioning problems is an important part of these studies. The fact that robots move safely along the route with minimal power consumption while performing the specified tasks constitutes the basis of academic and commercial studies.

In this thesis, a study based on probabilistic particle filter algorithms has been carried out on the path planning of mobile robots with distributed mobility. In the study, firstly, 2D map of the field where robots are located have been mapping by SLAM and Occupancy Grid Mapping methods. Using these methods, the multiple mobile robots within the map were accessed from a designated starting point to the destination using the Adaptive Monte Carlo Localization algorithm. The steps to reach the destination on the map known by the robots and the navigation parameter are controlled in real time via a master station. These processes are performed via SSH protocol over a wireless network. The dynamic obstacle avoidance of the robots is provided by realtime images taken from the kinect camera while route on the map. For this local path planning, Dynamic Window Algorithm was used. In the thesis, a system design and prototype which can perform many functions such as mapping, positioning and localization using modern algorithms and equipments have been achieved. In this respect, the thesis will contribute for R & D studies of mobile robots of different structures and features which are needed in many sectors those Industry 4.0 applications.

BÖLÜM 1. GİRİŞ

Günümüzde otonom araçlar fabrikalarda yük taşıma amacıyla kullanılmaktan kişisel otomobillere kadar birçok alanda tercih edilmektedir. İnsan sürüşünün otonom araçlarla değiştirilmesinde güvenlik ve çevresel faktörler göz önüne alındığında bu tercihin önemi artmaktadır. Çevresine tepki veren ve onunla etkileşime giren bir aracın, mümkün olduğu kadar çevresine ait birçok parametreyi bilmesi gerekmektedir. Bu kapsamda otonom araç mimarileri incelendiğinde alan taraması amacıyla çeşitli sensörlerin kullanıldığı görülmektedir. Bu sensörlerden en yaygın olanı 360 derecelik bir açı ile aracın etrafını tarayabilen alan tarayıcılardır. Özellikle depo, havaalanı ve ofis kullanım alanları gibi kapalı alan konumlandırmalarda ortama ait haritalandırma işlemleri bu türde alan tarayıcılar kullanılarak yapılabilmektedir. Haritanın eş zamanlı olarak oluşturulması ve araçların oluşan bu harita içerisindeki konumlanması işlemine SLAM denilmektedir. SLAM ve uygun navigasyon yöntemi kullanılarak gerçekleştirilecek bir yol planlama eylemi mobil aracın otonom bir şekilde kendisine verilen görevleri tamamlamasını sağlayabilmektedir. Otonom navigasyon eylemleri için diğer bir önemli husus ise çevrede oluşabilecek dinamik engellere karşı gösterilecek tepkidir. Otonom sistemlerde, navigasyonun aksatılmadan devam etmesi ve beraberinde engellerden kaçınım sağlayarak eş zamanlı konum belirleme işleminin de gerçekleştirilebilmesi gerekmektedir.

Bu tez çalışmasında, dağıtık hareket kabiliyetine sahip mobil robotların yol planlaması üzerine olasılıksal parçacık filtresi algoritmalarına dayanan bir çalışma yürütülmüştür. Bu çalışmada mobil robotların belirlenen görevleri yerine getirebilmesi için gerekli olan navigasyon işlemi otonom bir şekilde gerçekleştirilmiştir. Bu kapsamda mobil robotların, parçacık filtre esas alınarak saha optimizasyonları ve yol planlamaları gerçekleştirilmiştir.

Çalışmada öncelikle robotların bulunduğu sahanın 2 boyutlu haritası SLAM ve Doluluk Kafesi ile Haritalama yöntemleri kullanılarak oluşturulmuştur. Bu metodlar kullanılarak çıkarılan saha içerisindeki mobil robotun belirlenen bir başlangıç noktasından hedef noktaya parçacık filtre tabanlı AMCL algoritması kullanarak ulaşması sağlanmıştır. Robotlar tarafından bilinen harita üzerinde hedef noktaya ulaşma aşamaları ve parça filtre parametreleri gerçek zamanlı olarak bir sunucu istasyon üzerinden kontrol edilmektedir. Bu işlemler oluşturulan bir kablosuz ağ üzerinden SSH protokolü ile gerçekleştirilmektedir. Harita üzerinde AMCL algoritması ile oluşturulan global yol, robotlar tarafından takip edildiği sırada oluşabilecek dinamik engellerden lokal yol planlamada robotların kaçınımı ise Dinamik Pencere yaklaşımı ve kinect kamera kullanılarak gerçekleştirilmiştir.

Çalışmanın gerçeklenmesinde ROS (Robot Operating System) işletim sistemi ve Rviz arayüzleri, analiz ve doğrulama işlemleri için ise Matlab Robotics System Toolbox ve Gazebo kullanılmıştır. İlk olarak konu ile ilgili çalışmalar araştırılmış ve sonuçları sunulmuştur. Bölüm 3'te ise ROS, SLAM, Bayes filtreleri, Parçacık Filtreleri, Doluluk Kafesi ile Haritalama, Dinamik Pencere Yaklaşımı, Otonom Sistem Taksonomisi ve Yol Planlama gibi tez çalışmasının bilimsel dayanakları sunulmuştur. Daha sonra tez çalışması sırasında yürütülen navigasyon çalışmaları sonuçları ile birlikte verilmiştir. Son olarak genel değerlendirmeler ile birlikte gelecek çalışmalar hakkında bilgi verilerek tez çalışması tamamlanmıştır.

BÖLÜM 2. LİTERATÜR ÖZETİ

Mobil robotların lokalizasyonlarında kullanılan Eş Zamanlı Konum Belirleme ve Haritalama'ya yönelik literatürde çeşitli yaklaşımlar vardır. Bailey ve Durrant-Whyte SLAM ve yaklaşımlarına yaptıkları çalışmalarda ayrıntılı bir şekilde yer vermişlerdir [1,2]. Genişletilmiş Kalman Filtreleri bu yaklaşımlardan biridir ve mobil robotların haritalanma işlemlerinde yaygın olarak kullanılmaktadır. Bu yaklaşım temelde lokalizasyon sırasında yeni ölçümler elde edildikçe yinelemeli olarak güncellenen bir Gauss dağılımı üzerine çalışmaktadır. Atalı ve arkadaşları yapmış oldukları bir çalışmada Genişletilmiş Kalman Filtresi (EKF) kullanarak önceden bilinen bir ortamda mobil robotun global yol planlaması ile yönlendirmesini sağlamışlardır. Araştırmacılar lokalizasyon işlemlerini gerçekleştirirken robota ait odometri verileri ile birlikte görüntü işleme teknikleri ile elde ettikleri harita verilerini EKF ile optimize etmişlerdir [3].

Bilinen ortamlardaki mobil robotların konumunu belirlemek ve izlemek için bir başka popüler yaklaşım ise Dellaert ve arkadaşları tarafından önerilen Monte Carlo Lokalizasyonudur (MCL) [4]. MCL'de, bir parçacık filtresi, inanç dağılımını göstermek için durum uzayından elde edilen bir dizi örneği sağlar. Parçacık filtrelerinin Kalman Filtrelerine kıyasla önemli bir avantajı, gelişigüzel inanç dağılımlarını temsil etme yetenekleridir. Dellaert ile aynı çalışma grubunda yer alan Fox ve arkadaşları MCL algoritmasının mobil robotların lokalizasyonlarında kullanılmasını yürütmüş ve Markov lokalizasyonunun bir türevi olarak geliştirdikleri MCL algoritmasını olasılıksal yol planlama algoritmaları arasına kazandırmışlardır. Araştırmacılar yapmış oldukları deneysel çalışmalar sonucunda MCL'nin daha önceki yaklaşımlarla kıyasla daha az hesaplama sırası ve daha iyi bir doğruluk sağladığını belirtmişlerdir [5]. Thrun ve arkadaşları da yapmış oldukları bir çalışmada Mixture Monte Carlo Lokalizasyon adlı yeni bir mobil robot lokalizasyon

algoritması sunmuşlardır. Parçacık filtresini kullanan Mixture-MCL adlı yaklaşım, küçük örnek kümeleri için oluşan parçacık bozulmaları ile beklenmedik büyük durum değişikliklerinden kurtulmayı sağlayan bir MCL sürümüdür [6]. MCL ile ilgili bir diğer çalışmayı da Zhang ve arkadaşları gerçekleştirmişlerdir. Zhang ve arkadaşları RFID teknolojisi ve Markov Zinciri Monte Carlo (MCMC) algoritması ile optimize edilmiş bir mobil robot lokalizasyon sistemi önermişlerdir. Araştırmacılar RFID etiketi kullanarak mobil robotun kapalı alan lokalizasyonunu sağladıkları çalışmada konumlamanın optimizasyonu için MCMC yaklaşımını kullanmış ve bu metot ile gerçekleştirdikleri çalışmada düşük ortalama hatalar ile konumlama sağladıklarını belirtmişlerdir [7]. Luo ve arkadaşları ise parçacık filtresi algoritmasına dayanan gerçek zamanlı bir self lokalizasyon algoritması önermişlerdir. Araştırmacılar çalışmalarında Bayes tahmininin integral aşamasını çözmek için önerdikleri Monte Carlo yöntemine ait simülasyon çalışmalarını sunmuşlar ve global konumlandırma için partikül filtresine dayalı self lokalizasyonunun etkinliğini özetlemişlerdir [8]. Aini ve arkadaşları da ROS ortamında MCL algoritmasının lokalizasyonu etkileyen parametrelerini incelemiş ve deneysel amaçlı tasarladıkları bir robot ile parametre testleri gerçekleştirmişlerdir. Araştırmacıların sonuçlarına göre parçacık sayısının az olması, robotun gerçek pozisyonuna daha yakın sonuç elde ettiğini göstermiştir [9].

Chien ve arkadaşları ise MCL algoritmasında kullanılan parçacıkların yeniden örneklendirilmesi hakkında bir çalışma yürütmüşlerdir. Araştırmacılar çok amaçlı bir parçacık sürüsü optimizasyonu (MOPSO) kullanarak, global planlamadaki koveryans kalitesini lokalizasyon sırasında koruyarak parçacık çeşitliliğini iyileştirmek için bir arama yeteneğini geliştirmişlerdir [10]. Chien ve arkadaşları yapmış oldukları bir diğer çalışmada ise Monte Carlo Lokalizasyonundaki küresel lokalizasyon için erken yakınsama problemini çözmek amacıyla yeni bir yaklaşım önermiş ve önerdikleri yaklaşımı MCL algoritmasının çeşitli türevleri ile karşılaştırmalı olarak sunmuşlardır [11]. Medina Lee ve Buitrago'da bir çalışmalarında MCL algoritmasını Lego NXT robot platformunu kullanarak gerçeklemişlerdir. Alan taraması için ultrasonik sensör kullanılan çalışmada Java ile hazırlanan grafiksel bir arayüz ile robotun basit yön kontrolleri gerçekleştirilmiştir. Ayrıca robotun hareket planlamasının yapılacağı

ortama ait harita bilgisini, alanı belirli bir yükseklikten izleyen bir tepe kamerası ile elde etmişlerdir [12]. Garip ve arkadaşları da Lee ve Buitrago'nun çalışmasına benzer şekilde gerçekleştirmiş oldukları bir çalışmada robotun global yol planlamasını gerçekleştirmek amacıyla alanı izleyen bir tepe kamerası kullanmışlardır. Araştırmacılar bu kameradan aldıkları veriyi görüntü işleme teknikleri ile 2 boyutlu harita verisine dönüştürmüş ve robota ait yol planlama bilgisini A* algoritmasını kullanarak Matlab ortamında gerçekleştirmişlerdir [13].

Literatürde mobil robotların lokalizasyonlarında kullanılan MCL'ye benzer şekilde Bayes tabanlı farklı olasılıksal yöntemlerde mevcuttur. Örneğin Yılmaz ve arkadaşları olasılıksal konumlandırma problemlerinin çözümü için aykırı algılayıcı ölçümlerini yok sayan yeni bir algılayıcı modeli önermişlerdir. Araştırmacılar Aykırı Olasılıkları Yoksayan Algılayıcı Model (AOYAM) olarak adlandırdıkları modelde en yüksek sapmalı olasılığın olduğu tarafı tespit ederek yok etme yöntemine gitmişlerdir [14]. Duymaz ve arkadaşları da SLAM çalışmalarında kullanılmak üzere yeni bir yaklaşım ortaya koymuşlardır. Çalışmalarında önerdikleri yöntemin insansız kara sistemlerinde uygulanabilirliğini simülasyon ortamında incelenmişler ve parçacık akış filtresi tabanlı EZKH adını verdikleri yaklaşımın performans analizlerini ölçmüşlerdir. Araştırmacılar çalışmalarının parçacık sayısı, parçacık migrasyon adımı, süreç ve ortam gürültüleri gibi parametrelere bağlı olarak literatürde daha önce yer almış diğer tahmin yöntemlerine göre üstünlükler taşıdığı ve doğruluk açısından daha iyi sonuçlar verdiğini ancak hesaplama maliyeti açısından bazı gerçek zamanlı uygulamalar için kırılganlıklarının olabileceğini belirtmişlerdir [15]. Mobil robotların ızgara tabanlı olasılıksal lokalizasyonları için yapılan çalışmalardan biride Perez ve arkadaşlarının yapmış olduğu bulanık belirsizlik modellemesidir. Araştırmacılar özyinelemeli durum tahmini güncelleme döngüsünü, bulanık mantık sistemi ile lokalizasyon problemleri için uyarlamışlar ve çalışmalarını farklı üç robot platformu üzerinde gerçekleştirmişlerdir. Çalışmalarının sonucunda yüksek hareket belirsizliği durumlarında en modern olasılıksal lokalizasyon yönteminden daha istikrarlı tahminler sağladıklarını belirtmişlerdir [16]. Raza ve Fernandez de bağışıklık sisteminden esinlenerek heterojen mobil robotik sistemler için çok katmanlı bir taslak çalışması gerçekleştirmişlerdir. Karmaşık

labirentleri çözmek ve yapılandırılmamış senaryolarda arama kurtarma çalışmaları gerçekleştirmek amacıyla geliştirdikleri bu çalışma 3 katmandan meydana gelmektedir. Raza ve Fernandez çalışmalarını farklı robot konfigürasyonları ile birden fazla görevi yerine getirerek, kapsamlı benzetimsiz konfigürasyon senaryoları ile doğrulanmışlardır [17]. Mirkhani ve arkadaşları ise yapmış oldukları bir çalışmada, tarama eşleştirmesi yoluyla robot lokalizasyonu için harmonik arama algoritmasına dayanan yeni bir yöntem önermişlerdir. Çalışmalarında önerdikleri yöntem genetik algoritma tabanlı bir yaklaşımla karşılaştırıldığında, daha iyi doğruluk ve daha yüksek performansa sahip olduğunu belirtmişlerdir. Araştırmacılar ayrıca çalışmalarının bir sonucu olarak harmonik arama ve diferansiyel evrim algoritmalarına dayanan yeni bir melez algoritma önermişlerdir [18]. Miao ve Tian dinamik yol planlama için yeni bir benzetimli tavlama yaklaşımı önermişlerdir. Geliştirmiş oldukları yaklaşım standart benzetimli tavlama ile karşılaştırıldığında iki adet ilave matematiksel operatör içermektedir. Araştırmacılar yol seçimi sezgilerini standart benzetimli tavlamaya entegre edebilen bu yaklaşım ile robotun dinamik yol çözümlemesinde geçen işlem süresinde iyi bir performans sağladıklarını deneysel veriler ile ifade etmişlerdir [19].

Özellikle otomotiv alanında gerçekleştirilen otonom mobil robot çalışmalarında olasılıksal lokalizasyon çalışmalarından yararlanılmaktadır. Bu alandaki çalışmalarda güncelleme yapma kapasitesine sahip kalıcı bir harita oluşturmak için GPS, IMU ve Lidar'ın bir kombinasyonu kullanılmaktadır [20]. Rohde ve arkadaşları 3D lidar sensörü tarafından sağlanan verilerden oluşturulan çoklu harita katmanlarında aynı anda MCL gerçekleştirmişlerdir [21]. Vu ve arkadaşları ise dinamik dış çevrede gerçekleştirdikleri çalışmada aracın çevresini takip etmek için yerel bir ızgaralı harita kullanmışlardır [22]. Araştırmacılar çalışmalarında hareket modeline göre çizilen örneklere dayanan tarama eşleştirmesini konum güncellemeleri için kullanmışlardır. Bu çalışmalar incelendiğinde görülmektedir ki, 2D lidar sensörleri ve 3D lidar sensörleri birbirinden kullanım alanına göre farklılık teşkil etmektedir. 2D alan tarayıcı sensörler mobil robot uygulamaları için kolay çözülebilir bir veri seti ve maliyeti düşük yöntemler sunmaktadır. Ancak bu sensörler dinamik engellerin çok olduğu ortamlarda kısıtlı görüş alanına sahip oldukları için bazı olumsuzlukları

beraberinde getirmektedir. 2D tarayıcı sensörlere göre daha maliyetli olan 3D alan tarayıcı sensörler, yer düzlemine ait tarama noktalarının spesifik olarak seçilmesine izin vermekte, bu da dinamik engellerden bağımsız ölçümler yapılabilmesini olanak sağlamaktadır [23,24]. Alan tarayıcı bu sensörlere alternatif olarak lokalizasyon için uygulanan bir diğer yaklaşım ise kamera kullanımıdır [25]. Örneğin Cosgun ve arkadaşları otonom sürüşe sahip araçların teknik zorluklarını çözmeye yönelik gerçekleştirmiş oldukları çalışmalarında, yol işaretlerini kullanarak yanal konum tahminlerini geliştirmek için yolu gösteren bir kamera kullanmışlardır. [26].

Literatürde yer alan çalışmalar incelendiğinde görülmektedir ki mobil robotların otonom yol planlamaları üzerine modern algoritmalara dayanan birçok çalışma yürütülmektedir. Bu çalışmaların başında olasılıksal yöntemlere dayanan algoritmalar öncelikli çalışma alanlarıdır. Bu tez çalışmasında da dağıtık hareket kabiliyetine sahip mobil robotların dinamik yol planlaması üzerine olasılıksal parçacık filtre tabanlı bir sistem tasarımı gerçekleştirilmiştir.

BÖLÜM 3. MATERYAL VE YÖNTEM

Bu bölümde tez çalışması sırasında kullanılan standart teknikler ve yöntemler açıklanmaktadır. ROS, SLAM, Bayes Filtreleme, Monte Carlo Parçacık Filtreleri, Doluluk Kafesi ile Haritalama (Occupancy Grid Mapping) konularından örneklerle kısaca bahsedilmiştir.

3.1. Robot İşletim Sistemi (ROS)

ROS ilk olarak 2000'li yılların ortalarında Stanford Üniversitesinde STAIR (Standford AI Robot) ve PR (Personal Robot Program) adlı sistemlerin geliştirilmesine destek amaçlı ortaya çıkmıştır. 2007 yılında, Menlo Park, California'da bulunan bir şirket olan Willow Garage, önemli kaynaklar sağlayarak sistemin geliştirilmesini sağlamıştır. Bu durum ROS'un gelişmesine önemli ölçüde kaynak ve uzman sağlayarak BSD lisansı altında birçok çalışmaların başlangıcı olmuştur. Zamanla ROS, robotik topluluk araştırmaları arasında yaygın olarak kullanılan bir platform haline gelmiştir. 2013 yılında ise ROS'un gelişimi Open Source Robotics Foundation (OSRF) adlı kuruluşa aktarıldı ve halen bu kurum üzerinde devam etmektedir. Günümüzde ROS, dünyanın dört bir yanında deneysel ve endüstriyel alanlardaki projelerde on binlerce kullanıcı tarafından yürütülmektedir. Aynı zamanda geniş bir doküman paylaşımının yer aldığı bu robotik işletim sistemi Melodic Morenia [27], Lunar Loggerhead [28] ve Kinetic Kame [29] adlı işletim sistemleri ile hizmet vermektedir. Bu tez çalışmasında en uzun destek süresine sahip Kinetic Kame adlı versiyon tercih edilmiştir. Bu işletim sistemlerine ait versiyon ve destek tarihleri Tablo 3.1.'de sunulmuştur.

Tablo 3.1. ROS versiyonları

Yayıncı adı	Yayın tarihi	Son destek tarihi
Lunar Loggerhead	23.05.2017	05.2019
Kinetic Kame	23.05.2016	04.2021 (Xenial)
Jade Turtle	23.05.2015	05.2017
Indigo Igloo	22.07.2014	04.2019 (Trusty)

ROS işletim sisteminde bulunan yığınlar ve paketler, tüm dünyadaki üniversiteler, şirketler ve özel kişiler tarafından geliştirilmekte ve genellikle açık kaynak kod ile lisanslanmaktadır. Bir paket ve yığın, ana ROS web sitesinde bir dizine eklenerek tüm topluluklar arasında erişime açılabilmektedir [30]. ROS'un bu paylaşım ağı ROS yazılımlarının dünya üzerinde gittikçe yaygınlaştırılması açısından büyük öneme sahiptir.

ROS, robotik platformlar için özel bir yazılım ve çerçeve birimi olarak tasarlanmıştır. Temel olarak birçok robotik araç, donanımsal simülasyon ve çeşitli yazılımsal düğümler arasında mesaj aktarma sistemi sağlar. Düğümler bağımsız olarak çalışabilen, bire-çok abone modeli ve TCP / IP protokolü kullanarak söz konusu konular üzerinde iletişim kuran bağımsız modüllerdir. ROS'un bu özelliği dağıtılmış sistemlerle çalışırken büyük önem taşımaktadır. Örneğin, tekli bir robot işleminde bile gereken işlevsellik yani sensör ve aktüatör kontrolleri için bileşenler yoğun bir şekilde artabilmektedir. Haritalama, navigasyon ve engellerden kaçınım bu dağıtık sensör ağı ile gerçekleştirilebilecek uygulamalardan sadece birkaçını oluşturmaktadır. Her sistem hangi konunun hangi düğüm tarafından yayınlandığı ile ilgili bilgiye sahip düğümler sağlayan bir isim sunucusuna yani ROS-Master'a sahiptir. Bu sebeple, her bir düğüm, yayınladığı konuları ROS-Master'da kaydetmelidir. Bir konuya abone olmak isteyen başka bir düğüm, bu konudaki yayınların adresini aramak için ROS-Master'ı kullanır ve onlarla doğrudan bağlantı kurar. Bu nedenle, ilk arama aşamasından sonra, düğümler doğrudan eşler arası ağda olduğu gibi bağlantı sağlarlar.

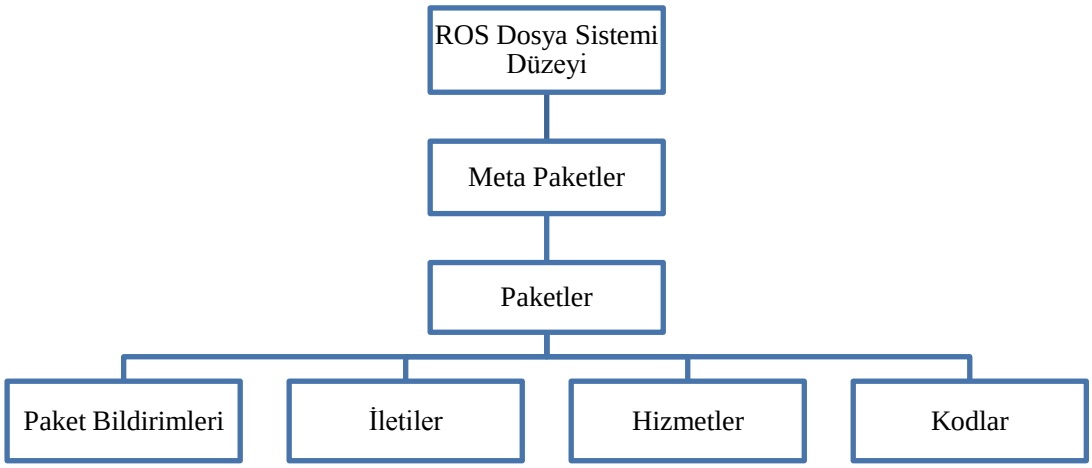
ROS'un kendisi esas olarak C++ ve Python'da yazılmıştır. ROS düğümleri C++, Python, Java, Matlab, Lisp ve diğer bazı diller için sağlanan bir istemci kütüphanesi

kullanır. Bununla birlikte, mesaj iletim sistemi ve protokol ROS-Master ile iletişim kurarken herhangi bir programlama dili kullanılabilir. Ayrıca ROS işletim sistemi esasen Ubuntu/Linux altında çalışmakta, diğer işletim sistemleri tarafından ise kısmen desteklenmektedir. Bununla beraber, ROS'un modüler yapısı, farklı ayarlar için kolayca çeşitli konfigürasyonlar oluşturulmasını sağlar. Gerçekleştirilen tez çalışmasında kullanılan kodlar simülasyon ve gerçek sistem üzerinde test edilmiştir. Gerçekleştirilen bu çalışmaların diğer ROS özellikli robotlarda çalıştırılabilmesi için sadece navigasyon modülünün robotun hareket ve sensör modeline göre uyarlanması gerekmektedir.

ROS yapısını üç farklı düzeyde ele almak mümkündür;

1) Dosya sistemi düzeyi,
2) Hesaplamalı Grafik düzeyi,
3) Topluluk Düzeyi

1-) Dosya sistemi düzeyi: Bir işletim sistemine benzer şekilde, ROS dosyaları da sabit diskte düzenlenir. Bu seviyede, dosyaların diskte nasıl organize edildiğini belirtilmektedir.
Şekil 3.1.'de ROS dosyalarının ve klasörünün hafızada nasıl düzenlendiğine ait şematik yapı gösterilmiştir [31].



Şekil 3.1. ROS dosya sistem mimarisi

Packages (Paketler): Paketler ROS'un önemli çekirdek bloklarıdır ve konfigürasyon dosyası, düğüm bilgileri, kütüphaneler gibi olması gerekli asgari içerikleri barındırır.

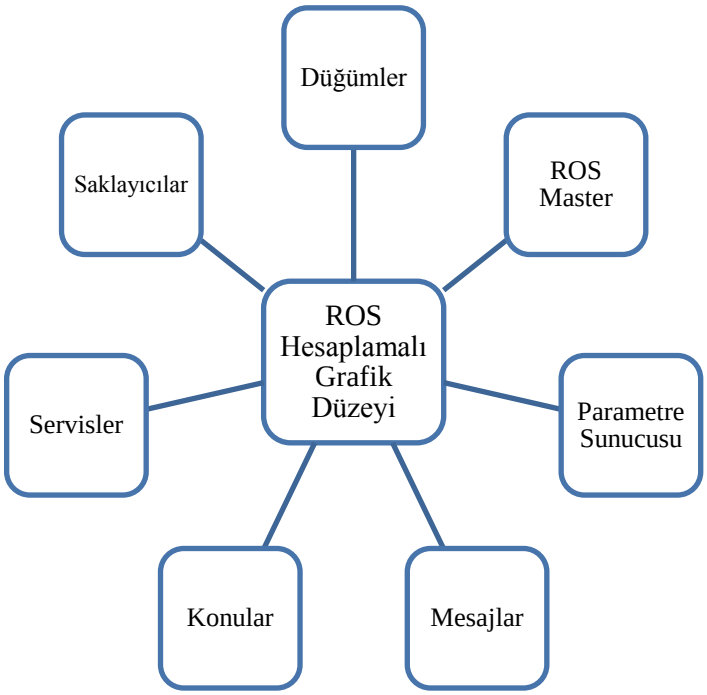
Manifests (Paket bildirimleri): Paket bildirim dosyası, paket hakkındaki bilgileri, yazar, lisans, gereklilikler, derleme bayrakları vb. bilgileri içermektedir. ROS paketinin içindeki package.xml dosyası, bu paketin manifest dosyasıdır.

Messages <.msg> (İletiler): ROS iletileri, bir ROS işleminden diğerine gönderilen bilgi türüdür.

Services <.srv> (Hizmetler): ROS hizmeti, işlemler arasında bir tür istek / yanıt etkileşimidir.

Repositories (Depolar): ROS paketlerinin çoğu git (github), subversion (svn), mercurial (hg) gibi bir Sürüm Kontrol Sistemi (VCS) tarafından depolanır.

2-) Hesaplamalı Grafik Düzeyi: ROS'taki hesaplama, ROS düğümleri adı verilen bir işlem ağı kullanılarak yapılır. Bu hesaplama ağı, hesaplama grafiği olarak da adlandırılabilir. Hesaplama grafiğindeki ana kavramlar, ROS düğümleri, ROS Master, Parametre Sunucusu, Mesajlar, Konular, Servisler ve ROS mesajlarının saklandığı Saklayıcılar kavramlarıdır (Şekil 3.2.).



Şekil 3.2. ROS hesaplamalı grafik düzeyi yapısı

3-) Topluluk düzeyi: Bu düzey ROS'un yazılım ve bilgi alışverişi için yeni bir topluluğa imkan veren ROS kaynaklarıdır. Bu kaynaklardan bazıları şunlardır; ROS Wiki, ROS Answers, Blogs.

3.2. Eş Zamanlı Konum Belirleme ve Haritalama

Robotun çevresine ait bir haritayı eş zamanlı olarak oluşturması ve oluşturulan bu haritada konumunu sürekli olarak belirlemesi durumuna Eş Zamanlı Konum Belirleme ve Haritalama denilmektedir. SLAM ile harita elde edebilmek için farklı yaklaşımlar mevcuttur. Bu yaklaşımlar;

- Aynı anda aynı ortamda gezinen ve haritayı oluşturmak için iş birliği yapan bir veya birden fazla robot ile yapılan SLAM,
- Haritanın, robotun anlamlı bir biçimde katkı sağlamasından önce kısmen ya da tamamen önceden yapıldığı çalışmalar,
- Haritanın, çevrenin özelliklerine veya robota bağlı olarak belirli bir alan ile sınırlandırılmış olduğu uygulamalardır.

Elbette SLAM için en önemli unsur robotların hareket edeceği çevreyi algılayabilmesi için gerekli olan sensörlerdir. Hareket çevresini algılayan bu sensörler sayesinde robotlar, hareket edecekleri alanları haritalar ve kendi konumlarını belirleyecek verileri toplayabilir. Farklı ortamlar, çalışmalara özgün olarak alana uygun farklı sensörler gerektirir. Chong ve arkadaşları SLAM için en yaygın sensörlerinin aşağıda belirtilen türler olduğunu belirtmişlerdir [32].

- Ultrasonik Sensörler
- Stereo Görüntü Sensörleri
- RGB-D Sensörleri
- Lazer Mesafe Bulucuları

Ultrasonik Sensörler: Mobil robotlar için, ultrasonik sensörler genellikle mekânsal algılamanın var olan en ucuz kaynağıdır. Ancak ultrasonik sensörler düşük mekansal çözünürlüğüne ve algılama aralığına sahiptir ve buna ek olarak çevre tarafından kolayca parazite edilebilmektedirler. Dolayısıyla bu sensörler bir bakıma ölçülen yüzeyin akustik yansıtıcılığına bağımlıdırlar. Jung ve arkadaşları yaptıkları bir çalışmada ultrasonik sensör kullanarak SLAM gerçekleştirmişlerdir [33].

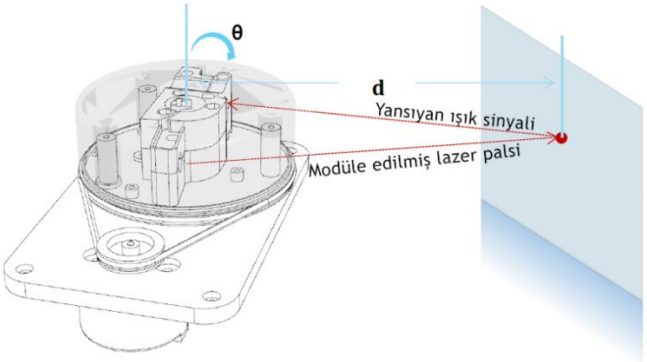
Stereo Görüntü Sensörleri: Stereo görüntü sensörleri, 3D haritalamalarda ve robotun pozisyon bilgisini bulmada kullanılan sensör tiplerindendir [34]. Robot çalışma çevresine ait özellikler ve mesafe gibi bilgiler bu kameralar tarafından çekilen çoklu görüntülerin analizi ile elde edilmektedir. Stereo görüntü sistemlerinde, 3D bilgi oluşturmak için farklı konumlardan alınan iki veya daha fazla 2D görüntü kullanılmaktadır [35]. Bir stereo kamera, uzaklık hakkında bilgiyi görüntünün pürüzlü alanlarındaki farklılıktan elde eder. Bir diğer görüntü sensörü ise monoküler kameralardır. Bu kameralar ise, özelliklerin paralaksını elde etmek için özelliklerin tekrar tekrar gözlemlenmesiyle derinlik bilgisinin elde edilmesini içermektedir. Gil ve arkadaşları çalışmalarında robot haritada bekleme noktaları olarak görsel noktaları gözlemlemeye ihtiyaç duyduğundan, özellik tabanlı SLAM kullanan bir sistem sunmuşlardır [36].

RGB-D Sensörleri: Bir RGB-D derinlik sensörü çevre hakkında derinlik bilgisi elde etmek için yapılandırılmış bir kızılötesi spektrum ışığı yansıtır ve daha sonra bir kızılötesi kamera tarafından bu ışığı algılar [37]. Bu işlem sonucunda piksel derinlik bilgisi başına bir RGB görüntüsü elde edilmektedir. Genelde, yapılandırılmış ışık sensörleri doğrudan güneş ışığı altında kullanılamaz, çünkü dış aydınlatmaya karşı hassastırlar. Endres ve arkadaşları Microsoft Kinect gibi RGB-D sensörleri için bir 3D SLAM sistemi sunmuşlardır. Sunulan yaklaşımda araştırmacılar, renkli görüntülerden görsel anahtar noktaları oluşturmuş ve bu noktalar ile robotun 3 boyutlu ortamda lokalizasyonunu sağlamışlardır [38].

LIDAR tabanlı SLAM: Bir LIDAR, yüksek düzeyde doğrulukla hızlı veri alımını sağlayabilmektedir. Bu sensörler otonom aracın yakınında bulunan nesnelerin mesafelerini temsil eden veri bulutu oluşturabilmekte ve hareket çevresini haritalayarak aracın aynı zamanda engellerden kaçınımını sağlayabilmektedir. Koch ve arkadaşları yapmış oldukları bir çalışmada iki adet robot üzerine yerleştirdikleri 2D LIDAR sensörleri vasıtasıyla paralel bir harita oluşturmuşlardır [39]. Araştırmacıların gerçekleştirdikleri bu sistem, Robocup Rescue 2015'de uygulanmış ve bu sistemin daha büyük ölçekli haritalar oluştururken Hector-SLAM gibi açık kaynaklı algoritmalara göre daha avantajlı olduğunu savunulmuştur. Amzajerdian ve arkadaşları ise çalışmalarında, bir uzay aracının iniş aşamasında, gezegensel cisimler üzerindeki yükselmeyi haritalamak için bir 3D LIDAR kullanabilme olasılığından behsetmişlerdir. Bu bilgi daha sonra navigasyon için ve inişin daha az tehlikeli olmasını sağlayacak en uygun ve güvenli iniş alanını tespit etmek için kullanılmıştır [40].

Tez çalışmasında 2 çeşit yakınlık sensörü kullanılmıştır; alan taraması yöntemi ile haritalandırma (SLAM) aşamasında RPLidar A1, robotun hareketi esnasında dinamik ve statik engellerden kaçınım sağlamak amacıyla KinectV1 RGB-D derinlik sensörü kullanılmıştır. LIDAR bir lazerle etrafını aydınlatarak yakınındaki nesnelere olan uzaklığını ölçen ve yansıtılan ışığı analiz eden bir uzaktan algılama aracıdır. Yansıyan lazer sinyali vizyon edinimi ile örneklenir ve sinyalin geri dönmesi için geçen süre hedefe olan mesafeyi gösterir. Şekil 3.3.'te mobil robot üzerine

yerleştirilen 2 Boyutlu (2D) LIDAR'ın bir duvar tarafından yansıtılan ve ardından vizyon edinimi ile örneklenen lazer darbesini nasıl yansıttığını gösterilmektedir. Şekil 3.3.'deki LIDAR kendi ekseni etrafında 360° dönmektedir ve bu şekilde çevreyi 2D düzlemde algılamaktadır.



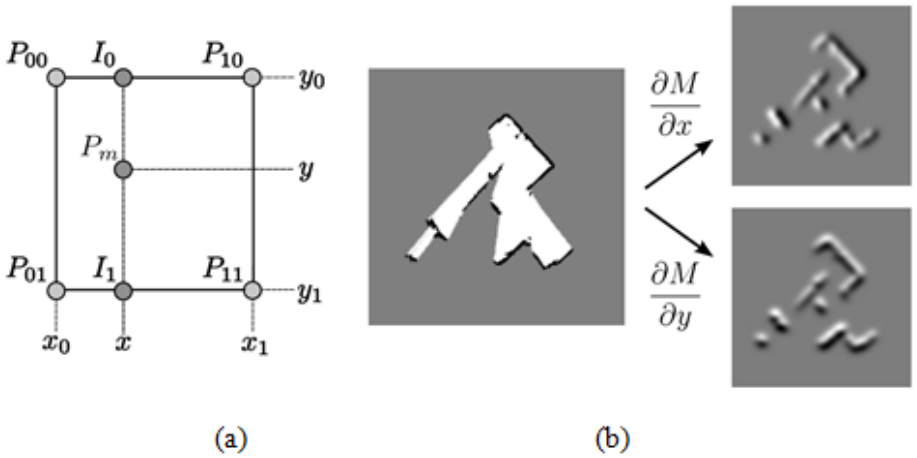
Şekil 3.3. SLAM için kullanılan 2D LIDAR [41]

3.2.1. Hector SLAM Algoritması

Hector SLAM algoritması açık kaynaklı bir 2D SLAM algoritmasıdır ve bu yöntemde, rastgele seçilmiş bir ortamı temsil edebilmek için, bir doluluk kafesi haritası kullanılmaktadır. SLAM çalışmaları esnasında lazer alan tarayıcılar ile ortamın nokta bulutu çıkarılmakta ve bu nokta bulutu ile çalışma alanı 2 boyutlu olarak ifade edilmektedir.

Doluluk Kafesinin ayrık yapısından dolayı Hector SLAM algoritması çalışırken, bi-lineer filtreleme yoluyla alt-ızgara hücre doğruluğuna izin veren bir enterpolasyon şeması kullanılmaktadır. Bu şema hem doluluk olasılıklarını hem de türevlerini tahmin etmek amacıyla kullanılmaktadır. Sezgisel olarak, ızgara harita hücre değerleri, sürekli olasılık dağılımının örnekleri olarak görülebilmektedir [42].

P_m'yi oluşturan sürekli bir haritadan Şekil 3.4'te gösterilen ve en yakın dört tamsayı koordinatı olan $P_{00}, P_{01}, P_{10}, P_{11}$ kullanılarak doluluk değeri $M(P_m)$ ve gradyan $\Delta M(P_m) = (\frac{\delta M}{\delta x}(P_m), \frac{\delta M}{\delta y}(P_m))$ tahmin edilebilir [43];



Şekil 3.4. (a) Doluluk kafesi haritasının bi-lineer filtrelenmesi. (Pm noktası enterpolasyonlu olan noktadır) (b) Doluluk kafesi ile haritalama ve mekansal türevleri [43]

Haritanın örnek noktaları (ızgara hücreleri) birbirinden 1 birim uzaklıkta düzenli bir ızgara üzerinde yer alıyorsa, x- ve y- ekseni boyunca doğrusal enterpolasyon Denklem 3.1'deki gibi elde edilmektedir.

$$M(P_m) \approx \frac{y-y_0}{y_1-y_0}\left(\frac{x-x_0}{x_1-x_0}M(P_{11}) + \frac{x_1-x}{x_1-x_0}M(P_{01})\right) + \frac{y_1-y}{y_1-y_0}\left(\frac{x-x_0}{x_1-x_0}M(P_{10}) + \frac{x_1-x}{x_1-x_0}M(P_{00})\right) \quad (3.1)$$

Denklem 3.1'e ait türevler ise;

$$\frac{\delta M}{\delta x}(P_m) \approx \frac{y-y_0}{y_1-y_0}\big(M(P_{11}) - M(P_{01})\big) + \frac{y_1-y}{y_1-y_0}\big(M(P_{10}) - M(P_{00})\big)$$

$$\frac{\delta M}{\delta y}(P_m) \approx \frac{x-x_0}{x_1-x_0}\big(M(P_{11}) - M(P_{10})\big) + \frac{x_1-x}{x_1-x_0}\big(M(P_{01}) - M(P_{00})\big)$$

şeklinde yaklaşık olarak tahmin edilebilir.

Hector SLAM algoritmasında yapılan taramaların hata oranlarının optimize edilmesi amacıyla lazer taramalarını birbiriyle veya hali hazırda var olan bir haritayla hizalama işlemi olarak tarama eşleştirilmesi yöntemi kullanılmaktadır. Lazer tarayıcılar düşük ölçüm gürültüsüne sahiptir ve hassasiyetleri birçok odometri verisinden çok daha yüksek olabilmektedir. Tarama eşleştirmesinin amacı, tarama denklemini en aza indiren rijit dönüşümü $\xi=(p_x,p_y,\psi)^T$ yi bulmak yani bir diğer deyişle, lazer taramasının harita ile en iyi hizalanmasını sağlayan dönüşümü bulmaktır (Denklem 3.2):

$$\xi^* = argmin_\xi \sum_{i=1}^{n}[1 - M(S_i(\xi))]^2 \tag{3.2}$$

Burada $S_i(\xi)$, tarama uç noktalarının robotun çalıştığı alandaki koordinatlarıdır. Bu koordinatlar, Denklem 3.3'te verilen alan koordinatlarında robot pozisyonunun (ξ) fonksiyonunu ifade etmekte kullanılır.

$$S_i(\xi) = \begin{bmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{bmatrix} \begin{bmatrix} s_{i,x} \\ s_{i,y} \end{bmatrix} + \begin{bmatrix} p_x \\ p_y \end{bmatrix} \tag{3.3}$$

$M(S_i(\xi))$, fonksiyonu harita değerini $S_i(\xi)$ pozisyonunda verir. Başlangıç konumunda verilen tahmin göz önüne alındığında, hata ölçüsünü optimize etmek için kullanılan $\Delta\xi$ tahmin edilebilir (Denklem 3.4):

$$\sum_{i=1}^{n}[1 - M(S_i(\xi + \Delta\xi))]^2 \rightarrow 0 \tag{3.4}$$

$M(S_i(\xi + \Delta\xi)$'ın birinci dereceden Taylor açılımıyla Denklem 3.5 elde edilmektedir:

$$\sum_{i=1}^{n}\left[1 - M(S_i(\xi)) - \nabla M\left(S_i(\xi)\right)\frac{\delta S_i(\xi)}{\delta\xi}\Delta\xi\right]^2 \rightarrow 0 \tag{3.5}$$

Denklem 3.5 daha sonra kısmi türevin $\Delta\xi$ bakımından sıfıra eşitlenmesiyle en aza indirilmektedir (Denklem 3.6):

$$2\sum_{i=1}^{n}\left[\nabla M\left(S_i(\xi)\right)\frac{\delta S_i(\xi)}{\delta\xi}\right]^T\left[1-M\left(S_i(\xi)\right)-\nabla M\left(S_i(\xi)\right)\frac{\delta S_i(\xi)}{\delta\xi}\Delta\xi\right]=0 \quad (3.6)$$

Denklem 3.6 bir adım ileriye taşındığında ise, $\Delta\xi$ için minimize edilmiş denklemin çözülmesi olan minimizasyon problemi için Gauss-Newton denklemi elde edilir (Denklem 3.7).

$$\Delta\xi=H^{-1}\sum_{i=1}^{n}\left[\nabla M\left(S_i(\xi)\right)\frac{\delta S_i(\xi)}{\delta\xi}\right]^T\left[1-M\left(S_i(\xi)\right)\right] \quad (3.7)$$

Burada:

$$H=\left[\nabla M\left(S_i(\xi)\right)\frac{\delta S_i(\xi)}{\delta\xi}\right]^T\left[\nabla M\left(S_i(\xi)\right)\frac{\delta S_i(\xi)}{\delta\xi}\right] \quad (3.8)$$

Denklem 3.8'de bulunan ilişkiyi kullanarak ve Denklem 3.3 ile birleştirerek, Denklem 3.9 elde edilir:

$$\frac{\delta S_i(\xi)}{\delta\xi}=\begin{bmatrix}1 & 0 & -sin(\psi)s_{i,x} & -cos(\psi)s_{i,y}\\ 0 & 1 & cos(\psi)s_{i,x} & -sin(\psi)s_{i,y}\end{bmatrix} \quad (3.9)$$

Ayrıca Denklem 3.7 (Gauss-Newton), $\nabla M\left(S_i(\xi)\right)$ ve $\frac{\delta S_i(\xi)}{\delta\xi}$ kullanılarak değerlendirilebilmekte ve $\Delta\xi$ asgari seviyeye getirilebilmektedir. Hector SLAM algoritması, harita gradyanının $\nabla M\left(S_i(\xi)\right)$ düzgün olmayan doğrusal yaklaşımları üzerinde çalışmakta ve bu sebeple asgari düzeyde yerel kuadratik yakınsama garanti edilememektedir. Ancak yine de, Kohlbrecher ve arkadaşları algoritmanın pratikte yeterli doğrulukla çalıştığı sonucuna varmıştır [42,43].

Tez çalışmasında Hector SLAM haritalama yöntemi ile birlikte AMCL parçacık filtre yöntemi kullanılarak gerçek zamanlı global ve lokal yol planlama

yapılmaktadır. Gerçek zamanlı olarak statik ve dinamik engellere göre robotun yol planlaması yeniden yapılandırılabilir bir yapıdadır ve bu işlemler temelde olasılıksal tahmin yöntemlerine dayanmaktadır.

3.3. Bayes Filtreleme

Bayes Filtresi, yeni bir ölçüm verisi seti her kullanıma sunulduğunda, x_t üzerinde *bel* inanç dağılımını güncelleyerek, t zamanında x_t durum vektörünü tahmin eder. Bu durum, tahmin ve ölçüm güncellemesi şeklinde iki adımda gerçekleşir. Ölçüm verileri kontrol girişlerinden veya u_t ve bir dizi z_t ölçümünden oluşur. Bu filtrelemede *öncel inanç* olarak adlandırılan önceki zaman adımının inancına $bel(x_{t-1})$ dayanarak, *sonraki inanç* olarak adlandırılan yeni inanç $bel(x_t)$ hesaplanır [44].

Bir güncelleme uygulandığında, her x_t için inanç güncellenir. Tahmin aşamasında:

$$bel(x_t) \leftarrow \int p(x_t \ / u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1} \tag{3.10}$$

her durumda x_t 'ye ulaşma olasılığı odometri verilerine göre hesaplanır. Denklem 3.10'da verilen $bel(x_t)$ değeri ölçüm güncellemeleri ile bir döngü içerisinde sürekli yenilenmektedir.

Ölçüm güncellemesinde inanç,

$$bel(x_t) \leftarrow \eta p(z_t \ / x_t) bel(x_t) \tag{3.11}$$

tahmin edilen durumda (x_t) gözlenen ölçüm olasılığı ile yani z_t ile çarpılır. Burada η, Bayes Teoremi uyguladıktan sonra ortaya çıkan normalizasyon faktörünü ifade etmektedir (Denklem 3.12):

$$p(x_t \ / z_{1:t}, u_{1:t}) = \frac{p(z_t \ / x_t, z_{1:t-1}, u_{1:t}) p(x_t \ / z_{1:t-1}, u_{1:t})}{p(z_t \ / z_{1:t-1}, u_{1:t})} \tag{3.12}$$

Filtreyi kullanmak için, başlangıç inancı $bel(x_0)$ belirlenmelidir. Genellikle, başlangıç durumu çeşitli donanımlar kullanılarak tam olarak bilinmektedir. Başlangıç durumunun tam olarak bilmediği durumlarda ise yaklaşık bir değer ile lokalizasyon başlatılmaktadır.

3.3.1. Parçacık filtresi

Parçacık filtreleri, robotun manipülasyonu sırasında bir sonraki $bel(x_t)$'yi, $bel(x_t)$'den rastgele seçilen sınırlı bir dizi durum örnekleriyle yakınsamaya yarar. Genellikle, parçacıklar olarak da adlandırılan durum örneklerinin sayısı (M), algoritmanın gerçeklemesi sırasında oluşan inanışı bir dereceye kadar doğru bir şekilde gösterecek kadar büyük olmalıdır [45]. Bu durumda parçacık sayısı, zamana veya diğer parametrelere bağlı olarak değişebilmektedir. Parçacık dizisi t zaman adımında Denklem 3.13'teki gibi gösterilmektedir:

$$X_t = \left\{ x_t^{[i]} \ / 1 \leq i \ \leq M \right\} \tag{3.13}$$

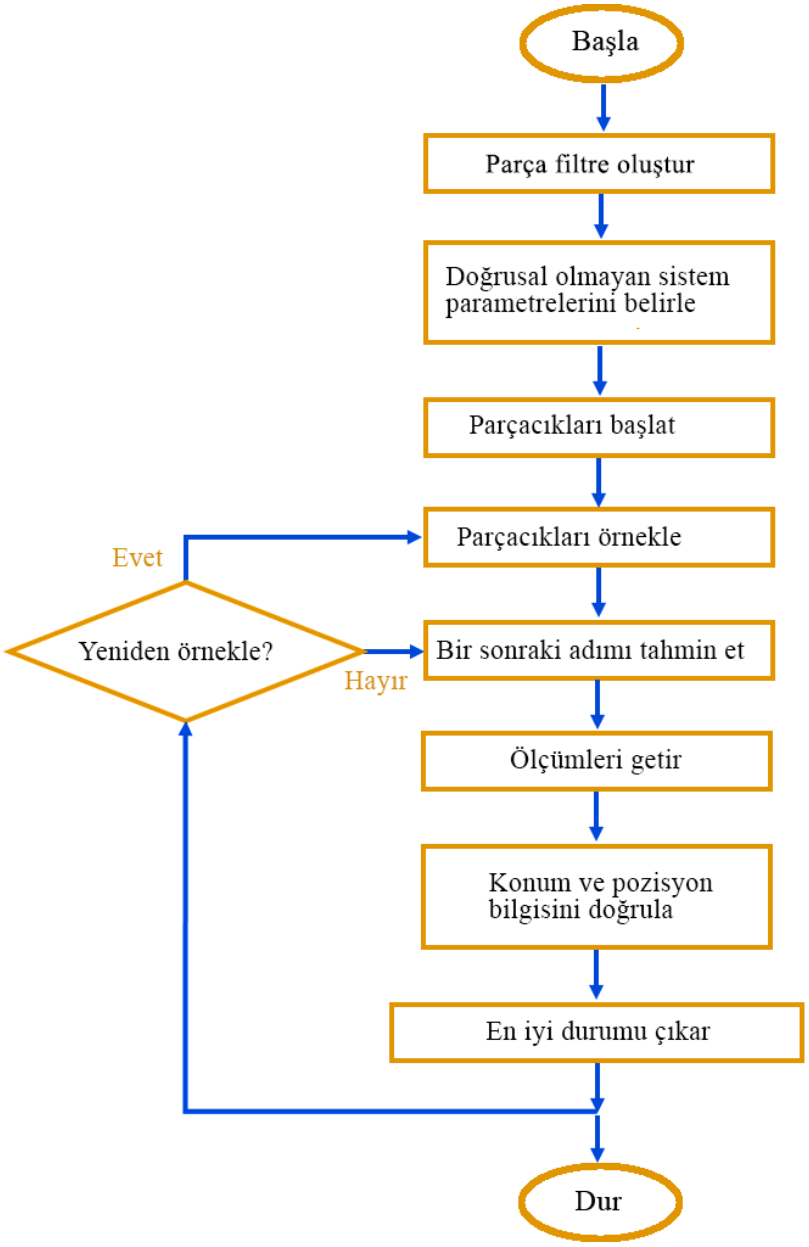
Bu durumda her bir parçacık zaman adımı $(x_t^{[i]})$, t'deki gerçek durum hakkında bir varsayımdır. X_t, $bel(x_t)$'ye yaklaştıkça, X_t'ye dahil edilecek bir x_t hipotez olasılığı, sonraki $bel(x_t)$'sine orantılı olmalıdır. Parçacıkların sayısının yeterince büyük olduğu varsayılırsa, parçacıkların yoğun olarak nüfuz ettiği durum uzayı bölümlerinin gerçek durumu içermesi oldukça muhtemeldir.

Tablo 3.2. Parçacık filtresi

1:	**function** Parçacık_Filtresi (X_{t-1}, u_t, z_t)
2:	**for all** i ∈{1, … , M} **do**
3:	örneklem $x_t^{[i]} \sim p(x_t \ / u_t, x_{t-1}^{[i]})$
4:	$w_t^{[i]} = p(z_t \ / x_t^{[i]})$
5:	**end for**
6:	$X_t = \emptyset$
7:	M parçacıkları olan $x_t^{[j]}$ leri x_t den olasılıksal $\propto w_t^{[j]}$ ile oluştur ve bunları X_t ye ekle
8:	**return** X_t
9:	**end function**

Tablo 3.2., yeni bir odometri veri seti ile birlikte bir ölçüm güncellemesi oluştuğunda gerçekleştirilen parçacık filtresinin yinelemeli güncelleme adımının basit bir uygulamasını göstermektedir. Burada öncelikle bir sonraki değeri sağlamak için ölçüm güncellemesini göz önünde bulundurmadan odometri verisine dayanan geçici $x_t^{[i]}$ kümesi oluşturulur.

Tahmin adımı, 2. adımdan 5. adıma kadar olan satırlarda döngü içerisinde gerçekleştirilir ve her parçacık odometri verisine dayalı bir güncelleme alır. Bunun için, durumuna ve durum geçiş dağılımına bağlı olarak her bir parçacık için bir durum hipotezi üretilir. Ayrıca, her yeni örneklenmiş hipotez için bir *önem faktörü* $w_t^{[i]}$ hesaplanmaktadır. Önem faktörü, yeni örneklenmiş hipotezi veren ölçüm olasılığı yani sözde *ölçüm olasılığı* ile sağlanmaktadır. Bu önem faktörlerini, parçacıkların normalize edilmemiş ağırlıkları olarak yorumladığımızda, parçacık kümesi sonrakine (posterior) $bel(x_t)$ yaklaşmaktadır ve *yeniden örnekleme* adı verilen bir sonraki adımda kullanılmaktadır [44]. Parçacık filtresinin matematiksel olarak türetilmesine ait detaylı bir anlatıma, Thrun ve arkadaşları Probabilistic Robotics adlı çalışmalarında yer vermişlerdir [45]. Parçacık filtre algoritmasına ait akış diyagramı ise Şekil 3.5.'te verilmiştir.



Şekil 3.5. Parçacık filtresi akış diyagramı

3.3.2. Monte Carlo Lokalizasyonu

Monte Carlo Lokalizasyonu konumlandırma işlemlerinin gerçekleştirilmesinde parçacık filtresi kullanmaktadır. MCL algoritması Tablo 3.3.'de belirtilen algoritmayı temel almaktadır.

Tablo 3.3. Monte Carlo Lokalizasyon algoritması

1:	**function** MCL (X_{t-1}, u_t, z_t, m)
2:	**for all** i $\in$ {1, … , M} **do**
3:	$x_t^{[i]}$ Örneklem_Hareket_Modeli (u_t, $x_{t-1}^{[i]}$)
4:	$w_t^{[i]}$ Ölcum_Modeli (z_t, $x_t^{[i]}$, m)
5:	**end for**
6:	X_t=∅
7:	M parçacıkları olan $x_t^{[j]}$ leri x_t den olasılıksal $\propto w_t^{[j]}$ ile oluştur ve bunları X_t ye ekle
8:	**return** X_t
9:	**end function**

Tablo 3.2.'de verilen parçacık filtresi ve MCL arasındaki temel fark, 2. Ve 5. Satırlar arasında verilen döngüdeki dağılımların bir hareket modeli ve bir ölçüm modeli ile değiştirilmesidir. Esasen bu durum, hareket modelinden örnekleme yapmanın ve verilen bir ölçümün belirli bir harita ile ne kadar iyi eşleştiğini tespit etmenin bir yolunu sunmaktadır [44].

Tablo 3.3.'de 6 ve 7 numaralı satırlarda, M parçacıkları, önceki aşamada belirlenen önem faktörleri ile orantılı olarak yeni örneklenmiş varsayımlar kümesinden seçilmektedir ve bu bir sonraki yaklaşım olan X_t 'i verir. Burada X_t genellikle, daha az olası varsayımlara dönüştüğü için 7. satırda çizilmeyen parçacıkların yerlerini alan kopyalar içereceği göz önünde bulundurulmalıdır. Bu adım, daha yüksek olasılıklı alanlarda önemli sayıda parçacığı tutuğu için, durum uzayı içerisinde düşük olasılıklı bölgelere yerleşen parçacıklar üzerinde çok fazla hesaplamadan kaçınmak için önemlidir. Bu kopya parçacıklar sonsuza kadar aynı kalmamaktadır. Bir sonraki aşamada her biri kendilerine uygulanan tahmin adımına sahip oldukları için birbirlerinden uzaklaşacaklardır.

Özetle parçacıkların güncellenme süreçleri şu şekildedir;

1. Parçacıklar pozisyon ve belirlenen hareket modeline göre yayılım gösterirler.
2. Sensör verisinden alınan bilgiler ile olasılığa dayanan ağırlık parçacıklara atanır.

3. Bu ağırlıklara dayanarak, robotun durum (pozisyon ve lokalizasyon) tahmini çıkarılır ve en yüksek ağırlığa sahip parçacık grubu, robotun konumunu tahmin etmek için kullanılır.
4. Son olarak, parçacıklar belirtilen yenileme aralığına göre yeniden örneklenir. Yeniden örnekleme, parçacık pozisyonlarını ayarlamanın yanı sıra yeni parçacık sayıları ile aynı zamanda performansı da artırır.

Bu hesaplamalar sonucunda algoritma tahmini pozisyonlama ve kovaryansı verir. Oluşan tahminler, parçacıkların en yüksek ağırlıklı kümelenmesinin ortalaması ve kovaryansıdır. Yukarıdaki adımlar parçacıkları yaymak, oluşan olasılıkları değerlendirmek ve en iyi durum tahmini için bir döngü içerisinde sürekli tekrarlanmaktadır.

Literatür incelendiğinde MCL algoritması üzerine birçok iyileştirmenin gerçekleştirildiği görülmektedir. Bunlardan bazıları;

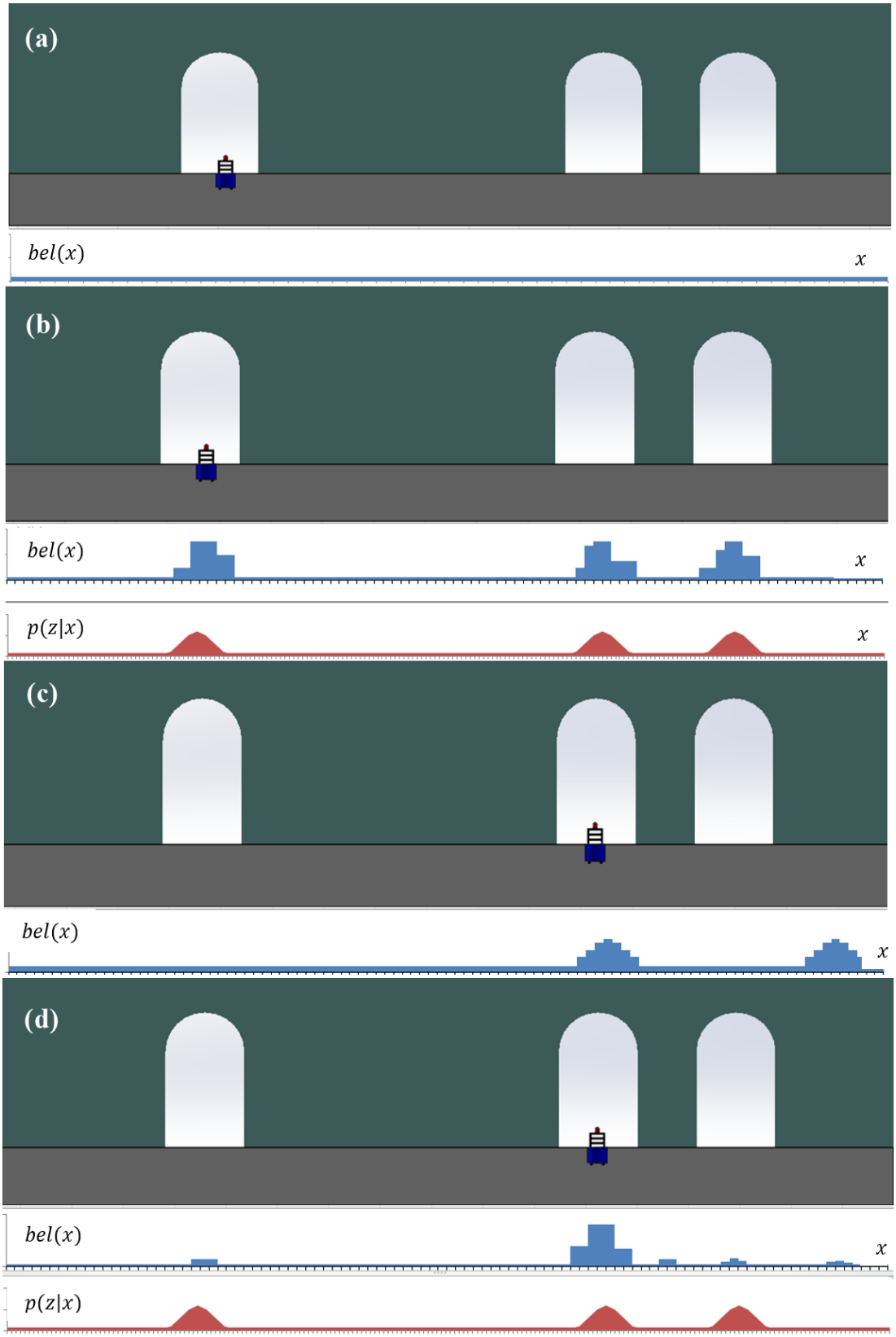
- Rastgele parçacıklar: MCL algoritmasında parçacıklar kaybolduktan sonra bir yere toplanma eğilimindedir ve filtrenin böyle bir durumdan kendi başına kurtulması pek olası bir durum değildir. Bu olasılığa karşı, her bir zaman adımında parçacık kümesine tüm durum uzayından eşit olarak örneklenmiş belirli sayıda parçacık eklenir. Bu da, belirten odometri verileri olmaksızın gerçek pozisyonu değiştirerek robotun mevcut konumundan kaçırılmasından kurtulmaya yardımcı olur.

- Kaynak-uyumlu M: Mevcut kaynaklardan en iyi şekilde yararlanmak için, Tablo 3.3.'ün 7. satırında sabit bir M ayarlamak yerine, ut−1 ve zt−1 elde edilinceye kadar daha fazla parçacık örneklenebilmektedir.

- KLD-Örnekleme: Kullback-Leibler mesafesini kullanarak parçacık filtresinin yaklaşma hatası ölçülebilir ve böylelikle M'nin dinamik olarak çok odaklı dağılımlar için daha yüksek olması sağlanabilir [46].

Şekil 3.6.'da tek boyutlu, önceden bilinen bir ortamda, sola ve sağa hareket edebilen bir robot örneği kullanılarak Monte Carlo Filtresinin çalışma prensibi gösterilmektedir. Görselde bulunan robot, katı duvarlar ve boşluklar (örneğin açık kapılar) arasında ayrım yapabilen bir sensöre sahiptir. Robot, hareketi öncesi pozisyon ve konum bilgisini bilmemektedir (Şekil 3.6.a). Parçacıkların inancı, robotun çevresinin aşağısında gösterilen parçacık kümesinde görüldüğü gibi, tüm durum uzayında eşit olarak dağıtılır. Şekil 3.6.b'de, robotun üzerinde bulunan sensör ortamdaki boşlukları yani açık kapıları algılamıştır. Her bir parçacık için ölçüm olasılığına $p(z|x)$ göre önem faktörlerine inanç $bel(x_t)$ atanmıştır ve bu da ortamdaki boşluk belirlemeleri için robotun boşluk algıladığında hissedeceği üç deliğin zirvelerini göstermektedir.

Şekil 3.6.c'de robot birkaç metre ileri yönde (sağ) hareket etmektedir. Bu adım, parçacıkların odometri verisiyle uyumlu bir şekilde yeniden örneklenip hareket ettirildikten sonraki halini göstermektedir. Burada inanç, robotun konumlanmak için en uygun bulduğu ortamdaki üç lokasyona karşılık gelen üç zirveyi göstermektedir. Bu üç lokasyonun hepsi, o zamana kadar işlenen tek sensör okuması esnasında aynı boyutta algılanmaktadır. Şekil 3.6.d'de, ağırlıklar tekrar parçacıklar ile tahsis edilir ve bu defa parçacıkların atamasının robotun doğru pozisyonu etrafında olmasının olasılığı artmaktadır. Çünkü sensör okuma dizileri ve o zamana kadar ki odometri verilerinin en çok uyuştuğu pozisyon robotun etrafında oluşmaktadır [44].



Şekil 3.6. Monte Carlo Lokalizasyon çalışma örneği [45]

3.4. Doluluk Kafesi ile Haritalama

Doluluk Kafesi ile Haritalama, robotun algıladığı ortam hakkında bilgi depolamasının temel alındığı yaygın kullanılan bir yöntemdir. Bu yöntemde, robot çevresini belirli bir boyuttaki dizi m_i hücresine ayırır. Bu hücreler genellikle eşit büyüklükte ve karedir. Üç boyutlu ızgara haritaları olmasına rağmen otonom mobil araç navigasyonu için en yaygın olanı iki boyutlu ızgara haritalarıdır. Birçok navigasyon çalışmasında ortamın genellikle yukarıdan aşağıya doğru olan görünümü sağlanarak iki boyuta yansıtılmış haritalar ile çalışılmaktadır. Burada amaç, verilen ölçümler ile robot tarafından kat edilen yolun harita m üzerinde bir sonraki değerinin hesaplamasıdır (Denklem 3.14).

$$p(m \,/\, z_{1:t}, x_{1:t}) \tag{3.14}$$

Karmaşık bir ızgara haritası için hücrelerinin sayısının ve dolayısıyla boyutların büyük olması nedeniyle bu sonraki değerin pratikte hesaplanması imkansızdır. Bunun yerine, bir sonraki değeri elde etmek için her bir hücreye ayrı ayrı bakmak standart uygulanan bir yöntem haline gelmiştir (Denklem 3.15).

$$p(m \,/\, z_{1:t}, x_{1:t}) \;=\; \prod_i p(m_i \;/ z_{1:t}, x_{1:t}) \tag{3.15}$$

Bu açıdan gerçek ortamla tam olarak uyuşmayan bir varsayımla hücreler birbirleri ile ilişkilendirilmemiştir. Bu durum, problemi her biri statik durumdaki bir dizi ikili tahmin problemine indirger ve navigasyon için tutulacak hücreler 1'e yakın değerlere, tutulamaması gereken hücreler ise 0'a yakın değerlere sahip olmaktadır. Konumlandırma için çok az bilgiye sahip olduğumuz ya da emin olamadığımız hücreler ise 0,5'e yakın değerlere sahip olmaktadır [44].

3.5. Küresel ve Bölgesel Haritalarda Yol Planlama

Statik engellerinin bilindiği varsayılan küresel ve bölgesel haritalarda robotun yol planlaması, birçok çözümü olan bir sorundur. Küresel yaklaşım, operasyon

alanındaki statik engelleri tanımlayan bölgesel ve küresel haritaların verdiği bilgileri kullanarak başlangıçtan hedefe bir yol tanımlayabilmektedir. Bu yaklaşımda, mevcut sensör ölçümlerine dayanarak sürekli oluşturulan bir haritadan faydalanılmaktadır. Küresel yaklaşımın olumsuz tarafı, küresel yöntemlerin genellikle çok fazla bilgiyi işlemesi gerektiğidir ki bu da bir yol üretmek için harcanan zamanı olumsuz etkilemektedir. İhtiyaç duyulan bu süre bir saniyeden daha az veya birkaç dakikalık bir aralıkta değişiklik göstermektedir. Bu sebeple, küresel bir yöntemin "reaksiyon süresinin" potansiyel olarak yavaş olabileceği ve hızla değişen bir ortam için uygun olmayacağı anlaşılmaktadır.

Ancak, eğer dinamik engellerden kaçınma, yol için lokal bir düzeltme olarak ifade edilirse, küresel haritada temsil edilmeyen yanlışlıkların ve dinamiklerin tespit edildiği ve yerel planlamaya dahil edildiği lokal çevre modelini ele almak için daha hızlı “reaksiyon süresine” sahip algoritmalardan faydalanılabilmek mümkündür.

3.6. Dinamik Engellerden Kaçınma

Tan ve arkadaşları çarpışma önleme sistemleri mimarisinin aşağıda belirtildiği şekilde üçe ayrılabileceği gösterilmiştir [47].

- Müzakere Mimarisi
- Reaktif Mimari
- Melez Mimari

Müzakereci mimari, global yol planlama çalışmalarında kullanılan küresel bir yol planlama mimarisidir. Bu mimaride dinamik engellerden bağımsız önceden bilinen statik çevrelerde çalışan algoritmalar yer almaktadır. Reaktif mimari ise, yerel yol planlama metodu olarak bilinmektedir. Reaktif mimari aynı zamanda bir sense-act (hisset-yap) yöntemi olarak ifade edilmektedir. Bu mimaride sensör ölçümleri, çevre haritasına ihtiyaç duyulmaksızın doğrudan kullanılabilmektedir. Bu da, daha iyi bir yanıt süresi sağlayan daha düşük bir hesaplama maliyetini beraberinde getirmektedir. Reaktif mimarinin dezavantajı, optimal olmayan yollara neden olabilmesi ve tek

başına kullanıldığında robotu kısıtlı değerlerde lokal bir çıkmaza düşürebilmesidir [48]. Bu nedenlerle daha uygun bir mimari oluşturmak için, müzakereci ve reaktif mimariler bir melez mimariye birleştirilmektedir. Bunu yaparak, reaktif katman ile lokal ortamdaki öngörülemeyen dinamik değişikliklerin çoğu dengelenir, müzakere katmanı ile de robotu hedefe doğru yönlendiren çevre için robotun yönleri alınabilmektedir.

3.7. Otonom Sistem Taksonomisi

Otonom bir sistem farklı özerklik seviyelerine sahiptir ve operatörün insan olduğu durumlarda karar vermede düşük seviyede bir yardımdan insan müdahalesi olmadan yüksek seviyede otonom karar vermeye kadar bir aralığı kapsayabilir. Bu kapsamda literatür incelendiğinde benzerlik ve farklılıklar açısından birçok taksonomiye rastlanmaktadır. Bu taksonomiler özerklik seviyeleri bakımından Fjellheim ve arkadaşları tarafından şu şekilde seviyelendirilmiştir [42,49];

- Seviye 1, İnsan Tarafından Çalıştırılan: Bu seviye otonom robot taksonomisinin ilk seviyesidir. Bu seviyede otonom eylem, insan tarafından başlatılan kontrol girişlerinin bir sonucudur. Sistem, sadece algılanan verilere bilgi verebilecek kapasitededir ve otonom bir kontrole sahip değildir.
- Seviye 2, İnsan Destekli: Bu seviyede sistem, insanın istenen aktiviteyi gerçekleştirmesini kolaylaştırmak için insan girdisine paralel olarak aktiviteyi önceden şekillendirebilmektedir.
- Seviye 3, İnsan Yetkilendirilmiş: Sistem, eğer yetki devredilmişse sınırlı bir kontrol faaliyetini yapabilmektedir.
- Seviye 4, İnsan Denetimli: Sistem, bir insan tarafından izin veya emir verilmişse, çeşitli aktiviteleri yapabilir. Sistem, süreç ve operasyon hakkında yeterli bilgiyi sağlayabilmektedir ve böylece insan denetleyici rolünü üstlenmektedir.
- Seviye 5, Karma Öncelikli: Bu seviyede ise hem insan hem de sistem verileri işleyebilmektedir. Operasyonları ve sistemin otoritesini düzenlemek için çeşitli araçların bulunduğu bir seviyedir.

- Seviye 6, Tam Otonom: Sistem, belirlenen tüm çevresel koşullardaki tasarlanmış aktivitelerden herhangi birini önceden oluşturmak için operatör gereksinimli bir işlem gerekmemektedir. Sistem kendi başına kritik kararlar verebilir ve durumsal farkındalığına dayanarak hareket edebilir.

Fjellheim ve arkadaşlarının yanı sıra Vagia ve arkadaşları hazırladıkları bir çalışmalarında otonom düzeylerin tanımları üzerine bir literatür taraması yapmışlardır. Araştırmacılar, farklı yazarların bir sistemin otonom düzeyini sınıflandırmak için kullandıkları farklı taksonomileri incelemiş ve karşılaştırmış ve kendi özgün taksonomilerinin sunmuşlardır. Araştırmacıların sunmuş olduğu taksonomi özetle şu şekildedir [50];

- Seviye 1, Manuel kontrol katmanı: Bilgisayar yardım sunmamaktadır.
- Seviye 2, Karar teklif katmanı: Bilgisayar, operatöre bazı kararlar sunar. Operatör karar vermek ve uygulamaktan sorumludur.
- Seviye 3, Operatör karar seçim katmanı: Operatör bir karar seçer ve bilgisayar tarafından bu karar yürütür.
- Seviye 4, Bilgisayar karar seçim katmanı: Bilgisayar bir karar seçer ve operatör onayı ile seçilen bu karar yürütür.
- Seviye 5, Bilgisayar yürütme ve insan bilgisi katmanı: Bilgisayar seçilen kararı yürütür ve operatörü bilgilendirir.
- Seviye 6, Bilgisayar yürütme ve çağrı üzerine insan bilgilendirme katmanı: Bilgisayar seçilen kararı yürütür ve sadece sorulduğu takdirde operatörü bilgilendirir.
- Seviye 7, Bilgisayar yürütme ve gönüllü bilgi katmanı: Bilgisayar seçilen kararı uygular ve karar verirse operatörü bilgilendirir.
- Seviye 8, Otonom kontrol katmanı: Bilgisayar, şartnamede olmayan bir hata meydana gelmesi haricinde her şeyi operatör bildirimi olmadan yapar. İstenilmeyen bir durum meydana gelmesi durumunda bilgisayar operatöre bilgi vermektedir.

3.8. Yol Planlama Problemi

Otonom robotların navigasyon işlemlerinde yol planlaması, en kısa yolun bulunması üzerinedir. En kısa yolun bulunması için olasılıksal ve metrik yöntemler üzerine kurgulanan birçok yöntem geliştirilmiştir. Campbell ve arkadaşları yapmış oldukları çalışmada açık alan yol planlama tekniklerinin popüler örnekleri olan Meadow Haritaları, Voroni Diyagramları, Düzenli Doluluk Kafesi ve Quadtree Haritalama'ya yer vermişlerdir [51].

Kapalı alan yol planlama problemlerinde ise ortamın statik ve dinamik olası durumuna göre farklı algoritmalar kullanılmaktadır. Bu algoritmalardan en bilinenleri A*, Dijkstra, Rapidly-exploring random tree (RRT) ve Probabilistic roadmap (PRM) algoritmalarıdır. Ancak bu algoritmalar ağırlıklı olarak statik çevreler üzerinde çalışmaktadır. Örnek olarak A* algoritması, en iyi yolu bulmak için buluşsal (heuristic) bir yaklaşım kullanmaktadır [42]. Bu algoritmaya ait f-skor olarak ifade edilen değerlendirme fonksiyonu $f(n)$ Denklem 3.16'da sunulmuştur.

$$f(n) = g(n) + h(n) \tag{3.16}$$

Burada $g(n)$, n'ye ulaşmak için diğer düğümleri geçerek başlangıç düğümünden itibaren biriken maliyettir. $h(n)$ ise, engelleri göz ardı ederek, n'den hedefe giden en uygun yolun buluşsal tahminidir. Bu durumda oluşan buluşsal maliyet Denklem 3.17'de gösterildiği şekildedir.

$$h(n) = \sqrt{(hedef_x - n_x)^2 + \left(hedef_y - n_y\right)^2} \tag{3.17}$$

Algoritma, ilerlemeyi takip etmek için açık liste ve kapalı liste olmak üzere iki liste ile çalışmaktadır. Açık liste, henüz değerlendirilmemiş olan tüm "keşfedilen" düğümlerin bir listesi olup, her bir yineleme için değerlendirilmek üzere en düşük f-skoruna sahip olanın seçildiği yer olarak ifade edilmektedir. Bu, algoritma her zaman olasılıksal ihtimali en yüksek olan düğümleri değerlendirerek yolu hesaplama

eğilimindedir. A* algoritmasında bir düğüm değerlendirildikten sonra yeniden değerlendirmekten kaçınmak amacıyla kapalı liste adı verilen bir listeye konulmaktadır. Arama bittikten sonra yolu yeniden oluşturabilmek için, her bir düğümün, en düşük f-skoruna sahip olan atasını / ebeveynini hatırladığından emin olunmalıdır, böylece arama bittiğinde bunun nereden kaynaklandığı belirlenebilmektedir. Sonuçta bu bilgi kullanılarak, en iyi yolu oluşturan düğüm kümeleri belirlenmektedir. Eğer buluşsal amaç, hedefe ulaşmak için minimum maliyeti tahmin etmiyorsa, A* algoritması hedefe en yakın olanını bulacaktır [52].

3.8.1. Dinamik pencere yaklaşımı

Tez çalışmasında, hareketli engellerin bulunduğu dinamik yapıda olan kapalı alan navigasyon sistemleri üzerine bir çalışma yürütülmüştür. Bu sebeple oluşabilecek hareketli engellerinde navigasyon sırasında gerçek zamanlı algılanabilmesi amacıyla Dinamik pencere yol planlama yaklaşımı tercih edilmiştir.

Dinamik pencere yaklaşımı ilk olarak Fox ve arkadaşları tarafından 1997 yılında ortaya konulmuştur [53]. Bu algoritma, otonom aracın dönüş hızının ve oranının belirli bir zaman aralığı içinde sabit olarak ayarlanabileceğini varsaymaktadır. Ek olarak, aracın sarsılma hızını göz ardı ederek, zaman aralığı boyunca aracın yörüngesi, bir hız vektörü kullanılarak hesaplanabilmektedir $[u_i, r_i]^T$. Bu, yörüngeler Denklem 3.18'de tanımlanan dairesel yörüngeler olarak hesaplanmaktadır.

$$a_x^i = x - \frac{u_i}{u_r} sin(\theta) \tag{3.18}$$

$$a_y^i = x - \frac{u_i}{u_r} cos(\theta)$$

Burada (x, y) aracın pozisyonunu, u_i ise i süresi boyunca aracın salınım hızını, r_i de i esnasındaki dönüş oranını ve θ aracın rotasyonunu göstermektedir. Bu rotasyonlar sırasında oluşan kavis (yay) yarıçapı Denklem 3.19 ile hesaplanmaktadır.

$$a_y^i = \left|\frac{u_i}{u_r}\right| \tag{3.19}$$

Bu algoritmada belirli bir zaman aralığı için en uygun yörüngeyi bulmak adına, farklı hız vektörlerinin oluşturduğu bir dizi yörüngenin değerlendirilmesi gerekmektedir. Dolayısıyla arama alanını azaltmak için, dinamik ve güvenlik kısıtlamaları önceden ayarlanmalıdır. Bu kısıtlamalar aşağıda alt başlıklar halinde özetlenmektedir.

Kabul Edilebilir Hızlar: Dinamik pencere yaklaşımındaki belirlenen kısıtlardan birisi, hız vektörünün $[\,u_i, r_i]^T$ kabul edilebilir olması gerektiğidir. Dolaysıyla buda hız vektörünün robotu güvenli bir yörünge boyunca yönlendirmesi gerektiği anlamına gelmektedir. Burada $(\,u_i, r_i\,)$ çifti, robotun ilgili yörüngedeki en yakın engele ulaşmadan durabilmesi durumunda kabul edilmektedir.

$$V_a = \left\{(u,r)|\ |u| \leq \sqrt{2.mesafe(u,r).\dot{u}_b}\ ,\ \wedge |r|\ \leq \sqrt{2.mesafe(u,r).\dot{r}_b}\ \right\} \tag{3.20}$$

Burada $mesafe\ (u,r)$ robotun bir engele çarpmadan yay boyunca seyahat edebileceği mesafedir. $\dot{u}_b$ ve $\dot{r}_b$ oluşan kırılmanın ivmesini ifade etmektedir. Sonuç olarak V_a, robotun bir engelle çapmadan durmasını sağlayan hız kümesi elde edilmektedir (Denklem 3.20).

Dinamik Pencere: Diğer bir kısıtlama hız arama alanının verilen aralık içinde ulaşılabilen hızlara düşürülmesi gerektiğidir. Bu kısıtlama, belirli bir aralıkta robot tarafından elde edilebilecek sınırlı ivmeleri temsil etmektedir. Bu hızlar Dinamik Pencere olarak gösterilir ve V_d olarak ifade edilir (Denklem 3.21).

$$V_d = \{(u,r)|\ u\ \in\ [u_a - \dot{u}.\Delta t, u_a + \dot{u}.\Delta t] \wedge r \in [r_a - \dot{r}\ .\Delta t, r_a + \dot{r}\ .\Delta t]\} \tag{3.21}$$

Burada $[\,u_a, r_a]^T$ kabul edilebilir hız vektörüdür.

Sonuç Arama Alanı: Hızlar üzerindeki sonuç kısıtlamaları, dinamik pencerede V_r dizisinin elde edilmesini sağlamaktadır. V_s, V_r'nin aşağıdaki gibi tanımlanmasını sağlayan tüm aday hızları içeren arama alanı olarak tanımlanmaktadır.

$$V_r = V_s \cap V_a \cap V_d$$

En Uygun Hızların Seçimi: Arama alanı belirlendikten sonra, pozisyon, mesafe ve hız kriterlerine bağlı olarak her bir hız vektörüne bir puan veren bir amaç fonksiyonunu maksimize ederek V_r dizisinden bir hız seçilmelidir (Denklem 3.22).

$$G(u,r) = \sigma(\alpha.pozisyon(u,r) + \beta.mesafe(u,r) + \gamma.hız(u,r) \quad (3.22)$$

Burada $pozisyon(u,r)$, aracın hedef pozisyon/yön ile hizalanmasını temsil etmekte ve $180 - \theta$ olarak ifade edilmektedir. θ, yörünge üzerinde aracın pozisyonuna göre tahmin edilen hedef noktanın açısıdır. $mesafe(u,r)$ belirli bir yörüngeyle kesişen en yakın engele olan mesafeyi temsil etmektedir. Belirli bir yörünge üzerinde herhangi bir engel yoksa bu değer büyük bir sabite ayarlanmaktadır. $hız(u,r)$, aracın belirli bir yörünge boyunca ulaşacağı hızı değerlendirmek için kullanılmaktadır. α, β ve γ sabitleri, her bir ölçütün önemini belirlemek için ayarlanması gereken ağırlık sabitleridir. σ fonksiyonu ise, oluşabilecek dalgalanmaları azaltmak için düşük geçiş filtresi gibi bir düzeltme işlevini içermektedir.

$G(u,r)$ hedef fonksiyonunun her üç bileşeni de hareket esnasında oluşabilecek dinamik ve statik engelleri önlemek için gereklidir. Örneğin, sadece $hız(u,r)$, ve $mesafe(u,r)$ fonksiyonlarının en üst düzeye çıkarılması, aracı hedefe doğru ilerletmeye yeterli değildir. Benzer şekilde sadece $pozisyon(u,r)$ fonksiyonu en üst düzeye çıkarılırsa herhangi bir engel ile karşılaşıldığında bir işlem yapılmayacağından robot, hedefe doğru giden yolda ilk engel tarafından durdurulacaktır. Ancak her üç parametre de dahil edildiğinde ve ağırlık sabitleri ayarlandığında araç hedefine ulaşmaya doğru ilerlemeye devam ederken, dinamik sınırlamaların altında, olabildiğince hızlı bir şekilde çarpışmaları atlatacaktır [42].

BÖLÜM 4. ARAŞTIRMA VE BULGULAR

4.1. Adaptif Monte Carlo Lokalizasyonu

Tez çalışmasında kullanılan lokalizasyon yöntemi Adaptif Monte Carlo Lokalizasyonu'dur. Bu yöntem temelde parçacık filtresini kullanan bir yapı içermektedir. Lokalizasyon işleminde her bir parçacık robotun olası pozisyon ve yönünü temsil etmekte, parçacıkların örneklemesine ve önemine dayalı olarak yeniden örneklem kümesi oluşturulmaktadır [46]. Bir başka deyişle AMCL, Bölüm 3'te sunulan MCL algoritmasına göre daha spesifik olarak, parçacıkların sayısını dinamik bir yapıda uyarlamaktadır. Tez çalışmasında gerçekleştirilen otonom yol planlama işlemlerinde bu özelliğinden dolayı AMCL algoritması tercih edilmiştir.

AMCL algoritması, bir başlatma fazı ve iki arda arda tekrarlanan fazlardan oluşan tahmin ve güncelleme aşamasından oluşmaktadır [54]. Başlatma aşamasında, bir parçacık filtresi, muhtemel konumların tüm harita üzerinde eşit olarak dağıtılmış bir N dizisi üretmektedir. Her parçacık s^i bir x, y ve dönme değerine sahiptir $s^i = (\hat{x}, \hat{y}, \hat{\theta})$. Parçacıklar genellikle, sadece geçerli konumların dikkate alındığı, yani haritanın dışında veya duvarların içinde olamayacak şekilde başlatılmaktadır.

Birinci yinelemeli adım, önceki popülasyonun parçacıklarının, robotun hareket modeline, yani odometriye dayanarak hareket ettirildiği tahmin aşamasıdır. Daha sonra, güncelleme aşamasında, parçacıklar, her bir parçacık için robotun ölçüm olasılığına göre ağırlıklandırılır. Bu ağırlıklı parçacık kümesi göz önüne alındığında, yeni popülasyon, eski popülasyondaki parçacıkların ağırlıklı dağılımına göre seçilecek şekilde yeniden örneklenir. Bu iki faza ait ayrıntılı açıklamalara aşağıdaki başlıklarda yer verilmiştir.

4.1.1. Tahmin aşaması

Her hareketin ardından, her bir parçacığın pozisyonu, ajan olarak belirlenen parçacığın inancına göre güncellenmektedir. Örneğin, eğer robot 10 cm ileri hareket ederse, her bir parçacık, dönme yönüne doğru 10 cm hareket ettirilir. Böylece, robot $x_k = (x_k, y_k, \theta_k)$ durumundan $x_{k+1} = (x_{k+1}, y_{k+1}, \theta_{k+1})$ durumuna taşınırsa, parçacıklar Denklem 4.1'deki şekilde çevrilir:

$$\begin{bmatrix} \hat{x}_{k+1} \\ \hat{y}_{k+1} \\ \hat{\theta}_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{x}_k + \rho\cos(\hat{\theta}_k + \Delta\theta) \\ \hat{y}_k + \rho\sin(\hat{\theta}_k + \Delta\theta) \\ \hat{\theta}_k + \Delta\theta \end{bmatrix} \tag{4.1}$$

Ancak, hem $\rho = \sqrt{\Delta x^2 + \Delta y^2}$ hem de $\Delta\theta = \theta_k - \theta_{k+1}$ aktüatörler ve odometri hatalarından dolayı oluşacak gürültüler ile bozulmaktadır. Bu nedenle, robotun hareket modeli ne kadar doğruysa, tahmin aşamasının performansı o kadar iyi olur.

4.1.2. Güncelleme aşaması

Robot üzerinde bulunan alan tarayıcı bir sensör güncellemesinden sonra, her parçacık için sonraki ölçüm hesaplanmaktadır. Bu sayede, ölçülen sensör değerlerinin, robotun parçacık konumunda olması durumunda beklenen gerçek ortam konumu ile karşılaştırılmış olur. Yeni ağırlık w_k^i, Denklem 4.2'de gösterildiği gibi k zamanında parçacık pozisyonu x_k^i verilen gerçek sensör ölçümünün z_k olasılığıdır:

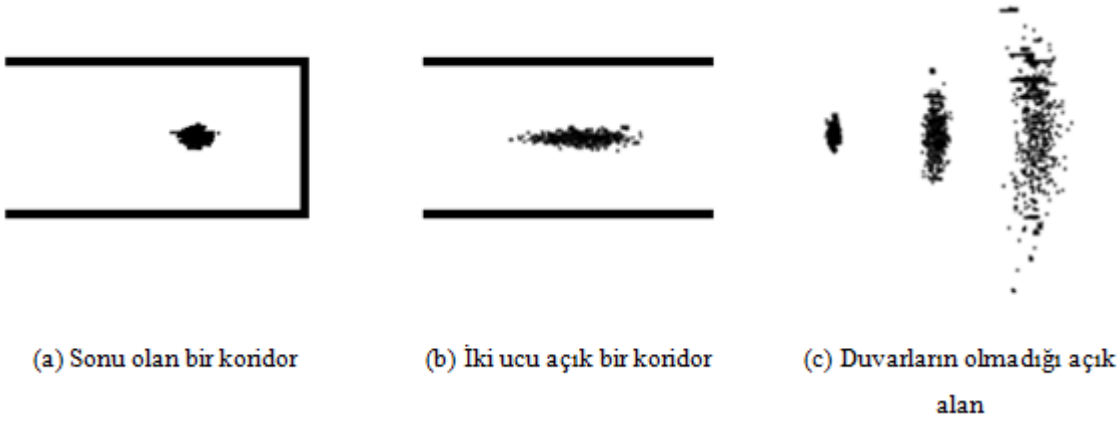
$$w_{k+1}^i = p(z_k | x_k^i) \tag{4.2}$$

w bir olasılık dağılımı olduğundan, her güncellemeden sonra, her bir parçacığın ağırlığı yeniden normalize edilmektedir.

Parçacık filtreleri, ilk durum bilinmediğinde konumu doğru şekilde tanımlamak için çok sayıda M örneğine ihtiyaç duymaktadır. Bununla birlikte, mevcut lokalizasyon doğrulandıkça, pozisyon değişikliklerini takip etmek için daha az parçacığa ihtiyaç

duyulmaktadır. Bu nedenle, numune sayısı pozisyon belirsizliğine bağlı olarak uyarlanabilmektedir. Tez çalışmasında gerekli olan minimum parçacık sayısını belirlemek amacıyla KLD-örnekleme yaklaşımı kullanılmıştır. Böylece robotun bir sonraki adımı ile örnek tabanlı yaklaşım arasındaki hata $1-\delta$ oranı ε'den küçük olmaktadır. Örnek sayısı Denklem 4.3'deki şekilde hesaplanabilir:

$$n = \frac{1}{2\varepsilon} X^2_{k-1,1-\delta} \tag{4.3}$$



Şekil 4.1. Parçacık dağılımlarının statik ortam durumlarına göre dağılımları [54]

Şekil 4.1.a'da görüldüğü üzere; eğer robotun hareket düzlemi, statik duvarlarla çevrili ise koridorun sonunda küçük bir varyansa sahip parçacık bulutu ile sonuçlanan iyi bir lokalizasyon oluşmaktadır. Şekil 4.1.b'deki gibi açık uçlu bir koridorda ise, sensör sadece kenarlar için geçerli okuma sağlayacağından, lokalize işlemi koridorun yönünde uzayan bir parçacık bulutu ile sonuçlanmaktadır. Ancak Şekil 4.1.c'deki gibi açık alanda, hiçbir sensör okuması olmadığı için sadece hareket modeline dayanan bir parçacık bulutu oluşmaktadır.

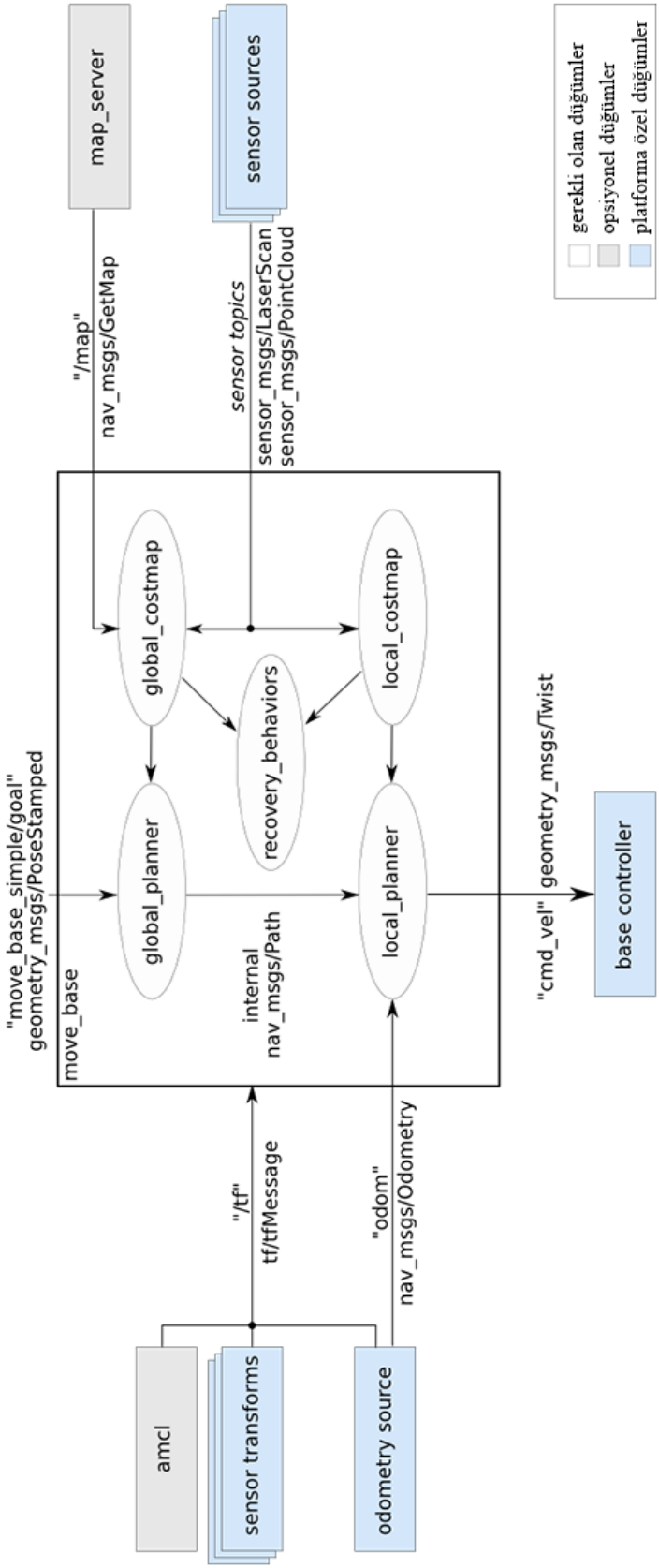
4.2. Navigation Yığın Yapısı

ROS işletim sisteminde navigasyon işlemi, bilinen bir harita üzerinde belirli bir hedef noktadan arzu edilen başka bir noktaya ulaşma aşmalarını içermektedir. Bu aşamaların gerçekleşmesi sırasında robotun tekerleklerinden alınan odometri verileri, ortam izlemesinde kullanılan sensör verileri ve çalışma ortamına ait bilinen harita kullanılmaktadır. Bu parametrelerin ilişkisel şemaları Şekil 4.2.'de sunulmuştur.

Robotların navigasyon işlemleri için Şekil 4.2.'de verilen yığın yapısından anlaşıldığı üzere;

- Ortama ait harita,
- Robot pozisyon bilgileri,
- Engel algılayıcı sensör verileri,
- Yol planlama ve sürüş işlemleri gereklidir.

Bu anlamada robot manipülasyonu sırasında gerçek zamanlı olarak önceden bilinen harita üzerindeki konumunu, tekerlerden alınan odometri verilerini ve planlanan yol üzerindeki engelleri algılayarak navigasyon işlemini gerçekleştirmektedir. Tez süresinde gerçekleştirilen deneysel çalışmalarda ROS işletim sistemine ait temel paketlerden olan *costmap_2d* kullanılmış.

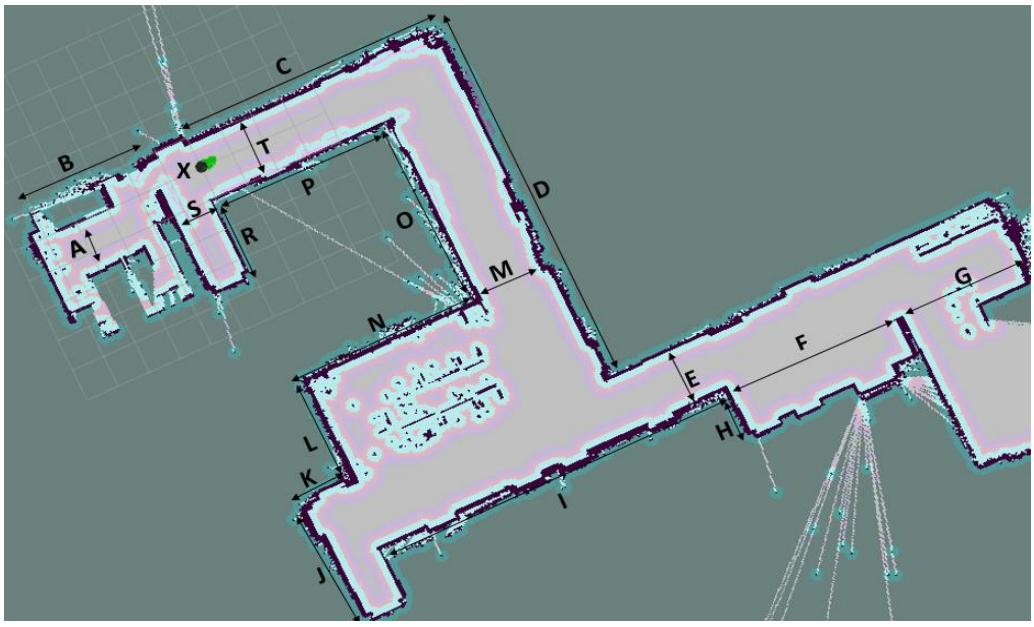


Şekil 4.2. ROS navigasyon yığın yapısı [55]

4.3. Ortama Ait Haritanın Elde Edilmesi

Navigasyon algoritmalarının çalışabilmesi için temel öncelik robotun çalışacağı ortama ait haritanın bilinmesidir. Bu kapsamda Bölüm 3'de anlatılan SLAM işlemi ile bir harita oluşturabilmek mümkündür. Bir başka yöntem ise önceden krokisi bilinen bir mekana ait kroki bilgisinin resim formatında (.png, .jpeg v.b) alınarak .yaml ve .pgm uzantılarına dönüştürülmesi ile sağlanmaktadır. Bahsi geçen her iki metot da tez çalışması sırasında uygulanmış ve sonuç alınmıştır. Dolayısıyla mobil robotun krokisi bilinen bir ortamlarda çalışabilmesi ve aynı zamanda böyle bir bilgi yoksa kendi haritasını çıkarabilmesi sağlanmıştır.

Tez çalışmasında, SLAM işlemi için, kendini merkez alarak 2 boyutlu bir kare harita oluşturan tek bir robot kullanılmıştır. Haritanın konfigürasyonuna bağlı olarak, robottan en uzak olan nokta yaklaşık 120 metre uzaklıktadır. Robot manipülasyonlarının gerçekleştirildiği ortama ait harita verileri ve karşılaştırmalı ölçüleri Şekil 4.3. ve Tablo 4.1.'de sunulmuştur.



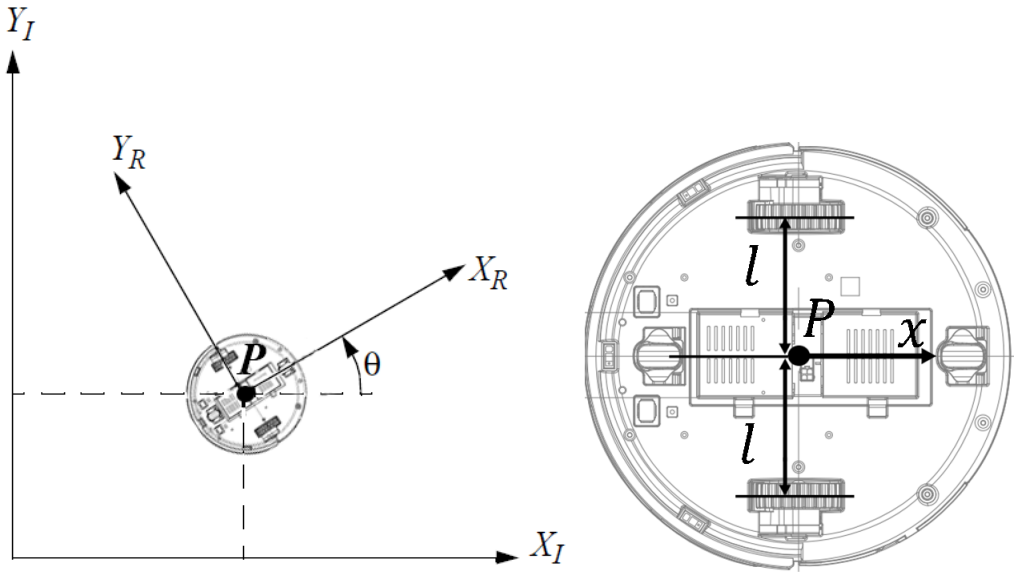
Şekil 4.3. Mekatronik Mühendisliği Bölümüne ait SLAM ile elde edilmiş kroki

Tablo 4.1. SLAM haritalama verilerinin gerçek ortam verileri ile karşılaştırılması

	Mesafe (m)			(%)	0-1
	Gerçek	SLAM	Fark	Yakınsama	Normalizasyon
A	1,56	1,53	0,03	98,08	0,829
B	3,88	3,8	0,08	97,94	0,815
C	7,11	7,85	-0,74	89,59	0
D	11,77	11,75	0,02	99,83	1
E	1,75	1,65	0,1	94,29	0,459
F	5,36	5,3	0,06	98,88	0,907
G	4	3,9	0,1	97,5	0,772
H	1,56	1,5	0,06	96,15	0,641
I	11,33	11,3	0,03	99,74	0,991
J	3,76	3,7	0,06	98,4	0,860
K	1,57	1,5	0,07	95,54	0,581
L	3,46	3,55	-0,09	97,4	0,763
M	2,28	2,25	0,03	98,68	0,888
N	5,69	5,65	0,04	99,3	0,948
O	6,15	6	0,15	97,56	0,778
P	5,92	5,65	0,27	95,44	0,571
R	2,41	2,5	-0,09	96,27	0,652
S	1,18	1,2	-0,02	98,31	0,852
T	1,8	1,75	0,05	97,22	0,745
X	Robot başlangıç konumu				

4.4. Mobil Robot Sisteminin Modellenmesi

Tez çalışmasında kullanılan Kobuki robot platformuna ait kinematik model hesaplanırken, robot, yatay bir düzlemde tekerlekler vasıtasıyla hareket eden katı bir kütle olarak modellenmiştir. Modelleme yapılırken teker aksları ve yönlendirme bağlantı elemanları gibi ek serbestlik ve esneklik dereceleri oluşturacak donanımlar görmezden gelinmiş ve sadece robot gövdesine ait kinematik modelleme gerçekleştirilmiştir. Robotun düzlemdeki konumunu belirlemek için düzlem ile robotun merkezi arasında Şekil 4.4.'deki gibi bir ilişki kurulmaktadır. Kullanılan diferansiyel tahrik robotunun iki tekerleği bulunmakta ve her bir tekerlek P merkez noktasından l kadar uzaklıktadır [56].



Şekil 4.4. Modellemeye ait referans noktalar

X_I, Y_I atalet referans noktaları ve P merkez noktasına göre X_R, Y_R robot referans noktaları iken robot pozisyonu Denklem 4.4'deki gibi tanımlanır;

$$\xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{4.4}$$

Robot referans noktaları ve atalet referansına bağlı olarak dikey dönüş matrisi ise Denklem 4.5'deki gibidir.

$$R(\theta) = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

$$\dot{\xi}_R = R(\theta)\dot{\xi}_I \tag{4.6}$$

Denklem 4.6'daki ileri hareket denklemi için X_R pozitif yöndedir ve her bir tekerleğin harekete katkısı;

$\dot{X_{R_1}} = r\dot{\varphi}_1$

$\dot{X_{R_2}} = r\dot{\varphi}_2$

$\dot{Y_R} = 0$

$\omega_{R_1} = \frac{r\dot{\varphi}_1}{l}$

$\omega_{R_2} = -\frac{r\dot{\varphi}_2}{l}$ şeklindedir.

Böylece hareket denklemi Denklem 4.7'deki gibi elde edilir.

$$\dot{\xi}_I = R(\theta)^{-1} \begin{bmatrix} r\dot{\varphi}_1 + r\dot{\varphi}_2 \\ 0 \\ \frac{r\dot{\varphi}_1 - r\dot{\varphi}_2}{l} \end{bmatrix} \tag{4.7}$$

$$R(\theta)^{-1} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

Sabit standart tekerleklerin dönme kısıtlarına ait matris Denklem 4.8'deki gibidir. Ayrıca mobil robotun tahriki açısından etkisi olmayan avare tekerler yüzeydeki sürtünmeleri önemsiz olduğu için modellemede göz ardı edilmiştir.

Hızın türevi olarak ξ_I vektörü $\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$ şeklinde ele alındığında $\dot{x}, \dot{y}$ ve $\dot{\theta}$ değerleri aşağıda belirtilen durumları ile hızın hesaplanabilmesi için $v = \sqrt{\dot{x}^2 + \dot{y}^2}$ denkleminde yerine konulduğunda Denklem 4.9 elde edilmektedir.

$\dot{x} = \frac{cos\theta}{2} r_l\varphi_l + \frac{cos\theta}{2} r_r\varphi_r$

$\dot{y} = \frac{sin\theta}{2} r_l\varphi_l + \frac{sin\theta}{2} r_r\varphi_r$

$\dot{\theta} = \frac{-r_l\varphi_l}{l} + \frac{r_r\varphi_r}{2l}$

$$v = r_l\varphi_l + r_r\varphi_2 \tag{4.9}$$

$$\dot{\theta} = \frac{-r_l\varphi_l}{l} + \frac{r_r\varphi_r}{l}$$

Lineer hareket için $\dot{\theta} = 0$ kabul edilirse;

$$0 = \frac{-r_l\varphi_l}{l} + \frac{r_r\varphi_r}{l}$$
$$r_l\varphi_l = r_r\varphi_r$$

$r_r = r_l = r_\omega$ ve $\varphi_r = \varphi_l = \varphi_\omega$ olduğuna göre lineer hız $v = V = r_\omega\varphi_\omega$ şeklinde olur ve radyan cinsinden gerekli olan tekerlek hızı ise $\varphi_\omega = \varphi_l = \varphi_r = \frac{V}{r_\omega}$ olur.

Diğer yandan açısal hızda $v = 0$ kabul edildiğinde;

$$0 = r_l\varphi_l + r_r\varphi_r$$
$$r_l\varphi_l = -r_r\varphi_r$$

$r_r = r_l = r_\omega$ olduğuna göre $-\varphi_l = \varphi_r = \varphi_\omega$ olur ve $\dot{\theta} = \omega = \frac{r_\omega\varphi_\omega}{l}$ şeklinde elde edilir. Radyan cinsinden gerekli olan açısal tekerlek hızı ise;

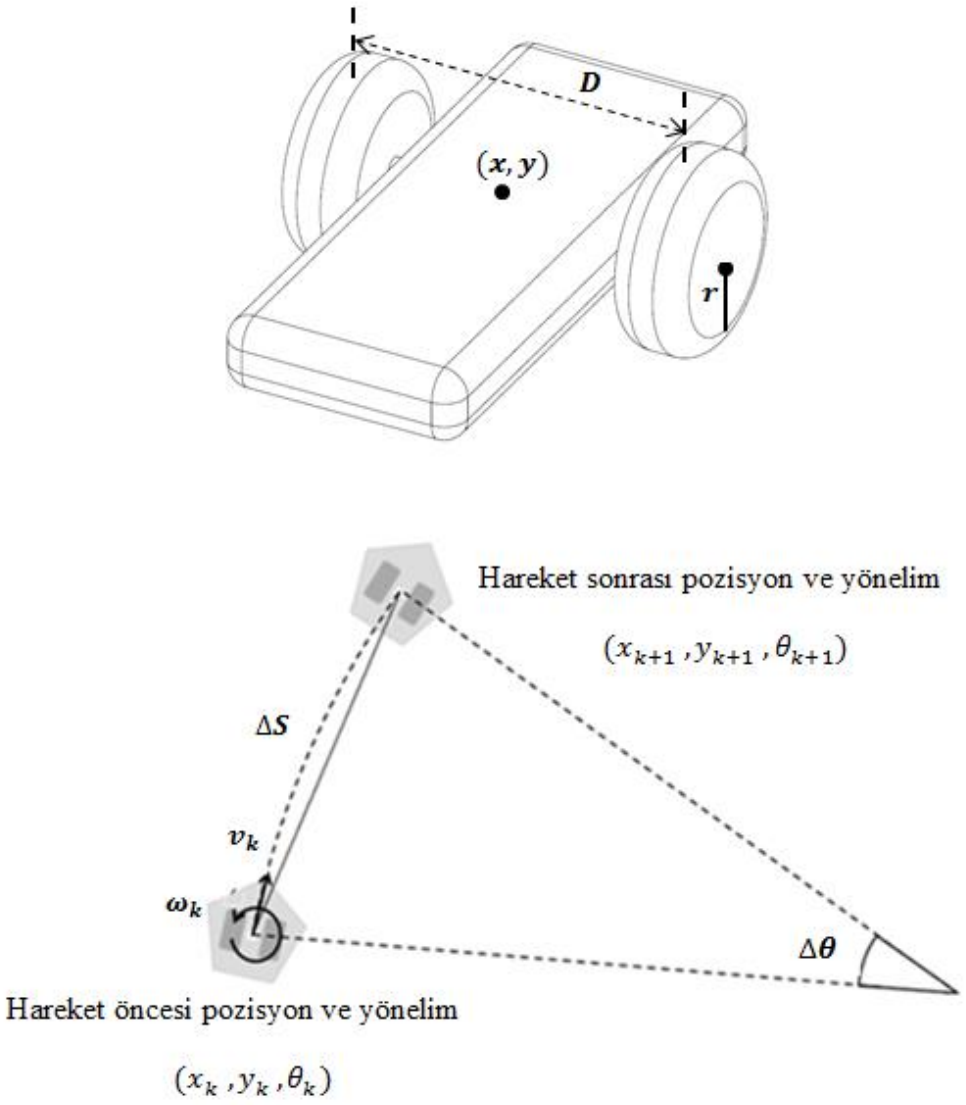
$$\varphi_\omega = \frac{\omega l}{r_\omega}$$
$$\varphi_r = \frac{\omega l}{r_\omega}$$
$$\varphi_l = \frac{-\omega l}{r_\omega}$$

şeklinde elde edilir.

4.5. Robot Pozisyon Bilgilerinin Elde Edilmesi

Robotların navigasyon sırasında kullandıkları en önemli parametreler arasında pozisyon ve konum bilgileri yer almaktadır. Otonom mobil robot çalışmaları incelendiğinde bu durumun, açık alan konumlama ve dış mekan konumlama şeklinde iki başlık altında toplandığı görülmektedir. Dış mekan konumlandırmalarında ortamın büyüklüğüne bağlı olarak GPS (Global Positioning System) tercih edilmektedir. Ofis, fabrika, okul, havaalanı gibi kapalı alan konumlandırma yapılacak mekanlarda ise GPS'in bir türevi olan DGPS (Differential GPS) kullanılmaktadır [57]. Fakat DGPS metotları yaklaşık 10 cm gibi bir hassasiyet ile çalışabilmekte ve maliyet açısından pahalı olması sebebiyle kapalı alan konumlandırmalarda tercih edilmemektedir. Bu problemlerin üstesinden gelmek için işaretleyici tanıma (marker recognition) ve iç mekan yeri tahmini (indoor location estimation) gibi çeşitli yöntemler oluşturulmuştur. Ancak bu yöntemler, maliyet ve doğruluk açısından genel kullanım için hala yetersiz kalmaktadır.

Günümüzde, tez çalışması içerisinde kullanılan mobil robotlara benzer hizmet robotları için en çok kullanılan kapalı pozisyonlama tahmini yöntemi göreceli bir pozisyonlama tahmini olan ölü hesaplar (dead reckoning) yöntemidir [58]. Bu yöntemde robotun hareket miktarı, tekerleğin dönüşüyle ölçülür. Bununla birlikte, tekerlek dönüşü ile hesaplanan mesafe ile gerçek hareket mesafesi arasında bir hata oluşmaktadır. Bu nedenle, IMU sensöründen gelen atalet bilgileri, hesaplanan değer ile gerçek değer arasındaki konum ve yönelim hatasını telafi ederek hatayı azaltmak için kullanılmaktadır [59]. Bu yönteme ait pozisyonlama bilgileri Şekil 4.5.'te sunulmuştur.



Şekil 4.5. Ölü hesaplara ait hesaplama parametreleri

Şekil 4.5.'te görüldüğü üzere tahrik tekerleri arasındaki mesafe D, tekerlerin yarıçapı ise r dir. T_e zamanı boyunca robotun kısa bir mesafe kat ettiğini varsayarsak sol ve sağ tekerleğin dönüş hızı (v_l , v_r) motor dönüş miktarları ile birlikte Denklem 4.10 ve Denklem 4.11 kullanılarak hesaplanmaktadır (güncel enkoder değeri E_lc , E_rc ve önceki enkoder değeri E_lp , E_rp).

$$v_l = \frac{(E_lc - E_lp)}{T_e} \times \frac{\pi}{180} \text{ (radyan/saniye)} \quad (4.10)$$

$$v_r = \frac{(E_r c - E_r p)}{T_e} \times \frac{\pi}{180} \text{ (radyan/saniye)} \quad (4.11)$$

Denklem 4.12 ve Denklem 4.13, sol ve sağ tekerleğin (V_l , V_r) hızını hesaplamakta, sol ve sağ tekerlek hızlarından da robotun doğrusal hızı (v_k) ve açısal hızı (ω_k) elde edilmektedir (Denklem 4.14 ve Denklem 4.15).

$$V_l = v_l \times r \text{ (metre/saniye)} \quad (4.12)$$

$$V_r = v_r \times r \text{ (metre/saniye)} \quad (4.13)$$

$$v_k = \frac{(V_r + V_l)}{2} \text{ (metre/saniye)} \quad (4.14)$$

$$w_k = \frac{(V_r - V_l)}{D} \text{ (radyan/saniye)} \quad (4.15)$$

Sonuç olarak elde edilen bu değerleri kullanarak robotun pozisyonu ($x_{(k+1)}$, $y_{(k+1)}$) ve yönü ($\theta_{(k+1)}$) elde edilmektedir (Denklem 4.16-4.19).

$$\Delta s = v_k T_e \qquad \Delta\theta = w_k T_e \quad (4.16)$$

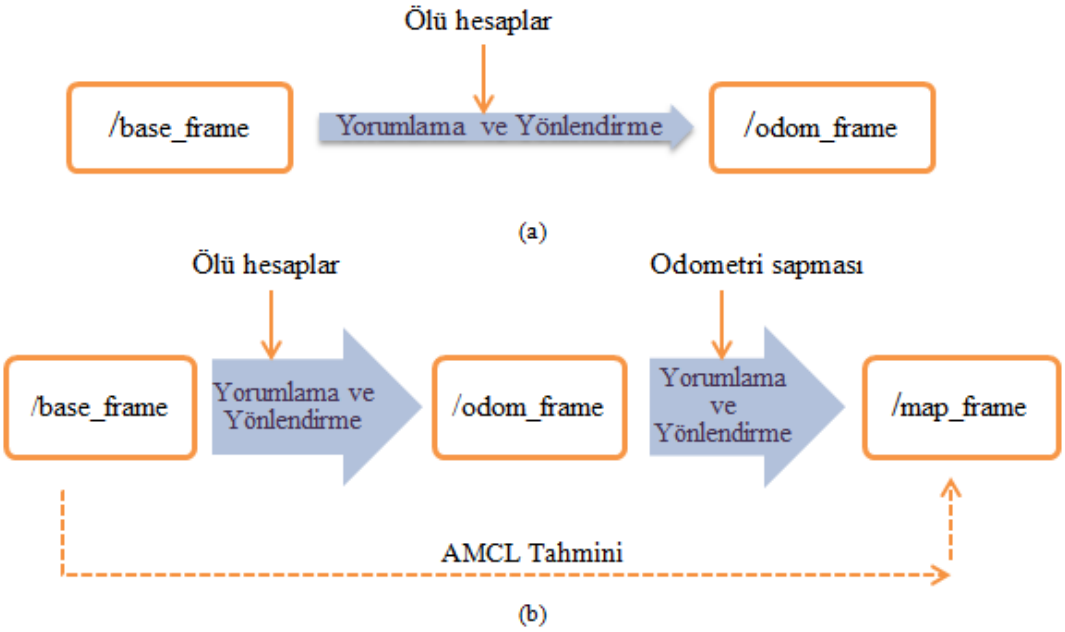
$$x_{(k+1)} = x_k + \Delta s \times cos\left(\theta_k + \frac{\Delta\theta}{2}\right) \quad (4.17)$$

$$y_{(k+1)} = y_k + \Delta s \times sin\left(\theta_k + \frac{\Delta\theta}{2}\right) \quad (4.18)$$

$$\theta_{k+1} = \theta_k + \Delta\theta \quad (4.19)$$

Şekil 4.6.a'da odometri verisinin tek başına kullanılması durumunda gerçekleştirilen lokalizasyon sunulurken Şekil 4.6.b'de ise odometri verisi ile beraber AMCL parametrelerinin bilinen harita üzerinde gerçekleştirdiği lokalizasyon şematize edilmiştir. Odometri verilerinin içerdiği sapma (özellikle kaygan zeminlerde oluşan patinaj) değerlerinin AMCL algoritması tarafından elimine edilmesi söz konusudur.

Tez çalışmasında, özellikle bilinen harita verisi üzerinden bir sonraki adımın tahminini içeren AMCL algoritması sayesinde, ölü hesaplar yönteminin çıkış değerleri optimize edilerek mobil robotun daha hassas pozisyonlaması sağlanmıştır.



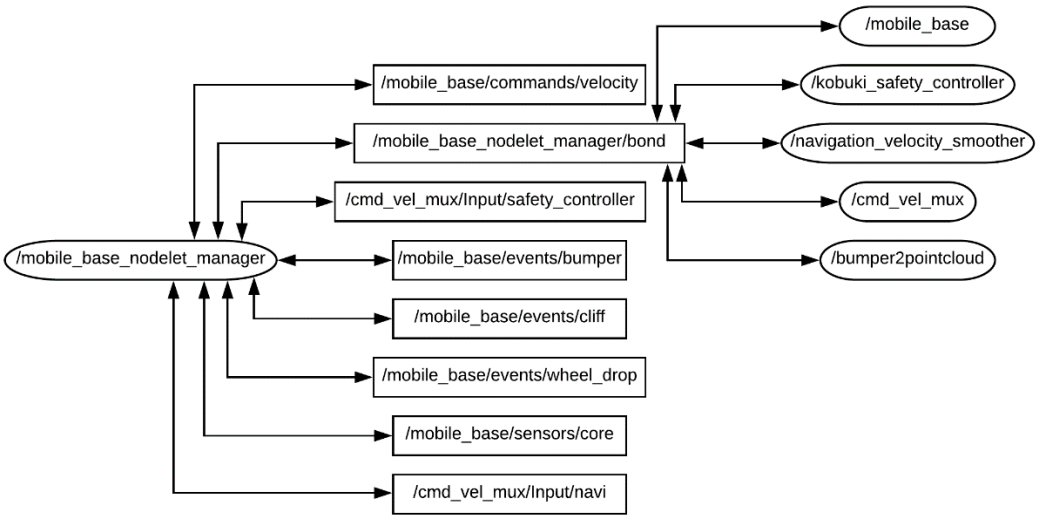
Şekil 4.6. Odometri ve AMCL lokalizsyonunun şematik karşılaştırması

4.6. Engel Algılayıcı Sensör Verileri

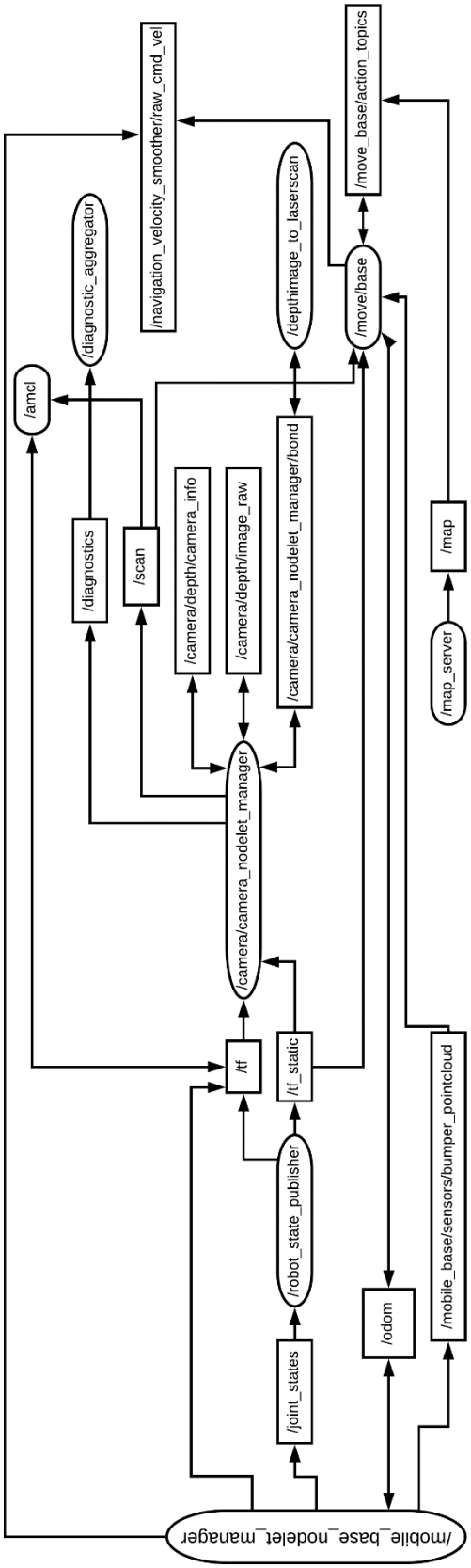
Navigasyonun tam anlamıyla gerçekleşebilmesinde bir diğer önemli husus ise duvarlar ve nesneler gibi engellerin olup olmadığını anlamaktır. Bu işlemler için mesafe ve görüş sensörleri gibi çeşitli sensör tipleri kullanılmaktadır. Mesafe sensörü olarak, lazer tabanlı mesafe sensörleri (LDS, LRF, LIDAR), ultrasonik sensörler ve kızılötesi mesafe sensörleri kullanır. Görüş sensörü olarak, stereo kameralar, mono kameralar ve çok yönlü kameralar kullanılmaktadır. Günümüzde robotların engellerden kaçınımlarını sağlamak amacıyla derinlik kamerası olarak RealSense, Kinect, AsusXtion gibi derinlik kameraları yaygın olarak tercih edilmektedir. Tez çalışmasında da bu RGB-D derinlik sensörlerinden birisi olan Kinect kamera kullanılmıştır. Navigasyon boyunca gerçek zamanlı olarak elde edilen kamera görüntüleri sayesinde statik ve dinamik engellerin algılanması sağlanmıştır.

4.7. Yol Planlama ve Sürüş İşlemleri

Haritası bilinen bir ortamda, ortamı algılayacak sensörler ile donatılmış ve doğru konumda pozisyonlanmış bir robotun navigasyonu için önemli özelliklerden biride, hedefe en uygun rotayı hesaplamak ve hesaplanan bu rotayı takip edebilmektir. Bu işlemlere yol arama ve planlama denilmekte ve A* algoritması, Potansiyel Alan Algoritması, Parçacık Filtresi, RRT gibi çeşitli algoritmalar ile gerçekleştirilebilmektedir. Bu algoritmaların birbirlerine göre üstün ve zayıf yönleri bulunmaktadır. Tez çalışmasında olasılıksal yol planlama yöntemlerinden birisi olan ve parça filtre kullanarak parçacıklarını tayin eden AMCL algoritması tercih edilmiştir. Diğer yol planlayıcılardan farklı olarak Dinamik Pencere yaklaşımı ile güçlendirilen ve parçacıkları mobil robotun hareketi sırasında çevresel etkenlere bağlı olarak dinamik bir yapıda güncelleyebilen yol planlama işlemleri için gerekli olan ROS düğümleri ve aralarındaki ilişki Şekil 4.7.'de şematize edilmiştir. ROS araçlarından birisi olan rqt_graph kullanılarak elde edilen bu düğüm göstergesi robotun navigasyon sırasında ROS-Master tarafından yayımlanan düğümlerini ve aralarındaki ilişkiyi göstermektedir. Esasen bu grafiklerin oluşturulması ve gerçek zamanlı izlenmesi oluşabilecek hataların takibi açısından önem arz etmektedir.



Şekil 4.7. Navigasyon esnasında kullanılan düğümler ve aralarındaki ilişki



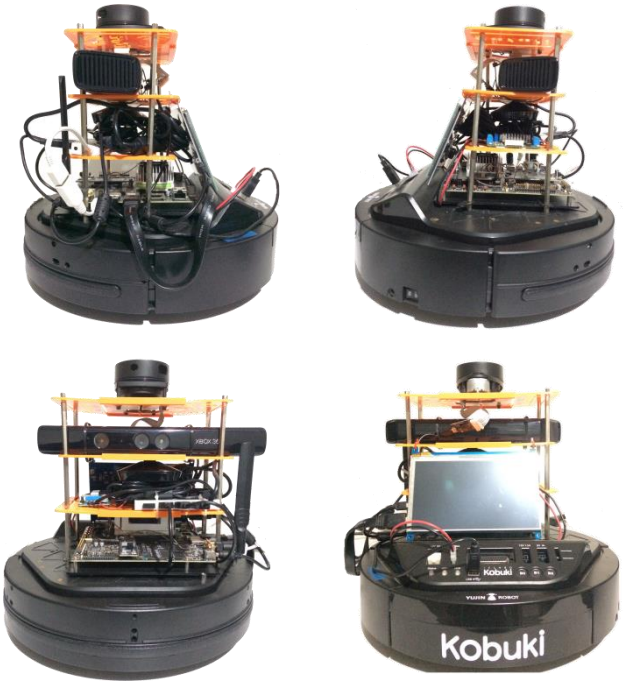
Şekil 4.7. (Devamı)

BÖLÜM 5. TESTLER VE GENEL DEĞERLENDİRME

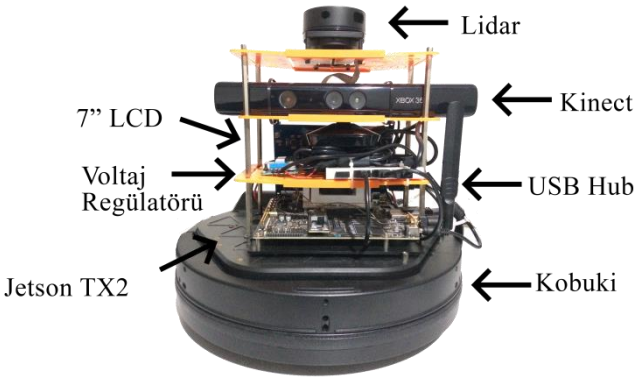
Tez çalışmasının bu bölümünde, gerçekleştirilen yazılımsal ve donanımsal iyileştirmeler ve çalışmaya ait kabuller ile yapılan test sonuçlarına yer verilmiştir. Kobuki robot platformu kullanılarak gerçekleştirilen tez çalışmasında, mevcut platform çeşitli donanımlar ile desteklenmiş ve geliştirme aşamasında, kullanılan ek donanımlar Tablo 5.1.'de sunulmuştur. Bu ek donanımlar ile beraber robot platformu yazılımsal açıdan da iyileştirilmiş ve Turtlebot platformuna dönüştürülmüştür. Yapılan donanımsal iyileştirmeler sonucunda elde edilen robot donanımına ait bilgiler Tablo 5.1.'de, oluşan donanım ise Şekil 5.1. ve Şekil 5.2.'de verilmiştir.

Tablo 5.1. Robot platformu için kullanılan ek donanımlar

Ek donanım	Adet	Kullanım amacı
NVIDIA® Jetson TX2	1	ROS ve diğer algoritmaların çalıştırılması
RPLIDAR A1 360^0 lazer alan tarayıcı	1	Eş Zamanlı Konum Belirleme ve Haritalama (SLAM)
Kinect V1 Model 1473 RGB-D Derinlik sensörü	1	Dinamik engellerin algılanabilmesi
Waveshare 7 inç dokunmatik ekran	1	Gerçek zamanlı olarak işlemlerin izlenmesi
LTC1871 Step Up Ayarlanabilir Voltaj Regülator Kartı - 3,5-30V 10A	1	Kontrol kartının ve ek donanımların beslenmesi
USB 3.0 – 4 port çoklayıcı	1	Ek donanımların kontrol kartı ile haberleşmesi
Vidalı yükseltme parçaları	16	Ek donanımların taşınması
3D baskı parçaları	12	Ek donanımların taşınması



Şekil 5.1. Mobil robotun sağ, sol, ön ve arka görünüşleri



Şekil 5.2. Mobil robota ait ek donanımların gösterimi

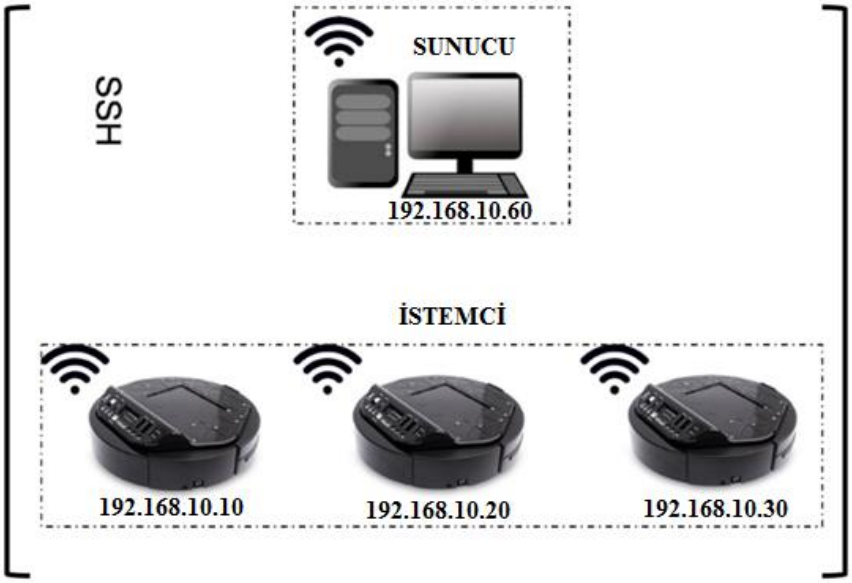
Geliştirilen robot platformu ile gerçekleştirilen otonom navigasyon işlemlerinin yürütülmesinde kullanılan AMCL algoritması üzerinde bir takım parametre kısıtlamaları gerçekleştirilmiştir. Parçacık filtresi kullanan bu algoritmanın

navigasyonu etkileyen temel parametreleri, yapılan deneysel çalışmalar için Tablo 5.2.'de sunulduğu şekliyle kabul edilmiştir.

Tablo 5.2. Deneysel çalışmalara ait kısıtlar

Değer	Kısıt	Aralık
Maksimum parçacık sayısı	2000	0-10.000
Minimum parçacık sayısı	500	0-1.000
Gerçek dağılım ve tahmini dağıtım arasındaki maksimum hata (kld_err)	0,05	0-1
Tahmini dağılımdaki hatanın kld_err'den daha az olması olasılığı (kld_z)	0,99	0-1
Yayınlanan dönüşümü güncelleştirmek için gereken zaman	1sn	0-2 (sn)
Robot hareket modeli	Diferansiyel	-
Yayınlanan global harita için güncelleştirme zamanı	0,5 sn	0-1 (sn)
Yayınlanan lokal harita için güncelleştirme zamanı	0,5 (sn)	0-1 (sn)
Global harita için kenar şişirme oranı	0,5	0-5
Lokal harita için kenar şişirme oranı	0,5	0-5
Maksimum doğrusal hız	0,8 m/sn	0-1 m/sn
Maksimum açısal hız	2,4 m/sn	0-3,14 m/sn
Maksimum doğrusal ivmelenme	1,0 m/s^2	0-10 m/s^2
Maksimum açısal ivmelenme	2 m/s^2	0-10 m/s^2

Mobil robotun otonom navigasyonu için gerekli olan global ve lokal yol planlama bilgilerinin gerçek zamanlı aktarımı için tez çalışmasında kablosuz bir ağ oluşturulmuş ve bilgiler bu ağ üzerinden aktarılmıştır. SSH protokolü ile çalışan bu ağ üzerinde sunucu-istemci mimarisi kullanılmış ve sabit IP adresleri ile etiketlenen robotlara erişim sağlanmıştır. Robotların eş zamanlı olarak aynı ağ üzerinden tek bir çalışma istasyonu (sunucu) tarafından yönlendirmesi sağlanırken, yine aynı ağ üzerinden robotların sunucuya anlık konum bilgilerini aktarılmıştır. Oluşturulan bu çift yönlü haberleşme ağına ait şematik yapı Şekil 5.3.'de verilmiştir.



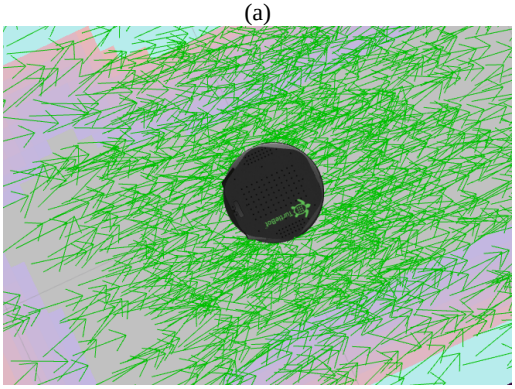
Şekil 5.3. Haberleşme ağ yapısı

5.1. Engelsiz Ortamda Gerçekleştirilen Navigasyon

Statik veya dinamik engellerden bağımsız bir ortamda gerçekleştirilen navigasyon işlemine ait başlangıç konumunun parçacık dağılımları ve mobil robotun gerçek ortam konumlanmasına ait görüntüleri Şekil 5.4.'te gösterilmiştir. Hedef konumunun parçacık dağılımları ve mobil robotun gerçek ortam konumlanmasına ait görüntüleri ise Şekil 5.5.'te gösterilmiştir. 10 tekrar ile gerçekleştirilen deney sonuçlarına ait ortalama rota verileri ise gerçek zamanlı simülasyon görüntüleri üzerinden Şekil 5.6.'da sunulmuştur. Harita üzerinde $x = 0{,}7$ ve $y = 0{,}9$ konumunda $\theta = 0^0$ açısı konumlanmış bir mobil robotun $x = 5$ ve $y = 8{,}6$ konumuna yönlendirilmesi sağlanmıştır. Bu iki nokta arasında yaklaşık olarak 9 metre yol kat edilmektedir.

Engelsiz ortamda gerçekleştirilen bu 10 tekrar neticesinde mobil robotun hedef noktaya ortalama 20 saniyede vardığı gözlemlenmiştir. Başlangıç için belirlenen 2000 parçacık hedef noktaya vardığında algoritma tarafından geçen süre zarfında 572'ye indirgenmiş ve robot hedefe %2 hassasiyet ile doğru konumlanmıştır.

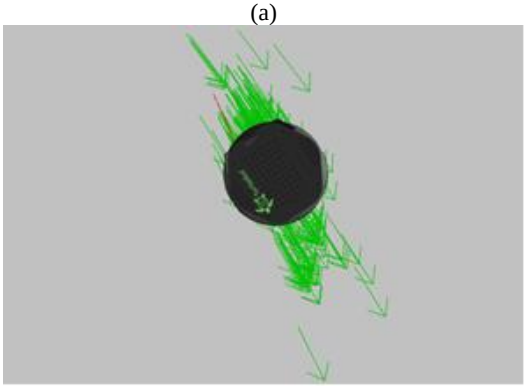
(a)



(b)

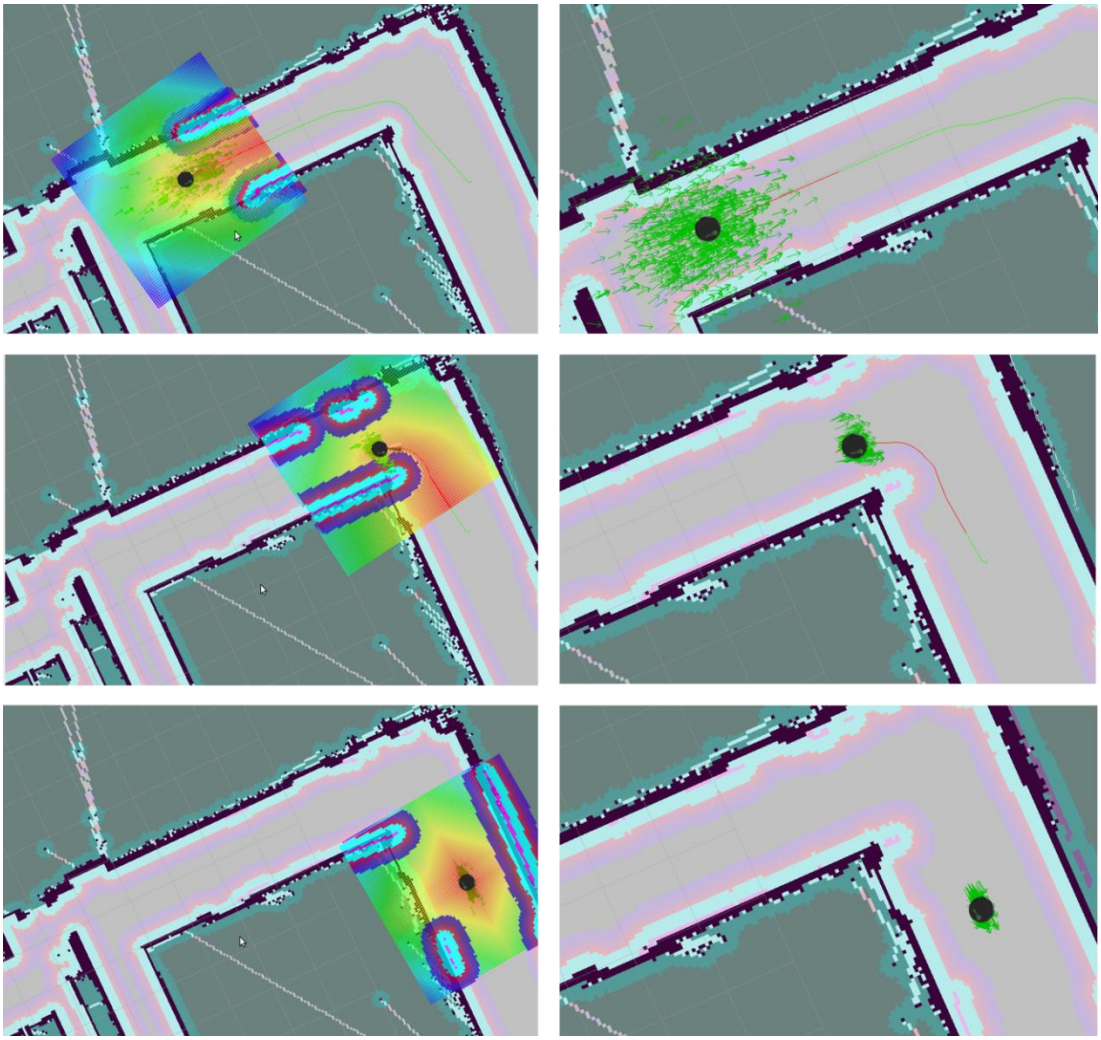
Şekil 5.4. Engelsiz ortam navigasyonu için başlangıç (a) Gerçek ortam konumlanması (b) Parçacık dağılımı

(a)



(b)

Şekil 5.5. Engelsiz ortam navigasyonu için hedef (a) Gerçek ortam konumlanması (b) Parçacık dağılımı

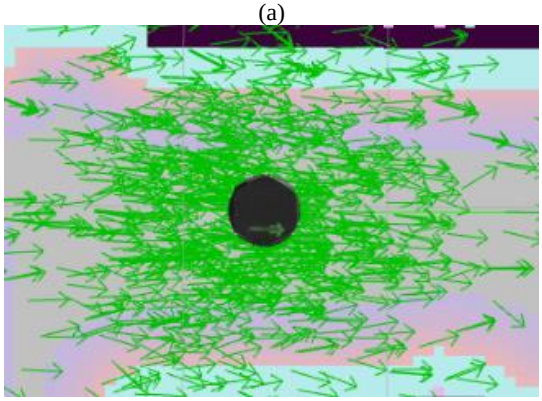


Şekil 5.6. Engelsiz ortamda gerçekleştirilen navigasyon işlemine ait ortalama rota görüntüleri ve parçacık dağılımları

5.2. Statik Engelli Ortamda Gerçekleştirilen Navigasyon

Statik engele sahip ortamda gerçekleştirilen navigasyon işlemine ait başlangıç ve hedef konumlarının parçacık dağılımları, gerçek ortam görüntüleri ile birlikte Şekil 5.7. ve Şekil 5.8.'de verilmiştir. 10 tekrar ile gerçekleştirilen deney sonuçlarına ait ortalama rota verileri ise gerçek zamanlı simülasyon görüntüleri üzerinden Şekil 5.9.'da sunulmuştur.

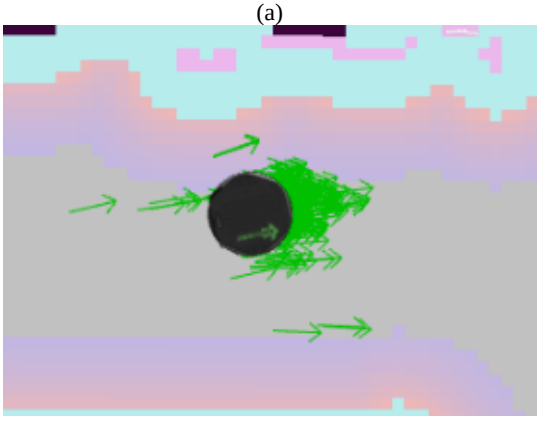
(a)



(b)

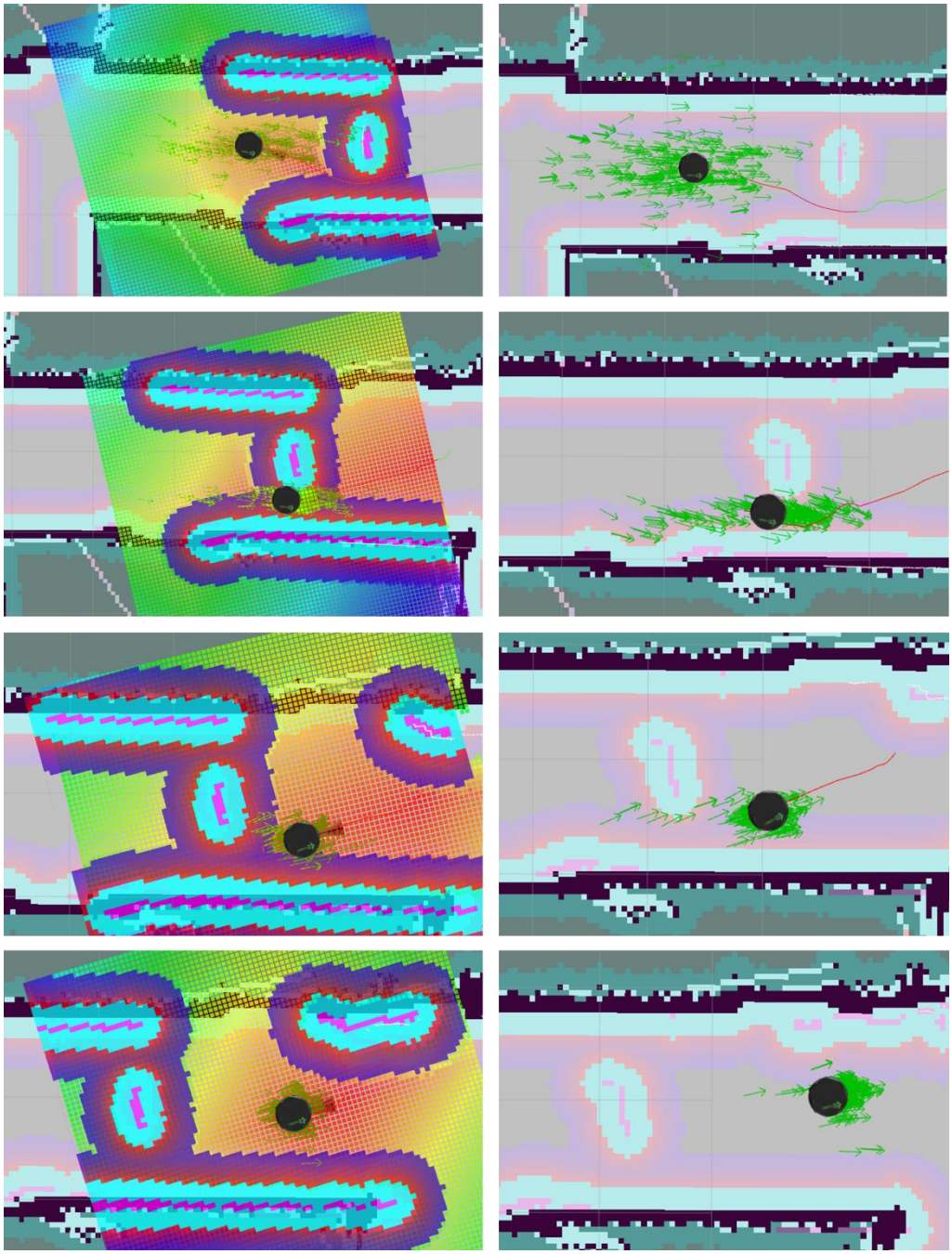
Şekil 5.7. Statik engelli navigasyon için başlangıç (a) Gerçek ortam konumlanması (b) Parçacık dağılımı

(a)



(b)

Şekil 5.8. Statik engelli navigasyon için hedef (a) Gerçek ortam konumlanması (b) Parçacık dağılımı



Şekil 5.9. Statik engelli ortam navigasyonu için ortalama rota görüntüleri ve parçacık dağılımları

Ayrıtları 40x40x75 olan statik bir engel kullanılarak gerçekleştirilen deneylerde 10 tekrar neticesinde mobil robotun 5,2 metrelik mesafeyi ortalama 11,65 saniyede aldığı gözlemlenmiştir. Başlangıç için belirlenen 2000 parçacık hedef noktaya vardığında algoritma tarafından geçen süre zarfında 624'e indirgenmiş ve robot hedefe %1,5 hassasiyet ile doğru konumlanmıştır. Başlangıç noktasından 260 cm uzaklıkta olan engel 20 cm mesafe kala tespit edilmiş ve yaklaşık 12 derecelik bir açı

ile kaçınım sağlanmıştır. Engelden kaçınım esnasında yürütülen manevra hareketleri Şekil 5.10.'da sunulmuştur.

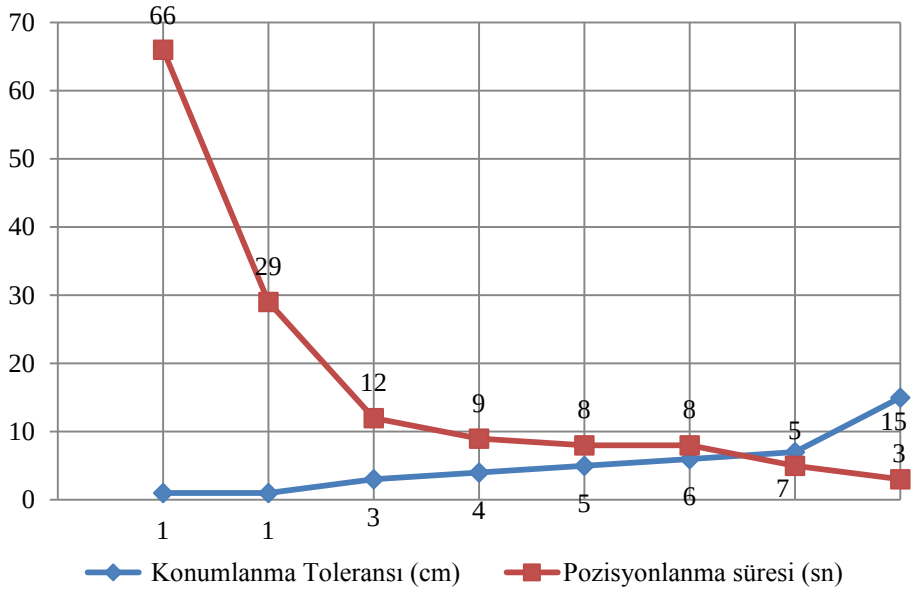
Şekil 5.10. Statik engelli ortamdaki yol planlamaya ait yol planlama adımları

5.3. Konumlama Hassasiyetini Belirlenmesi

Robotik uygulamalarda önemli parametrelerden biriside konumlama hassasiyettir. Robotların belirlenen hedefe varıldığında istenilen pozisyona en yakın konumlamayı doğru bir pozisyonlanma ile gerçekleştirmesi beklenmektedir. Bu kapsamda tez çalışmasında robot navigasyonu için uygulanan algoritmaların konumlanma hassasiyet testleri yapılmıştır. Test işlemlerinde pürüzsüz bir zeminde Dinamik Pencere yaklaşımı kullanılarak gerçekleştirilen pozisyonlamalarda, konumlama hassasiyetleri ölçülmüş ve ölçüm sonuçları Tablo 5.3.'te verilmiştir.

Tablo 5.3. Konumlanma hassasiyetlerine göre pozisyonlanma süreleri

Deney	Hedef (x, y) Konumlanma Toleransı (cm)	Pozisyonlanma süresi (sn)
1	1	66
2	2	29
3	3	12
4	4	9
5	5	8
6	6	8
7	7	5
8	15	3



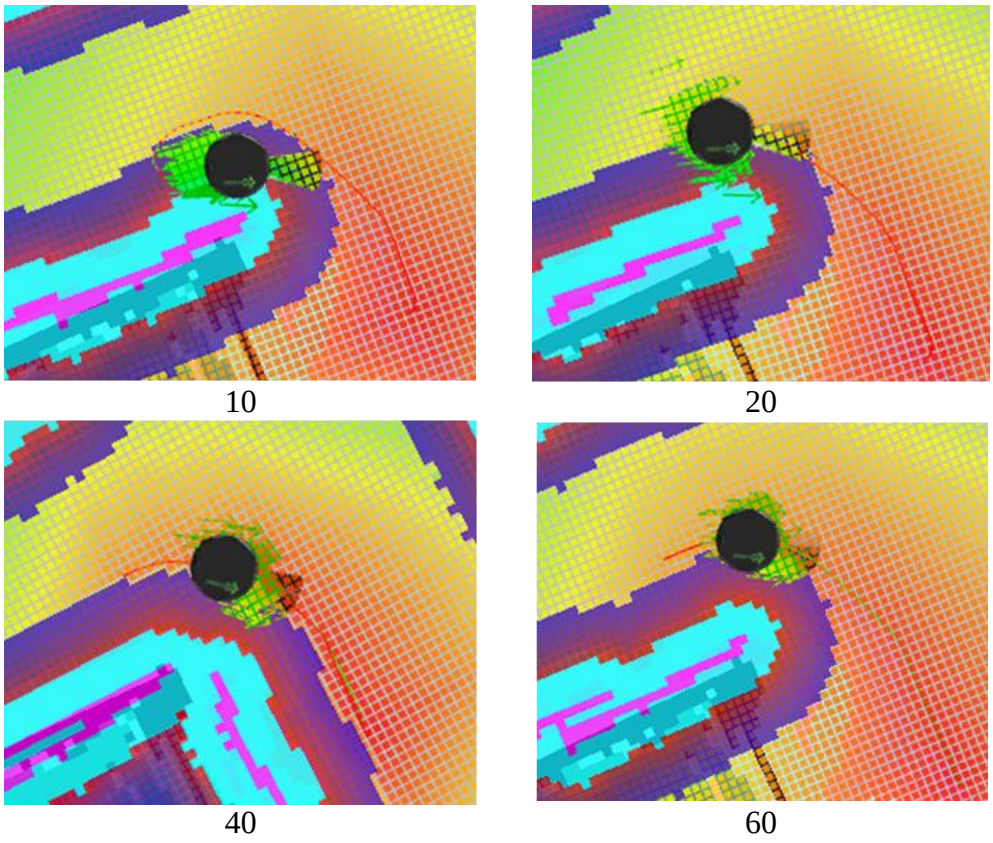
Şekil 5.11. Konumlanma hassasiyetlerine göre pozisyonlanma süreleri

Tablo 5.3. ve Şekil 5.11.'de verilen sonuçlar incelendiğinde görülmektedir ki robotun hedef noktaya vardığı zaman 1 cm gibi bir hassasiyet ile doğru pozisyona oturması 1 dakikadan uzun bir süre almaktadır. Bu durum elbette hedef noktada hareketli bir şekilde bulunma süresi için olumsuzluk teşkil etmektedir. Pozisyonlanma hassasiyeti 15 cm olarak belirlendiğinde ise robotun pozisyona oturma süresi 3 sn sürmektedir. Bu durum süre açısından olumlu fakat hedef noktaya varış mesafesi açısından olumsuzdur. Hedef nokta konumlama toleransı 7 cm değerinde bir hassasiyet ile gerçekleştirildiğinde ise robotun yaklaşık 5 saniyelik bir süre sonunda hedef noktada pozisyonlandığı görülmektedir. Endüstriyel uygulamalar incelendiğinde en düşük seviyede enerji tüketimi ve hedef noktada gerçekleştirilecek olası operasyon için 5 saniye olan bu sürenin optimal düzeyde olduğu sonucuna varılmıştır.

Aynı zamanda yapılan testlerde hedef nokta konumlama toleransının 100 cm olarak belirlenmesi durumunda robotun hedefe 120 cm uzakta pozisyonlama hatası ile varış sağlamadığı gözlemlenmiştir. Benzer şekilde bu oran 200 cm olarak belirlendiğinde ise pozisyonlama hata oranı artmakta ve 238 cm uzaklıkta hedefe varmadan konumlandırmanın sonuçlandığı görülmüştür.

5.4. Yol Planlama için Optimum Maliyet Parametrelerinin Belirlenmesi

Tez çalışmasında, robotun lokal yol planlaması için AMCL algoritması ile beraber Dinamik Pencere yaklaşımı kullanılmıştır. AMCL algoritmasındaki parçacık dağılımları DWA ile beraber lokal yol planlama esnasında engellerden kaçınımı sağlamaktadır. Bu iki yaklaşımın ortak kullanımı ile birlikte, belirlenen ölçülerde robotun etrafını saran bir kare matris ile robotun dinamik hareketi esnasında yapısal bir tarama işlemi gerçekleştirilmektedir. Statik engellerin mevcut harita üzerindeki şişirme (genişletme) oranlarına bağlı olarak, mobil robotun kendisine verilmiş olan yola ne kadar yakın kalması gerektiğiyle ilgili ağırlık katsayısı (*path_distance_bias*) üzerinde değişikliklerine giderek bu parametre analiz edilmiş ve deney sonuçlarına Şekil 5.12.'de yer verilmiştir.

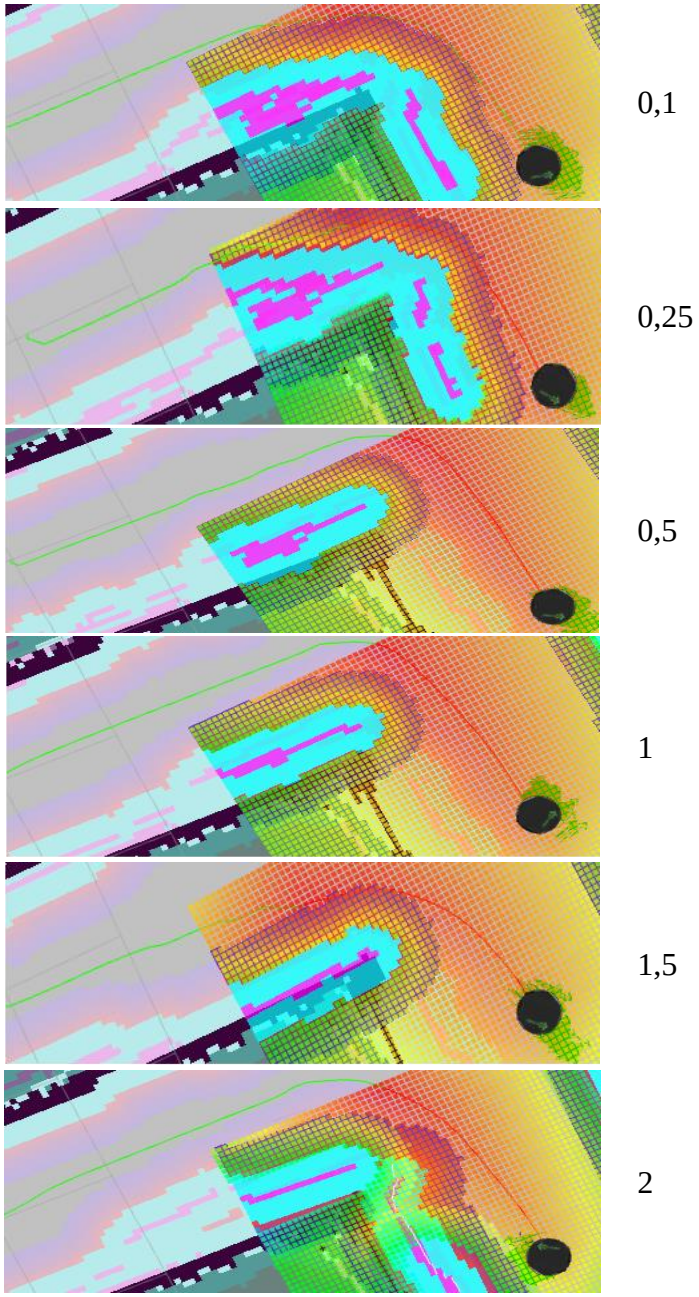


Şekil 5.12. DWA Planlayıcı için uygulanan *path_distance_bias* parametreleri ve sonuçları

Şekil 5.12. incelendiğinde robotun hedefe giden yolda güvenli bir şekilde en kısa yoldan varabilmesi için Dinamik Pencere planlayıcısındaki yaklaşım katsayısı için en ideal değerin 40 olduğu ortaya çıkmıştır. Bu katsayının 8-20 bandı arasına azaltılması sonucunda robot manipülasyonu emniyet sınırlarının dışına çıkmaktadır. Diğer taraftan 45-60 bandı arasına artırılması sonucunda ise yol maliyeti açısından olumsuzluk teşkil etmekte ve hedefe giden yol, dolayısıyla sarf edilen enerji artmaktadır.

Yol maliyeti açısından önemli olan bir diğer parametre ise Adaptif Monte Carlo algoritmasına ait transfer toleransıdır (*amcl_transform_tolerance*). Bu parametrenin belirlenmesi sırasında gerçekleştirilen test sonuçları Şekil 5.13.'te verilmiştir. Yapılan testler sonucunda transform toleransı 0,1 ile 0,25 saniye arasında tutulduğunda robot en kısa köşe dönüşlerini tercih etmekte fakat dinamik pencere yakınsama kısıtlarının sınırlarında hareket etmektedir. Bu değer 1 ile 2 saniye

arasında tutulduğunda ise hareket planlaması dinamik pencere kısıtları açısında problem teşkil etmemekte ancak bu defa robotun geniş bir açıda köşe dönüşleri sağladığı izlenmiştir. Dolayısıyla yol maliyeti açısından durum incelenecek olunursa yapılan testler sonucunda robotun hedefe giden yolda güvenli bir şekilde en kısa yoldan varabilmesi için optimum transfer toleransı 0,5 saniye olarak belirlenmiştir.

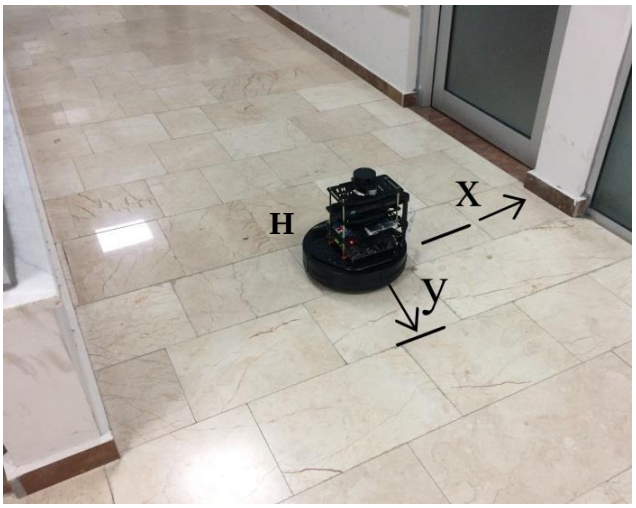


Şekil 5.13. AMCL transform toleransı parametreleri ve sonuçları

5.5. Tekrarlanabilirlik Testleri

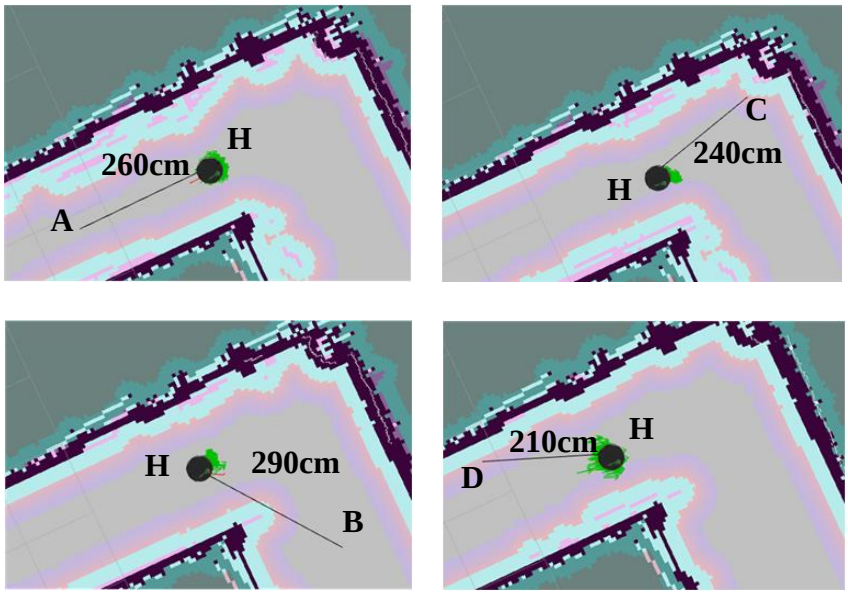
Mobil robot çalışmalarında önemli olan noktalardan biri de robotların belirli hassasiyetlerle farklı noktalardan aynı hedefe ulaşabilmesidir. Tekrarlanabilirlik olarak adlandırılan bu gereksinim robotların saha çalışmalarında, belirlenen hedef noktalara olan konumlanma hassasiyetlerinin bir parçasını oluşturmaktadır. Hedef noktaya olabildiğince az sapma değerlerinde varış sağlamak, hedef noktada gerçekleştirilecek operasyon için önem taşımaktadır.

Tekrarlanabilirlik testlerinin yürütülmesi amacıyla Şekil 5.14.'te 'H' ile gösterilen hedef noktaya, 4 farklı noktadan parçacık filtre tabanlı rota oluşturulmuştur.



Şekil 5.14. Tekrarlanabilirlik gerçek ortam hedef noktası

Açısal ve doğrusal rota bilgileri içeren ve yaklaşık 3 metre uzaklıkta bulunan başlangıç noktalarından hedef noktaya ayrı ayrı testler gerçekleştirilmiştir. Testler sonuncunda ortalama değerlere sahip, gözlemlenen 4 farklı sonuca (A-B-C-D) ait başlangıç noktaları ve hedefe olan uzaklık bilgileri Şekil 5.15.'te sunulmuştur.



Şekil 5.15. Tekrarlanabilirlik testlerine ait başlangıç noktaları ve hedef noktaya olan uzaklıkları

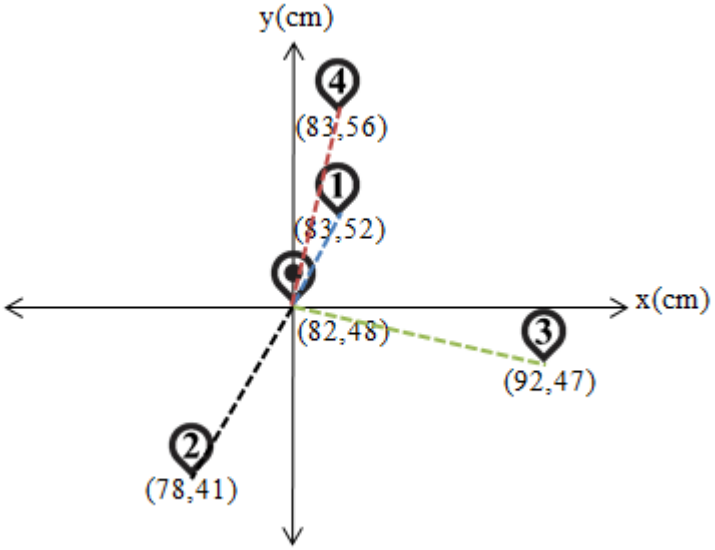
Tekrarlanabilirlik testleri sonucunda robotun istenilen hedef noktaya yaklaşımları Şekil 5.14.'de gösterilen iki nokta referans alınarak ölçümler gerçekleştirilmiştir. Bu referans hedef noktaya göre oluşan x, y sapma değerleri ise Tablo 5.4. ve Şekil 5.16.'de verilmiştir. Engelsiz ortamda gerçekleştirilen 10 tekrarlı deney sonucunda robotun hedef noktaya minimum 4 cm, maksimum 10 cm uzaklıkta konumlandığı tespit edilmiştir.

Tablo 5.4. Konumlanma hassasiyetlerine göre sapma değerleri

Deney	x	y	θ	x fark	y fark	Sapma
1 (A-H)	83	52	-1,5	-1,00	-4,00	4,12
2 (B-H)	78	41	-6,2	4,00	7,00	8,06
3 (C-H)	92	47	-2,4	-10,00	1,00	10,05
4 (D-H)	83	56	-1,7	-1,00	-8,00	8,06
Referans hedef	82	48				

Testler sonucunda görülmektedir ki açısal değerler içeren navigasyonlarda hedef nokta yönelim açısı olan θ değeri, doğrusal navigasyonlara göre daha geniş açılarda

olmaktadır. Ayrıca rota üzerinde oluşabilecek konumlama sapmaları da yine açısal değere bağlı olarak değişiklikler göstermektedir.



Şekil 5.16. Tekrarlanabilirlik test sonuçlarına ait (x, y) konum grafiği

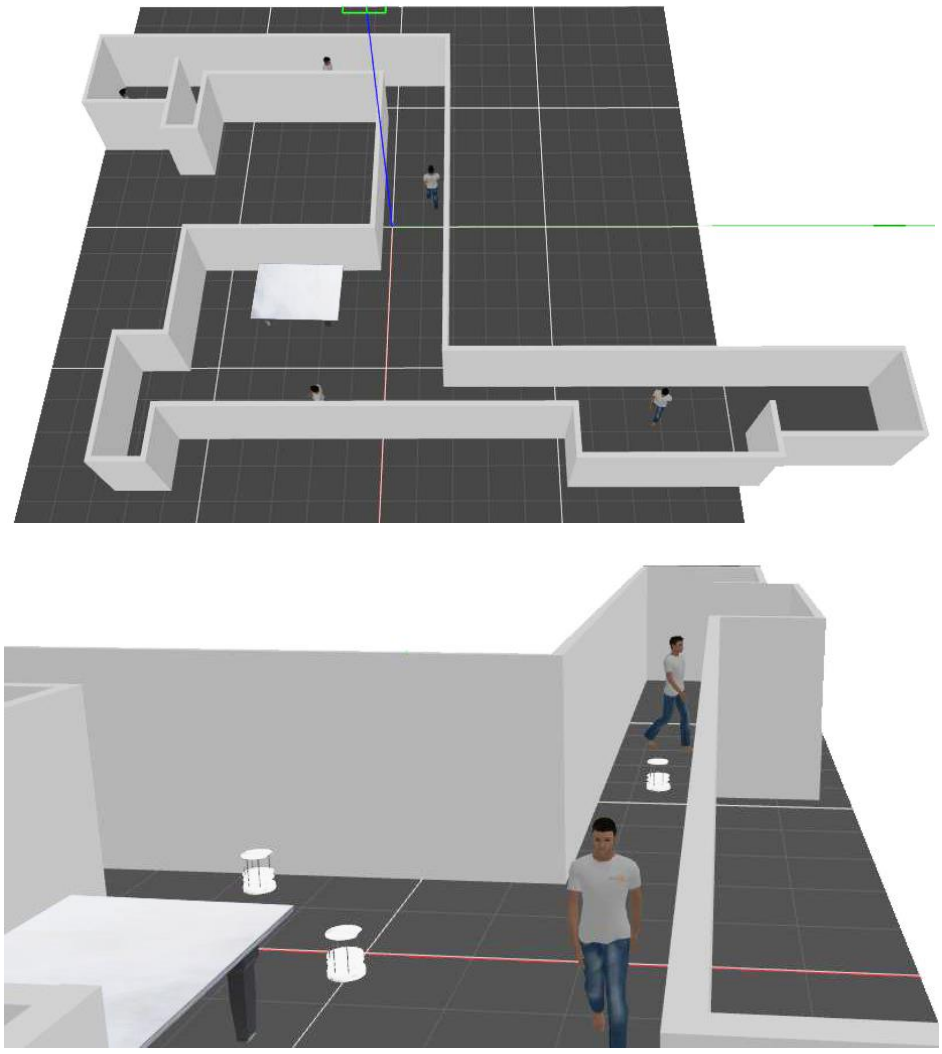
5.6. Simülasyon Çalışmaları

Tez çalışmasında gerçekleştirilen deneyler, simülasyon ortamında gerçek saha ölçeklemesi ile pratik deney kısıtlar gözetilerek tekrarlanmış ve sonuçlar gerçek saha verileri ile karşılaştırılmıştır. Simülasyon çalışmalarında ROS robotlarında yaygın olarak kullanılan Gazebo simülasyon programından yararlanılmıştır [60]. Gazebo simülasyon programı birçok robot platformuna ait kütüphaneyi desteklemekte ve ayrıca özgün geliştirilen robotlara ait URFD dosyaları ile simülasyonlar yapılmasını sağlamaktadır. Bu simülasyon programının bir diğer önemli özelliği ise gerçek kroki bilgilerini işleyerek eş ölçülerde saha görüntüleri elde edilebilmektedir.

Test çalışmasında gerçekleştirilen simülasyon çalışmaları deneysel çalışmalar ile benzer şekilde sunucu-istemci ilişkisine dayanan bir mimari ile gerçekleştirilmiştir. Sunucu olarak belirlenen bir çalışma istasyonu üzerinde Matlab Robotic Toolbox kullanılarak AMCL parametrelerini içeren bir yazılım oluşturulmuştur. Oluşturulan yazılım ile mobil robotun sanal ortamda 50 iterasyonda parçacık filtre algoritması ve

dinamik pencere yaklaşımı ile engellerden kaçınım sağlayarak ilerlemesi sağlanmıştır. Çalışma istasyonundan robota global yol için AMCL parçacıklarının dağılım kümesi gönderilirken, eş zamanlı olarak mobil robottan çalışma istasyonuna da simülasyon içerisindeki anlık lokal yol bilgileri aktarılmaktadır. Bu işlemler için deneysel çalışmalardakine benzer şekilde SSH protokolünden yararlanılmıştır.

Sakarya Uygulamalı Bilimler Üniversitesi Teknoloji Fakültesi Mekatronik Mühendisliği Bölümü'ne ait kroki Gazebo'da gerçek boyuttaki ölçülerle sanallaştırılmış ve Şekil 5.17.'te sunulmuştur.



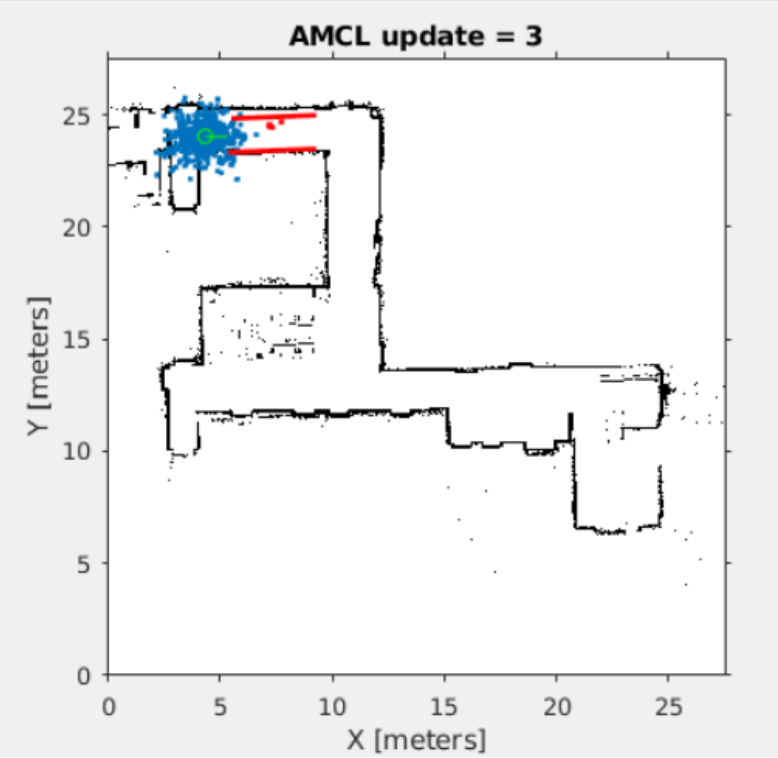
Şekil 5.17. Sanallaştırılmış simülasyon görüntüsü

Tez çalışması sırasında mobil robotun simülasyon ortamında belirlenen bir başlangıç noktasından sabit bir bitiş noktasına navigasyonunu gerçekleştirilmiştir. Birim karelerden oluşan simülasyon ekseninde $x = 4,\ y = 24$ konumuna $\theta = 0^0$ açısı ile yerleştirilen bir robotun 50 iterasyon sonucunda 10 tekrarlı deney sonuçları Tablo 5.5.'de verilmiştir.

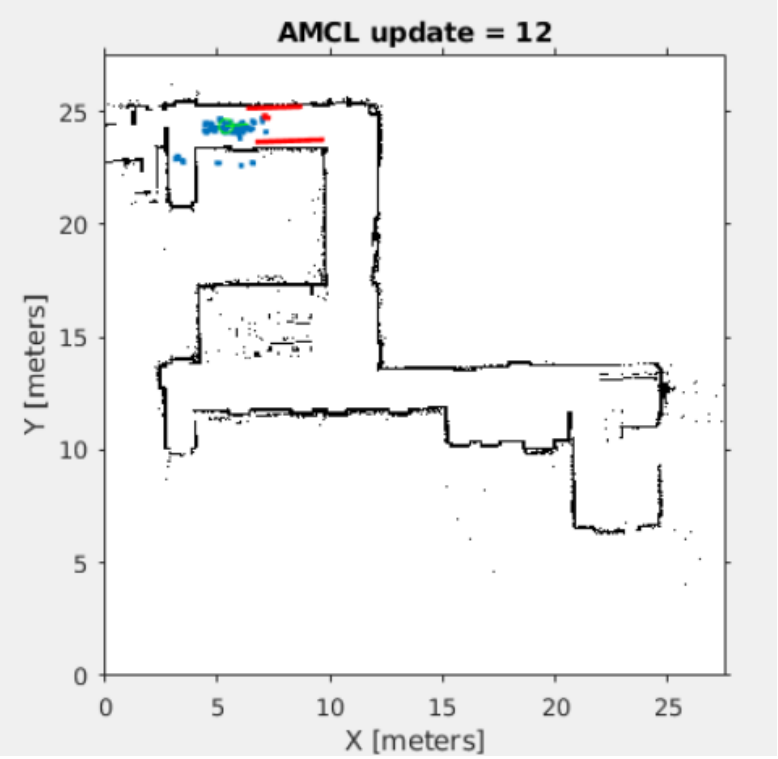
Tablo 5.5. Konumlanma hassasiyetlerine göre pozisyonlanma süreleri

Deney No	x	y	θ
1	10,6137	20,8166	-1,5819
2	10,8135	24,1083	-1,1620
3	10,3969	24,2824	-0,3542
4	10,7448	23,5990	-1,2629
5	10,9065	23,2308	-1,3313
6	10,8903	22,7569	-1,5906
7	10,9664	22,3876	-1,6513
8	10,6408	20,6941	-1,5875
9	10,8811	22,9788	-1,3519
10	10,8725	23,9258	-1,3352

Simülasyon ortamında gerçekleştirilen deneyler, gerçek ortamdaki odometri gürültülerinden bağımsız olduğu için toleransları düşük sonuçların elde edildiği görülmüştür. Robotun 50 iterasyon sonucunda ortalama $1{,}32^0$ lik bir sapma ile x ve y eksenine benzer yakınlıktaki noktalara konumlandığı tespit edilmiştir. Ayrıca ara iterasyonlarda oluşan parçacık dağılımları Şekil 5.18.'de verilmiştir. Şekil 5.18.a'da robot ilk hareketinden hemen 3. adımda sonra parçacıkların robot etrafında yoğun bir dağılım gösterdiği görülmüştür. Şekil 5.18.b'de ise parçacıkların yeniden örneklem dağılımlarının 12. iterasyonda koridor boyunca yayılım gösterdiği görülmektedir. Şekil 5.18.c'de ise son iterasyon olan 50. adıma yaklaştıkça parçacıkların robot etrafında kümelendiği gözlemlenmiştir. Şekil 5.18.d'de ise hedef noktaya varıldığında ağırlıkları zayıf olan parçacıkların elendiği, yüksek olanların ise robota en yakın mesafede pozisyonlandığı görülmüştür.

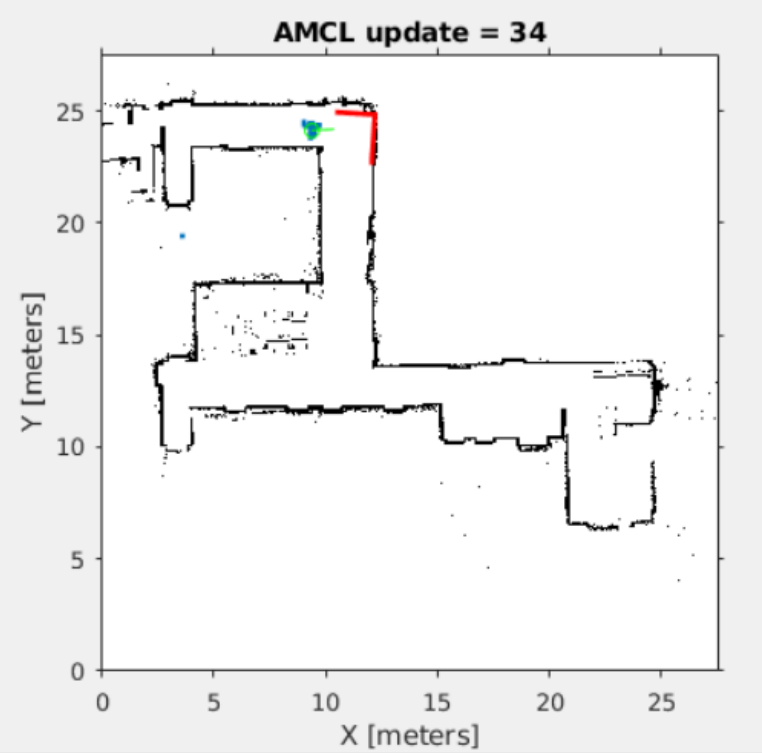


(a)

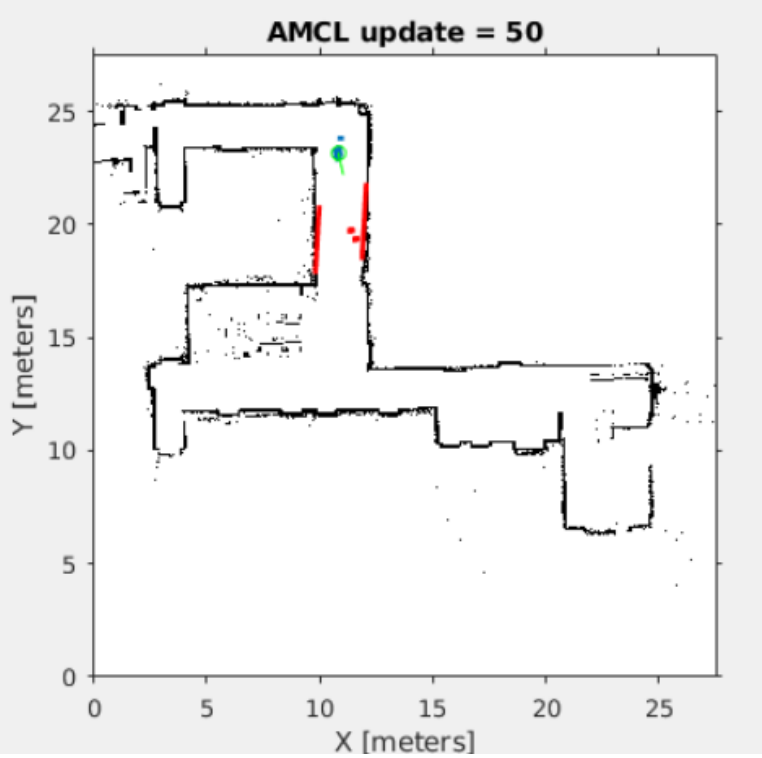


(b)

Şekil 5.18. İterasyon ara değerlerine ait parçacık dağılımları



(c)



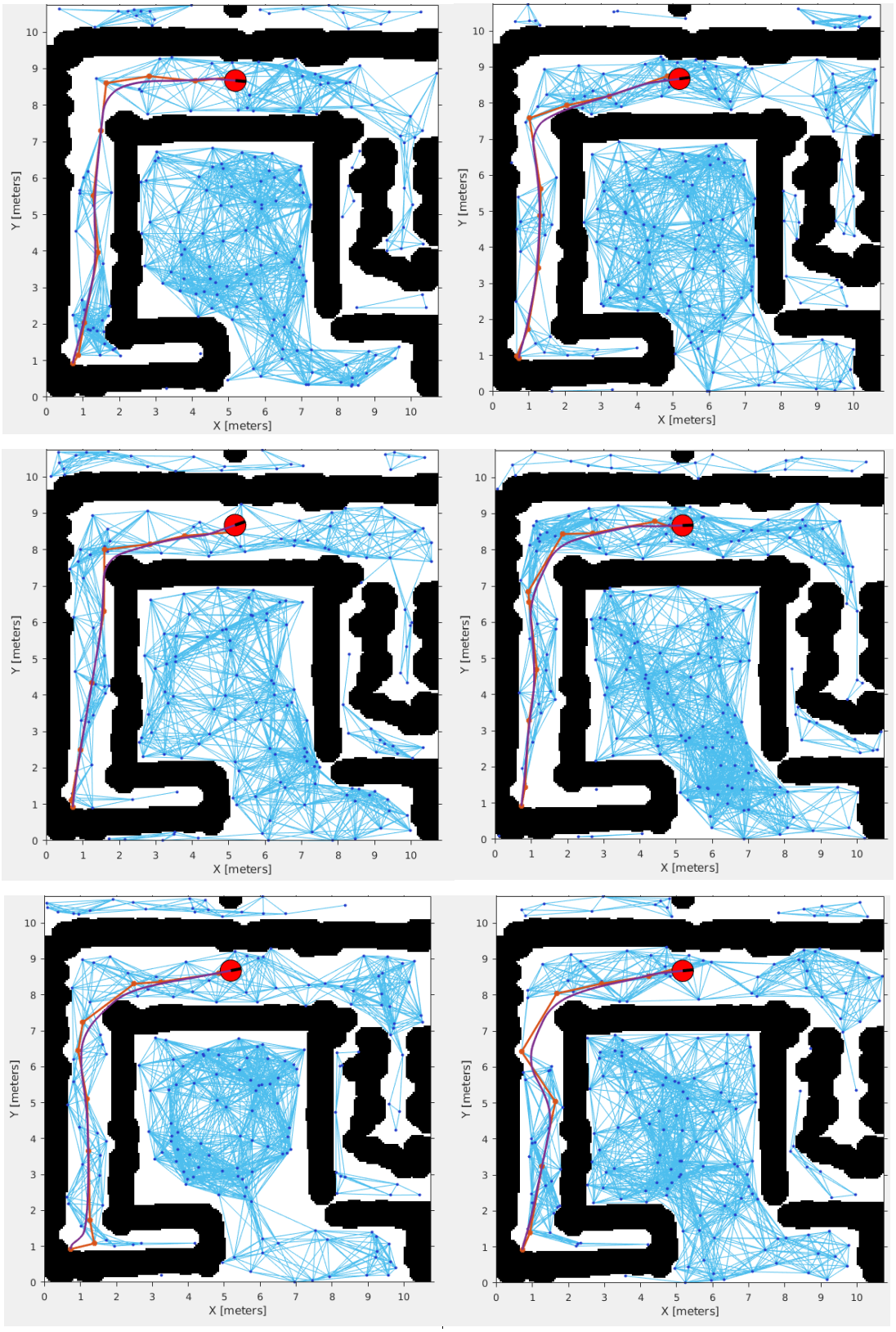
(d)

Şekil 5.18. (Devamı)

5.7. Olasılıklı Yol Haritası Metodu ile Karşılaştırma

Olasılıksal Yol Haritası (PRM), tez çalışmasında gerçekleştirilen konumlama ve pozisyonlama çalışmalarına benzer, olasılıksal mobil robot yönlendirme algoritmasıdır [61]. PRM algoritmasında belirli bir haritadaki boş ve dolu alanlar referans alınarak olası yolların bir ağ grafiği oluşturulur. Çalışma sahasının boyutlarına bağlı olarak önceden belirlenen sayıda rastgele düğümler ile oluşturulan olası yol bilgileri oluşturulurken düğümler birbirlerine engel durumları ve bağlantı parametreleri referans alınarak bağlanır. Bağlantılı düğümler üzerinden hedefe giden en kısa yolun seçimi algoritma tarafından yapılır ve mobil robotun global yol planlaması bu şekilde tamamlanmış olur.

Tez çalışmasında gerçekleştirilen navigasyon ile PRM algoritmasının karşılaştırılması testlerinde diğer deneylerde kullanılan Şekil 4.3.'te verilen haritadan yararlanılmıştır. Harita üzerinde $x = 0{,}7$ ve $y = 0{,}9$ konumunda $\theta = 0^0$ açısı konumlanmış bir mobil robotun $x = 5$ ve $y = 8{,}6$ konumuna PRM algoritması ile yönlendirilmesi sağlanmıştır. Bu konumlama işlemi, engelsiz ortamda gerçekleştirilen testlerde olduğu gibi 9 metre mesafe kat edilerek gerçekleştirilmiştir. Alana rastgele dağıtılan 200 düğüm ile mobil robotun başlangıç noktasından hedef noktaya kadar olan yolu 10 tekrarlı test sonuçlarında 26 saniyede aldığı gözlemlenmiştir. Gerçekleştirilen bu test sonuçlarına ait rota görüntülerinden bazıları Şekil 5.19.'da verilmiştir. Test sonuçları incelendiğinde engelsiz ortamda PRM algoritmasına göre tez çalışmasında tasarlanan sistem 9 metrelik mesafeyi 6 saniye daha erken tamamlamıştır. Dolayısıyla tez çalışmasında gerçekleştirilen yeni sistem olasılıksal bir algoritma olan PRM algoritmasına göre %30 oranında başarılı olmuştur. Ayrıca robotun rotada üzerindeki ilerlemesi PRM algoritmasına göre daha düzenli adımlarla gerçekleşmektedir. Global yol planlamasını için rassal olarak oluşturan düğümlerin dağılımları, çalışma sahasının yapısına göre değişiklik arz etmektedir. Bu durum düğümlerin optimum dağılımlarında parça filtresine göre olumsuzluk teşkil etmektedir.



Şekil 5.19. PRM testlerine ait 6 ayrı deney sonucu

5.8. Genel Değerlendirme

Bu tez çalışmasında otonom mobil robotların navigasyonu sırasındaki lokalizasyon ve pozisyonlama problemlerinin optimizasyonu üzerine bir çalışma yürütülmüştür. İlk olarak test çalışmalarının gerçekleştirilebilmesi açısından çalışma sahasının 2 boyutlu ızgara tabanlı haritası oluşturulmuştur. Bunu için SLAM metodu kullanılmış ve oluşturulan sayısal haritanın gerçek ortam ölçülerine olan yakınsamaları normalizasyon işlemine tabi tutularak karşılaştırmalı olarak verilmiştir. Çalışma alanına ait harita verilerinin elde edilmesinden sonra üzerinde donanımsal ve yazılımsal açıdan iyileştirmelerin gerçekleştirildiği Kobuki robot platformu ile otonom navigasyon işlemleri yürütülmüştür.

Dinamik ve statik engellerin bulunduğu çalışma sahasında belirlenen bir başlangıç konumundan operatör tarafından verilen hedef noktaya olan global yol bilgisi kablosuz SSH protokolü ile çalışma istasyonu üzerinden aktarılmıştır. Benzer şekilde mobil robot tarafından gerçekleştirilen gerçek zamanlı yol planlaması ise aynı protokol ile çalışma istasyonuna iletilmiştir. Global ve lokal olan bu yol planlama işlemleri için ise olasılıksal yol planlama algoritmalarından Adaptif Monte Carlo Lokalizasyon algoritması kullanılmıştır. AMCL algoritmasının navigasyonu etkileyen parametreleri üzerinde çeşitli değişiklikler yapılarak yapılandırılmış olan mobil robotun SLAM ile elde edilen harita üzerindeki otonom manipülasyonları gözlemlenmiştir.

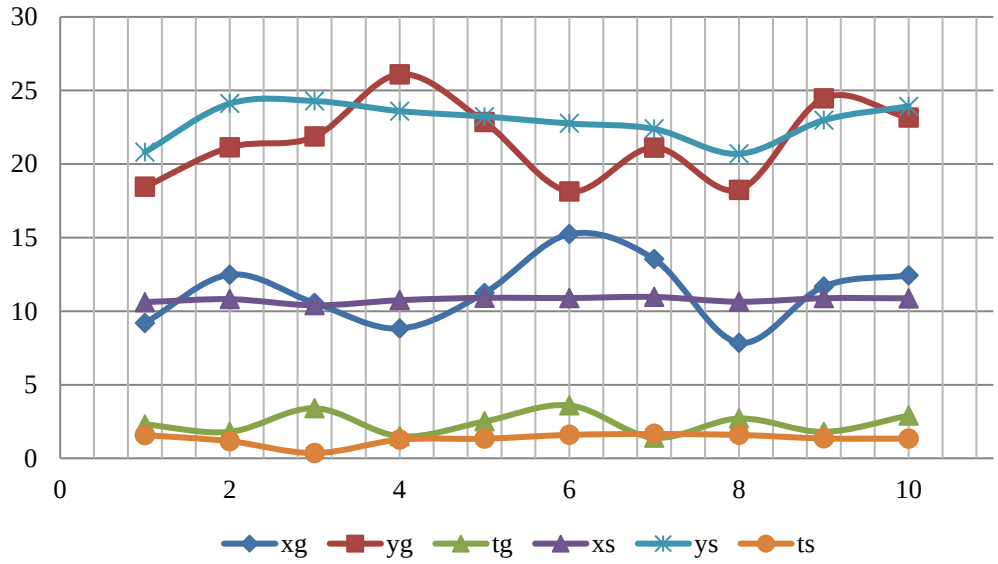
Tez çalışmasında parçacık filtre algoritmalarından olan AMCL'nin Kullback-Leibler mesafesi ile beraber kullanılması ile birlikte robot navigasyonunu sağlayan parçacıkların dinamik olarak gerçek zamanlı ayarlanması sağlanmıştır. KLD örnekleme adı verilen bu iyileştirme metodu ile parçacıkların yaklaşım hataları ölçülmüş olup mobil robotun statik ve dinamik ortamlardaki navigasyonu sağlanmıştır. Diferansiyel hareket modelindeki mobil robotun otonom hareketi esnasında hareketli engellerin algılanması amacıyla derinlik sensörü olarak RGB-D tipinde Kinect sensör kullanılırken, ortama ait haritanın SLAM ile elde edilmesinde

ise 8 metre tarama mesafesi bulunan 2 boyutlu bir lazer alan tarayıcı sensör kullanılmıştır.

Tez çalışması sırasında kullanılan algoritmalar, robotik çalışmalar için özel olarak geliştirilmiş olan ROS işletim sistemi ile kullanılarak, yüksek işlem kapasiteli yeni nesil bir geliştirme kartı olan JetsonTX2 üzerinden çalıştırılmıştır. Yapılan deneysel çalışmalar ayrıca Gazebo simülasyon ortamında Matlab Robotics System Toolbox kullanılarak doğrulanmıştır. Gerçek ortam testleri ve simülasyon çalışmalarına ait sonuçlar Tablo 5.6. ve Şekil 5.20.'te karşılaştırmalı olarak verilmiştir.

Tablo 5.6. Gerçek ortam verileri ve simülasyon verilerinin karşılaştırması

	Gerçek			Simülasyon			Fark (g-s)		
Deney No	xg	yg	tg	xs	ys	ts	xf	yf	tf
1	9,2	18,45	2,3	10,61	20,82	1,58	1,41	2,37	0,72
2	12,48	21,14	1,8	10,81	24,11	1,16	1,67	2,97	0,64
3	10,55	21,87	3,4	10,40	24,28	0,35	0,15	2,41	3,05
4	8,82	26,1	1,5	10,74	23,60	1,26	1,92	2,50	0,24
5	11,23	22,86	2,5	10,91	23,23	1,33	0,32	0,37	1,17
6	15,23	18,14	3,6	10,89	22,76	1,59	4,34	4,62	2,01
7	13,54	21,12	1,4	10,97	22,39	1,65	2,57	1,27	0,25
8	7,85	18,25	2,7	10,64	20,69	1,59	2,79	2,44	1,11
9	11,68	24,47	1,8	10,88	22,98	1,35	0,80	1,49	0,45
10	12,43	23,15	2,9	10,87	23,93	1,34	1,56	0,78	1,56



Şekil 5.20. Gerçek ortam verileri ve simülasyon verilerinin karşılaştırması

Tablo 5.6.'da belirtilen ölçümler, birim karelerden oluşan Gazebo simülasyonu ve gerçek ortamın temsil edildiği Rviz verilerinden elde edilmiş ve başlangıç konumu olarak $x = 4$ ve $y = 24$ konumunda $\theta = 0^0$ açısı ile yerleştirilen mobil robotun 50 iterasyon sonucunda 10 tekrarlı deney sonuçları verilmiştir. Elde edilen konumlanma için ulaşılan noktalar gerçek ortam için (xg, yg, tg) olarak verilirken simülasyon ortamı için bu değerler (xs, ys, ts) olarak verilmiştir.

Yapılan bu testler sonucunda en düşük $-x$ ekseni konumlanma sapma farkının simülasyon ve gerçek ortam verileri arasında xf=0,8 olduğu, en yüksek $-x$ ekseni konumlanma sapma değerini ise xf=4,34 olduğu görülmüştür. Benzer şekilde bu sapma değerleri $-y$ ekseni için yf=0,37 ile yf=4,62 arasında değişmektedir. Toplamda 9 metre mesafe kat edilerek ölçülen bu değerler sonucunda oluşan yaklaşık 4,25 birim karelik hatanın, odometri verilerinin oluşturduğu gürültüden kaynaklandığı tespit edilmiştir. Simülasyon ortamının sürtünme ve benzeri gürültülerden etkilenmemesi sonucunda oluşan ortalama hata sanal ortamda en fazla 22,88 değerinde iken bu değer gerçek ortamda 21,55 dir. Bu iki değer karşılaştırıldığında kurulan modelin simülasyon sonuçlarıyla $-x$ ekseninde %94,18 doğrulukta, $-y$ ekseninde ise %96,30 doğrulukta eşleşmiştir.

Pozisyonlama için $\theta = 0\,^\circ$ ile başlayan ilk iterasyonun 50 iterasyon sonucunda gerçek ortam için tg=1,4 ile tg=3,6 arasında değiştiği, simülasyon ortamı için ise bu değerin ts=0,35 ile ts=1,65 arasında değiştiği görülmüştür. Gerçek ortam ve simülasyon ortamında pozisyonlama açısından oluşan bu sapma değerleri de konumlandırmada oluşan sebeplerle benzer şekildedir.

Konumlanma hassasiyetlerine göre pozisyonlanma süresi incelendiğinde ise kullanılan mobil robot platformunun teknik karakteristiği ve ortam gürültülerine bağlı olarak en ideal şekilde hedefe 7 cm toleransla 5 sn sürede pozisyonlandığı görülmüştür. Pozisyonlanma toleransının daha az olması muhtemeldir fakat, pozisyonlama süresi pozisyonlama toleransı ile ters orantılıdır. Dolayısıyla mobil robotun kullanım alanı ve görevi ile alakalı olarak bu durum incelenerek uygun toleransa karar vermek mümkündür. Konumlandırılan hedef nokta için mobil

robotun pozisyonlanma toleransının minimum olması durumunda belirlenen pozisyona oturma süresi maksimum olurken, tersi durumda ise pozisyonlanma süresinin minimum olduğu deneyler sonucunda ortaya çıkmıştır.

İleriki çalışmalarda; tez çalışmasında kullanılan yöntem ve algoritmalar endüstriyel alanlarda robot filolarının navigasyon işlemlerinde kullanılabilir. Odometri verilerinin filtre devreleri ile sadeleştirilmesi sonucunda otonom navigasyonu etkileyen olumsuz koşulların giderilmesi sağlanabilir. Tez çalışmasında kullanılan algoritmaların sürü ve işbirlikçi robotlara uyarlanması sağlanabilir. Ayrıca Endüstri 4.0 kapsamında tasarlanacak özgün mobil robot filoları ile ülkemiz endüstrisine katkı sağlanması beklenmektedir

.

BÖLÜM 6. SONUÇLAR VE GELECEK ÇALIŞMALAR

Bu tez çalışmasında, dağıtık hareket kabiliyetine sahip mobil robotların yol planlaması üzerine olasılıksal parçacık filtre tabanlı algoritmalara dayanan bir çalışma yürütülmüştür. Eş Zamanlı Konum Belirleme ve Haritalama metotları ile çalışma sahasının 2 boyutlu haritası lazer tarayıcı sensörler kullanarak çıkarılmış ve mobil robotların saha içerisindeki otonom konumlama ve pozisyonlanma işlemleri gerçekleştirilmiştir. Operatör tarafında belirlenen başlangıç noktasından hedef bir noktaya statik ve dinamik engellerden kaçınım sağlayarak otonom bir şekilde en kısa yoldan en az maliyetle mobil robotun ulaşabilmesi sağlanmıştır. Bu kapsamda gerçekleştirilen global ve lokal yol planlama işlemlerinde AMCL algoritması üzerinde çalışılmıştır. Uygun konfigürasyonlar ile mobil robotun konumlamasını sağlayan parçacıkların ideal dağılımları üzerine çalışılmıştır. Diferansiyel hareket modeline sahip gerçekleştirilen prototip mobil robotun hem sanal ortam hem de gerçek saha çalışmaları yürütülmüş ve başarı oranları karşılaştırmalı olarak sunulmuştur. Ayrıca olasılıksal bir algoritma olan PRM algoritmasına göre performans karşılaştırılması gerçekleştirilmiştir. Aynı çalışma sahası içerisinde dağıtık olarak bulunan ve farklı görevleri yerine getirebilen mobil robotların yönlendirilmesi amacıyla kablosuz bir ağ oluşturulmuş ve mobil robotların bu ağ üzerinden sunucu istemci mimarisi ile kontrolü sağlanmıştır.

Sonuç olarak tez çalışmasında, modern algoritma ve donanımları kullanarak haritalama, pozisyonlama ve konumlama gibi birçok fonksiyonu yerine getirebilen bir sistem tasarımı ve prototipi gerçekleştirilmiştir. Bu yönüyle çalışma, Endüstri 4.0 uygulamalarının arttığı günümüzde birçok sektörde ihtiyaç hissedilen farklı yapı ve özellikteki mobil robotların ARGE çalışmaları için faydalı olabilecek niteliktedir.

KAYNAKLAR

[1] Durrant-Whyte, H., Bailey, T., Simultaneous localization and mapping:part I. IEEE Robotics & Automation Magazine, 13(2), 99-110, 2006.

[2] Bailey, T., Durrant-Whyte, H., Simultaneous localization and mapping (SLAM) Part II. IEEE Robotics & Automation Magazine, 13(3), 108-117, 2006.

[3] Atalı, G., Garip, Z., Karayel D., Özkan, S. S., Localization of mobile robot using odometry, camera images and extended Kalman Filter, Acta Physica Polonica A., 134(1), 204-207, 2018.

[4] Dellaert, F., Fox, D., Burgard, W., Thrun, S., Monte Carlo Localization for Mobile Robots, IEEE International Conference on Robotics and Automation, Detroit, 1322-1328, 1999.

[5] Fox, D., Burgard, W., Dellaert, F., Thrun, S., Monte Carlo Localization: efficient position estimation for mobile robots, Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI99), Florida, 343-350, 1999.

[6] Thrun, S., Fox, D., Burgard W., Dellaert, F., Robust Monte Carlo localization for mobile robots, Artificial Intelligence, 128, 99-141, 2001.

[7] Zhang, H., Chen, J. C., Zhang, K., RFID-based localization system for mobile robot with Markov Chain Monte Carlo, Proceedings of 2014 Zone 1 Conference of the American Society for Engineering Education (ASEE Zone 1), Connecticut, 86-90, 2014.

[8] Luo, F., Du, B., Fan, Z., Mobile robot localization based on particle filter, Proceeding of the 11th World Congress on Intelligent Control and Automation, Shenyang, 26-34, 2014.

[9] Aini, F. R., Jati, A. N., Sunarya, U., A study of Monte Carlo Localization on robot operating system, 2016 International Conference on Information Technology Systems and Innovation (ICITSI), Bali, 276, 2016.

[10] Chien, C. H., Hsu, C. C., Wang, W. Y., Kao, W. C., Chien, C. J., Global localization of Monte Carlo Localization based on multi-objective particle Swarm Optimization, International Conference on Consumer Electronics, Nevada, 260-279, 2016.

[11] Chien, C. H., Hsu, C. C., Wang W. Y., Kao, W. C., Mechanism for Preventing Premature Convergence in Robot Localization, IEEE 5th International Conference on Consumer Electronics, Berlin, 445-446, 2015.

[12] Medina Lee, J. F., Buitrago, J. A., Map generation and localization for a LEGO NXT robot, IEEE 2nd Colombian Conference on Automatic Control (CCAC), Manizales, 1-5, 2015.

[13] Garip, Z. B., Karayel, D., Ozkan, S. S., Atalı, G., Path planning for multiple mobile robots using A* algorithm, Acta Physica Polonica A, 132(3), 685-688, 2017.

[14] Yılmaz, S., Kayır, H. E., Kaleci, B., Parlaktuna, O., Parçacık süzgeci tabanlı konumlandırma yöntemi için yeni bir algılayıcı modeli, Otomatik Kontrol Ulusal Toplantısı, İstanbul, 428-434, 2010.

[15] Duymaz, E., Oğuz, A. E., Temeltaş, H., 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, 32(4), 1255-1270, 2017.

[16] Perez, D. H., Barbera, H. M., LeBlanc K., Saffiotti, A., Fuzzy uncertainty modeling for grid based localization of mobile robots, International Journal of Approximate Reasoning, 51, 912–932, 2010.

[17] Raza, A., Fernandez, B. R., A multi-tier immuno-inspired framework for heterogeneous mobile robotic systems, Applied Soft Computing, 71, 333-352, 2018.

[18] Mirkhani, M., Forsati, R., Shahri, A. M., Moayedikia, A., A novel efficient algorithm for mobile robot localization, Robotics and Autonomous Systems, 61, 920-931, 2013.

[19] Miao H., Tian, Y. C., Dynamic robot path planning using an enhanced simulated annealing approach, Applied Mathematics and Computation, 222, 420-437, 2013.

[20] Levinson, J., Thrun, S., Robust Vehicle Localization in Urban Environments Using Probabilistic Maps, IEEE International Conference on Robotics and Automation, Alaska, 4372-4378, 2010.

[21] Rohde, J., Jatzkowski, I., Mielenz, H., Zöller, J. M., Vehicle Pose Estimation in Cluttered Urban Environments Using Multilayer Adaptive Monte Carlo Localization, 19th International Conference on Information Fusion (FUSION), Heidelberg, 1774-1779, 2016.

[22] Vu, T. D., Aycard, O., Appenrodt, N., Online Localization and Mapping with Moving Object Tracking in Dynamic Outdoor Environments, Intelligent Vehicles Symposium IEEE, Istanbul, 190-195, 2007.

[23] Chong, Z., et al., Synthetic 2d lidar for precise vehicle localization in 3d urban environmet, Robotics and Automation (ICRA), IEEE International Conference, 2013, 1554-1559.

[24] Levinson, J., Montemerlo, M., Thrun, S., Map-Based Precision Vehicle Localization in Urban Environments, Robotics: Science and Systems, 2007.

[25] Wu, Y., Image Based Camera Localization: an Overview, In: arXiv preprint arXiv:1610.03660, 2016.

[26] Cosgun, A., Towards Full Automated Drive in Urban Environments: Demonstration in GoMentum Station, California, arXiv:1705.01187v1 [cs.RO], 2017.

[27] http://wiki.ros.org, Erişim Tarihi: 27.08.2018.

[28] http://wiki.ros.org/lunar, Erişim Tarihi: 27.08.2018.

[29] http://wiki.ros.org/kinetic, Erişim Tarihi: 27.08.2018.

[30] http://wiki.ros.org, Erişim Tarihi: 27.08.2018.

[31] Joseph, L., Mastering ROS for Robotics Programming, Packt Publishing, 2015.

[32] Chong, T. J., et al., Sensor technologies and simultaneous localization and mapping (slam), Procedia Computer Science, 76, 174-179, 2015.

[33] Jung, S., Kim, J., Kim, S., Simultaneous localization and mapping of a wheel-based autonomous vehicle with ultrasonic sensors, Artificial Life and Robotics, 14(2), 186-190, 2009.

[34] Lemaire, T., Berger, C., Jung, K., Lacroix, S., Vision-based slam: Stereo and monocular approaches, International journal of Computer Vision, 74(3), 343-364, 2007.

[35] Mustafah, Y. M., Azman, A. W., Akbar, F., Indoor uav positioning using stereo vision sensor, Procedia Engineering, 575-579, 2012.

[36] Gil, A., Reinoso, O., Ballesta, M., Julia, M., Multi-robot visual slam using a rao-blackwellized particle filter, Procedia Engineering, 58(1), 68-80, 2010.

[37] Henry, P., Krainin, M., Herbst, E., Ren, X., Fox, D., RGB-D Mapping: Using kinect-style depth cameras for dense 3d modeling of indoor enviroments, The International Journal of Robotics Research, 0(0), 1-17, 2012.

[38] Endres, F., Hess, J., Sturm, J., Cremers, D., Burgard, W., 3d mapping with an RGB-D camera, IEEE Transactions on Robotics, 30(1), 177-187, 2014.

[39] Koch, P., May, S., Michael Schmidpeter, S. A., Multi-robot localization and mapping based on signed distance functions, Journal of Intelligent & Robotic Systems, 88(3-4), 409-428, 2016.

[40] Amzajerdian, F., Pierrottet, D., Petway, L., Hines, G., Roback, V., Lidar systems for precision navigation and safe landing on planetary bodies, Proceedings of SPIE, 6, 8192, 2011.

[41] http://www.slamtec.com/en/lidar/a1, Erişim Tarihi: 27.08.2018.

[42] Follestad, M. H., Autonomous Path-Planning and Following for a Marine Surface Robot, Norwegian University of Science and Technology, 2017.

[43] Kohlbrecher, S., Von Stryk, O., Meyer, J., Klingauf, U., A fexible and scalable slam system with full 3d motion estimation, IEEE International Symposium on Safety, Security, and Rescue Robotics, 155-160, 2011.

[44] Weitbrecht, F., Monte Carlo Localization in Dynamic Enviroments Based on an Automotive Lidar Sensor Cocoon, Institute of Parallel and Distributed Systems Machine Learning and Robotics Lab., 2017.

[45] Thrun, S., Burgard, W., Fox, D., Probabilistic Robotics: MIT Press, 2005.

[46] Fox, D., KLD-sampling: Adaptive particle filters, 4th edition, MIT Press, 713-720, 2002.

[47] Tan, C. S., Sutton, R., Chudley, J., Collision avoidance systems for autonomous underwater vehicles part a: A review of obstacle detection, Journal of Marine Science and Environment, Part C, 39-50, 2004.

[48] Borenstein, J., Koren, Y., Collision avoidance systems for autonomous underwater vehicles part a: A review of obstacle detection, IEEE Transactions on Robotics and Automation, 15(3), 278-288, 1999.

[49] Fjellheim, R., Landre, E., Nilssen, R., Steine, T. O., Andreas, A., Autonomous systems: Opportunities and challenges for the oil and gas industry, Norwegian Society of Automatic Control, 7, 2018.

[50] Vagia, M., Transeth, A. A., Fjerdingen, S. A., Applied Ergonomics, 53rd Edition, Elsevier. 190-202, 2015.

[51] Campbell, S., Naeem, W., Irwin, G. W., A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres, Annual Reviews in Control, 36(2), 267-283, 2012.

[52] Hart, P., Nilsson, N., Raphael, B., A formal basis for the heuristic determination of minumum cost paths, IEEE Transactions on Systems Science and Cybernetics, 4(2), 100-107, 1968.

[53] Fox, D., Burgard, W., Thrun, S., The dynamic window approach to collision avoidance, IEEE Robotics and Automation Magazine, 4(1), 23-33, 1997.

[54] Claes, D., Collision Avoidance Under Bounded Localization Uncertainty, Maastricht University Faculty of Humanities and Sciences, 2012.

[55] http://wiki.ros.org/navigation/Tutorials/RobotSetup, Erişim Tarihi: 27.08.2018.

[56] Toquica Cáceres, H. M., Workshop 1: Kinematics Modeling, National University of Colombia, Colombia, 2017.

[57] https://en.wikipedia.org/wiki/Differential_GPS, Erişim Tarihi: 27.08.2018.

[58] https://en.wikipedia.org/wiki/Dead_reckoning, Erişim Tarihi: 27.08.2018.

[59] Y S Pyo, H C Cho, R W Jung, and T H Lim, *ROS Robot Programming*. Seoul, Korea: Robotis, 2017.

[60] http://gazebosim.org/tutorials?tut=ros_overview, Erişim Tarihi: 27.08.2018.

[61] Kavraki, L. E., Svestka, P., Latombe, J. C., Overmars, M. H., Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Transactions on Robotics and Automation, 12(4), 566-580, 1996.

EKLER

EK A: Doktora Tezi Kapsamında Yapılan Bilimsel Çalışmalar

Doktora tez kapsamında yapılan çalışmalar aşağıda verilmiştir.

[1] G Atalı, Z Garip, D Karayel, and S S Özkan, "Localization of Mobile Robot using Odometry, Camera Images and Extended Kalman Filter," *Acta Physica Polonica A*, vol. 134, no. 1, pp. 204-207, 2018. DOI: 10.12693/APhysPolA.134.204

[2] Z B Garip, D Karayel, S S Ozkan, and G Atalı, "Path Planning for Multiple Mobile Robots Using A* Algorithm," *Acta Physica Polonica A*, vol. 132, no. 3, pp. 685-688, 2017. DOI: 10.12693/APhysPolA.132.685

[3] Z. Batık, G. Atalı, D. Karayel, and S. S. Özkan, "Using heuristic distance functions in path planning of mobile robots," Electronics World, pp. 34–36, Oct. 2018.

[4] Z. Batık, D. Karayel, S. S. Özkan, and G. Atalı, "A Comparative Study on Heuristic Distance Functions in Path Planning of Mobile Robots," presented at the ICOME2017, İstanbul, 2017.

[5] Z. Batık, G. Atalı, S.S. Özkan and D. Karayel, "Path Plannıng of Mobile Robots Based on QR Code" International Symposium on Innovative Technologies in Engineering and Science ISITES2018.

EK B: Tez Çalışmasında Kullanılan Donanımlar ve Özellikleri

1- Kobuki Mobil Robot Platformu

Donanım

- PC bağlantısı: USB veya paralel port üzerinden RX/TX pinleri
- Motor aşırı yük Tespiti: > 3A
- Odometri: 52 ticks/enc rev, 2578.33 ticks/wheel rev, 11,7 ticks/mm
- İvme: Fabrika kalibrasyonu, 1 eksen (110 deg/s)
- Bumpers: sol, merkez, sağ
- Platform boşluk düşme sensörü: sol, merkez, sağ
- Tekerlek düşme sensörü: sol, sağ
- Güç konnektörleri: 5V/1A, 12V/1.5A, 12V/5A
- Çıkış pinleri: 3.3V/1A, 5V/1A, 4 x analog in, 4 x digital in, 4 x digital out
- Ses: Çeşitli programlanabilir bip dizileri
- Programlanabilir LED: 2 x üç renk led
- Durum LED: 1 x üç renk led [Yeşil - yüksek, Turuncu - düşük, Yeşil veyanıp sönen – şarj oluyor]
- Buton: 3 x dokunmatik buton
- Güç kaynağı: Lithium-Ion, 14,8V, 2200 mAh (4S1P - küçük), 4400 mAh (4S2P - büyük)
- Donanım güncelleyebilme: USB
- Sensör veri hızı: 50Hz
- Şarj adaptörü: Giriş: 100-240V AC, 50/60Hz, 1.5A max; Çıkış: 19V DC, 3.16A
- Netbook şarj konektörü (sadece robot yeniden şarj edildiğinde kullanılabilir): 19V/2.1A DC
- Sabitleme IR alıcısı: sol, merkez, sağ

Yazılım

- C++ , Linux
- ROS
- Gazebo

2- Kontrol Kartı – Nvidia Jetson TX2

- GPU: NVIDIA Pascal™, 256 CUDA cores
- CPU: HMP Dual Denver 2/2 MB L2
- Video: 4K x 2K 60 Hz Encode (HEVC) 4K x 2K 60 Hz Decode (12-Bit Support)
- Memory: 8 GB 128 bit LPDDR4 59,7 GB/s
- Display: 2x DSI, 2x DP 1.2 / HDMI 2.0 / eDP 1.4
- CSI: Up to 6 Cameras (2 Lane) CSI2 D-PHY 1.2 (2.5 Gbps/Lane)
- PCIE: Gen 2 | 1x4 + 1x1 OR 2x1 + 1x2
- Veri saklama: 32 GB eMMC, SDIO, SATA
- Haberleşme: CAN, UART, SPI, I2C, I2S, GPIOs
- USB: USB 3.0 + USB 2.0
- Ethernet: 1 Gigabit Ethernet, 802.11ac WLAN, Bluetooth
- Mekanik: 50 mm x 87 mm (400-Pin Compatible Board-to-Board Connector)

3- 360° Lazer Alan Tarayıcı – RPLIDAR/A1

- Algılama Mesafesi: 0,15-6 m
- Açı: 360^0
- Örnekleme süresi: 0,5 ms
- Örnekleme frekansı: >2000Hz
- Bağlantı: USB
- Döndürme motoru besleme: 5V-10V
- Tarayıcı besleme: 4,9V-5,5V

4- Hareket algılama sensörü - Kinect V1

- Görüş açısı: 43° dikey x 57° yatay
- Dikey eğim aralığı: ±27°
- Frame rate (derinlik ve renk akışı): 30 FPS
- Ses formatı: 16-kHz, 24-bit mono pulse code modulation (PCM)
- Ses girişi özellikleri: 4-microphone array, 24-bit ADC
- İvmeölçer özellikleri: 2G/4G/8G accelerometer

5- Diğer Donanımlar

- LTC1871 Step Up Ayarlanabilir Voltaj Regulator Kartı - 3,5-30 V 10 A
- 7 inch 800x480 çözünürlüklü dokunmatik ekran
- Power kablo ve soketleri
- Kinect adaptör
- 3D baskı raf
- 63mm ayırıcı vida x12
- 7mm ayırıcı vida x4
- USB 3.0 / 4 port HUB

ÖZGEÇMİŞ

Gökhan ATALI, 14.01.1985 de Ankara'da doğdu. İlk, orta ve lise eğitimini Ankara'da tamamladı. 2003 yılında Gazi Anadolu Meslek Lisesi, Bilgisayar Bölümü'nden mezun oldu. 2005 yılında başladığı Selçuk Üniversitesi Seydişehir Meslek Yüksekokulu, Bilgisayar Teknolojileri ve Programcılığı Programını 2007 yılında 2.lik ile bitirdi. Ardından aynı yıl Sakarya Üniversitesi Teknik Eğitim Fakültesi, Bilgisayar Sistemleri Öğretmenliği Bölümü'nü kazandı ve 2011 yılında bölüm 2.incisi olarak mezun oldu. 2012 yılında Sakarya Üniversitesi Mekatronik Mühendisliği Bölümü'nde yüksek lisansa başladı ve 2015 yılında "Döner Tablalı Mekatronik Sistemler için Tek Hatlı Çift Yönlü Haberleşme Protokolü Tasarımı" adlı tez çalışması ile yüksek lisans eğitimini tamamladı. 2014 yılında Sakarya Üniversitesi Sakarya Meslek Yüksekokulu Mekatronik Programı'nda Öğretim Görevlisi olarak başlayan akademik yaşantısı şuan Sakarya Uygulamalı Bilimler Üniversitesi Adapazarı Meslek Yüksekokulu Mekatronik Programı'nda Öğretim Görevlisi olarak devam etmektedir.