BİR İNSANSIZ HAVA ARACI İÇİN DİNAMİK ORTAMLARDA YOL PLANLAMASI

Demet CANPOLAT TOSUN

Doktora Tezi

Havacılık Elektrik ve Elektroniği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Yasemin IŞIK

Eskişehir

Eskişehir Teknik Üniversitesi

Lisansüstü Eğitim Enstitüsü

Haziran 2021

ÖZET

BİR İNSANSIZ HAVA ARACI İÇİN DİNAMİK ORTAMLARDA YOL PLANLAMASI

Demet CANPOLAT TOSUN

Havacılık Elektrik ve Elektroniği Anabilim Dalı

Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Haziran 2021

Danışman: Dr. Öğr. Üyesi Yasemin IŞIK

İnsansız Hava Araçları (İHA) birçok otonom uygulamayı gerçekleştirecek geniş bir potansiyele sahiptir. Otonom uygulamalar düşünüldüğünde, hava aracının bir noktadan başka bir noktaya giderken en temel gerekliliklerinden biri kesinlikle güvenilir bir engelden sakınma mekanizmasıdır.

Bu tez çalışmasında, dört motorlu bir insansız hava aracı modeli için, bir başlangıç noktasından hedef noktasına, statik ve dinamik engellerin bulunduğu bir ortamda yol planı yapılmıştır. Engel tespitinde lazer mesafe algılayıcı kullanılmıştır. Engelden sakınma ve yol planı oluştururken, hava aracının hem hedefe en kısa yoldan hem de statik ve dinamik engellerden sakınarak ulaşmasını sağlamak amacı ile, ortam bilgisinin mevcut olduğu global bir yol planlama yöntemi olan A* ve sadece sensör verisine göre kısıtlı bir bölgede çalışabilen VFH+ algoritması birlikte kullanılmıştır.

Tez kapsamında incelenen problem için, donanım ve yazılım araçlarının bir araya getirilmesinde kolaylık sağlayan Robot İşletim Sistemi (Robot Operating System-ROS) kullanılmıştır. Algoritmanın testleri, ROS ile entegre çalışabilen Gazebo simülatöründe gerçekleştirilmiştir. MATLAB'te yazılan algoritma kodları ROS Toolbox vasıtası ile Gazebo ortamındaki hava aracında test edilmiştir.

Farklı statik ve dinamik ortamlarda algoritmanın işleyişi değerlendirilmiştir. ROS platformunda yer alan, yer robotlarının navigasyonu için oluşturulmuş algoritmalar (Navigasyon Yığınları) ile kıyaslandığında, önerilen yöntemin yol planının uygulama hızı açısından başarımının yüksek olduğu görülmüştür.

Anahtar Sözcükler: İHA, Dinamik ortam, Yol Planlama, ROS, Gazebo.

ABSTRACT

PATH PLANNING FOR AN UNMANNED AERIAL VEHICLE IN DYNAMIC ENVIRONMENTS

Demet CANPOLAT TOSUN

Department of Avionics

Eskişehir Technical University, Institute of Graduate Programs, June 2021

Supervisor: Asst. Prof. Dr. Yasemin IŞIK

Unmanned Aerial Vehicles (UAVs) have wide potential to carry out many autonomous applications. Considering autonomous applications, one of the most basic requirements of an aerial vehicle when travelling from one point to another is absolutely a reliable obstacle avoidance mechanism.

In this thesis, a path plan is designed for a quadrotor model, from a starting point to a target point in an environment with static and dynamic obstacles. A laser distance sensor is used to detect obstacles. While building a path plan with obstacle avoidance, A* and VFH+ are used together to ensure that the aerial vehicle reaches the target both by the shortest path and by avoiding static and dynamic obstacles. A* is a global path planning method where environmental information is available, while the VFH+ algorithm can only work in a local area according to sensor data.

The Robot Operating System (ROS), which provides convenience in combining hardware and software tools, is used for the problem examined within the scope of the thesis. The tests of the algorithm are carried out in Gazebo simulator, which can work integrated with ROS. Algorithm codes written in Matlab are tested in the aerial vehicle in Gazebo environment using ROS Toolbox.

The operation of the algorithm in different static and dynamic environments is evaluated. Compared to the algorithms built for the navigation of mobile robots (Navigation Stacks) on the ROS platform, the proposed method is successful in terms of the speed of implementation of the path plan.

Keywords: UAV, Dynamic environment, Path planning, ROS, Gazebo.

TEŞEKKÜR

Tez çalışmam süresince, başta danışmanım Dr. Öğr. Üyesi Yasemin Işık olmak üzere, desteğini esirgemeyen Dr. Öğr. Üyesi Hakan Korul'a ve tezin başlangıcından bitişine kadar her zaman yanımızda olan, yönlendiren değerli hocamız Prof. Dr. Osman Parlaktuna'ya teşekkürlerimi sunarım.

Tez jürimde yer alarak, değerli fikir, görüş ve önerilerde bulunan Dr. Öğr. Üyesi Sinem Kahvecioğlu, Doç. Dr. Işıl Yazar ve Dr. Öğr. Üyesi Gökhan Dındış hocalarıma teşekkür ederim.

Bu süreçte yanlarında tüm sıkıntılarımı unuttuğum sabrı, sevgisi ve anlayışıyla beni her daim umutlandıran, destek olan eşime ve oğluma, bugünlere gelmemde çok büyük emek ve fedakârlık göstermiş olan aileme sonsuz teşekkür ederim.

Tezimi ilham veren çalışmaların başlangıcı olması dileğiyle, zorlu geçen pandemi sürecinde kreşe göndermek zorunda kaldığım canım oğlum Rüzgar'ıma ithaf ediyorum.

Demet CANPOLAT TOSUN

28/06/2021

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarında bilimsel etik ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; bu çalışmanın Eskişehir Teknik Üniversitesi tarafından kullanılan "bilimsel intihal tespit programı"yla tarandığını ve hiçbir şekilde "intihal içermediğini" beyan ederim. Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçları kabul ettiğimi bildiririm.

Demet CANPOLAT TOSUN

İÇİNDEKİLER

Sayfa

TABLOLAR DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

q	:	Konum Vektörü
α	:	Sektör Açısı
$c^*_{i,j}$	:	(i, j) Hücresinin Kesinlik Değeri
$m_{i,j}$	:	Aktif Hücre Büyüklüğü
h_k	:	Polar Engel Yoğunluğu
θ_{steer}	:	Robot Yönlendirme Açısı
s_{max}	:	Maksimum Sektör Sayısı
μ_1, μ_2, μ_3	:	Maliyet Fonksiyonu Ağırlıklandırmaları
f	:	Toplam Maliyet Fonksiyonu
g	:	Gerçek Maliyet Fonksiyonu
h	:	Sezgisel Maliyet Fonksiyonu
F^B, M^B	:	Gövde Eksenine göre Toplam Kuvvet ve Momentler
F_i	:	Herbir Motorun İtkisi
p, q, r	:	Yalpa, Yunuslama, Sapma Açısal Hızları
$R_{1,2,3}$	:	Rotasyonel Dönüşüm Matrisleri
$T_{B/R}, T_{R/B}$	:	Koordinat Dönüşüm Matrisleri
u, v, w	:	Üç Eksendeki (x, y, z) Hızlar
V^B, w^B	:	Gövde Eksenine göre Öteleme ve Dönme Hızları
φ, θ, ψ	:	Yalpa, Yunuslama, Sapma Açıları
L, M, N	:	Yalpa, Yunuslama, Sapma Momentleri
ACO	:	Ant Colony Optimization (Arı Kolonisi Optimizasyonu)
BFS	:	Breadth First Search (Genişlik Öncelikli Arama)
DOF	:	Degree of Freedom (Serbestlik Derecesi)
DWA	:	Dynamic Window Approach (Dinamik Pencere Yaklaşımı)
FCU	:	Flight Control Unit (Uçuş Kontrol Birimi)
İHA	:	İnsansız Hava Aracı
PRM	:	Probabilistic Road Map (Olasılıksal Yol Haritası)
ROS	:	Robot Operating System (Robot İşletim Sistemi)
RRTs	:	Rapidly-exploring Random Trees (Rastgele Ağaçlar Yöntemi)

SITL	:	Software in the Loop (Döngüde Yazılım Simülatörü)
VFF	:	Virtual Force Field (Sanal Kuvvet Alanı)
VFH	:	Vector Field Histogram (Vektör Alan Histogramı)

1. GİRİŞ

Yol planlama, literatürde araştırmacılar tarafından oldukça ilgi gören bir problem olmuştur. Bu problem, özellikle mobil robotlarda farklı yaklaşımlarla ele alınmıştır. Hava araçlarında ise gerek araç dinamiği gerekse hava aracının hareket kabiliyetleri göz önünde bulundurulduğunda, daha karmaşık bir problem olarak karşımıza çıkmaktadır.

Yol planlama, kapsamı gereği, aracın ortamda engellere çarpmaksızın yolunu bulabilmesi ile ilgilenir. Problem, bilinen bir ortamda hedefe giden en kısa yolu bulma olabilir. Bilinmeyen bir ortamda yol planlama veya bilinen bir ortamda, ortama sonradan eklenen engeller veya hareketli engellerle yol planlama biçiminde olabilir. Tüm bu durumların üstesinden gelebilecek bir algoritma hem sağlıklı sonuçlar vermeli hem de hızlı çalışmalıdır.

1.1. Amaç ve Katkı

Bu tezin amacı dört motorlu bir insansız hava aracı için dinamik ortamlarda engelsiz bir yol planı oluşturmaktır. Literatürde yer alan çalışmalarda, yol planlama algoritmalarına sadece global yöntemler, sadece lokal yöntemler kullanılarak sıklıkla yer verilmiştir. Bu algoritmalar incelendiğinde dinamik bir ortamda en kısa yoldan hedefe ulaşan yol planı yapabilmek için global bir yöntemin yapısına müdahale edilmesi, örneğin A* algoritmasında oluşturulan sabit maliyet fonksiyonları yerine engelin durumuna göre değişken maliyet fonksiyonları kullanılması veya global ve lokal yöntemleri bir arada kullanarak yapılması gerekmektedir. Global yöntemin yapısına müdahale çoğu zaman işlem yükünü arttırmaktadır. Bu çalışmada A* ile VFH+'nın birlikte kullanılmasının sebebi, A* algoritmasının çok yaygın kullanılan temel bir global algoritma olması, VFH+'nın ise hava aracının gerçek zamanlı uygulamalarda hız ve engele tepki açısından kullanımının uygun olmasıdır. GPS'ten gelebilecek düşük hassasiyetli konum verisinin algoritmaya entegre edilebilmesi mümkündür.

Bu çalışmada, literatürde yer alan hibrit yöntemlerden farklı olarak, dört motorlu insansız hava aracı için A* algoritması tabanlı, lokal algoritma (Vektör Alan Histogramı) destekli bir hibrit algoritma oluşturulmuştur. ROS ve Gazebo platformu kullanılarak bilinen ve bilinmeyen ortamlarda hava aracının hedefine güvenli ve en kısa yoldan ulaşması amaçlanmıştır.

1.2. İçerik

Tezin ikinci bölümünde, yol planlamada kullanılan teknikler kapsamındaki algoritmalar ile ilgili teorik bilgiler verilmiştir.

Tezin üçüncü bölümünde, yol planlama algoritmaları ile ilgili yapılan kapsamlı bir literatür araştırmasının özetine yer verilmiştir.

Tezin dördüncü bölümünde, tez kapsamında hava aracı için kullanılan yol planlama tekniğinden ve algoritmanın işleyişinden bahsedilmiştir.

Tezin beşinci bölümünde, yine tez kapsamında kullanılan hava aracının dinamikleri ve yazılım araçları (ROS, Gazebo, ArduPilot) tanıtılmış ve sistem şeması verilmiştir.

Tezin altıncı bölümünde, yol planlama problemi öncesinde hava aracının Gazebo ortamında otomatik kalkış inişi, belirli hedeflere gidişi, engel tespiti ve potansiyel alanlar yaklaşımı kullanılarak engelden sakınması test edilmiştir.

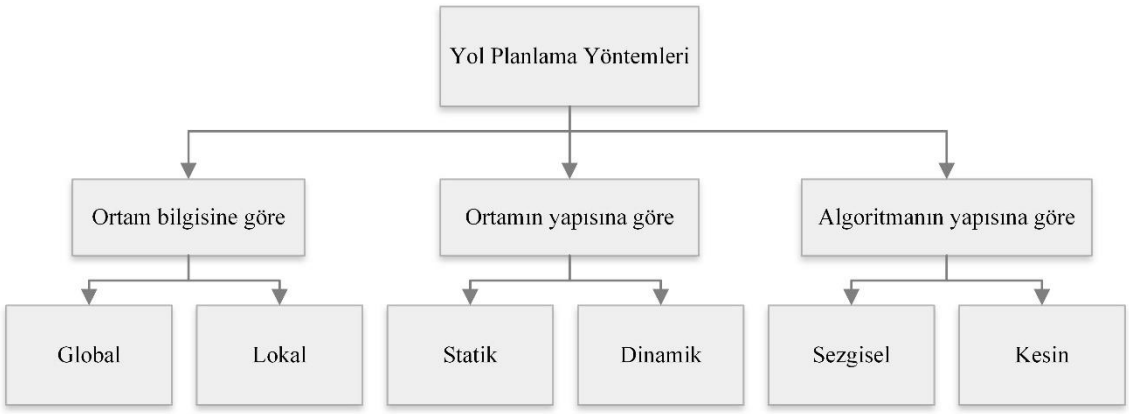
Tezin yedinci bölümünde sadece Vektör Alan Histogramı algoritmasının ve bu algoritmanın A* algoritması ile birlikte farklı test ortamlarındaki sonuçlarına yer verilmiştir.

Son bölüm olan sekizinci bölümde ise sonuçların genel değerlendirilmesi yapılarak mevcut literatür ile bulgular arasındaki ilişkiler açıklanmıştır.

2. YOL PLANLAMA YÖNTEMLERİ

Bu bölümde yol planlama problemi, karşılaşılan farklı problemlere dayanarak sınıflandırılacaktır. Sınıflandırma sonrasında, her bir kategorinin içerdiği algoritmalar hakkında teorik bilgiler verilecektir.

Yol planlama problemi, robotun çalıştığı ortamın bilgisinin varlığına, ortamın yapısına ve algoritmanın yapısına bağlı olarak temelde üç farklı sınıfa ayrılabilir. Bu sınıflar Şekil 2.1'de gösterilmiştir.



Şekil 2.1. *Yol planlama yöntemlerinin sınıflandırılması (Koubaa A. , ve diğerleri, 2018)*

Ortam bilgisinin varlığı, robotun başlangıç konumu, hedef konumu ve bunlar arasındaki bağlantının bilinmesidir. Bu bilgi, yol planlama algoritmasının tasarımında önemli rol oynar. Robotun ortam hakkında bilgisine göre yol planlama algoritmaları, iki başlıkta incelenir. Bunlardan ilki, robotun ortamı bildiği, yani çalışılan ortamın haritasının mevcut olduğu durumdur. Bu türden yol planlamaya 'Global Yol Planlama' adı verilir. Diğeri ise, robotun ortam hakkında hiçbir ön bilgiye sahip olmadığı durumdur. Robot sahip olduğu sensörler yardımı ile çevreyi algılayıp anlık plan yapmaktadır. Bu türden yol planlamaya ise 'Lokal Yol Planlama' adı verilir.

Ortamın yapısına göre planlama ise statik ve dinamik ortam olmak üzere iki ayrı sınıfta incelenebilir. Statik ortam, başlangıç, hedef ve engel konumlarının zamanla değişmediği ortamdır. Dinamik ortamda ise engel konumları robotun hareket ettiği süre içerisinde değişebilir. Dinamik ortamda yol planlama ortamdaki belirsizlikler sebebiyle statik ortama göre daha karmaşık bir problemdir. Dolayısıyla statik ortamlara uygun yol planlama yaklaşımları dinamik ortamlar için uygun değildir.

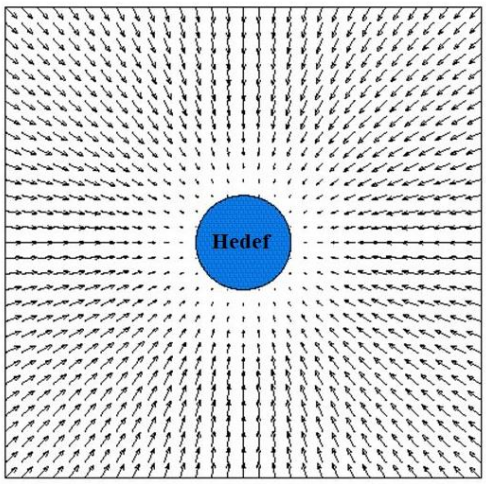
Yol planlama algoritmaları, algoritmanın yapısına göre ise sezgisel (heuristic) ve kesin (exact) olmak üzere ikiye ayırılır. Kesin algoritmalar, varsa en uygun (optimal) çözümü bulur, yoksa çözümün olmadığını gösterir. Sezgisel algoritmalar çözüme daha kısa sürede ulaşır ancak çözümün optimum olmasını garanti etmez.

Bu çalışmada kullanılan hibrit yöntem global ve lokal yöntemin dinamik ortamda birlikte kullanıldığı A*'dan dolayı sezgisel bir yöntemdir. Yol planlama algoritmalarının sınıflandırılması bu şekilde iken hibrit algoritmanın temelini oluşturan yöntemlerin detayları alt bölümlerde verilmiştir.

2.1. Potansiyel Alanlar Yaklaşımı

Potansiyel alanlar yaklaşımı, doğadan ilham alınarak geliştirilmiş bir yöntemdir. Manyetik alanda şarj olan bir parçacık veya tepeden aşağıya doğru yuvarlanan bir topun davranışından yola çıkılarak bu hareket stratejisi robotiğe uygulanmıştır. Robotun çevresinde, onu hedefe doğru çekecek yapay potansiyel alanlar oluşturularak robotun hareket etmesi sağlanmaktadır. İlk olarak Oussama Khatib (Khatib, 1985) tarafından önerilmiştir.

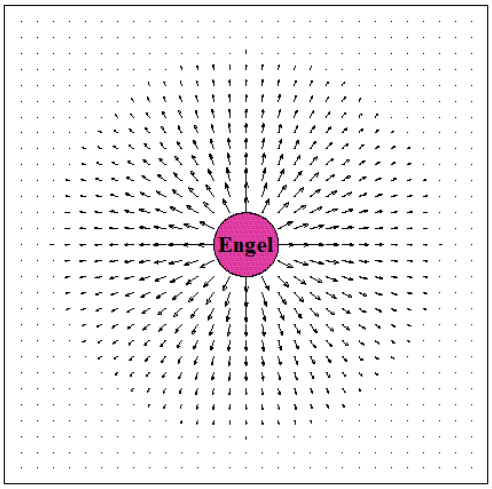
Robot, çevresindeki engelleri üzerindeki herhangi bir mesafe sensörü vasıtası ile tespit etmektedir. Robotun çevresinde herhangi bir engel yok iken, Şekil 2.2'deki gibi, robotu hedefe yönlendiren, çekici bir potansiyel alan oluşturmak yeterlidir. Çevredeki tüm boş alanlar için potansiyel alanlar tanımlanır.



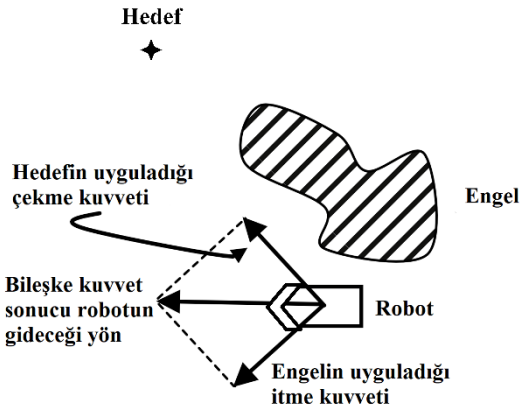
Şekil 2.2. *Hedefin çekme potansiyeli (Goodrich, 2002)*

Robotun çevresinde engeller var ise, Şekil 2.3'teki gibi bu engellerin itici bir potansiyel oluşturması sağlanır. Böylelikle robot engele yaklaştığında, engel, robotu

kendisinden uzaklaştırır. Robotun hedefe gitmesi ve engelden sakınması isteniyorsa, hedefin etrafında çekici bir potansiyel alan, hedefe giden yoldaki her engelin etrafında ise itici potansiyel alan oluşturulur. Robot bu alanların ürettiği kuvvetle Şekil 2.4'teki gibi hareket eder ve hedefine ulaşır.



Şekil 2.3. *Engelin itme potansiyeli (Goodrich, 2002)*



Şekil 2.4. *Hedefin uyguladığı çekme kuvveti, engelin uyguladığı itme kuvveti ve robotun hareketi*

En basit durumda, hesaplamalarda bir noktasal robot için, robotun anlık konumu q, Denklem 2.1'deki ifadeyle tanımlanır.

$$q = [x, y]^T \tag{2.1}$$

Çekme ve itme potansiyelleri sırası ile $U_{hedef}(q)$ ve $U_{engel}(q)$ olmak üzere, toplam potansiyel $U(q)$ Denklem 2.2 ile ifade edilir.

$$U(q) = U_{hedef}(q) + \sum U_{engel}(q) \tag{2.2}$$

Bu durumda üretilen yapay kuvvet F,

$$F = -\nabla U(q) = -\begin{pmatrix} \partial U/\partial x \\ \partial U/\partial y \end{pmatrix} \tag{2.3}$$

olacaktır (Dudek & Jenkin, 2000).

Çevreyi modellemek için hedef ve her bir engelin temsilinde potansiyel alan fonksiyonunu belirlemek gerekir. Bu fonksiyon literatürde, Denklem 2.4'te görüldüğü gibi, parabolik olarak karşımıza çıkmaktadır.

$$U_{hedef}(q) = \alpha \|q - hedef\|^2 \tag{2.4}$$

Denklem 2.4'te α pozitif bir çarpan, $\| \ \|$ simgesi ise hedef ile anlık robot konumu q, arasındaki Öklid mesafesini temsil eder.

Engel için ise yine β pozitif bir çarpan olmak üzere Denklem 2.5 geçerlidir. Burada robot konumu ve engel arasındaki mesafe bilgisi, robot üzerindeki sensör vasıtası ile elde edilmektedir.

$$U_{engel}(q) = \beta \|q - engel\|^{-2} \tag{2.5}$$

Potansiyel alanlar yaklaşımının başarısız olduğu bazı noktalar mevcuttur. Bunlar (Yolal, 2015),

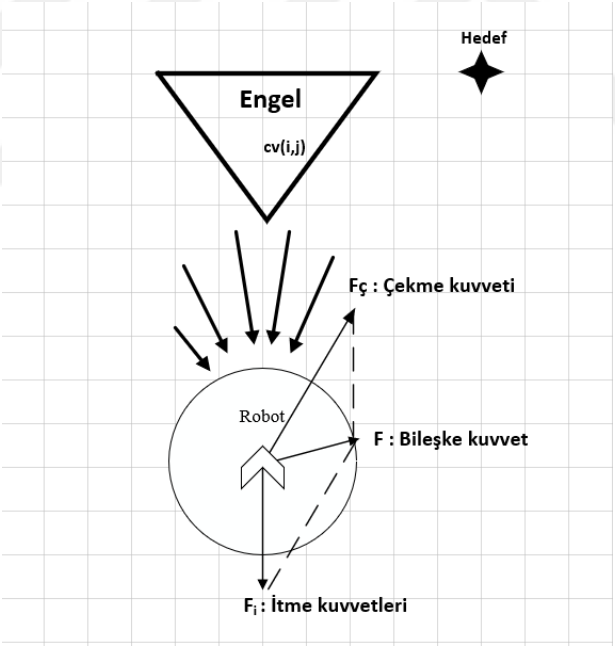
1. Robotun çalışma uzayında birden fazla yerel minimum noktası mevcut ise robot bunlardan herhangi birine takılarak hareket edemeyebilir.
2. Dar geçitlerde salınımlar meydana gelir.
3. Hedefe çok yakın engel mevcut ise robot hedefine ulaşamaz.

2.2. Sanal Kuvvet Alanları Yaklaşımı (The Virtual Force Field-VFF)

Sanal kuvvet alanları yöntemi, potansiyel alanlar yönteminde kullanılan ultrasonik veya hassasiyeti düşük sensörlerden alınabilecek düşük doğruluktaki verilerin eksikliğini karşılamak amacıyla Koren ve Borenstein (Borenstein & Koren, 1989) tarafından geliştirilmiştir. Bu amaçla, histogram grid (C) adı verilen 2 boyutlu kartezyen koordinat sistemi kullanılır. Bu histogram gridler, sensörden gelen, robot ve engel arasındaki mesafe verilerini temsil etmek amacıyla kullanılır. Histogramdaki her bir hücre (i, j), o hücrede engel bulunma olasılığını temsil eden kesinlik değeri (certainty value-cv) adı

verilen bir değere sahiptir. Kesinlik grid (certainty grid) kavramı ilk olarak Moravec ve Elfes (Moravec & Elfes, 1985) tarafından ortaya konulmuştur. Histogram griddeki her bir hücrenin kesinlik değeri, kullanılan mesafe sensöründen gelen verilere göre güncellenir. Eğer hücrelerde engel tespit edilir ise ilgili hücrenin kesinlik değeri arttırılır.

Histogram grid oluşturma ve potansiyel alanlar yaklaşımının birlikte kullanılması ile tasarlanmış VFF yönteminde, gerçek zamanlı sensör verisi, anlık olarak hücrelere işlenir. Bu işlem engel varlığını doğrular. Dolayısı ile potansiyel alanlar yaklaşımındaki sensörden gelebilecek düşük hassasiyetli veya hatalı mesafe verisinin yaratacağı olumsuz etkiler ortadan kalkmaktadır. Şekil 2.5'te robotun çalışma uzayının histogram grid üzerindeki durumu gözlenmektedir. Engele yakın hücrelerin kesinlik değerleri daha büyüktür, bu nedenle robota daha büyük bir kuvvet uygular. Robot ise engellerin uyguladığı itme kuvvetleri ve hedefin uyguladığı çekme kuvvetinin bileşkesi yönünde hareket eder.



Şekil 2.5. *Robota uygulanan kuvvetlerin gösterimi*

Sanal kuvvet alanı yöntemi de potansiyel alanlar kullanılarak geliştirilen bir yöntem olduğundan, potansiyel alanlar yönteminde karşılaşılan yerel minimum, dar geçitler ve bazı durumlarda hedefe ulaşamama gibi problemler bu yöntemde de karşımıza çıkmaktadır.

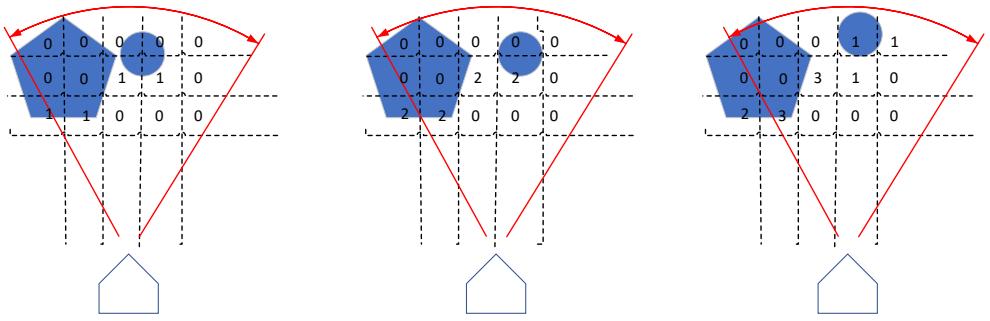
2.3. Vektör Alan Histogramı (The Vector Field Histogram-VFH)

Borenstein ve Koren (Borenstein & Koren, 1991) yeni bir yaklaşımla, yer robotları için bilinmeyen bir çevrede, mesafe sensöründen gelen verilerle, anlık olarak, engelsiz yol planı oluşturan Vektör Alan Histogramı (VFH) algoritmasını önermişlerdir.

VFH algoritması hem kara hem hava araçları için uygundur. Pek çok farklı robotik platformda geliştirilmiş ve uygulanmıştır. Algoritma üzerinde yapılan modifikasyonlardan en popülerleri, robot dinamiğini de ele alan VFH+ (Ulrich & Borenstein, 1998) ve lokal bir algoritma olmanın dezavantajından kurtaran VFH* (Ulrich & Borenstein, 2000) algoritmalarıdır.

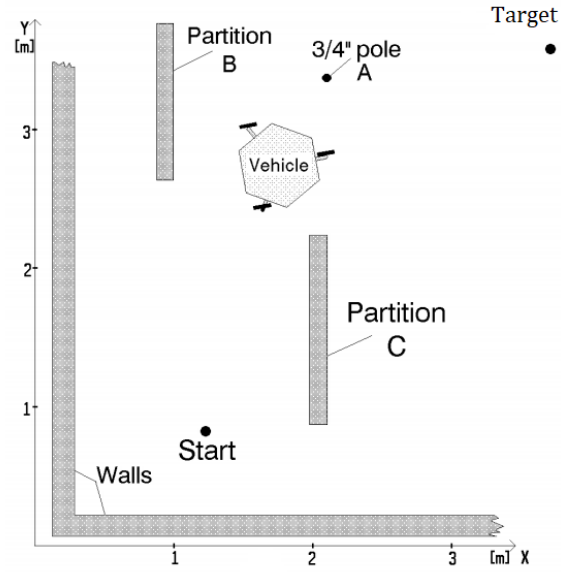
Algoritmanın işleyişine bakılacak olur ise; algoritma, 2 boyutlu kartezyen histogram grid, polar histogram sektörleri ve aday çukurlar olmak üzere üç temel parçadan oluşur.

Robot üzerindeki ultrasonik sensör, lidar gibi mesafe sensörlerinden gelen veriler, histogram grid üzerinde yer alacak engelleri haritalamak için kullanılır. Bunun için ise öncelikle robotu merkeze alan bir aktif pencere (C*) tanımlanır. Aktif pencere boyutu uygulamaya bağlı olarak ayarlanır. Aktif pencere, robot ile birlikte hareket etmektedir ve aktif pencere içinde yer alan hücreler aktif hücre olarak isimlendirilir (Borenstein & Koren, 1991). Aktif hücre değerleri başlangıçta 0 atanır. Daha sonra Şekil 2.6'da görüldüğü gibi robotun üç farklı konumuna göre (bir önceki konuma göre ilerlediği düşünüldüğünde), her mesafe okumada, o hücrede engel tespit edildiği takdirde, engel olasılığını ifade eden Kesinlik Değeri (Certainty Value-CV) 1 arttırılır. Robot ilerlerken ilgili hücrede engel tespiti yapılamaz ise 1 azaltılır. Bu işlem robot hareket ederken tekrarlanır. Böylece engellerin olasılıklı bir dağılımı hesaplanır. Değerin yüksek olması çarpışma olasılığının yüksek olmasını temsil eder.

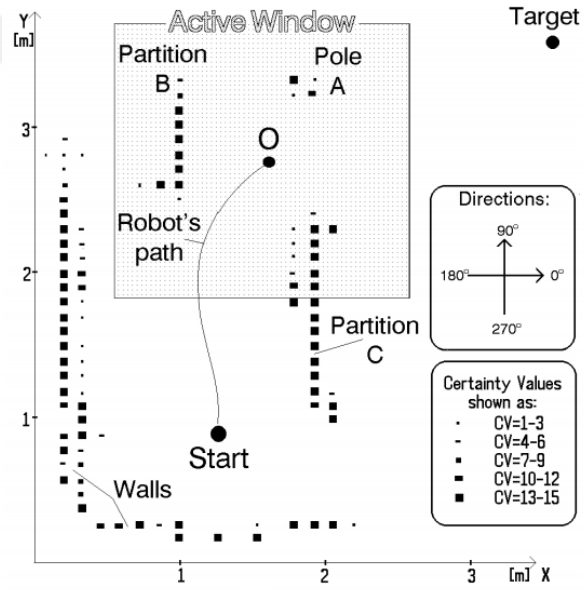


Şekil 2.6. *Sensör verilerine göre hücre değerlerinin güncellenmesi*

Algoritmanın işleyişindeki ikinci adım olan polar histogram sektör oluşturma aşaması Şekil 2.7'deki örnek ortam referans alınarak açıklanmıştır (Borenstein & Koren, 1991). Şekil 2.7'deki ortama karşılık gelen histogram grid temsili Şekil 2.8'de verilmiştir.



Şekil 2.7. *Örnek ortam (Borenstein & Koren, 1991)*



Şekil 2.8. *Histogram grid temsili (Borenstein & Koren, 1991)*

Şekil 2.8'de görüldüğü üzere A, B ve C engellerinin bulunduğu bölgelerin kesinlik değeri (CV) yüksektir. Aktif pencerenin merkezi, dolayısı ile robotun anlık konumu O noktası ile gösterilmiştir.

Engeller haritalandıktan sonraki adımda polar histogram oluşturulur. Polar histogram oluşturulur iken, robotun çevresi sektör adı verilen eşit sayıda açılara bölünmüş kısımlara ayrılır. Örneğin etrafını 360° tarayan sensöre sahip bir robot *n* tane sektöre bölünmüş ise sektör açısı *α*,

$$\alpha = \frac{360}{n} \tag{2.6}$$

olacaktır.

Aktif hücreler polar haritaya Denklem 2.7 ve Denklem 2.8 ile dönüştürülür (Borenstein & Koren, 1991):

$$\beta_{i,j} = \tan^{-1}\left(\frac{y_j - y_0}{x_i - x_0}\right) \tag{2.7}$$

$$k = INT\left(\frac{\beta_{i,j}}{\alpha}\right), k = 0,1,2, \dots n-1 \tag{2.8}$$

Burada (x_0, y_0) aktif hücre konumunu, (x_i, y_j) ise robotun anlık konumunu temsil eder. $\beta_{i,j}$ aktif hücreden aracın merkezine olan yön, *k* ise aktif hücreye karşılık gelen sektördür.

Aktif hücreleri tek boyutlu bir polar histograma dönüştürdükten sonra, aktif hücre büyüklüğü $m_{i,j}$,

$$m_{i,j} = \left(c^*_{i,j}\right)^2 (a - bd_{i,j}) \tag{2.9}$$

$$a - bd_{max} = 0 \tag{2.10}$$

olacaktır. Denklem 2.9'da $c^*_{i,j}$, (i, j) aktif hücresinin kesinlik değerini, $d_{i,j}$ aktif hücre ve robot arasındaki mesafeyi verir. *a* ve *b* ise Denklem 2.10 numaralı ifadeyi sağlayan pozitif sabitlerdir. d_{max}, aktif pencerenin köşegeninin yarısıdır, bir hücrenin aktif penceredeki en uzak yerini temsil eder. *a* keyfi bir tamsayı seçilir ise, *b* de kolaylıkla belirlenebilir.

k sektörü yönündeki engelle karşılaşma olasılığını temsil eden polar engel yoğunluğu h_k Denklem 2.11 ile hesaplanır:

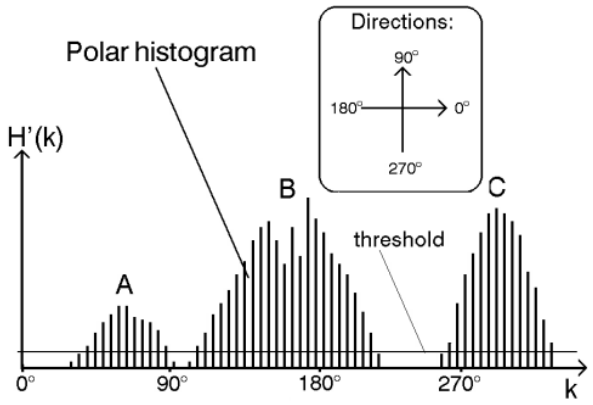
$$h_k = \sum_{i,j} m_{i,j} \tag{2.11}$$

Polar engel yoğunluğu dağılımının ayrık olması, engel olasılığının ayrık olmasına neden olabilir. Bu yüzden, daha düzgün bir polar engel yoğunluğu elde etmek için Denklem 2.12'deki gibi bir fonksiyona başvurulmuştur:

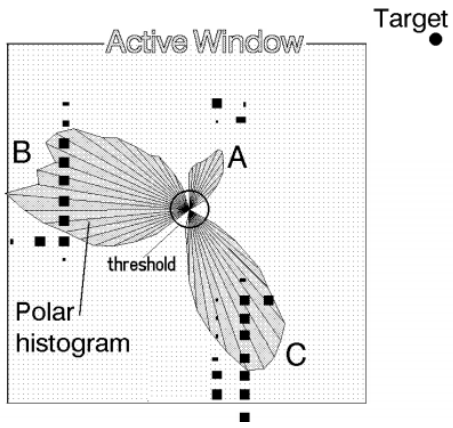
$$h'_k = \frac{h_{k-l} + 2h_{k-l+1} \ldots + lh_k + \cdots + 2h_{k+l+1} + h_{k+l}}{2l+1} \tag{2.12}$$

Denklem 2.12'de, k, sektör sayısı, l ise deneysel olarak seçilen sabit bir tamsayıdır.

Bu işlemlerden sonra Şekil 2.7'deki örnek ortam için elde edilmiş polar histogram Şekil 2.9'da verilmiştir. Bu polar histogramın, histogram grid üzerindeki gösterimi ise Şekil 2.10'da yer almaktadır.



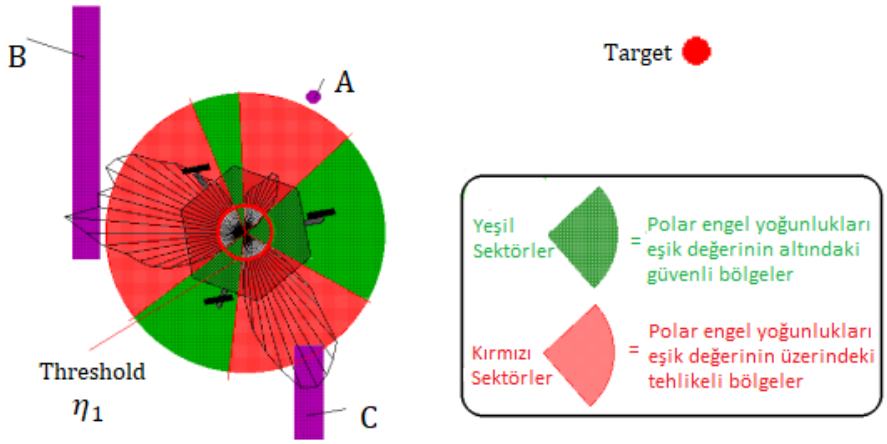
Şekil 2.9. *Polar histogram (Borenstein & Koren, 1991)*



Şekil 2.10. *Histogram grid üzerine yerleştirilmiş polar histogram (Borenstein & Koren, 1991)*

Polar histogramda yer alan tepe bölgeler engel bulundurma olasılığının yüksek olduğu bölgeleri, çukur bölgeler ise bu olasılığın düşük olduğu bölgeleri gösterir.

Bir sonraki aşama ise robotun gideceği yönü belirlemektir. Bunun için ilk olarak, robotun çevresindeki tüm sektörler Şekil 2.11'de gösterildiği gibi bir eşik değeri η_1 kullanılarak, polar engel yoğunluğu η_1 seviyesinin altındaki bölgeler yeşil renk ile gösterilen güvenli bölge, η_1 eşiğinin üstündeki bölgeler ise kırmızı renk ile gösterilen tehlikeli bölgelere ayrılmıştır. Robotun seçeceği aday yön, güvenli bölgeden olacaktır.

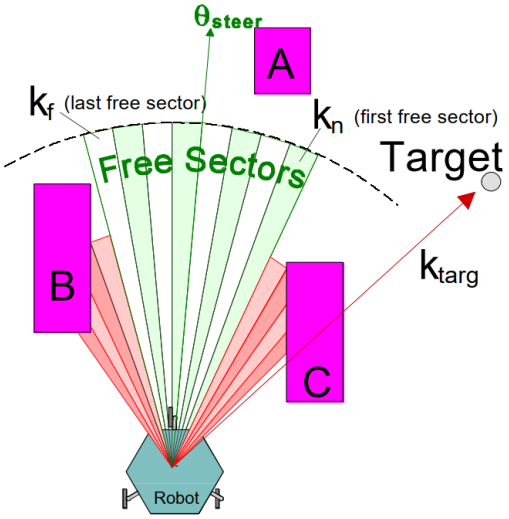


Şekil 2.11. *Sektörlerin engel yoğunluğuna göre güvenli ve tehlikeli bölge olarak ayrılması (Borenstein & Koren, 1991)*

Robotun hareket edebileceği aday bölgeler seçildikten sonra ise bu aday bölgeler polar histogramda çukur olarak da anılırlar- içerdikleri sektör sayısına göre başka bir eşik değeri η_2 kullanılarak dar ve geniş olarak sınıflandırılırlar. Aday çukurlar, s_{max} ile belirlenen bir sektör sayısından daha az sektör içeriyor ise dar çukur, bu sektör sayısından daha fazla sektör içeriyor ise geniş çukur olarak isimlendirilir.

Robot, dar ya da geniş olarak sınıflandırılan bu çukurlardan aday yönünü seçmektedir. Aday yön seçimi ise iki parametreye bağlı olarak yapılır. Bunlardan ilki Şekil 2.12'de gösterildiği gibi hedefe en yakın ilk boş sektör olan k_n'dir. Diğeri ise k_n'den s_{max} kadar uzakta yer alan k_f sektörüdür. Robotun seçeceği yön, θ_{steer}, Denklem 2.13'te verildiği gibi k_n ve k_f bu sektörlerinin orta noktası olacaktır.

$$\theta_{steer} = \frac{k_n + k_f}{2} \tag{2.13}$$

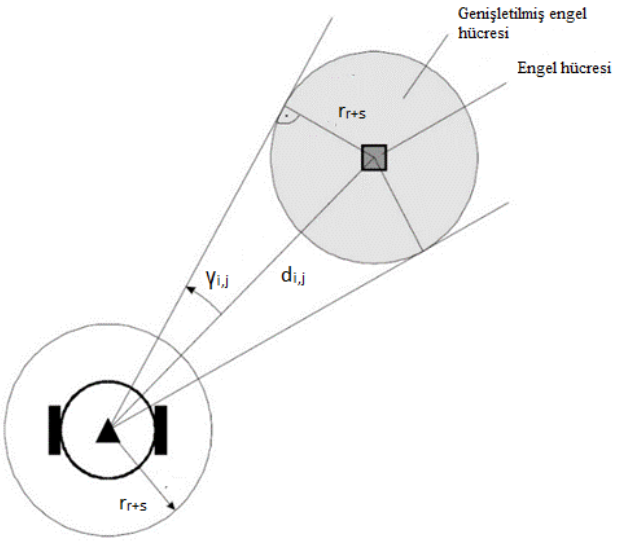


Şekil 2.12. *Robotun seçeceği yönün belirlenmesi (Borenstein & Koren, 1991)*

2.3.1. VFH+ algoritması

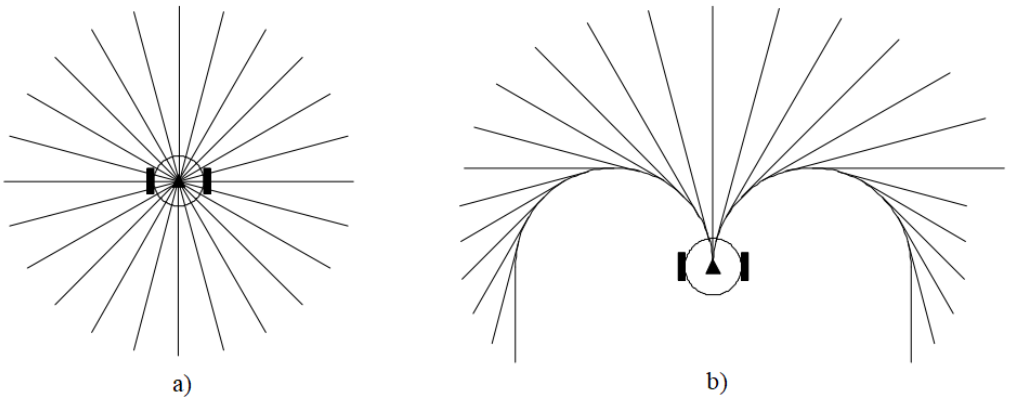
VFH algoritmasında robot noktasal kabul edilmekte, dolayısıyla robotun her yöne gidebileceği varsayılmaktadır. VFH+ algoritmasında (Ulrich & Borenstein, 1998), robotun dinamiği ve uygun yollar hesaba katılarak VFH algoritması iyileştirilmiştir. VFH'de kullanılan iki basamaklı veri indirgeme süreci yeni algoritmada dört basamağa çıkmıştır.

İlk basamakta VFH+ algoritması, iki boyutlu kartezyen histogram gridden robot dinamiğini (robotun güvenli hareket alanını) de hesaba katan tek boyutlu birincil polar histogram oluşturur. Robotun dinamiğini hesaba katmak için, ilk oluşturulan tek boyutlu histogramda aktif bölgedeki her bir engel hücresi Şekil 2.13'teki gibi genişletilir.



Şekil 2.13. *Robot genişliğinin ve güvenli bölgenin hesaba katıldığı durumda engel gösterimi (Ulrich & Borenstein, 1998)*

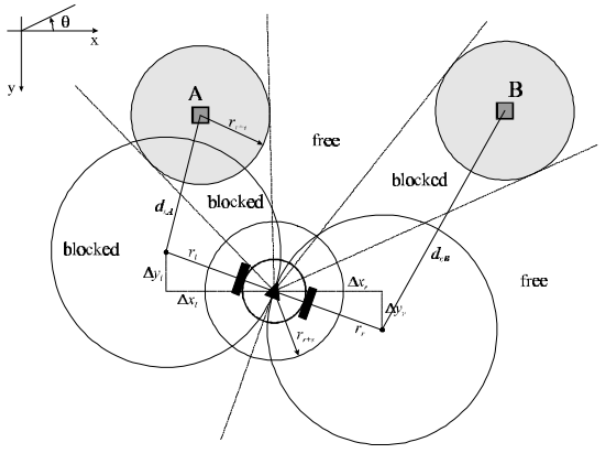
VFH+ yöntemi birçok yer robotunun hareketine benzer bir yörünge kullanır. Şekil 2.14 (a)'da robotun her yöne hareket edebildiği (robot noktasal veya holonomik varsayılabilir) durum görülmektedir. Şekil 2.14 (b)'de ise VFH+ yönteminin dikkate aldığı, robotun kontrol edilebilir serbestlik derecesinin toplam serbestlik derecesinden az olduğu dolayısıyla her yöne hareket edemediği (nonholonomic) durum görülmektedir.



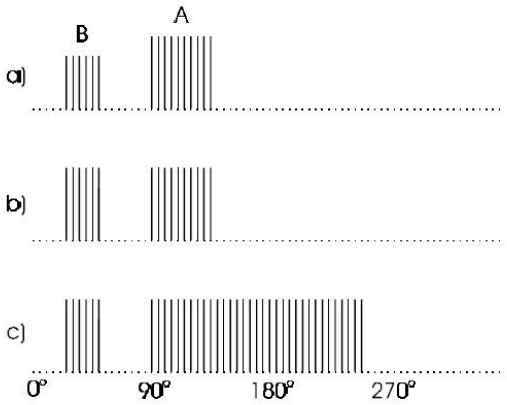
Şekil 2.14. *Robotun hareket kapasitesi* **a)** *Dinamiği dikkate alınmayan robot* **b)** *Dinamiği dikkate alınan robot (Ulrich & Borenstein, 1998)*

İlk basamakta, Şekil 2.15'te yer alan ortam için oluşturulan birincil polar histogram Şekil 2.16 (a)'da verilmiştir. Burada, A engeli için hesaplanan vektör büyüklüğünün B engeli için hesaplanan değerden büyük olduğu görülmektedir. Bu durum, A engelinin robota B engelinden daha yakın olduğunu göstermektedir.

İkinci basamakta ise bu birincil polar histogram, Şekil 2.15 (b)'de görüldüğü gibi, engelin varlığına göre boş veya dolu olarak değerlendirilmek için bir ikili (binary) polar histogram gride dönüştürülmüştür. Bu gösterime göre, robotun sol tarafı boş görülmektedir. Ancak, robotun yalnızca dairesel yörüngeler ile hareketi mümkün olduğundan, bu bölgeye gidememektedir. Üçüncü basamakta bu bölgeler dolu olarak gösterilmek kaydıyla maskelenmiş bir histograma çevrilir. Böylece, robotun gidemeyeceği yörüngeler ve hareket kısıtları hesaba katılmış olur.



Şekil 2.15. *Örnek ortam (Ulrich & Borenstein, 1998)*



Şekil 2.16. *Histogram gösterimi* **a)** *Birincil polar histogram* **b)** *İkili polar histogram* **c)** *Maskelenmiş histogram (Ulrich & Borenstein, 1998)*

Dördüncü basamakta ise VFH+ algoritması, maskelenmiş histogramda bulunan açıklıklara denk gelen bölgelerden birine robotu yönlendirmek için uygun yönü seçer. Bu seçim, VFH algoritmasında daha çok hedef odaklıdır, VFH+ algoritmasında ise aday yönler belirlendikten sonra bu yönler, Denklem 2.14'te verilen bir maliyet fonksiyonu ile değerlendirilir.

$$g(c) = \mu_1.\Delta(c, k_t) + \mu_2.\Delta(c, \theta_i/\alpha) + \mu_3.\Delta(c, k_{n,i-1}) \qquad (2.14)$$

Burada c aday yönü, k_t robotun o anki konumuna göre hedefin yönünü, θ_i robotun yönünü, α polar histogramın açısal çözünürlüğünü, $k_{n,\,i-1}$ ise bir önceki hareket yönünü temsil etmektedir. Δ fonksiyonu iki sektör arasındaki açıyı hesaplamaktadır.

İlk terim $\Delta(c, k_t)$, aday yön ile hedef yön arasındaki fark ile ilgili maliyeti temsil eder. Bu fark ne kadar fazla ise robot hedeften o kadar uzaklaşır.

İkinci terim $\Delta(c, \theta_i/\alpha)$, aday yön ile robot tekerleğinin yönü arasındaki fark ile ilgili bir maliyettir. Bu fark ne kadar fazla ise robot o kadar yön değiştirmek zorunda kalır.

Üçüncü terim $\Delta(c, k_{n,i-1})$ ise aday yön ve seçilen bir önceki yön arasındaki fark ile ilgili bir maliyettir. Bu fark ne kadar fazla ise robot bir önceki seçilen yönden o kadar sapar.

Maliyet fonksiyonundaki bu üç terimin ağırlıklandırılması ise μ değerleri ile yapılır. μ_1'e atanan yüksek bir değer, robotun hedef odaklı bir yön seçmesini sağlar. μ_2 değerinin yüksek seçilmesi, agresif manevraların önüne geçerek daha verimli yollar seçilmesine olanak sağlar. Son olarak μ_3 değerinin yüksek seçilmesi, robotun daha düzgün bir yörüngeden gitmesini sağlar.

Ağırlıklandırmalar, çalışılan robotun ne tür bir davranışı ön plana çıkarması isteniyor ise ona uygun μ değerlerinin seçilmesi ile sağlanır. Hedef odaklı bir davranış için VFH+ algoritmasının geliştiricilerinin önerdikleri koşul Denklem 2.15'te verilmiştir (Ulrich & Borenstein, 1998).

$$\mu_1 > \mu_2 + \mu_3 \tag{2.15}$$

2.3.2. VFH* algoritması

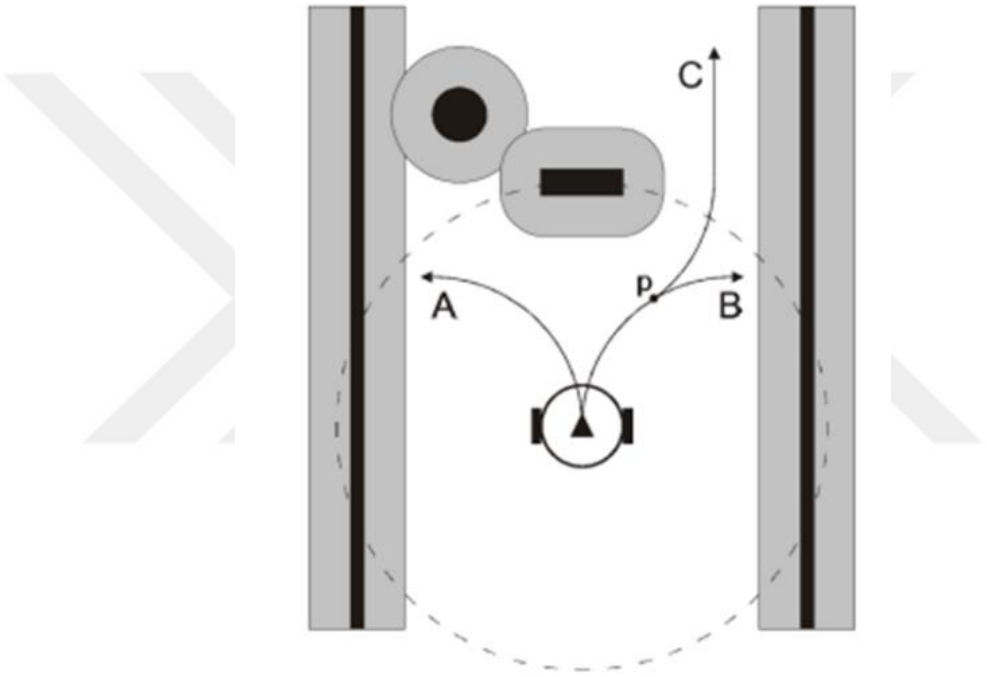
VFH+ algoritmasına yapılan iyileştirmeler 2000'li yıllarda yeni bir algoritma olan VFH* algoritmasının (Ulrich & Borenstein, 2000) geliştirilmesini sağlamıştır. VFH*, lokal engelden kaçınma algoritmaları (VFF, VFH, VFH+) için problem yaratan durumları göz önünde bulundurur. Algoritma işleyişi, hedef yön seçme aşamasına kadar VFH+ algoritması ile aynı şekildedir. Hedef yön seçiminde VFH* algoritmasının farkı ortaya çıkar. VFH* algoritması, robotu hedefe yönlendiren en iyi yönü seçmek amacı ile aday yönleri önceden değerlendirir.

Şekil 2.17'de verilen örnek ortamda robotun, engellerden kaçınmak için VFH+ algoritmasını kullandığı durumda, A ve B ile gösterilen aday yönlerden herhangi birini seçme olasılığı aynıdır. Bunun nedeni, robotun çevresindeki kesikli alan ile gösterilen aktif bölge dışındaki engelleri dikkate almamasıdır. Bu durumda robot, A yönünü seçtiğinde engele çarpabilir.

VFH* algoritması, VFH+ algoritmasını A* arama algoritması ile birleştirir. VFH+ algoritması ile aday yönler belirlendikten sonra, algoritma ileriye dönük doğrulama yapar.

Bunun için, her bir aday yönün seçimi sonrasında bu yönün seçiminin uygun sonuçlanıp sonuçlanmayacağı değerlendirilir. Aday yönlerden hangisinin seçilmesi gerektiği ise bir sonraki bölümde teorisi verilecek olan A* algoritması ile belirlenir. A* algoritmasında her bir yön için maliyet fonksiyonu oluşturulur. Maliyeti en düşük olan yön robotun hareket edeceği yön olarak belirlenir.

Şekil 2.17'deki durumda VFH* algoritması, A aday yönünün çıkmaza yol açacağını belirleyebilir iken, C yönünü sezgisel ve maliyet fonksiyonunu azaltacak en uygun yön olarak gördüğü için B yönünde gider.



Şekil 2.17. *Optimum yolu belirlemek için ileriye dönük doğrulama kullanan robot (Ulrich & Borenstein, 2000)*

2.4. A* Algoritması

A* algoritması, iki nokta arasında en kısa yolu bulmayı sağlayan bir algoritmadır. İlk olarak bir projede (Hart, Nilsson, & Raphael, 1968) robotun hareketlerini planlamak için kullanılmıştır. Ancak günümüzde kullanımı, bilgisayar oyunlarına kadar yaygınlaşmıştır. Oldukça popüler olmasının sebepleri arasında bir algoritmada aranan optimallik ve eksiksizlik (completeness) özelliğinin olması yer alır. Bu özellikler tanımlanacak olursa; bir algoritma olası en iyi çözümü veriyor ise o algoritma optimaldir. Eğer bir problemin en az bir çözümü mevcut ise ve bir algoritma sonlu bir zaman

aralığında bu çözümü bulabiliyor ise o algoritma eksiksiz (complete) olarak değerlendirilir.

Algoritmanın işleyişi şöyledir: Çalışma uzayındaki hücreler *f(n)* maliyet fonksiyonu ile değerlendirilir (Koubaa A. , ve diğerleri, 2018):

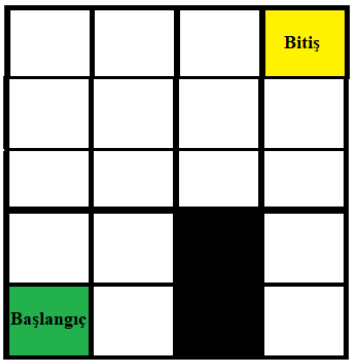
$$f(n) = h(n) + g(n) \tag{2.16}$$

A noktasından B noktasına giden bir yol planı düşünüldüğünde, *g(n)* başlangıç hücresi A'dan mevcut hücre *n*'ye olan en kısa mesafeyi temsil eder. *h(n)* ise mevcut hücre *n*'den hedef hücre B'ye olan en kısa mesafenin sezgisel maliyetidir. Mevcut hücre *n*'den hedef hücreye olan Öklid mesafesi ile ifade edilir. Bu durumda *f(n)* başlangıç hücresi A'dan, hedef hücre B'ye olan tüm yolların minimumunu ifade eden bir maliyet fonksiyonudur.

Algoritmanın işleyişi şöyledir:

1. Açık ve kapalı liste adı verilen boş iki liste oluşturulur. Açık liste gidilecek hücreleri, kapalı liste ise daha önce gidilmiş hücreleri temsil eder. Başlangıç hücresi açık listeye eklenir.
2. Hedef hücre bulunana veya açık liste boş kalana kadar şu işlemler tekrarlanır:
   a) Açık listedeki en düşük *f* maliyetine sahip hücre bulunur ve kapalı listeye alınır.
   b) Bu hücre ve tüm komşu hücreler için şu işlemler yapılır:
      i. Hücre kapalı listede ise göz ardı edilir.
      ii. Açık listede değil ise açık listeye eklenir, mevcut hücre komşu hücrenin ebeveyni olarak kaydedilir ve *f*, *g*, *h* maliyetleri hesaplanır.
      iii. Açık listede ise g maliyetleri kıyaslanır, yeni yol daha az maliyetli ise komşu hücrenin ebeveyn hücresi değiştirilir. *f*, *g*, *h* maliyetleri hesaplanır.
3. Açık liste boş ve hedef bulunamadıysa hedefe giden çözüm yolu yoktur.

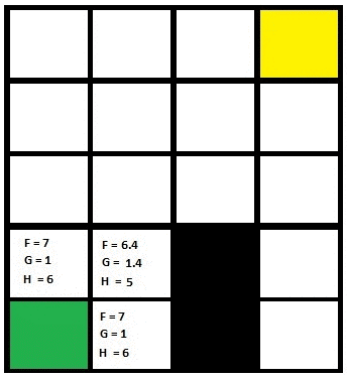
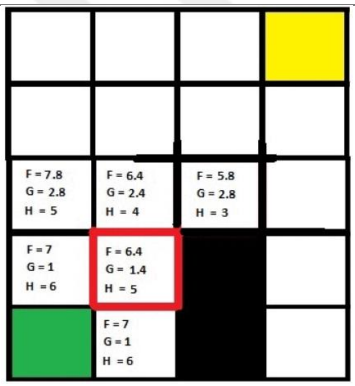
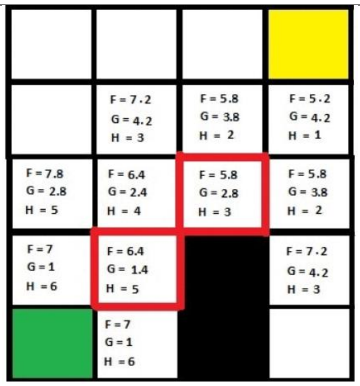
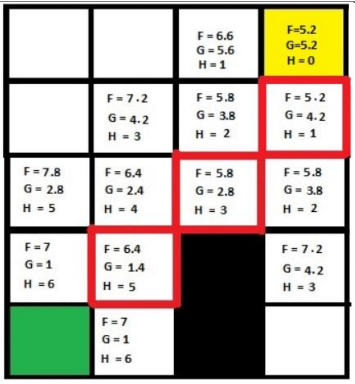
Yukarıda verilen işleyişe göre, Şekil 2.18'de gösterilen başlangıç noktasından bitişe olan yolun A* algoritması ile nasıl bulunduğu açıklanacaktır.



Şekil 2.18. *Örnek çalışma alanı*

Tablo 2.1'de görüldüğü gibi öncelikle üzerinde bulunulan hücrenin komşu hücrelerinin maliyetleri hesaplanmıştır. Bir sonraki adımda, toplam maliyeti (*f*) en düşük olan hücre gidiş yolu olarak seçilmiştir. Yine komşu hücrelerin maliyetlerine bakılarak en düşük maliyetli hücreye gidilmiştir. Bu işlem bitiş hücresini bulana kadar devam etmiştir.

Tablo 2.1. *A* algoritması ile örnek yol planı (Abiy, Pang, Tiliksew, Moore, & Khim, 2021)*

1. Adım	2.Adım
	
3. Adım	**4. Adım**
	

2.5. İncelenen Yol Planlama Yöntemlerinin Değerlendirilmesi

Bir yol planlama algoritmasından beklenen özellik, belirli bir noktadan başka bir hedef noktasına engellerden kaçınarak, yerel minimum noktalara takılmadan ulaşmasıdır.

Potansiyel alanlar yönteminin yerel minimumlara yakalanma problemi mevcuttur. VFF algoritması, yerel minimumlara takılma durumunu yeni sanal kuvvetler oluşturarak çözmeye çalışmıştır ancak dar koridor sorun yaratmıştır. Bu yüzden algoritma, hedefe erişebilir bir yol oluştursa da engeller etrafında optimal bir çözüm olmamıştır.

Benzer şekilde VFH algoritmasında da robot dinamiği ihmal edilmektedir. Robotun genişliğini dikkate almak ve polar histogramı düzgünleştirmek (smoothing) için alçak geçiren bir filtre kullanılmaktadır. Bu alçak geçiren filtrenin ayarlanması zordur ve uygulamada iyi ayarlanmış filtrelerle bile engele çarpma olasılığı olabilmektedir. VFH+ algoritmasında robotun genişliği ve hareket kabiliyeti dahilinde gidebileceği yollar dikkate alınmakla birlikte robot genişliğine göre engel hücreleri genişletilmektedir. VFH+ algoritmasının VFH algoritmasına göre üstünlüğüne bir noktada daha karşılaşılmaktadır. VFH'de tek bir eşik değeri vardır ve buna göre robotun gidebileceği açıklık (engel bulunmayan bölge) belirlenmektedir. Histogramda tek eşiğe bağlı olarak araç boyutundan biraz büyük bölgeler açıklık olarak değerlendirilmektedir. Aracın bu açıklıklardan geçerken zorlanması veya çarpması muhtemeldir. VFH+'da bu sorun iki eşik değeri belirlenmesi ile çözülmektedir. Bu iki eşik değerine göre polar engel yoğunluğu yerine küçük eşik değerinin altında kalan sektörler "0", boş, büyük eşik değerinin üstünde kalan sektörler "1", dolu, olarak değerlendirilmektedir. Böylece ikili (binary) polar histogram oluşturularak boş alanlar belirlenmektedir.

VFH* algoritması ise hem robot dinamiğini hesaba katan hem de yerel minimumlara takılma problemi olmayan bir algoritmadır. Bunun yanı sıra VFH*, VFH+'ya ek olarak kısmi olarak global planlayıcı A* ile engeller etrafında oluşturulan rotayı doğrular.

Tez kapsamında oluşturulan algoritma da VFH+ ve A* algoritmasını bir araya getirmiştir ancak VFH*'dan farklıdır. VFH* algoritmasında (Ulrich & Borenstein, 2000; Van Breda & Smit, 2016) tüm aday yönler bulunana kadar VFH+ algoritması çalışır. Aday yönlerin ileriye dönük değerlendirilmesi yapılırken, bir başka deyişle her bir aday

yönün seçimi sonrası oluşturulacak yol planının hangisinin seçileceği konusunda A* algoritmasına başvurulur.

Önerilen yöntemde ise ilk olarak hava aracının çalışacağı bölge taranarak ortam haritası çıkarılır. Daha sonra bu haritaya göre, hedef koordinatı da verilerek A* ile yol planı oluşturulur. Hava aracı bu yol planına bağlı kalarak ilerler. Bu plan dışında, ortama sonradan eklenen bir engel var ise bu engelden sakınmak amacıyla VFH+ algoritmasını kullanır. Engelden sakınırken A* ile oluşturulmuş yol planının dışına çıkılırsa yeniden planlama yapılır.

3. LİTERATÜR ARAŞTIRMASI

Bir yerden başka bir yere gitmek insanlar için oldukça basit bir iştir. İnsan, birkaç saniye içinde nasıl hareket edeceğine karar vererek harekete geçebilir. Ancak bir robot için bu basit görünen iş, zor bir görevdir. Robotik alanında, otonom bir robot için yol planlama temel bir problemdir. Bu temel problem gerek bir mobil robot için gerek bir robotik kol için gerekse bir hava aracı için başlangıç konumundan hedef konumuna güvenli bir şekilde ulaşmaktır. Bu problem, literatürde çalışılan ortam, robot türü ve uygulamanın şekline bağlı olarak birçok şekilde ele alınmıştır.

Yol planlama algoritmalarının temelleri 1960'lı yılların sonlarında atılmıştır. Nils John Nilsson, Görünürlük Grafiği Yöntemi'ni (Nilsson, 1969) bir mobil robot sistemi için geliştirmiştir. Bu çalışmasının yanı sıra Nilsson, A* sezgisel arama yöntemini (Hart, Nilsson, & Raphael, 1968) geliştirenlerden biridir. Yol planlama 1970'lerde de araştırma konusu olmuştur ancak asıl atılımlar bilgisayarların geliştirilmesi ile 1980'lerde meydana gelmiştir. Bu dönemde çok sayıda yeni yöntem geliştirilmiştir. Bu algoritmaların temellerine ve gelişimine, Jean-Claude Latombe 1990'ların başında yayınladığı kitabında (Latombe, 1991) yer vermiştir. Yol planlama konusunda 90'ların trendi ise çalışma uzayını temel alan rastgeleleştirmeyi kullanan planlayıcılardır. Günümüzde ise modern yöntemler, örneklemeye dayalı yaklaşımlar kullanmaktadır.

Yol planlamada en yaygın yaklaşımlardan biri çizge arama algoritmalarıdır. Bu yaklaşımda arama uzayı, ağırlıklandırılmış hücrelerle oluşturulan çizgeler ile temsil edilir. Çizge arama algoritmalarının büyük çoğunluğunun temeli Dijsktra algoritmasına (Dijkstra, 1959) dayalıdır. Dijsktra algoritması bir düğümden komşu düğüme geçmenin maliyetinin değerlendirerek ve düğümleri en düşük maliyetle bağlayacak şekilde çalışır. Genişlik Öncelikli Arama (Breadth First Search-BFS) (Russel & Norvig, 2010) algoritması da Dijsktra ile benzer şekilde çalışan bir başka algoritmadır ancak BFS mevcut düğümden hedefe olan mesafeyi kullanarak hedefe daha yakın olan düğümü seçer ve böylece arama süresini kısaltır. Bir sonraki bölümde ayrıntılı anlatılacak olan A* algoritması ise düğümler arasındaki dolaşma maliyeti ve hedefe ulaşma maliyetini birleştirir. Dinamik A* (Liao, 2012) algoritması ise kenar maliyetlerinin dinamik olarak arttırılmasını veya azaltılmasını sağlayarak A* algoritmasını dinamik ortamlarda da uygulanabilir hale getirmiştir. Bir diğer algoritma Theta* (Daniel, Nash, Koenig, & Felner, 2010) ise çizge düğümleri arasındaki farklı bağlantıları ele alarak diğer

algoritmalardan farklılaşmıştır. A* ve Theta* algoritmalarının araç dinamiği göz ardı edilerek simülasyon ortamında performansları değerlendirilmiştir (Filippis, Guglieri, & Quagliotti, 2012). Theta*'ın daha düzgün yol planı yapabildiği ancak A*'ın işlem süresi açısından daha başarılı olduğu görülmüştür. Bu algoritmalarda göze çarpan durum araç dinamiklerini ihmal etmeleridir. 2007 yılında yapılan bir başka çalışmada (Hwangbo, Kuffner, & Kanade, 2007) düğüm bağlama kuralları araç dinamiklerini de içerecek şekilde tanımlanmıştır. Sabit kanatlı bir İHA için minimum dönüş yarıçapına ve maksimum tırmanma açısına göre grid yapıda bulunan hücre boyutları belirlenmiştir. Benzer şekilde Filippis ve Guglieri çalışmalarında (Filippis & Guglieri, 2012), yine sabit kanatlı bir İHA için araç kinematiğini de dikkate alarak yol planı yapmıştır. Bu algoritmalar, uygulama kolaylığı ve uyarlanabilirliği sayesinde birçok araştırmacının ilgisini çekmiştir. Ancak arama uzayının boyutu hesap yükünü arttırmaktadır ve bu da istenen bir durum değildir.

Rastgele Ağaçlar Yöntemi (Rapidly-exploring Random Trees-RRT) (LaValle, 1998; Lin & Saripalli, 2015) ise bir başka popüler arama algoritmasıdır. Olasılıksal Yol Haritası (Probabilistic Road Map-PRM) metodu gibi (Kavraki, Svestka, Latombe, & Overmars, 1996) örnekleme tabanlı bir yol planlama yöntemidir. İkisinin de temeli arama alanından rastgele örneklenen noktaları birleştirme fikrine dayanır.

Karınca Kolonisi Optimizasyonu (Ant Colony Optimization-ACO) (Dorigo & Thomas, 2004), karıncaların doğada yiyecek arama davranışlarından esinlenerek geliştirilen bir algoritmadır. Karıncaların yuvalarından besin kaynağına giden en kısa yolu buldukları bilinmektedir. Bunu, yolları boyunca salgıladıkları feromon adı verilen kimyasalın izi ile başarırlar. Bu iz, diğer karıncalar tarafından tespit edilebilir ve besin kaynağına ulaşmak için tercih edilir. Bu çözüm, telekomünikasyon ağlarında ağ yönlendirmesi, gezgin satıcı problemi, iş çizelgeleme ve robot yol planlaması gibi çeşitli problemleri çözmek için kullanılmıştır (Brand, Masuda, Wehner, & Yu, 2010; Englot & Hover, 2011; Gigras, 2012; Rashid ve diğerleri, 2016; Wen & Cai, 2006). İHA yol planlama problemine de uygulanmıştır (Ma, Duan, & Liu, 2007; Zhang, Zhen, Wang, & Li, 2010). Planlama olarak iyi sonuçlar vermiş olsa da hesaplama süresi açısından çevrimiçi uygulamalara uygun olmadığı görülmüştür.

Yapay Potansiyel Alanlar yöntemi (Khatib, 1985), fiziksel potansiyel alanlardan esinlenerek geliştirilen bir yaklaşımdır. Bu yaklaşımda araca, çekici ve itici güçlerin

neden olduğu bir alanda hareket eden yüklü bir parçacık olarak davranılır. Gidilecek yol noktaları çekici kuvvetlerle ilişkilendirilir ve itici kuvvetler engellerin etrafına yerleştirilir. İtme kuvvetinin büyüklüğü, araç ile engel arasındaki mesafeye bağlı olacaktır. Çekici ve itici kuvvetlerin toplamı, aracı hedefe doğru götüren net bir potansiyel alanı ortaya çıkarmaktadır.

Potansiyel alanlar temelli yöntemler bir önceki bölümde ayrıntılı bir şekilde incelenmiştir. Bu yöntemler hem statik hem de dinamik engellerden kaçınarak düzgün uçulabilir yollar sağlar. Diğer yaklaşımlar ile karşılaştırıldığında, bu yöntemi kullanmak hesaplama açısından maliyetli değildir, bu nedenle gerçek zamanlı olarak kullanılabilir. Ayrıca, çarpışmadan kaçınma yeteneklerine sahip formasyon uçuşları için de yaygın olarak kullanılan bir yaklaşımdır (Paul, Krogstad, & Gravdahl, 2008).

Literatürde, statik ve dinamik ortamlarda yol planlama probleminin çözümüne yönelik hibrit yöntemler yer almaktadır. Bu çalışmalarda Parçacık Sürü Optimizasyonu (PSO)- Potansiyel Alanlar (Mandava, Bondada, & Vundavilli, 2019), Dijsktra- TEB (Time elastic band) (Marin-Plaza, Hussein, Martin, & Escalera, 2018) ve birkaç farklı kombinasyonun birarada kullanılmasına yer verilmiştir.

4. YÖNTEM

Tezin bu bölümünde hava aracının engelden sakınarak hedefine güvenli bir şekilde ulaşmasını sağlayacak yöntemin detayları verilecektir.

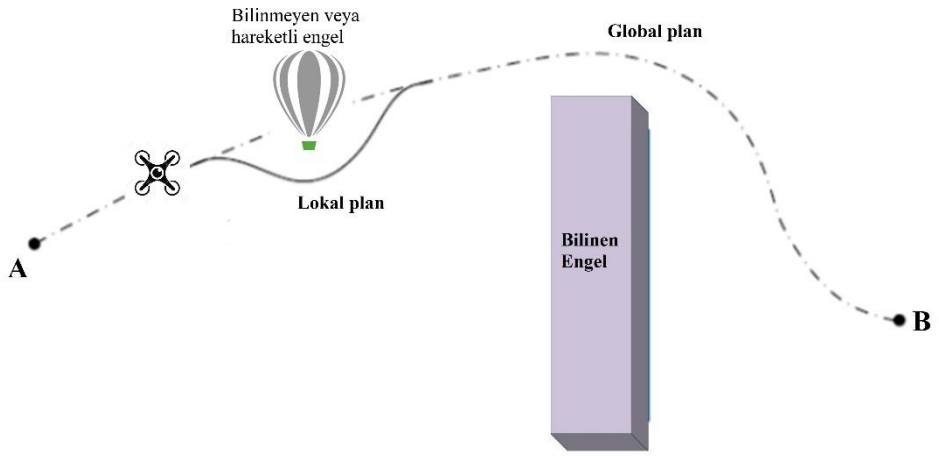
Literatürde yer alan yol planlama algoritmalarının işleyişi 2. bölümde verilmiştir. Bu yöntemler değerlendirildiğinde, tez kapsamındaki probleme ancak global ve lokal yöntemler bir arada kullanılarak çözüm getirilebileceği sonucuna varılmıştır.

A* algoritması bir başlangıç noktasından bir hedef noktasına en kısa yoldan ulaşmayı sağlayan global bir yöntemdir. Dezavantajı, işlem yükünün daha ilkel algoritmalara göre fazla olmasıdır.

VFH+ algoritması ise araç dinamiği dikkate alınarak geliştirilmiş, ortamdaki mevcut ve sonradan eklenen engellerden sakınabilen lokal bir algoritmadır. Lokal yöntemlerin doğasından kaynaklanan lokal minimuma takılma, osilasyon gibi sorunların varlığından dolayı dezavantajları mevcuttur.

Hava aracının holonomik bir sistem olmasına rağmen VFH yerine VFH+ kullanılmasının nedeni araç genişliğini dikkate alarak engele çarpma olasılığını en aza indirmek ve aracın engel bulunmayan ancak geçemeyeceği kadar dar açıklıklara yöneliminin önüne geçmektir. Bunların yanı sıra uygulamada araç konum bilgisine erişimde kullanılan GPS verisi çok hassas olmadığından bir belirsizlik bölgesi oluşabilir. Bu belirsizlik araç boyutuna dahil edilerek algoritmanın daha sağlıklı sonuçlar vermesi sağlanabilir.

Bu iki yöntemi Şekil 4.1'deki gibi bir arada kullanmak, ikisinin de dezavantajlarını geri planda bırakarak avantajlarını öne çıkaracaktır. Şekil 4.1 incelendiğinde hava aracı A noktasından B noktasına giderken öncelikle A* algoritması ile global bir plan oluşturulur. Daha sonra yolda karşılaşılan bilinmeyen bir engel için VFH+ ile lokal plan uygulanarak hava aracının hedefe ulaşması sağlanır. A* algoritmasını bir kere çalıştırmak işlem yükünden kaçınmayı sağlayacak, VFH+ ise önceden verilmiş harita dışındaki ortama sonradan dahil olabilecek engellerden kaçınmayı sağlayacaktır.

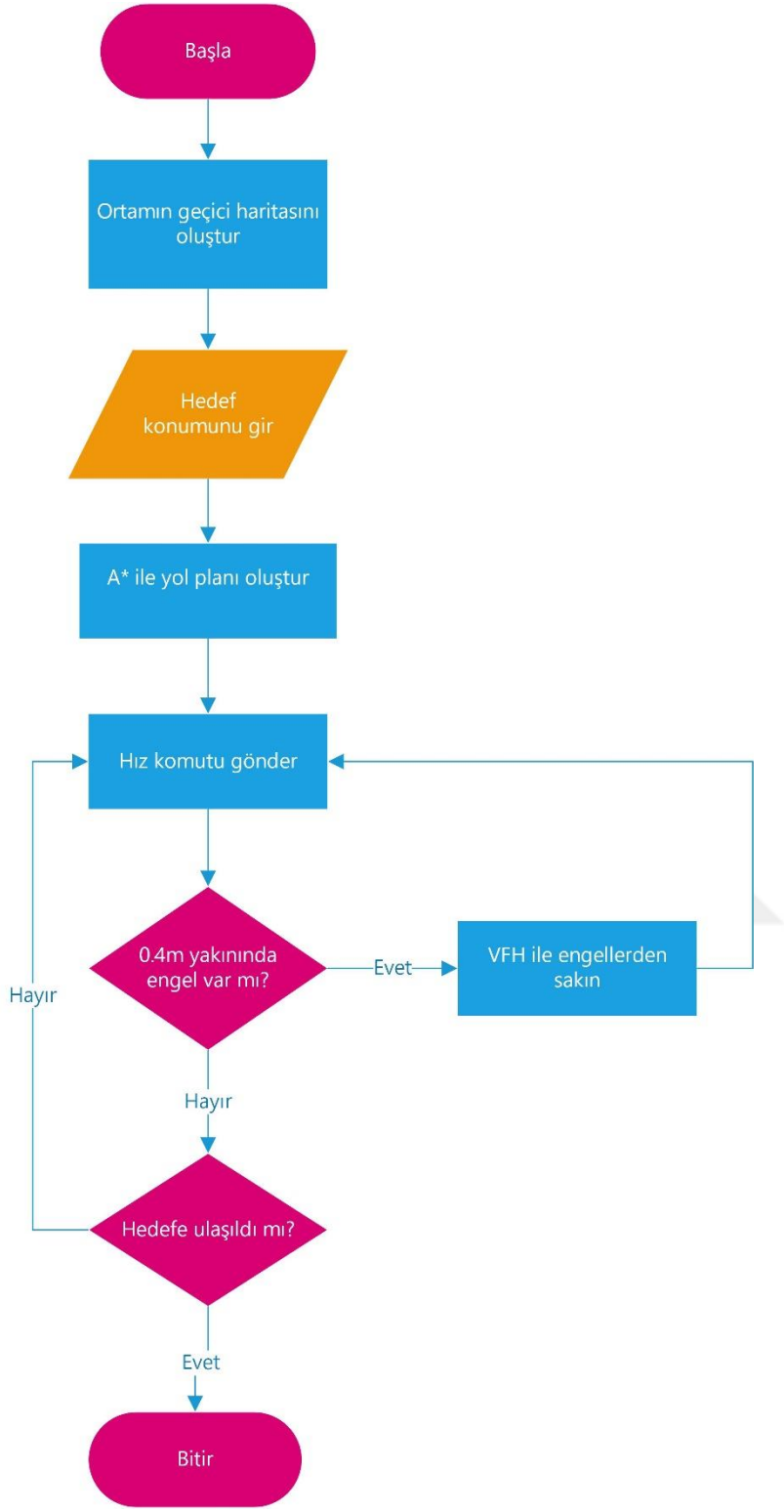


Şekil 4.1. *Hibrit plan kulanımı*

Oluşturulan hibrit yöntemde, hava aracının boyutlarından dolayı engele yaklaşma mesafesi açısından ve keskin dönüşleri engellemek için minimum dönüş yarıçapı belirlemek amacıyla mesafeler konulmuştur. Bunlar dışında yöntem modelden bağımsız çalışmaktadır.

4.1. Algoritmanın İşleyişi

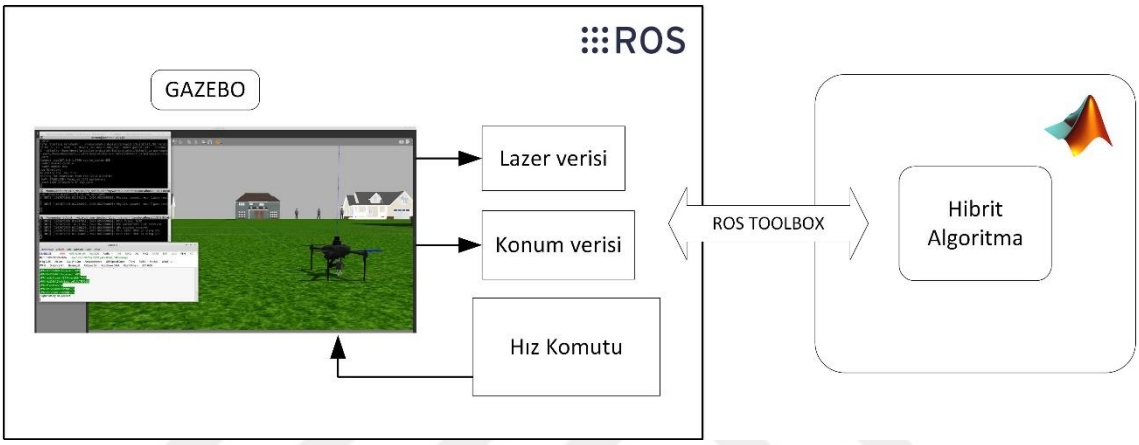
Algoritmanın akış diyagramı Şekil 4.2'de verilmiştir. Buna göre hava aracı, kullanıcı tarafından atanan belirli bir sürede etrafını üzerindeki lidar vasıtası ile tarar ve çevresindeki engelleri haritalamasını sağlayacak doluluk ızgarası oluşturur. Harita oluştururken engelden sakınma, VFH+ algoritması ile sağlanmaktadır. Harita oluşturulduktan sonra, hava aracının haritalamayı durdurduğu konum, başlangıç konumu olarak atanır. Bu aşamada, kullanıcıdan hedef konumunun girilmesi istenir. Hedef ve engellerin konumuna göre A* algoritması ile global bir yol planı oluşturulur. Bu yol planı ayrık noktalardan oluşmaktadır. Bu ayrık noktalardan oluşturulacak rota bir sonraki alt bölümde verilecektir. Hava aracı global plana uyarak hedefine doğru ilerlerken herhangi bir engele 0,4 metreden fazla (hava aracı boyutlarına göre güvenlik açısından belirlenen) yaklaştığında veya planda olmayan bir engelle karşılaştığında global plandan çıkarak VFH+ ile engelden sakınarak yeni engel bulunmadığı takdirde yine global planla hedefine ulaşmaya çalışacaktır.



Şekil 4.2. *Hibrit algoritma akış diyagramı*

Algoritma MATLAB programında kodlanmıştır ve ROS ile iletişim ise MATLAB'in sağladığı ROS Toolbox aracılığı ile kurulmuştur. Şekil 4.3'te görüldüğü üzere Gazebo ortamındaki hava aracı üzerindeki lazer sensörden alınan veriler ve hava aracının GPS'ten alınan konum bilgisi MATLAB'te yazılan algoritmada kullanılmak üzere ROS Toolbox ile alınmaktadır.

MATLAB'te yazılan algoritmanın hesapladığı hız kumandaları ise benzer şekilde ROS Toolbox ile hava aracına gönderilmektedir.



Şekil 4.3. *MATLAB-ROS iletişimi*

ROS'ta hava aracının konum bilgisi birçok ROS başlığından okunabilmektedir. Bu bilgiler farklı ROS başlıkları ile sadece GPS'ten, sadece uçuş kontrol biriminden (FCU) veya her ikisinden gelen veriler birlikte alınabilmektedir. Yer eksenine göre (global koordinatlara) ya da aracın kendi eksenine göre okunabilmektedir.

Konumla ilgili yayınlanan başlıklar şöyledir:

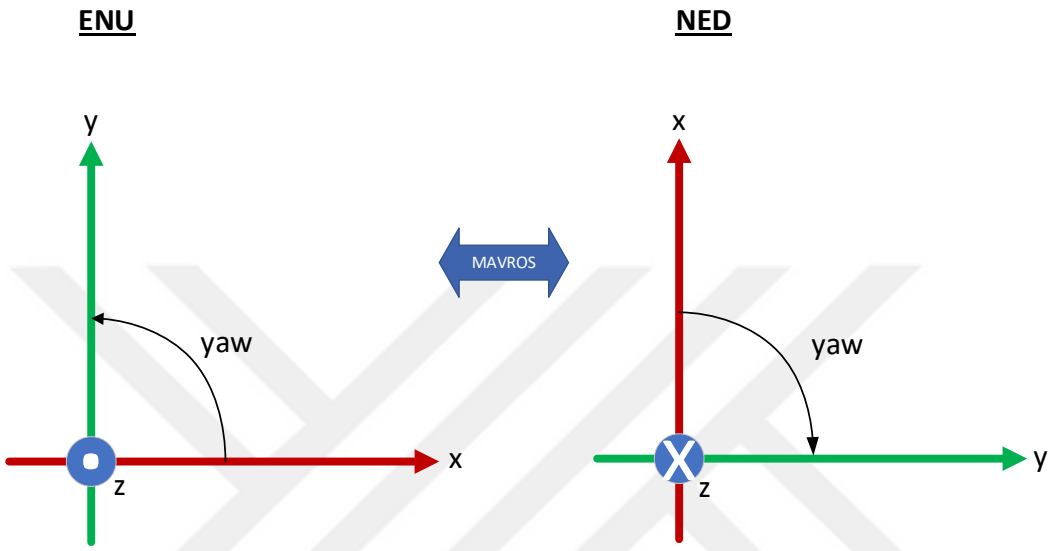
• "/mavros/global_position/global" başlığı sadece GPS'ten gelen verileri göstermektedir.

• "/mavros/local_position/pose" başlığı sadece uçuş kontrol biriminden gelen verileri göstermektedir.

• /mavros/global_position/local başlığı ise GPS ve uçuş kontrol biriminden gelen verileri birlikte filtrelenmiş olarak göstermektedir.

Bu verilerden hangisine ihtiyaç duyuluyorsa o başlığa abone olunarak kullanılabilir. Algoritmada hava aracının konumu, daha yüksek doğruluklu veriyi kullanmak için "/mavros/global_position/local" başlığı ile alınmıştır. Bu başlıktan okunan veri ENU

(East-North-Up) koordinat sistemine göre yani aracın +x ekseni Doğu'yu, +y ekseni Kuzey'i, +z ekseni ise yukarı yönü gösterecek şekilde belirlenmiştir. Ancak, hava aracı için algoritmada kullanılan koordinat sistemi NED (North-East-Down) olduğundan koordinat dönüşümüne ihtiyaç duyulmuştur. Bu dönüşüm ise Şekil 4.4'te görüldüğü gibi Mavros paketi ile yapılabilmektedir.



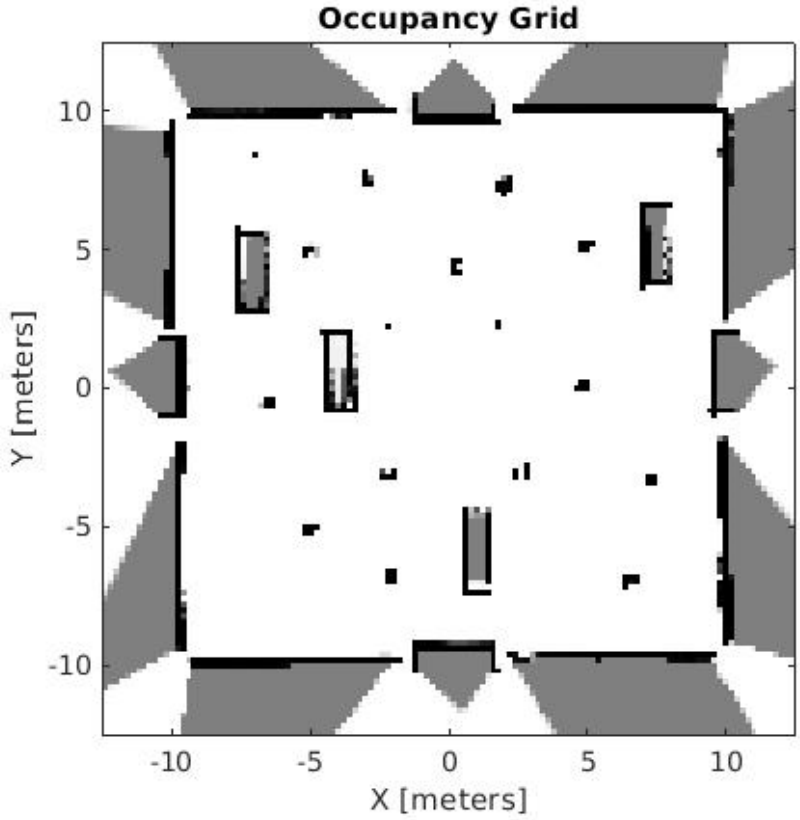
Şekil 4.4. *ENU-NED koordinat sistemleri dönüşümü*

4.2. Doluluk Izgaraları

Doluluk ızgaraları yöntemi ile haritalama ilk olarak Moravec ve Elfes (Moravec & Elfes, 1985) tarafından önerilmiştir. Yöntem, yol planlama algoritmalarında engelsiz yollar bulmak, robotun kendi konumunu tespit etmesini sağlamak (lokalizasyon) gibi amaçlarla robotik uygulamalarda sıklıkla kullanılmaktadır.

Haritası oluşturulacak bölge eşit aralıklara bölünür ve bunların her birine hücre adı verilir. Bu hücreler engel bulundurma olasılıklarına göre belirlenen bir değerle temsil edilir. Engel bulunma durumunda o hücrenin değeri 1, bulunmama durumunda 0, taranmamış ise -1 ile ifade edilmektedir.

Tez kapsamında yol planı yapabilmek amacıyla, hava aracı çalışacağı ortamda belirli süre uçuş yaparak lidar sensöründen alınan veriler ile ortamdaki engeller tespit edilerek ortamın doluluk haritası elde edilmiştir. Bu yöntemle oluşturulan bir harita Şekil 4.5'te verilmiştir.

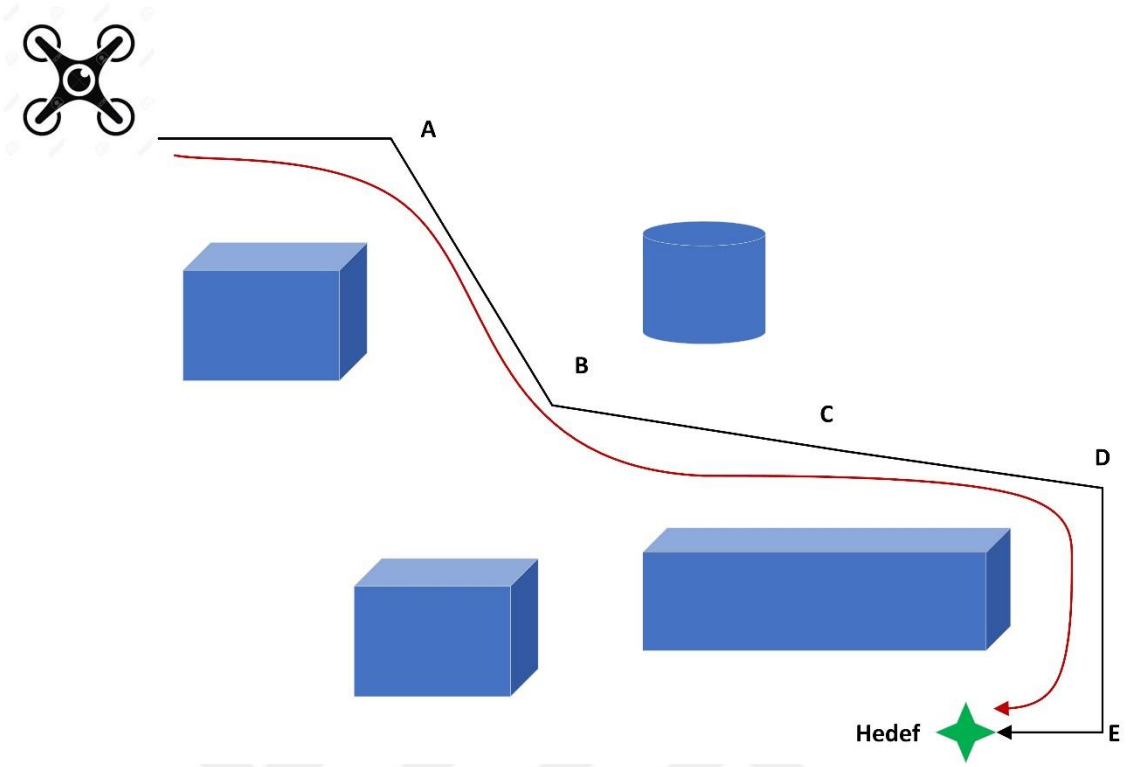


Şekil 4.5. *Doluluk haritası örneği*

Şekil 4.5'te verilen haritada, engelsiz alanlar beyaz, engel bulunan alanlar siyah, taranmamış alanlar gri ile temsil edilmektedir.

4.3. Rota Oluşturma

Yol planlama algoritması kullanılarak elde edilen yol noktaları düz çizgiler ve keskin dönüşler içerebilir. Şekil 4.6'da örnek bir ortamda hava aracının hedefe gittiği iki farklı yol görülmektedir. Siyahla çizilen, algoritmanın oluşturduğu noktaların birleşiminden oluşan yol, keskin dönüşler içermektedir ve bu araç dinamiği açısından istenen bir davranış değildir. Dolayısı ile bu yolun kırmızı ile gösterilen, daha düzgün (smooth) bir rotaya dönüştürülmesi gerekmektedir.



Şekil 4.6. *Yol düzgünleştirme örneği*

Bir noktadan bir noktaya rota planlama, belirli noktalarda konumların zamanın bir fonksiyonu olarak temsil edilmesi ile mümkündür. Rota boyunca hız ve ivmeler, konumun türevinin hesaplanması ile elde edilir. Düzgün bir yol planı için hızda süreksizlikler olmamalıdır.

Yol noktaları ve koşullara göre düzgün bir yol planı polinomlar ile hesaplanabilir. Literatürde rota planı için kullanılan polinomlar (Spong, Hutchinson, & Vidyasagar, 2005) incelendiğinde, sınır koşulu sayısının polinom derecesini belirlemede kullanıldığı görülmüştür. Sınır koşulu n ise rota için kullanılacak polinomun derecesi (*n-1*) olmalıdır.

Tez kapsamında ele alınan problem için, A* algoritması ile elde edilen yol noktalarını düzgün bir rotaya dönüştürmek gerekir. Sınır koşulları olarak başlangıç ve bitiş noktalarındaki konum, hız ve ivme koşulları ayrı ayrı dikkate alındığında n=6 olmaktadır. Dolayısıyla rotayı en az beşinci dereceden (quintic) polinom kullanarak oluşturmak uygun olacaktır (Öcal, 1990).

4.2.1. Beşinci dereceden polinomlar kullanarak rota planlama

5. dereceden bir polinom ile iki nokta arasındaki rota Denklem 4.1'deki gibi tanımlanabilir (Spong, Hutchinson, & Vidyasagar, 2005).

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{4.1}$$

q(t) ile gösterilen konum ifadesinin türevinin alınması ile Denklem 4.2'deki hız ve Denklem 4.3'teki ivme ifadeleri elde edilir.

$$\dot{q}(t) = v(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \tag{4.2}$$

$$\ddot{q}(t) = a(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \tag{4.3}$$

Yukarıdaki ifadelerden yola çıkarak başlangıç zamanı t_0 ve varış zamanı t_f olmak üzere, bu anlardaki konum denklemleri, birinci türevleri ile hız denklemleri, ikinci türevleri ile de ivme denklemleri sırasıyla Denklem 4.4 ve Denklem 4.9 arasında verildiği gibi elde edilir.

$$q(t_0) = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 t_0^4 + a_5 t_0^5 \tag{4.4}$$

$$v(t_0) = a_1 + 2a_2 t_0 + 3a_3 t_0^2 + 4a_4 t_0^3 + 5a_5 t_0^4 \tag{4.5}$$

$$a(t_0) = 2a_2 + 6a_3 t_0 + 12a_4 t_0^2 + 20a_5 t_0^3 \tag{4.6}$$

$$q(t_f) = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \tag{4.7}$$

$$v(t_f) = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \tag{4.8}$$

$$a(t_f) = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 \tag{4.9}$$

Bu denklemler matris formunda Denklem 4.10'daki gibi yazılabilir.

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ a_0 \\ q_f \\ v_f \\ a_f \end{bmatrix} \tag{4.10}$$

Buradan başlangıç ve bitiş konum, hız ve ivme bilindiği takdirde a_0, a_1, ..., a_n katsayıları elde edilebilir.

Tez kapsamındaki problem ele alınacak olur ise A* algoritması ile elde edilen yol noktaları $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ çiftlerinden oluşmaktadır.

Denklem 4.10'daki matris formundan yola çıkılarak oluşturulan yeni matrisler Denklem 4.11, Denklem 4.12 ve Denklem 4.13'teki notasyonlarla ifade edilebilir.

$$Q = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ & & & \ldots & & \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \end{bmatrix} \tag{4.11}$$

$$A = [a_0\ a_1\ \cdots a_n]^T \tag{4.12}$$

$$B = [q_0\ v_0\ q_1\ v_1 \cdots q_n\ v_n]^T \tag{4.13}$$

Hava aracının hareketine başladığı zaman ($t_0=0$) ve hedefine ulaştığı zaman, t_f, başlangıç ve hedef konumları arasındaki mesafeye uygun olarak belirlendiğinde *Q* matrisi elde edilebilir.

A* algoritması ile elde edilen yol noktaları bilindiği ve her bir yol noktasında hava aracının sahip olması istenen bir ortalama hız değeri atandığı takdirde *B* matrisi de elde edilebilir. Bu durumda herbir yol noktasındaki konum ve hızlar bilindiği takdirde *A* katsayılar matrisinin elemanları Denklem 4.14 ile belirlenebilir.

$$A = Q^{-1}B \tag{4.14}$$

Katsayılar matrisinin elde edilmesi sonucu hava aracının A* algoritması ile elde edilen yol noktalarına gittiği süre boyunca izlediği konumlara, hızlara ve ivmelere ait denklemler elde edilebilir.

Elde edilen hız, hava aracına gönderilerek istenen yol planı dahilinde gitmesi sağlanmaktadır.

5. KULLANILAN YAZILIM ARAÇLARI

Bu bölüm ve alt bölümlerde tez çalışması süresince benzetim ortamında kullanılacak olan hava aracı, yazılım araçları ve kurulan sistemin genel mimarisi hakkında bilgiler verilmiştir.

5.1. Hava Aracı Modeli

Bu tez çalışmasında kullanılan dört rotorlu hava aracı Şekil 5.1'de görülen 3DR firmasının Iris+ isimli modelidir.

Şekil 5.1. *Iris+ hava aracı (3D Robotics Inc., 2014)*

Quadrotor veya quadkopter olarak da isimlendirilen dört rotorlu hava araçları 4 motor/pervaneden oluşur. Kararlılığı sağlamak amacıyla pervanelerden karşılıklı ikisi saat yönünde diğer ikisi de saat yönününün tersi yönde döner. Motor ve pervaneler quadrotorların kütle merkezine eşit uzaklıkta kare şeklinde yerleştirilir. Herbir motor bir itki kuvveti üretir ve üretilen toplam itki kuvveti, taşıma kuvveti (lift) oluşturarak, yerçekimi kuvvetinden büyük olduğu takdirde quadrotorun havalanmasını sağlar.

Quadrotorlar helikopterlere kıyasla çok daha basit mekaniğe sahiptirler, maliyetleri daha düşüktür ve daha küçük motorlara sahip olduklarından kaza durumunda daha az hasara sebebiyet verirler. Manevra kabiliyetleri açısından diğer çok rotorlu hava araçlarından daha etkindirler. Trikopterlerin kontrolü zordur ve yük taşıma kapasiteleri düşüktür. Hekzakopterlerin ve diğer çok rotorluların taşıma kapasiteleri yüksektir ancak hareket kabiliyetleri esnek değildir.

Helikopter veya sabit kanatlı hava araçlarının uçuş davranışları hareketli kontrol yüzeyleri ile sağlanır. Quadrotorlarda ise kontrol yüzeyleri mevcut değildir, aracın uçuş esnasındaki hareketi dört motorun her birinin dönüş hızı ve yönü ile belirlenir. Bu suretle, Iris+ model quadrotora etkiyen kuvvet ve momentler incelenmiştir. Hareket denklemleri, Newton'ın ikinci yasası kullanılarak aracın dinamiklerini temsil etmektedir. Kuvvet ve

momentlerin oluşumunda rotorlar, yer çekimi ve gövde aerodinamiği etkilidir. Bu çalışmada, Iris+'nın dinamiği analiz edilirken aracın simetrik olduğu ve ağırlık merkezinin geometrik merkez olduğu varsayımlarında bulunulmuştur.

5.1.2. Koordinat sistemleri

Iris+ hava aracının dinamiği incelenirken *x* ekseninin aracın ileri yönünü, *y* ekseninin aracın sağ tarafını, *z* ekseninin ise aracın aşağı yönünü gösterdiği gövde eksen takımı [x^B, y^B, z^B] kullanılmıştır. Kuvvet F^B ve momentler M^B aracın ağırlık merkezine uygulanmıştır. Öteleme hızı V^B ve dönme hızı w^B, ivmeler $\dot{V}^B$, $\dot{w}^B$ de aracın gövde ekseninde tanımlanmıştır (Liu, 2017).

$$F^B = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}, \quad M^B = \begin{bmatrix} L \\ M \\ N \end{bmatrix} \tag{5.1}$$

$$V^B = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad w^B = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{5.2}$$

$$\dot{V}^B = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix}, \quad \dot{w}^B = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \tag{5.3}$$

Denklem 5.1-5.3'te *L*, *M*, *N* yalpa, yunuslama ve sapma momentlerini; *u*, *v*, *w* aracın *x*, *y* ve *z* yönlerindeki hızlarını; *p*, *q*, *r* ise yalpa, yunuslama ve sapma açısal hızlarını temsil etmektedir.

Hava aracının yalpa, yunuslama ve sapma davranışlarını temsil etmek için kullanılan Euler açıları sırasıyla φ, θ, ψ'dır. Açıların türevleri ve hızlar arasındaki ilişki Denklem 5.4'te verildiği gibidir.

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan\theta \sin\varphi & \tan\theta \cos\varphi \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi/\cos\theta & \cos\varphi/\cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{5.4}$$

Iris+'nın konum ve hızları yer eksen takımına göre ölçülür. Yer eksen takımı, yer (dünya) merkezli ancak dönmediği varsayılan, birbirine dik üç vektör ile ifade edilen eksen takımıdır. Ölçülen verileri gövde eksen takımına dönüştürmek için $T_{B/R}$ dönüşüm matrisi tanımlanmıştır. $T_{R/B}$ matrisi ise $T_{B/R}$ matrisinin transpozudur.

$$T_{B/R} = R_1(\varphi).R_2(\theta).R_3(\psi) \tag{5.5}$$

$$R_1(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \tag{5.6}$$

$$R_2(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{5.7}$$

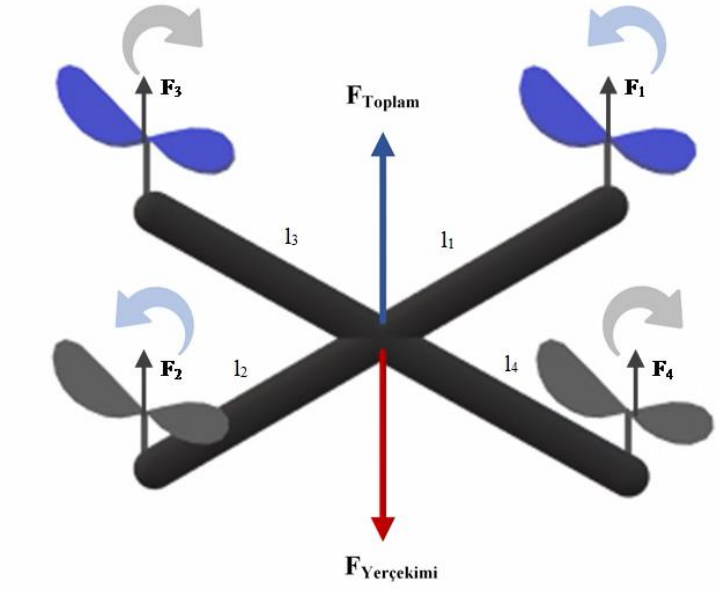
$$R_3(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

Dönüşüm matrisini kullanılarak elde edilen öteleme hızları referans koordinat sistemine (yer eksen takımı) göre Denklem 5.9'da verildiği gibidir:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = T_{R/B} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{5.9}$$

5.1.3. Kuvvet ve momentler

Iris+ hava aracına etki eden kuvvet ve momentler Şekil 5.2'de görülmektedir. 4 pervane $F_{1\text{-}4}$ itki (thrust) kuvvetlerini oluşturur. Pervanelerin ürettiği toplam itki kuvveti, F_{Toplam} yerçekimi kuvvetinin etkisine karşı koyar (Liu, 2017).



Şekil 5.2. *Hava aracına etki eden kuvvet ve momentler*

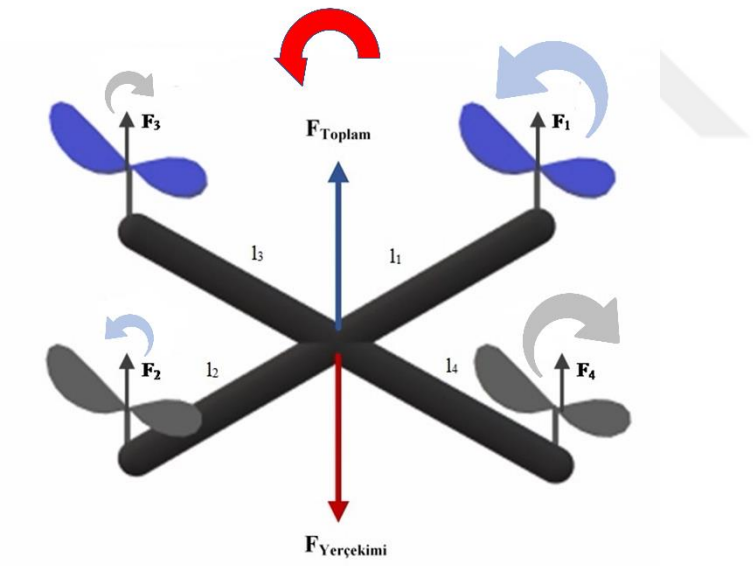
Pervanelerin hepsi aynı hızda döndüğünde hava aracının ağırlık merkezindeki toplam tork ve kuvvet dengesi sağlanır ve hava aracı askıda kalır.

Askıda kalma durumunda kuvvet ve momentlerin toplamı sıfırdır.

$$F_{Yerçekimi} - F_1 - F_2 - F_3 - F_4 = 0 \tag{5.10}$$

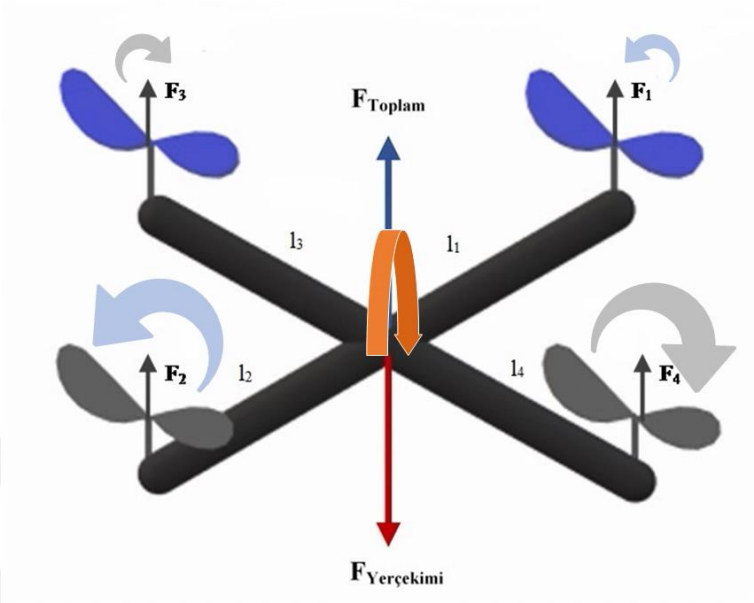
$$L = M = N = 0 \tag{5.11}$$

Hava aracının farklı eksenlerdeki hareketleri, herbir motorun dolayısıyla pervanenin farklı hızda (RPM) döndürülmesi ile sağlanır. Motorun hızındaki değişiklik motorun ürettiği itkinin ve oluşan momentin değişimine neden olacaktır. Hava aracına etkiyen bileşke moment değişimi aracı dönmeye zorlayacaktır. Şekil 5.3'te 1 ve 4 numaralı motor hızları arttırılarak 2 ve 3 numaralı motor hızları aynı oranda azaltılarak aracın sola doğru yalpa momenti kazandığı görülmektedir.



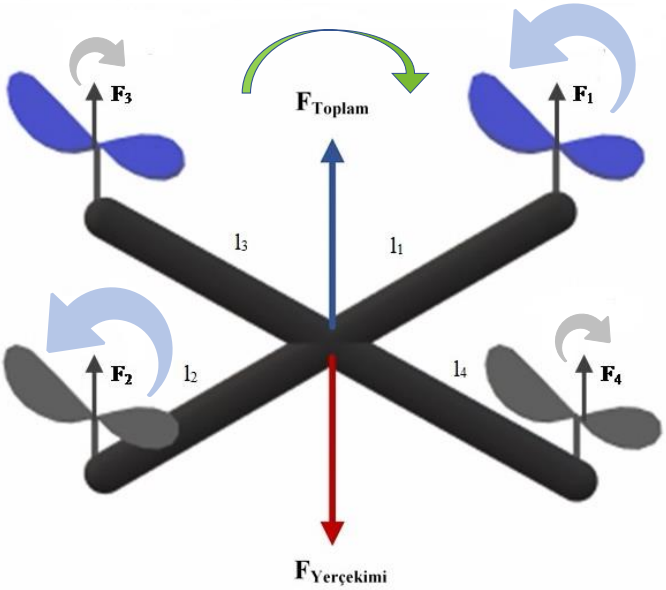
Şekil 5.3. *Yalpa momentinin oluşumu*

Şekil 5.4'te 2 ve 4 numaralı motor hızları arttırılmış 1 ve 3 numaralı motor hızları ise aynı oranda azaltılmıştır. Böylece hava aracı yunuslama momenti kazanmıştır.



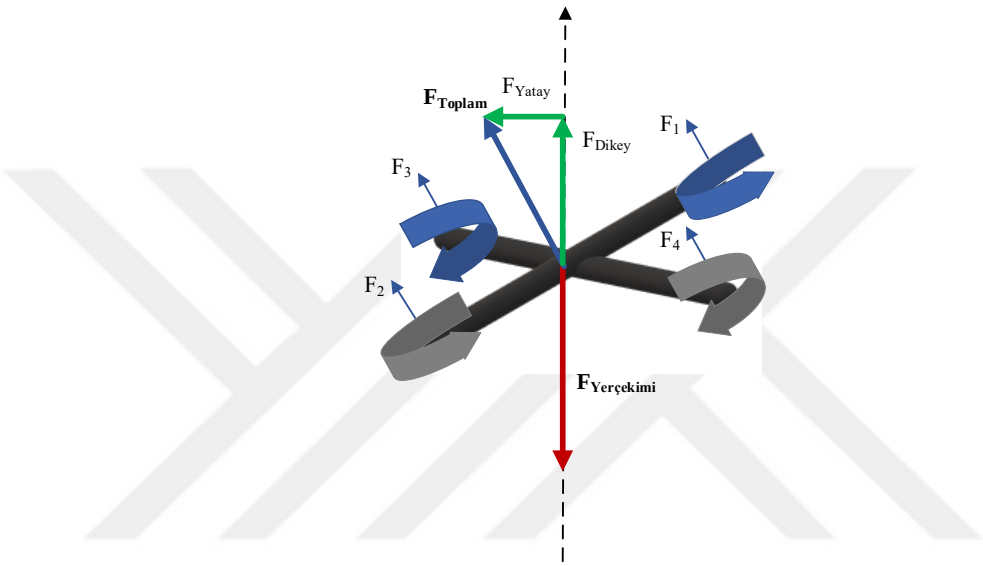
Şekil 5.4. *Yunuslama momentinin oluşumu*

Şekil 5.5'te ise 1 ve 2 numaralı motor hızları arttırılmış 3 ve 4 numaralı motor hızları ise aynı oranda azaltılmıştır. Böylece hava aracı sapma momenti kazanmıştır.



Şekil 5.5. *Sapma monentinin oluşumu*

Burada dikkat edilmesi gereken durum, araç yalpa veya yunuslama hareketi yaptığında itki kuvvetinin yatay bir bileşeni oluşmaktadır. Bu durum Şekil 5.6'da görülmektedir. Bu kuvvet bileşeni hava aracına bir ivme verir ve araç yana doğru hareket edebilir. İtki kuvvetinin dikey bileşenindeki bir değişiklik ayrıca dikey bir ivme de ortaya çıkarır. Farklı rotor RPM'leri ile quadrotor, 6 serbestlik derecesine (Degree of Freedom-DOF) sahiptir. Bu, quadrotorun x, y, z eksenleri boyunca hareket etme ve yalpa, yunuslama, sapma eksenleri etrafında dönme kabiliyetine sahip olduğunu göstermektedir.



Şekil 5.6. İtki kuvvetinin yatay ve düşey bileşenleri

5.2. Robot İşletim Sistemi (Robot Operating System-ROS)

Uygulamada farklı konfigürasyonda, farklı görevler yapmak için tasarlanmış oldukça fazla sayıda robot mevcuttur. Bu robotlar için bir ortamda çalışırken, robot konfigürasyonlarında, kullanılan sensör ve algoritmalarda ortak özellikler olabilmektedir. Bu amaçla 2000'li yılların ortalarında Stanford Üniversitesi'nden bir grup araştırmacı robotik alanındaki fikir paylaşımı yapmak, kaynak oluşturmak ve paylaşmak için ROS platformunu oluşturmuşlardır (Quigley, ve diğerleri, 2009). ROS başta robot kontrolünü ve programlamayı sağlayan araçlar olmak üzere beraberinde (Pyo, Kurazume, & Watanabe, 2015)

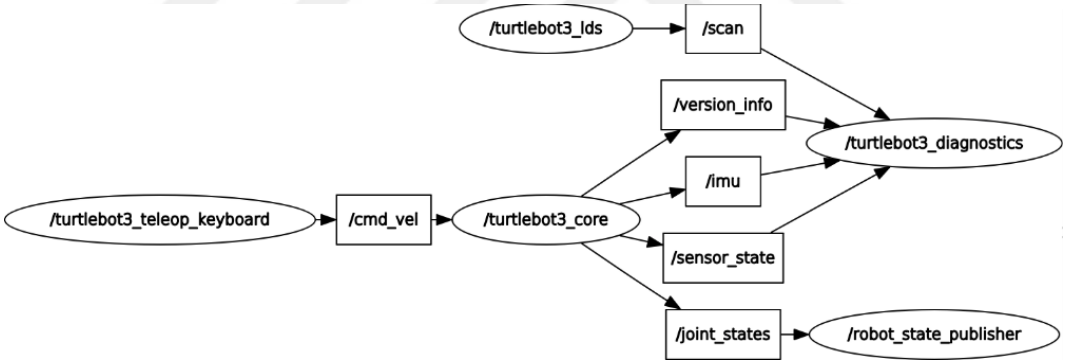
- sensörlerden veri okumayı sağlayan ve motorlara, eyleyicilere kumanda göndermeyi sağlayan sürücü setleri,
- robotik uygulamalarda kullanılabilen birçok algoritma,
- karmaşık robot sistemlerinin bileşenlerini bağlama ve kendi algoritmanızla birleştirme

- çeşitli hesaplama altyapıları
- robot görselleştirme ve hatalı davranışlarda hata ayıklamayı ve sensör verilerini kaydetmeyi kolaylaştıran geniş yelpazede araç setine sahiptir.

ROS platformunun gücü yukarıdaki maddelerde de belirtildiği üzere, çok yönlülüğüne dayanmaktadır. Kullanıcı, ROS aracılığıyla projesine özel, asıl problemlere odaklanırken, problemin temel noktaları (konumlandırma, sensör sürücülerini yazma vb.) platformun kendisi tarafından çözülmektedir.

5.2.1. ROS yapısı

ROS (ROS Wiki, 2020), yayın yapma ve abone olma yapısını kullanarak aralarında haberleşen düğümlerden oluşur. Düğümler çeşitli alt görevleri yerine getirmek üzere yazılmış programlardır. Düğümlerde yapılan işlem sonucu elde edilen veriler, çeşitli başlıklar altında yayınlanabilir veya başka bir düğümün yayınladığı veriye ilgili başlığa abone olunarak erişilebilir. ROS bu şekilde birbiriyle mesajlar vasıtası ile haberleşen çok sayıda bağımsız programdan oluşur. Tüm sistem diyagramlarla ifade edilebilir.



Şekil 5.7. *Turtlebot robotunun klavye ile kumanda edilmesi*

Örnek diyagram, Turtlebot yer robotu modelinin klavye ile kumanda edilmesi esnasında ROS'ta gerçekleşen işlemlere aittir. Oval bloklar düğümleri, dikdörtgen bloklar ise mesajlar vasıtası ile haberleşen başlıkları temsil etmektedir. Klavye ile kumanda işlemini yerine getiren düğüm "/turtlebot3_teleop_keyboard" robot düğümüne hız kumandasını ilgili başlık "/cmd_vel" vasıtasıyla yayınlar. Robot düğümü aynı zamanda robot üzerinde bulunan çeşitli sensör ve robotun eklemlerini ilgili başlıklar vasıtasıyla yayınlar. Bu başlıklar hangi düğüm tarafından kullanılacaksa abone olunur.

ROS kullanırken anlaşılması gereken ilk önemli kavram, düğüm kavramıdır.

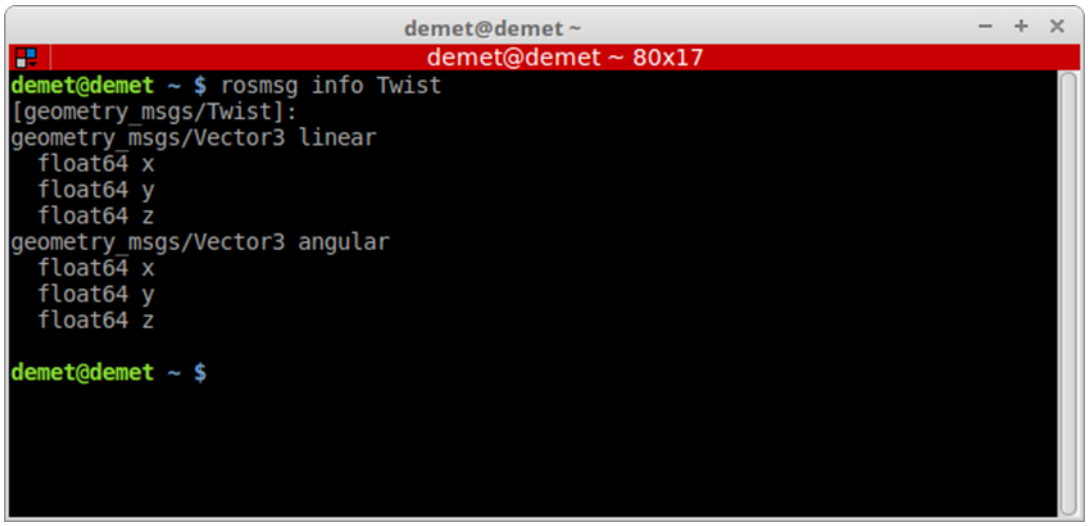
5.2.1.1. Düğüm (Node)

Düğümler belirli görevleri yerine getirmek için yazılmış programlardır. Robot uygulamalarında, robotun her bir alt görevi başka bir düğüm tarafından kontrol edilir. Örneğin bir düğüm robotun konumlandırmasını yaparken bir başka düğüm kameradan aldığı görüntüyü işler. Düğümler hakkındaki tüm bilgiler Linux terminal ekranına "rosnode info" yazdıktan sonra düğüm ismi yazılarak elde edilebilir. İlgili düğümün abone olduğu ve yayınladığı başlıklar görülebilir. Her bir düğüm ROS ağında mesajlar vasıtası ile veri alışverişinde bulunur.

5.2.1.2. Mesaj

Bir düğüm başka bir düğümle iletişim kurmak için ilgili başlığa mesaj yayınlar. Her mesajın bir formatı vardır. ROS'ta bu mesaj yapılarının birçoğu mevcuttur ancak yeni format da tanımlamak mükündür. Bir mesajın yapısına Linux terminal ekranına "rosmsg info" yazdıktan sonra ilgili mesajı yazarak ulaşılabilir.

Şekil 5.8'de robota gönderilecek hareket mesajı olan Twist mesajının formatı görülmektedir. Mesaj, doğrusal ve açısal hız bileşenlerinden oluşmaktadır. Robotun x, y ve z eksenleri boyunca istenen doğrusal ve açısal hız değerlerini ifade etmek için float tipinde sayılar kullanılmaktadır.



Şekil 5.8. *rosmsg komut örneği*

Mesajlar aracılığıyla düğümler arasında tüm bu veri ve bilgi alışverişi, başlıkların kullanımı sayesinde mümkündür.

5.2.1.3. Başlık (Topic)

Bir konu, bir yayın yapma/abone olma iletişim sürecini uygulayan bir mesaj akışıdır. Bir başlıktan mesaj almak isteyen düğümler, master düğüme bir istekte bulunarak o başlığa abone olabilir.

Aşağıdaki kod parçası yayın yapma/abone olma sürecini açıklamak amacıyla yazılmıştır.

Tablo 5.1. *Örnek kod parçası*

1	#! /usr/bin/env python
2	import rospy
3	from geometry_msgs .msg import Twist
4	rospy. init_node ("robot_follower")
5	sub = rospy . Subscriber (" leader/odom", Odometry , newOdom)
6	pub = rospy . Publisher (" follower/cmd_vel ", Twist , queue_size =1)

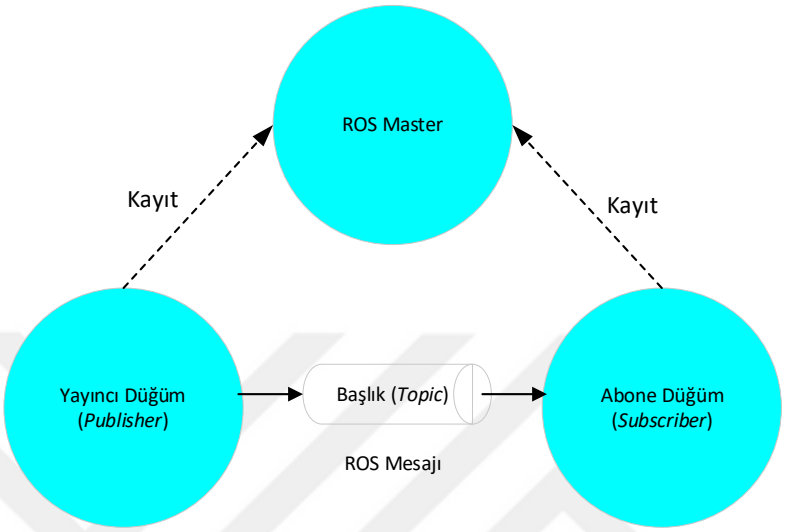
İlk satır, sonraki kod için doğru yorumlayıcının seçilmesi için gereklidir. 2. satır Python dilinde kod yazmak için gerekli kütüphane olan rospy kütüphanesini aktarmak için ve 3. satır ise robotun hız bilgisine erişmek için geometry_msgs mesajlarına ait Twist mesajını okumak için gereklidir. 4. satırda ise takip uygulaması için yeni düğüm yazılmıştır. Bu başlatma master düğüm ile iletişimi sağladığı için oldukça önemlidir. Yeni yazılan düğüm "/odom" başlığına abone olarak Odometry mesaj tipi ile takip edilen robotun konumuna erişecektir. Son olarak hız komutu "/cmd_vel" başlığı ile Twist mesaj tipinde takipçi robota gönderilecektir.

ROS'ta görüldüğü üzere tüm düğümler birbirleri ile iletişim kurabilir ancak tüm bunların olabilmesi için bütün düğümlerin her zaman iletişimde olduğu ve ilk olarak başlatılması gereken Master düğüm vardır.

5.2.1.4. ROS master

ROS platformunda her yeni düğüm oluşturulduğu zaman, bu düğümün diğer düğümlerle bağlatısını sağlamak için gereken tüm bilgiler ROS master tarafından sağlanır. ROS master düğümü "roscore" adı verilen komut çalıştırıldığında oluşur. Master düğüm oluşturulmadan hiç bir işlem yapılamaz.

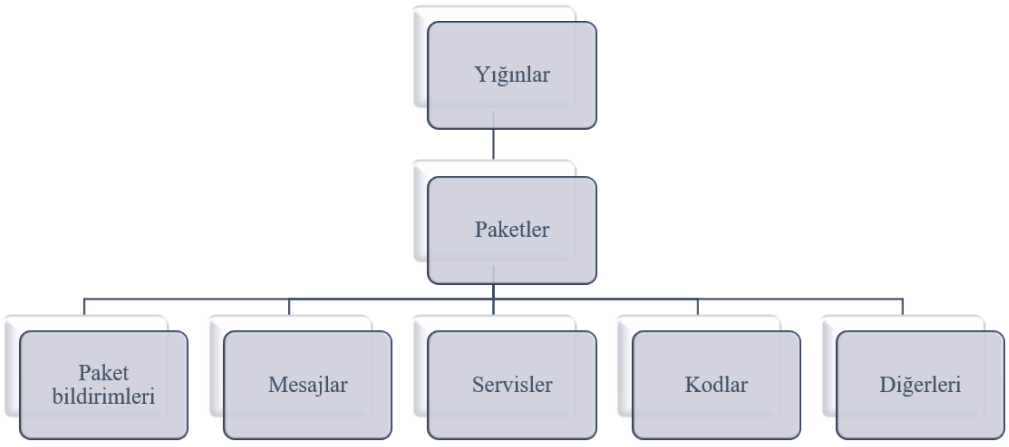
Şekil 5.9'da gösterilen örnekte düğümlerin ROS Master düğümüne kayıtları görülmektedir. Kayıt işlemi düğümün aktifleşmesini sağlar. Oluşturulan herhangi bir düğüm ilk olarak master düğüme bilgi verir, daha sonra diğer düğümlerle iletişim kurabilir.



Şekil 5.9. *Master düğümün diğer düğümler ile ilişkisi*

5.2.2. ROS dosya sistemi

ROS'un bir işletim sistemine benzer bir dosya yapısı mevcuttur. Bu yapı belirli bir düzende kaydedilir. Dosya yapısının genel şeması Şekil 5.10'daki gibidir.



Şekil 5.10. *ROS dosya sistemi*

Şemadaki blokların görevleri şöyledir (Joseph, 2015):

Paketler: ROS yazılımının en temel birimi paketlerdir. Düğümler, araç kütüphaneleri, konfigürasyon dosyaları gibi dosyaları içerirler. Belirli görevi yerine getirmek için düzenlenirler.

Paket Bildirimleri: Paketin içinde yer alan paketin yazarı, lisansı, kütüphane bağımlılıkları ve derleme bayrakları gibi verileri içeren dosyalardır. Bir paketin içinde yer alan manifest.xml dosyası o paketin bildirim dosyasıdır.

Yığınlar: Belirli amaçla yazılan paketlerin bir araya getirilmesi ile oluşturulmuş dosyalardır. Meta paketleri olarak da isimlendirilirler. ROS navigasyon yığınları örnek bir meta paketidir.

Mesajlar: Bir ROS işleminden diğerine gönderilen bilgilerdir. Mesajlar belirli fortmatta yazılarak, paketlerin içinde oluşturulmuş msg dosyalarının içine .msg uzantısı ile kaydedilir.

Servisler: ROS servisleri istek/cevap işlevini yerine getirir. İstek ve cevap veri tipleri paket dosyalarının içinde oluşturulmuş srv dosyalarının içine .srv uzantısı ile kaydedilir.

5.2.3. Mavros

Mavros paketi (Mavros Wiki, 2018), MAVLink protokolünü kullanan farklı tip otopilotlarla iletişim kurmaya sağlayan ROS paketidir.

MAVLink, hava araçlarında en yaygın kullanılan iletişim protokollerinden biridir. Ticarileştirilmiş İHA'ların çoğu halihazırda MAVLink protokolünü kullanmaktadır. Hem yer kontrol istasyonu (Ground Control Station-GCS) ile insansız araçlar arasındaki iletişim için hem de araçların kendi arasındaki iletişimi için kullanılır. USB veya telemetri ile çalışır. Komut gönderme, parameter ayarlama, yön, GPS konumu, hız gibi birçok bilginin iletilmesi için bir dizi mesaj gönderir. Mavros ise ROS ile MAVLink arasında köprü görevi gören ROS paketidir.

5.3. Gazebo

Gazebo (Open Source Robotics Foundation, 2014), robotik uygulamalar için geliştirilmiş ve robotlar ile diğer nesneler arasında fiziksel iletişim sağlayan, sensör geribildirimine imkan veren, üç boyutlu benzetim ortamıdır. Ücretsizdir ve oldukça büyük bir topluluk tarafından desteklenmektedir. ROS'un ortaya çıkması ile birlikte ROS

ve Gazebo'yu birlikte kullanmayı sağlayan ROS paketi geliştirilmiştir ve Gazebo cazip özellikleri ile ROS geliştiricileri tarafından kullanılan en önemli araçlardan biri haline gelmiştir.

Gazebo, gerçek dünyayı oldukça iyi yansıtan, yüksek kaliteli grafikler ve arayüzlerden oluşur. İhtiyaca göre tasarım yapmak ve var olan tasarımları özelleştirmek mümkündür. Bu sayede robotları doğru ve verimli bir şekilde simüle etmeye imkan sağlar. Oluşturulmak istenen ortamdaki tüm unsurları tanımlamak ve konumlandırmak için "world" adı verilen dosyalar kullanılır. Model dosyaları ile de her türlü robot modellenebilir.

Gazebo, sunucu/istemci ilişkisi ile çalışır. Sunucu, benzetim ortamını verilen özelliklere göre oluşturur, dinamikleri simüle eder, istemci ise sunucuya bağlanarak görselleştirme işlevini yerine getirir.

Gerçek dünyada gürültüsüz veriye ulaşmak mümkün değildir. Yapılacak çalışmanın fiziksel ortamda testini yapmadan önce güçlü bir benzetim ortamında test etmek çalışmanın verimliliği açısından oldukça önemlidir. Gazebo sensörlere gürültü eklemeye de imkan sağlar. Güçlü sensör ve eklenti desteği, kütüphaneleri, mimarisi ve işlevselliği ile Gazebo diğer simulasyon ortamlarına göre oldukça verimlidir.

5.4. ArduPilot

ArduPilot (ArduPilot Dev Team, 2020), multikopter, sabit kanatlı uçak, helikopter, yer robotu gibi her türlü otonom araçta kullanılabilen, aracı kontrol etmeyi sağlayan açık kaynak kodlu otopilot yazılımıdır. ArduPilot versiyonu kullanılan araç türüne göre farklılık göstermektedir. Uçaklar için ArduPlane, multikopterler için ArduCopter, yer robotları için ArduRover, sualtı araçları için ise ArduSub geliştirilmiştir.

Tez kapsamında ArduPilot'un multikopterler için olan versiyonu ArduCopter kullanılmıştır. ArduCopter, sensörlerden yararlanan bir kontrol sistemi içerir ve kalkış, iniş, dengede durma ve belirli bir yükseklikteki noktaya havalanma gibi otomatik manevraları yerine getirme yetisine sahiptir. Bu sistem, hava aracının hareketi için belirli parametrelerin alımına ve istenilen amaç doğrultusunda kullanılması için birçok sensörün kullanılmasına imkân sağlar. ArduCopter yazılımı sayesinde kullanılan hava aracını manuel ya da otomatik kumanda etmek mümkündür.

Bu özelliklerin yanı sıra, yazılım, hava aracıyla farklı yörünge ve uçuşların denenebileceği birçok uçuş modunu içerir.

Temel uçuş modları aşağıdaki gibidir (ArduPilot Dev Team, 2020):

Stabilize: Hava aracını manuel uçurmaya imkân sağlar.

ALT_Hold: Sabit irtifada yatış, yunuslama, yalpa kontrolüne imkân sağlar.

Loiter: Mevcut konum, baş açısı ve irtifanın sabit tutulmasını sağlar.

RTL (Return to Launch): Hava aracının başlangıç konumuna dönmesini sağlar.

Auto: Hava aracının otopilottaki önceden planlanmış görevi yerine getirme modudur.

Bu beş temel modun dışında Acro, Sport, Drift, Guided, Circle, Position, Land, Follow Me, Simple ve Super Simple gibi ikincil modlar da mevcuttur.

Ek olarak, yazılım tamamen açık kaynak olduğundan önceden programlanmış parametreler ve konfigürasyonlar kullanıcı isteği doğrultusunda değiştirilebilir ve geliştirilebilir.

Yazılımın bir diğer avantajı ise APM Planner, Mission Planner, QGround Control gibi birçok yer kontrol istasyonu programı gibi ile uyumlu çalışmasıdır. Aynı zamanda bu programlar, sensörlerin farklı değerlerinin, parametrelerin ve hava aracı konfigürasyonlarının farklı tipte görselleştirilmelerine imkân sağlar.

Tez çalışması kapsamında, ArduCopter için SITL (Software in the Loop) simülatörü kullanılacaktır.

5.5. Döngüde Yazılım Simülatörü (Software in the Loop-SITL)

Bu simülatör fiziksel sistem olmadan hava aracının davranışını test etmeye imkan sağlar. Yazılan otopilot kodunun fiziksel sistemde nasıl bir davranışa neden olacağını gösterir. Kısaca, SITL aracılığıyla ArduPilot doğrudan bir bilgisayarda çalıştırılabilir. Simulatörde uçuş dinamiğine uygun sensör verileri okunabilir. Hâlihazırda birçok araç modeli barındıran Ardupilot'a bu simülatörde yeni sensörler ve araçlar eklenebilir.

Otopilotun sahip olduğu dahili sensörler Ataletsel Ölçüm Birimi (Inertial Measurement Unit-IMU), barometre, pusuladır. IMU, ivmeölçer, jiroskop ve manyetometreden oluşur. Böylece 3 eksende yön, ivme ve pusula bilgisi sağlar.

Barometre ise basınç değişiminden irtifa bilgisini verir. Hava aracının baş açısı (heading) bilgisi ise pusula (compass) kullanılarak belirlenir. Pusula, Dünya'nın manyetik alanına göre hava aracının hangi baş açısıyla uçtuğunu gösterir.

Böyle bir simulatör kullanımının bir diğer avantajı da yazılan koddaki hataların etkileşimli ve dinamik bir şekilde ayıklanmasını sağlaması, yeni özellikler geliştirme ve test etmeyi kolaylaştırmasıdır.

5.6. Sistem Mimarisi

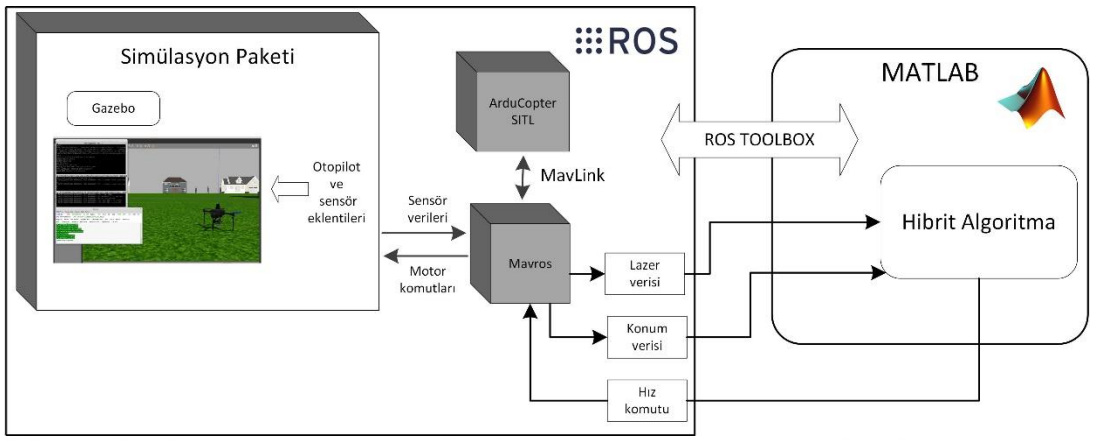
Bu bölümde yazılım araçlarının birbirleriyle ilişkisi şematik olarak açıklanacaktır.

Bir önceki alt bölümde bahsedilen otopilot yazılımını kullanan hava aracını, iki yolla kontrol etmek mümkündür. Bunlar: RC (Radio Control) kumanda veya yer control istasyonudur. Yer kontrol istasyonu ve RC kumanda cihaza MAVLink protokolünü kullanarak RC sinyaller gönderirler. Bu da hava aracının tamamen kontrol edilebilir olmasına izin verir ve bu protokol sayesinde batarya durumu, irtifa, GPS verisi, hız gibi çeşitli sensör verilerinin iletimi sağlanır. Simülasyonda kullanılan Mavros paketi bu noktada önemlidir. Bu paket MAVLink haberleşme protokolünü kullanır. Bu protokol sayesinde ROS hava aracına çeşitli kumandalar gönderebilir. Sonuç olarak, hava aracı seyrüseferi için planlanan ve oluşturulan algoritma doğrudan uçuş denetleyicisine Mavros paketi vasıtasıyla gönderilir.

Şekil 5.11'de görüldüğü üzere ROS'ta farklı görevleri yerine getiren paketler oluşturulmuştur. Simülasyon paketinde Gazebo'yu çalıştıran, hava aracının modelini içeren dosya, uçuş kontrol birimi (Flight Control Unit – FCU) otopilot yazılımı ve sensörlerin simülasyon eklentileri bulunmaktadır. Otopilot yazılımını çalıştıran ROS paketi ArduCopter SITL'dir. Mavros paketi, yukarıda da belirtildiği üzere, otopilot yazılımı ile iletişimden sorumludur. Hava aracının Gazebo modelinden alınan sensör verileri ArduCopter uçuş kontrol biriminde değerlendirilir ve filtrelenmiş sensör verileri ROS başlıkları ile okunabilir. Hava aracı kontrolünde istenen motor komutları kullanıcı tarafından uçuş kontrol birimine gönderilir ve uygun olduğu takdirde Gazebo ortamındaki hava aracına iletilir. Mavros paketi vasıtası ile her türlü uçuş verisi (batarya durumu, hava aracının konumu, motorların durumu gibi) okunabilmektedir.

ROS platformundan alınan hava aracına ilişkin sensör ve konum verileri ROS Toolbox vasıtası ile MATLAB'te yazılan hibrit algoritma tarafından alınmaktadır. Hibrit

algoritma bu verilere uygun yol planı yapmak için Gazebo ortamındaki hava aracına hız komutu göndermektedir.



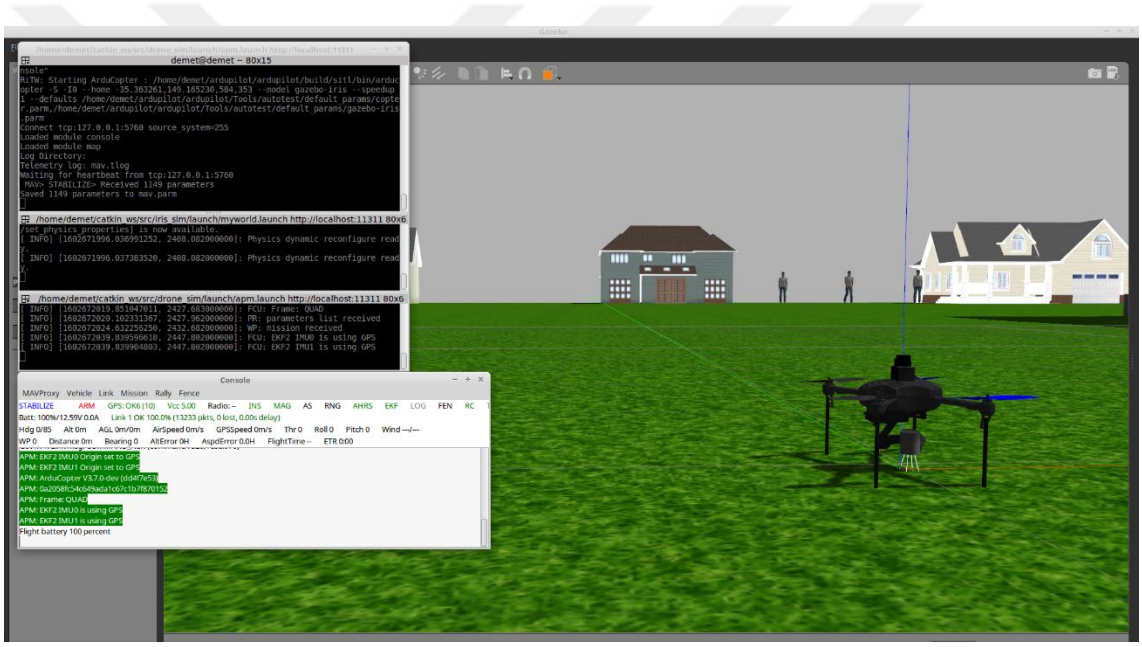
Şekil 5.11. *Sistem mimarisi*

Tüm bu eklentiler ile oluşturulan algoritmada hava aracı tepkisini gözlemek mümkün olmuştur. Şemada bulunan ROS paketleri, ancak bir Linux tabanlı işletim sisteminde çalıştırılabilir. Tez kapsamında kullanılacak olan Linux dağıtımı Ubuntu 16.04'tür. Bu sisteme uyumlu olarak, ROS Kinetic ve Gazebo 7 versiyonlarıyla çalışılmıştır.

6. ÖN ÇALIŞMA

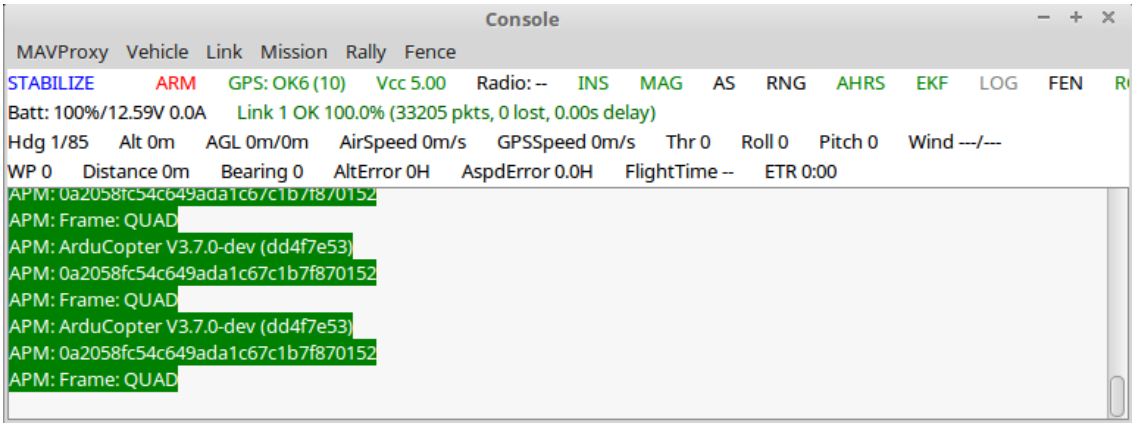
Bu bölümde yol planlama algoritma çalışmalarının öncesinde hava aracının Gazebo ortamında uçuş denemeleri yapılmış ve hava aracının en temel engelden sakınma yöntemlerinden biri olan potansiyel alanlar yöntemi ile engelden sakınması incelenmiştir.

Test ortamı, Şekil 6.1'deki gibi Gazebo fiziksel benzetim ortamı kullanılarak oluşturulmuştur. Gazebo benzetim ortamı gerçekleme ve sensör desteği bakımından diğer benzetim ortamlarına göre oldukça başarılıdır ve benzetim ortamı, gerçek ROS modüllerini kullanır, böylece hava aracına verilen komut ve sistemin çalışmasına güvenilebilir. Simülasyonda kullanılan hava aracı bir önceki bölümde de bahsedilen 3DR firmasının geliştirdiği Iris+'dır.



Şekil 6.1. *Örnek Gazebo ortamı*

Hava aracı Mavros komutları kullanılarak kontrol edilir. Mavros paketi komuta merkezi (yer kontrol istasyonu) ile hava aracı arasında bir MAVLink iletişim köprüsü sağlar. Aynı zamanda, hava aracının çalışması, bağlantıların çalışması ve hava aracı ve ilgili sensörlerin durumu hakkında uçuş açısından kritik bilgiler sağlayan Şekil 6.2'deki konsol görünümünden görüntülenebilir.



Şekil 6.2. *Mavros konsol görünümü*

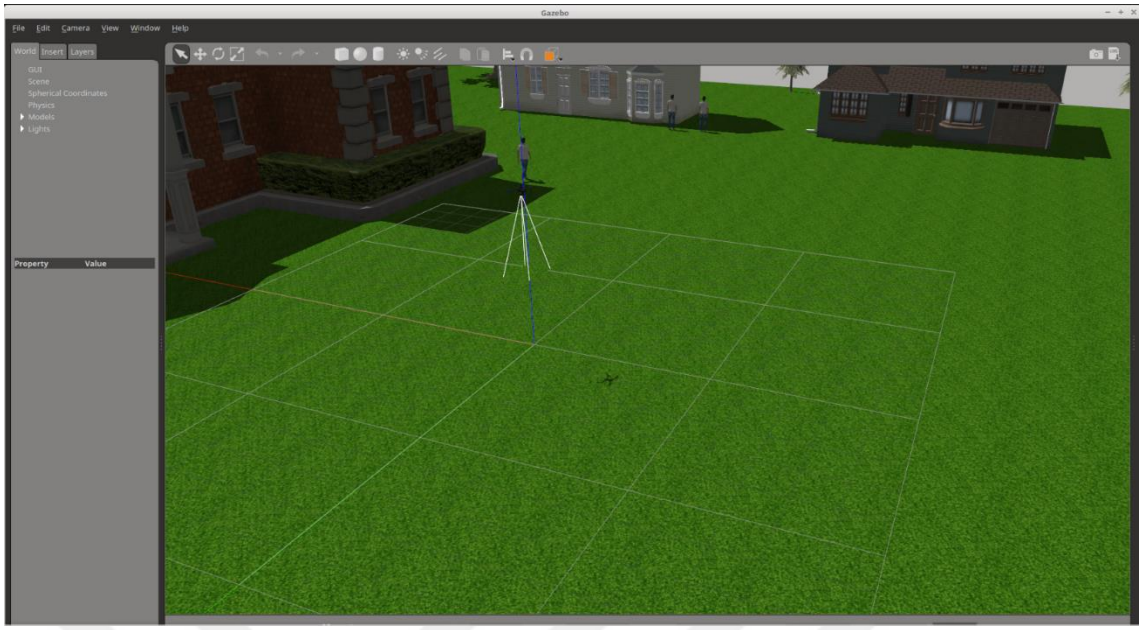
Şekil 6.2'de herhangi bir an için hava aracınının durumu görülmektedir. Buradan motorların durumu, hava aracının uçuş modu, GPS uydularının kaç tanesine bağlanıldığı, batarya durumu gibi uçuşa ait bilgiler takip edilebilir.

6.1. Test Aşamaları

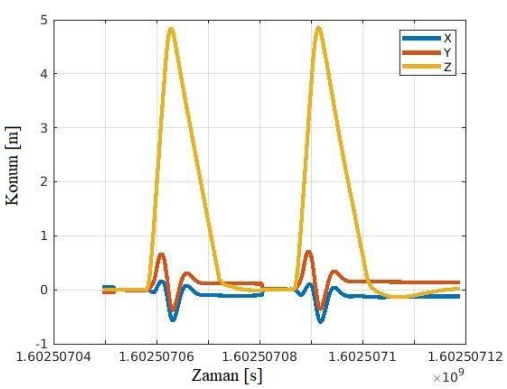
Bu bölümde hava aracının kalkış, iniş, belirli yol noktalarına gidiş komutlarına otopilot yazılımı (ArduPilot) vasıtası ile nasıl tepki verdiği incelenmiştir. Sonrasında engelden sakınma potansiyel alanlar yöntemi ile test edilmiştir.

6.1.2. Otomatik kalkış ve iniş

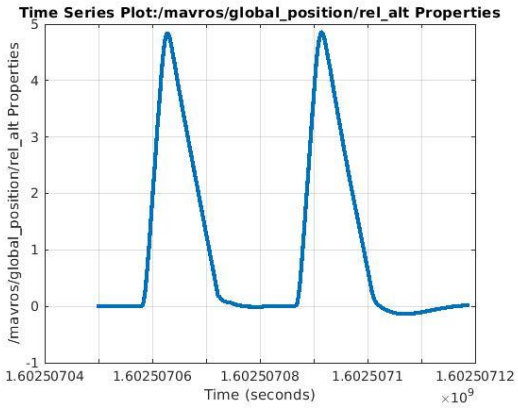
İlk aşama olarak, hava aracının otomatik olarak kalkış ve iniş yapması sağlanmıştır. Araç, 5 metreye yükselmiş ve inmiştir. Bu esnada hava aracının hareketi Şekil 6.3'te, konum grafikleri ise Şekil 6.4 ve Şekil 6.5'te verilmiştir.



Şekil 6.3. *Otomatik kalkış sırasında hava aracının Gazebo ortamında görünümü*



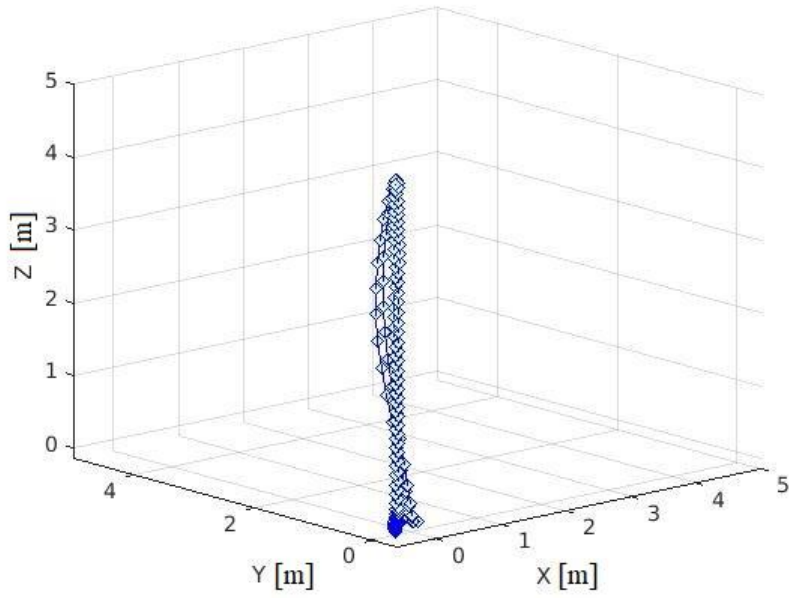
a) *3 eksende konum değişimi*



b) *İrtifa değişimi*

Şekil 6.4. *Konum değişimleri*

Şekil 6.3'te hava aracının Gazebo ortamında yükseldiği anlar görülmektedir. Şekil 6.4'te ise yükselme esnasında 3 eksende (X, Y, Z) meydana gelen konum değişimleri görülmektedir. Hava aracına yaklaşık 12 saniye süren yükselme ve inişten sonra tekrar yükselme komutu gönderilmiştir ve aynı tepki alınmıştır. X ve Y eksenlerinde kalkış esnasında maksimum 0,5 metrelik sapma görülmektedir. Bu durum olağandır ve hava aracı bu eksenlerdeki sapmayı kısa sürede (yaklaşık 6 saniye) toparlamıştır.

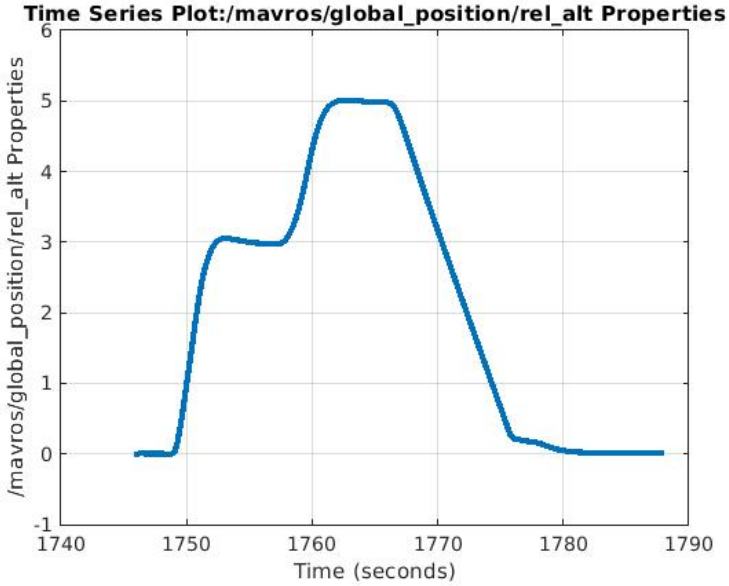


Şekil 6.5. *3 boyutta konum değişimi*

Şekil 6.5'te ise yükselme hareketi üç boyutlu grafikle gösterilmiştir.

6.1.3. Kalkış-aşamalı yükselme-iniş

Bu aşamada ise hava aracının otomatik kalkışı sağlanmıştır. Öncelikle 3 metreye yükselerek 5 saniye konumunu koruması beklenmiştir. Daha sonra 5 metrede de 5 saniye konumunu nasıl koruduğu test edilerek iniş yapması sağlanmıştır. Bu esnada hava aracının konum değişim grafiği Şekil 6.6'daki gibidir.

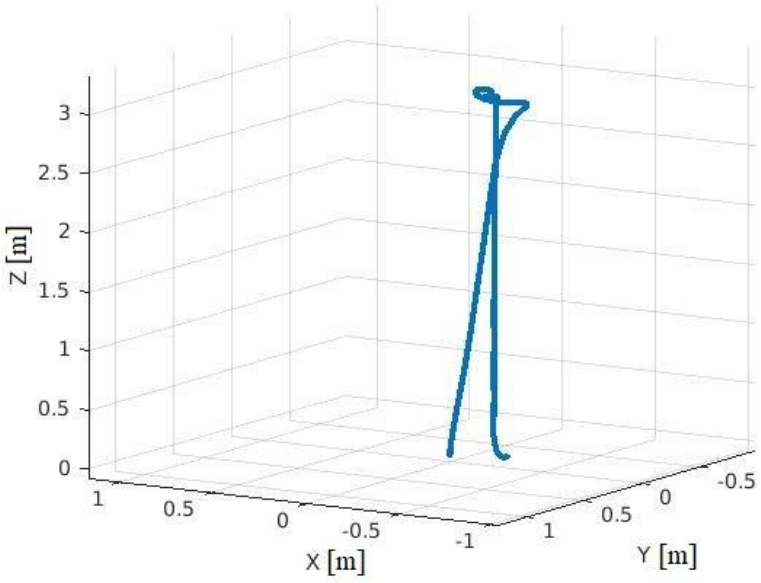


Şekil 6.6. *Aşamalı kalkış ve inişte irtifa değişimi*

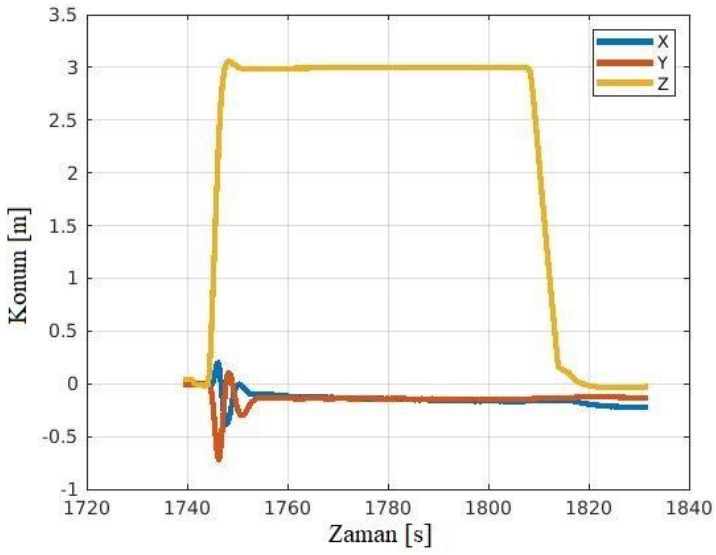
Şekil 6.6'da kalkış, aşamalı yükselme ve iniş aşamalarının 30 saniyelik bir zaman diliminde gerçekleştiği gözlemlenmektedir.

6.1.3. Konum koruma

Bu aşamada hava aracının otomatik kalkış yaparak 3 metrede 60 saniye kalması sağlanmıştır. Elde edilen konum grafikleri, 3 boyutta Şekil 6.7'de, 3 ayrı eksende ise Şekil 6.8'de verilmiştir.



Şekil 6.7. *3 boyutta konum değişimi*

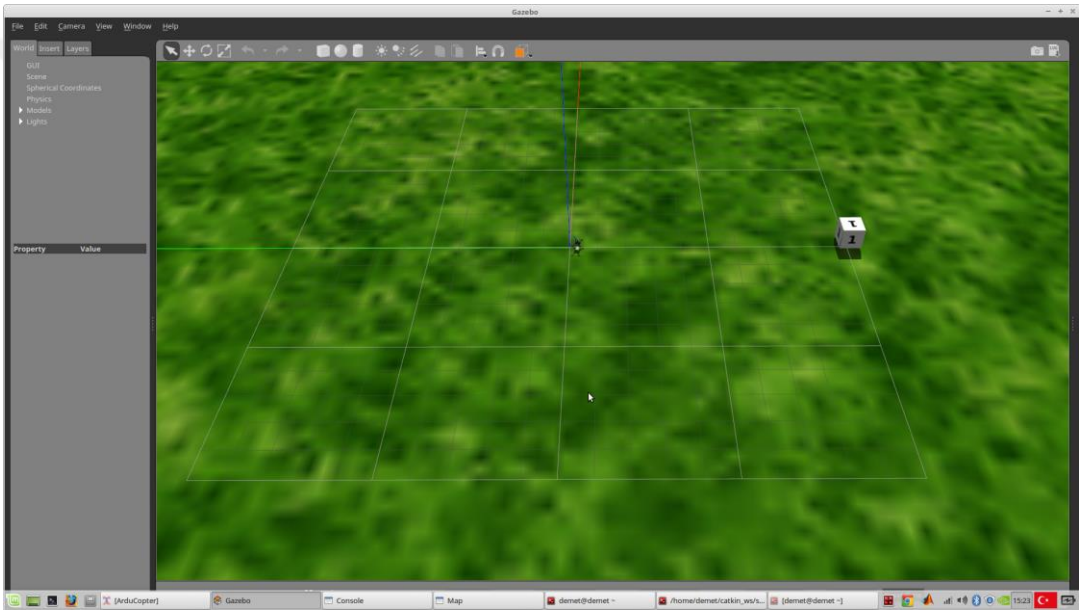


Şekil 6.8. *3 eksende konum değişimi*

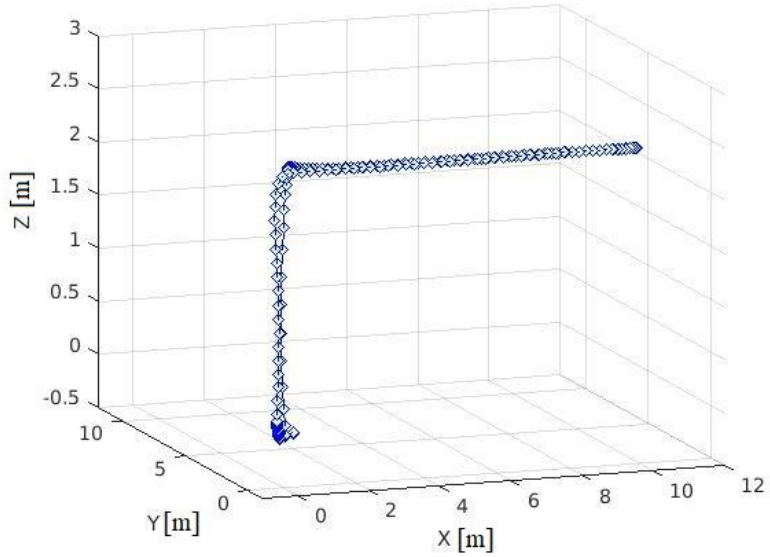
Şekil 6.8'de kalkış, belirli konumda 60 saniye kalma ve iniş aşamalarının yaklaşık 72 saniye sürdüğü gözlenmektedir. Hava aracı kalkış aşamasında X ve Y eksenlerinden bir miktar sapmıştır fakat 5 saniye gibi kısa bir zamada bu eksenlerdeki hareketini toparlamıştır.

6.1.4. Bir noktadan başka bir noktaya gidiş

Hava aracının Şekil 6.9'daki örnek Gazebo ortamında bir noktadan başlayarak başka bir (1 numaralı kutucuk) noktaya ulaşması, daha sonra başlangıç noktasına gelerek iniş yapması sağlanmıştır. Bu esnada hava aracının konumunda meydana gelen değişimler Şekil 6.10'da verilmiştir.



Şekil 6.9. *Örnek Gazebo ortamı*



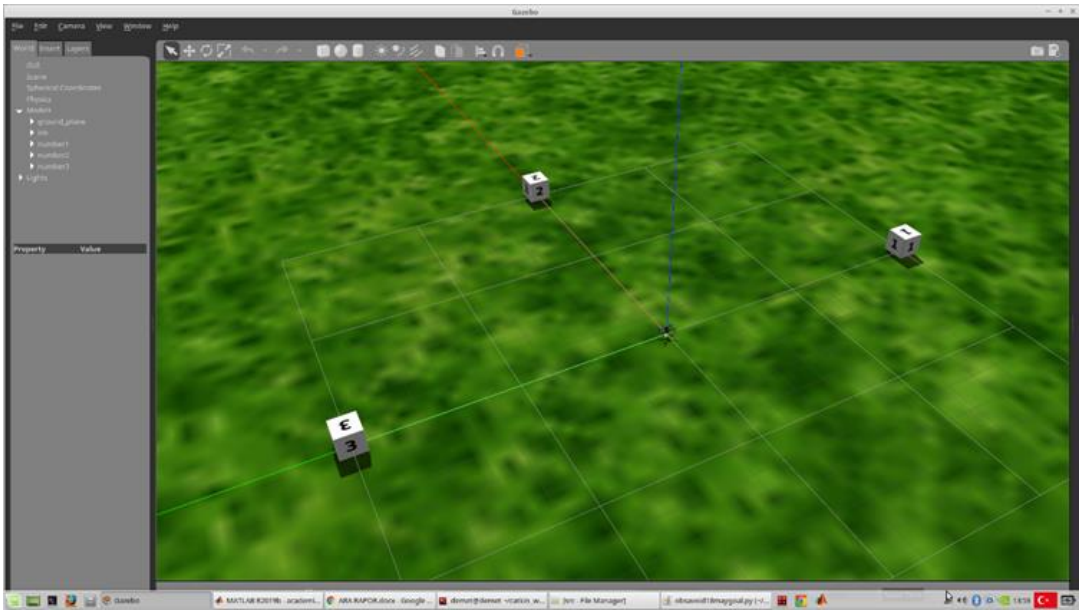
Şekil 6.10. *3 boyutta konum değişimi*

Şekil 6.10'da 1 numaralı hedefin bulunduğu konumun üzerine gelerek aynı irtifadan dönüş yapması esnasında hava aracının hareketi 3 boyutlu grafik ile gösterilmiştir.

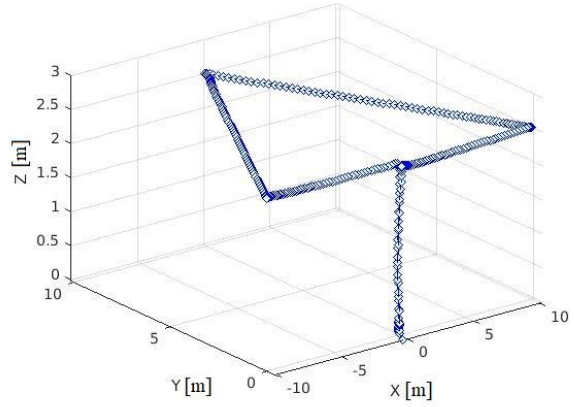
6.1.5. Üç farklı yol noktasına gidiş

Bir diğer senaryoda hava aracı Şekil 6.11'deki gibi bir ortamdadır. Araç, üç farklı hedefe, sırasıyla 1, 2 ve 3 numaralı kutulara gittikten sonra başlangıç konumuna iniş yapmaktadır.

Bu esnada hava aracının izlediği rota Şekil 6.12'de verilmiştir.



Şekil 6.11. *Örnek Gazebo ortamı*



Şekil 6.12. *3 boyutta konum değişimi*

Şekil 6.12'de hava aracı sırası ile 1, 2 ve 3 numaralı kutuların bulunduğu konumun üzerine giderek aynı irtifadan başlangıç konumuna dönmesi esnasında hareketi 3 boyutlu grafik ile gösterilmiştir.

6.1.6. Potansiyel alanlar yöntemi ile engelden sakınma

Bu bölümde hava aracının çevresindeki engellerden sakınmasını sağlamak amacıyla Bölüm 2.1'de teorisi verilen potansiyel alanlar yöntemi test edilmiştir.

Hava aracının üzerinde otopilotun sağladığı sensör verileri dışında çevresindeki engelleri algılayabilmesi için lidar sensörü eklenmiştir.

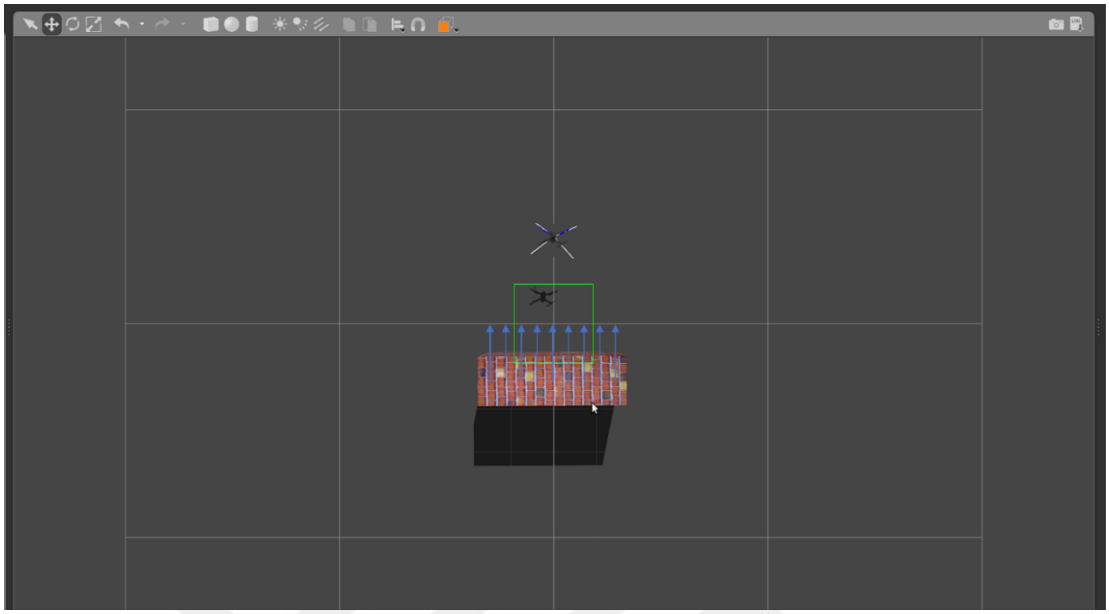
Lidar (Light Detection and Ranging), ultrasonik sensör ile benzer çalışma prensibine sahip bir mesafe sensörüdür. Tek fark ses dalgaları yerine ışık kullanmasıdır.

Bu tez çalışmasında hava aracının karşılaştığı engelleri algılaması için Hokuyo UTM-30LX model lidar kullanılmıştır. Tablo 6.1'de özellikleri verilen lidarın eklentisi Gazebo ortamındaki hava aracı modeline eklenmiştir. Sensör ölçümleri polar koordinatlarda (açı-mesafe) ilgili ROS başlığından okunmaktadır.

Tablo 6.1. *Hokuyo UTM-30LX sensör özellikleri (Hokuyo Automatic Co. Ltd, 2012)*

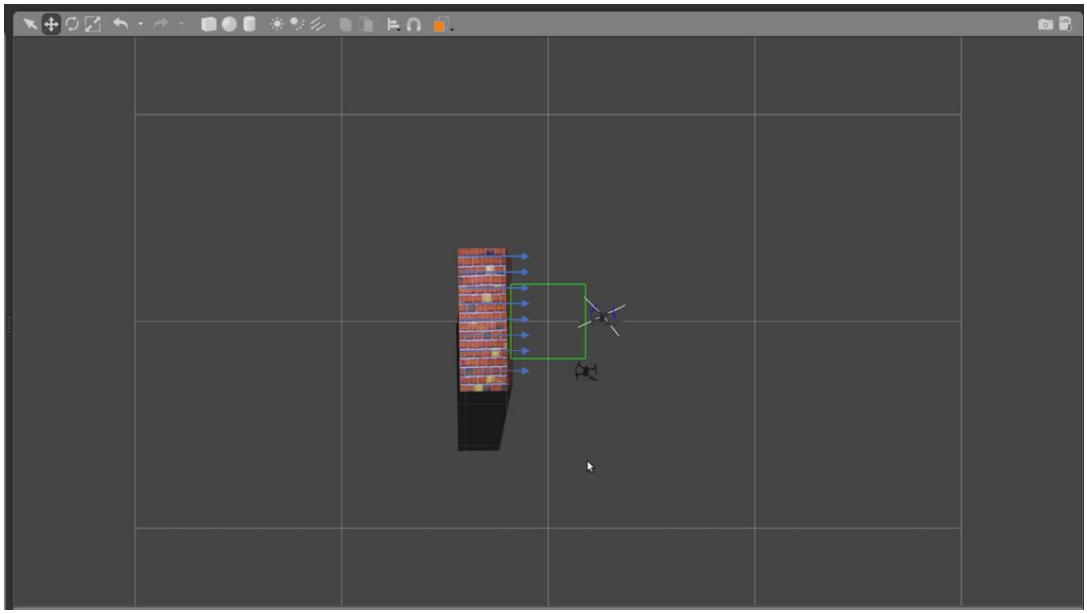
Parametre	**Değer**
Verimli ölçüm mesafesi	0.1-30m
Hassasiyet	0.1-10m: ±30mm
Tarama hızı	25ms/tarama
Tarama açısı	270°
Açısal çözünürlük	0.25°

Örnek bir ortamda hava aracına etkiyen kuvvetler Şekil 6.13 ve Şekil 6.14'te verilmiştir.



Şekil 6.13. *Engelin uyguladığı itici kuvvetler (1)*

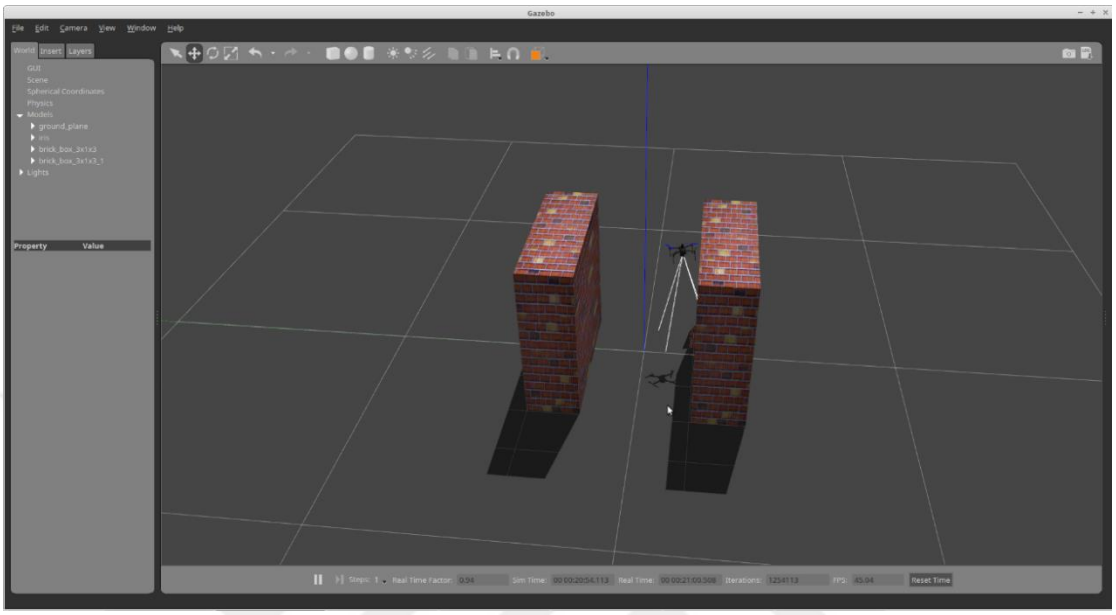
Şekil 6.13'te hava aracı yeşil renkli karenin merkezinde iken yaklaşan bir duvardan sakınmak için öne doğru hareket etmiştir. Bu hareketin sebebi ise duvarın uyguladığı itme kuvvetidir.



Şekil 6.14. *Engelin uyguladığı itici kuvvetler (2)*

Şekil 6.14'te ise hava aracı sabit bir konumda asılı duruyor iken soldan yaklaşan engelin oluşturduğu itme kuvvetinin etkisi ile saga doğru hareket etmiştir.

Potansiyel alanlar yönteminin hava aracı için testinde yöntemin teorisinde bahsedilen problemlerle karşılaşılmıştır. Şekil 6.15'te bu durum iki engelin arasında kalan hava aracı için gözlenmiştir.



Şekil 6.15. *İki engel arasında kalan hava aracının görünümü*

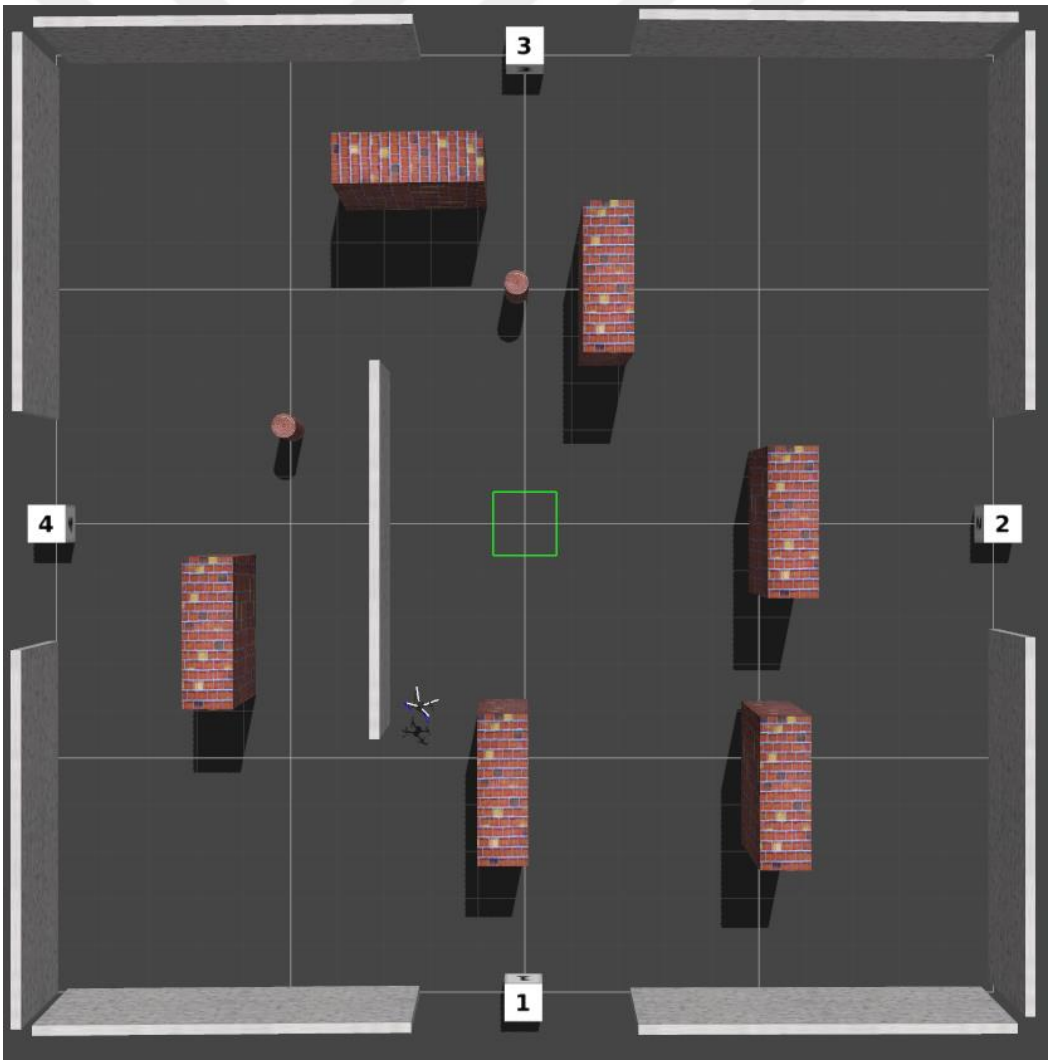
Şekil 6.15'te hava aracı iki engelin de oluşturmuş olduğu itme kuvvetinin etkisi ile osilasyona uğradıktan sonra engellere çarparak kırıma uğramıştır.

7. BULGULAR

Bu bölümde tez kapsamında incelenen statik ve dinamik ortamlarda yol planlama problemine ilişkin sonuçlar verilmiştir. İlk olarak sadece Vektör Alan Histogramı kullanılarak elde edilen sonuçlar incelenmiştir. Sonrasında A* ve VFH+ algoritmalarının birlikte kullanıldığı hibrit yöntem farklı ortamlarda test edilerek oluşturulan yol planları verilmiştir. Hibrit yöntemin oluşturulan yol planı ve süre açısından başarımı ise ROS'un navigasyon yığınları kullanılarak değerlendirilmiştir.

7.1. Vektör Alan Histogramı Algoritması ile Engelden Sakınma

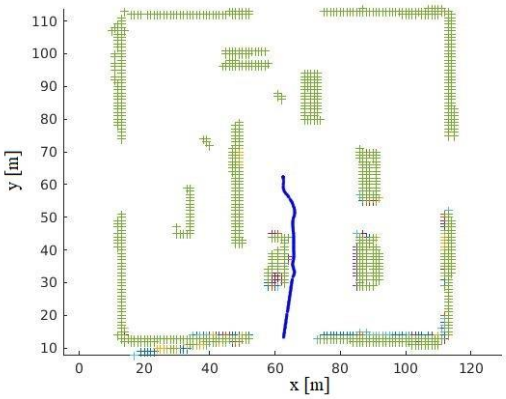
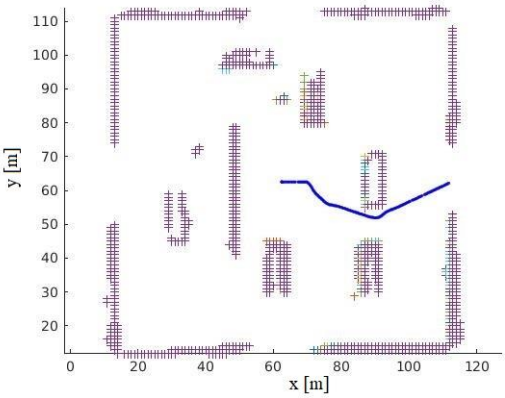
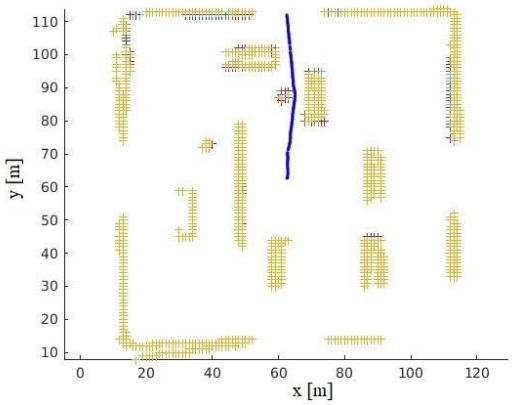
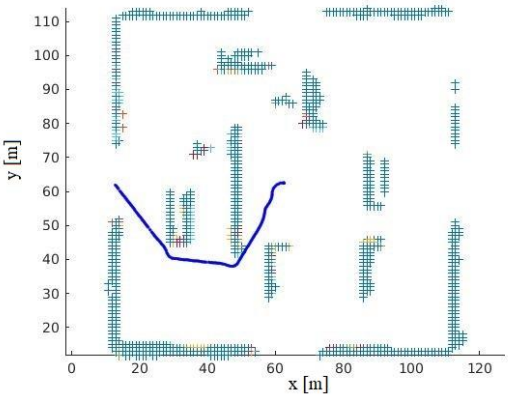
Lokal bir engelden sakınma algoritması olan Vektör Alan Histogramı ile hava aracının Şekil 7.1'de görülen 4 farklı konumda bulunan kutucuklara ulaşması Gazebo ortamında statik ve dinamik ortam oluşturularak analiz edilmiştir.



Şekil 7.1. *Örnek Gazebo ortamı*

Hava aracının başlangıç konumundan ayrı ayrı 1, 2, 3 ve 4 numaralı hedeflere ulaşmaya çalışırken izlediği yollar Tablo 7.1'de görüldüğü gibidir.

Tablo 7.1. *VFH+ algoritması ile 4 farklı hedefe gidiş yolları*

	
1 nolu hedef için izlenen yol	2 nolu hedef için gidilen yol
	
3 nolu hedef için gidilen yol	4 nolu hedef için gidilen yol

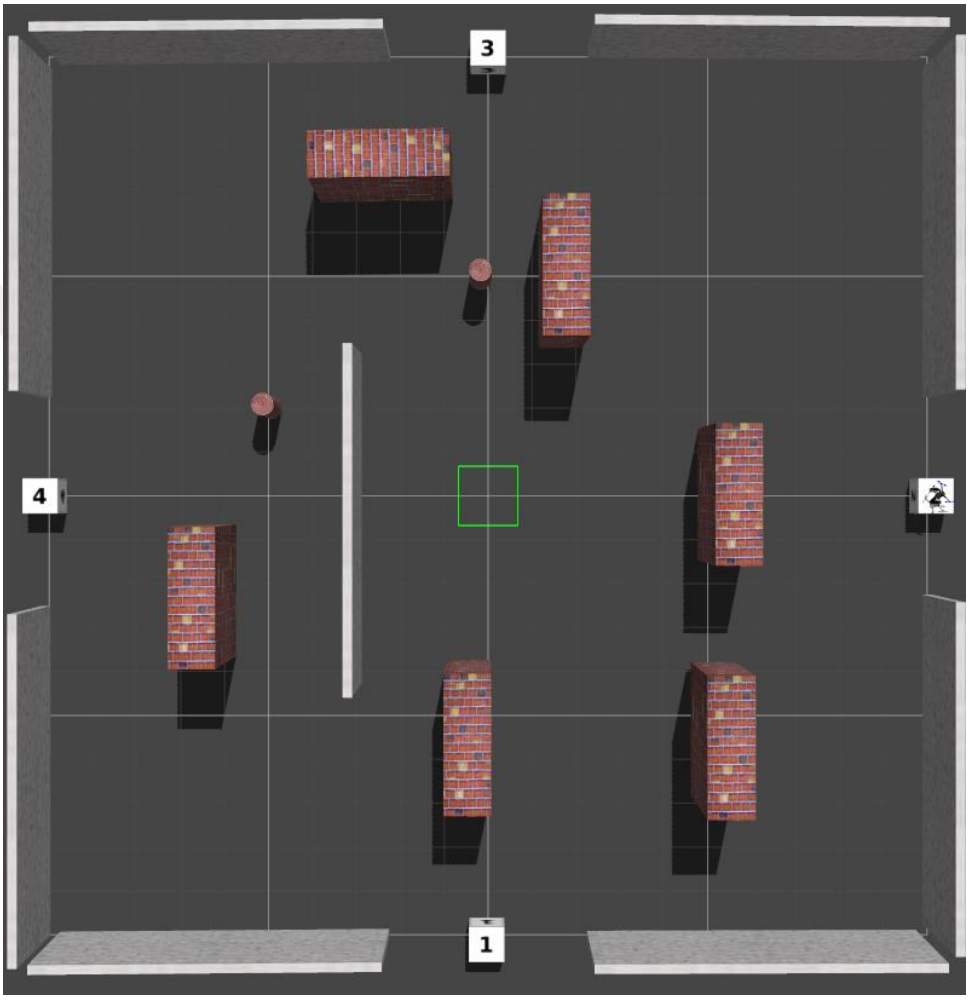
Her bir hedefe olan Öklid mesafesi 10 metredir. Yukarıdaki tabloda grafikler 5:1 ölçeğinde çizdirilmiştir. Hava aracının engele olan mesafesi 0,4 metre ile sınırlandırılmıştır. Hava aracının hedefe ulaşma süreleri ise Tablo 7.2'de verilmiştir.

Tablo 7.2. *Hava aracının hedefe ulaşma süresi*

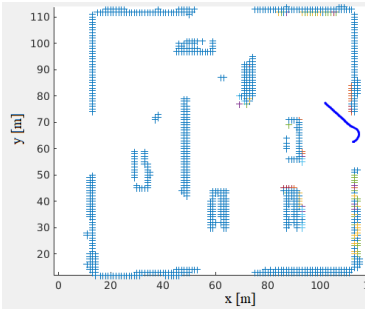
1 nolu hedef	29,7852s
2 nolu hedef	32,2520s
3 nolu hedef	29,9496s
4 nolu hedef	43,6040s

Algoritmanın dinamik ortam başarımı ise yine aynı ortamda hava aracı hedefine doğru ilerlerken engellerin yerinde değişiklik yapılarak elde edilmiştir.

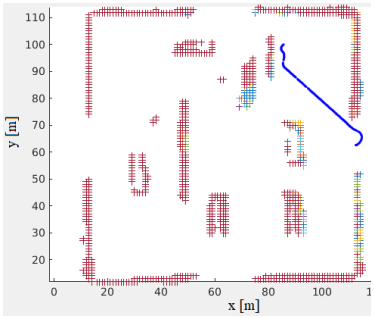
Şekil 7.2'de görüldüğü üzere hava aracı başlangıçta 2 nolu kutucuğun üzerindedir. 3 nolu hedefe giderken ortama yeni eklenen engel sonucunda hava aracının seçtiği yol sırasıyla Şekil 7.3, Şekil 7.4 ve Şekil 7.5'te gösterilmiştir.



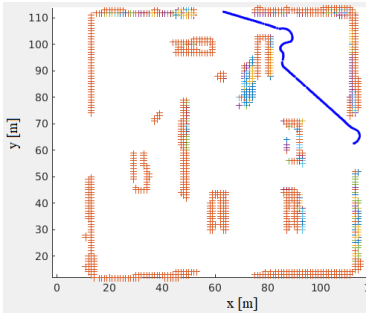
Şekil 7.2. *Örnek Gazebo ortamı*



Şekil 7.3. *Aracın hareketine başlaması*



Şekil 7.4. *Ortama yeni eklenen engele göre aracın hareketi*

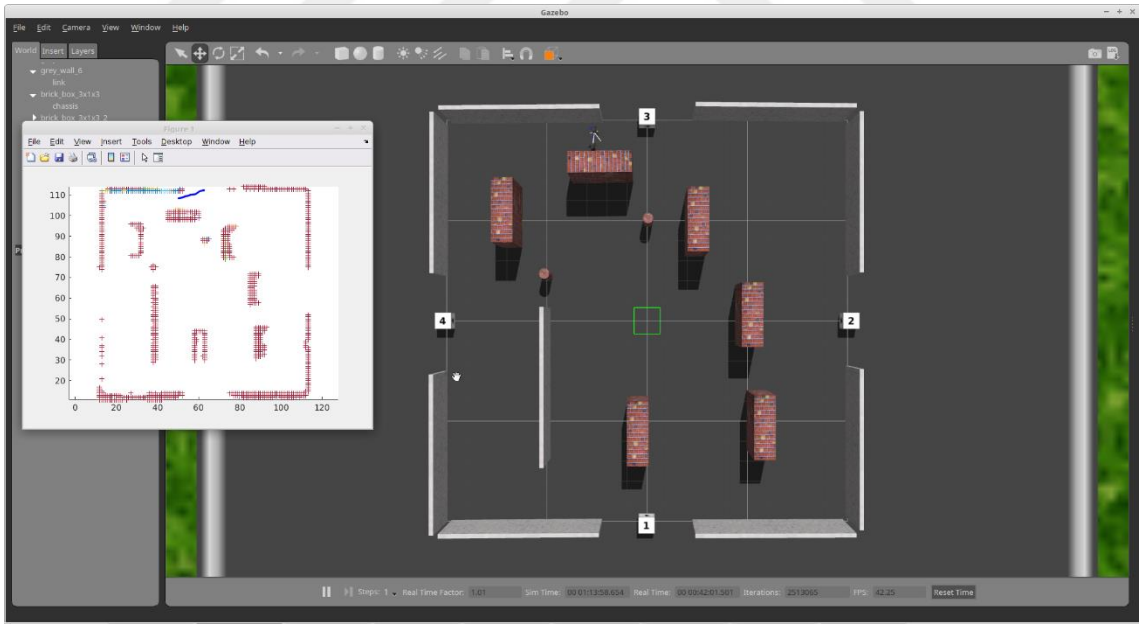


Şekil 7.5. *Aracın izlediği nihai yol*

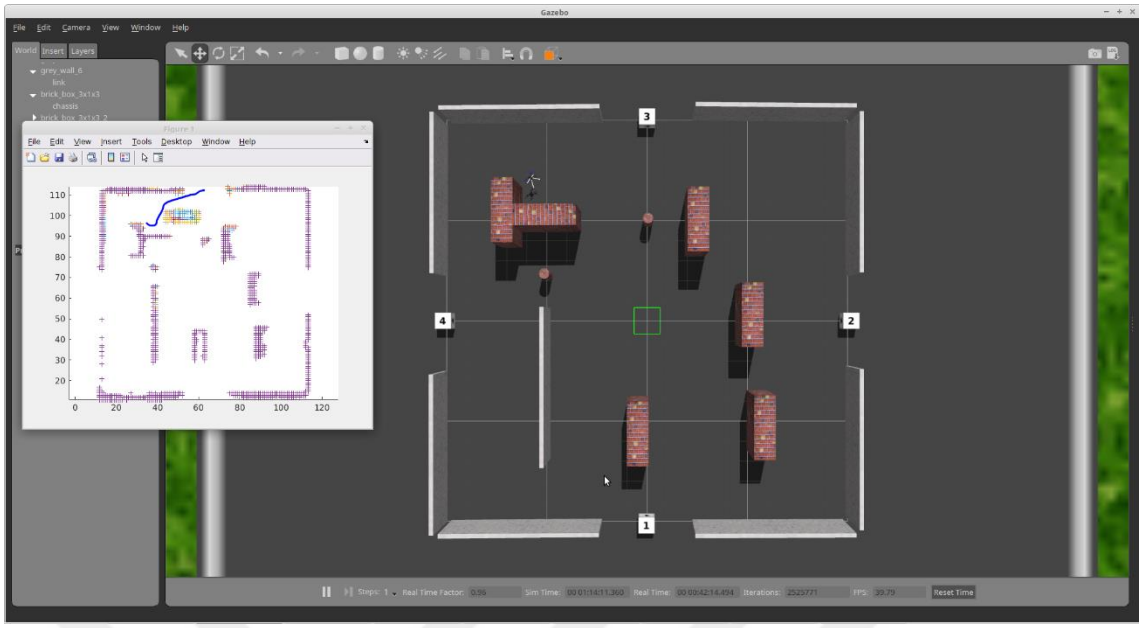
Hava aracı Şekil 7.3'teki ortamda hareketine engelsiz bir şekilde devam edecek iken ortama yeni eklenen engel sonucunda hareketini Şekil 7.5'teki gibi tamamlamıştır.

İlk ortamda hedefine ulaşma süresi 42,2113 saniye iken, ortama yeni eklenen engel ile birlikte bu süre 48,7760 saniyeye çıkmıştır.

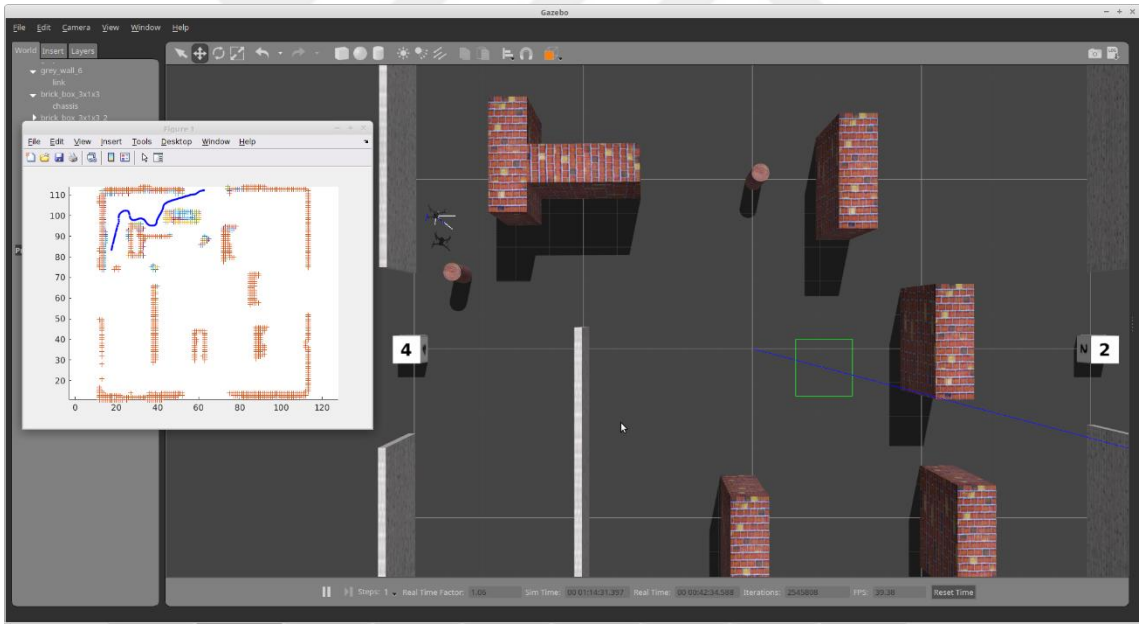
Aynı ortamda 3 numaradan hareketine başlayan hava aracı için dinamik ortam oluşturduğumuzda hava aracının izlediği yollar sırasıyla Şekil 7.6, Şekil 7.7 ve Şekil 7.8'deki gibi olacaktır.



Şekil 7.6. *Hava aracının 4 numaralı hedefe doğru yönelmesi (1. adım)*



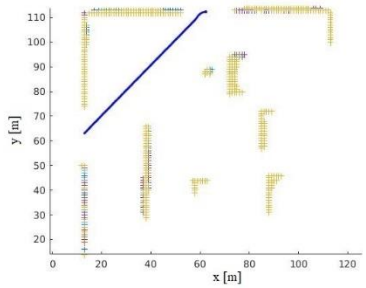
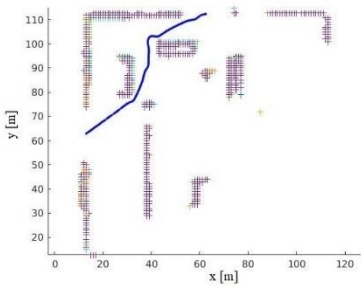
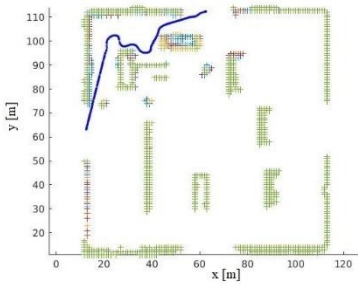
Şekil 7.7. *Ortama eklenen yeni engellere göre hava aracının hareketi (2. adım)*



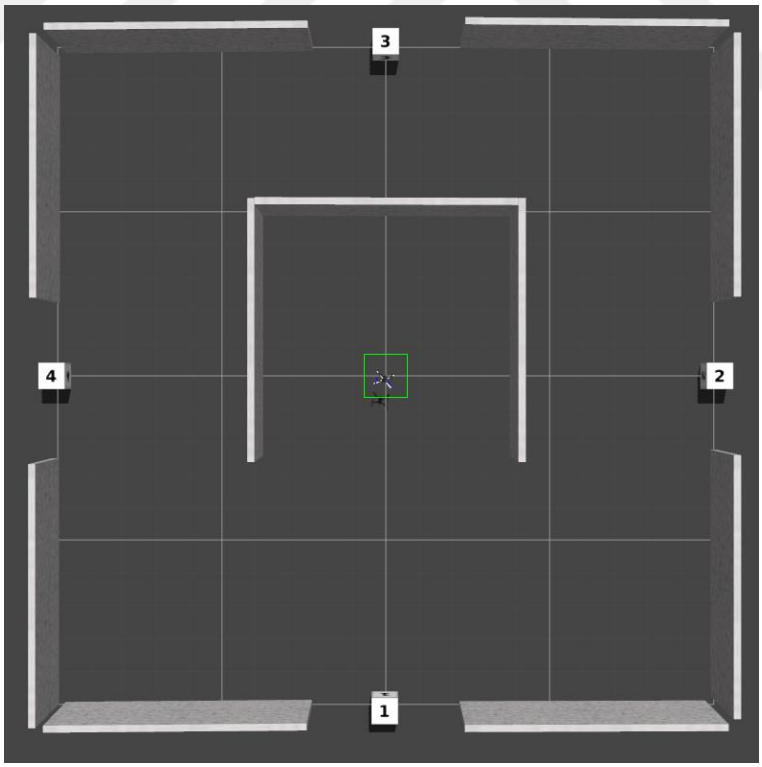
Şekil 7.8. *Ortama eklenen yeni engellere göre hava aracının hareketi (3. adım)*

Hava aracının ilk ortamda ve sonradan engel eklenen ortamda izlediği yollar ve süreler Tablo 7.3'te verildiği gibidir.

Tablo 7.3. *İzlenen yollar ve işlem süreleri*

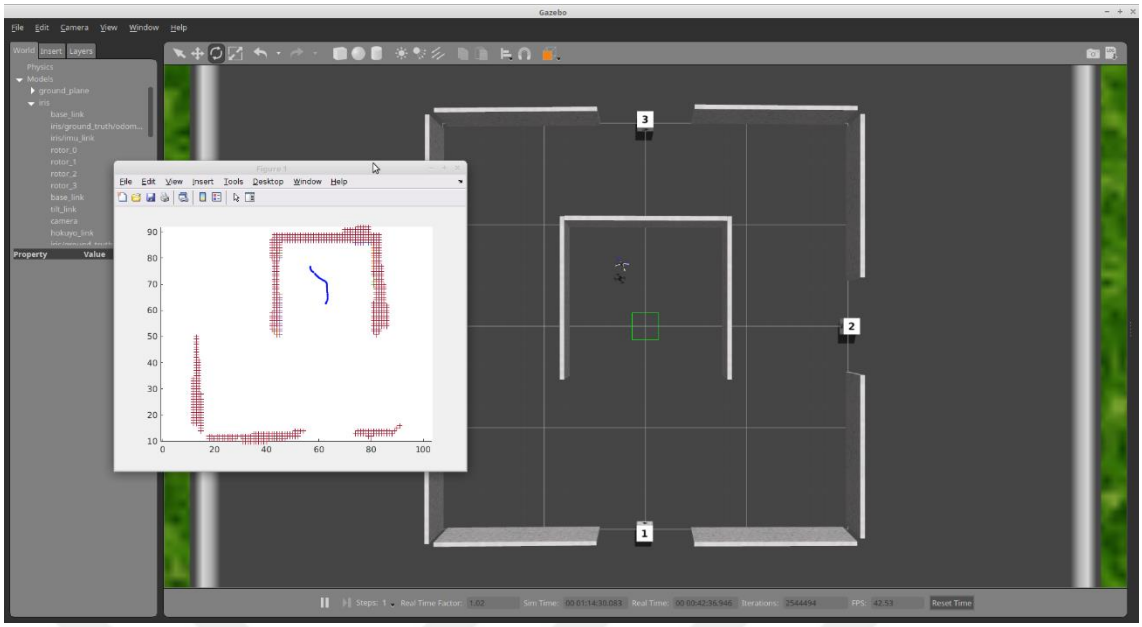
		
41.4363 s	44,6309 s	52.5540 s

Görüldüğü üzere Vektör Alan Histogramı algoritmasının statik ortamlarda başarımı yüksektir. Dinamik ortamlarda da engellerle baş etme konusunda oldukça başarılıdır. Ancak, lokal algoritmaların doğasından kaynaklanan yerel minimuma takılma sorunlarına Vektör Alan Histogramı Algoritması'nda da karşılaşılmıştır. Bu durum Şekil 7.9'daki ortam oluşturularak hava aracının bulunduğu konumdan 3 numaralı konuma ulaşması ile test edilerek açıklanmıştır.



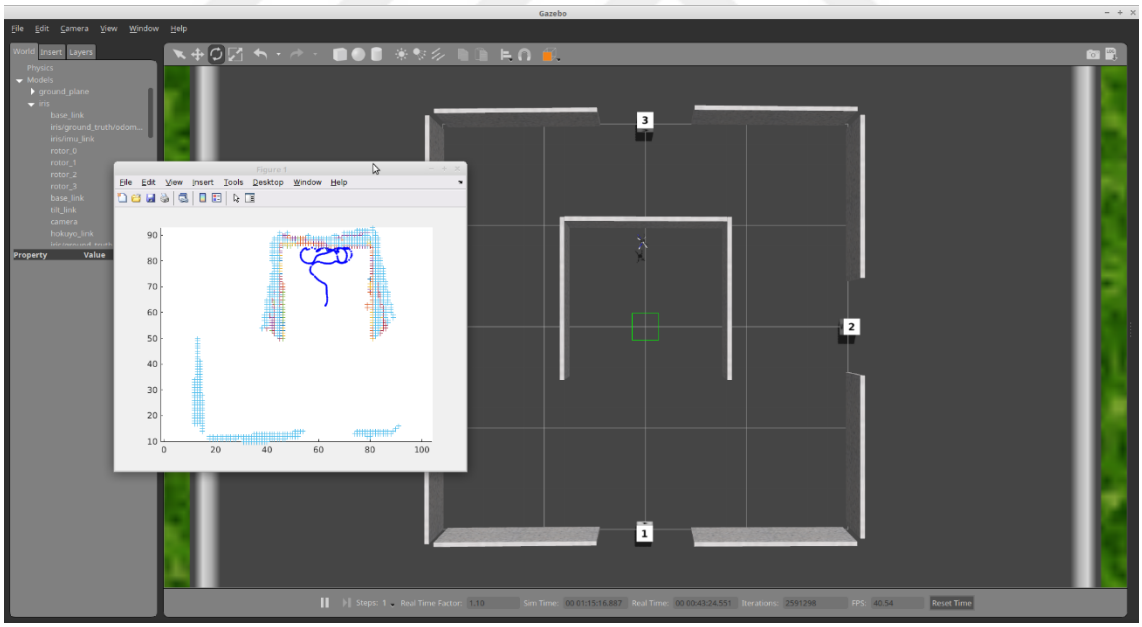
Şekil 7.9. *Örnek Gazebo ortamı*

Hava aracı ilk aşamada yoluna Şekil 7.10'daki gibi devam etmiştir.

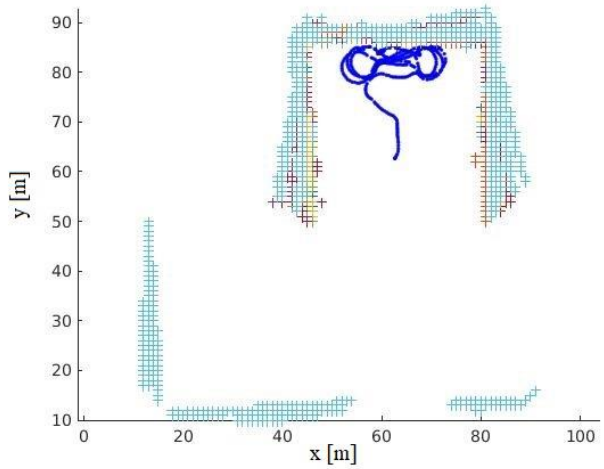


Şekil 7.10. *Hava aracının 3 numaralı hedefe yönlenmesi*

Sonrasında ise Şekil 7.11'deki gibi salınım yapmıştır ve Şekil 7.12'deki gibi hedefine ulaşamamıştır.



Şekil 7.11. *Hava aracının hedefine giderken yaptığı salınım*



Şekil 7.12. *U şeklinde engeller karşısında hava aracının izlediği yol*

Hava aracına eşit mesafede duran engeller dar bir koridor oluşturmuştur, engellerin uyguladığı itme kuvvetlerinin sonucunda Şekil 7.12'de de görüldüğü üzere hava aracı osilasyona girmiştir. Uçuş, hava aracının engele çarpması ile sonlanmıştır.

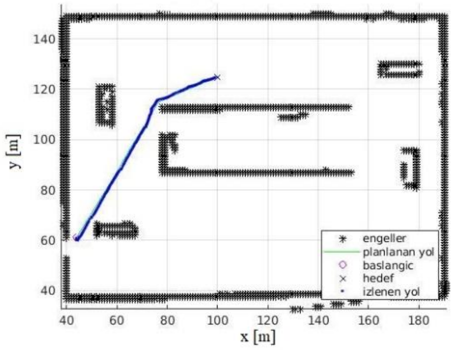
7.2. Hibrit (A* ve VFH+) Yöntem Sonuçları

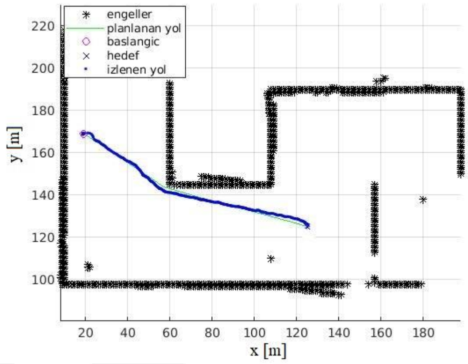
Bu bölümde, tezin temelini oluşturan ve 4. bölümde işleyişi hakkında detaylı bilgilerin yer aldığı A* tabanlı VFH+ destekli engelden sakınan yol planlama algoritmasının başarımı Gazebo ortamında oluşturulmuş farklı test ortamları ile değerlendirilmiştir.

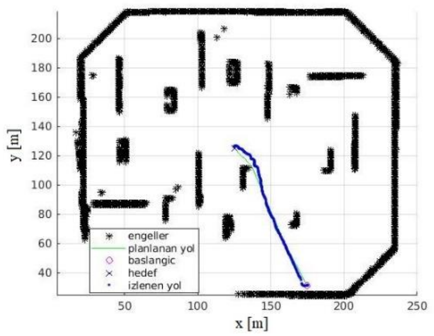
7.2.1. Statik ortam testleri

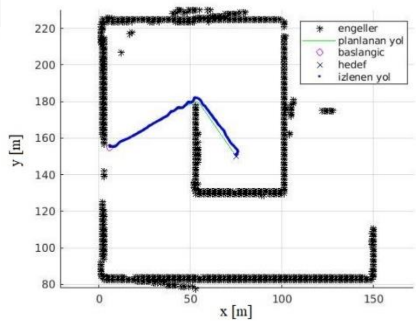
Bu bölümde, algoritmanın çalışması ilk olarak hava aracının bilinen bir ortamda hedefine ulaşırken izlediği yol planı ile değerlendirilecektir. Farklı ortamlarla, farklı başlangıç ve bitiş noktaları için elde edilen yol planları Tablo 7.4'te verilmiştir.

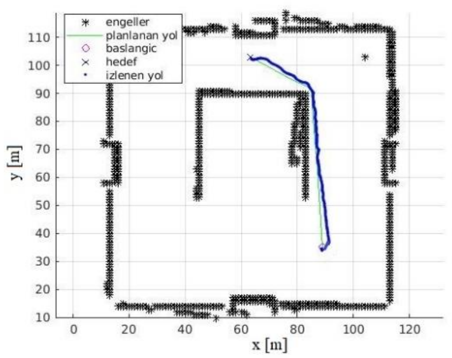
Tablo 7.4. *Statik ortam testleri*

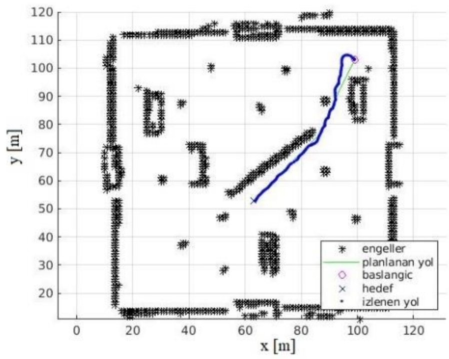










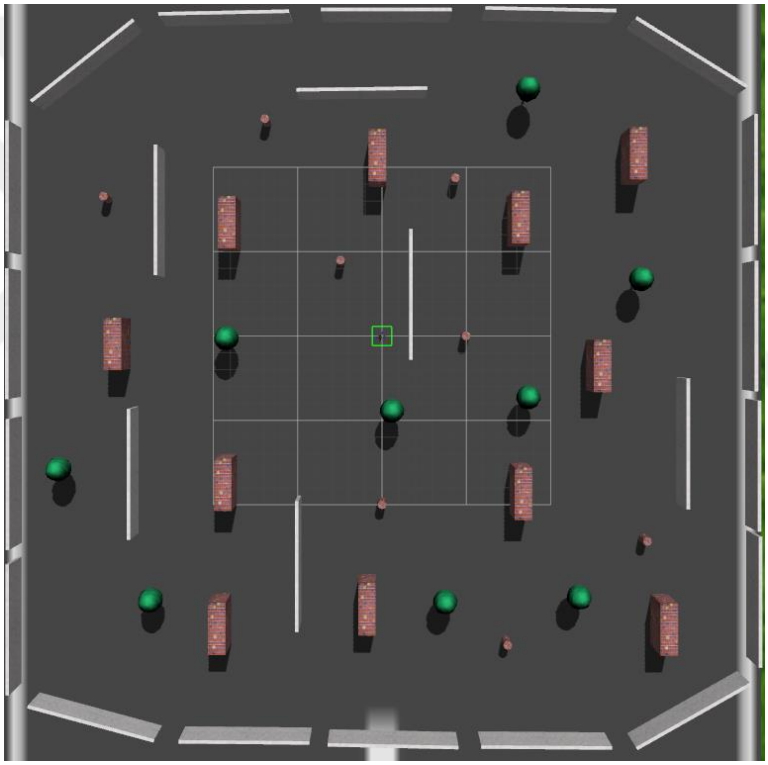


Statik ortamda elde edilen sonuçlar, hava aracının 0,4 metre yakınına engel yaklaşmadığı sürece hava aracının VFH+'ı kullanmadan A* ile oluşturulmuş yol planını izlediğini göstermektedir.

7.2.2. Düzgün haritalanamamış bölgelerde hava aracının davranışı

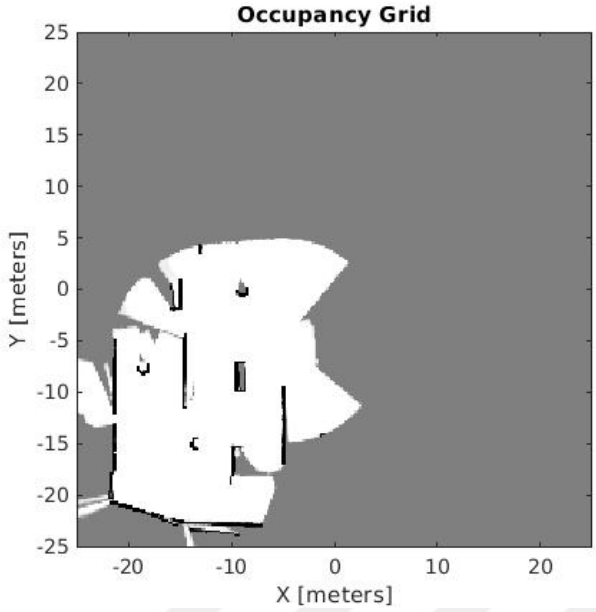
A* algoritması ile yol planı önceden verilen bir haritaya göre oluşturulmaktadır. Bu harita, hava aracının çalışma bölgesinde belirli bir süre uçuş yaparak lidar vasıtasıyla engelleri tespit etmesi ile elde edilmektedir.

Şekil 7.13'teki Gazebo ortamında hava aracı 30 saniye uçuş yaptıktan sonra elde edilen doluluk haritası Şekil 7.14'te verilmiştir.



Şekil 7.13. *Örnek Gazebo ortamı*

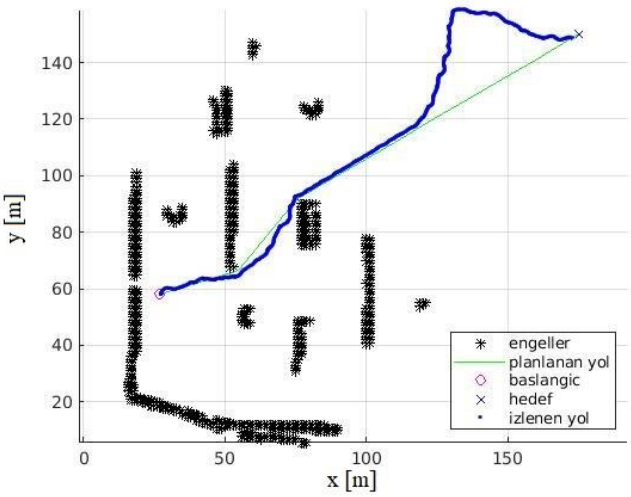
Oluşturulan haritada gri bölgeler taranmamış, siyah bölgeler engel bulunan ve beyaz bölgeler ise yol planında yer alabilecek, engel bulunmayan bölgeleri temsil etmektedir.



Şekil 7.14. *Ortamın 30 saniyede çıkarılan doluluk haritası*

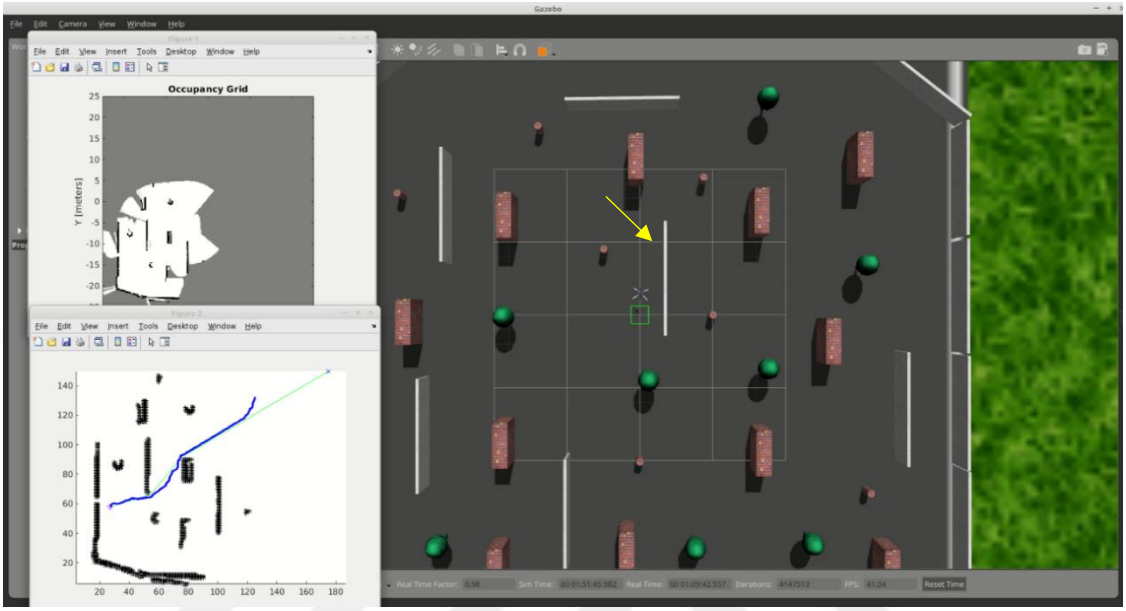
Bir önceki bölümde, hava aracının gideceği hedef, bilinen bölgelerde seçilmiştir. Bu bölümde ise hava aracının gideceği hedef, bilinmeyen, daha önce taranmamış bölgede olduğu durumda, yapılacak yol planı ve hava aracının davranışı incelenmiştir.

Kısmi bir harita oluşturulduktan sonra algoritma gereği hava aracı askıda beklemektedir ve kullanıcıdan bir hedef konumu girmesi beklenmektedir. Kullanıcı hedef konumu girdikten sonra hava aracı askıda durduğu konumu başlangıç konumu varsaymaktadır ve belirlenen hedefe giderken izlediği yol planı Şekil 7.15'te görülmektedir.



Şekil 7.15. *Bilinmeyen bölgedeki engele ulaşırken hava aracının izlediği yol*

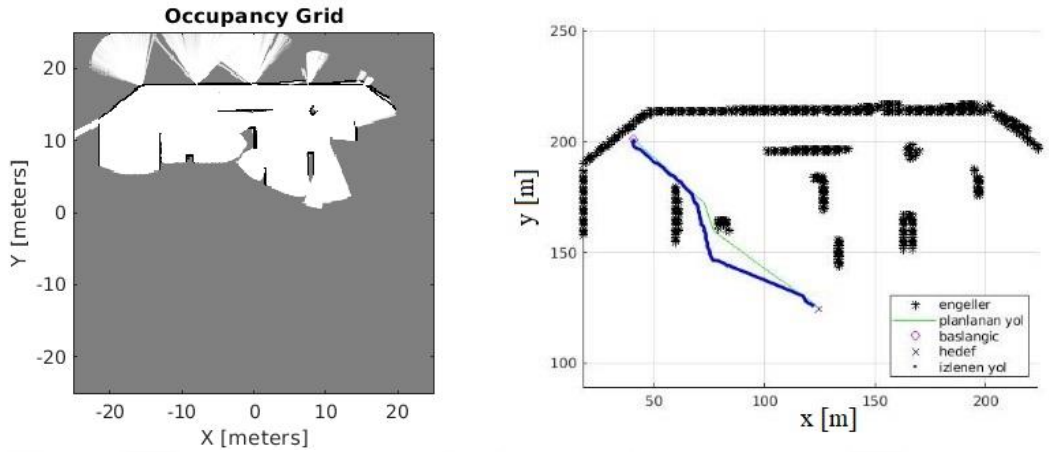
Şekil 7.16'da A* algoritması ile gri bölgede engel olmadığı varsayılarak yeşil çizgiyle gösterilen plan yapılmıştır. Bu plana göre hareket eden hava aracı, gri bölgede, daha önce algılanmamış engel (Şekil 7.16'da sarı okla belirtilen engel) ile karşılaştığı anda global plandan çıkarak VFH+ ile engelden sakınmıştır.

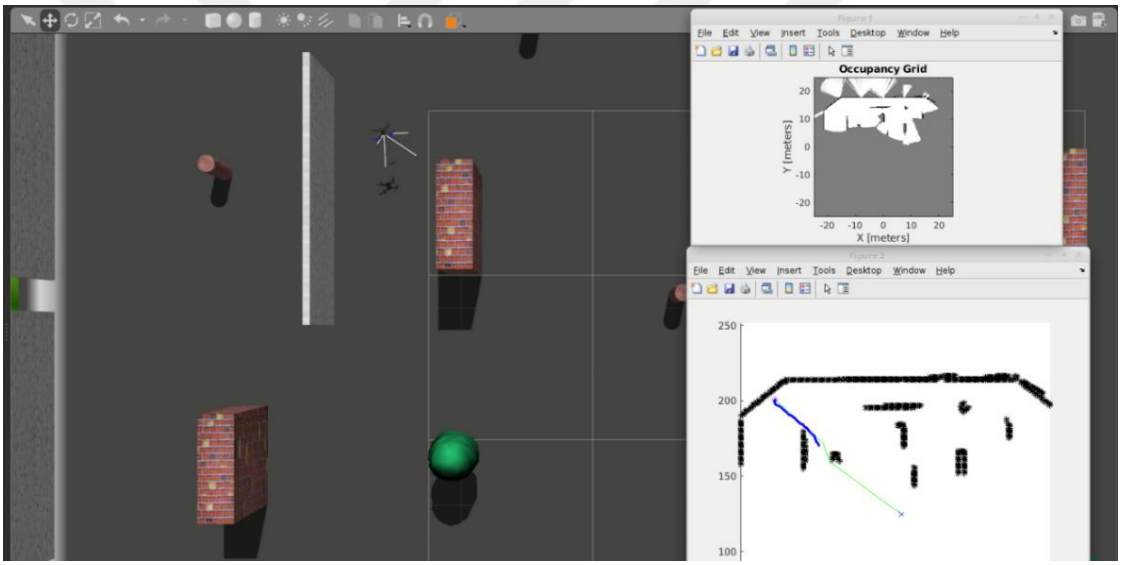


Şekil 7.16. *Bilinmeyen bölgede bulunan hedefe giderken hava aracının davranışı*

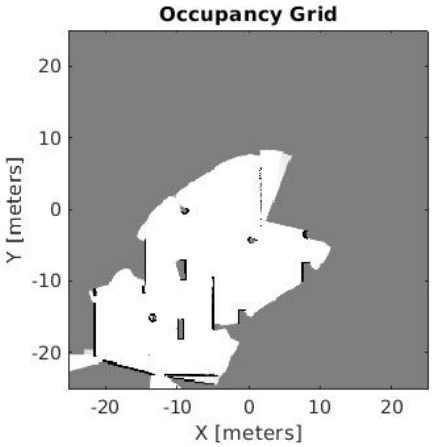
Yine eksik çıkarılmış haritaya göre hava aracının aynı ortamda farklı başlangıç noktalarından farklı hedeflere gittiği test sonuçları Tablo 7.5 ve Tablo 7.6'da verilmiştir. Bu iki tabloda hava araçlarının Gazebo ortamındaki anlık konumları, ortamın kısmi haritası ve nihai yol planları görülmektedir.

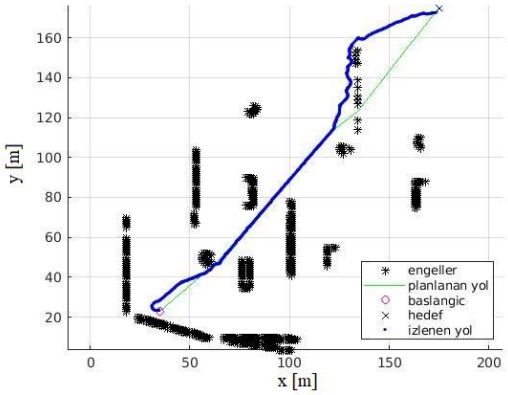
Tablo 7.5. *Eksik haritalanmış ortamlarda hava aracının davranışı (1)*

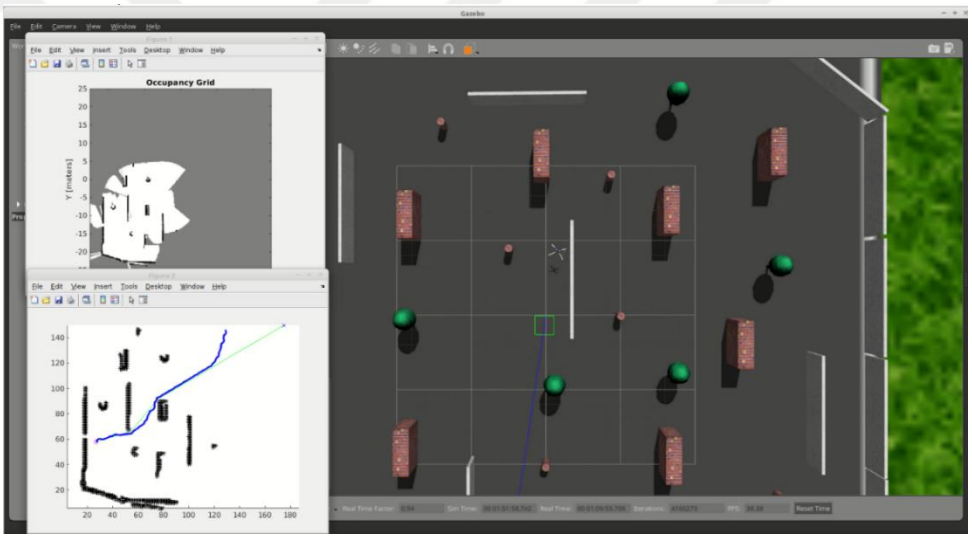




Tablo 7.6. *Eksik haritalanmış ortamlarda hava aracının davranışı (2)*





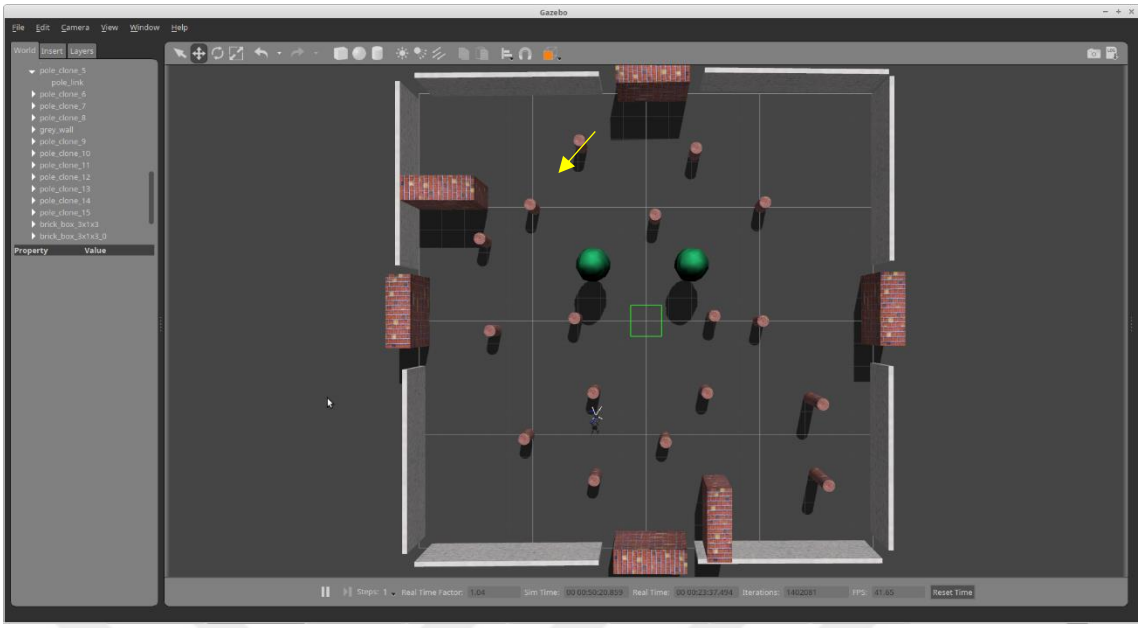


Görüldüğü üzere ortam haritasının eksik çıkarıldığı bölgelerde de algoritma hava aracını güvenli bir şekilde hedefine ulaştırmaktadır.

7.2.3. Dinamik ortam testleri

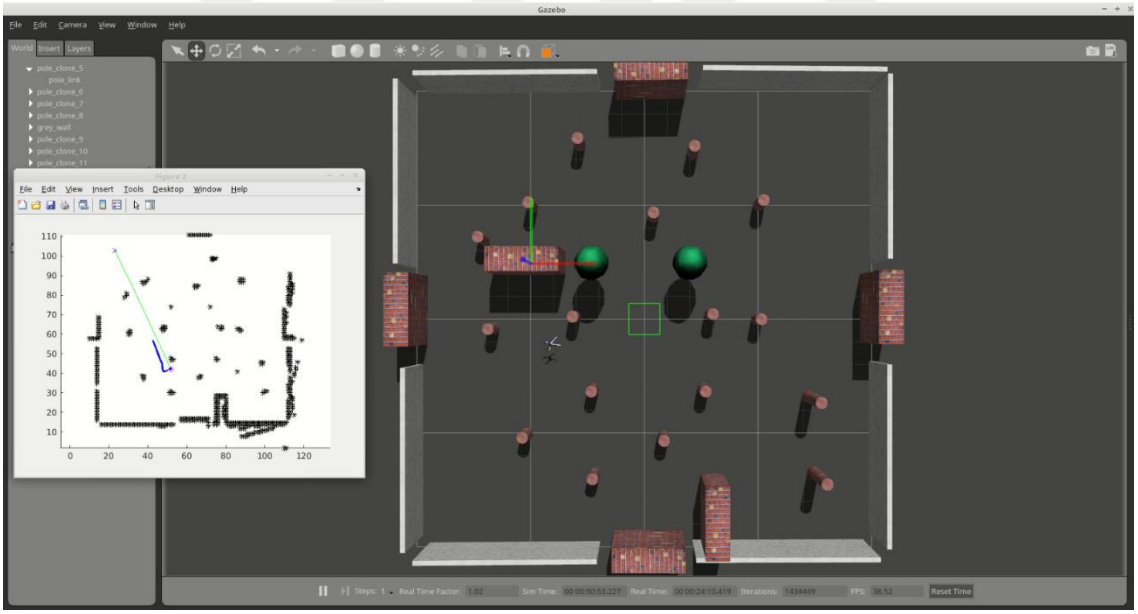
Algoritmanın dinamik ortamlarda çalışmasını test etmek amacıyla, ortama A* ile yol planı yapıldıktan sonra engeller eklenmiş ve/veya engel konumları değiştirilmiş, engellerin hareket dinamiği dikkate alınmamıştır.

Hava aracı başlangıçta Şekil 7.17'deki gibi bir ortamdadır.



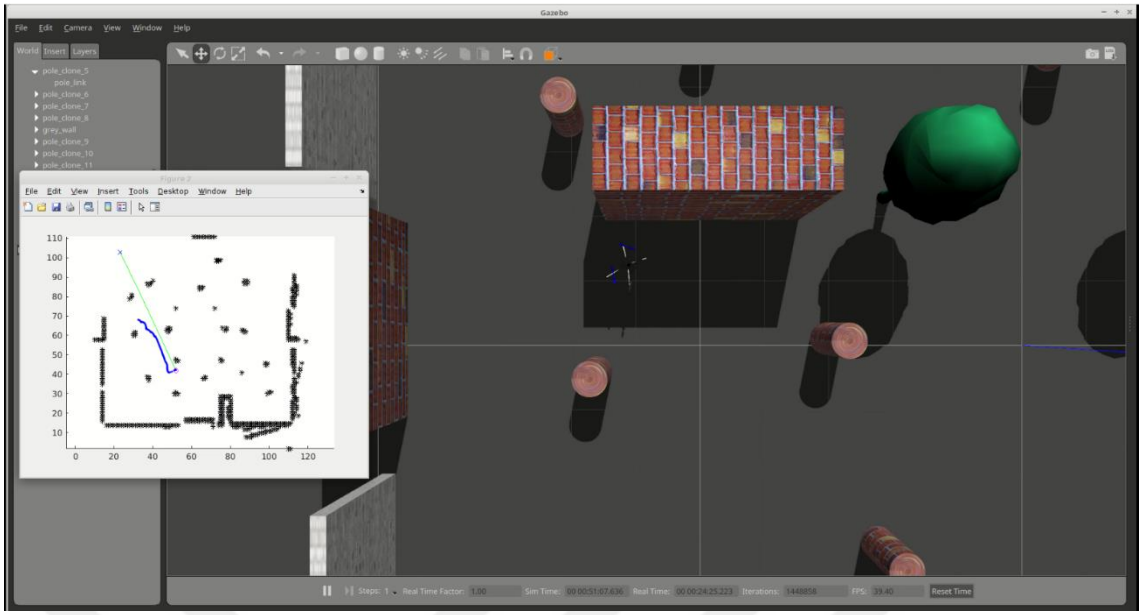
Şekil 7.17. *Örnek Gazebo ortamı*

Hava aracı hedefine doğru giderken Şekil 7.17'de sarı okla işaretlenen engelin yeri Şekil 7.18'deki gibi değiştirilmiştir.



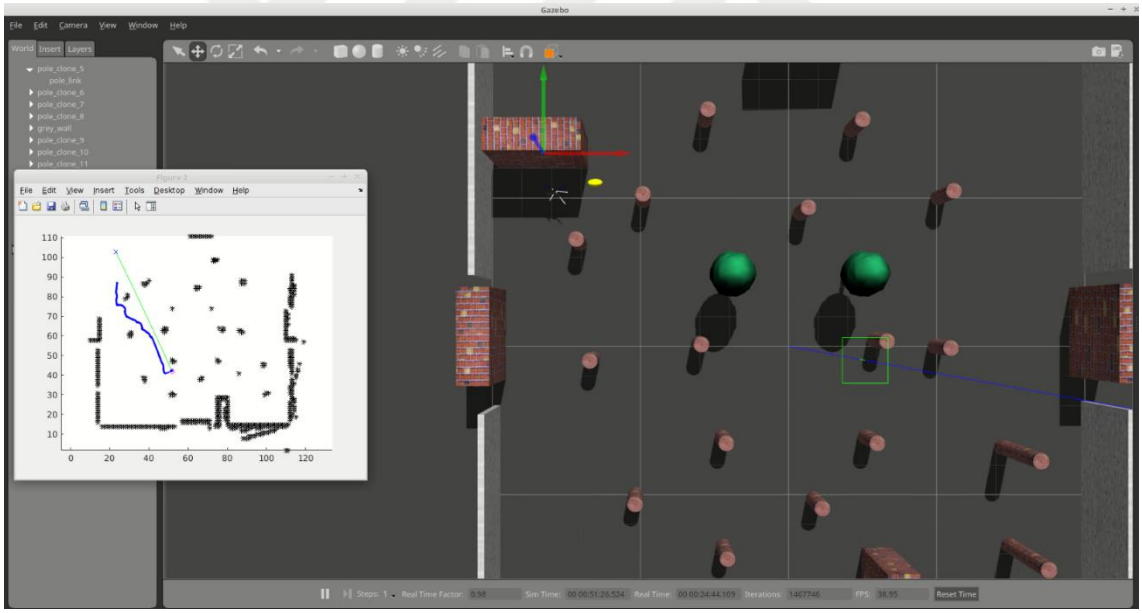
Şekil 7.18. *Engel konumunun değişimi*

Yeni engeli lidar ile algılayan hava aracı Şekil 7.19'da yeşil çizgi ile çizilen rotadan çıkarak Şekil 7.20'deki gibi hareketini sürdürmüştür.



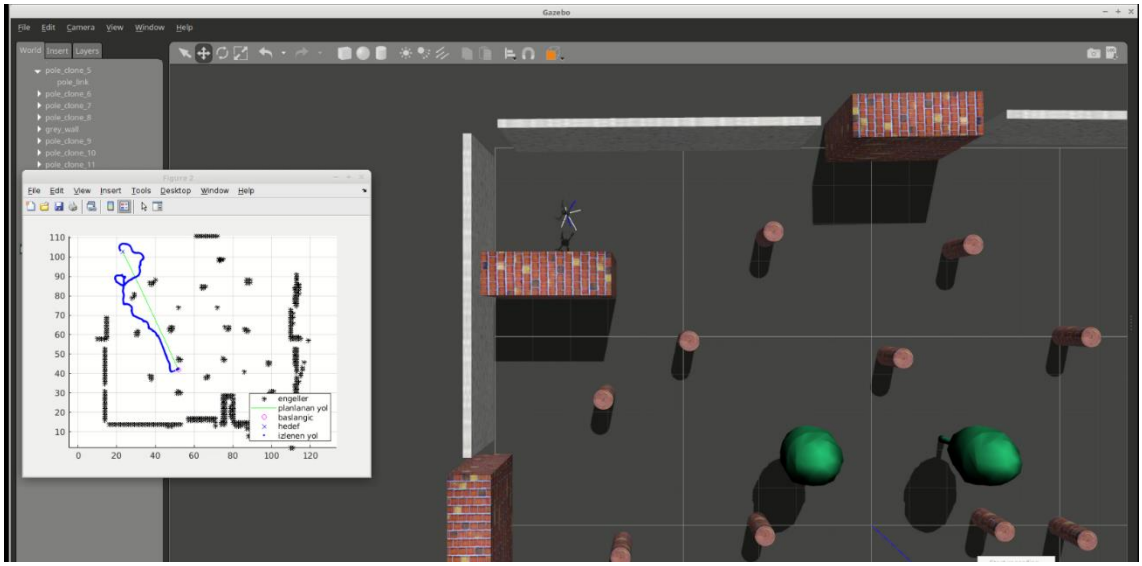
Şekil 7.19. *Engel konumunun değişimi (2)*

Sonrasında aynı engel hava aracının önüne Şekil 7.20'deki gibi sürüklenmiştir

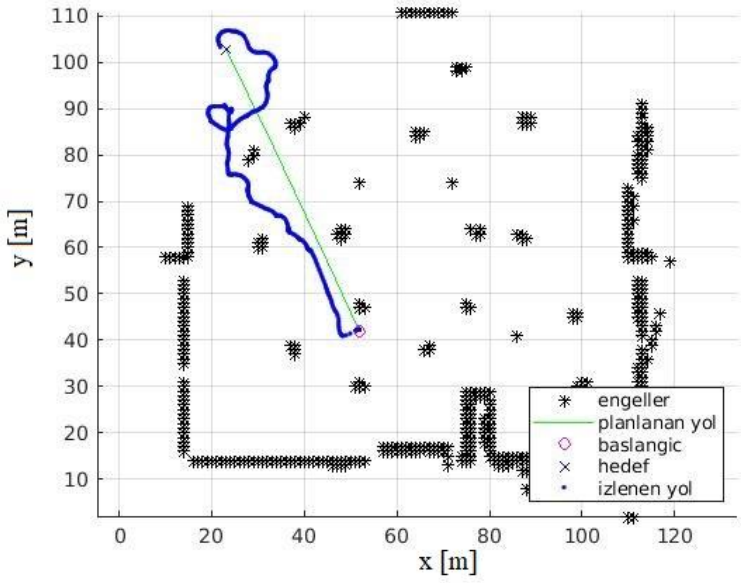


Şekil 7.20. *Engel konumunun değişimi (3)*

Önüne çıkan engel karşısında yeniden global plandan çıkarak Şekil 7.21'deki gibi hedefine ulaşan hava aracının izlediği nihai yol Şekil 7.22'de verilmiştir.

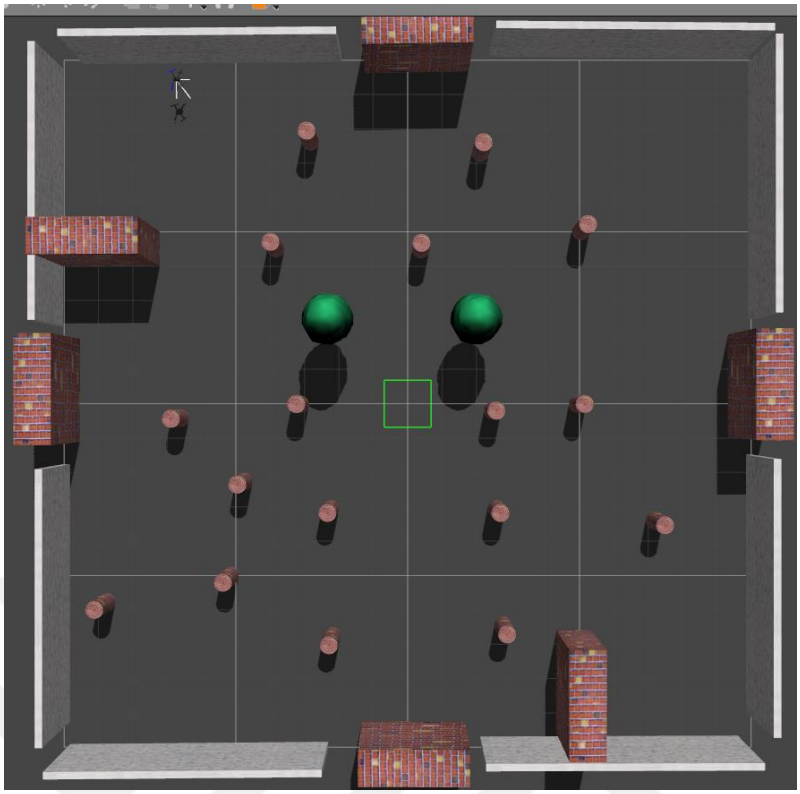


Şekil 7.21. *Hava aracının hedefine ulaşırken izlediği yol ve Gazebo ortamı*



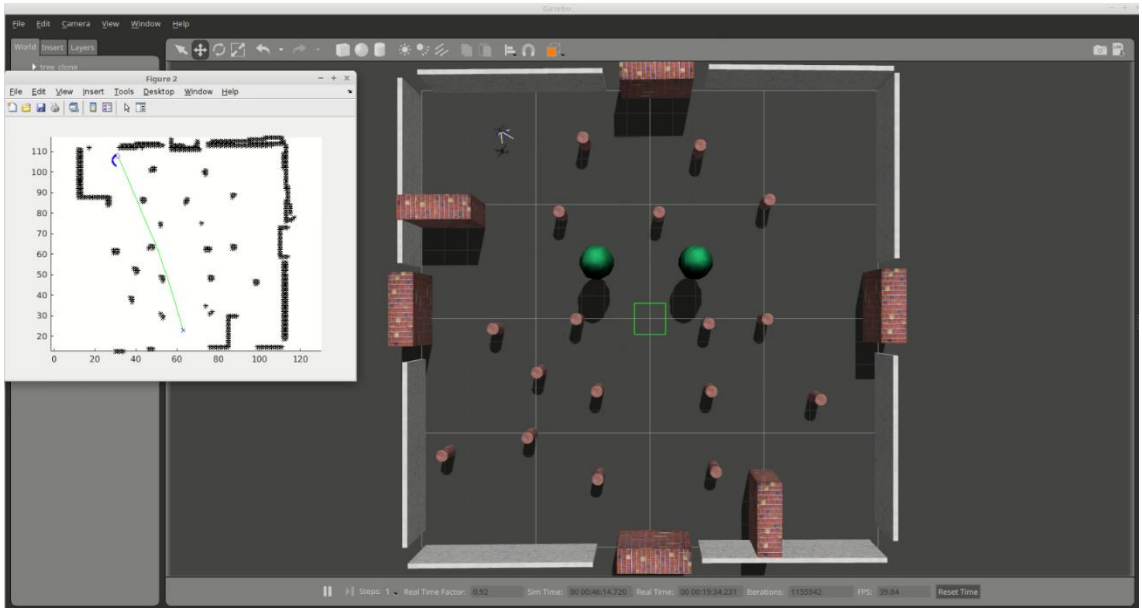
Şekil 7.22. *Hava aracının dinamik engellere karşı izlediği nihai yol planı*

Bir diğer dinamik ortam testi ise Şekil 7.23'teki ortamda gerçekleştirilmiştir.



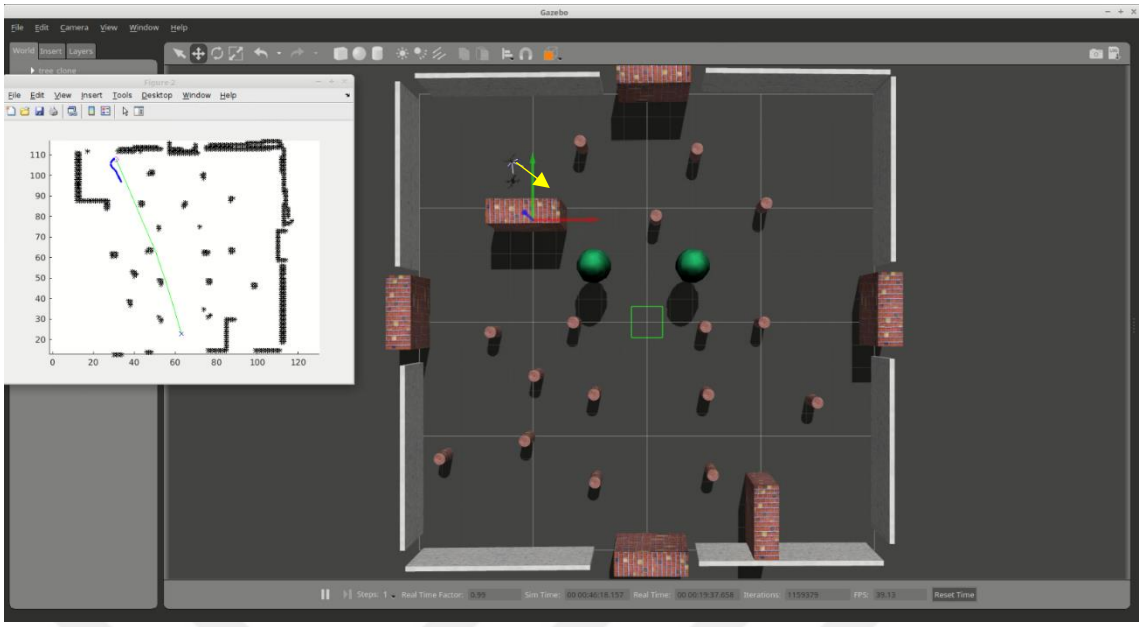
Şekil 7.23. *Örnek Gazebo ortamı*

Başlangıç noktasından harekete geçen hava aracı için A* algoritması ile oluşturulan yol planı Şekil 7.24'deki yeşil çizgide görüldüğü gibidir.



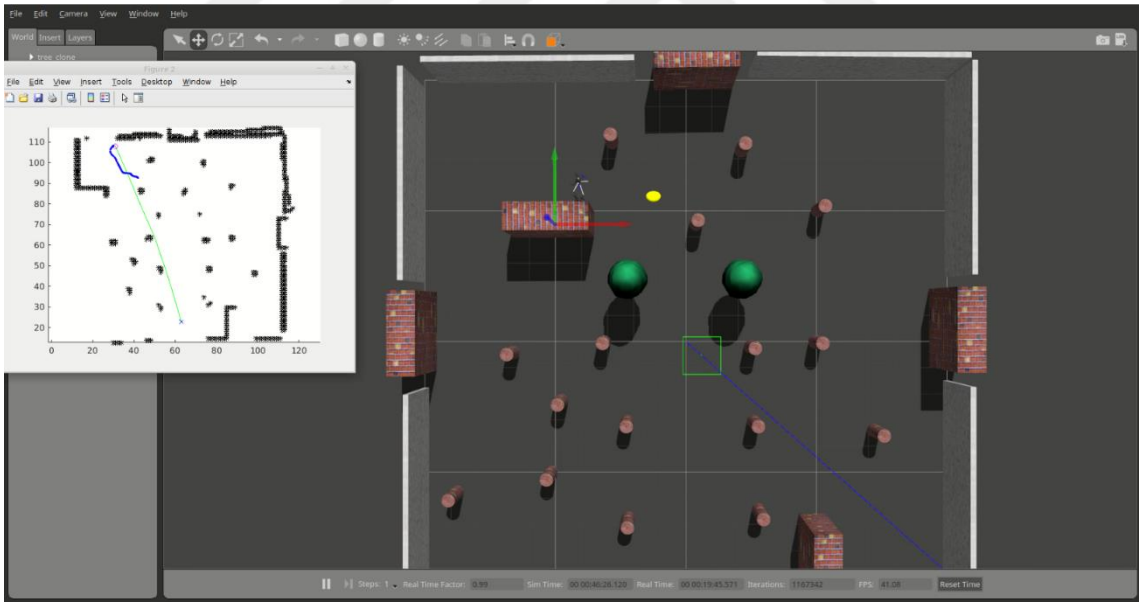
Şekil 7.24. *Hava aracının hedefe doğru yönlenmesi*

Daha sonra Şekil 7.25'te sarı okla belirtilen engel sürüklenerek aracın plandan çıkması sağlanmıştır.

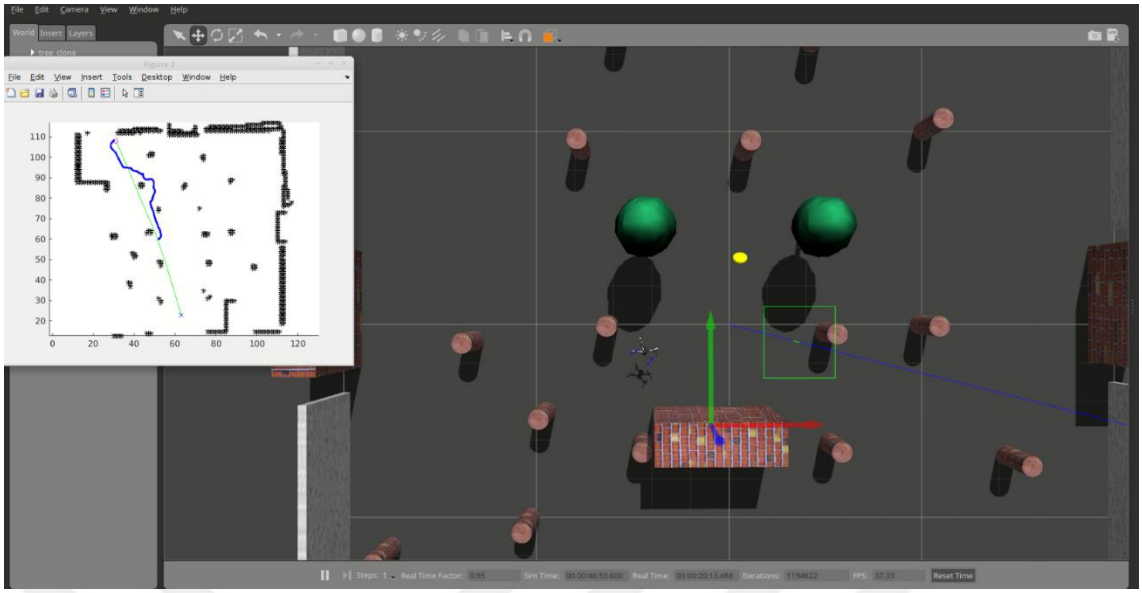


Şekil 7.25. *Engel konumunum değişimi (1)*

Hava aracı ilk engel yer değişimini Şekil 7.26'daki gibi aştıktan sonra bu engel, Şekil 7.27'deki gibi tekrar hava aracının yoluna sürüklenmiştir.

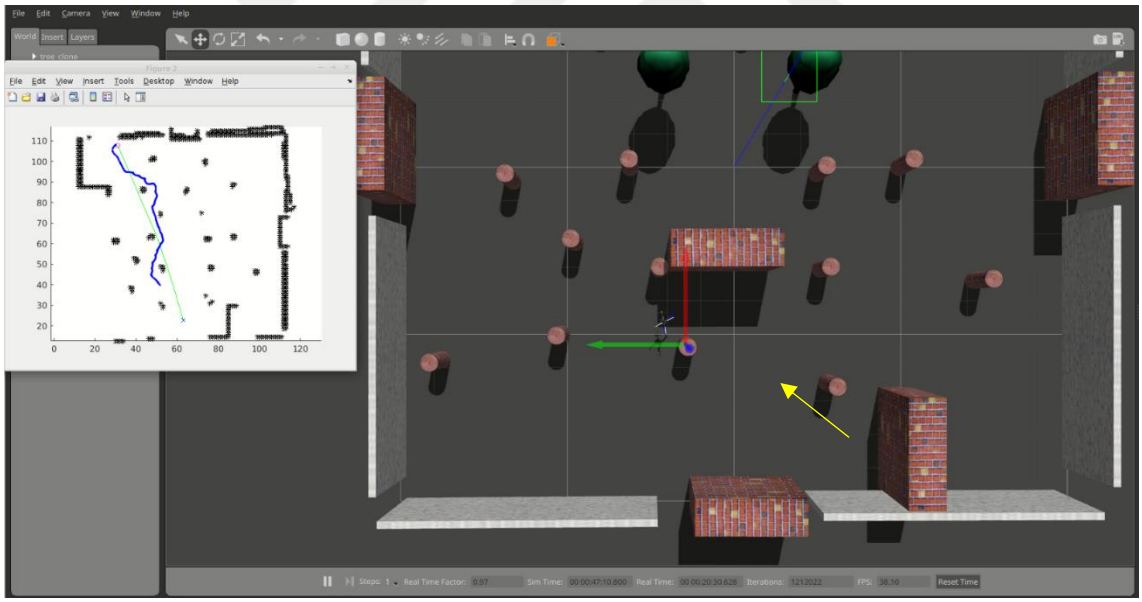


Şekil 7.26. *Engel konumunun değişimi (2)*

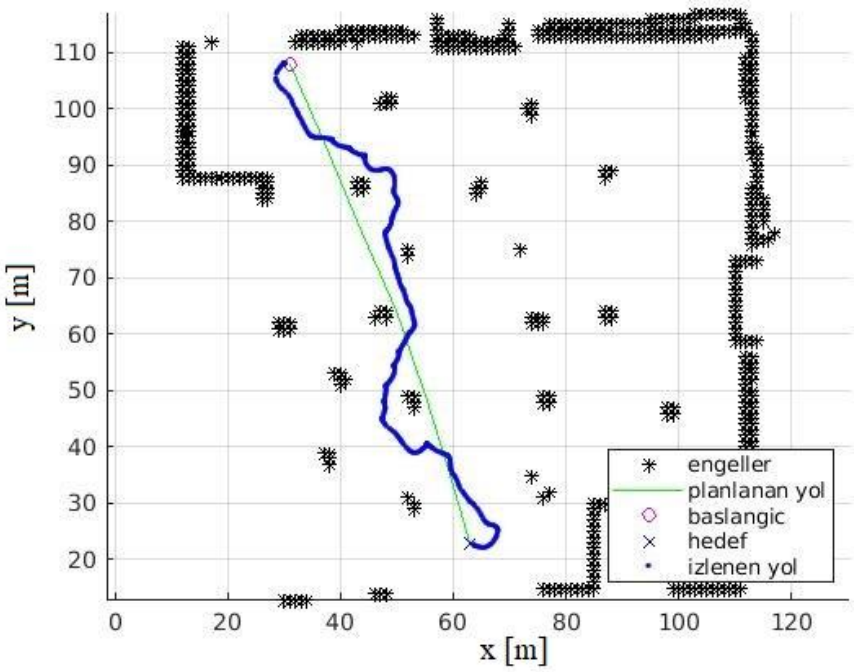


Şekil 7.27. *Engel konumunun değişimi (3)*

Son olarak Şekil 7.28'de sarı okla işaretli silindir şeklindeki kolon engelinin hava aracının önüne sürüklenmesi sonucu izlenen yeni yol Şekil 7.29'da görüldüğü gibidir.

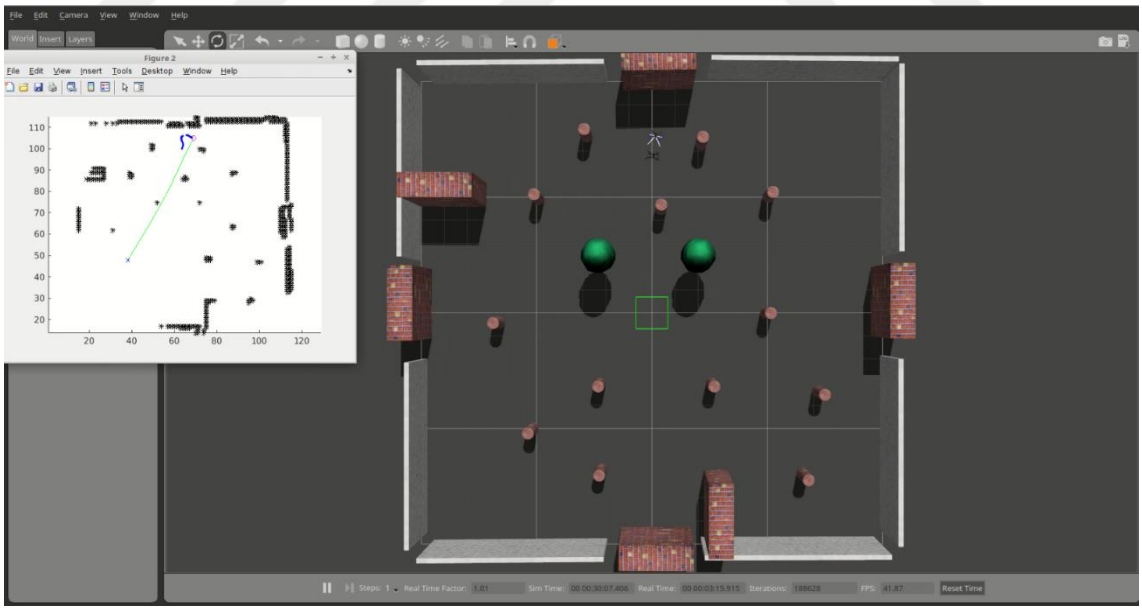


Şekil 7.28. *Engel konumunun değişimi (4)*



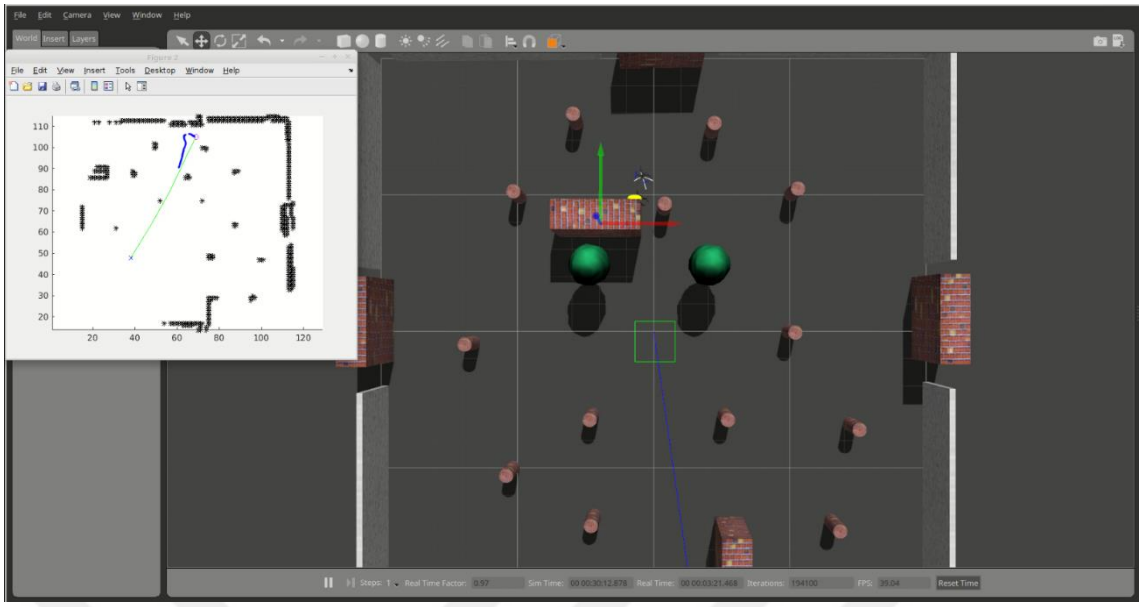
Şekil 7.29. *Hava aracının dinamik engellere karşı izlediği nihai yol planı*

Dinamik ortam testleri için oluşturulan bir diğer ortam ise Şekil 7.30'da verilmiştir.

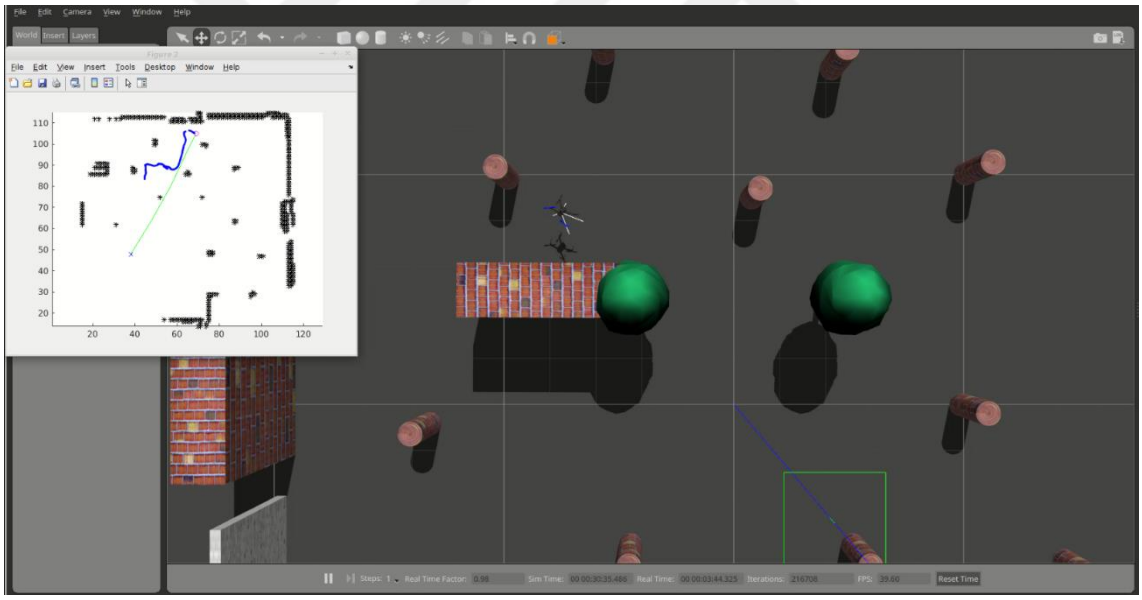


Şekil 7.30. *Hava aracının hedefine yönlenmesi*

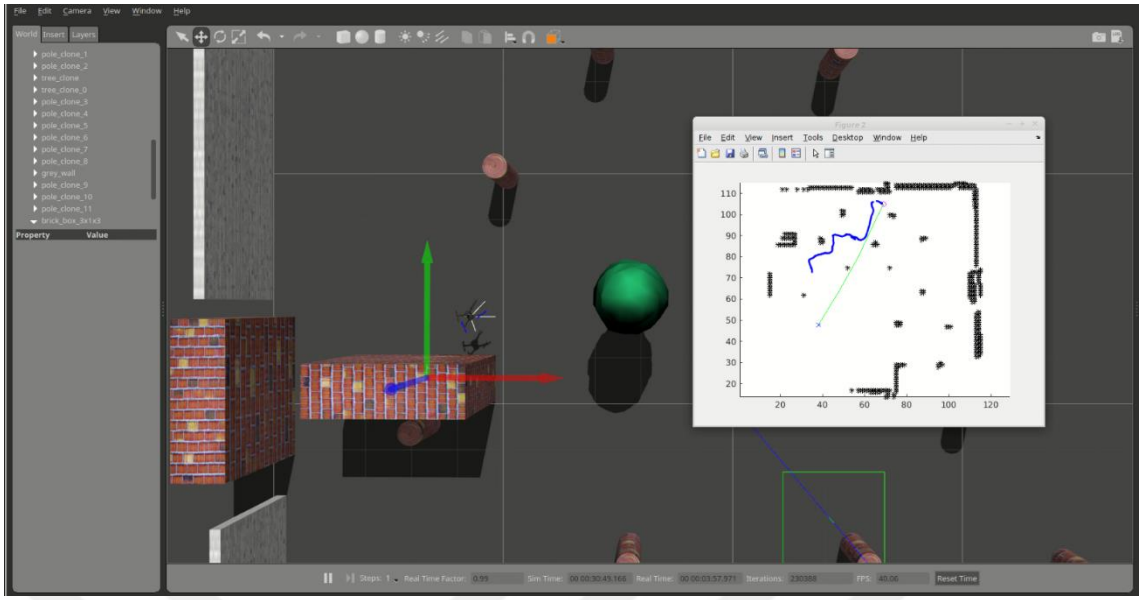
Hava aracının Şekil 7.31'deki gibi önüne çıkan engel karşısında davranışı sırasıyla Şekil 7.32, Şekil 7.33 ve Şekil 7.34'te görülmektedir.



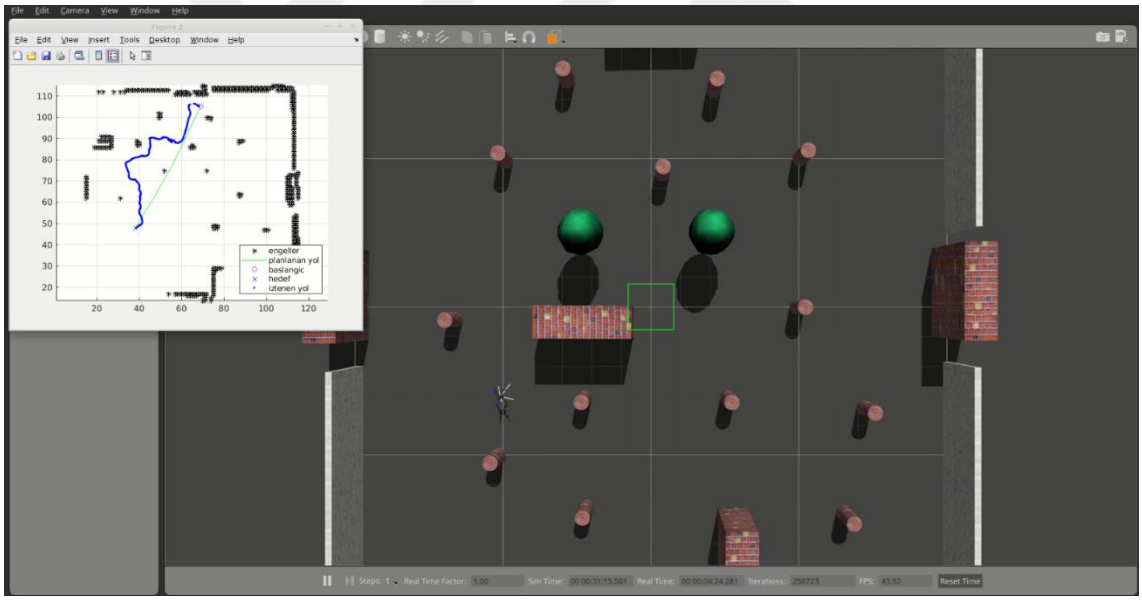
Şekil 7.31. *Engel konumunun değişimi (1)*



Şekil 7.32. *Engel konumunun değişimi (2)*

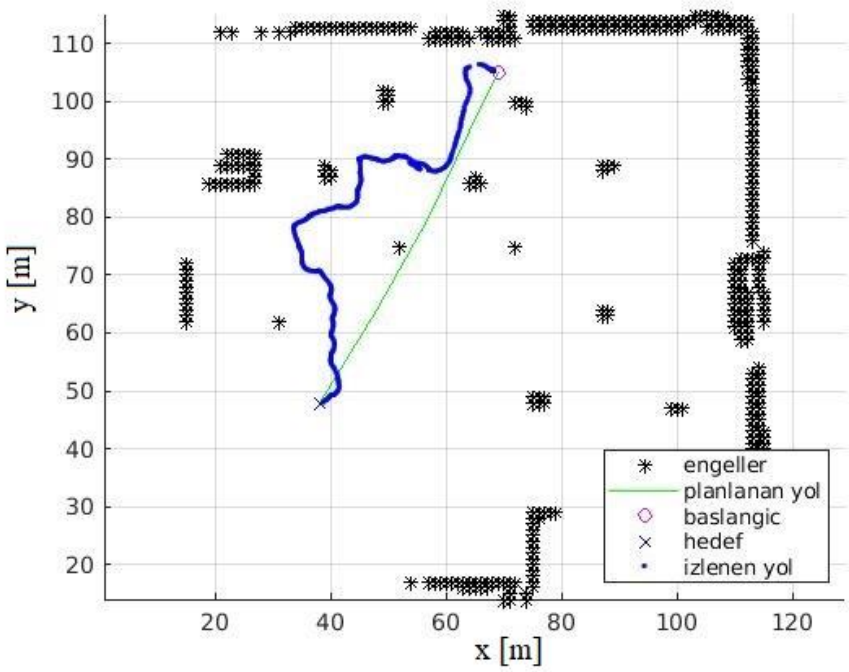


Şekil 7.33. *Engel konumunun değişimi (3)*



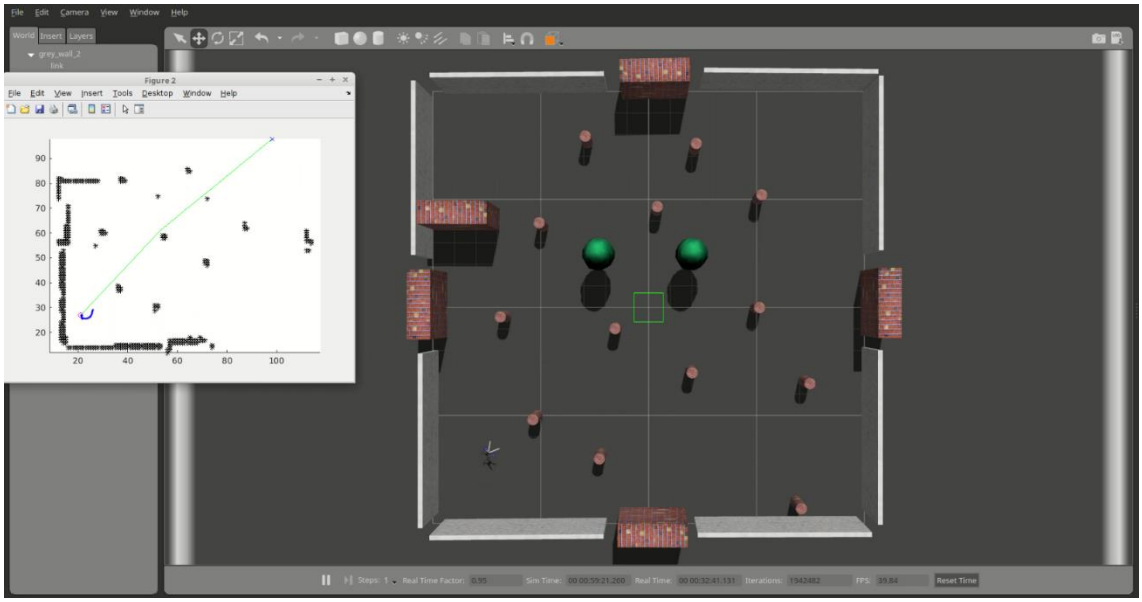
Şekil 7.34. *Hava aracının hedefine ulaşırken izlediği yol ve Gazebo ortamı*

Hava aracının dinamik engeller sonucunda izlediği nihai yol Şekil 7.35'te verilmiştir.

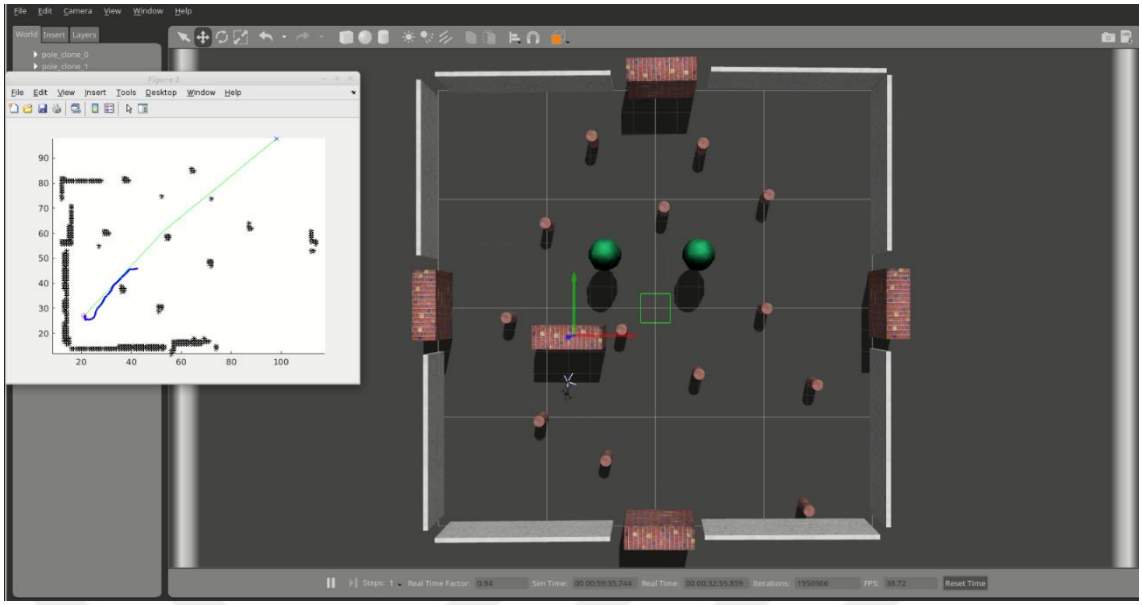


Şekil 7.35. *Hava aracının dinamik engellere karşı izlediği nihai yol planı*

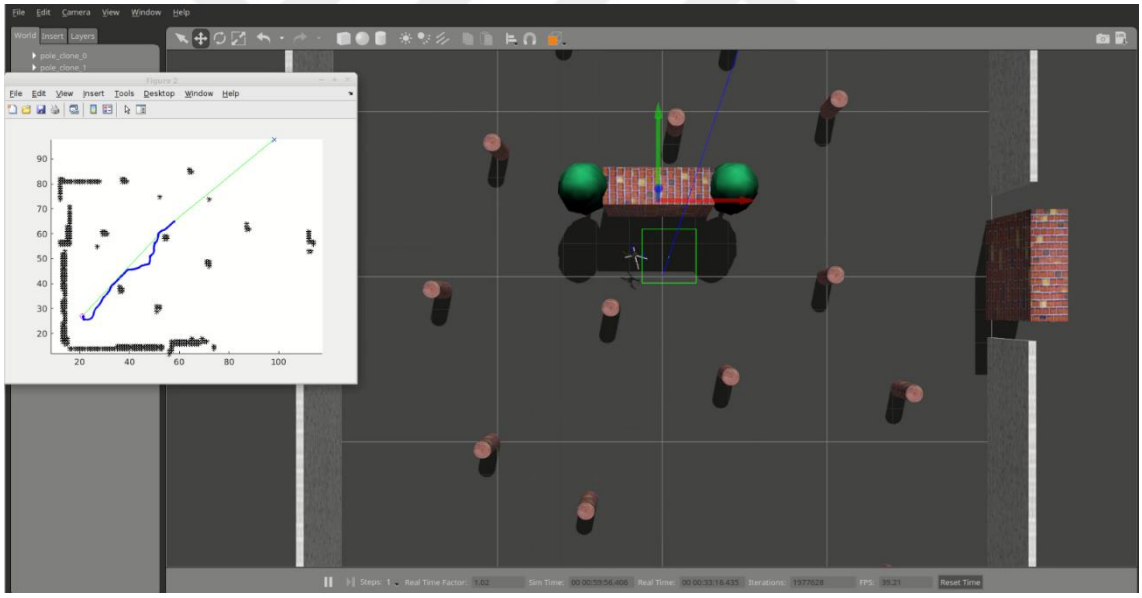
Son olarak başlangıç noktası Şekil 7.36'da görülen hava aracı için oluşturulan ortam Şekil 7.37, Şekil 7.38, Şekil 7.39'da görüldüğü gibi engellerin konum değiştirmesi ile dinamik hale getirilmiştir.



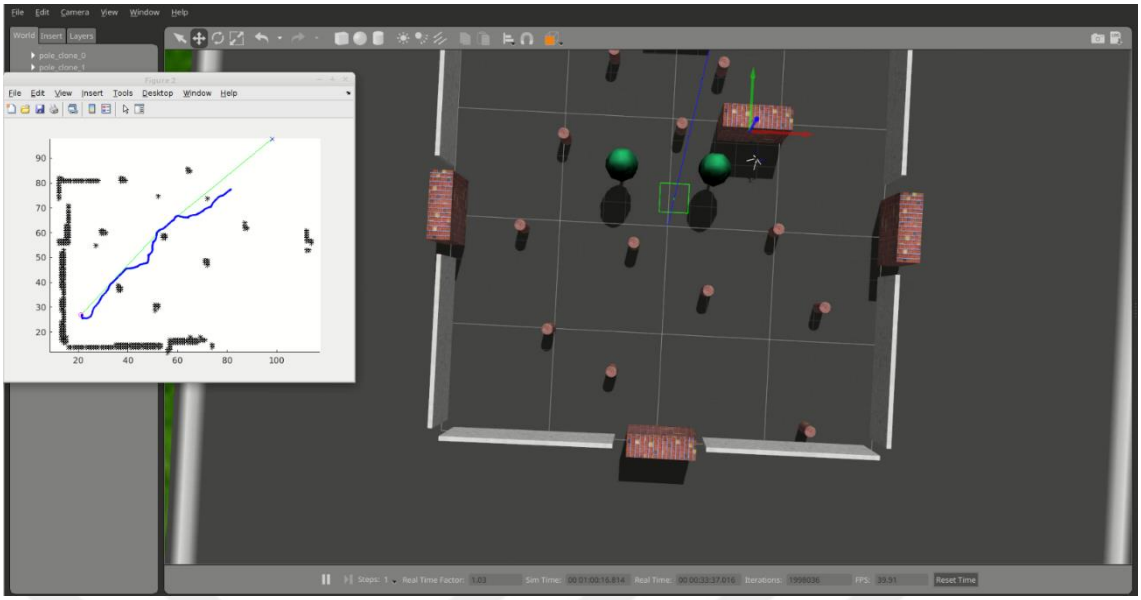
Şekil 7.36. *Hava aracının hedefine yönelmesi*



Şekil 7.37. *Engel konumunun değişimi (1)*

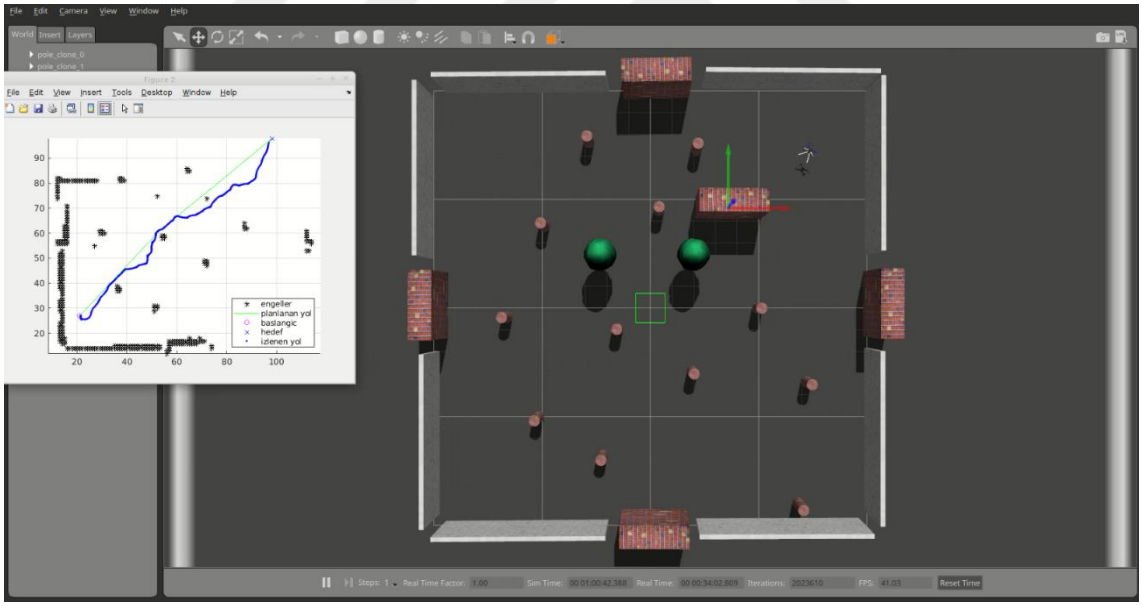


Şekil 7.38. *Engel konumunun değişimi (2)*



Şekil 7.39. *Engel konumunun değişimi (3)*

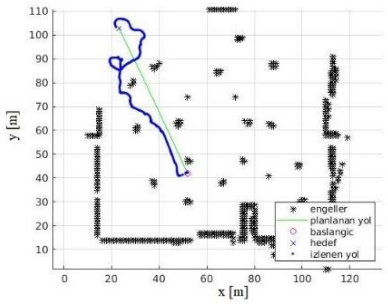
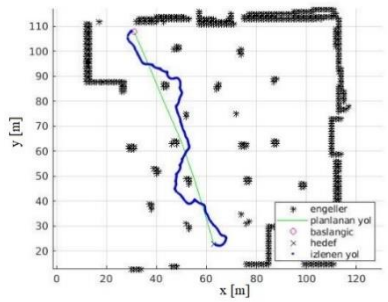
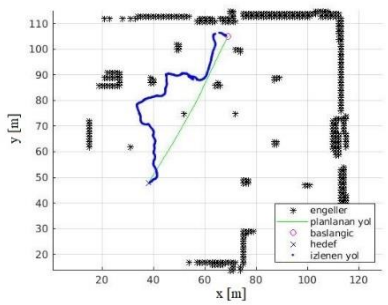
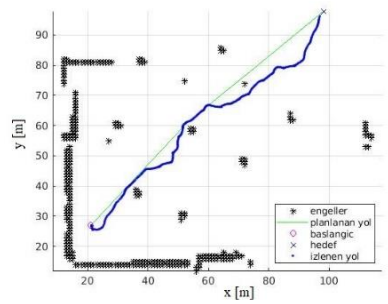
Dinamik engeller sonucu hava aracının hedefe ulaşırken izlediği yol, Şekil 7.40'ta görülmektedir.



Şekil 7.40. *Dinamik engellerle karşılaşan hava aracının izlediği nihai yol ve Gazebo ortamı*

Tablo 7.7'de ise dinamik ortam testlerinin gerçekleştirildiği ortamlar, başlangıç ve hedef konumları ile işlem süreleri yer almaktadır.

Tablo 7.7. *Dinamik ortam test sonuçları*

Başlangıç Konumu	**Hedef Konumu**	**Süre(s)**	**Yol planı**
(-2.0178,-4.0389, 2)	(-8, 8, 2)	86,9147	
(-6.3988, 9.1096, 2)	(0, -8, 2)	83,5654	
(1.2179, 8.5659, 2)	(-5, -3, 2)	73,0589	
(-8.3984, -7,0569)	(7, 7, 2)	80,8681	

Görüldüğü üzere hibrit algoritma, statik ortamlarda yol planı yapıyor iken ortamda meydana gelen değişimlere göre de global plandan çıkarak engelden sakınabilmektedir.

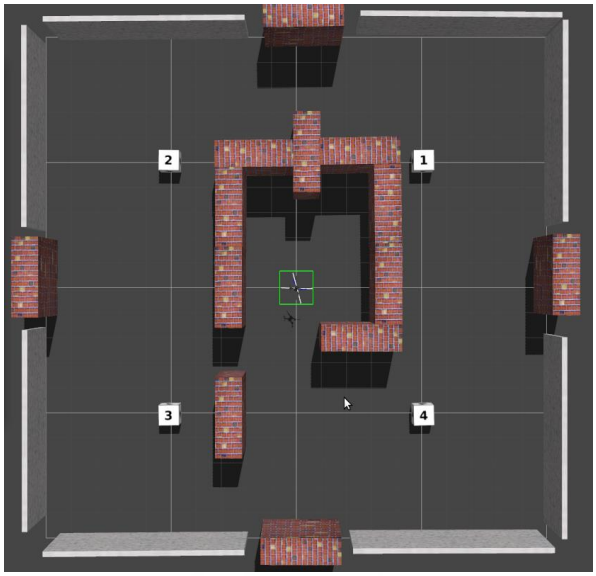
Oluşturulan yol planlarının engelden kaçınma başarımı yüksektir ancak işlem süresi açısından bir değerlendirme yapmak yalnızca hibrit algoritma ile mümkün değildir. Bir sonraki bölümde hem süre açısından hem de oluşturulan yol planı açısından ROS navigasyon yığınları da kullanılarak değerlendirme yapılacaktır.

7.2.3. ROS navigasyon yığınları ve hibrit algoritma performansı

ROS platformunda yer robotları için oluşturulmuş navigasyon yığınları (ROS Navigation, 2020) mevcuttur. Bu navigasyon yığınları içerdiği ROS düğümleri vasıtası ile robotun konumunu, mesafe algılayıcı sensörden gelen verileri ve hedef konumunu kullanarak robotu güvenli (engelsiz) bölgede hareket ettirecek hız komutları gönderirir. Bu hız komutları, global ve lokal planlayıcılar ile hesaplanmaktadır. Navigasyon yığınları global planlayıcı olarak Dijsktra algoritmasını, lokal planlayıcı olarak ise Dinamik Pencere Yaklaşımı (DWA-Dynamic Window Approach) algoritmasını kullanmaktadır. Dijsktra algoritması, A* algoritmasının daha ilkel versiyonudur. DWA ise hız tabanlı bir yöntemdir. Robotun çalışma uzayında robot dinamiğine uygun olası hızları hesaplar ve bu hızlardan en uygun (engele uzaklığı en fazla, hedefe en yakın giden) rotayı oluşturan hızı seçmeye yönlendirir.

ROS navigasyon yığınlarının planlayıcı parametreleri hava aracı için güncellenerek ve aynı ortamda hem hibrit algoritmanın hem de navigasyon yığınlarının süre ve planlanan yol açısından performansı değerlendirilmiştir.

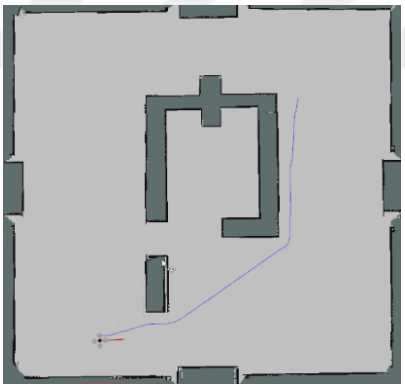
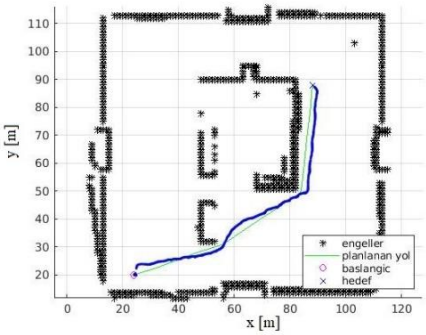
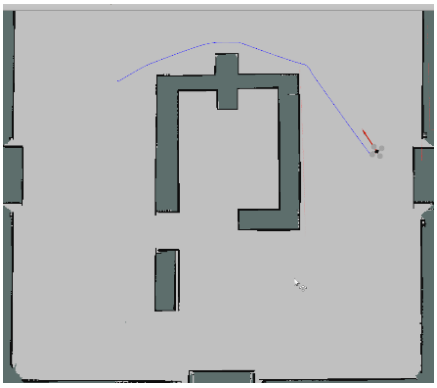
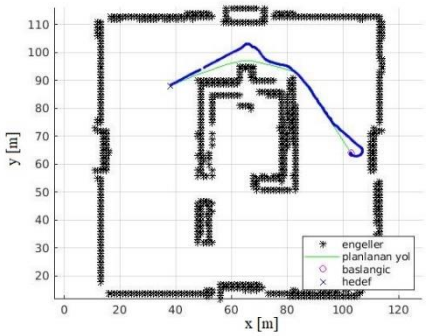
Test ortamı olarak aşağıdaki Şekil 7.41'deki Gazebo ortamı oluşturulmuştur.

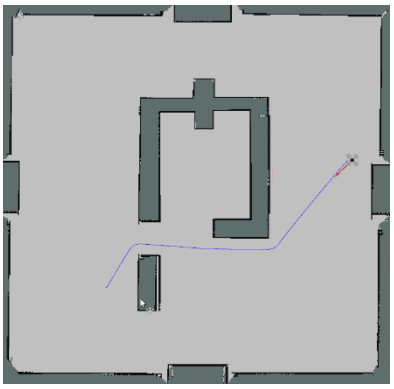
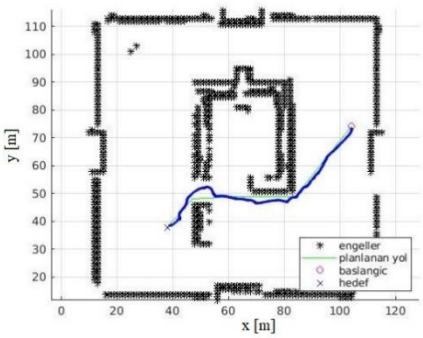
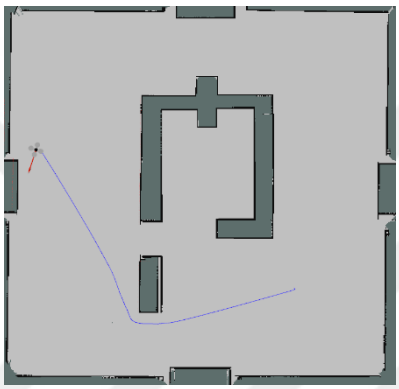
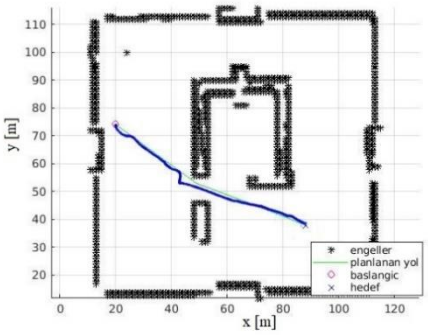
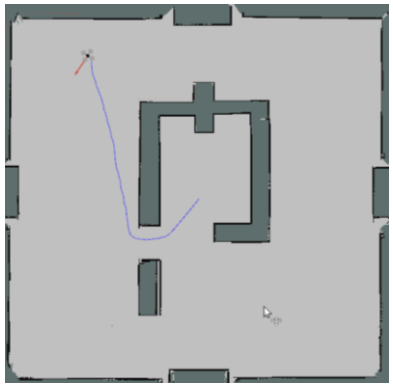
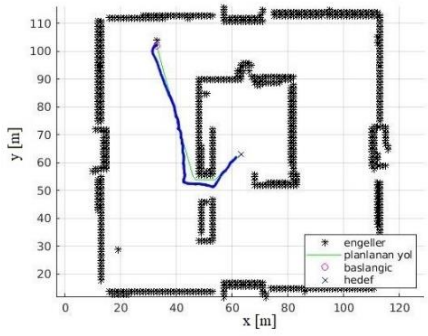


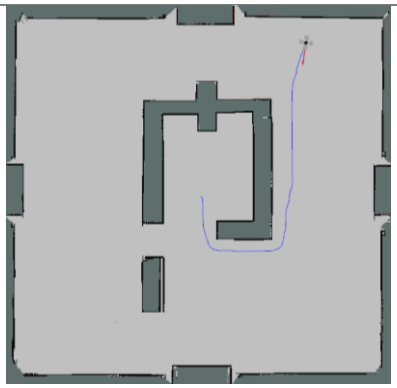
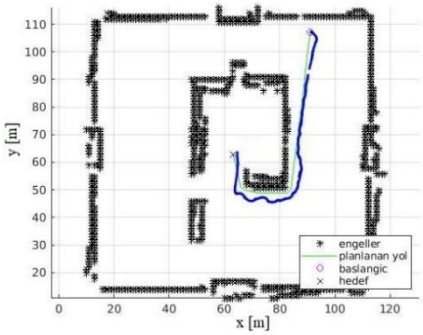
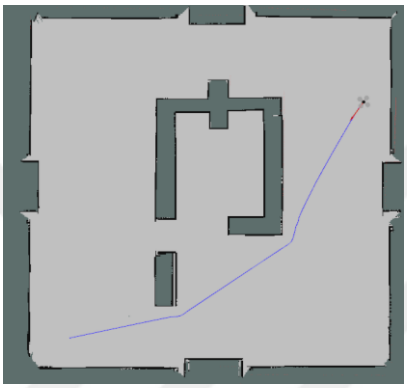
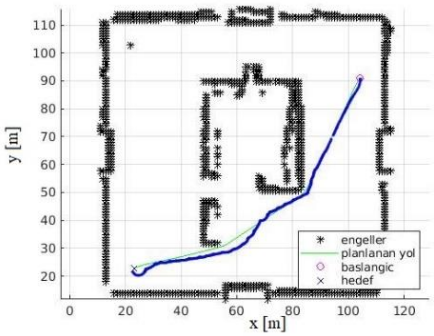
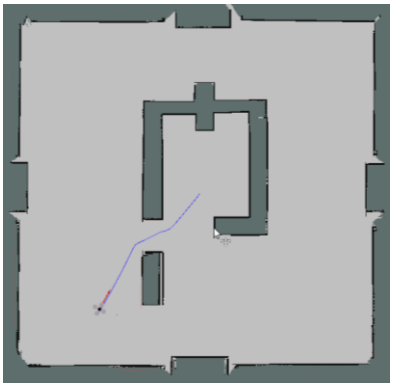
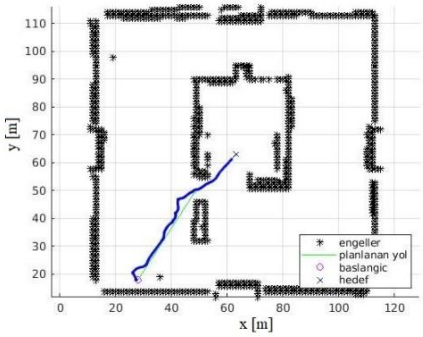
Şekil 7.41. *Örnek Gazebo ortamı*

Tablo 7.8'de işlem süresi ve oluşturulan yol planları açısından sonuçlar verilmiştir.

Tablo 7.8. *Algoritmaların aynı ortamda test edilmesinden elde edilen sonuçlar*

	ROS Navigasyon Yığınları	**Hibrit Algoritma**
Test-1-		
Süre(s)	56,40	77,99
Test-2-		
Süre(s)	70,99	61,10

Test-3-		
Süre(s)	130,39	64,06
Test-4-		
Süre(s)	158,79	60,56
Test-5-		
Süre(s)	129	53,59

Test-6-		
Süre(s)	157,8	89,79
Test-7-		
Süre(s)	192,2	87,21
Test-8-		
Süre(s)	27,2	40,9

Elde edilen sonuçlar değerlendirildiğinde 8 uçuş verisi açısından süre artış yüzdeleri Tablo 7.9'da hesaplanmıştır.

Tablo 7.9. *Algoritma işlem sürelerinin kıyaslanması*

	Hibrit Algoritma İşlem Süresi (s)	**ROS Navigasyon Yığınları İşlem Süresi (s)**	**Süre Artış Yüzdesi**
Test-1-	56,40	77,99	38,28014
Test-2-	61,10	70,99	16,18658
Test-3-	64,06	130,39	103,5436
Test-4-	60,56	158,79	162,2028
Test-5-	53,59	129,00	140,7166
Test-6-	89,79	157,80	75,7434
Test-7-	87,21	192,20	120,3876
Test-8-	27,20	40,90	50,36765

Tablo 7.9'a göre, aynı ortamda, aynı başlangıç ve bitiş noktaları için hibrit algoritma, ROS navigasyon yığınlarına göre oldukça hızlı sonuçlar vermiştir. Hibrit algoritmada sadece başlangıçta, bir kez çalışan A* algoritması, sonrasında sadece gerek duyulduğunda çalışan VFH+ algoritması, ROS navigasyon yığınlarında çalışan algoritmalara göre süre açısından avantaj sağlamıştır. Bu duruma sebep olarak A* algoritmasının daha ilkel olan Dijsktra algoritmasından daha hızlı çalışması gösterilebilir.

8. SONUÇLAR VE TARTIŞMA

Tez kapsamında, dört motorlu bir insansız hava aracı modeli için statik ve dinamik ortamlarda yol planlama algoritması oluşturulmuştur. Oluşturulan algoritmanın, statik ve dinamik ortamlarda sorunsuz çalışabilmesi için global ve lokal yöntemler bir arada kullanılmıştır. Hava aracının hedefine en kısa yoldan gitmesini sağlamak amacıyla global bir yol planlama algoritması olan A* algoritması ile hava aracının ortama sonradan eklenen engellerin de üstesinden gelebilmesi için lokal bir yöntem olan VFH+ algoritması birlikte kullanılmıştır. Önerilen hibrit algoritma, literatürde yer alan VFH* algoritmasından işleyiş olarak farklılık göstermektedir. VFH* algoritmasında hedef yön seçimi aşamasına kadar VFH+, yön seçiminde A* algoritması kullanılırken, önerilen hibrit yöntemde, ilk aşamada A* algoritması ile global plan yapılmakta, sonradan planda olmayan bir engel ile karşılaşıldığı durumda VFH+ algoritması kullanılmaktadır.

Algoritmanın testi, son yıllarda robotik uygulamaların test aşamasında sıkça kullanılan, sensör desteği ve gerçekleme bakımından diğer benzetim ortamlarına göre çok daha verimli çalışan Gazebo ortamında gerçekleştirilmiştir. Farklı büyüklüklerde ve şekillerde engellerle çevrili ortamlar oluşturularak hava aracının rastgele seçilen hedeflere nasıl ulaştığı incelenmiştir. Aynı test ortamında, hava aracının hedefine ulaşması için ROS platformunda yer alan navigasyon yığınları kullanılarak da yapılmıştır. Yapılan testler sonucunda ROS navigasyon yığınlarının ve hibrit yöntemin benzer yol planları oluşturduğu ancak hibrit algoritmanın oluşturulan planı oldukça hızlı gerçekleştirdiği görülmüştür.

Elde edilen sonuçlar değerlendirildiğinde, yol planının düzgünlüğü (smoothness), engellerden kaçınma kabiliyeti ve işlem süresi açısından tatmin edici sonuçlar elde edilmiştir. Hibrit algoritmanın sadece düzgün haritalanmış alanlarda değil haritalamanın doğru yapılmadığı veya eksik yapıldığı alanlarda da hava aracını güvenli bir şekilde hedefine ulaştırması başarı kriteri olarak görülebilir.

Oluşturulan hibrit yöntemin yol planlama görevinin yanı sıra, yol planlamanın yan görev olduğu asıl görevin hava araçları ile yapılabilecek sulama, ilaçlama, tohumlama gibi tarım faaliyetlerinde, arama-kurtarma, haritalama, güvenlik, ürün dağıtımı gibi pek çok uygulamada kullanılabilirliği sağlanabilir. Görüldüğü üzere algoritma, hava aracının konumunu GPS ile tespit ettiğinden dış ortamda kullanılmaya uygundur. İç ortamda

kullanmak ise kamera veya konumlandırma sisteminin mevcut olması durumunda mümkündür.

Son olarak, Ek-1'de yer alan kısıtlar dolayısıyla algoritmaların gerçeklemesi mümkün olmamıştır. İlerleyen dönemlerde gerekli bütçe temini ile eksikler giderilerek algoritmanın testi gerçek ortamlarda yapılabilecektir. Böylelikle Gazebo benzetim ortamında karşılaşılmayan sorunlar, algoritma kaynaklı ise algoritmada iyileştirmeler yaparak veya donanım kaynaklı ise değişim yapılarak giderilmeye çalışılacaktır.

KAYNAKÇA

3D Robotics Inc. (2014). IRIS+ operation manual. Berkeley: 3D Robotics.

Abiy, T., Pang, H., Tiliksew, B., Moore, K., & Khim, J. (2021). *A* search*. Mart 12, 2021 tarihinde Brilliant.org: https://brilliant.org/wiki/a-star-search/ adresinden alındı

ArduPilot Dev Team. (2020). *ArduCopter*. Mart 18, 2018 tarihinde ArduPilot: https://ardupilot.org/copter/ adresinden alındı

ArduPilot Dev Team. (2020). *Flight modes*. Mart 18, 2018 tarihinde ArduPilot: https://ardupilot.org/copter/docs/flight-modes.html adresinden alındı

Bhavesh, V. A. (2015). Comparison of various obstacle avoidance algorithms. *International Journal of Engineering Research & Technology (IJERT)*, 629-632.

Borenstein, J., & Koren, Y. (1989). Real-time obstacle avoidance for fast mobile robots in cluttered environments. *IEEE Transactions on Systems Man and Cybernetics*, 1179-1187.

Borenstein, J., & Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation* , 278-288.

Brand, M., Masuda, M., Wehner, N., & Yu, X.-H. (2010). Ant colony optimization algorithm for robot path planning. *2010 International Conference On Computer Design and Applications* (s. 436-440). Qinhuangdao: IEEE.

Choset, H., Lynch, K., Hutchinson, S., Kantor, G. A., Burgard, W., Kavraki, L. E., & Thrun, S. (2005). *Principles of robot motion:theory, algorithms, and implementations.* Cambridge: MIT Press.

Daniel, K., Nash, A., Koenig, S., & Felner, A. (2010). Theta*: Any-angle path planning on grids. *Journal of Artificial Intelligence Research*, 533-579.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 269-271.

Dorigo, M., & Thomas, S. (2004). *Ant colony optimization.* Cambridge: MIT Press.

Dudek, G., & Jenkin, M. (2000). *Computational principles of mobile robotics.* Cambridge: Cambridge University Press.

Englot, B., & Hover, F. (2011). Multi-goal feasible path planning using ant colony optimization. *2011 IEEE International Conference on Robotics and Automation* (s. 2255-2260). Şangay: IEEE.

Filippis, L., & Guglieri, G. (2012). Advanced graph search algorithms for path planning of flight vehicles. *Recent Advances in Aircraft Technology*, 159-192.

Filippis, L., Guglieri, G., & Quagliotti, F. (2012). Path planning strategies for UAVs in 3D environments. *Journal of Intelligent and Robotic Systems*, 247-264.

Gigras, Y. (2012). Ant colony based path planning algorithm for autonomous robotic vehicles. *International Journal of Artificial Intelligence & Applications*, 31-38.

Goodrich, M. A. (2002). Computer Science Lecture Notes. *Potential fields tutorial*.

Hart, P., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics 4*, 100–107.

Hokuyo Automatic Co. Ltd. (2012). Scanning laser range finder UTM-30LX/LN specification. Osaka: Hokuyo Automatic Co. Ltd.

Hwangbo, M., Kuffner, J., & Kanade, T. (2007). Efficient two-phase 3D motion planning for small fixed-wing UAVs. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 1035-1041.

Joseph, L. (2015). *Mastering ROS for robotics.* Birmingham: Packt Publishing.

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

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, 500-505.

Koubaa, A., Bennaceur, H., Chaari, I., Trigui, S., Ammar, A., Sriti, M.-F., . . . Javed, Y. (2018). Introduction to mobile robot path planning. *Robot Path Planning and Cooperation: Foundations, Algorithms and Experimentations* (s. 3-12). içinde Cham: Springer International Publishing.

Koubaa, A., Bennaceur, H., Chaari, I., Trigui, S., Ammar, A., Sriti, M.-F., . . . Javed, Y. (2018). *Robot Path Planning and Cooperation.* Cham: Springer.

Latombe, J.-C. (1991). *Robot motion planning.* Boston: Springer.

Lavalle, S. M. (1998). *Rapidly-exploring random trees: a new tool for path planning.* Iowa: Iowa Eyalet Üniversitesi.

LaValle, S. M. (2006). *Planning algorithms.* Cambridge: Cambridge University Press.

Liao, T. (2012). RPAS collision avoidance using A* algorithm. *Yüksek Lisans Tezi*. Alabama: Auburn Üniversitesi.

Lin, Y., & Saripalli, S. (2015). Sense and avoid for Unmanned Aerial Vehicles using ADS-B. *IEEE International Conference on Robotics and Automation (ICRA)* (s. 6402-6407). Seattle: IEEE.

Liu, Y. (2017). Development of a controller and indoor flight test experiment for research on quadrotor gust response. *Yüksek Lisans Tezi*. Philadelphia: Pennysylvania Eyalet Üniversitesi.

Ma, G., Duan, H., & Liu, S. (2007). Improved ant colony algorithm for global optimal trajectory planning of UAV under complex environment. *International Journal of Computer Science & Applications* , 57-68.

Mandava, R. K., Bondada, S., & Vundavilli, P. R. (2019). An optimized path planning for the mobile robot using potential field method and PSO algorithm. *Soft Computing for Problem Solving* (s. 139-150). içinde Varşova: Springer.

Marin-Plaza, P., Hussein, A., Martin, D., & Escalera, A. (2018). Global and local path planning study in a ROS-based research platform for autonomous vehicles. *Journal of Advanced Transportation*, Wiley-Hindawi.

Mavros Wiki. (2018, Mart 3). Mart 18, 2020 tarihinde http://wiki.ros.org/mavros adresinden alındı

Moravec, H., & Elfes, A. (1985). High resolution maps from wide angle sonar. *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, 116-121.

Nilsson, N. J. (1969). Mobile automation: an application of artificial intelligence techniques. *Proc. 1st Int. Joint Conf. Artificial Intelligence*, 509-520.

Open Source Robotics Foundation. (2014). *Gazebo robot simulator*. Mart 18, 2019 tarihinde http://www.gazebosim.org/ adresinden alındı

Öcal, İ. (1990). Joint trajectory methods for a manipulator and applications for stanford manipulator. *Yüksek Lisans Tezi*. Orta Doğu Teknik Üniversitesi: Fen Bilimleri Enstitüsü.

Paul, T., Krogstad, T. R., & Gravdahl, J. T. (2008). Modelling of UAV formation flight using 3D potential field. *Simulation Modelling Practice and Theory*, 1453-1462.

Pyo, Y., Kurazume, R., & Watanabe, Y. (2015). *ROS robot programming.* ROBOTIS Co.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., . . . Ng, A. (2009). ROS: an open-source Robot Operating System. *ICRA Workshop on Open Source Software.* Kobe.

Rashid, R., Perumal, N., Elamvazuthi, I., Tageldeen, M. K., Khan, M. A., & Parasuraman, S. (2016). Mobile robot path planning using ant colony optimization. *2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation* (s. 1-6). Ipoh: IEEE.

ROS Navigation. (2020, Eylül 14). Mart 18, 2020 tarihinde http://wiki.ros.org/navigation adresinden alındı

ROS Wiki. (2020, Haziran 11). Mart 18, 2020 tarihinde http://wiki.ros.org adresinden alındı

Russel, S., & Norvig, P. (2010). *Artificial intelligence: a modern approach.* New Jersey: Prentice Hall.

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2005). *Robot modelling and control.* Wiley.

Ulrich, I., & Borenstein, J. (1998). VFH+: reliable obstacle avoidance for fast mobile robots. *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, 1572-1577.

Ulrich, I., & Borenstein, J. (2000). VFH*: local obstacle avoidance with look-ahead verification. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, 2505-2511.

Van Breda, R., & Smit, W. J. (2016). Applicability of vector field histogram star (vfh*) on multicopters. *International Micro Air Vehicles, Conferences.* Pekin.

Wen, Z. Q., & Cai, Z. X. (2006). Global path planning approach based on ant colony optimization algorithm. *Journal of Central South University of Technology*, 707-712.

Yolal, E. (2015). Mobil robot simülatörleri ve ileri seviyeli simülasyonlar. *Yüksek Lisans Tezi*. İstanbul Teknik Üniversitesi : Fen Bilimleri Enstitüsü.

Yufka, A., & Parlaktuna, O. (2009). Performance comparison of BUG algorithms for mobile robots. *5th International Advanced Technologies Symposium (IATS'09)*, (s. 13-15). Karabük.

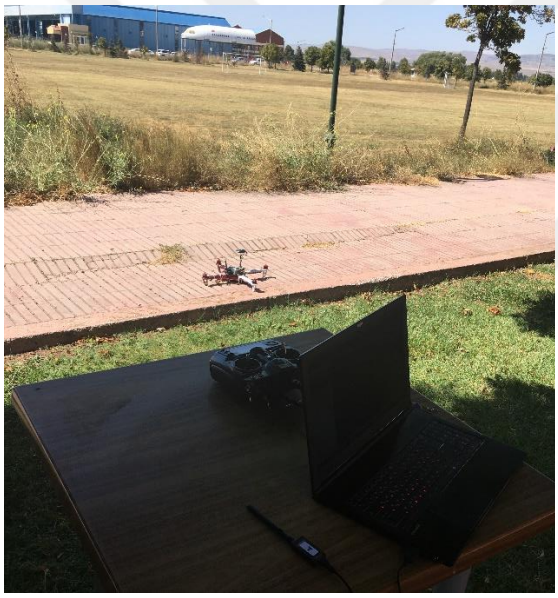
Zhang, C., Zhen, Z., Wang, D., & Li, M. (2010). UAV path planning method based on ant colony optimization. *2010 Chinese Control and Decision Conference* (s. 3790-3792). Xuzhou: IEEE.

Zhao, L. (2015). 3D obstacle avoidance for unmanned autonomous system. *Doktora Tezi*. Reno: Nevada Üniversitesi.

EKLER

EK-1. Kısıtlar

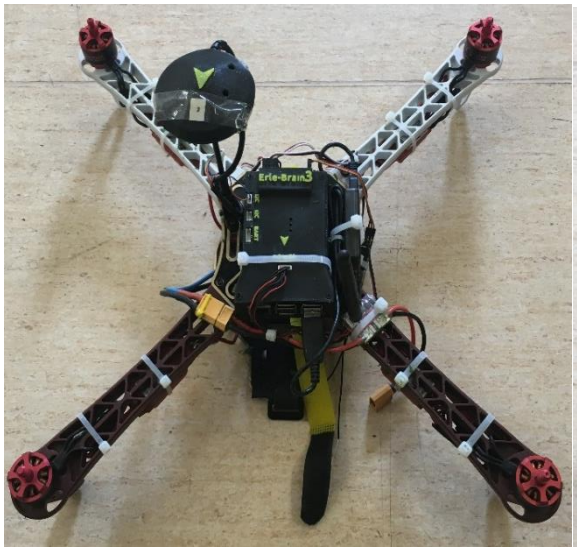
Tez çalışması süresince gerçek ortam testleri de planlanmıştır. Bu amaçla Eskişehir Teknik Üniversitesi, Havacılık ve Uzay Bilimleri Fakültesi, Havacılık Elektrik ve Elektroniği Bölümü bünyesinde bulunan Kontrol ve Aviyonik Laboratuvarı'nda yer alan ErleCopter hava aracının konfigürasyonuna uygun model Gazebo ortamında kurulmuştur. ErleCopter'in üzerinde açık kaynak kodlu, Linux işletim sistemi tabanlı otopilot kartı ErleBrain mevcuttur. ErleBrain otopilotuna kurulu otopilot yazılımını da simule edebilmek için otopilotun Gazebo eklentisi de kurulmuştur. Otopilot kartı Linux tabanlı olduğu için ROS çalıştırılabilir ve Mavros paketi ile hava aracı ile yerden iletişim sağlanabilir.

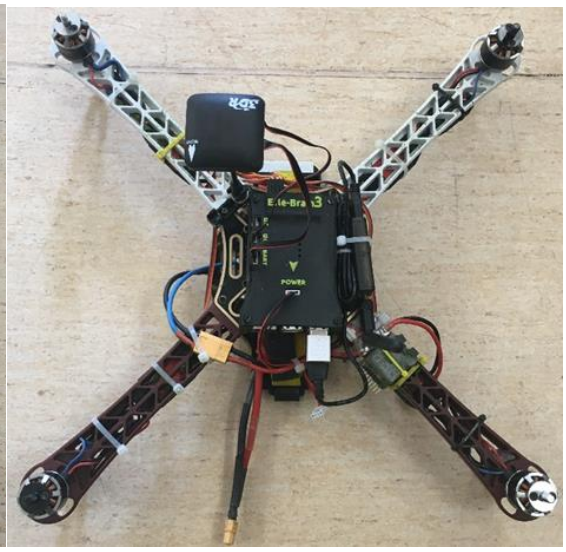


Algoritma çalışmalarını belirli seviyeye getirdikten sonra donanıma geçilmiştir. Hava aracının kurulumu ve kalibrasyonu yapılmıştır. Sonrasında yapılan uçuş denemeleri esnasında çeşitli problemlerle karşılaşılmıştır. Bunlar;

- Uçuş esnasında GPS'in uydularla bağlantısının anlık olarak kopması dolayısıyla istenen uçuş modunda (GPS gerektiren ve kodu çalıştırabileceğimiz) uçuş yapılamamıştır. Uçuş testlerinde istenen moda ya geçilememiştir ya da o moda geçince hava aracı kendini 'Fly Away'e almıştır. 'Fly Away', hava aracının kontrolsüzce uçup gitmesi anlamına gelmektedir. 'Fly Away'e neden olan durumlara bakıldığında;
  - uçuş denetleyicisi ile bağlantının kesilmesi,
  - elektromanyetik girişimler,
  - düşük batarya,
  - kötü hava koşullarında uçuş,
  - hava aracının kontrol merkezinden çok uzakta uçurulması gibi olasılıklar sebep gösterilmiştir.

Bunlardan mevcut duruma en uygun çözüm GPS-Pusula modülünün değişimi olarak görülmüştür ve aşağıdaki şekilde görüldüğü gibi modülün değişimi yapılmıştır ancak değişime rağmen, GPS'ten konum alma sorunu devam etmiştir.





- Mevcut bataryaların hava aracının görevini tamamlaması için gereken süre kadar uçuşa yeterli olmadığı görülmüştür.

Belirtilen problemlerin bütçe ve zaman kaynaklı kısıtlar dolayısıyla uçuş testleri sonlandırılmıştır.

ÖZGEÇMİŞ

Genel Bilgiler:

Adı Soyadı: Demet CANPOLAT TOSUN

Yabancı Dili: İngilizce

Öğrenim Durumu:

Doktora: Eskişehir Teknik Üniversitesi / Lisansüstü Eğitim Enstitüsü / Havacılık Elektrik ve Elektroniği Anabilim Dalı / 2021

Yüksek Lisans: Anadolu Üniversitesi / Fen Bilimleri Enstitüsü / Havacılık Elektrik ve Elektroniği Anabilim Dalı / 2015

Lisans: İnönü Üniversitesi / Mühendislik Fakültesi / Elektrik-Elektronik Mühendisliği Bölümü / 2011

Mesleki Geçmişi:

2011-2018, Araştırma Görevlisi, Anadolu Üniversitesi, Havacılık ve Uzay Bilimleri Fakültesi, Havacılık Elektrik ve Elektroniği Bölümü.

2018-Halen, Araştırma Görevlisi, Eskişehir Teknik Üniversitesi, Havacılık ve Uzay Bilimleri Fakültesi, Havacılık Elektrik ve Elektroniği Bölümü.

Uluslararası Hakemli Dergilerde Yayımlanan Makaleler:

1. Korul Hakan, Işık Yasemin, Canpolat Tosun Demet (2015). Bir Uçağın Yatış Durumu Kontrol Sisteminde Bulanık PD Denetleyici Performansının İncelenmesi. Electronic Journal Of Occupational Improvement And Research, 2(3), 10-16.

2. Canpolat Tosun Demet, Işık Yasemin, Korul Hakan (2015). Comparison of PID and LQR controllers on a quadrotor helicopter. International Journal of Systems Applications, Engineering & Development, 9, 136-143.

Uluslararası Bilimsel Toplantılarda Sunulan ve Bildiri Kitaplarında (Proceedings) Basılan Bildiriler:

1. Korul Hakan, Canpolat Tosun Demet, Işık Yasemin (2015). A Model Reference Adaptive Controller Performance of an Aircraft Roll Attitude Control System. 10th International Conference on Dynamical Systems and Control (Tam Metin Bildiri)

2. Canpolat Tosun Demet, Işık Yasemin, Korul Hakan (2015). LQR Control of a Quadrotor. 2015 International Conference on Applied Physics, Simulation and Computers (Tam Metin Bildiri)

3. Canpolat Tosun Demet, Işık Yasemin (2020). Observer-Based Fault Diagnosis of An Aircraft. International Symposium on Electric Aviation and Autonomous Systems 2020, International Symposium on Aircraft Technology, MRO & Operations 2020, International Course on Unmanned Aerial Vehicle 2020, 22-24 Eylül 2020, Kiev, Ukrayna / Online (Tam Metin Bildiri)

Ulusal Bilimsel Toplantılarda Sunulan ve Bildiri Kitaplarında Basılan Bildiriler:

1. Korul Hakan, Işık Yasemin, Canpolat Tosun Demet (2015). Fuzzy PD Controller Performance of an Aircraft Roll Attitude Control System. 1st. International Workshop on Construction and Electricity Applications on Vocational Education (IWCEA 2015) (Tam Metin Bildiri/Sözlü Sunum) (Yayın No:3401209)