T.C.
DÜZCE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OTONOM ARAÇLARDA EŞ ZAMANLI LOKASYON VE HARİTALANDIRMA İLE GENETİK ALGORİTMA KULLANILARAK OPTİMUM YOL SEÇİMİ

MERVE NUR DEMİR

YÜKSEK LİSANS TEZİ
ELEKTRİK – ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN
DOÇ. DR. YUSUF ALTUN

DÜZCE, 2019

T.C.
DÜZCE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OTONOM ARAÇLARDA EŞ ZAMANLI LOKASYON VE HARİTALANDIRMA İLE GENETİK ALGORİTMA KULLANILARAK OPTİMUM YOL SEÇİMİ

Merve Nur DEMİR tarafından hazırlanan tez çalışması aşağıdaki jüri tarafından Düzce Üniversitesi Fen Bilimleri Enstitüsü Elektrik – Elektronik Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Doç. Dr. Yusuf ALTUN

Düzce Üniversitesi

Jüri Üyeleri

Doç. Dr. Yusuf ALTUN
Düzce Üniversitesi _____

Dr. Öğr. Üyesi Arafat ŞENTÜRK
Düzce Üniversitesi _____

Dr. Öğr. Üyesi Okan ERKAYMAZ
Zonguldak Bülent Ecevit Üniversitesi _____

Tez Savunma Tarihi: 14/11/2019

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını beyan ederim.

14 Kasım 2019

(İmza)

Merve Nur DEMİR

TEŞEKKÜR

Tez çalışmam sırasında kıymetli bilgi, birikim ve tecrübeleri ile bana yol gösterici ve destek olan değerli danışman hocam sayın Doç. Dr. Yusuf ALTUN'a, ilgisini ve önerilerini göstermekten kaçınmayan Prof. Dr. Resul KARA, Dr. Öğretim Üyesi Okan ERKAYMAZ, ve adını yazamadığım nice hocalarıma sonsuz teşekkür ve saygılarımı sunarım.

Çalışmalarım boyunca yardımını hiç esirgemeyen değerli çalışma arkadaşlarıma ve dostlarıma teşekkürü bir borç bilirim. Hayatım boyunca her koşulda bana destek veren, önceliklerini her zaman benim önceliklerime göre düzenleyen ve bunun karşılığını hiçbir zaman tam olarak ödeyemeceğim, bugünlere gelmemde en büyük katkıları olan, bu hayatta hiçbir şeye asla değişmeyeceğim babama, anneme, kız kardeşlerime ve çalışma arkadaşlarıma sonsuz teşekkürlerimi sunarım.

14 Kasım 2019 **Merve Nur DEMİR**

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

ÇİZELGE LİSTESİ

Sayfa No

KISALTMALAR

2D	2 Boyut
3D	3 Boyut
ACC	Adaptif Hız Sabitleyici
AI	Yapay Zekâ
CEKF	Sıkıştırılmış Genişletilmiş Kalman Filtresi
CML	Eş Zamanlı Haritalama ve Lokasyon
CPU	Merkezi İşlem Birimi
DART	Dinamik Canlandırma ve Robotik Araçlar
EIF	Genişletilmiş Bilgi Filtresi
EKF	Genişletilmiş Kalman Filtresi
EM	Beklenti Maksimizasyonu
GA	Genetik Algoritma
GPS	Küresel Konumlandırma Sistemi
GRV	Gauss Rastgele Değişkeni
GSYS	Gelişmiş Sürücü Yardım Sistemi
GUI	Grafiksel Kullanıcı Arabilimi
ICRA	Uluslararası Robotik ve Otomasyon Konferansı
IF	Bilgi Filtresi
ISRR	Uluslararası Robot Araştırmaları Sempozyumu
KF	Kalman Filtresi
MMSE	Minimum Ortalama Kare Hata
ODE	Açık Dinamikler Motoru
PF	Parçacık Filtresi
PSLAM	Olasılıksal SLAM
RAM	Rasgele Erişimli Bellek
ROS	Robot İşletim Sistemi
Rviz	Robot İşletim Sistemi Görselleştirici
SAE	Otomotiv Mühendisleri Derneği
SLAM	Eş Zamanlı Lokasyon ve Haritalama
SMC	Sıralı Monte Carlo
UKF	Kokusuz Kalman Filtresi

ÖZET

OTONOM ARAÇLARDA EŞ ZAMANLI LOKASYON VE HARİTALANDIRMA İLE GENETİK ALGORİTMA KULLANILARAK OPTİMUM YOL SEÇİMİ

Merve Nur DEMİR
Düzce Üniversitesi
Fen Bilimleri Enstitüsü, Elektrik-Elektronik Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi
Danışman: Doç. Dr. Yusuf ALTUN
Kasım 2019, 55 sayfa

Teknolojik gelişmeler ve bu zamana kadar biriken bilgilerin ışığında otonom sistemlerde muazzam bir ilerleme kaydedilmiştir. Bu sayede otonom sistemler çarpışmadan kaçınma, trafik işareti tespiti, haritalama vb. sayısız akıllı işlevleri gerçekleştirebilmektedir. Gerçek zamanlı otonom araçların en zorlu problemi aracın kendi kendine haritalandırma ve lokasyon işlemlerini yapabilmesidir. Genetik Algoritma (GA) kullanarak optimize edilmiş lokasyon uygulaması ile otonom araçlar için sürüş güvenliğinin artması beklenmektedir. Bu çalışma da lazer tabanlı bir lokalizasyon ve haritalama tekniğinin üzerine odaklanılmıştır. Gerçekleştirilen sistemde sanal bir test ortamı kurulmuş ve bir otonom araç üzerinde denemeler yapılmıştır. Çalışma kapsamında sanal makineler oluşturularak üzerlerine Linux işletim sistemi kurulmuştur. Sonra bu sanal makinelere ROS ortamında TurtleBot3 kurulmuş ve iç mekân lokalizasyonu yapılarak bir harita elde edilmiştir. Bu harita genetik algoritma ile en kısa mesafelerin bulunmasını sağlamak için kullanılmaktadır. Gözlemler neticesinde simülasyon ortamındaki robot yüksek başarımla istenilen konuma gidebildiği sonucuna ulaşılmıştır.

Anahtar sözcükler: Genetik Algoritma, Haritalama, Lokasyon, Otonom Araç, SLAM.

ABSTRACT

OPTIMAL ROAD SELECTION BY USING GENETIC ALGORITHM AND SIMULTANEOUS LOCATION AND MAPPING IN AUTONOMOUS VEHICLES

Merve Nur DEMİR
Düzce University
Graduate School of Natural and Applied Sciences, Department of Electrical -Electronics Engineering
Master's Thesis
Supervisor: Assoc. Prof. Dr. Yusuf ALTUN
November 2019, 55 pages

Significant progress has been made in autonomous systems in the light of technological advances and accumulated knowledge to date. In this way, autonomous systems, collision avoidance, traffic sign detection, mapping and so on. It can perform numerous intelligent functions. The most challenging problem of real-time autonomous vehicles is that the vehicle can perform self-mapping and location operations. Optimized location application using Genetic Algorithm (GA) is expected to increase driving safety for autonomous vehicles. This study focuses on a laser-based localization and mapping technique. In the system, a virtual test environment was established and experiments were performed on an autonomous vehicle. Within the scope of the study, virtual machines were created and Linux operating system was installed on them. Then, TurtleBot3 was installed in these virtual machines in ROS environment and a map was obtained by localizing the interior. This map is used to find the shortest distances by genetic algorithm. As a result of the observations, it was concluded that the robot in the simulation environment can go to the desired position with high performance.

Keywords: Genetic Algorithm, Mapping, Localization, Autonomous Vehicle, SLAM.

1. GİRİŞ

Teknolojik gelişmeler ve bu zamana kadar biriken bilgilerin ışığında otonom mekanizmalar dünya çapında yüzlerce alanda kullanılmaya başlamıştır. Robotik sistemlerin geliştirilmesinde çevresine daha hızlı ve daha kararlı yanıtlar verebilen mekanizmalar üretebilmek ve bunun için de insanların sahip olduğu algılayıcılara benzer algılayıcılar ile donatılmış robotların çalışma hızının; gerçek zamanlı ve insanın algılama ve yanıt verme hızına yakın olması bilim insanlarının en büyük hedeflerinden biridir. Mevcut çalışmalar ve gelecekteki eğilimler bize, son zamanlardaki destek sistemlerinin gelecekte tamamen otonom bir araca yönelik sürücü yardım sistemi olarak önemli rol oynayacağını göstermektedir [1].

Bilgisayar bilimleri ve otomotiv endüstrisi için sorunun temeli; sürücüyü, ona yardımcı olacak ve sürüş güvenliğini artıracak olan insansız/otonom araçlarla nasıl birleştirilebileceğidir [2]. Otonom mobil robotlar değişen çevresel koşullara etkili bir şekilde adapte olabilmelidirler. Bunun yanı sıra, mobil robotların çevrelerini herhangi bir nesne ve engelle çarpmadan keşfetmeleri gerekmektedir [3].

Son on yılda, Gelişmiş Sürücü Yardım Sistemi (GSYS) ve otonom sürüş konusunda muazzam bir ilerleme kaydedilmiştir. Bu tür sistemler çarpışmadan kaçınma, şerit ayrılma uyarısı, trafik işareti tespiti, haritalama vb. sayısız akıllı işlevleri gerçekleştirebilmektedir [4]. Bu sistemleri uygularken, makine öğrenmesi, algılayıcı verilerinin yorumlanması ve çevreyi anlama ve anlamlandırma önemli bir rol oynar. Otonom sürüşün başlıca zorluklarından biri, çoğu otonom sürüş sisteminin ağır görsel algılamaya dayanmasıdır. Örneğin yağmur, sis vb. çevresel değişikliklere algılayıcıların duyarlı olması sebebiyle otonom bir sürüş sistemi kusursuz olamamaktadır. Sağlam bir otonom sürüş sistemi oluşturmak için, olası tüm çevresel senaryolar üzerinden araç performansını doğrulamak çok önemlidir [5].

Karayolu taşıtları için geniş görüşlülük tabanlı bir araç rehberlik sistemi üzerine yapılan çalışmalar dört ana hedefe odaklanmaktadır:

- Yol tespiti,

- Engel tespiti,
- İşaret tanıma ve
- Haritalama ve Lokasyon.

İlk üç hedef için uzun yıllardır çalışılmıştır ve Haritalama ve Lokasyon hala açık bir çalışma alanıdır [6].

Çalışmalar son yıllarda daha da yoğun bir şekilde yürütülmekte ve birçok algoritma getirilmiştir. Çalışmaların çoğu, belirli bir hızda yol boyunca hareket eden arabadan elde edilen video akışlarıyla gerçekleştirilmiştir. Bu çalışmaları göz önünde bulundurduktan sonra, haritalama ve lokasyon tespiti sorunu şöyle belirtilebilir:

- Renk ve çevre bilgisi, değişken ışıklandırmadan etkilenir [5], [7], [8].
- Kötü hava koşulları,
- Kir ve akromatik gibi çevresel etkiler.
- Kullanılan algılayıcıların ne kadar ideal çalıştığı,
- Ayrıca, algoritmaların gerçek zamanlı uygulamaya uygun olma durumu [5].

Gerçek zamanlı otonom araçların iki önemli problemi; sağlam ve hassas bir haritalandırma ve lokasyonun araç tarafından tanımlanmasıdır. Konum tahminlerin sağlamak için çeşitli algılayıcılar ve algoritmalar kullanılabilir [9]. Harita oluşturma işleminin doğruluğu, başlangıçtaki pozisyona dayanmaktadır [10].

Dış mekânda otonom sürüş yapan robotlar, Küresel Konumlandırma Sistemi (GPS) kullanarak zamanlama sinyallerine göre konumlarını boylam ve enlem açısından hesaplayabilir. Fakat iç mekânda bulanan robotlar GPS sinyallerini alamaz ve mesafe algılayıcıların yardımına ihtiyaç duyarlar.

Günümüzde sürücüler, araçları üzerinde tam kontrol sahibidirler. Sürüş, tamamen görsel bilgiye dayanan bir iştir [8]. Sürücü etrafına bakarak kafasında o anki duruma göre nerede olduğunu, gitmek istediği yere nasıl gidebileceğini bilebilmektedir. Peki, otonom bir araç hiç bilmediği bir ortama bırakıldığında ortamın haritasını ve kendi konumunu çıkararak kendisinden istenilen yere gidebilir mi? Bilimsel literatürde “Eş Zamanlı Lokalizasyon ve Haritalama (Simultaneous Localization and Mapping)” yöntemi olarak bilinen SLAM bu probleme bir çözüm önerisi olabilmektedir. 1986 yılında, Durrant-Whyte, Crovley ve Cheeseman yapmış olduğu SLAM çalışmaları olasılıksal yöntemler içeren bir çalışmadır

[11].

Haritalama yöntemleri içinde bazı SLAM algoritmaları geliştirilmiştir. Örneğin, Jin Jung Myung ve arkadaşları [12] 'de parçacık ağırlığı bazlı doluluk ızgara haritasını kullanan bir hızlı-SLAM yöntemi sunmuştur. Performansı, Monte Carlo Lokasyon algoritması veri ilişkilendirmeleriyle birleştirilen parçacık başına maksimum olabilirlik olasılığı ile geliştirilmiştir. Yusuke Misono [13], dış mekân mobil robotları için Lazer Menzil Bulucu kullanan göreceli engel gözlem profillerini kullanarak SLAM uygulamıştır. Küresel bir harita, göreceli gözlemler kullanılarak adım adım inşa edilir ve aynı zamanda genişletilmiş Kalman filtresiyle robotun bulunduğu yerin sınırlı bir tahminini hesaplar. Momotaz Begum [14], SLAM için bulanık mantık ve genetik algoritmayı (GA) bütünleştiren bir yöntem geliştirmiştir.

GA, tabiatta gözlemlenen evrimsel sürece benzer şekilde çalışan optimizasyon ve arama yöntemidir. Karmaşık çok boyutlu arama uzayında en iyinin hayatta kalması ilkesine göre bütünsel en iyi çözümü arar ve sayısal optimizasyon yöntemlerinde sıkça kullanılmaktadır. Klasik yöntemlerle karşılaştırıldığında daha iyi sonuç verdiği gözlemlenmiştir. Sezgisel bir yöntem olması sebebiyle, bulunan çözüm en iyi çözüm olmayabilir. Ancak, geleneksel yöntemler ile çok uzun sürebilecek problem çözümlerinin, gözle görülür bir şekilde kısa bir sürede elde edilmesinde faydalı olmaktadır [15]. Son zamanlarda, karmaşık ve oldukça doğrusal olmayan çok amaçlı optimizasyon problemlerini başarılı bir şekilde çözmek için uygulama alanlarında evrimsel yöntemler uygulanmaktadır. GA, bugüne kadar robot lokalizasyonu ya da haritalama problemine uygulanan en yaygın evrimsel algoritmalar olmuştur [16]. Bir önceki çalışmalarını geliştirme amacıyla temel fikri, örnek güçsüz/yetersiz kromozomların sayısını azaltmak ve hesaplama maliyetini düşürmek için harita parçacığını / kromozomlarını sıkıştırmak olan Rao-Blackwellized GA Filtreli SLAM tekniği önerilmiştir [17].

Bu tezde lazer tabanlı bir lokalizasyon ve haritalama tekniğinin optimizasyonu üzerine odaklanılacaktır. Amaç otonom araçlar için haritalama algoritmalarının incelenmesi ve daha etkili bir algoritmanın geliştirilmesi; gerçekleştirilen sistemin benzetim ortamında sanal bir otonom araç üzerinde test edilmesidir. Yapılacak olan bu haritalama uygulaması ile otonom araçlar için sürüş güvenliğini ve verimliliğini artırması beklenmektedir. Çalışma kapsamında bilgisayarda işletim sistemi Linux olan sanal makineler kurulmuştur. Sonra ki adımda bu sanal makinelere ROS işletim sistemi ve TurtleBot3

kurulmuştur. Gazebo benzetim ara yüzü ile iç mekân lokalizasyonu yapılarak bir harita elde edilmiştir ve eş zamanlı olarak Rviz ortamında izlenmiştir. Elde edilen bu harita GA ile en kısa mesafelerin bulunmasını sağlamak için kullanılmaktadır. Gözlemler neticesinde benzetim ortamındaki robot yüksek başarımla istenilen konuma gidebildiği sonucuna ulaşılmıştır.

Bu tez kapsamında Bölüm 2'de Otonom Araçlar hakkında bilgi verilmiş ve kullanılan yazılım ve benzetim ortamları tanıtılmıştır. Bölüm 3'de SLAM ve GA kullanılan yöntemler olarak anlatılmıştır. Bölüm 4'de GA'nın sisteme uygulanması anlatılmıştır. Bölüm 5'de simülasyon sonuçları verilmiştir. Bölüm 6'da simülasyon sonuçları yorumlanmış ve gelecek çalışmalara öneriler verilmiştir. Bölüm 7'de kullanılan kaynaklar verilmiştir.

2. OTONOM ARAÇLAR

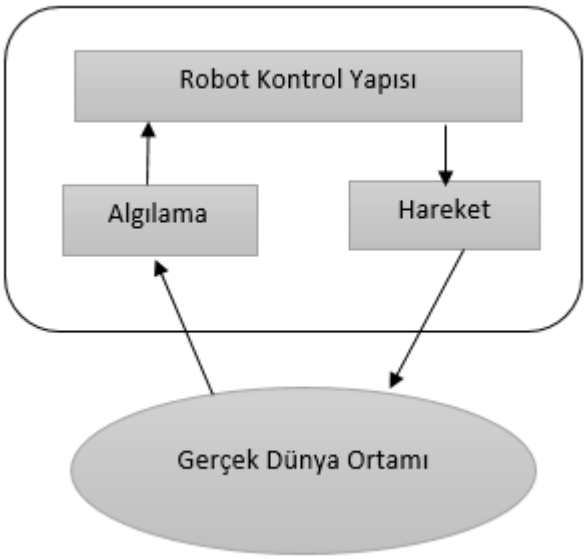
Otonom sürüş konusundaki ilk deneyler 1920'lerden başladı [18] ve ilk prototipler 1950'lerden itibaren ortaya çıkmaya başlamıştır. 1980'lerde araştırmacılar daha fazla prototip ortaya çıkardılar; 1984'te Carnegie Mellon Üniversitesi'nin Navlab [19] ve ALV [20] [21] projeleri, projeler ve 1987'de Mercedes-Benz ve Bundeswehr Üniversitesi Münih'in Eureka Prometheus Projesi [22] bunlardan bazılarıdır. 1997 yılında Japonya'da Tsukuba Makine Mühendisliği Laboratuvarı ilk gerçek otonom aracı geliştirdi. ABD Ulusal Otomatik Karayolu Sistemi programı, otomatik araçlarla otoyol ağının 1997'de başarıyla küçük bir ölçekte birleştirildiğini göstermiştir [23]. Navlab, 1995 yılında Amerika'da otonom bir şekilde 4501 kilometrenin %98'ini sürdü [24], ve bir Audi, Delphi teknolojisinin [25] yardımı ile 5472 kilometrenin %99'unu kullandığında 2015 yılına kadar rekor kırılmamıştı. ABD'nin birçok eyaleti bundan sonra kamuya açık yollarda testlere izin verdi [26]. Son rekor, Waymo'nun otonom araçlarının Temmuz 2018'de 13.000 kilometreden fazla yol kat ettiğini duyurmasıydı [27].

Otonom araçlar çevresinden topladıkları verileri içinde bulunan karar mekanizmaları ve algoritmalar sayesinde yorumlayıp kendi kendine hareket eden mekanizmalardır.

Otonomluğu gerçekleştirmek için bir yaklaşım, araçlar arasında iletişim ağları kurmaktır. Kooperatif durum, uzak sürücülü sistemin kısaltmasıdır. Otomasyon derecesini standartlaştırmak için, Uluslararası Otomotiv Mühendisleri Derneği (OMD- Society Of Automotive Engineers- SAE), 2014 yılında bir sınıflandırma sistemi yayınladı ve bu sistem 2016 yılında güncelledi. SAE'nin sınıflandırması, tamamen manuelden tamamen otonom araçlara kadar değişen altı otomasyon seviyesi tanımlar. Her seviyenin kısa açıklaması aşağıda verilmiştir [28]:

- Seviye 0: Sistem uyarıları idare eder ancak araç kontrolü devam etmemiştir.
- Seviye 1: Sürücü ve sistem birlikte çalışır, örneğin adaptif hız sabitleyici (ACC) ve park yardımı.
- Seviye 2: Sistem hızlanma, frenleme ve yönlendirmeyi kontrol eder, ancak sürücünün süreci izlemesi ve sistem arızalandığında kontrolü ele almaya hazır olması gerekir.

- Seviye 3: Sürücü, sürüş sırasında “gözleri kapalı” olarak adlandırılan eylem (bir metin okuma veya bir filmi izleme gibi) yapılabilir. 2018'de, Audi A8 Luxury Sedan, seviye 3 otomasyonlu ilk ticari otomobildi, ancak sadece yüksek yollarda tek yönlü trafik için çalışıyor.
- Seviye 4: Artık güvenlik sorunu yok. Sürücü, yolcu koltuğuna oturabilir veya uyuyabilir.
- Seviye 5: Artık insan katılımı yok. Olası bir örnek robotik taksi.



Şekil 2.1. Mobil robot sistemi blok şeması.

2.1. OTONOM ARAÇ SİMÜLASYONU VE YAZILIM SİSTEMLERİ

2.1.1. ROS Yazılım Sistemi

ROS (Robot Operating System) bir işletim sistemi değil, bilgisayar üzerinden robot bileşenlerimizi ve robotlarımızı kontrol etmemizi sağlayan açık kaynak kodlu ve BDS lisanslı bir yazılım sistemidir. Donanım soyutlama, düşük seviye cihaz kontrolü, işlemler arasında mesaj geçişi ve paket yönetimi dâhil bir işletim sisteminden beklenen hizmetleri sağlar. ROS gerçek zamanlı bir çerçeve değildir, ancak ROS’u gerçek zamanlı kodla bütünleştirmek mümkündür. ROS’un temel amacı, robotik araştırma ve geliştirmede kodun yeniden kullanılmasını desteklemektedir. ROS şu anda Unix, Fedora, Gentoo, Arch Linux ve diğer Linux platformlarında çalışabilmektedir [29].

ROS kullanılmasının sebepleri;

i. Robot geliştirme sürecini hızlandırmakta ve kolaylaştırmaktadır.

ii. Non-ROS bir uygulamayı ROS uygulamasına dönüştürmek için sadece ufak bir kod bloğu eklemek yeterlidir. İhtiyacınız olan birçok araç hazır gelmekte, bu sayede algoritma geliştirme dışında harcamanız gereken süre azalmaktadır.

iii. Geliştirilen uygulamanın başka cihazlarda da kullanılabilmesi mümkündür.

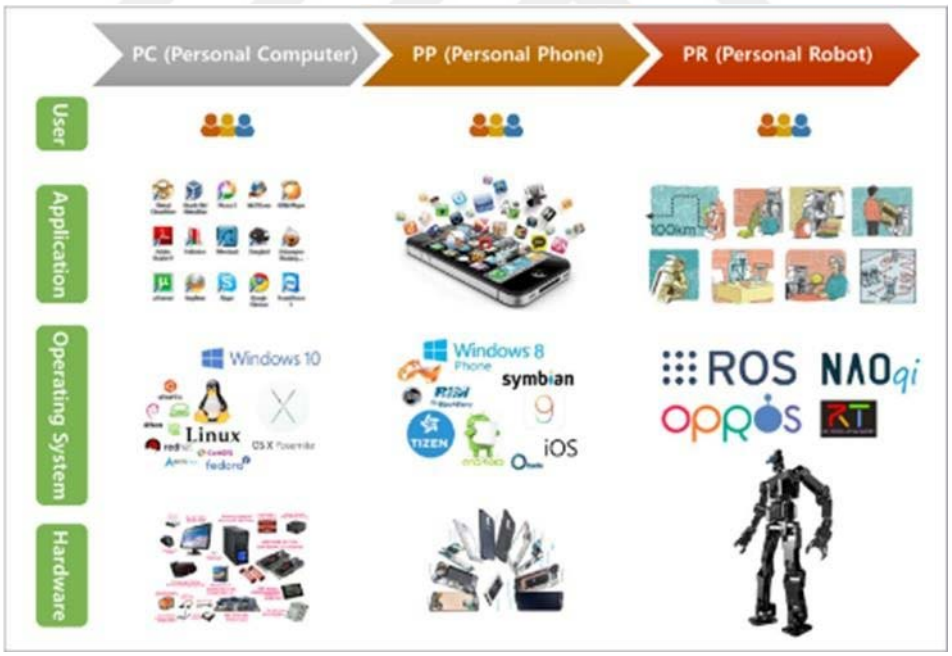
iv. ROS haberleşme tabanlı bir programdır.

v. ROS içerisinde bir fonksiyon işlevine göre küçük parçalara ayrılır. Fonksiyonun yaptığı her iş ayrı düğümlere ayrılır. Veriler bu düğümler arasında akmaktadır.

vi. Büyük ve aktif bir topluluk desteği bulunmaktadır.

vii. Gazebo ve Rviz ortamlarıyla tam uyumlu çalışması sayesinde kodları benzetim ortamında test edebilirsiniz, böylece hızlı bir geliştirme sağlanabilir.

viii. Farklı dillerle yazılmış kodları beraber çalıştırabilirsiniz.



Şekil 2.2. ROS işletim sisteminin diğer sistemler ile gösterilmesi.

2.1.1.1. *ROS Terminolojisi*

1. *Master;* Düğümler arası bağlantı ve haberleşme için bir isim sunucusu gibi

davranır. Master olmadan düğümler arası bağlantı ve haberleşme mümkün olmamaktadır.

2. *Node(Düğüm);* ROS içindeki en küçük birim gibi düşünülebilir. Düğümler publisher veya subscriber gibi davranabilirler.

3. *Topic(Konu);* Yayın yapan düğüm öncelikle kendi konusunu (topic) mastera iletir ve bu topic'e yayın yapmaya başlar. Bu yayını dinlemek isteyen başka bir düğüm master içerisinde bu topic'i dinlemeye başlayabilir.

4. *Publisher;* bu node, mastera kendi bilgilerini ve hangi konuda yayın yaptığını iletir ve kendisine bağlanan düğümlere mesajları gönderir.

5. *Subscriber;* bu node kendi bilgisini ve dinlemek istediği topic'i mastera iletir ve masterdan bu topic'i yayın yapan düğüme ait bilgileri alır. Bu bilgilere bakarak dinlemek istediği düğüme doğrudan bağlanır ve yayın yaptığı mesajı almaya başlar. Topic haberleşmeleri eş zamanlı olarak yapılmaz.

6. *Roscore;* Bu komut ROS Master'ı çalıştırmaktadır. Eğer ağda birden fazla bilgisayar var ise sadece tek bir yerde bu komutun çalıştırılması yeterlidir. Master'a ait IP adresi lokal IP adres, PORT ise 11311 olarak başlatılacaktır.

7. *Rosrun;* Bu komut ROS'un temel çalıştırma komutudur. Tek bir düğüm çalıştırmak için kullanılır.

8. *Roslauch;* rosrun komutunun aksine birden fazla düğümü tek seferde çalıştırmak için kullanılır. Bu komut launch uzantılı dosya kullanır. Bu dosyanın içerisinde çalıştırılacak düğümleri olmalıdır.

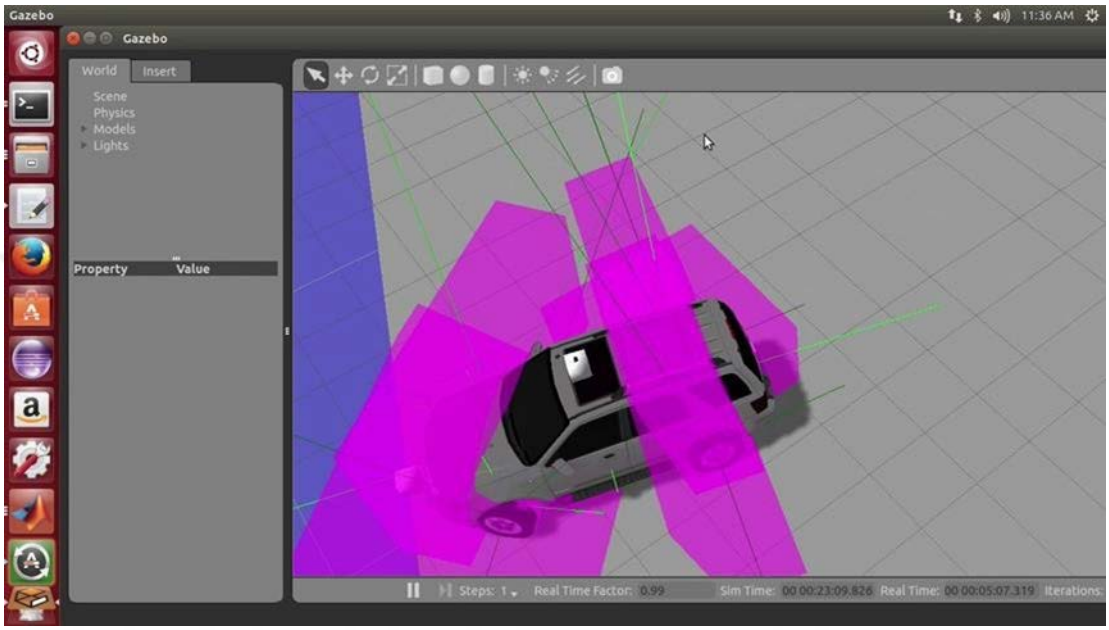
9. *Bag;* ROS içerisinde mesajlar bag formatında kaydedilebilir. Bu kayıt. bag uzantılı bir dosyaya kaydedilir. Bu mesajlar daha sonra tekrar kullanılabilir. Örnek olarak araçtan bir kere lidar verisi kullanılarak bag formatında kaydedilirse, daha sonra bu veriler aracı tekrar sürmeden veya lidarı çalıştırmadan kullanılabilir.

2.1.2. Gazebo

Robot simülasyonu, her robotikçinin araç kutusundaki önemli bir araçtır. İyi tasarlanmış bir simülatör, algoritmaları hızlı bir şekilde test etmeyi, robotları tasarlamayı, regresyon testi yapmayı ve AI sistemini gerçekçi senaryolar kullanarak eğitmeyi mümkün kılar. Gazebo, karmaşık iç ve dış ortamlarda robot popülasyonlarını doğru ve verimli bir şekilde

simüle etme yeteneği sunar. Gazebo projesi 2002 yılında California'da Dr. Andrew Howard ve öğrencisi Nate Koenig tarafından başlatılmıştır. Linux ortamında geliştirilmiştir [30].

Fiziksel ortamlarda gürültüsüz veriye ulaşılması mümkün değildir. Bu yüzden simülasyon ortamında ideal robot tasarımı için algılayıcıların gürültülü veri üretmesini sağlanması gerekmektedir. Gazebo gürültü eklenmiş algılayıcılara imkân sağlamaktadır.



Şekil 2.3. Gazebo simülasyon ortamının bir görüntüsü.

Bir Gazebo simülasyonu, karmaşık iç ve dış ortamlarda robot popülasyonlarını doğru ve verimli bir şekilde simüle etme becerisine sahip bir 3D simülatör olan Gazebo ile yapılan bir robot simülasyonudur. Oyun motorlarına benzer, ancak daha iyi simülasyonlar üretmektedir. Hem kullanıcılar hem de programlar için bir dizi algılayıcı ve arayüz sunar. Aşağıdaki ana bileşenlere sahiptir:

- *World files:* Bir simülasyondaki tüm unsurları içermektedir; robotlar, ışıklar, algılayıcılar ve statik nesneler.
- *Models:* bireysel unsurları temsil eder. Üç robot ve önündeki nesne modeldir.
- *gzserver:* "iş atı" Gazebo programıdır dene bilir. Bir dünya oluşturmak ve doldurmak için dünya dosyasını okur.
- *gzclient:* gzserver'a bağlanır ve elemanları görselleştirir. Kabuk çıktısında, "NODES" altında hem gzserver hem de gzclient listelenmiştir.
- *gzweb:* WebGL kullanarak, gzclient web sürümüdür.

Gazebo ortamı robotları, robotlara zarar vermeden zor veya tehlikeli senaryolarda değerlendirip test edebilebilme imkânı sunar. Çoğu zaman gerçek robotta tüm senaryoyu başlatmak yerine bir simülatör çalıştırmak daha hızlıdır ve daha güvenlidir. Başlangıçta Gazebo, robotların algoritmalarını değerlendirmek için tasarlanmıştır. Çok robotlu bir simülatöre ihtiyaç duyulduğundan Gazebo geliştirilmiş ve iyileştirilmiştir. Birçok uygulama için, robot uygulamasını hata işleme, pil ömrü, yerelleştirme, navigasyon ve kavrama gibi test etmek önemlidir ve tüm bunlar Gazebo dünyalarında test edilebilir.
ROS robot geliştiricileri için çok işlevli bir araç olan Gazebo aşağıdakileri desteklemektedir [31]:

- Robot modellerin tasarlanması
- Hızlı prototipleme ve algoritmaların test edilmesi
- Gerçekçi senaryolar kullanarak regresyon testi
- İç ve dış ortamların simülasyonu
- Lazer telemetre, 2D / 3D kameralar için algılayıcı verilerinin simülasyonu, kinect tarzı algılayıcılar, kontak algılayıcıları, kuvvet-tork ve daha fazlası
- Nesne Yönelimli Grafikler kullanan gelişmiş 3D nesneler ve ortamlar Render Motoru
- Gerçek dünya dinamiklerini modellemek için birkaç yüksek performanslı fizik motoru (Açık Dynamics Motoru- ADM (Open Dynamic Engine-ODE), Bullet, Simbody ve Dinamik Canlandırma ve Robotik Araç Seti-DCRS (Dynamic Animation and Robotics Tools-DART))

TurtleBot3 simülasyonda sanal bir robotla programlanabilen ve geliştirilebilen geliştirme ortamını desteklemektedir. Bunu yapmak için iki geliştirme ortamı vardır, biri sahte düğüm ve 3D görselleştirme aracı Rviz, diğeri ise 3D robot simülatörü Gazebo'yu kullanmak.
Sahte düğüm yöntemi, robot modeli ve hareketi ile test etmek için uygundur, ancak algılayıcıları kullanamaz. SLAM ve Navigasyon'u test etmek içinse, simülasyonda IMU, LDS ve kamera gibi algılayıcılar kullanabilen Gazebo kullanılmalıdır.

2.1.3. Rviz (Robot İşletim Sistemi Görselleştiricisi)

Rviz, ROS için güçlü bir 3D görselleştirme aracıdır. Kullanıcının benzetilmiş robot modelini görüntülemesini, robot algılayıcılarından algılayıcı bilgilerini kaydetmesini ve

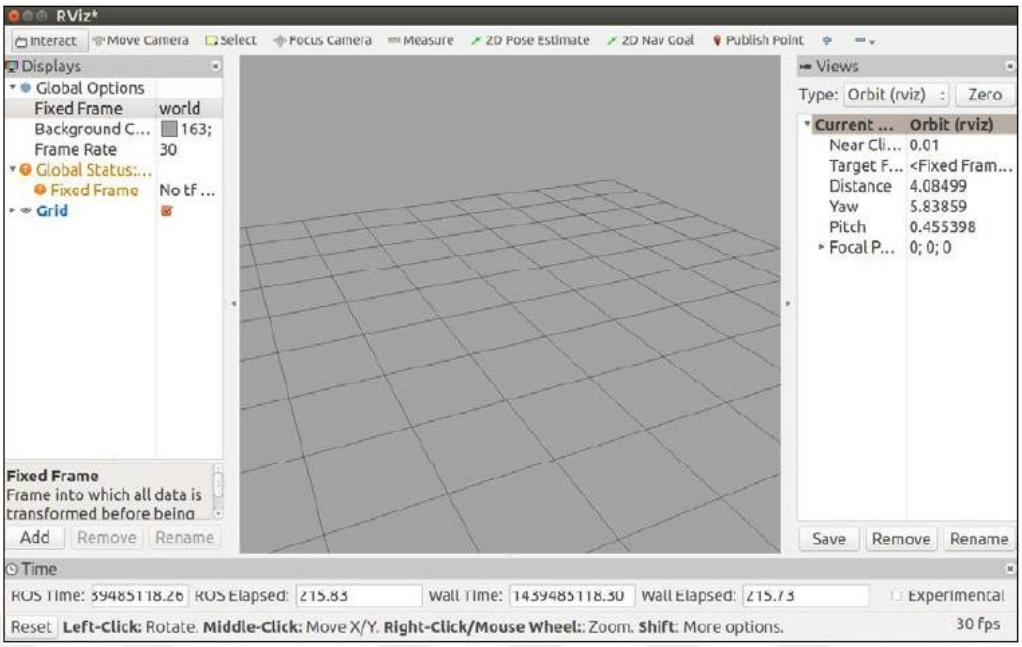
kayıtlı algılayıcı bilgilerini tekrar oynatmasını sağlar. Robotun ne gördüğünü, düşündüğünü ve yaptığını görselleştirerek, kullanıcı algılayıcı girişlerinden planlı (veya planlanmamış) eylemlere kadar bir robot uygulamasında hata ayıklayabilir.

Rviz, stereo kameralardan, lazerlerden, Kinektlerden ve diğer 3D cihazlardan gelen 3D algılayıcı verilerini nokta bulutları veya derinlik görüntüleri şeklinde görüntüler. Web kameralarından, RGB kameralardan ve 2D lazerli uzaklık ölçerlerden gelen 2D algılayıcı verileri, görüntü verisi olarak Rviz'de görüntülenebilir.

Gerçek bir robot, Rviz çalışan bir iş istasyonuyla iletişim kuruyorsa, Rviz, robotun mevcut yapılandırmasını sanal robot modelinde görüntüler. ROS konuları, herhangi bir kamera, kızılötesi algılayıcı ve robot sisteminin bir parçası olan lazer tarayıcılar tarafından yayınlanan algılayıcı verilerine dayanarak canlı gösterimler olarak gösterilecektir. Bu robot sistemlerini ve kontrol cihazlarını geliştirmek ve hata ayıklamak için yararlı olabilir. Rviz, kullanıcının yalnızca mevcut görevle ilgili bilgileri görüntülemesini sağlamak için yapılandırılabilir bir Grafiksel Kullanıcı Arabirimi-GAK (Graphical User Interface-GUI) sağlar. Gazebo'da modelimizi 3D çevre çevresinde hareket ettirirken fiziğin modelimiz üzerindeki etkilerini Rviz üzerinde görebiliriz [32].

Rviz ana ekranındaki merkezi pencere, 3B ortamın dünya görüşünü sunar. Genellikle, ortadaki pencerede yalnızca ızgara görüntülenir veya pencere boştur.

Ana ekran dört ana ekran alanına bölünmüştür Şekil 2.4. Rviz çalışma ortamı.: orta pencere, soldaki Ekranlar paneli, sağdaki Görünümler paneli ve altta bulunan Zaman paneli. Bu ekran alanlarının üst kısmında araç çubuğu ve ana ekran menü çubuğu bulunur. Rviz ana ekranın bu alanlarının her biri aşağıdaki bölümlerde açıklanmıştır. Bu genel bakış, Rviz GUI ile aşinalık kazanmanız için sağlanmıştır.



Şekil 2.4. Rviz çalışma ortamı.

Rviz özelliklerinden bazıları kısaca şu şekilde sıralanabilir [33];

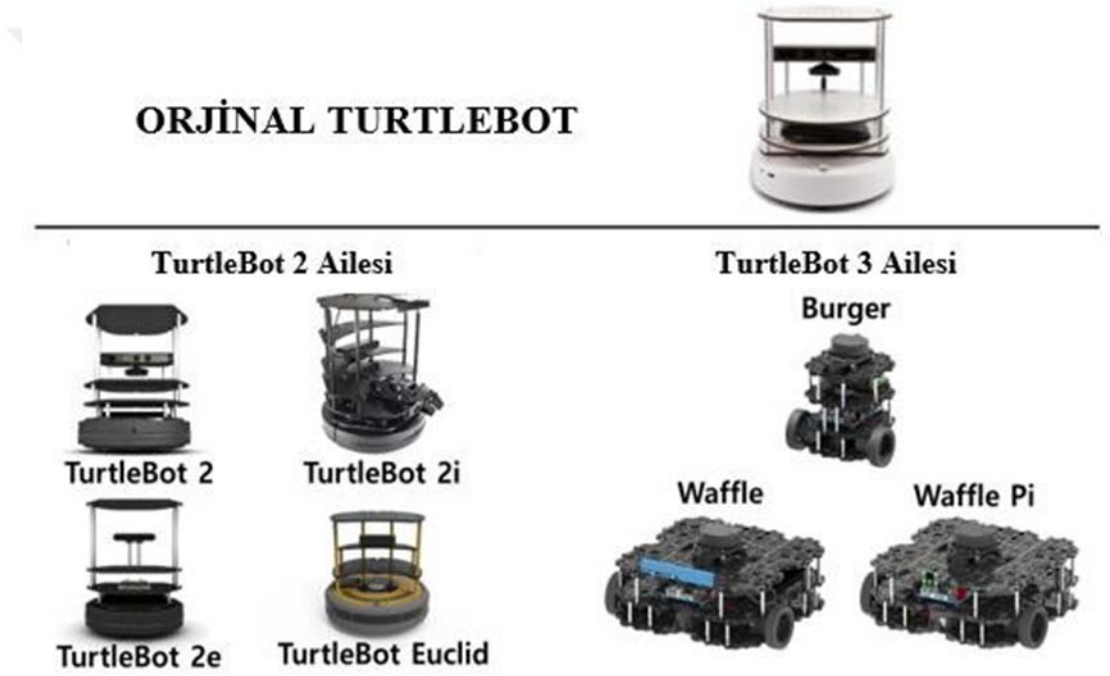
- Rviz, bir vizyon sisteminde neyin yanlış gittiğini bulmak için mükemmel bir ortamdır. Soldaki listede her öğe için bir onay kutusu bulunur. Herhangi bir görsel bilgiyi anında gösterilebilir veya gizlenebilir.
- Rviz, sadece fare ile gezinebilen 3D görselleştirme sağlar.
- Rviz' in en iyi kısmı etkileşimli işaretleyicidir. Bu gerçekten yaratıcı olunabilecek kısımdır. 3D göreceli olarak belirli bir alanı seçmeyi kolaylaştırır. Bu nedenle, görüş alanını, çalışma alanı gibi belirli bir alanı seçmek ve diğer bölgeleri yok saymak gibi hala çalışıyorken manuel olarak ayarlanabilir.
- Rviz' de görsel verilerini gösteren çoklu görsel işlemlere sahip olunabilir. ROS yayınlama yöntemini kullanarak göstermek istediğiniz nokta bulutu veya şeklini yayınlamak yeterlidir.

2.2. TURTLEBOT3

Bu tezde TurtleBot3 robotunun simülasyon ortamında kullanılmıştır. TurtleBot, ROS standart platformlu bir robottur. Turtle, 1967'de eğitici bilgisayar programlama dili logosu tarafından sürülen Turtle robotundan türetilmiştir. Kaplumbağa simgesini ROS'un bir sembolü olarak kullanılmaktadır. ROS logosunda kullanılan dokuz nokta, kaplumbağanın arka kabuğundan elde edilmiştir. Logo TurtleBot kaynaklıdır ve TurtleBot aracılığıyla

ROS'a yeni başlayan insanlara kolayca öğretmek ve bilgisayar programlama dilini öğretmek için tasarlanmıştır. O zamandan beri TurtleBot, geliştiriciler ve öğrenciler arasında en popüler platform olan ROS'un standart platformu oldu.

TurtleBot serisinin 3 versiyonu vardır (Şekil 2.5. TurtleBot ailesi ve çeşitleri.). TurtleBot1 ROS yayılımı/oluşturulması için iRobot'un Roomba-dayalı araştırma robotunun başında olan Willow Garage'dan Tully(Open Robotics'te Platform Yöneticisi) ve Melonee(Fetch Robotics'in CEO'su) tarafından geliştirilmiştir ve 2011'den beri satıştatır. 2012'de, TurtleBot2, araştırma robotu iClebo Kobuki'ye dayanan Yujin Robot tarafından geliştirilmiştir. 2017 yılında, TurtleBot3, seleflerinin eksik işlevlerini ve kullanıcıların taleplerini tamamlayacak özelliklerle geliştirilmiştir.



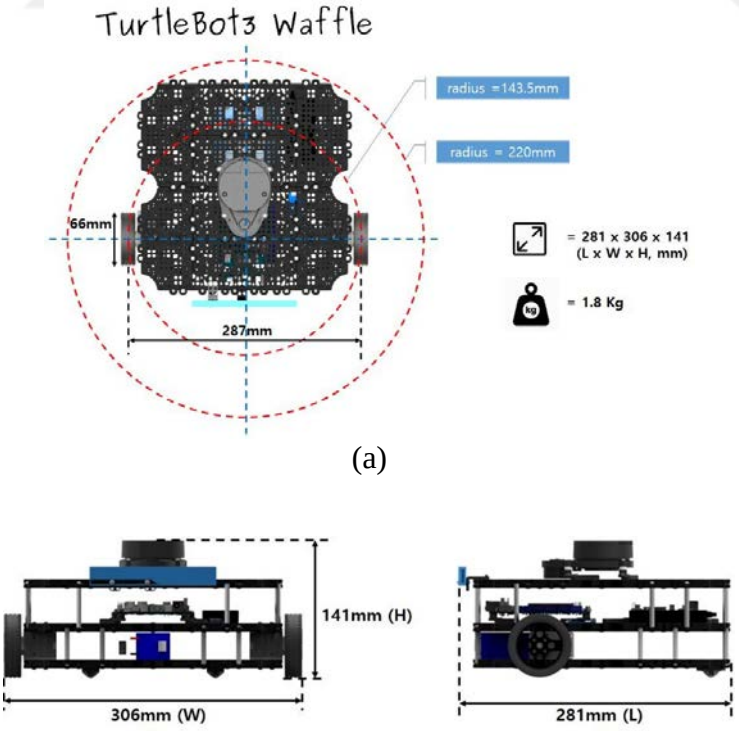
Şekil 2.5. TurtleBot ailesi ve çeşitleri.

TurtleBot3, eğitim, araştırma, hobi ve ürün prototiplemesinde kullanım için küçük, uygun fiyatlı, programlanabilir, ROS tabanlı bir mobil robottur. TurtleBot3'ün amacı, platformun boyutunu önemli ölçüde azaltmak ve aynı zamanda genişletilebilirlik sunarken, işlevselliğini ve kalitesini ödün vermeden fiyatı düşürmektir. TurtleBot3, mekanik parçaları nasıl yeniden yapılandırdığınıza ve bilgisayar ve algılayıcı gibi isteğe bağlı parçaları kullanmanıza bağlı olarak çeşitli şekillerde özelleştirilebilir. Ek olarak, TurtleBot3, sağlam gömülü sistem, 360 derece mesafe algılayıcı ve 3D baskı teknolojisi için uygun maliyetli ve küçük boyutlu SBC ile geliştirilmiştir.

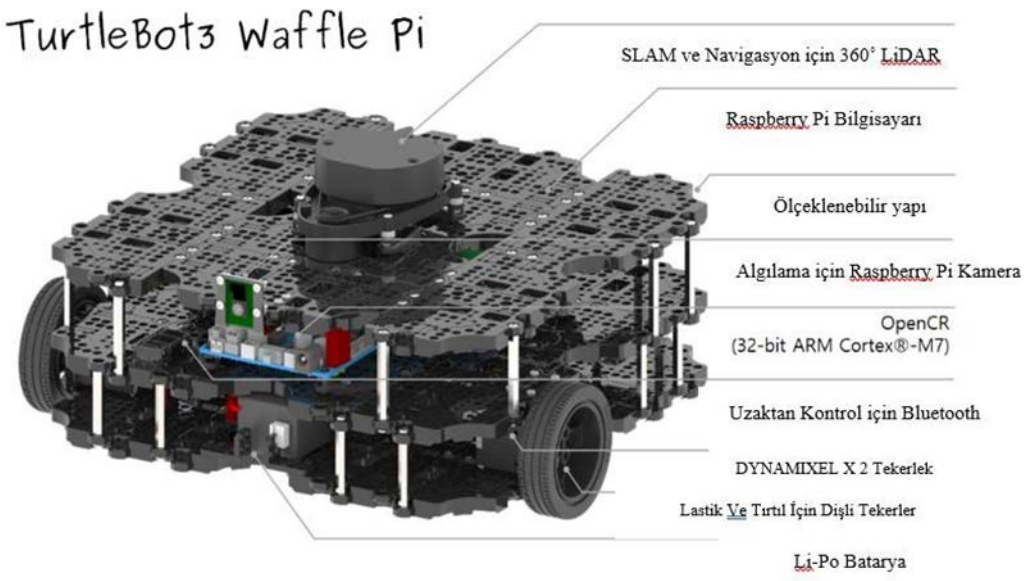
TurtleBot3'ün ana teknolojisi SLAM, Navigasyon ve Manipülasyondur. TurtleBot, bir harita oluşturmak için SLAM algoritmalarını çalıştırabilir. Ayrıca, bir dizüstü bilgisayar, joypad veya Android tabanlı bir akıllı telefondan uzaktan kontrol edilebilir. TurtleBot, bir kişi odada yürüdüğünde onun bacaklarını da takip edebilir. Ayrıca TurtleBot3, OpenMANIPULATOR gibi bir manipülatör ekleyerek bir nesneyi manipüle edebilen bir mobil manipülatör olarak kullanılabilir. OpenMANIPULATOR, TurtleBot3 Waffle ve Waffle Pi ile uyumlu olma avantajına sahiptir [34].

Bu tez simülasyonda TurtleBot3 ailesinde Waffle Pi kullanılmıştır. Turtlebot3 Waffle Pi, robotik öğreticileri ve araştırmacılar için tasarlanmıştır ve farklı çeşitlerde uygulamaların gerçekleştirimi için uygundur. Çoğunlukla otonom navigasyon (çizgi takip ve engel tespiti) ve robot manipülasyonu için yapılmış mobil bir robottur.

Tıpkı TurtleBot3 Burger gibi, Waffle Pi da bugüne kadar dünyanın en popüler robot işletim sistemi olan ROS'u kullanmaktadır. Waffle Pi eğitim ve araştırma mobil robotu son derece özelleştirilebilir ve tamamen açık kaynak kodlu (donanım ve yazılım) olma özellikleriyle, deneysel robotik üzerinde çalışanlar için gerçek bir destekçi olma yetisine sahiptir.



Şekil 2.6. (a) Waffle Pi üsten görünüş ve ölçüleri, (b) Waffle Pi yandan görünüş ve ölçüleri.



Şekil 2.7. Waffle Pi bazı teknik özellikleri.

TurtleBot3 Waffle Pi robot teknik özellikleri;

- Maksimum dönüşüm hızı: 0,26 m / s
- Maksimum dönme hızı: 1.82 rad / s (104,27 ° / s)
- Maksimum taşıma kapasitesi: 30 kg
- Boyutlar: 281 x 306 x 141 mm
- Ağırlık (programlama kartı, akü ve algılayıcılar ile): 1,8 kg
- Çalışma süresi: yaklaşık 2 saat
- Şarj süresi: yaklaşık 2 saat 30 m
- 2 Dinamiksel XM430-W210-T servolar
- Mikrodenetleyici: Raspberry Pi 3
- Gömülü denetleyici: Open-CR (32-bit ARM® Cortex®-M7)
- Raspberry Pi kamera
- LiDAR 360 ° lazer tarayıcı

3. KULLANILAN YÖNTEMLER

3.1. EŞ ZAMANLI LOKASYON VE HARİTALAMA

Eş zamanlı Lokalizasyon ve Haritalama (Simultaneous Localization and Mapping - SLAM) aynı zamanda Eşzamanlı Haritalama ve Yerelleştirme (Concurrent Mapping and Localization - CML) olarak da bilinir ve robotun bir haritadaki yerini belirlerken bir çevre haritası oluşturma zorunluluğuyla ilgilenir. SLAM, bir mobil robotun bir çevre haritası oluşturabileceği ve aynı zamanda konumunu belirlemek için bu haritayı kullanabileceği bir süreçtir. Başlangıçta hem harita hem de araç konumu bilinmemektedir, araç bilinen bir kinematik modele sahiptir ve yapay ya da doğal yerler ile doldurulan bilinmeyen ortamda hareket etmektedir. Hem robot hem de yer işareti konumlarının eşzamanlı tahmini gerekir. Bunu yapabilmek için, araç, yer işaretleri ile aracın kendisi arasındaki göreceli konumun ölçümlerini alabilen duyusal bir sistemle donatılmalıdır [35].

3.1.1. Eş Zamanlı Lokasyon ve Haritalama Tarihi

1986'da San Francisco, California'da düzenlenen IEEE Robotik ve Otomasyon Konferansında 'Olasılıksal Eş Zamanlı Lokasyon ve Haritalama (PSLAM)' problemi ilk kez ele alındı. O yıllar olasılıklı yöntemlerin hem robotik hem de yapay zekaya-YZ (artificial intelligence-AI) henüz yeni çalışılmaya başlandığı zamanlardı. Peter Cheeseman, Jim Crowley ve Hugh Durrant Whyte'nın da içinde bulunduğu bazı araştırmacılar, haritalandırma ve yerelleştirme problemlerine tahmin-kuramsal yöntemler uygulamak istiyorlardı [36]. Bu konferansın sonucunda, tutarlı-olasılıksal haritalamanın, temel kavramlar ve hesaplamalarla etraflıca robotikte temel bir sorun olarak ele alınması gerektiğine karar verildi. Devam eden birkaç yıl boyunca bu konu ile ilgili bir dizi anahtar makale üretildi. Smith ve Cheesman [37] ve Durrant-Whyte [38] tarafından yapılan çalışmalarda, işaretler ve geometrik belirsizlik yönlendirme arasındaki ilişkiyi tanımlamak için istatistiksel bir temel oluşturdular. Aynı zamanlarda Ayache ve Faugeras [39] görsel navigasyonda öncü çalışmalar yürüttüler. Crowley [40] ve Chatila ve Laumond [41], Kalman filtre tipi algoritmalar kullanarak mobil robotların sonar tabanlı navigasyonunda çalıştılar.

Eş zamanlı haritalama ve lokasyonun en temel problemi robotun/aracın haritalama yapabilmesi için bulunduğu ortamdaki yerini(lokasyonunu) bilmesi gerektiği, robotun/aracın lokasyonunu bilebilmesi içinde ortamın haritasına ihtiyacı olmasıydı ('Tavuk mu yumurtadan çıkar? Yumurtamı tavuktan?' handikabı gibi). Bu karmaşıklıktan dolayı SLAM üzerine teorik çalışmalar geçici olarak durdu, devam eden çalışmalarda ise genellikle ya haritalama ya da lokasyon ayrı ayrı problemler olarak ele alındı.

Daha önce tek bir tahmin problemi olarak formüle edilen eş zamanlı haritalama ve lokasyon probleminin aslında yakınsak olduğunun ortaya çıkışı kavramsal bir dönüm noktası oldu. En önemlisi, çoğu araştırmacının küçültmeye çalıştığı işaretler arasındaki ilişkinin aslında sorunun kritik bir parçası olduğu ve küçültmenin aksine, bu ilişki büyüdükçe çözümün daha iyi olduğu kabul edildi. SLAM probleminin yapısı, yakınsama sonucu ve kısaltması SLAM'ın ilk olarak 1995 Uluslararası Robotik Araştırmaları Sempozyumu'nda sunulan bir mobil robotik araştırma makalesinde sunuldu [42]. Yakınsama ile ilgili temel teori ve ilk sonuçların çoğu, Csorba [43], [44] tarafından geliştirilmiştir. Haritalama ve lokasyon üzerinde çalışmakta olan birçok grup, özellikle de Massachusetts Institute of Technology [45], Zaragoza [46], [47], Sydney'deki ACFR [48], [49] ve diğerleri [50], [51], iç mekân, dış mekân ve denizaltı ortamlarında aynı zamanda eşzamanlı haritalama ve yerelleştirme olarak da adlandırılan SLAM üzerinde en ciddi şekilde çalışmaya başladı.

1999'da Uluslararası Robotik Araştırmaları Sempozyumu (ISRR99) [52] ilk SLAM oturumunun yapıldığı ve Thrun [53] tarafından tanıtılan Kalman filtre bazlı SLAM yöntemleri ile olasılıksal lokalizasyon ve haritalama yöntemleri arasında bir dereceye kadar yakınlaşma elde edilmesi açısından önemli bir buluşma noktasıydı.

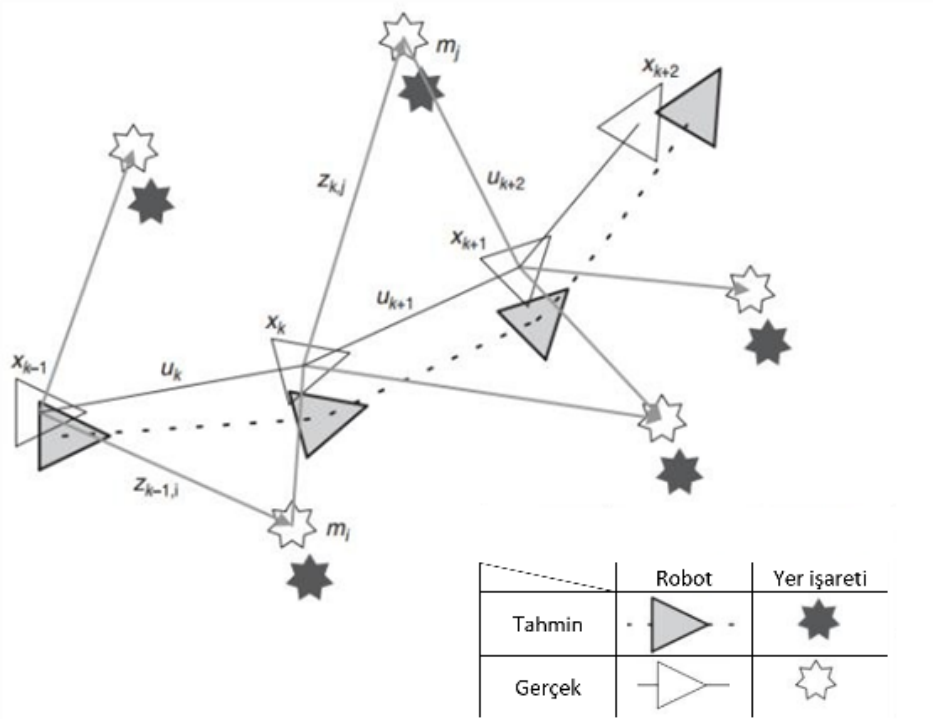
İlki 2000 IEEE Robotik ve Otomasyon Konferansı (ICRA) Çalıştayı olmak üzere devam eden birkaç yıllar boyunca her sene artan katılımcısı ile SLAM için çalışma kampları dünyanın birçok ülkesinde gerçekleştirildi ve bu alanın inşasında büyük bir başarı elde ettiler.

3.1.2. SLAM Probleminin Formülasyonu ve Yapısı

SLAM, bir mobil robotun bir çevre haritası oluşturabileceği ve aynı zamanda konumunu belirlemek için bu haritayı kullanabileceği bir süreçtir. SLAM 'da hem platformun yörüngesi hem de tüm yer işaretlerinin yerleri önceden belirlenmiş bir konum bilgisine ihtiyaç duyulmadan aktif olarak tahmin edilmektedir.

Robot üzerinde bulunan ve Şekil 3.1. Temel Slam probleminin şekil üzerimde gösterimin'de olduğu gibi bir algılayıcı kullanarak, bilinmeyen yer işaretlerinin göreceli olarak gözlemlendiği bir ortamda hareket eden bir mobil robot olduğunu varsayalım. Bir k anı için nicelikler aşağıdaki gibi tanımlanabilir:

- $\mathbf{x_k}$: aracın konumunu ve yönünü tanımlayan durum vektörü
- $\mathbf{u_k}$: aracı k zamanında x_k durumuna getirmek için $k-1$ anında uygulanan kontrol vektör
- $\mathbf{m_i}$: gerçek konumu zamanla değişmeyen kabul edilen yer işaretinin konumunu tanımlayan bir vektör
- $\mathbf{z_{ik}}$: k anında yer işaretinin bulunduğu yerin araçtan yapılmış bir gözlem. Herhangi bir zamanda birden fazla yer işareti gözlemi olduğunda veya belirli bir yer işareti olmadığında, gözlem basitçe z_k olarak ifade edilecektir.



Şekil 3.1. Temel SLAM probleminin şekil üzerinde gösterimi [35].

Hem robot hem de yer işareti konumlarının eşzamanlı tahmini gerekir. Gerçek yerler hiçbir zaman doğrudan bilinmez veya ölçülmez. Gerçek robot ve dönüm noktası yerler arasında gözlemler yapılır.
Buna ek olarak, aşağıdaki gibi tanımlamalar yapılabilir:

Aracın lokasyon geçmişi;

$$X_{0:k} = \{x_0, x_1, \dots , x_k \} = \{ X_{0:k-1} , x_k\} \quad (3.1)$$

Kontrol girişlerinin geçmişi;

$$U_{0:k} = \{u_0, u_1, \dots , u_k \} = \{ U_{0:k-1}, u_k\} \quad (3.2)$$

Tüm yerler işaretçilerin kümesi;

$$m = \{m_1, m_2, \dots , m_n \} \quad (3.3)$$

Tüm yer işaretçilerinin gözlem kümesi;

$$Z_{0:k} = \{z_1, z_2, \dots , z_k \} = \{ Z_{0:k-1} , z_k\} \ (z_0 = 0) \quad (3.4)$$

3.1.2.1. *Olasılıksal SLAM Formülasyonu*

Olasılık formunda eşzamanlı lokalizasyon ve harita oluşturma (SLAM) problemi olasılık dağılımını tüm k anları için şu şekilde ifade edebiliriz;

$$P(x_0, m \mid Z_{0:k}, U_{0:k}, x_0) \quad (3.5)$$

Bu olasılık dağılımı, kaydedilen gözlemler ve aracın ilk durumu ile birlikte k anına kadar ve k anıda dâhil olmak üzere, kaydedilen gözlemler ve kontrol girdileri göz önüne alındığında, yer işareti konumlarının ve araç durumunun arka eklem yoğunluğunu tarif eder. Genel olarak, SLAM problemine tekrarlamalı bir çözüm istenmektedir. Denklem (3.5)'de verilen olasılık dağılımı için k-1 anında bir tahminle başlayarak, bir kontrol u_k ve gözlem z_k'yi izleyen arka eklem Bayes teoremi kullanılarak hesaplanır. Bu hesaplama, sırasıyla kontrol girişi ve gözlem etkisini açıklayan bir durum geçiş modeli ve gözlem modelinin tanımlanmasını gerektirir.

Gözlem modeli, taşıt konumu ve yer işaretleri konumları bilindiğinde ve bir gözlem z_k genellikle aşağıdaki formda tanımlandığı gibi yapma olasılığını açıklar.

$$P(z_k \mid x_k, m) \tag{3.6}$$

Aracın konumu ve haritasının tanımlanmasından sonra, harita ve mevcut araç durumu göz önüne alındığında gözlemlerin şartlı olarak bağımsız olduğunu varsaymak mantıklıdır.

Taşıt için hareket modeli, durum geçişlerinde olasılık dağılımı aşağıdaki formdaki gibi tanımlanabilir;

$$P(x_k \mid x_{k-1}, u_k) \tag{3.7}$$

Yani, durum geçiş sırasında sonraki durum x_k, sadece hemen bir önceki x_{k-1} durumuna bağlıdır ve uygulanan kontrol u_k gözlem ve harita bağımsız olan bir Markov süreci olduğu kabul edilir.

SLAM algoritması artık standart iki aşamalı 'tekrarlamalı(sıralı)- tahmin (zaman-güncelleme)- düzeltme (ölçüm-güncelleme)' formunda denklem (3.8)'deki gibi gösterilebilir,

Zaman güncelleme:

$$P(x_k, m \mid Z_{0:k-1}, U_{0:k}, x_0) = \int P(x_k \mid x_{k-1}, u_k) x P(x_{k-1}, m \mid Z_{0:k-1}, U_{0:k-1}, x_0)\, dx_{k-1} \tag{3.8}$$

Ölçüm güncelleme:

$$P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0) = \frac{P(z_k \mid x_k, m)\, P(x_k, m \mid Z_{0:k-1}, U_{0:k}, x_0)}{P(z_k \mid Z_{0:k-1}, U_{0:k})} \tag{3.9}$$

Denklem (3.8) ve (3.9); k anına kadar ve k anı dâhil bütün gözlemler $Z_{0:k}$ ve $U_{0:k}$'ya dayalı

robot durumu x_k ve harita m için arka eklem $P(x_k, m \mid Z_{0:k}, U_{0:k}, x_0)$ 'sını hesaplamak için, bir tekrarlı prosedürü sağlamaktadır. Tekrarlamalı, $P(x_k \mid x_{k-1}, u_k)$ araç modelinin ve $P(z_k \mid x_k, m)$ gözlem modelinin bir işlevidir.

Harita oluşturma probleminin, koşullu yoğunluğu $P(m \mid X_{0:k}, Z_{0:k}, U_{0:k})$ hesaplamak için formüle edilebileceğine dikkat edilmelidir. Burada, x_k aracının konumunun, her zaman ilk konum bilgisine tabi olarak bilindiğini (veya en azından belirleyici) olduğunu varsayılır. Bir harita m, daha sonra farklı konumlardan gözlemlerin kaynaştırılmasıyla oluşturulur. Bunun aksine, lokalizasyon sorunu olasılık dağılımını $P(x_k \mid Z_{0:k}, U_{0:k}, m)$ ile formüle edilebilir. Bu, merkezi konumların kesin olarak bilindiğini varsayar ve amaç bu noktalara göre araç konumunun bir tahminini hesaplamaktır.

3.1.2.2. *Olasılıksal SLAM Yapısı*

SLAM yapısını basitleştirmek için, Denklem (3.5)'deki tarihsel değişkenler üzerinde koşullandırma bırakılacak ve harita üzerinde ve araç konumundaki gerekli arka eklem bağlamın izin verdiği şekilde $P(x_k, m \mid z_k)$ ve hatta $P(x_k, m)$ olarak yazabiliriz.

Gözlem modeli $P(z_k \mid x_k, m)$, gözlemlerin hem araç hem de önemli noktalardaki konumlara bağımlılığını açıkça ortaya koymaktadır. Denklem (3.10)' da verilen ifade arka eklemin bariz bir şekilde bölümlenemediğini ifade etmektedir;

$$P(x_k, m \mid z_k) \neq P(x_k \mid z_k)\, P(m \mid z_k) \tag{3.10}$$

İlk yapılan çalışmalarda ki tutarlı haritalama [35], [36] konusundaki ifadelerle karşılaştırıldığında, denklem (3.10) da verilen ifadenin tutarsız tahminlere yol açacağı bilinebilmektedir. Ancak, SLAM problemi, bu denklemlerden anlaşılacağından daha fazla yapıya sahiptir.

Şekil 3.1'e bakıldığında, tahmin edilen ve gerçek yer işaretleri konumları arasındaki hatanın aslında tek bir kaynak yüzünden yani yer işaretleri gözlemleri yapıldığında robotun nerede olduğuna dair bilgi hataları sebebiyle olduğu görülebilmektedir. Bu, yer işaretlerindeki lokasyon tahminleri ile hataların yüksek oranda ilişkili olduğunu göstermektedir. Pratik olarak, bu, herhangi bir yer işareti $P(m_i - m_j)$ arasındaki göreceli konumun, bir dönüm noktası m'nin mutlak konumu oldukça belirsiz olsa bile, yüksek

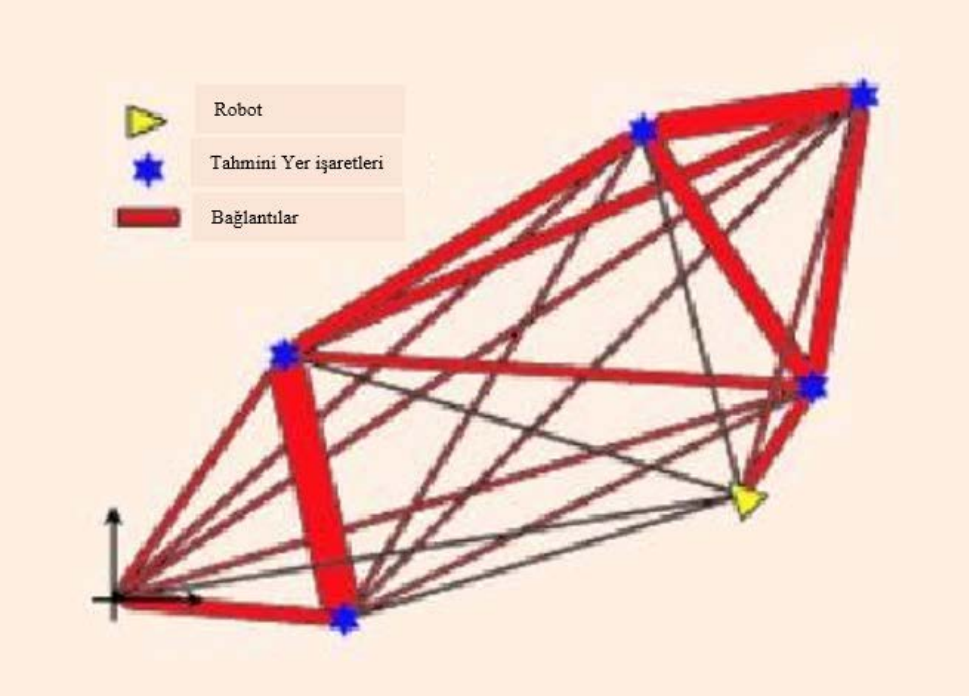
doğrulukta bilinebileceği anlamına gelir. Olasılık formunda, bu, P (m_i- m_j) işaret çifti için birleşik olasılık yoğunluğunun, marjinal yoğunluklar P (m_i) oldukça dağılabildiği zaman bile yüksek oranda doruğa ulaştığı anlamına gelmektedir.

SLAM konusundaki en önemli kavrama, yer işaretleri tahminleri arasındaki ilişki daha fazla gözlem yapıldıkça monoton bir şekilde arttığının farkına varıldı. Bu sonuçlar yalnızca doğrusal Gauss durumu için kanıtlanmıştır [54]. Daha genel olasılıklı durumlar için resmi kanıt açık bir sorun olarak kalmaktadır. Pratik olarak, bu, yer işaretlerinin göreceli konumu hakkındaki bilginin, robot hareketinden bağımsız olarak daima iyileştirildiği ve hiçbir zaman ayrılmadığı anlamına gelmektedir. Olasılık terminolojisi ile bu, P(m) işaretlerindeki bütün olasılıkların ortak olasılık yoğunluğunun, daha fazla gözlem yapıldıkça monoton olarak yoğunlaştığı anlamına gelmektedir.

Bu yakınsama, robot tarafından yapılan gözlemlerin yer işaretleri arasındaki göreceli konumun "neredeyse bağımsız" ölçümleri olarak kabul edilebileceği için oluşur. Yine Şekil 3.1'e bakıldığında, iki yerin mi ve m_j'yi gözlemleyerek robotu x_k konumunda olduğunu düşünelim. Gözlenen yer işaretlerinin göreceli konumu açıkça aracın koordinat çerçevesinden bağımsızdır ve bu sabit konumdan gelen art arda yapılan gözlemler, yer işaretleri arasındaki göreceli ilişkinin daha bağımsız ölçümlerini sağlayacaktır. Şimdi, robot xk+1 konumuna hareket ettikçe, tekrar m_j işaretini gözlemler, bu, robotun ve yer işaretinin tahmini konumunun önceki x_k konumuna göre güncellemesini sağlar. Bu da dönüm noktası yeni yerden görülmese bile m_i 'yi güncellemek için tekrar yayılır. Bu, iki yer işaretinin önceki ölçümlerden yüksek oranda ilişkili olması (göreceli konumlarının iyi bilinmesi) nedeniyle oluşur. Ayrıca, aynı ölçüm verilerinin bu iki noktayı güncellemek için kullanılması gerçeği, onları daha fazla ilişkili hale getirir. Neredeyse bağımsız ölçüm terimi, gözlem hatalarının ardışık araç hareketleriyle ilişkilendirileceği için uygundur. Ayrıca, x_{k+1} konumundaki Şekil 3.1'de robotun m_j 'ye göre iki yeni yer işareti gözlemlediğine dikkat edin. Bu yüzden bu yeni işaretler derhal bağlanır veya haritanın geri kalanıyla ilişkilendirilir. Bu yer işaretlerinde daha sonra yapılan güncellemeler ayrıca m_j ve mi bu yer işaretleri aracılığıyla da güncellenecektir. Yani, tüm işaretler, bir gözlem yapıldığında ki göreceli konum veya ilişiğinde ki kesinliği veya değeri artan bir ağ oluşturur.

Bu işlem, Şekil 3.2'de ki gibi tüm yer işaretlerini birbirine bağlayan bir yay ağı veya tüm yer işaretlerinin gömülü olduğu bir lastik örtü olarak görselleştirilebilir. Robot bu çevrede ilerledikçe ve yer işaretlerini gözlemledikçe, yer işaretleri arasındaki yaylar gittikçe artar

(ve monoton olarak) sertleşir. Limitte, sabit bir yer haritası veya çevrenin doğru bir bağıl haritası elde edilir. Harita oluşturuldukça, haritaya göre ölçülen robotun konum doğruluğu yalnızca haritanın kalitesi ve bağıl ölçüm algılayıcı ile sınırlıdır. Teorik sınırda, robot göreceli konum doğruluğu, belirli bir harita ile elde edilebilecek yerelleştirme doğruluğuna eşit olur. Yer işaretleri, yerler arasındaki ilişkiyi tanımlayan yaylarla bağlanır. Araç çevrede ileri geri hareket ettikçe, yay sertliği veya ilişiği artmaktadır (kırmızı bağlantılar kalınlaşır). Yer işaretleri gözlemlenip tahmini yerler düzeltildikçe, bu değişiklikler yay ağı yoluyla yayılır. Robotun kendisinin de harita ile ilişkili olduğuna dikkat edilmelidir [36].



Şekil 3.2. SLAM haritalamada ki yay ağı benzetimi [35].

3.1.3. SLAM Teknikleri

Birkaç araştırmacı SLAM'da çalışmış ve en sık kullanılan algılayıcılar lazer tabanlı, sonar tabanlı ve görsel tabanlı sistemlerde sınıflandırılabilir olduğunu ifade etmişlerdir. Robot durumu bilgisini ve dış dünyayı, pusulalar, kızılötesi teknoloji ve Global Konumlandırma Sistemi (GPS) gibi daha iyi algılamak için ek duyusal kaynaklar kullanılabilir olduğunu belirtmişlerdir [35]. Bununla birlikte, tüm bu algılayıcılar, genellikle ölçüm gürültüsü olarak adlandırılan belirli hatalar taşırlar. Ayrıca çevrede gezerken gerekli olan çeşitli sınırlamalara sahiptirler; örneğin, ışık ve ses duvarlara nüfuz edemez.

- *Lazer sınıflandırma sistemleri;* hassas etkin algılayıcılardır. En bilinen şekli ile uçuş prensibi sırasında nesneye doğru dar bir ışın içinde bir lazer darbesi göndererek ve darbenin hedeften yansıtılması ve gönderene geri gönderilmesi için geçen süreyi ölçerek çalışır.
- *Sonar tabanlı sistemler;* hızlıdır ve görüşe benzer miktarda bilgi içerirler ancak görünüm verisi eksikliğinde olan ölçümler ve tanıma kapasiteleri vardır. Bununla birlikte, kilometre sayacı gibi atalet algılayıcılarına bağımlılığı, küçük bir hatanın sonraki konum tahminleri üzerinde büyük etkileri olabileceği anlamına gelmektedir [55].
- *Görüş sistemleri;* pasif, uzun menzilli ve yüksek çözünürlüğe sahiptirler, ancak hesaplama maliyeti oldukça yüksektir. İyi görsel özelliklerin elde edilmesi ve eşleştirilmesi daha zordur. Görüş sistemi, 3D yapısını, özellik konumunu ve robot pozunu, örneğin stereo kamera çiftleri veya hareket kurtarma işleminden yapılı monoküler kameralar aracılığıyla tahmin etmek için kullanılır.

3.1.4. SLAM Filtreleri

Robotik harita oluşturma 25 yıl öncesine kadar izlenebilir. 1990'lardan bu yana olasılıklı yaklaşımların yani, Kalman Filtreleri (KF), Parçacık Filtreleri (PF) ve Beklenti Maksimizasyonu (EM) SLAM'da baskın hale gelmiştir. Bu üç teknik tekrarlamalı Bayes kuralının matematiksel türevleridir. Bu olasılıksal teknik popülaritesinin ana nedeni, robot haritalamanın belirsizlik ve algılayıcı gürültüsü ile karakterize edilmesi ve olasılık algoritmaları, farklı gürültü kaynaklarını ve bunların ölçümler üzerindeki etkilerini açıkça modelleyerek sorunu çözmektedir [55].

3.1.4.1. *Kalman Filtresi (KF)*

Kalman filtreleri, Gauss kullanan sonsal, yani az sayıda parametre ile kompakt bir şekilde temsil edilebilecek tek biçimli çok değişkenli dağılımları temsil eden Bayes filtreleridir. KF SLAM, durum geçişi ve ölçüm fonksiyonlarının ilave Gauss gürültüsü ile lineer olduğu ve ilk sonsal Gauss olduğu varsayımına dayanır. Modern SLAM'de iki ana KF varyasyonu vardır: Genişletilmiş Kalman Filtresi (EKF) ve ilgili Bilgi Filtreleme (IF) veya Genişletilmiş IF (EIF). EKF, doğrusal hareketleri kullanarak robot hareket modeline yaklaşarak, gerçek dünyadaki doğrusal olmayanları barındırır. Mevcut çeşitli SLAM yaklaşımları EKF [56]-[59] kullanılmıştır. IF, durum hata kovaryansı matrisinin tersini

yaymak suretiyle uygulanır. IF filtresinin KF'ye göre birkaç avantajı vardır. Öncelikle, veri, bilgi matrislerini ve vektörünü basitçe toplayarak filtrelenir ve daha kesin tahminler sağlar [60]. İkincisi, IF, KF'den daha kararlıdır [61]. Son olarak, EKF, yüksek boyutlu haritaları tahmin ederken nispeten yavaş çalışmaktadır, çünkü her araç ölçümü genellikle Gauss'ın tüm parametrelerini etkiler, bu nedenle güncellemeler birçok yer işaretine sahip ortamlarla uğraşırken engelleyici zamanlar gerektirir [62].

Kokusuz Kalman Filtresi (UKF) [63], EKF'nin yaklaşım sorunlarını ve KF'nin doğrusallık varsayımlarını ele almaktadır. KF, lineer durumlarda doğru şekilde çalışır ve Gauss Rastgele Değişkenini (GRV) doğrusal bir sistem dinamiği üzerinden analitik olarak yaymak için etkili bir yöntem olarak kabul edilir. Doğrusal olmayan modellerde EKF, dinamik denklemleri doğrusallaştırarak en uygun terimlere yaklaşır. EKF, optimum çözüme birinci dereceden bir yaklaşım olarak görülebilir. Bu yaklaşımlar, gerçek arka ortalama ve kovaryansta büyük hatalara neden olabilir ve bu da bazen filtrenin sapmasına neden olabilir. Bu örnek noktaları, GRV'nin gerçek ortalamasını ve kovaryansını tamamen yakalar ve gerçek doğrusal olmayan sistemin içinden geçirildiğinde arka ortalamayı ve kovaryansı herhangi bir doğrusal olmayan için 3. sıraya doğru olarak yakalar. Bunu yapmak için, kokusuz dönüşüm kullanılır.

EKF ve KF uygulamasının ana dezavantajlarından biri, uzun süreli görevler için yer imi sayısının artması ve sonuç olarak da haritanın gerçek zamanlı olarak güncellenmesi için bilgisayar kaynaklarının yeterli olmamasıdır. Bu ölçeklendirme sorunu, her bir yer işareti diğer tüm yer işaretleri ile ilişkili olduğu için ortaya çıkmaktadır. Bağıntı, yeni bir yer işaret gözleminin mobil robot üzerine monte edilmiş bir algılayıcıyla elde edilmesinden dolayı ortaya çıkmaktadır. Bu nedenle yer işareti konum hatası, araç konumundaki hata ve haritanın diğer yer işaretlerindeki hatalarla ilişkilendirilecektir. Bu bağıntılar, algoritmanın uzun vadeli yakınsaması için temel öneme sahiptir ve görevin tamamı boyunca sürdürülmesi gerekmektedir.

Sıkıştırılmış Genişletilmiş Kalman Filtresi (CEKF) [62] algoritması, sonuçların doğruluğunda herhangi bir ceza vermeden hesaplama gereksinimini önemli ölçüde azaltır. Bir CEKF, yerel bir alanda toplanan tüm bilgileri, bölgedeki yer işaretlerinin sayısının karesine orantılı bir maliyetle depolar ve saklar. Bu bilgiler daha sonra, tam SLAM'a benzer ancak sadece bir tekrarlama ile aynı maliyetle dünya haritasının geri kalanına aktarılır.

KF ve varyantlarının avantajı, Robot ve merkezi konumlar durumunun optimal Minimum Ortalama Kare Hata (MMSE) tahminlerini sağlaması ve kovaryans matrisinin güçlü bir şekilde birleştiği görülüyor. Bununla birlikte, Gauss gürültüsü varsayımı KF'nin veri birliği ve yer işareti sayısı için uyarlanabilirliğini sınırlamaktadır.

3.1.4.2. *Parçaçık Filtresi (PF)*

Sıralı Monte-Carlo (SMC) yöntemi olarak da adlandırılan PF, Monte Carlo simülasyonlarında uygulanan tekrarlamalı bir Bayes filtresidir. SMC tahminini, Bayes arkasını temsil eden bir dizi rastgele nokta kümesi ("parçacıklar") ile yürütülür. Parametrik filtrelerin (örneğin, KF) aksine, PF, bu dağılımdan alınan bir dizi numune dağılımını temsil eder, bu da onu lineer olmayan algılayıcıları ve Gauss olmayan gürültüyü kullanma yeteneğine sahiptir. Bununla birlikte, bu yetenek, yeni işaretler tespit edildikçe durum boyutunda hesaplamalı karmaşıklıkta bir büyüme üreterek gerçek zamanlı uygulamalar için uygun değildir [65]. Bu nedenle, PF sadece lokalizasyona başarıyla uygulanmıştır, yani robotun konumunu ve yönünü belirlerken, harita yapımı için, yani yer işaret konumu ve yöneliminde uygulamamıştır. Bu nedenle, tüm SLAM çerçevesi için PF'yi kullanan önemli bir makale yoktur, ancak PF'nin diğer tekniklerle bir kombinasyonu kullanılarak yapılan çalışmalar mevcuttur; örneğin, FastSLAM [65] ve fastSLAM2.0 [66]. FastSLAM SLAM probleminin (bilinen veri birliği ile), yer işaretleri tahminleri, robotun yoluna bakıldığında koşullu olarak bağımsız olma özelliğinden faydalanır [67]. FastSLAM algoritması, SLAM problemini bir robot lokalizasyon problemine ayırır ve robot üzerinde şartlandırılmış bir yer işaretleri tahmin problemleri derleme yapmaktadır. FastSLAM'ın temel bir özelliği, her parçacığın kendi yerel veri birliğini oluşturmasıdır. Buna karşılık, EKF teknikleri, tüm filtre için tek bir veri ilişkilendirme hipotezine bağlı kalmalıdır. Ek olarak, FastSLAM, robot yollar üzerinde örnekleme yapmak için standart bir EKF veya KF'den daha az bellek kullanımı ve hesaplama süresi gerektiren bir parçacık filtresi kullanır.

3.1.4.3. *Beklenti Maksimizasyonuna (EM)*

EM, maksimum olasılık (ML) tahmini bağlamında geliştirilen ve haritalama ve konumlandırma için ideal bir çözüm sunan istatistiksel bir algoritma yöntemidir. EM algoritması, örneğin robotun içinde bulunduğu ortamı bildiğinde, beklentileri olan bir harita oluşturabilir [68]. EM iki aşamayı yinelemektedir: belirli bir harita için arkadan robotun üzerinde pozlar oluşturduğu bir beklenti adımı (E-adım) ve bu poz beklentileri

göz önüne alındığında en muhtemel haritanın hesaplandığı maksimizasyon adımı (M-adım). Nihai sonuç giderek daha doğru haritalar dizisidir. EM'in KF'ye göre temel avantajı, yazışma problemini (veri birliği problemi) şaşırtıcı derecede iyi çözebilmesidir [55]. Bu, E-adımda mevcut haritaya göre robotu art arda lokalize etmesi ve robotun nerede olabileceği konusunda çeşitli hipotezler yaratması sayesinde mümkündür (farklı haberleşmede olabilir). İkinci M-adımında, bu yazışmalar haritadaki özelliklere çevrilir, daha sonra bir sonraki E-adımda güçlendirilir veya yavaş yavaş kaybolur. Bununla birlikte, aynı verileri en muhtemel haritayı elde etmek için birkaç kez işleme tabi tutma ihtiyacı, verimsiz kılar, artımlı ve gerçek zamanlı uygulamalar için uygun değildir [69]. Kesikli yaklaşımlar kullansanız bile, robotun pozunu tahmin ederken, maliyet haritanın boyutuyla katlanarak artar ve hata sınırlanmaz. Bu nedenle elde edilen harita uzun çevriminden sonra kararsız hale gelir. Eğer veri birliği biliniyorsa [70], E-adım basitleştirilmiş veya elimine edilmişse, aynı olan bu problemlerden kaçınılabilir. Bu nedenle EM, genellikle robotun olabileceği poz tahminini temsil eden bir dizi parçacık (örnek) ile arkaları temsil eden PF ile birleştirilir. Örneğin, bazı pratik uygulamalar, konumlandırmayı farklı yollarla yaparken mesela kilometre sayacı okumalarından gelen pozları tahmin etmek için PF'ye dayalı lokalizatör kullanılabilir [55], haritayı oluşturmak için EM'yi kullanır (yalnızca M adımını).

Çizelge 3.1. SLAM çerçevesine uygulanan filtreleme yaklaşımlarının avantaj ve dezavantajlarının listesi [55].

Avantajlar	*Dezavantajlar*
Kalman Filtresi/Genişletilmiş Kalman Filtresi (KF/EKF) [57]-[59]	
- Yüksek yakınsama - Belirsizlikle başa çıkmak	- Gauss varsayımı - Yüksek boyutlu haritalarda yavaş
Sıkıştırılmış Genişletilmiş KF (CEKF) [63]	
- Belirsizliği azaltır - Hafıza kullanımının azaltılması - Geniş alanları ele alabilir - Harita tutarlılığını arttır	- Çok sağlam özellikler gerektirir - Veri birliği sorunu - Çoklu harita birleştirme gerektirir
Bilgi Filtreleme (IF) [60],[61]	
- Kararlı ve basit - Doğru - Yüksek boyutlu haritalar için hızlı	- Veri ilişkilendirme sorunu - Durum tahminlerini iyileştirmek gerekebilir - Yüksek D hesaplamalı olarak pahalı
Parçacık Filtreleme (PF) [65],[66]	
- Doğrusal olmayanları ele almak - Gauss olmayan gürültüyü ele almak	- Karmaşıklıktaki büyüme
Beklenti Maksimizasyonu (EM) [60],[68]	
- Harita oluşturma için en uygun - Veri ilişkisini çöz	- Verimsiz, maliyet artışı - Büyük senaryolar için kararsız - Sadece harita oluşturmada başarılı

3.2. GENETİK ALGORİTMA

Optimizasyon problemlerine çözüm önerisi sunmak isteyen araştırmacılar çoğunlukla doğayı ve canlıları taklit eden algoritmalar geliştirmişlerdir. Karınca, böcek ve arı gibi canlılar incelenerek, bu canlıların davranışları bilgisayarda modellenmiş ve birçok problem ve sistem üzerinde uygulanmıştır. Yapay Zekâ'nın hızla gelişen bir dalı olan Genetik Algoritma da evrimsel sürecin bilgisayar dilinde modellenmiş bir optimizasyon yöntemidir [71]. GA evrim teorisinden etkilenerek geliştirilmiştir. Diğer bir değişle GA doğal seçim ilkelerine dayanan optimizasyon ve arama metodudur denebilir. GA, bir problem için en iyi çözümü bulmaya çalışan algoritmalardır.

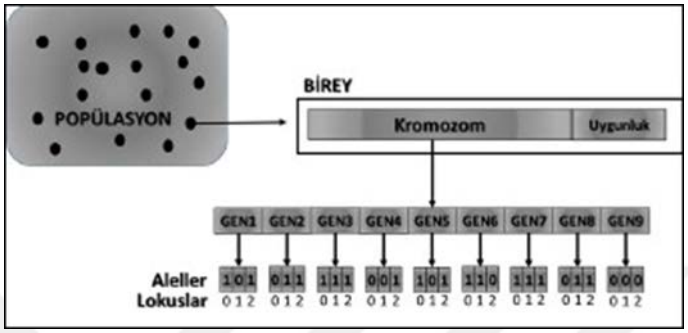
GA ile ilgili ilk çalışmalar 1950'li yıllarda biyoloji ve bilgisayar uzmanlarından oluşan bir grubun ortak projesi olarak başlamıştır. 1960'larda John Holland ve bir grup meslektaşı Michigan Üniversitesi'nde genetik algoritmaların ve optimizasyonun temel ilkeleri oluşturulmuştur. GA ilgili teorik kısmı ve uygulamaları hakkında birçok uluslararası konferans da düzenlenmektedir. GA, hücresel üretim, çizelgeleme, mekanik öğrenme, fonksiyon optimizasyonu, tasarım gibi alanlarda başarılı uygulamaları bulunmaktadır.

GA da çözüm uzayının hepsi değil yalnızca bir kısmını taranır. Bu şekilde etkin arama yapılmış olur ve çözüme çok daha kısa bir sürede ulaşılır. Problemi değişkenlerinin fazla olduğu durumlarda sıklıkla kullanılır. GA olasılık kurallarına göre çalışır. Bu sebeple ne kadar iyi çalışacağı önceden tahmin edilemez. Kıstaslar değiştirilerek probleme uygun çözüm/ler aranır. GA önemli bir özelliği de muhtemel çözümlerden oluşan popülasyonu eş zamanlı incelemeleri ve yerel en iyi çözüme takılmamalarıdır. Nüfus nesilden nesile geliştikçe iyi çözümler daha iyi çözümler oluşturmak için kullanılma eğilimindeyken, kötü çözümler ise yok olma eğilimindedir. Diğer bir değişle, GA en iyi çözümün hayatta kalma ilkesine bağlı olarak çözüm kümesini oluşturur.

Genetik sistemlerin tartışılması, doğrudan sağlamlığa dayanan iki genel gereksinimi vurgulamıştır; (1) Uyarlanabilir akış diyagramı, daha önceki plan-çevre etkileşimlerinin geçmişinin bölümleriyle birlikte, daha önce yapılmış olan ilerlemeleri korumalıdır. (2) Akış diyagramı, geçmiş tamamı uzadıkça üretilen uyum yapılarının oranını artırmak için tutulan tarihi kullanmalıdır [72].

GA, tek bir çözüme değil de bir çözüm kümesine odaklanarak çalışır. Bu çözüm kümesi popülasyonu farklı bireylerden oluşur. Popülasyondaki her bir birey bir çözümü temsil

eder. Bireyler yeni bireyler oluşturmak için çaprazlamaya tabi tutulmalarıyla en iyi olan bireylerin çözümü oluşturmasına dayanır. Çözüm kümesindeki bireyler oluşturduğu topluluğa popülasyon, popülasyonu oluşturan her bir çözüme kromozom, kromozomların hesaplanan değerine uygunluk değeri, bir çözüm oluşturan değişkene gen, genlerin kromozom içindeki konumlarına lokus, genlerde saklan bilgilere de alel denilmektedir [74].



Şekil 3.3. Popülasyon içindeki bir kromozom gösterimi [73].

3.2.1. Genetik Algoritma ile İlgili Temel Kavramlar

GA'nın nasıl çalıştığı anlatılmadan önce algoritmada geçen kavramlar açıklanacaktır:

a) GEN: Genetik bilgi taşıyan ve kendi başına anlamı olan ve en küçük genetik birimdir.

   i. Bir gen 0 veya 1 ile ifade edilen bir bit veya bit dizisi olabileceği gibi A, B gibi bir karakter olabilir. Örneğin bir cismin x koordinatındaki yerini gösteren bir gen 101010 şeklinde ifade edilebilir.

b) KROMOZOM: Bir veya daha fazla genin birleşimi ile oluşurlar. Sorunun tüm bilgilerini içerirler. Kromozomlar bireylere veya toplum üyelerine karşılık gelirler. Probleme alternatif çözümler için adaydırlar.

c) POPÜLASYON (Topluluk): Bireyler veya kromozomlar topluluğudur. Nüfus sorunu için alternatif çözümler kümesidir.

d) KODLAMA: Çözümlerin kromozomlarla nasıl oluşturulması gerektiğini gösterir.

   i. İkili Kodlama: İkili dizi ile ifade edilen her bir kromozom 0 ve 1'lerden oluşan bit dizisidir.

Kromozom A → 10101110

Kromozom B → 11010011

ii. Permütasyon Kodlama: Bu kodlama iş sıralama problemleri ve gezgin satıcı problemi gibi sıralama istenen problemlerde kullanılır.

Kromozom A → 30278601

Kromozom B → 75230410

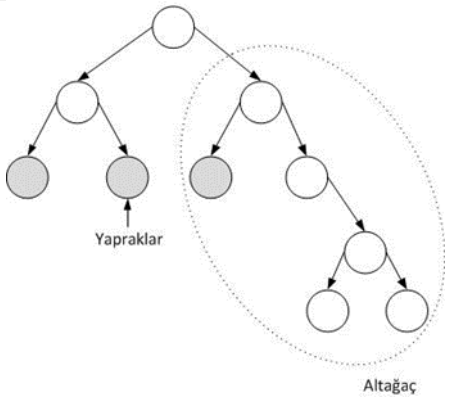
iii. Değer Kodlama: Bu kodlama gerçek gibi karmaşık sayıların yer aldığı problemlerde kullanılır.

Kromozom A → 4.2 6.1 0.9 8.3

Kromozom B → ABCDE

Kromozom C → (ileri), (sağ), (geri), (sol)

iv. Ağaç kodlama: Değişen gelişen programlar veya değerler için kullanılır. (Şekil 4.2. Ağaç kodlama şekil benzetimi)



Şekil 3.4. Ağaç kodlama şekil benzetimi.

e) SEÇİLİM: Yeni nesili oluşturmak için mevcut nüfustan sırasıyla çaprazlama ve mutasyon işlemine tabi tutulacak bireylerin seçilmesi gerekmektedir. En bilinen yöntemleri;

i. Rulet Seçilimi: Seçim yapılacak nüfusta ki tüm bireylerin uygunluk değerleri toplanır. Her bireyin seçilme olasılığı, uygunluk değerinin bu toplam değere oranı kadardır.

ii. Sıralı Seçilim: En kötü uygunlukta olan kromozoma 1 değeri verilir, ondan daha iyi olana 2, daha iyisine 3 değeri verilerek devam edilir.

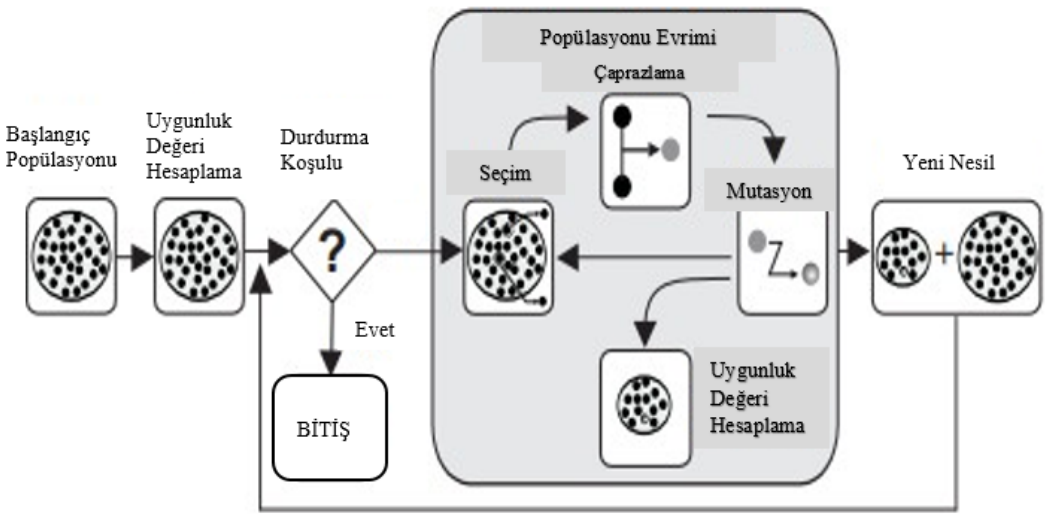
iii. Turnuva Seçilimi: Topluluk içerisinden rastgele t adet (3,6,9..) birey alınır. Bu bireylerin içerisinden uygunluk değeri en iyi olan birey seçilir.

f) ÇAPRAZLAMA: Amaç, ata kromozomun yerlerini değiştirerek çocuk kromozomlar üretmek ve böylelikle zaten uygunluk değeri yüksek olan ata kromozomlardan daha yüksek uygunluklu çocuk kromozomlar üretmektir. Çaprazlama sonuncunda çocuk kromozomlar ata kromozomlar ile aynı değildir lakin ataya ait iyi özellikleri taşır. Çaprazlama GA performansını en çok etkileyen en önemli faktörlerden biridir.

g) MUTASYON: Kromozomların kendi genleri veya genleri oluşturan küçük birimleri üzerinde değişiklik yapılmasını sağlayan işlemcidir. GA'da değişimin sağladığı avantaj, problemin çözüm alanını araştırmada yön değişikliklerini sağlayarak 'Mutasyon (Değişim)' yardımıyla araştırmanın kısır döngüye girmesini önlemektir. Mutasyon en basit haliyle bir boyuttaki bilginin terslenmesiyle yapılabilir.

h) ELİTİZM: Mevcut popülasyondaki uygunluk değeri en iyi olan birey/lerin olduğu gibi yeni topluluk havuzuna aktarılma işlemidir.

i) SONUÇ: Her kuşakta, GA, çaprazlama ve mutasyon gibi genetik operatörleri kullanarak yeni bir popülasyon oluşturur. Topluluktaki bireylerden birisi istenilen sonucu veriyor ise algoritma sona ermektedir. Eğer istenilen sonuç elde edilmediyse algoritma yeni popülasyon oluşturarak istenilen sonuç elde edilene kadar devam eder.



Şekil 3.5. Tipik bir GA'nın çalışma prensibi [74].

Birçok alanda uygulamaları olan GA akış adımları şu şekilde sıralanabilir (*Şekil 3.5. Tipik bir GA'nın çalışma prensibi [74].*);

i. Arama uzayındaki tüm olası çözümler dizi olarak kodlanır.

ii. Genellikle rastgele seçilen bir çözüm kümesi başlangıç popülasyonu olarak kabul edilir. Başlangıç popülasyonu içindeki her çözüm bir bireyi temsil etmektedir. Popülasyonun büyüklüğü aynı zamanda çözüm kümesinin de büyüklüğünü göstermektedir.

iii. Her bir dizi için bir uygunluk değeri hesaplanır, bulunan uygunluk değerleri dizilerin çözüm kalitesini gösterir.

iv. Bulunan uygunluk değerlerinin algoritmayı sonlandırma özelliğine sahip olup olmadığına bakılır. Eğer istenilen çözüme ulaşıldıysa algoritmadan çıkılır. En iyi çözüm; çözüm kümesi içerisinde ki en iyi bireydir.

v. Eğer istenilen çözüme ulaşılmadıysa yeni popülasyonun oluşturulması gerekmektedir. Bir grup dizi belirli bir olasılık değerine göre rastgele olarak seçilip çoğalma işlemi gerçekleştirilir.

vi. Yeni bireylerin uygunluk değerleri hesaplanarak, çaprazlama ve mutasyon işlemlerine tabi tutulur. Burada amaç; seçilen çözüm kümesinden daha iyi çözümler üretebilmektir.

vii. Çaprazlama işlemi popülasyonun çeşitliliğini artırmaktadır. Mutasyon işlemi ise çözüm kümesinin kısır döngüye girmesini engeller.

viii. Yeni oluşturulan popülasyon içindeki her bir dizi için uygunluk değeri hesaplanır. Bulunan uygunluk değerlerinin algoritmayı sonlandırma özelliğine sahip olup olmadığına bakılır. Eğer istenilen çözüme ulaşıldıysa algoritmadan çıkılır. En iyi çözüm; çözüm kümesi içerisinde ki en iyi bireydir.

ix. Eğer uygun çözüm bulunmadıysa en son belirlenen popülasyon ile yukarıdaki işlemler devam ettirilir.

x. Tekrarlama, belirlenen kuşak sayısına ulaşınca işlem sona erdirilir.

Her yöntemde olduğu gibi GA'nın da güçlü ve zayıf yönleri bulunmaktadır. Güçlü yönleri;

i. Optimizasyon problemlerinin çözümüne uygundurlar,

ii. Çok çeşitli değişken tiplerine uygundurlar,

iii. Açıklanabilir sonuçlar elde edilebilir,

iv. Yapay Sinir Ağları ile uyumlu çalışmaları söylenebilir.

Zayıf yönleri;

i. Bilgisayar hesaplamaları çok fazla zaman ve kaynak kullanımı gerektirebilir,

ii. Bazı problemlerin modellenmesi zordur,

iii. Optimum çözümün bulunmasını garantilemez,

iv. Ticari paket programlarını sayısının az olması söylenebilir. [75]

4. GENETİK ALGORİTMANIN SİSTEME UYGULANMASI

Simülasyonun yapıldığı bilgisayarın sahip olduğu bazı özellikleri söylemek gerekirse;

- İntel i7 7300 serisi işlemci
- 16 GB ve 2400 MHz RAM
- 2GB ekran kartı

Bu tezde benzetim ortamı için bilgisayarda bazı programların kurulumu yapılmıştır. Bilgisayar üzerinde VMware Workstation programı kuruldu. Bu program üzerinden iki adet sanal makine oluşturuldu. Sanal makinelerden birisine 1 CPU, 1 GB RAM, ve 128 MB ekran kartı özellikleri verildi. Diğer makinede benzetim programları kullanılacağı için 2 CPU, 10 GB RAM, ve 1 GB ekran kartı özelliklerine sahip olacak şekilde oluşturuldu.

Simülasyonun yapılacağı sanal bilgisayarlara Linux-Ubuntu 16.04 kurulmuştur. En dengeli olması sebebiyle ve bir sorun ile karşılaşıldığında çözümünün kolay olması sebebiyle terminal üzerinden ROS-kinetic kurulumu gerçekleştirilmiştir [76]. Daha sonra yönergeler [77] takip edilerek TurtleBot3 yüklemesi yapılmıştır. ROS'un tüm paketleri içerisinde Gazebo ve Rviz benzetim ortamları otomatik olarak bilgisayara yüklenmektedir. Bu benzetim ortamlarının kurulumu için ayrıca bir kurulum yapılmamıştır.

Be tezde önerilen yöntemimiz de aracı daha önceden bilmediği bir ortama bırakarak öncelikle ortamın haritasını çıkartmaktır. İkinci aşamada haritası çıkarılan ortam üzerinden aracın harici olarak istenen noktaya en kısa seçimini Genetik algoritma kullanılarak yol seçim işleminin optimize edilmesidir.

Gerekli ayarlamalar ve yüklemeler yapıldıktan sonra TurtleBot3 Waffle Pi aracı kullanılarak ve ROS paketlerinden yararlanılarak adım adım şu işlemler gerçekleştirilmiştir;

i. Bilgisayar üzerinden terminal açılarak roscore komutu çalıştırılır.

ii. İlk önce klavye ile aracı kontrol etmek için araç 1.terminalden gerekli kodlarla

çağrılır.

iii. 2.terminalden Gazebo ortamında haritasının çıkarılmasını istediğimiz alan çağrılır.

iv. 3.terminalden Rviz ortamı açılır. Aracın üzerinde bulunan lidardan gelen ilk verilere göre haritayı oluşturmaya başlar.

v. 4.terminalde TurtleBot3 Waffle Pi aracının otonom sürüşü için gerekli kodlar girilir. Araç ortamda bulunan hiçbir engele çarpmadan ortamın haritasını çıkarmaya başlar.

vi. Açılan 5.terminalde harita çıkarma işlemi bittikten sonra harita masaüstüne kaydedilir.

vii. Haritalama işlemleri bittiği için bütün terminallere ctrl+c yapılarak, Gazebo ve Rviz dâhil bütün programlar ve terminaller kapatılır.

viii. Genetik algoritma da kaydedilen harita kullanılmak için masaüstünde Genetik algoritma dosyasına kopyalanır ve burada derleme işlemi gerçekleştirilir.

ix. Yeni bir terminal açılarak araç çağrılır.

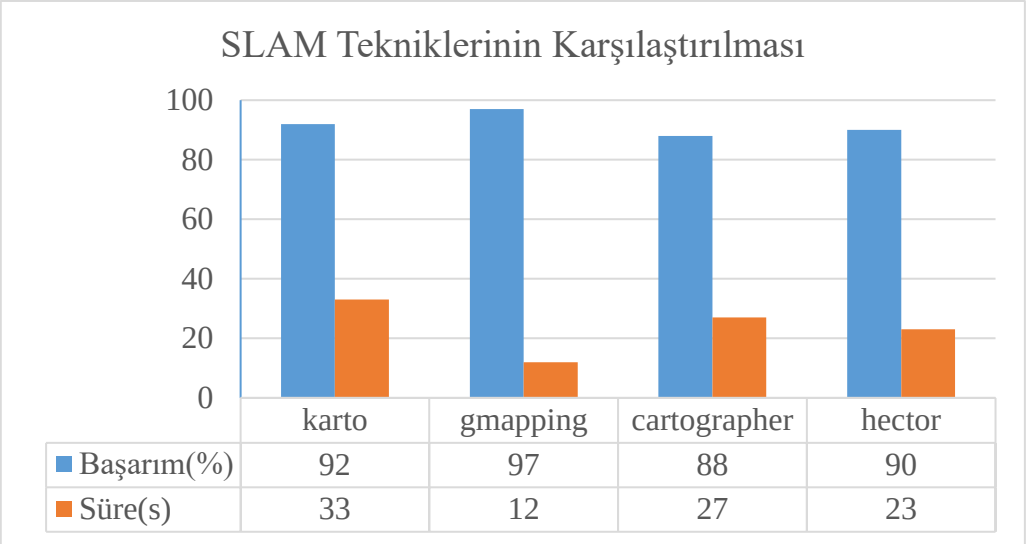
x. 2.yeni bir terminalden haritasını çıkardığımız Gazebo ortamı tekrar çağrılır.

xi. 3.terminalden optimum yer belirleme işlemi simülasyonunu yapacak olan Rviz ortamı çağrılır. Rviz'de üst menü seçeneklerinden '2D Nav Goal' tuşuna basılarak harita üzerinde herhangi bir yere (aracın gitmesini istediğimiz yere) tıklanır. Robot bir süre hareketsiz kalıp harita üzerindeki yerini belirliyor ve daha sonra gitmesi istenen nokta ve kendisi arasında ki en kısa yolu bulup ilerlemeye başlıyor. Yeni hedef vermek için tekrar üst menü seçeneklerinden '2D Nav Goal' yardımıyla yeni rota hesaplanması istenebilir.

xii. Tüm istenen işlemler bittikten sonra bütün terminallere ctrl+c yapılarak, Gazebo ve Rviz dâhil bütün programlar ve terminaller kapatılır.

Yapılan araştırmalar sırasında öğrenilen SLAM yöntemleri bu çalışmada da denenmiştir. Bu yöntemler; gmapping, karto, hector, cartographer. Tüm bu yöntemler için ayrı ayrı 10 deneme yapılmış ve sonuçlar *Çizelge 4.1. SLAM tekniklerinin kendi arasında karşılaştırılması.*'nda verilmiştir.

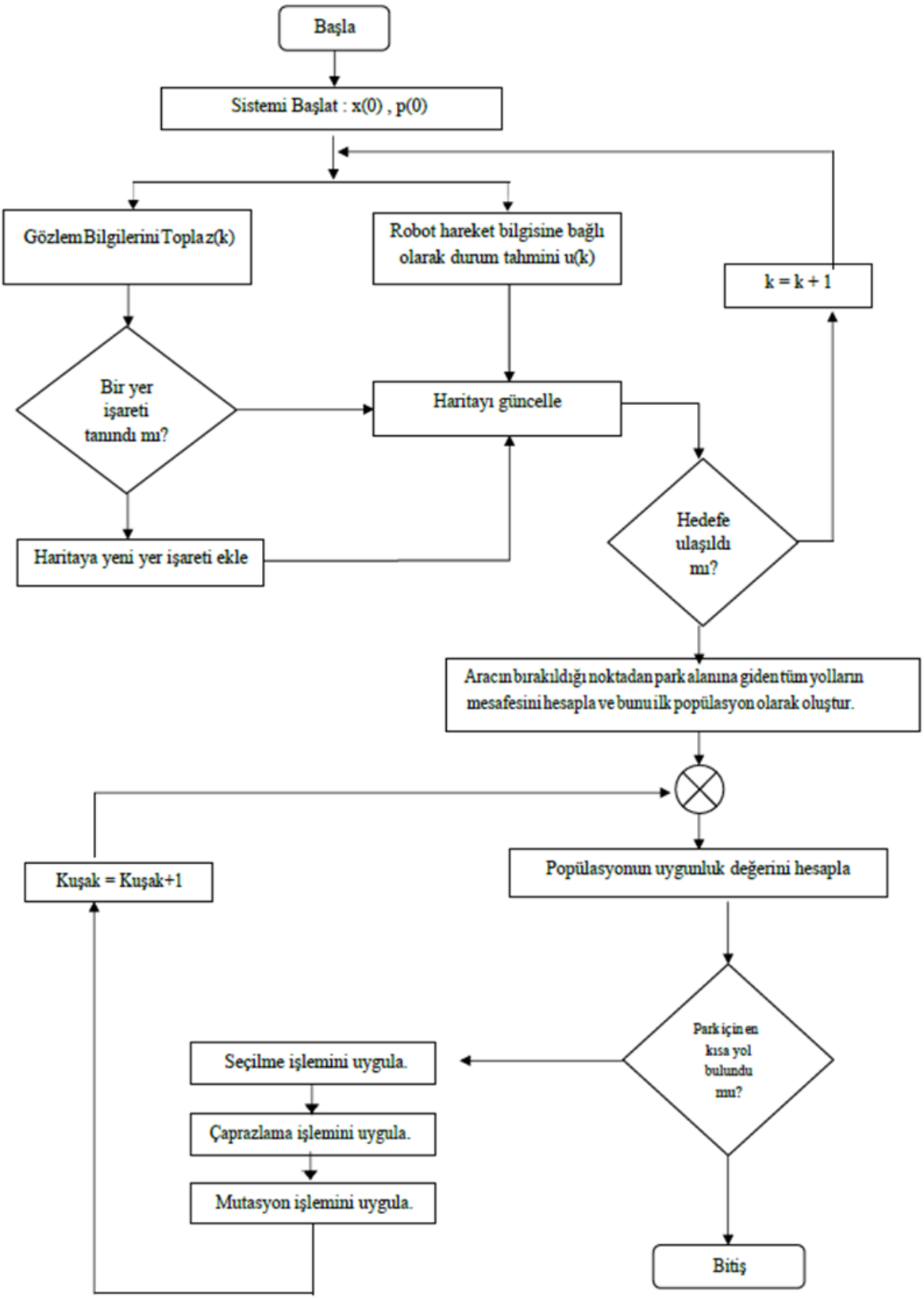
Çizelge 4.1. SLAM tekniklerinin kendi arasında karşılaştırılması.



	karto	gmapping	cartographer	hector
Başarım(%)	92	97	88	90
Süre(s)	33	12	27	23

Şekil 4.1. Tasarım akış *diyagramı*'nda önerilen yöntem 4.adımdan itibaren çalışmaya başlar. Genetik algoritma haritası çıkarılmış ortam üzerinden aracın gitmesi istenen noktaya optimum yol seçimini gerçekleştirmektedir.

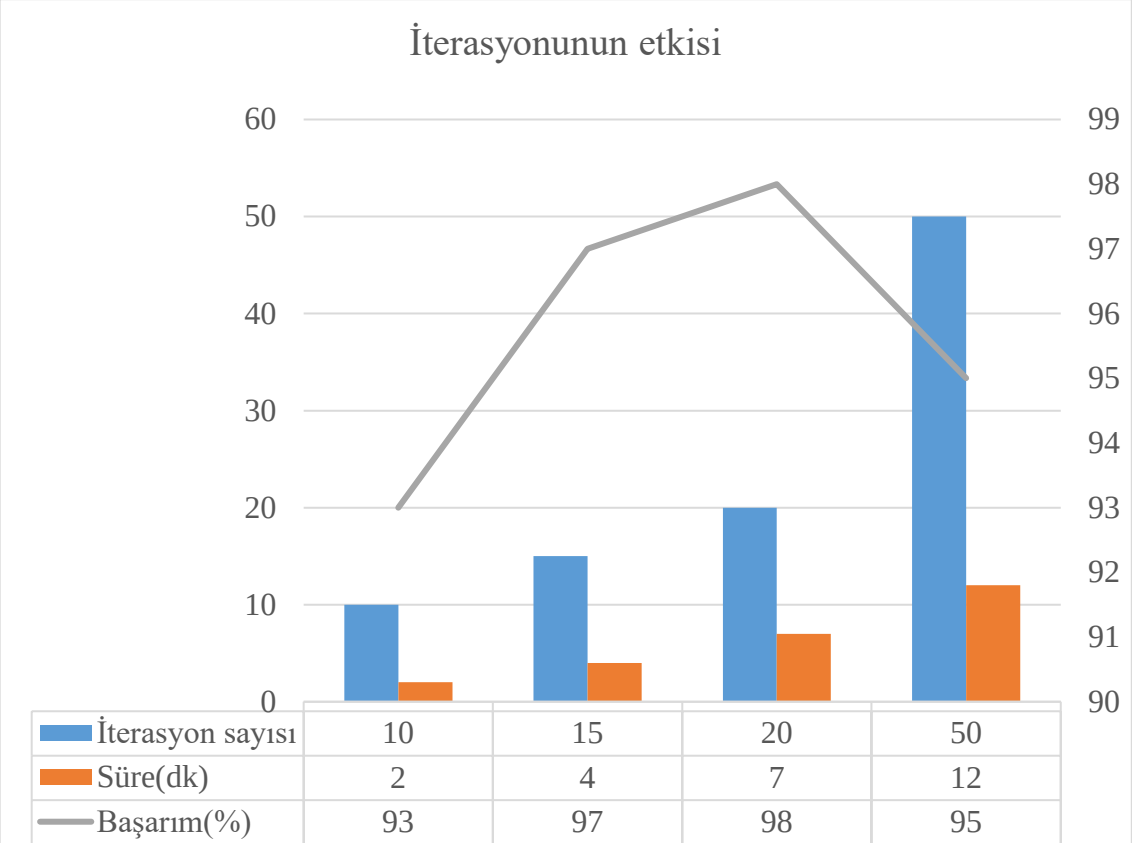
GA, kaydedilen haritayı tarayarak nesne, boşluk, bilinmeyen ve doluluk oranı olarak 4 gruba ayırmaktadır. Daha sonra boş hücreler/pikseller iki kategoride incelenmektedir. Birincisi kategori de pikselin 4 komşusuna, yani sağ, sol, alt ve üst piksellere bakılır. İkinci kategoride ise pikselin 8 komşusuna, yani sağ üst, üst, sol üst, sol, sol alt, alt, sağ alt, sağ piksellere bakılır. Bu komşuluk ilişkisine göre bir yol var mı ve ne kadar mesafede olduğu hesaplanmaktadır ve bu yollar Excel dosyası olarak kaydedilmektedir. Araç haritalama işlemini bitirdikten sonra, kaydedilen haritadan alınan bilgilerle GA'da derlenen bir pikselden diğer piksele en kısa yol bilgisi ile araç 11.adımdan itibaren hareket etmeye başlar. Araç ilk önce kaydedilen harita üzerindeki hangi pikselde olduğunu bulur. Daha sonra, gidilmesi istenen nokta, araç için piksel noktası, arasında ki en kısa mesafeyi Excel dosyasına kaydedilen hesaplamalar yardımıyla bulur.



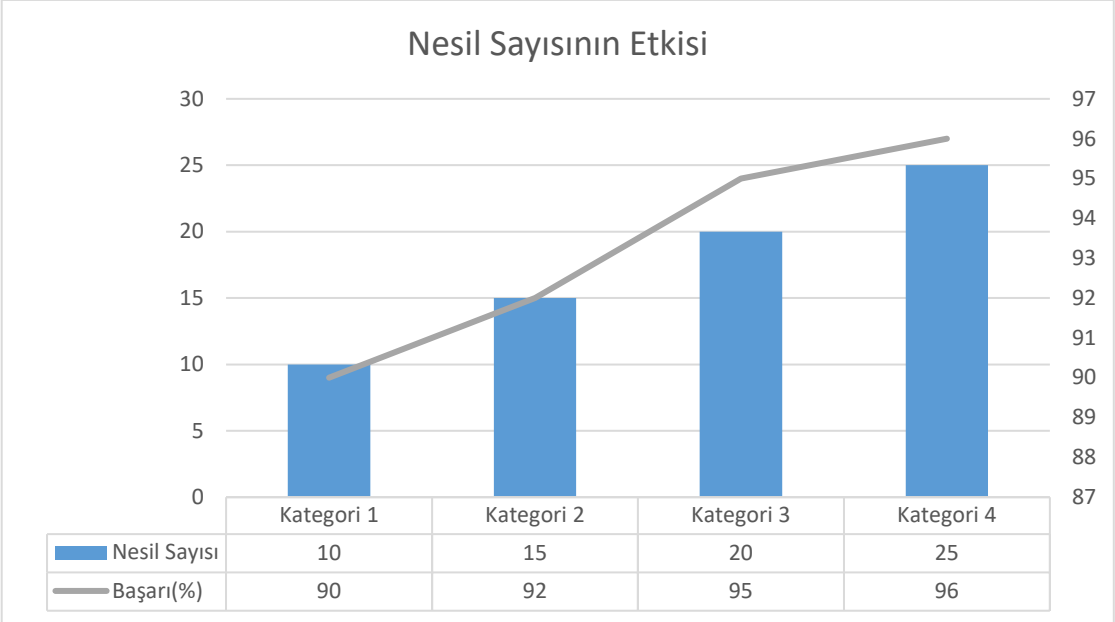
Şekil 4.1. Tasarım akış diyagramı.

Bu çalışmada GA'da kullanılan kıstaslar değiştirilerek sistemi nasıl etkilediği incelenmiştir. GA'nın olasılık kurallarına göre çalıştığı unutulmamalıdır. Bu sebeple GA'nın problemlere ne kadar iyi çözüm vereceği önceden bilinememektedir. GA en iyi çözüme en yakın çözümü sunduğunu da unutmamak gerekmektedir. Kıstaslar değiştirilerek probleme uygun çözüm/ler aranmıştır. *Çizelge 4.2*'ye bakıldığında iterasyon sayısını Genetik Algoritma süresini ciddi oranda etkilediği, ama GA'nın başarısında etkisinin az olduğu gözlemlenmiştir. *Çizelge 4.3*'e bakıldığın nesil sayısının artışı sistemi olumlu yönde etkilediği açıkça görülmektedir. *Çizelge 4.4*'e bakıldığında çaprazlama oranın artırılması sistem optimumluğunu düşürdüğü gözlemlenmiştir. Yapılan denemeler sırasında en yüksek başarımı elde edilen kriterler, *Çizelge 4.5*'sinde verilmiştir.

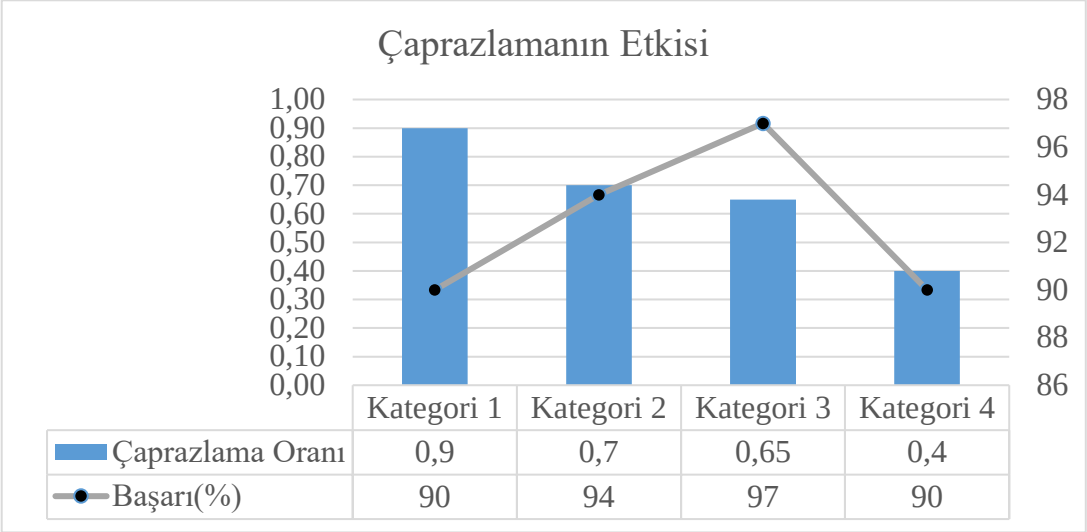
Çizelge 4.2. İterasyon sayısındaki değişimin Genetik Algoritmaya etkisi.



Çizelge 4.3. Nesil Sayındaki değişimin Genetik Algoritmaya Etkisi.



Çizelge 4.4. Çaprazlama sayısındaki değişimin Genetik Algoritmaya etkisi.



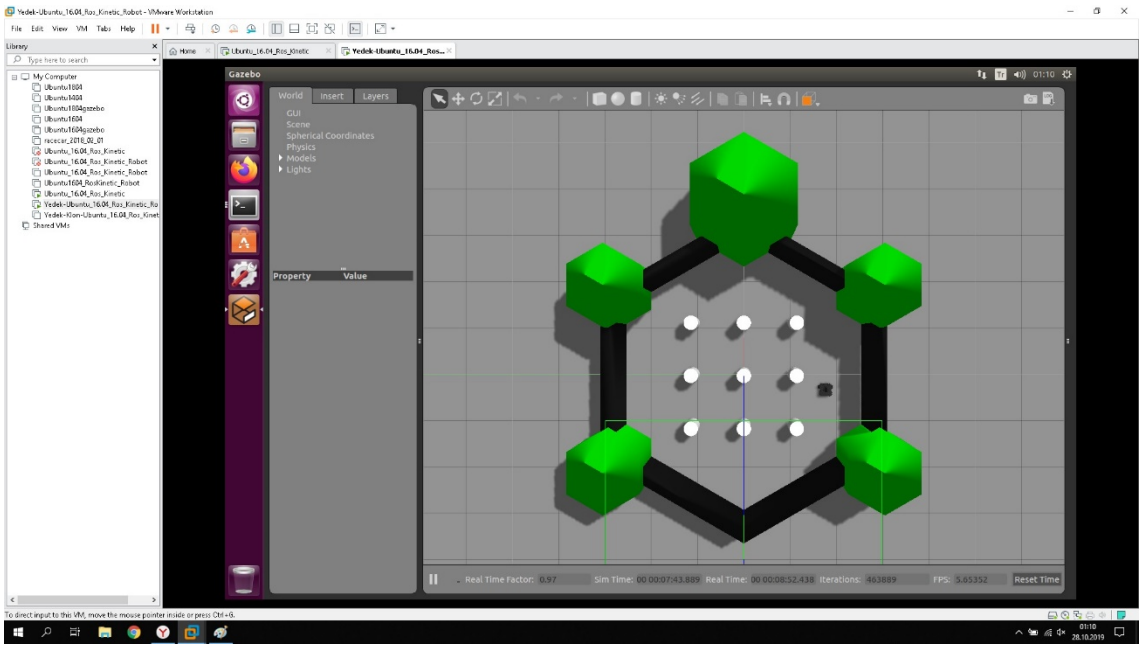
Çizelge 4.5. Genetik Algoritma optimum kriterleri.

Genetik Algoritma Kriterleri	
Nesil sayısı	20
İterasyon sayısı	15
Çaprazlama olasılığı	0.65
Mutasyon olasılığı	0.03

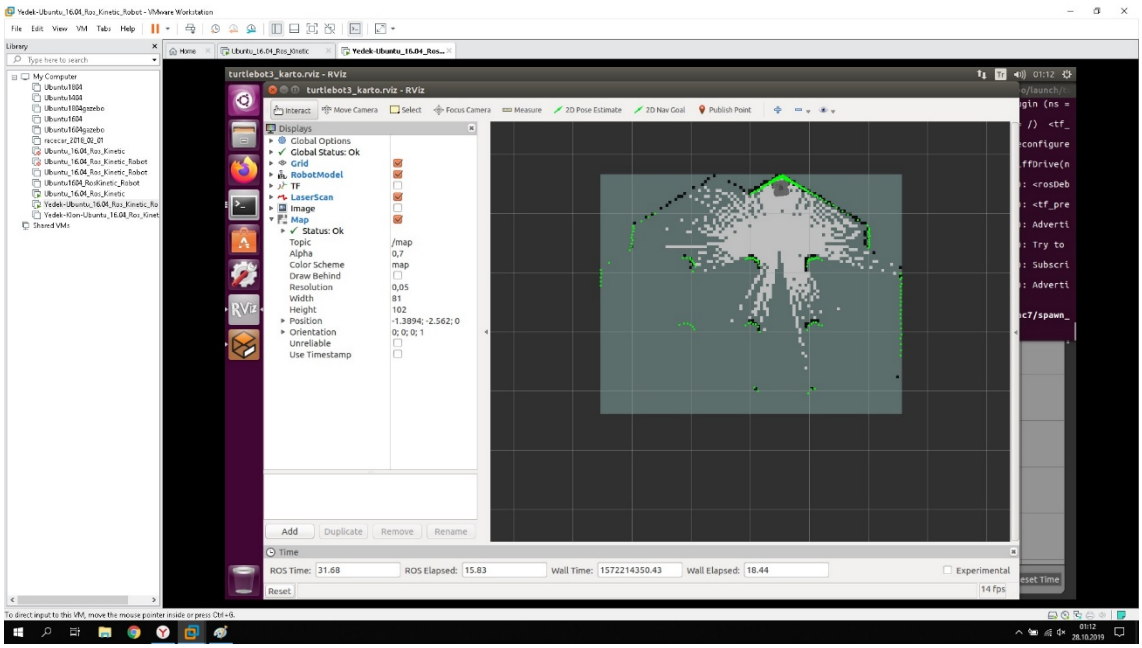
5. SİMÜLASYON SONUÇLARI

Bu tezde otonom araçla genetik algoritma kullanılarak oluşturulan harita üzerinden lokasyon yapabilmek için simülasyon ortamında test yapılmıştır. Otonom araç simülasyonunda Gazebo ara yüzü kullanılarak sanal bir ortam belirlenmiş ve bu ortamın haritası çıkarılmıştır. Rviz; robotun çıkarttığı haritanın ve izlediği yolun görüntülenebileceği ROS paketidir. Rviz kullanılarak dışarıdan aktarılan robotun simülasyondaki görüntüleri ve haritalama özellikleri işlenmiştir.

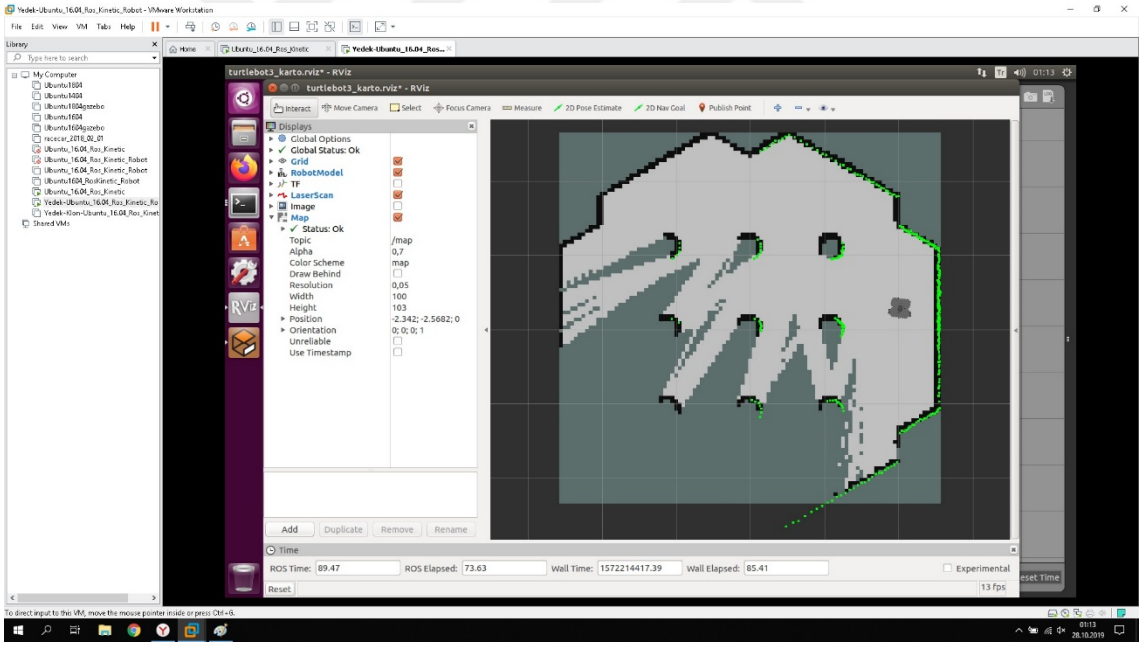
Simülasyon deney ortamında oluşturulan sanal makinelerde Gazebo ortamında sanal dünyalarından biri çağrılmış ve araç bu ortamda otonom hareket ederek Bölüm 3'te SLAM yöntemi kullanılarak bulunduğu ortamın haritasını çıkarmaya başlamıştır. Rviz ortamında bu işlem eş zamanlı olarak izlenebilmektedir. Araç haritalama işlemini bitirdiğinde haritayı kayıt eder.



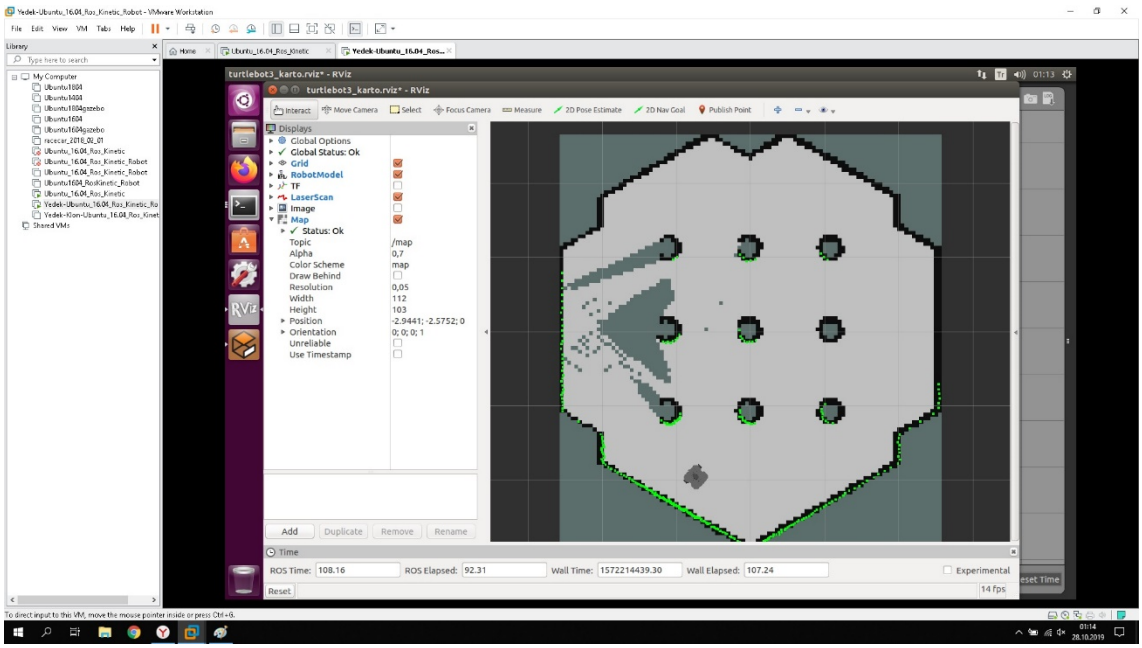
Şekil 5.1. Gazebo ortamında aracın konumu.



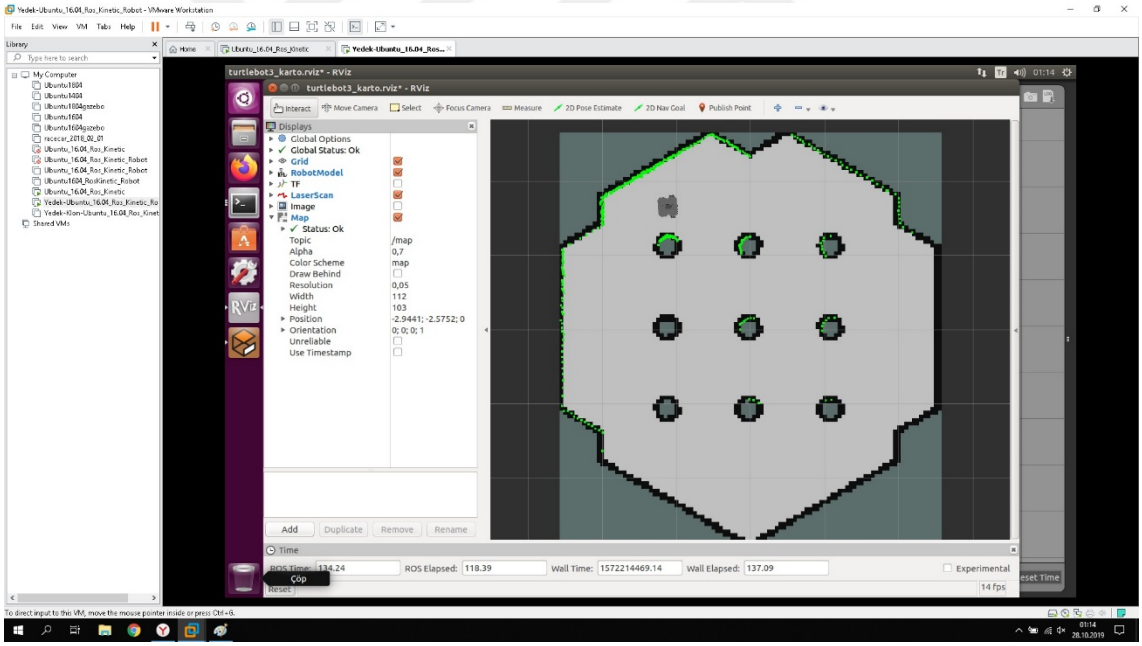
Şekil 5.2. Rviz ortamında aracın ilk konumu ile haritalamaya başlaması.



Şekil 5.3. Rviz ortamında aracın haritalamaya devam etmesi-1.



Şekil 5.4. Rviz ortamında aracın haritalamaya devam etmesi-2.

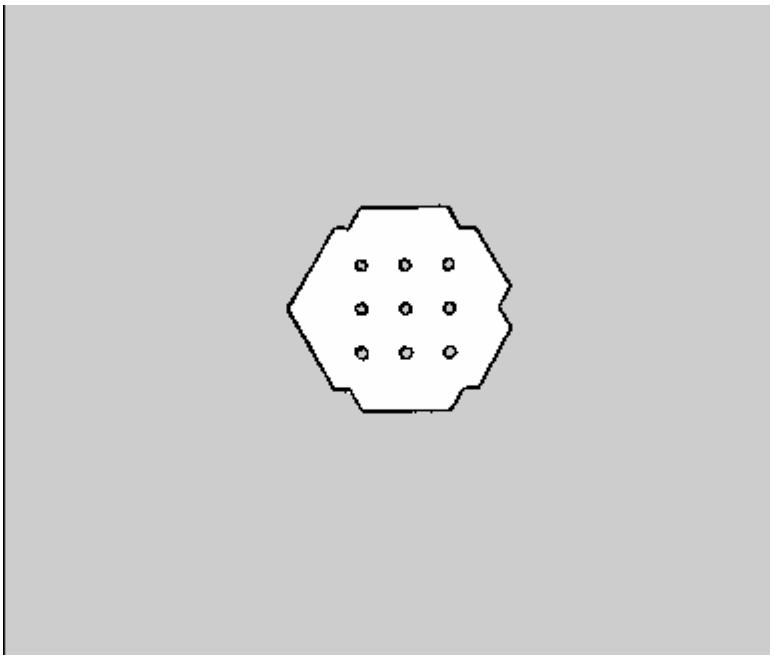


Şekil 5.5. Rviz ortamında aracın ortamın haritalandırmasını bitirmesi.

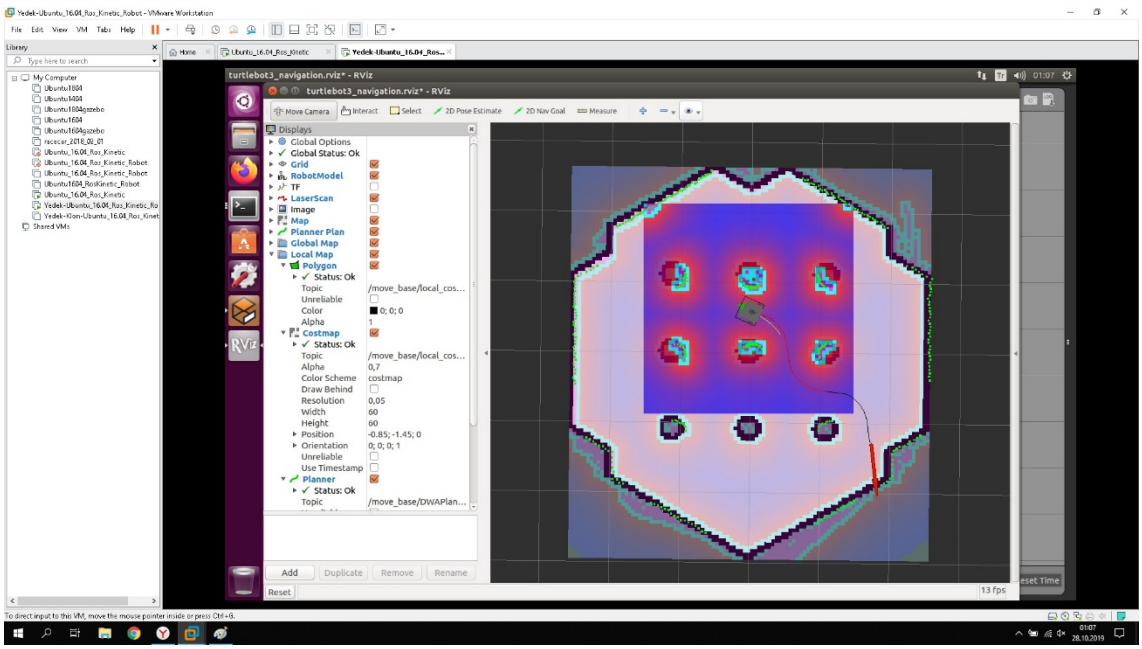
Şekil 5.1'de TurtleBot3 aracının Gazebo ortamındaki ilk konumu görülmektedir. Rviz ortamı ayrı bir terminalden çağrılarak araç hareketi eş zamanlı olarak hem Gazebo ortamında hem de Rviz ortamında gözlenebilmektedir. Şekil 5.2, Şekil 5.3, Şekil 5.4 ve Şekil 5.5'te aracın Gazebo ortamındaki otonom hareketi esnasındaki haritalama işlemleri farklı zamanlar için gösterilmektedir. Şekil 5.5'te Gazebo ortamının haritalamasını bitiren

aracın Rviz ortamındaki görüntüsü verilmiştir.

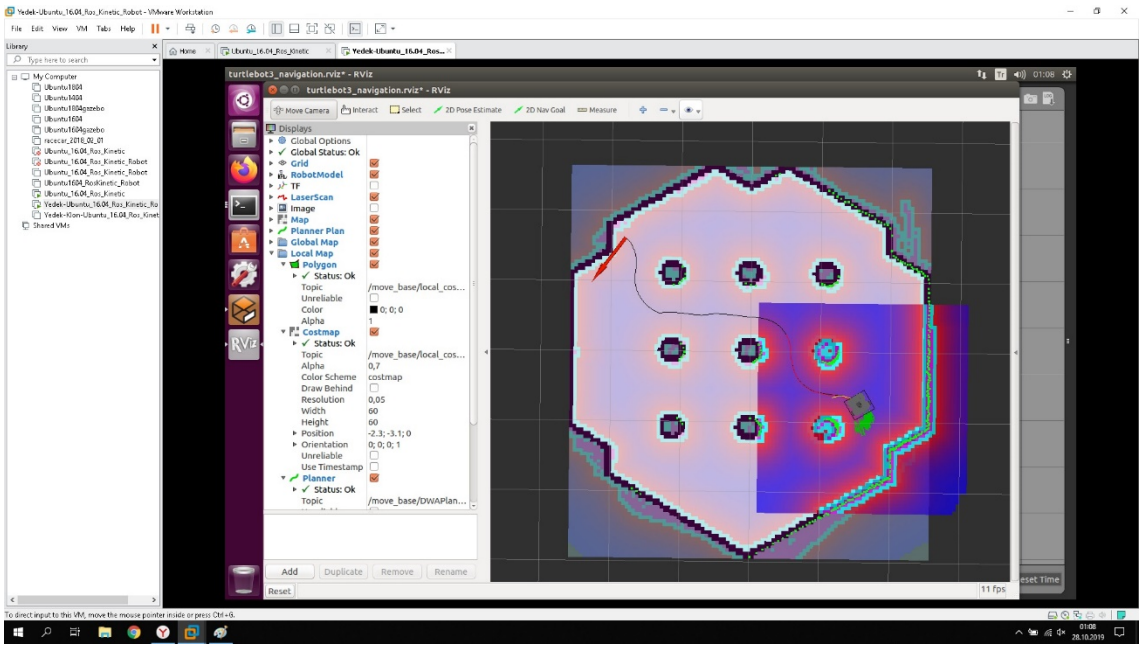
Bu aşamadan sonra elde edilen harita genetik algoritma ile birleştirilerek, sanal ortamda bulunan aracımızın istediğimiz noktaya en kısa yolu bulup belirlenen noktaya gitmesini bekliyoruz.



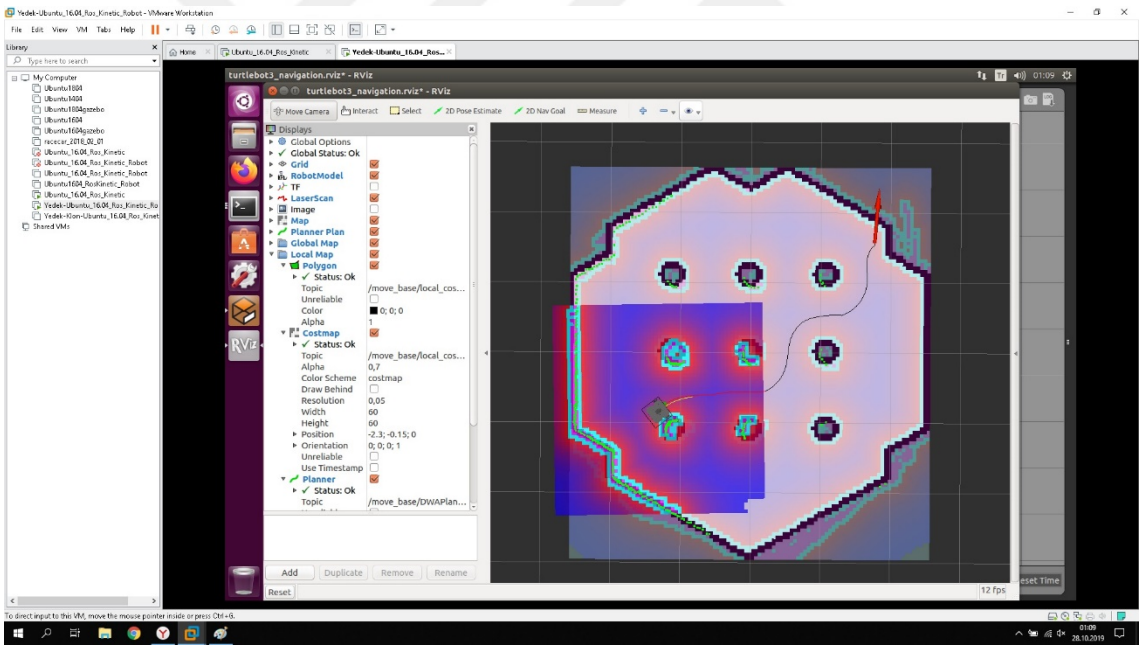
Şekil 5.6. SLAM ile elde edilen harita görüntüsü.



Şekil 5.7. Aracın bulunduğu yerden kırmızı çubukla gösterilen yer arasında bulduğu en kısa yol güzergâhı 1.



Şekil 5.8. Aracın bulunduğu başka bir konumdan kırmızı çubukla gösterilen yer arasında bulduğu en kısa yol güzergâhı 2.



Şekil 5.9. Aracın bulunduğu yine başka bir konumdan kırmızı çubukla gösterilen yer arasında bulduğu en kısa yol güzergâhı 3.

Yeni bir terminal çağrılarak elde edilen harita genetik algoritma kısmında arka planda çağrılır. Çağrılan harita Rviz ortamında Şekil 5.7'da ki gibi görülmektedir. Şekil 5.8 ve Şekil 5.9'de farklı noktalara çağrılan aracın genetik algoritma ile en kısa yolu bulduğu açıkça görülmektedir. Gözlemler neticesinde simülasyon ortamındaki robot yüksek başarımla istenilen konuma gidebildiği sonucuna ulaşılmıştır.

6. SONUÇLAR VE ÖNERİLER

Bu çalışmada amaç SLAM yöntemi ile GA'yı birleştirerek otonom araçlarda optimum yol seçimini gerçekleştirmektir. Optimum yol seçimini yapabilmek için otonom araçta genetik algoritma uygulaması, alan kullanımı ve araç verimliliği açısından önemlidir. Genetik algoritma, önerilen sistemin çok fazla zaman kaybı olmadan en kısa yol güzergâhı için arama işlemini optimize etmesine yardımcı olur. Önerilen yöntemin avantajı, etkili bir en kısa yolu sağlaması ve kontrollü çalışma alanını daha geniş bir bölgeye yaymasıdır.

Sunulan GA tabanlı algoritma etkili ve uygulaması kolaydır. En uygun çözümü bulmak için biyolojik evrimi simüle edilmiştir. Belirlenen alanda aracın otonom hareket ederek yol seçme işlemini gerçekleştirme probleminde GA'da farklı parametreler kullanılarak denemiş ve bu parametrelerin çözüm üzerindeki etkileri incelenmiştir. Nüfus sayısındaki birey sayısının artışı yapılan simülasyonlardaki sonuçların bulunmasına olumlu yönde katkı yaptığı görülmüştür. Simülasyon test ortamındaki denemeler esnasındaki gözlemler neticesinde robot yüksek başarımla istenilen konuma gidebildiği sonucuna ulaşılmıştır.

Bu çalışma geliştirilerek gerçek bir arabada uygulanabilir. Otonom araçların uygun yere park yapma işlemini gerçekleştirmede kullanılabilir. Dolu park yerleri araçla önceden paylaşılarak, araç kendine en yakın mesafede ki park alanına otonom olarak gidebilir. Otonom yarışlarında -park işlemi görevi olan durumlarda- araçların veri toplaması için yarış pistinin kullanımı sırasında lidar ile toplanacak veriler ile haritalandırma yapılarak yarış esnasından park işlemi için yine önerilen yöntem üzerinde gerekli değişiklikler yapılarak kullanılabilir.

7. KAYNAKLAR

[1] H. Yu, X. Li, R. M. Muray, S. Ramesh and C. J. Tomlin, *Safe, Autonomous and Intelligent Vehicles*, Cham, İsviçre, Springer, 2019, ss. 5-33.

[2] C. Little. (2019, 6 Mayıs). The Intelligent Vehicle Initiative: Advancing Human-Centered Smart Vehicle. US Department of Transportation Federal Highway Administration, [Online]. Erişim: http://www.tfhrc.gov/pubrds/pr97- 10/p18.html

[3] B. Yıldız, "Object detection and mapping using lidar for a mobile robot," Yüksek lisans tezi, Mekatronik Mühendisliği, Fen Bilimleri Ensitüsü, Dokuz Eylül Üniversitesi, İzmir, Türkiye, 2016.

[4] M. Fu and Y. Huang, "A survey of traffic sign recognition," *International Conference on Wavelet Analysis and Pattern Recognition-ICWAPR*, Qingdao, Çin, 2010, ss. 119–124.

[5] Road Sign Recognition Survey. (2019, 28 Nisan). [Online] Erişim: http://euler.fd.cvut.cz/research/rs2/files/skoda-rs-survey.html

[6] A. De la Escalera, L. E. Moreno, M. A. Salichs and J. M. Armigol, "Road traffic sign detection and classification," *IEEE Transactions on Industrial Electronics*, c. 44, sayı. 6, ss. 848-859,1997.

[7] R. Janssen, W. Ritter, F. Stein, and S. Ott. "Hybrid approach for traffic sign recognition," *In Proc. of Intelligent Vehicles Conference*, 1993, ss. 390-395.

[8] C.Y. Fang, S. W. Chen and C. S. Fuh, "Road-Sign detection and tracking," *IEEE Transactions on Vehicular Technology*, Eylül, c. 52, sayı. 5, ss. 1329- 1341, 2003.

[9] T.Stahl, A. Wischnewski, J. Bethz and M. Lienkamp, "ROS-based localization of a race vehicle at highspeed using Lidar," *E3S Web of Conferences ICPEME*, 2010, c. 95.

[10] M. Yaktubay, "A genetic algorithm based solution approach for vehicle routing problem," Yüksek lisans tezi, Endüstri Mühendisliği, Fen Bilimleri Enstitüsü, Adana Bilim ve Teknoloji Üniversitesi, Adana, 2018.

[11] K.Paslıoğlu, "Otonom mobil robotlarda dağılımlı kalman filtresi tabanlı eş zamanlı lokalizasyon ve haritalama," Yüksek lisans tezi, Fizik Mühendisliği, Fen Bilimleri Ensitüsü, İstanbul Teknik Üniversitesi, İstanbul, 2010.

[12] M.-J. Jung, H. Myung, S.-G. Hong, D. Park, H.-K. Lee, and S. Bang, "Structured light 2D range finder for simultaneous localization and map-building (SLAM) in home environments," *in Proceedings of the 2004 International Symposium on Micro-Nanomechatronics and Human Science*, 2004 and *The Fourth Symposium Micro-Nanomechatronics for Information-Based Society*, Nagoya, Japonya, 2004, ss. 371–376.

[13] Y. Misono, Y. Goto, Y. Tarutoko, K. Kobayashi, and K. Watanabe, "Development of laser rangefinder-based slam algorithm for mobile robot navigation," *in Proceedings of the 2007 Annual Conference The Society of Insrument and Control Engineers*, Takamatsu, Japonya, 2007, ss. 392–396.

[14] M. Begum, G. K. Mann and R. G. Gosine, "Integrated fuzzy logic and genetic

algorithmic approach for simultaneous localization and mapping of mobile robots," *Applied Soft Computing*, c. 8, sayı. 1, ss. 150 – 165, 2008.

[15] A. Tuncer, "Otonom araçlar için yol bulma probleminin genetik algoritmalar ve fpga ile çözümü ve gerçekleştirilmesi," Doktora tezi, Elektronik ve Bilgisayar Eğitimi, Fen Bilimleri Enstitüsü, Kocaeli Üniversitesi, Kocaeli, 2013.

[16] D. J. Feng, S. Wijesoma and A. P. Shacklock, "Genetic algorithmic filter approach to mobile robot simultaneous localization and mapping," *IEEE 9th International Conference on Control, Automation, Robotics and Vision*, Singapore, 2007.

[17] D. J. Feng and S. Wijesoma, "Improving rao-blackwellised genetic algorithmic filter slam through genetic learning," *IEEE 10th International Conference on Control, Automation, Robotics and Vision*, Hanoi, Vietnam, 2008.

[18] M. Sentinel, "Phantom auto'will tour city. The Milwaukee Sentinel," Google News Archive, ed., 1926, ss. 4.

[19] Navlab, (2019, 28 Eylül). The Carnegie Mellon University Navigation Laboratory. http://www.cs.cmu.edu/afs/cs/project/alv/www/index.html.

[20] T. Kanade, C. Thorpe, and W. Whittaker, "Autonomous land vehicle project at cmu," *In Proceedings of the 1986 14th ACM Annual Conference on Computer Science*, 1986, ss. 71–80.

[21] R. S. Wallace, A.Stentz, C.E. Thorpe, H. P. Moravec, W. Whittaker, and T. Kanade, "First results in robot road-following," *In International Joint Conference on Artificial Intelligence*, 1985, ss. 1089–1095.

[22] J. Schmidhuber, "Prof. Schmidhuber's Highlights of Robot Car History," *Istituto Dalle Molle di Studi sull'Intelligenza Artificiale*, 2011.

[23] M. Novak, (2019, 29 Eylül). The National Automated Highway System That Almost Was, [Online]. Erişim: https://www.smithsonianmag.com/history/the-national-automated-highway-system- that-almost-was-63027245/.

[24] S. Crowe, "Back to the future: Autonomous driving in 1995," *Robotics Business Review*, 2015.

[25] A. Davies, (2019, 30 Ağustos). This is Big: A Robo-Car Just Drove Across The Country, [Online]. Erişim: https://www.wired.com/2015/04/delphi-autonomous-car-cross-country/.

[26] J. Ramsey, (2019, 23 Temmuz). Self-driving cars to be tested on Virginia highways. [Online]. Erişim: http://www.richmond.com/news/article_b1168b67-3b2b-5274-89148a3304f2e417.html.

[27] J. Ramsey, (2019, 25 Temmuz). On the Road – Waymo. [Online]. Erişim: https://waymo.com/ontheroad/ .

[28] *Levels Of Driving Automation Are Defined in New Sae*, International Standard J3016, 2014.

[29] ROS (Robotik işletim sistemi), (2019, 13 Mayıs). [Online]. Erişim: http://wiki.ros.org/.

[30] Gazebo, (2019, 13 Mayıs). Gazebo Simülasyonu. [Online]. Erişim: http://gazebosim.org/tutorials.

[31] C. Fairchild and Dr. T. L. Harman, *ROS Robotics By Example*, 1.baskı, Birmingham, UK, Packt Publishing Ltd., 2016, c. 2, ss. 57-69.

[32] C. Fairchild and Dr. T. L. Harman, *ROS Robotics By Example*, 1.baskı Birmingham, UK, Packt Publishing Ltd, 2016, c. 2, ss.57.

[33] RVIZ, (2019, 25 Eylül). Rviz: a good reason to implement a vision system in ROS [Online] Erişim: https://computervisionblog.wordpress.com/2012/11/18/rviz-a-good-reason-to- implement-a-vision-system-in-ros/.

[34] TurtleBot3, (2019, 12 Eylül). ROBOTIS e-El Kitabı [Online] Erişim: http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#overview.

[35] J. Aulinas, Y. Petillot, J. Salvi and X. Lladó, "The SLAM problem: a survey," *Proc. 11th International Conference of the Catalan Association for Artificial Intelligence (CCIA)*, İspanya, 2008.

[36] H. Durrant-Whyte and T. Bailey "Simultaneous Localization and Mapping: Part I," *IEEE Robotics & Automation Magazine*, c. 13, sayı. 2, ss. 99-110, 2006.

[37] R. Smith and P. Cheesman, "On The Representation of Spatial Uncertainty," *The International Journal of Robotics*, c. 5, sayı. 4, ss. 56–68, 1987.

[38] H.F. Durrant-Whyte, "Uncertain Geometry in Robotics," *IEEE Transactions on Robotics and Automation*, c. 4, sayı. 1, ss. 23–31, 1988.

[39] N. Ayache and O. Faugeras, "Building, registrating, and fusing noisy visual maps," *The International Journal of Robotics Research*, c. 7, sayı. 6, ss. 45–65, 1988.

[40] J. Crowley, "World modeling and position estimation for a mobile robot using ultra- sonic ranging," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1989, ss. 674–681.

[41] R. Chatila and J.P. Laumond, "Position referencing and consistent world modeling for mobile robots," *in Proceedings IEEE International Conference on Robotics and Automation*, 1985, ss. 138–143.

[42] H. Durrant-Whyte, D. Rye, and E. Nebot, "Localisation of automatic guided vehicles," *in Robotics Research: The 7th International Symposium (ISRR'95)*, 1996, ss. 613–625.

[43] M. Csorba, "Simultaneous localisation and map building," Doctorate thesis, University of Oxford, England, 1997.

[44] M. Csorba and H.F. Durrant-Whyte, "A new approach to simultaneous localisation and map building," *Aerospace/Defense Sensing and Controls*, Orlando, Florida, ABD, 1996.

[45] J.J. Leonard and H.J.S. Feder, "A computational efficient method for large-scale concurrent mapping and localisation," *in Robotics Research, The Ninth International Symposium (ISRR'99)*, New York: Springer-Verlag, 2000, ss. 169–176.

[46] J.A. Castellanos, J.M. Martnez, J. Neira, and J.D. Tardós, "Experiments in multisensor mobile robot localization and map building," *3rd IFAC Symposium Intelligent Autonomous Vehicles*, 1998, ss. 173–178.

[47] J.A. Castellanos, J.D. Tardós, G. Schmidt, "Building a global map of the

environment of a mobile robot: the ımportance of correlations," *IEEE International Conference on Robotics and Automation*, 1997, ss. 1053–1059.

[48] J. Guivant, E.M. Nebot, and S. Baiker, "Localization and map building using laser range sensors ın outdoor applications," *The Journal of Intelligent and Robotic Systems*, c.17, sayı. 10, ss. 565–583, 2000.

[49] S.B. Williams, P. Newman, G. Dissanayake and H.F. Durrant-Whyte, "Autonomous underwater simultaneous localisation and map building," *in Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, Kaliforniya, USA, 2000, ss. 1793–1798.

[50] K.S. Chong and L. Kleeman, "Feature-Based mapping in real, large scale environments using an ultrasonic array," *The International Journal of Robotics Research*, c. 18, sayı. 1, ss. 3– 19, 1999.

[51] M. Deans and M. Hebert, "Experimental comparison of techniques for localization and mapping using a bearing-only sensor," *in Proceedings International Symposium Experimental Robotics*, 2000, ss. 395–404.

[52] J. Hollerbach and D. E. Koditscheck, *Robotics Research, The Ninth International Symposium(ISRR'99)*, 1.baskı, New York, ABD: Springer-Verlag, 2000.

[53] S. Thrun, D. Fox, and W. Burgard, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Autonomous Robots*, c.5 sayı.3-4, ss. 253-271, 1998.

[54] G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark and M. Csobra, "A solution to the simultaneous localisation and mapping (slam) problem", *IEEE Transactions on Robotics and Automation*, c. 17, sayı. 3, ss. 229-241, 2001.

[55] S. Thrun et.all, "Robotic mapping: a survey," Exploring Artificial Intelligence in the New Millenium. The Morgan Kaufmann Series in Artificial Inteligence, 1.baskı. San Francisco, Kaliforniya, ABD: Morgan Kaufmann Publishers, 2003, böl.1, ss. 1-35.

[56] A.J. Davison and D. Murray, "Simultaneous localization and map-building using active vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, c.24, sayı.7 ss.865–880, 2002.

[57] P.M. Newman and J.J. Leonard, "Consistent convergent, and constant-time slam," *International Joint Conference on Artificial Intelligence (IJCAI)*, Meksika, 2000, ss. 1143-U1150.

[58] P. Jensfelt, D. Kragic, J. Folkesson, M. Björkman, "A framework for vision based bearing only 3d slam", *Proceedings IEEE International Conference on Robotics and Automation, ICRA*, 2006, ss. 1944–1950.

[59] S. Se, D. Lowe and J. Little, "Mobile *Robot Localization and Mapping with Uncertainty Using Scaleinvariant Visual Landmarks*," *The International Journal of Robotics Research*, c.21, sayı.8, ss.735–758, 2002.

[60] S. Thrun and Y. Liu, "Multi-Robot slam with sparse extended ınformation filers," *11th International Symposium of Robotics Research (ISRR'03)*, Sienna, Italya, 2003.

[61] S. Thrun, C. Martin, Y. Liu, D. Hähnel, R. Emery-Montemerlo, D. Chakrabarti and W. Burgard, "A real-time expectation maximization algorithm for acquiring multi- planar maps of ındoor environments with mobile robots," *IEEE*

Transactions on Robotics and Automation, c.20, sayı.3, ss.433–442, 2004.

[62] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters", *International Journal of Robotics Research*, 2004, c.23, sayı.7-8, ss.693–716.

[63] E. Wan and R. van der Merwe, "The Unscented Kalman Filter," Kalman filtering and neural networks, 3. baskı, New York, ABD: Wiley-Interscience Publication, 2001, böl. 7, ss. 221-277.

[64] J.E. Guivant and E.M, "Nebot optimization of the simultaneous localization and map- building algorithm for real-time ımplementation," *IEEE Transactions on Robotics and Automation*, c.17, sayı.3, ss. 242-257, 2001.

[65] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM: a factored solution to the simultaneous localization and mapping problem," *Proceedings of the National Conference on Artificial Intelligence*, 2002, ss. 593–598.

[66] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges", *18th International Joint Conference on Artificial Intelligence (IJCAI)*, Meksika, 2003, ss. 1151–1156.

[67] M. Montemerlo and S. Thrun, "FastSLAM 1.0", FastSLAM: a scalabale method for the simultaneous localizaton and mapping problem in robotics, Springer Tracts in Advanced Robotics c.27, Berlin, Almanya: Springer, 2007, böl.3, ss. 27-62.

[68] W. Burgard, D. Fox, H. Jans, C. Matenar and S. Thrun, "Sonar-based mapping with mobile robots using EM," *16th International Conference on Machine Learning*, 1999.

[69] Z. Chen, J. Samarabandu and R. Rodrigo, "Recent advances in simultaneous localization and mapbuilding using computer vision," *Advanced Robotics*, c.21, sayı.3-4, ss.233–265, 2007.

[70] S. Thrun, "A probabilistic online mapping algorithm for teams of mobile robots", *International Journal of Robotics Research*, c.20, sayı.5, ss.335–363, 2001.

[71] S.Solak, "Gezgin Robotların Konom Belirleme ve Engel Sakınım Probleminin Tek Kartlı Bilgisayar Sistemi Kullanılarak Çözümü," Doktora tezi, Elektronik ve Bilgisayar Eğitimi, Fen Bilimleri Ensitüsü, Kocaeli Üniversitesi, Kocaeli, 2016.

[72] J. H. Holland, "Illustrations," Adaptation in Natural Artificial Systems, 1. baskı, Massachusetts, Londra, İngiltere: MIT Press, 1992, böl.3. ss. 32-66.

[73] Ö.Aksoy, "Askeri hava taşıma probleminde risk yönelimli eşzamanlı rotalama–çizelgeleme ve çözüm önerileri," Doktora tezi, Endüstri Mühendisliği, Fen Bilimleri Ensitüsü, Eskişehir Osmangazi Üniversitesi, Eskişehir,2018.

[74] E. Alba and B. Dorronsoro, "Introduction to cellular genetic algorithms,"Cellular Genetic Algorithm, Malaga, İspanya: Springer, 2008, böl.1, pp. 3-10.

[75] M. A Berry, G. L. Lınoff, "Automatic cluster detection," Data Mining Techniques For Marketing, Sales And Customer Support, 2. Baskı, New York, ABD: John Wiliey Publishing, 1997,böl.11, ss.356-358.

[76] ROS, (2019, 15 Mayıs). Ubuntu install of ROS Kinetic, [Online]. Erişim: http://wiki.ros.org/kinetic/Installation/Ubuntu .

[77] TurtleBot3, (2019, 15 Mayıs). TurtleBot3 PC Setup, [Online]. Erişim: http://emanual.robotis.com/docs/en/platform/turtlebot3/pc_setup/.

8. EKLER

8.1. EK 1: BAZI ROS KOMUTLARI

Çizelge 8.1. Bazı ROS Bilgi Komutları ve Açıklamaları.

Komut	Açıklama
rostopic	ROS Topic Bilgileri
rosnode	ROS Node Bilgileri
rosbag	ROS Mesajlarını kayıt ve tekrar oynatma
rospack	ROS Paket yönetimi, paket bilgilerini gösterme gibi
rostopic list	Aktif olan Topicleri listeler
rostopic echo [TOPIC_NAME]	Gerçek zamanlı olarak belirtilen topicteki mesaj içeriği gösterir.
rostopic find [TYPE_NAME]	Belirli mesaj tipinde veri ileten topici bul
rostopic type [TOPIC_NAME]	Belirtilen topicin mesaj tipini göster
rostopic bw [TOPIC_NAME]	Show the message data bandwidth of a specific topic
rostopic info [TOPIC_NAME]	Belirtilen topic hakkında bilgileri göster
rostopic list	Aktif olan Topicleri listeler

Çizelge 8.2. Bazı alt rosnode ve rosbag komutları ve açıklamaları.

Komut	Açıklama
rosnode list	Altif Node'ları listeler
rosnode ping [NODE_NAME]	Belirtilen Node'a ait bağlantı testi
rosnode info [NODE_NAME]	Belirtilen Node hakkında bilgileri gösterir
rosnode machine [PC_NAME OR IP]	Belirtilen makinede çalışan nodeları listeler
rosnode kill [NODE_NAME]	Belirtilen nodeu kapatır
rosnode cleanup	Bağlantı bilgilerinin kontrol edilemediği hayalet nodeların kayıtlı bilgilerini silin
rosbag record [OPTION] [TOPIC_NAME]	Belirtilen topice ait mesajları bsg dosyasına kaydet
rosbag info [FILE_NAME]	bag file hakkında bilgileri göster
rosbag play [FILE_NAME]	Belirtilen bag dosyasını oynat
rosbag compress [FILE_NAME]	Belirtilen bag dosyasını sıkıştır
rosbag decompress [FILE_NAME]	Belirtilen bag dosyasını aç
rosbag filter [INPUT_FILE] [OUTPUT_FILE] [OPTION]	Belirtilen mesajları filtreleyerek yeni bir bag dosyası oluştur
rosbag reindex bag [FILE_NAME]	Reindex
rosbag check bag [FILE_NAME]	Belirtilen bag dosyasının sistemde oynatılabileceği tespit et
rosbag fix [INPUT_FILE] [OUTPUT_FILE] [OPTION]	Uyumsuz versiyondaki bag dosyalarını düzelt

Çizelge 8.2. (Devam) Bazı alt rosnode ve rosbag komutları ve açıklamaları.

rosbag record [OPTION] [TOPIC_NAME]	Belirtilen topice ait mesajları bsg dosyasına kaydet
rosbag info [FILE_NAME]	bag file hakkında bilgileri göster
rosbag play [FILE_NAME]	Belirtilen bag dosyasını oynat

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Merve Nur DEMİR

Doğum Tarihi ve Yeri : Tokat / 1993

Yabancı Dili : İngilizce

E-posta : demirmnur@gmail.com

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Y. Lisans	Elektrik Elektronik Müh.	Düzce Üniversitesi	2019
Lisans	Elektrik Elektronik Müh.	Bülent Ecevit Üniversitesi	2016
Lise		Tokat Anadolu İmam Hatip Lisesi	2011

YAYINLAR

M.N. Demir ve Y. Altun, “Otonom Araçla Genetik Algoritma Kullanılarak Haritalama ve Lokasyon”, Düzce Üniversitesi Bilim ve Teknoloji Dergisi, Basımda.