ÖZ

Çoklu İşbirlikçi Sualtı İnsansız Araçların Otonom Kontrol Algoritmaları

Ömer Faruk Çelik

STAJ

FIRAT ÜNİVERSİTESİ

Mühendislik Fakültesi Bilgisayar Mühendisliği

Yapay Zekâ/Robotik

2025, Sayfa:52

Bu staj projesinin temel amacı, çoklu otonom sualtı araçlarının (AUV) dinamik, kısmen bilinmeyen ve iletişim kısıtlarının bulunduğu zorlu sualtı ortamlarında etkili ve güvenilir görev gerçekleştirebilmesini sağlayacak hibrit bir kontrol mimarisi tasarlamak ve geliştirmektir. Günümüzde mevcut literatürde yer alan yaklaşımlar genellikle ya önceden tanımlanmış, statik haritalara dayanarak planlama yapmakta ya da anlık ve reaktif kararlar vererek stratejik bir öngörüden yoksun kalmaktadır. Bu durum, gerçek dünya operasyonlarında karşılaşılan belirsizlikler ve çevresel değişkenlikler karşısında yetersiz kalmakta ve AUV'ların görev başarısını olumsuz yönde etkilemektedir.

Bu çalışma ise, her bir otonom aracın kendi zekâsını kullanarak çevresel verileri analiz edip karar verebildiği, aynı zamanda filo düzeyinde koordinasyonun sağlandığı hiyerarşik çoklu-agent mimarisini önermektedir. Böylece, hem global planlama hem de lokal kontrol katmanları arasında etkin bir etkileşim kurulmakta; bilinmeyen engellerin tespiti ve dinamik ortam koşullarına hızlı adaptasyon mümkün hale gelmektedir. Bu mimari, çoklu AUV sistemlerinin otonomluğunu artırarak, iletişim kısıtları ve çevresel belirsizliklere rağmen görevlerini başarılı ve güvenli biçimde tamamlamalarını sağlamayı hedeflemektedir. Tasarlanan sistem, simülasyon ortamında test edilerek gerçek dünyadaki uygulamalar için önemli bir temel oluşturmaktadır.

Anahtar Kelimeler: Çoklu otonom sualtı araçları, Hibrit kontrol mimarisi, Rota planlama, A* algoritması, Pekiştirmeli öğrenme

İÇİNDEKİLER

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

KISALTMALAR VE SEMBOLLER LİSTESİ

- **B** : Kontrol Matrisi (Kontrol girdilerini sistem dinamiğine bağlar)
- **f(n)** : A* algoritmasında bir düğümün toplam tahmini maliyeti
- **g(n)** : A* algoritmasında başlangıçtan n düğümüne kadar olan gerçek maliyet
- **h(n)** : A* algoritmasında n düğümünden hedefe kadar olan tahmini (sezgisel) maliyet
- **J** : Toplam maliyet fonksiyonu
- **Q** : Durum hataları için ağırlık matrisi
- **R** : Kontrol eforu için ağırlık matrisi
- **s** : Pekiştirmeli öğrenmede anlık durum (state)
- **u** : Pekiştirmeli öğrenmede anlık eylem (action)
- **U** : Sistemin genel kontrol vektörü
- $\mathbf{u_i}$: i'inci AUV'nin kontrol vektörü (kuvvet/tork)
- **X** : Sistemin genel durum vektörü
- $\mathbf{x_i}$: i'inci AUV'nin durum vektörü
- $\dot{\mathbf{x}}$: Durum vektörünün zaman türevi (hızlar ve ivmeler)
- **α (Alfa)** : Yapay zekâ modelinde öğrenme oranı (learning rate)
- **γ (Gama)** : Pekiştirmeli öğrenmede gelecekteki ödüller için indirgeme faktörü (discount factor)
- **ε (Epsilon)**: Pekiştirmeli öğrenmede keşif (exploration) oranı

- **ACO**: Ant Colony Optimization (Karınca Kolonisi Optimizasyonu)
- **API**: Application Programming Interface (Uygulama Programlama Arayüzü)
- **AUV**: Autonomous Underwater Vehicle (Otonom Sualtı Aracı)
- **CSV**: Comma-Separated Values (Simülasyon verilerini kaydetmek için kullanılan dosya formatı)
- **DoF**: Degrees of Freedom (Serbestlik Derecesi, örn. BlueROV2 için 6-DoF)
- **GA**: Genetic Algorithm (Genetik Algoritma)
- **GPU**: Graphics Processing Unit (Grafik İşlem Birimi, YZ model eğitimini hızlandırmak için)
- **GUI**: Graphical User Interface (Grafiksel Kullanıcı Arayüzü)
- **HJB**: Hamilton-Jacobi-Bellman
- **IMU**: Inertial Measurement Unit (Ataletsel Ölçüm Birimi)
- **MAE**: Mean Absolute Error (Ortalama Mutlak Hata, bir performans metriği)
- **MAS**: Multi-Agent System (Çoklu-Agent Sistemi)
- **MAUV**: Multiple Autonomous Underwater Vehicles (Çoklu Otonom Sualtı Araçları)
- **MPC**: Model Predictive Control (Model Öngörülü Kontrol)
- **RL**: Reinforcement Learning (Pekiştirmeli Öğrenme)
- **RMSE**: Root Mean Squared Error (Kök Ortalama Kare Hatası, rota takip hassasiyeti için)
- **ROS**: Robot Operating System (Robot İşletim Sistemi)
- **SONAR**: Sound Navigation and Ranging (Sesle Navigasyon ve Mesafe Tespiti)
- **TF**: Transform Framework (ROS'ta farklı koordinat sistemleri arasındaki dönüşümleri yöneten yapı)
- **URDF**: Unified Robot Description Format (ROS'ta robot modelini tanımlayan dosya formatı)

BİRİNCİ BÖLÜM

I. GİRİŞ

Otonom sualtı sistemleri, gezegenimizin en büyük ve en az keşfedilmiş bölgesi olan okyanusların sırlarını açığa çıkarmak ve sualtı operasyonlarını daha güvenli ve verimli hale getirmek için vazgeçilmez bir teknoloji olarak öne çıkmaktadır. Bu sistemlerin geliştirilmesi, bilimsel keşiflerden endüstriyel verimliliğe, çevresel korumadan ulusal güvenliğe kadar geniş bir yelpazede stratejik bir önem taşımaktadır. Bu açılış bölümü, projenin temelini oluşturan bu vizyonu ve bu vizyona ulaşmanın önündeki engelleri sistematik bir şekilde ele almaktadır. İlk olarak, çoklu robot sistemlerinin sualtı ortamında karşılaştığı temel zorluklar ortaya konarak projenin problem tanımı netleştirilecektir.

Problemin tanımlanmasının ardından, bu zorlukların üstesinden gelmek için belirlenen projenin amacı, kapsamı ve önemi detaylandırılacaktır. Bu temeller üzerine, raporun literatüre sunduğu özgün katkı olan A* yol planlaması ve Pekiştirmeli Öğrenme ile zenginleştirilmiş hibrit çoklu-agent mimarisi fikri sunulacaktır. Son olarak, projenin bilimsel çerçevesini sağlamlaştırmak amacıyla çalışmanın dayandığı varsayımlar ve bilinçli olarak dışında bırakılan sınırlılıklar belirtilecek ve metin boyunca kullanılacak anahtar terimlerin tanımları yapılacaktır. Bu yapı, okuyucuya projenin neden önemli olduğunu, neyi hedeflediğini ve bu hedefe nasıl bir yaklaşımla ulaşmayı planladığını adım adım gösteren bir yol haritası sunmaktadır.

1.1. Problem Tanımı: Sualtı Ortamında Otonom Sistemlerin Zorlukları

Otonom sualtı araçları (AUV), okyanus keşfi, deniz tabanı haritalama, endüstriyel tesis denetimi ve savunma uygulamaları gibi alanlarda insan erişiminin zor ve tehlikeli olduğu görevler için kritik bir teknoloji haline gelmiştir. Ancak, tek bir aracın görev kapasitesi ve operasyonel alanı sınırlıdır. Bu sınırlılığı aşmak amacıyla, birden fazla aracın işbirliği içinde çalıştığı çoklu AUV (MAUV) sistemleri geliştirilmektedir. Bu sistemler, görevleri daha hızlı tamamlama ve sistem dayanıklılığını artırma gibi önemli avantajlar sunsa da, sualtı ortamının doğasından kaynaklanan temel zorluklarla karşı karşıyadır.

Sualtı ortamı; elektromanyetik dalgaların etkin bir şekilde yayılamaması nedeniyle GPS gibi küresel konumlandırma sistemlerinin kullanılamadığı, iletişimin ise düşük bant genişliğine, yüksek gecikmeye ve sık kesintilere maruz kalan akustik sinyallerle sınırlı olduğu zorlu bir arenadır. Bu iletişim kısıtları, araçlar arasında anlık ve güvenilir bir koordinasyon kurmayı son derece güçleştirmektedir. Literatürdeki mevcut kontrol ve planlama yaklaşımları genellikle bu zorluklar karşısında yetersiz kalmaktadır. Statik planlama yöntemleri (örn. A*), ortamın önceden bilindiğini varsayar ve dinamik, yani aniden ortaya çıkan olaylara adapte olamaz. Davranış tabanlı veya saf pekiştirmeli öğrenme gibi reaktif yöntemler ise anlık durumlara hızlı tepki verebilmelerine rağmen, genellikle stratejik bir öngörüden ve global bir optimallikten yoksundur. Bu durum, sualtı sistemlerinin hem stratejik plan yapabilen hem de bu planı dinamik koşullara göre anlık olarak adapte edebilen hibrit bir kontrol mekanizmasına ihtiyaç duyduğunu açıkça ortaya koymaktadır.

1.2. Araştırmanın Önemi ve Motivasyon

Bu projenin önemi, literatürdeki planlama-tabanlı ve reaktif-tabanlı yaklaşımlar arasındaki boşluğu doldurarak, otonom sualtı sistemlerini daha akıllı, esnek ve güvenilir hale getirme potansiyelinde yatmaktadır. Bu projede geliştirilmesi hedeflenen hibrit kontrol mimarisi, sualtı robotlarının sadece önceden programlanmış komutları izleyen araçlar olmaktan çıkıp, karşılaştıkları beklenmedik durumlara karşı akıllıca çözümler üretebilen otonom varlıklara dönüşmesine katkı sağlamayı amaçlamaktadır.

Bu projenin motivasyonu, sualtı operasyonlarının verimliliğini ve güvenliğini artırmaktır. Geliştirilecek sistemin, boru hattı denetimi gibi endüstriyel görevlerin maliyetini düşürmesi, arama-kurtarma gibi kritik operasyonların başarı oranını artırması ve oşinografik veri toplama gibi bilimsel keşiflerin kapsamını genişletmesi hedeflenmektedir. En temel motivasyon, sualtı görevlerinde insan faktörünü ve riskini en aza indirerek tam otonom operasyonların önünü açmaktır.

1.3. Araştırmanın Amacı ve Kapsamı

Bu projenin temel amacı, çoklu otonom sualtı araçları için global rota planlama yeteneği ile yerel dinamik adaptasyon kabiliyetini birleştiren hibrit bir kontrol mimarisi tasarlamak, geliştirmek ve bu mimarinin etkinliğini simülasyon ortamında doğrulamaktır.

Bu amaç doğrultusunda, projenin kapsamı aşağıdaki şekilde belirlenmiştir:

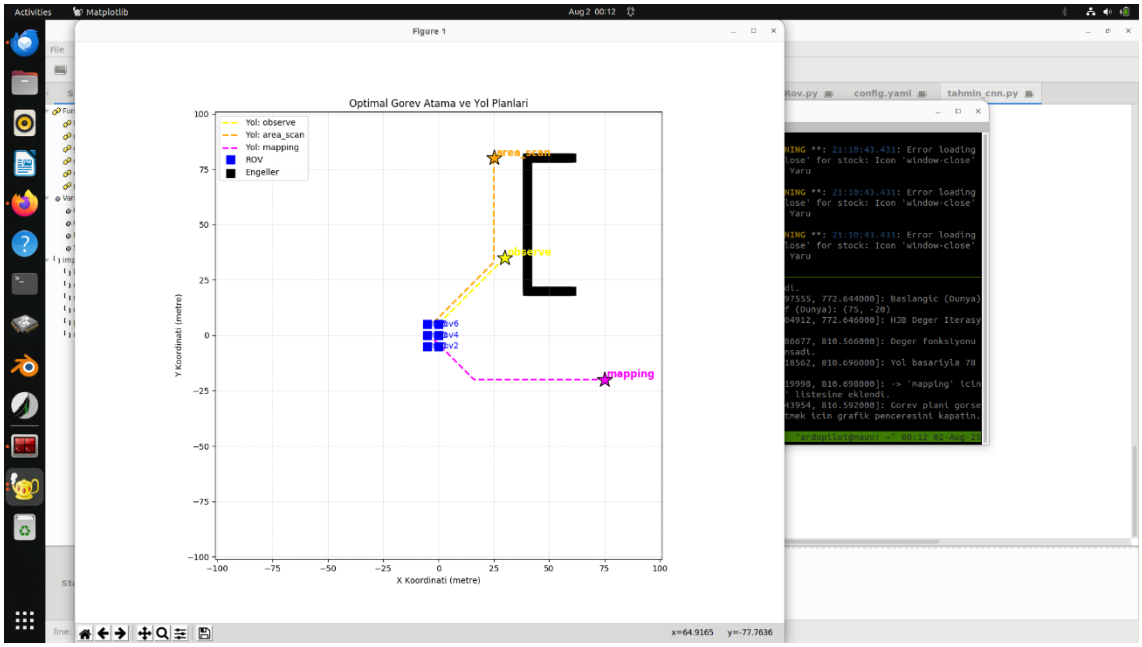
- Sistemdeki her bir AUV, kendi kararlarını alabilen bir "akıllı agent" olarak modellenecektir.
- A* algoritması, bilinen harita üzerinde enerji ve mesafe maliyetini minimize eden global rotaları planlamak için kullanılacaktır.
- Pekiştirmeli Öğrenme (RL) tabanlı bir yerel karar modülü, A* tarafından oluşturulan plana uymayan anlık engellerden kaçınma gibi dinamik adaptasyon görevlerini üstlenecektir.
- Geliştirilen hibrit mimari, gerçekçi fiziksel dinamikleri içeren Robot İşletim Sistemi (ROS) ve Gazebo tabanlı bir simülasyon ortamında farklı görev senaryoları altında test edilecektir.
- Proje, kontrol ve karar verme algoritmalarının tasarımına odaklanacak, donanım tasarımı veya akustik iletişim protokolü geliştirmeyi kapsamayacaktır.

1.4. Raporun Özgün Değeri: A* Yol Planlaması ve RL ile Zenginleştirilmiş Çoklu-Agent Mimarisi

Bu raporun özgün değeri, literatürde genellikle ayrı ayrı ele alınan ve farklı problemlere çözüm sunan üç farklı karar mekanizmasını – **global stratejik planlama**, **yerel taktiksel yeniden planlama** ve **anlık adaptif refleksler** – tek ve bütünleşik bir **hiyerarşik karar mimarisi** altında birleştirmesinde yatmaktadır. Otonom sistemler alanındaki klasik ikilem, ya her şeyin önceden bilindiği varsayımıyla en verimli yolu çizen statik planlayıcılara ya da anlık durumlara tepki veren ancak büyük resmi göremeyen reaktif sistemlere dayanmaktır. Önerilen hibrit model, bu ikileme çok katmanlı ve pratik bir

çözüm sunarak, her bir otonom aracı (agent) farklı senaryolar için en uygun "düşünme" biçimini kullanabilen zeki bir varlık olarak donatır.

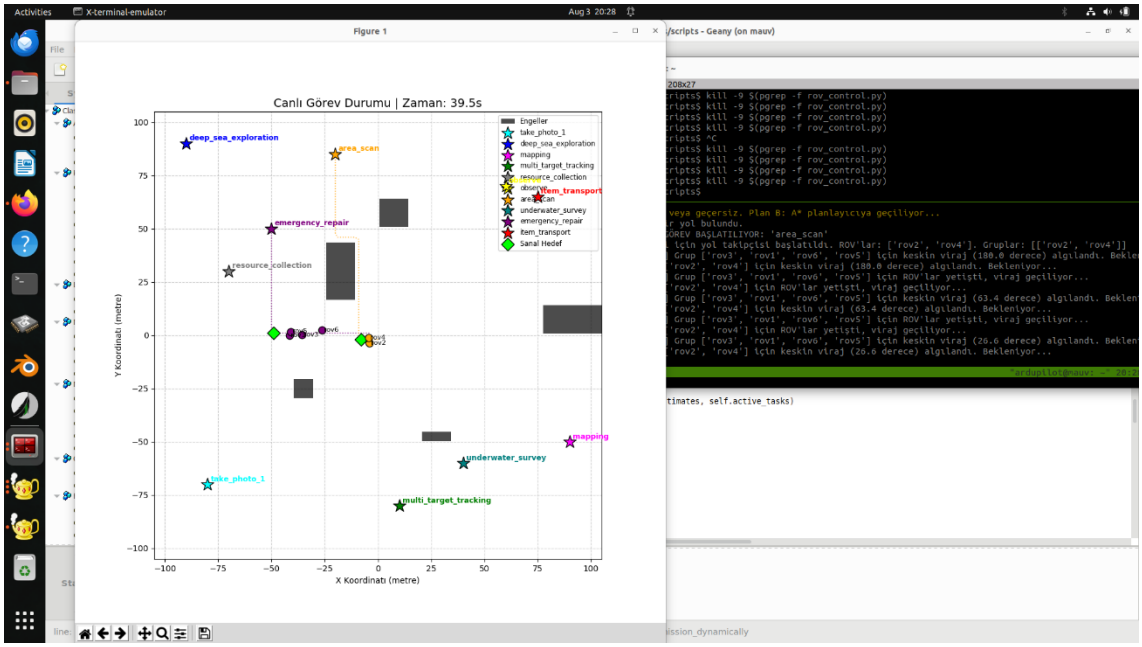
Bu projenin temelindeki sinerjik entegrasyon, sistemin sadece görevini en verimli şekilde tamamlamasını değil, aynı zamanda bunu belirsiz ve değişen sualtı ortamlarının tüm zorluklarına karşı dayanıklı bir şekilde yapmasını hedefler. Bu sayede, her bir yaklaşımın tekil kullanımının ötesinde, hem verimli hem de zeki bir otonom sistem performansı potansiyeli sunulmaktadır. Bu hiyerarşik yapı, aşağıda sunulan şekillerle kavramsal olarak açıklanmaktadır.



Şekil 1.4.1. Global Stratejik Planlama Katmanının Kavramsal Gösterimi (A*)

Şekil 1.4.1'de, önerilen mimarinin en üst katmanı olan **stratejik planlama** gösterilmektedir. Bu katman, sistemin "uzun vadeli stratejisti olarak görev yapar. Görev başlamadan önce, mevcut harita bilgileri ve bilinen statik engeller kullanılarak, A* algoritması aracılığıyla başlangıç noktasından hedefe giden global olarak en optimal rota hesaplanır. Bu rota, enerji tüketimini ve toplam mesafeyi minimize ederek görevin genel verimlilik çerçevesini çizer.

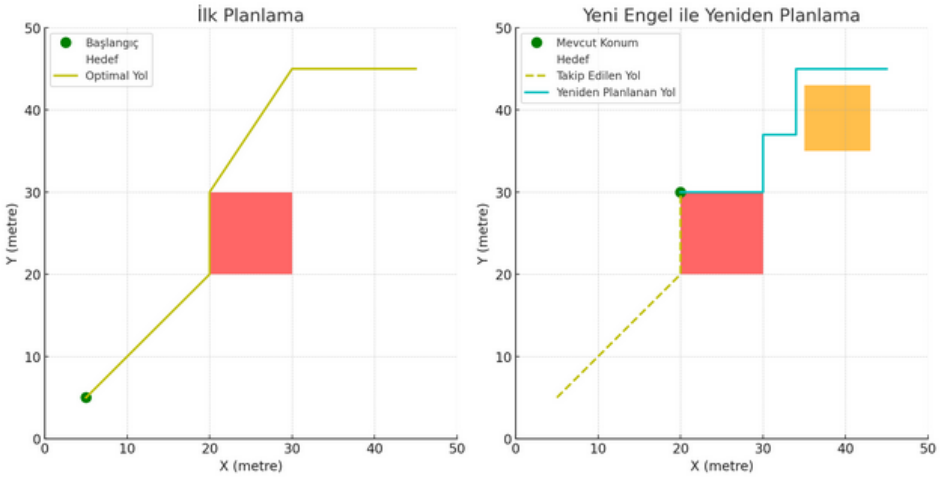
Bu plan, sistemin tüm operasyonunun temelini oluşturur ve araçların rastgele hareket etmek yerine, kanıtlanabilir şekilde en verimli genel yolu takip etmesini sağlar. Bu katman, "Göreve en iyi nasıl başlanır ve bitirilir?" sorusuna cevap verir, ancak tek başına çevrenin tamamen bilindiği varsayımına dayanır.



Şekil 1.4.2. Yerel Taktiksel Planlama Katmanının Devreye Girmesi (Yerel Yol Planlayıcı)

Şekil 1.4.2, mimarinin pratik zekasını ortaya koyan **taktiksel yeniden planlama katmanını** sergilemektedir. Gerçek dünya görevlerinde, başlangıçtaki haritanın her zaman eksiksiz olması beklenemez. Bu şekilde gösterildiği gibi, agent global rota üzerinde ilerlerken, sensörleri aracılığıyla önceden haritada olmayan, büyük ve karmaşık bir statik engel (örneğin yeni oluşmuş bir kum yığını veya bilinmeyen bir batık) tespit eder. Bu durumda basit bir kaçınma manevrası yetersiz kalacaktır.

İşte bu noktada, yerel yol planlayıcı devreye girer. Agent, ana rotayı terk eder ve bu yeni engelin etrafından dolanarak tekrar ana rotaya en verimli şekilde dönecek bir **yerel detour (sapma) rotası** hesaplar. Bu katman, "Karşıma çıkan bu büyük ve bilinmeyen engeli en akıllıca nasıl aşarım?" sorusuna cevap vererek, sistemin keşif yeteneğini ve bilinmeyenlere karşı dayanıklılığını artırır



Şekil 1.4.3. Anlık Adaptif Refleks Katmanının Devreye Girmesi (RL).

Şekil 1.4.3, hiyerarşinin en hızlı ve en reaktif katmanı olan **anlık adaptif refleksleri** göstermektedir. Bu katman, "anlık tehlikelere karşı hayatta kalma içgüdüsü" olarak işlev görür. Şekilde görüldüğü gibi, agent (global veya yerel) rotasını takip ederken, aniden önüne çıkan ve planlama için zaman olmayan dinamik bir tehdit (örneğin başka bir hareketli araç veya sürüden ayrılan bir deniz canlısı) ile karşılaşır.

Bu durumda, en alt seviyedeki Pekiştirmeli Öğrenme (RL) modülü anında devreye girer. RL modülü, çarpışmayı önlemek için en hızlı ve en güvenli eylemi ("dur", "sağa kaç", "yüksel" gibi) anlık olarak uygular. Bu katman, "Tehlike! Hemen tepki ver!" komutunu yerine getirerek, sistemin en öngörülemez anlarda bile güvenliğini sağlar. Bu üç katmanlı karar mekanizması, sistemin farklı karmaşıklıktaki sorunlara farklı zekâ seviyeleriyle yanıt vermesini sağlayarak, bütünsel performansını ve görev başarı oranını önemli ölçüde artırır.

1.5. Varsayımlar ve Sınırlılıklar

Bu projenin geçerliliği ve kapsamı, aşağıdaki varsayımlar ve sınırlılıklar çerçevesinde değerlendirilmelidir:

Varsayımlar:

Araçların, gecikmeli ve kayıplı da olsa, birbirleriyle temel düzeyde akustik iletişim kurabildiği varsayılmıştır.

Her aracın kendi içsel durumunu (konum, hız, batarya seviyesi) yerleşik sensörler aracılığıyla (hatalı da olsa) bildiği kabul edilmiştir.

A* algoritmasının çalışabilmesi için ortamın genel hatlarını içeren statik bir haritanın başlangıçta mevcut olduğu varsayılmıştır.

Sınırlılıklar:

Bu proje, algoritma ve mimari tasarımına odaklanmış olup, geliştirilen sistemin doğrulaması gerçek dünya donanımları yerine yüksek sadakatli bir simülasyon ortamıyla sınırlandırılmıştır.

Sualtı akıntıları, dalgalanmalar ve karmaşık hidrodinamik etkiler gibi çevresel faktörler, modelin karmaşıklığını yönetilebilir kılmak adına başlangıçta ihmal edilmiştir.

İletişim modeli, gerçek bir akustik modemin tüm fiziksel katman özelliklerini değil, gecikme ve paket kaybı gibi temel etkileri modelleyen bir soyutlamadır.

1.6. Temel Tanımlar

- **Akıllı Agent:** Çevresini algılayan, hedeflerine ulaşmak için otonom olarak muhakeme yapan, karar alan ve eylemlerde bulunan hesaplamalı bir varlıktır. Bu rapor kapsamında her bir AUV, bir agent olarak kabul edilmektedir.

- **Çoklu-Agent Sistemi (MAS):** Belirli bir amacı gerçekleştirmek üzere birbirleriyle etkileşen birden fazla akıllı agent'tan oluşan bir sistemdir.
- **Rota Planlama (Path Planning):** Bir agent'ın veya aracın, bir başlangıç noktasından bir hedef noktasına, belirli kısıtlar altında en uygun rotanın bulunması problemidir. Bu kısıtlar, engellerden kaçınmayı, enerji tüketimini minimize etmeyi veya görev süresini kısaltmayı içerebilir. Bu raporda, hem bilinen (statik) hem de bilinmeyen (dinamik) engelleri dikkate alan bir rota planlama çözümü geliştirilmiştir.
- **A* Algoritması:** Bir başlangıç noktasından bir hedef noktasına giden en düşük maliyetli yolu bulmak için kullanılan bir grafik arama ve yol bulma algoritmasıdır.
- **Pekiştirmeli Öğrenme (RL):** Bir agent'ın, bir ortamda deneme-yanılma yoluyla hangi eylemlerin en yüksek ödülü getireceğini öğrendiği bir makine öğrenmesi alanıdır.
- **Hiyerarşik Kontrol Mimarisi:** Farklı seviyelerdeki karar verme problemlerini çözmek için katmanlar halinde organize edilmiş bir kontrol sistemidir. Bu raporda kullanılan mimari; en üstte global ve stratejik planlama, ortada taktiksel yeniden planlama ve en altta anlık refleksif kararlar olmak üzere üç katmandan oluşur.
- **Durum Uzayı (State Space):** Bir agent'ın veya sistemin içinde bulunabileceği tüm olası durumların matematiksel bir temsilidir. Bu projede bir AUV için durum; konumu, yönelimi ve hızları gibi değişkenleri içerir. Rota planlama ve karar verme algoritmaları, bu durum uzayı içerisinde en uygun eylemleri bulmaya çalışır.
- **Simülasyon:** Gerçek dünyadaki bir sürecin veya sistemin, bilgisayar üzerinde matematiksel bir modelinin çalıştırılmasıdır. Bu projede simülasyon, geliştirilen kontrol algoritmalarının, gerçek bir AUV'yi riske atmadan, kontrol edilebilir ve tekrarlanabilir senaryolar altında (ROS ve Gazebo ortamında) test edilmesi ve doğrulanması için kullanılmıştır.
- **Sezgisel Fonksiyon (Heuristic Function):** A* gibi arama algoritmalarında, bir noktadan hedefe olan tahmini en kısa yolu hesaplayan bir fonksiyondur. Etkili bir sezgisel fonksiyon, algoritmanın gereksiz yolları keşfetmesini engelleyerek çözüm süresini önemli ölçüde kısaltır.
- **Ödül Fonksiyonu (Reward Function):** Pekiştirmeli Öğrenme'de, bir agent'ın belirli bir durumda gerçekleştirdiği bir eylemin "ne kadar iyi" veya "ne kadar kötü" olduğunu belirleyen fonksiyondur. Agent'ın tüm öğrenme süreci, bu fonksiyondan aldığı geri bildirimlere göre toplam ödülü maksimize etme üzerine kuruludur.
- **Robot İşletim Sistemi (ROS):** Robot yazılımları geliştirmek için kütüphaneler ve araçlar sağlayan, esnek bir "meta-işletim sistemi"dir. Bu projede, simülasyon ortamı ile kontrol algoritmaları arasındaki iletişimi, sensör verilerinin akışını ve agent'lar arası haberleşmeyi yönetmek için kullanılmıştır.

İKİNCİ BÖLÜM

II. KURAMSAL ÇERÇEVE VE LİTERATÜR ARAŞTIRMASI

Bu bölümde, çoklu otonom sualtı araçlarının kontrolü ve koordinasyonu alanındaki temel kavramlar, tarihsel gelişim, teorik yaklaşımlar ve mevcut çalışmalar incelenmiş, bu çalışmaların ortaya koyduğu sınırlılıklar ve bu raporun konusunu oluşturan araştırma boşluğu tespit edilmiştir. Bu analiz, projenin üzerine inşa edildiği teorik zemini oluşturmaktadır.

2.1. Otonom Sualtı Araçlarının (AUV) Gelişimi ve Sınıflandırılması

Otonom sualtı araçlarının (AUV) tarihi, 1957 yılında okyanus akustiği verisi toplamak amacıyla geliştirilen SPURV (Special Purpose Underwater Research Vehicle) ile başlar [3]. Başlangıçta tekil ve basit görevler için tasarlanan bu araçlar, mikroişlemci devrimi ve yapay zekâdaki gelişmelerle birlikte evrim geçirmiştir [4][5]. 2000'li yıllara gelindiğinde, tekil AUV'lerin yerini, daha karmaşık görevleri birlikte başarmak için tasarlanmış çoklu işbirlikçi sualtı araçları (MAUV) almaya başlamıştır [6][7]. Literatürde de belirtildiği üzere, bir AUV'nin tek başına başaramayacağı görevleri, kolektif bir yapıya sahip olan MAUV sistemleri daha yüksek bir başarımla yerine getirebilmektedir [6][10].

Bu evrim, Tablo 2.1'de özetlenmiştir.

Zaman Aralığı	**Ana Odak / İtici Güç**	**Teknolojik Özellik**	**Örnek Araç**
1957	Okyanus Akustiği (Askeri)	İlk konsept, mekanik otopilot	SPURV
1960 – 1980	Soğuk Savaş (Askeri)	Analog sistemler, ilkel otonomi	EPAULARD
1980 - 1990	Bilimsel Araştırma	Mikroişlemci, daha akıllı görev planlaması	ABE, REMUS
2000 - 2020	Ticarileşme ve Gözlem	Minyatürleşme, yapay zekâ, sürü sistemi, glider	Slocum Glider, Bluefin-21
2020 - Günümüz	Otonomi ve Veri Temelli Keşif	Derin öğrenme, RL, HJB, işbirlikçi karar yapıları	Iver4-900 AUV, Hydroid REMUS NG

Tablo 2.1. Tarih Boyunca AUV Gelişimi

Tablo 2.1'de de görüldüğü üzere, AUV'lerin gelişimi, dönemin ihtiyaçları ve teknolojik imkânları doğrultusunda şekillenmiştir [4][6]. Bu evrim, günümüzde yapay zekâ algoritmalarının AUV sistemlerine entegre edilmesiyle en büyük dönüşümünü yaşamaktadır [15][16]. Bu teknolojik sıçramanın en çarpıcı yönü ise, tekil araçlardaki gelişmeden ziyade, çoklu AUV sistemlerinin kolektif bir zekâyla donatılmasıdır [10][13].

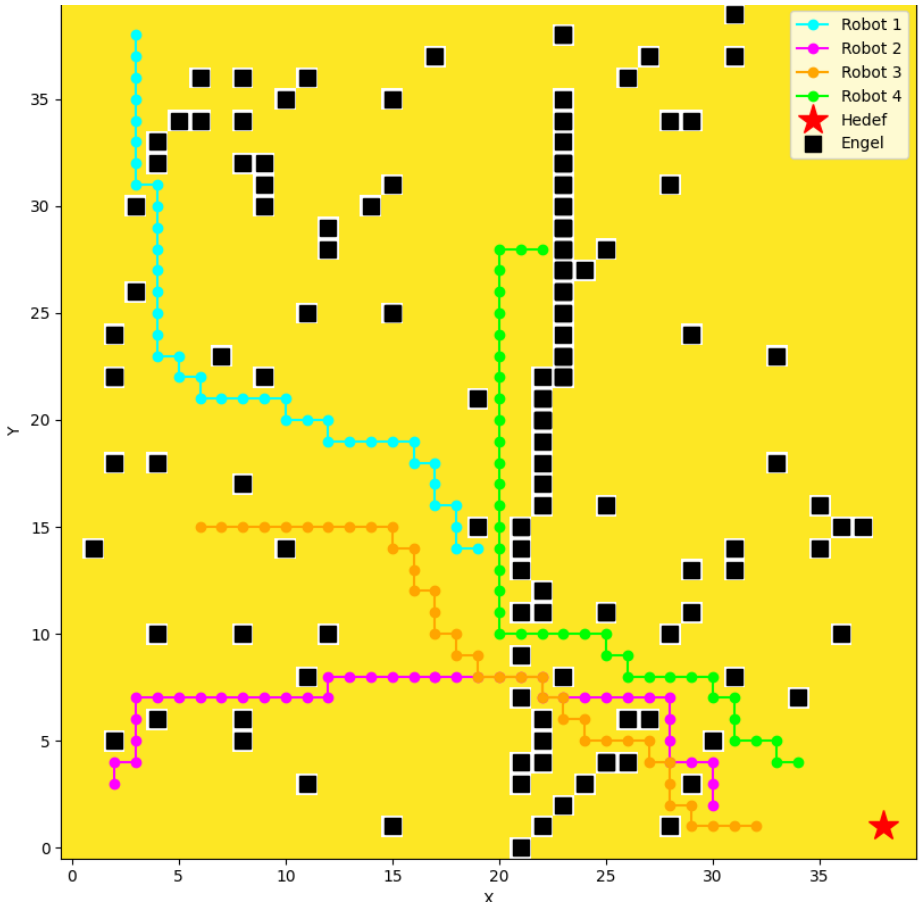
Bu durum, çoklu araçların koordinasyonu problemini merkezi bir araştırma konusu haline getirmiştir [6][11].

2.2. Çoklu AUV Sistemleri İçin Temel Kontrol Stratejileri

Çoklu AUV sistemlerinin koordinasyonu, temel olarak rota planlama ve anlık karar verme problemlerine çözüm bulmayı gerektirir [8][14]. Literatürde bu problemlere yönelik farklı felsefeleri benimseyen temel yaklaşımlar bulunmaktadır [7][10].

2.2.1. Grafik Tabanlı Yöntemler: A* Algoritması ve Varyasyonları

Grafik tabanlı yöntemler, ortamı bir dizi düğüm ve bu düğümleri birbirine bağlayan kenarlardan oluşan bir yapı olarak modeller ve bu yapı üzerinde en kısa yolu arar [17]. Bu alandaki en bilinen algoritmalardan biri, Şekil 2.1'de prensibi gösterilen A* (A-Star) algoritmasıdır [17].



Şekil 2.1. A* Algoritmasının Statik Ortamda Rota Planlama Prensibi.

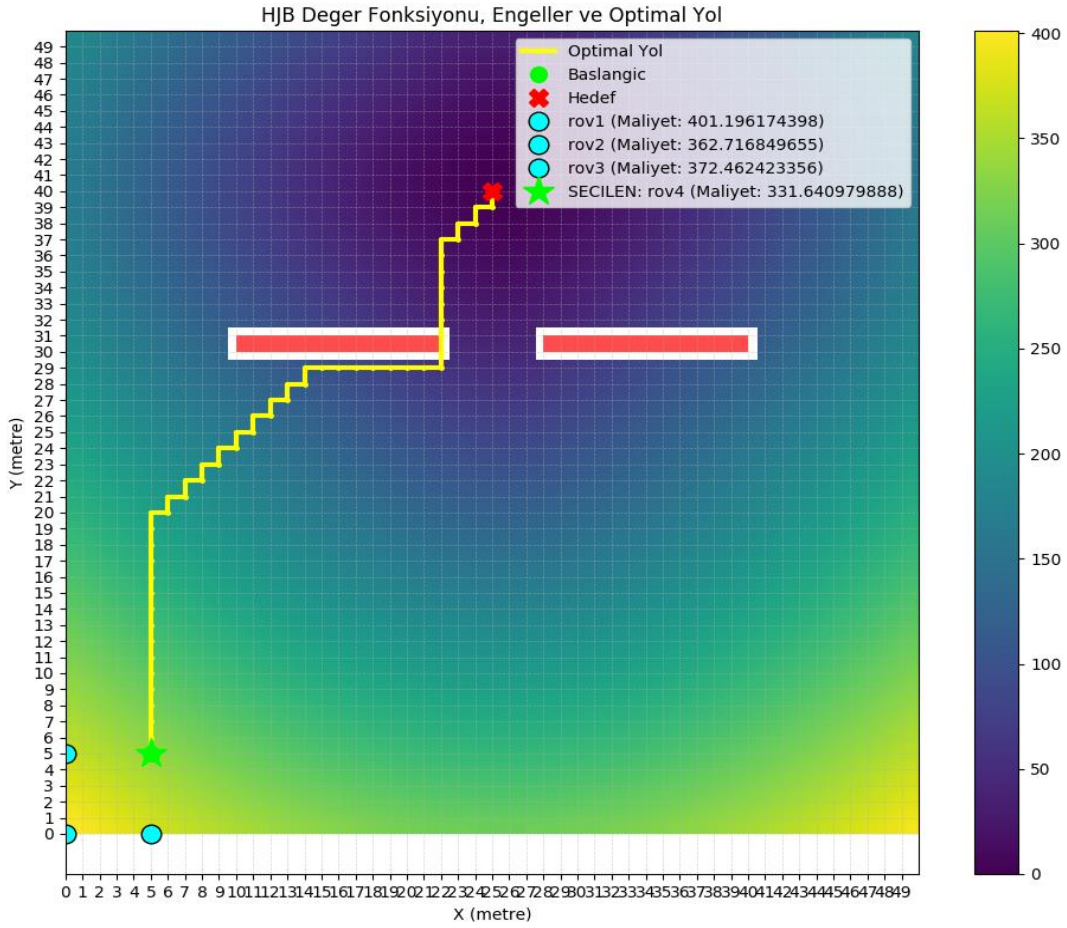
Şekil 2.1'de görüldüğü gibi, A* algoritması bir başlangıç ve hedef noktası arasında, bilinen statik engellerin etrafından dolaşan en düşük maliyetli rotayı hesaplar. Bunu, her adımda f(n) = g(n) + h(n) formülünü kullanarak yapar; burada g(n) başlangıçtan o anki noktaya kadar olan gerçek maliyeti, h(n) ise o noktadan hedefe olan tahmini (sezgisel)

maliyeti ifade eder [17][18]. Bu yaklaşımın temel avantajı, belirli koşullar altında matematiksel olarak en optimal yolu bulmayı garanti etmesidir [17].

Ancak bu yöntemin en büyük kısıtlaması, haritanın ve üzerindeki tüm engellerin önceden bilinmesi gerektiği varsayımıdır [18]. Görev sırasında ortaya çıkan dinamik veya bilinmeyen engeller karşısında A* tek başına bir çözüm sunamaz ve bu durum, algoritmanın gerçek dünya uygulamalarındaki esnekliğini sınırlar [18][19]. Bu nedenle, A* genellikle hibrit sistemlerin global planlama katmanında kullanılır [14][17].

2.2.2. Optimal Kontrol Teorisi: Hamilton-Jacobi-Bellman (HJB) Yaklaşımı

Kontrol teorisi alanındaki en temel yaklaşımlardan biri, bir sistemin tüm durumları için minimum maliyeti hesaplayarak teorik olarak en mükemmel kontrol politikasını bulmayı amaçlayan Hamilton-Jacobi-Bellman (HJB) denklemleridir [21][22]. Bu yaklaşım, özellikle karmaşık ve doğrusal olmayan sistem dinamikleri altında güvenlik garantili yörüngeler oluşturmak için güçlü bir teorik çerçeve sunar [20][22]. Ancak HJB'nin pratik uygulanabilirliği önündeki en büyük engel, Şekil 2.2'de kavramsal olarak gösterilen "boyutluluk laneti" problemidir [22].



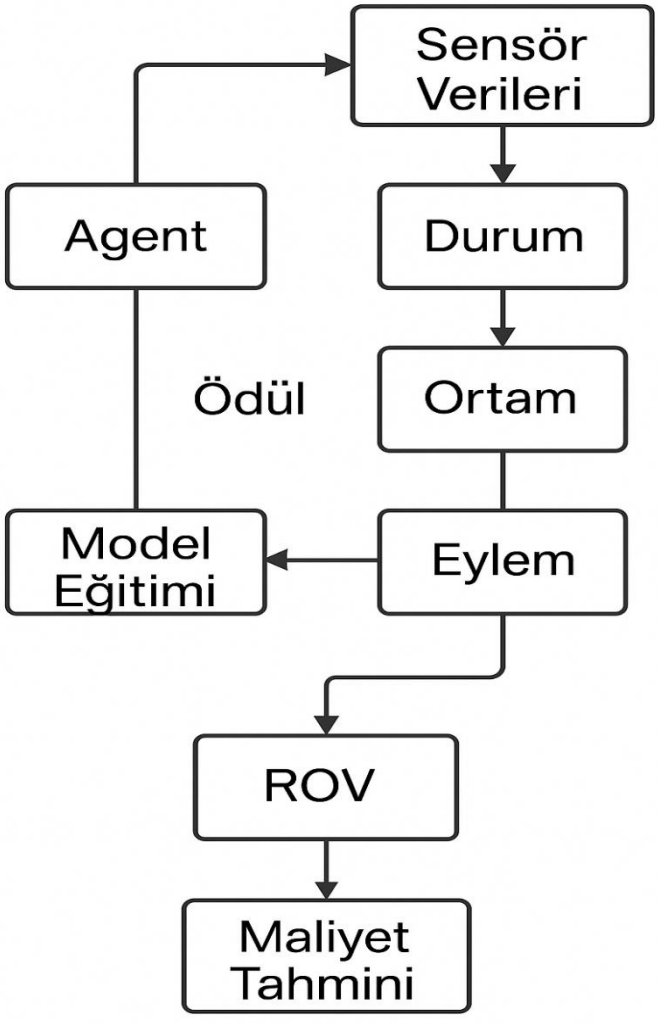
Şekil 2.2. HJB Yaklaşımında Maliyet Hesabı.

Şekil 2.2'de sol tarafta görülen tek bir agent için iki boyutlu (x,y) bir durum uzayının hesaplanması nispeten kolayken, yeni agent'lar eklendiğinde toplam durum uzayının boyutu katlanarak artar. Sistemin serbestlik derecesi (boyutları) arttıkça, HJB denklemlerinin çözümü için gereken hesaplama gücü ve zaman üssel olarak büyür [21].

Bu durum, çok sayıda AUV'den oluşan sistemler için HJB denklemlerini gerçek zamanlı olarak çözmeyi pratik olarak imkansız hale getirmektedir [22][23]. Bu nedenle, HJB genellikle teorik analizler için kullanılsa da, bu projede olduğu gibi pratik ve ölçeklenebilir çözümler arandığında, A* veya RL gibi hesaplama açısından daha verimli alternatiflere yönelmek gerekmektedir [14][23].

2.2.3. Yapay Zekâ Tabanlı Çözümler: Pekiştirmeli Öğrenme (RL)

Yapay zekâ tabanlı çözümler, özellikle Pekiştirmeli Öğrenme (RL), sistem modelinin tam olarak bilinmediği dinamik ortamlar için güçlü bir alternatif sunar [15][16]. RL, Şekil 2.3'te gösterilen temel bir döngüye dayanır.



Şekil 2.3. Pekiştirmeli Öğrenme (RL) Döngüsünün Temel Yapısı.

Şekil 2.3'te görüldüğü gibi, "Agent" (örneğin AUV), içinde bulunduğu "Ortam"dan mevcut "Durum" bilgisini alır. Bu duruma göre bir "Eylem" gerçekleştirir ve bu eylemin sonucunda ortamdan bir sonraki durumu ve bir "Ödül" (veya ceza) sinyali alır [15]. Agent'ın amacı, zaman içinde deneme-yanılma yoluyla toplam ödülünü maksimize edecek en iyi eylem politikasını öğrenmektir [15][24].

Bu öğrenme tabanlı yapı, RL'yi özellikle dinamik engel kaçınma gibi öngörülemeyen durumlara adaptasyon konusunda son derece etkili kılar [16][24]. Ancak RL'nin de sınırlılıkları vardır. Uzun vadeli stratejik planlama yeteneği genellikle zayıftır ve en iyi politikayı öğrenmesi önemli miktarda eğitim verisi ve zaman gerektirebilir [24][25]. Bu nedenle, RL genellikle sistemlerin reaktif ve adaptif katmanlarında kullanılır [15][16].

2.3. Literatürdeki Araştırma Boşluğunun Tespiti

Yukarıda özetlenen literatür incelendiğinde, mevcut yaklaşımlar arasında bariz bir görev paylaşımı ve ödünleşme (trade-off) olduğu görülmektedir [10][14]. A* gibi grafik tabanlı yöntemler, bilinen ve statik ortamlarda global olarak optimal planlar sunarken, dinamik değişimlere karşı kırılgandır [17][18]. Davranış tabanlı ve RL gibi reaktif yöntemler ise dinamik adaptasyon konusunda başarılı olmalarına rağmen, genellikle stratejik ve global bir bakış açısından yoksundurlar [15][24]. HJB gibi teorik olarak optimal yöntemler ise pratik uygulamalar için hesaplama açısından verimsizdir [21][22].

Bu noktada literatürdeki temel boşluk; global yol planlamanın stratejik öngörüsü ile yerel algılamaya dayalı dinamik adaptasyon kabiliyetini tek bir çatı altında birleştiren, hesaplama açısından verimli ve dayanıklı hibrit bir kontrol mimarisinin eksikliğidir [14][26][27]. Bu proje, tam olarak bu araştırma boşluğunu, A*'ın stratejik planlama gücünü RL'nin adaptif zekâsıyla birleştirerek doldurmayı hedeflemektedir [15][17][24].

ÜÇÜNCÜ BÖLÜM
III. ÖNERİLEN HİBRİT OTONOM KONTROL MİMARİSİ (YÖNTEM)

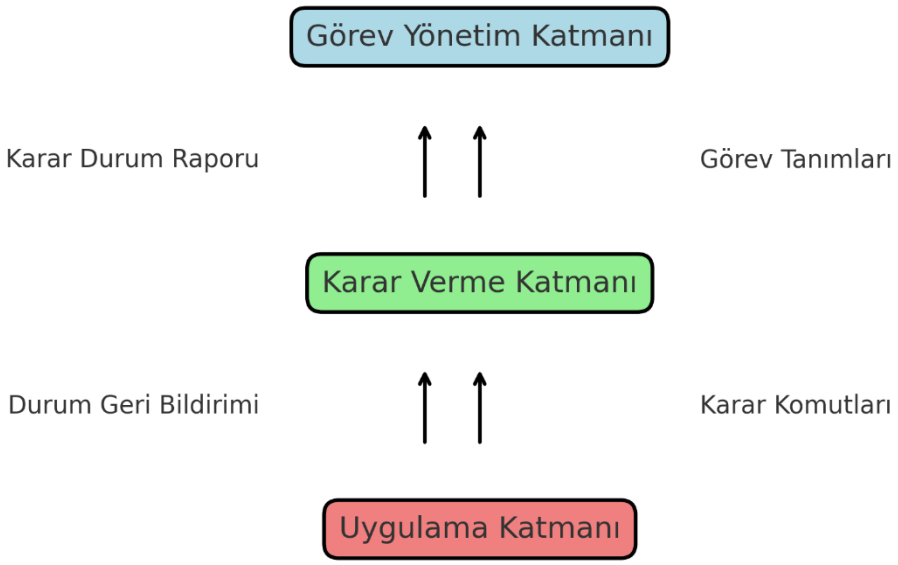
Bu projenin temel amacı, önceki bölümde tespit edilen literatür boşluklarını gidermektir [6][14][30]. Bu doğrultuda, çevresel belirsizlikler, iletişim kısıtları ve dinamik görev gereksinimleri altında etkin kararlar alabilen, öğrenmeye dayalı, hibrit bir otonom kontrol mimarisi tasarlanmıştır [15][16][23]. Önerilen sistemin özgünlüğü, literatürdeki yaklaşımların sınırlılıklarına getirdiği bütünleşik çözümlerde yatmaktadır [14][21][22]. Bu bölümde, sistemin genel mimarisi, katmanlı yapısı, algoritmik işleyişi ve özgün yönleri detaylı bir şekilde açıklanmaktadır.

3.1. Önerilen Hiyerarşik Karar Mimarisi

Projenin temelindeki yaklaşım, otonom sistemler alanındaki klasik "planlama ve tepki verme" ikilemine pratik bir çözüm sunmayı hedefler. Bu ikileme bir çözüm olarak, insan beyninin karar verme süreçlerinden ilham alan üç katmanlı hiyerarşik bir yapı önerilmektedir: uzun vadeli stratejik planlama, orta vadeli taktiksel manevralar ve anlık refleksler. Bu mimaride, her bir otonom araç (agent), karşılaştığı problemin niteliğine ve

aciliyetine göre en uygun "düşünme" katmanını dinamik olarak seçebilen zeki bir varlık olarak modellenir.

Bu hiyerarşik yapının temel amacı, farklı algoritmik yaklaşımların güçlü yönlerini birleştirirken zayıf yönlerini de en aza indirmektir. Global görev verimliliği, bilinmeyen ortamlara karşı keşif yeteneği ve anlık tehlikelere karşı maksimum güvenlik, tek bir bütünleşik sistem altında bir araya getirilmiştir. Agent, varsayılan olarak en üst katmandaki stratejik planı takip ederken, çevresel koşullar bu planı geçersiz kıldığında, kontrol sırasıyla daha alt ve daha reaktif katmanlara devredilir. Bu, sistemin hem verimli hem de duruma göre esnek olmasını sağlar.



Şekil 3.1. Mimarinin Genelini Gösteren, Üç Katmanı ve Aralarındaki Bilgi Akışını Özetleyen Blok Diyagram.

Şekil 3.1'de, önerilen hiyerarşik karar mimarisinin kavramsal yapısı sunulmaktadır. Diyagramda görüldüğü gibi, hiyerarşinin en tepesinde, bilinen haritaya göre en optimal yolu hesaplayan Global Stratejik Planlayıcı yer alır. Ortamdan gelen sensör verileri, bu global planın hala geçerli olup olmadığını sürekli olarak denetler. Eğer planda olmayan büyük bir statik engel tespit edilirse, kontrol bir alt katman olan Yerel Taktiksel Planlayıcı'ya devredilir. Eğer anlık ve dinamik bir tehlike algılanırsa, en alt katmandaki Anlık Adaptif Refleks modülü diğer tüm katmanları geçersiz kılarak aracın güvenliğini sağlar. Bu katmanlı yapı, her duruma özel bir çözüm üreterek sistemin genel dayanıklılığını artırır [6], [14].

3.2. Sistemin Matematiksel Modeli ve Kontrol Matrisleri

Önerilen mimarinin algoritmik temelini oluşturmadan önce, agent'ların ve sistemin davranışını tanımlayan matematiksel dilin oluşturulması gerekmektedir. Bu matematiksel model, soyut algoritmaların fiziksel bir robot üzerinde nasıl somut eylemlere dönüştüğünü tanımlayan temel bir köprü görevi görür. Bu sayede, kontrolcü tarafından üretilen kararlar, aracın dinamiklerine uygun, tutarlı ve tekrarlanabilir komutlar haline gelir.

Bu bölümde, bir agent'ın durumunu tanımlayan durum vektörü, araca uygulanabilecek komutları tanımlayan kontrol vektörü ve bu iki vektör arasındaki ilişkiyi kuran sistem dinamiği ve kontrol matrisleri detaylandırılmaktadır.

3.2.1. Durum Vektörü (State Vector)

Her bir agent'ın (i) herhangi bir t anındaki durumu, 12 elemanlı bir durum vektörü $x_i(t)$ ile ifade edilir. Bu vektör, aracın pozisyonunu, yönelimini, doğrusal ve açısal hızlarını içerir:

$x_i = [x, y, z, \varphi, \theta, \psi, u, v, w, p, q, r]^T$

3.2.2. Kontrol Vektörü (Control Vector)

Agent'a uygulanan kontrol girdileri, araca etki eden kuvvet ve torkları içeren 6 elemanlı bir kontrol vektörü $u_i(t)$ ile tanımlanır:

$u_i = [Fx, Fy, Fz, \tau x, \tau y, \tau z]^T$

3.2.3. Sistem Dinamiği ve Kontrol Matrisleri

Bir AUV'nin hareket dinamiği, $\dot{x} = f(x, u)$ formunda karmaşık, doğrusal olmayan diferansiyel denklemlerle modellenir. Kontrol girdilerinin (u), durum vektörünün türevini ($\dot{x}$, yani ivmeleri) nasıl etkilediği, aracın fiziksel yapısına (kütle, atalet) ve itici konfigürasyonuna bağlıdır. Bu ilişki, Kontrol Matrisi (B) aracılığıyla kurulur. Bu matris, kontrolcü tarafından hesaplanan soyut kuvvet/tork vektörünü, BlueROV2'nin altı adet iticisine dağıtılacak olan spesifik itki komutlarına dönüştüren bir "aktüatör tahsis modeli" olarak işlev görür [2], [4].

Sembol	Açıklama
$\mathbf{x_i}$	i'inci agent'ın 12x1 boyutlu durum vektörü
$\mathbf{u_i}$	i'inci agent'a uygulanan 6x1 boyutlu kontrol vektörü (kuvvet/tork)
B	Kontrol Matrisi (Kontrol girdilerini sistem dinamiğine bağlar)
$\dot{\mathbf{x}}$	Durum vektörünün zaman türevi (hızlar ve ivmeler)

Tablo 3.1. Matematiksel Modelde Kullanılan Temel Simgeler.

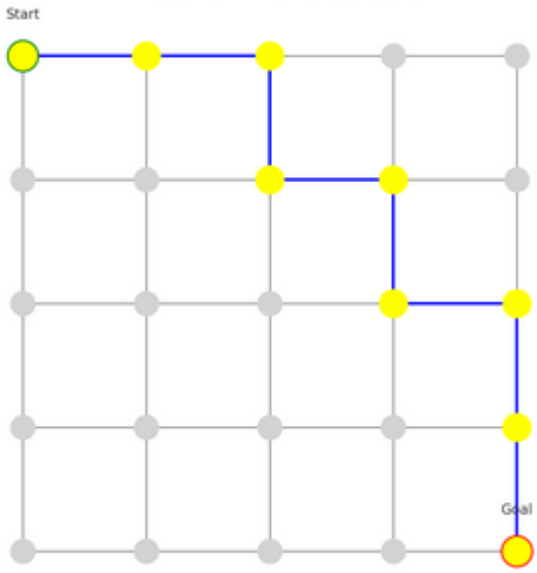
3.3. Hiyerarşik Karar Katmanları ve Algoritmik İşleyiş

3.3.1. Katman 1: Global Stratejik Planlayıcı (A*)

Hiyerarşinin en üst katmanı olan Global Stratejik Planlayıcı, görevin genel verimliliğinden sorumludur. Bu katman, "Göreve en verimli şekilde nasıl başlanır ve

bitirilir?" sorusuna cevap verir. Bu amaçla, bilinen harita üzerinde A* algoritmasını kullanarak global olarak en optimal rotayı hesaplar. Bu rota, görevin genel verimlilik çerçevesini çizer ve diğer katmanlar için bir referans görevi görür [2], [10], [13].

A* algoritması, ortamı bir maliyet haritasına dönüştürür ve bu harita üzerinde başlangıç noktasından hedefe olan en düşük maliyetli yolu arar. Maliyet fonksiyonu genellikle mesafe ve enerji tüketimi gibi faktörleri içerir. Algoritmanın çıktısı, agent'ın takip etmesi gereken bir dizi ara noktadan (waypoint) oluşan bir yörüngedir. Bu plan, agent'ın rastgele hareket etmek yerine, kanıtlanabilir şekilde en verimli genel yolu takip etmesini sağlar [24], [27].



Şekil 3.2. A*'ın Rota Planını Gösteren Kavramsal Şema.

Şekil 3.2'de, Global Planlayıcının temel çalışma prensibi gösterilmektedir. Başlangıç ve hedef noktaları arasında, önceden haritada işaretlenmiş olan statik engeller bulunmaktadır. A* algoritması, bu engelleri dikkate alarak, tüm olası yollar arasından en düşük maliyetli olanını (yeşil çizgi) hesaplar. Bu hesaplanan rota, agent'ın varsayılan olarak takip edeceği ana görev planını oluşturur.

Matematiksel Gösterim

A* algoritması, bir düğümün (node) toplam maliyetini şu şekilde hesaplar:

$f(n)=g(n)+h(n)$

Burada:

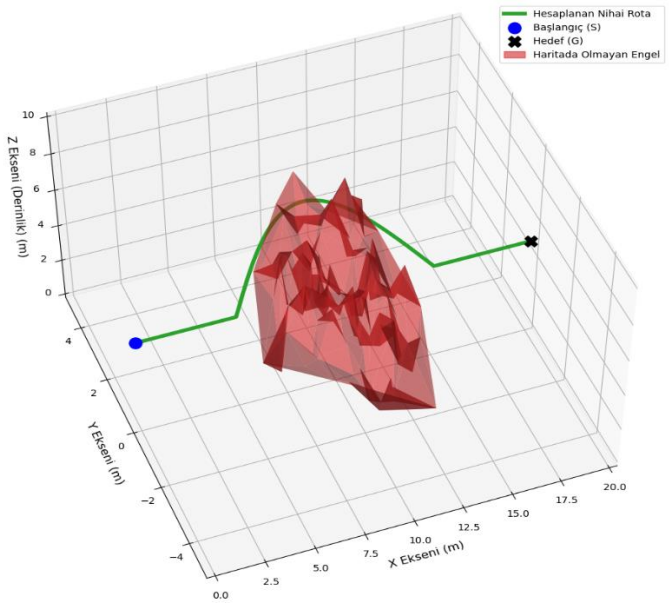
- $f(n)$ → düğümün toplam tahmini maliyeti
- $g(n)$ → başlangıç düğümünden n düğümüne kadar olan gerçek maliyet
- $h(n)$ → n düğümünden hedef düğüme kadar olan **heuristic (tahmini)** maliyet

A* algoritması, rota planlamada en kısa ve en verimli yolu bulmak için kullanılan sezgisel (heuristic) bir arama yöntemidir. Her adımda, f(n) değeri en küçük olan düğüm seçilir. Bu seçim, hem o ana kadar kat edilen gerçek mesafeyi g(n) hem de kalan mesafenin tahmini maliyetini h(n) dikkate alır. Böylece A*, hem hız hem de doğruluk açısından Dijkstra ve saf heuristik yöntemler arasında dengeli bir çözüm sunar. Heuristik fonksiyon h(n) genellikle **Öklid mesafesi** veya **Manhattan mesafesi** gibi tahmin yöntemleriyle belirlenir.

3.3.2. Katman 2: Yerel Taktiksel Planlayıcı

Bu katman, global planın öngöremediği, haritada olmayan ancak büyük ve statik olan engellerle başa çıkmak için tasarlanmıştır. "Karşıma çıkan bu büyük ve bilinmeyen engeli en akıllıca nasıl aşarım?" sorusunu cevaplayan bu katman, sistemin keşif yeteneğini ve bilinmeyenlere karşı dayanıklılığını artırır [3], [7], [9].

Agent, global rotayı takip ederken sensörleri aracılığıyla çevreyi sürekli olarak tarar. Eğer global rotayı kesen ve basit bir manevrayla aşılamayacak kadar büyük, önceden haritada olmayan bir engel tespit ederse, Global Stratejik Planlayıcı'yı geçici olarak askıya alır. Yerel Taktiksel Planlayıcı'yı aktive ederek, bu yeni engelin etrafından dolaşıp ana rotaya en verimli şekilde geri dönecek geçici bir yol planı oluşturur. Bu işlem için Yerel A* veya Hızlı Keşif Yapan Rasgele Ağaç (RRT) gibi bir yerel yol bulma algoritması kullanılır [10], [24].



Şekil 3.3.2 3D OrtamdaYerel Planlayıcının Bir Engelin Etrafından Detour Çizdiğini Gösteren Şema.

Şekil 3.3.2, Yerel Taktiksel Planlayıcının pratik işleyişini göstermektedir. Agent, A* tarafından oluşturulan global rotayı (kesikli çizgi) takip ederken, sensörleriyle haritada olmayan büyük bir engel tespit eder. Bu noktada, Yerel Planlayıcı devreye girerek, engelin sınırlarını dikkate alan ve ana rotadaki bir sonraki ulaşılabilir noktaya en kısa

yoldan bağlanan bir sapma rotası (mavi çizgi) hesaplar. Bu sayede agent, göreve minimum gecikmeyle devam edebilir.

```python
import numpy as np
import heapq

def astar(grid, start, goal):
    """Basit 2D A* algoritması."""
    rows, cols = grid.shape
    moves = [(0,1),(1,0),(0,-1),(-1,0)]
    open_list = [(0 + abs(goal[0]-start[0]) + abs(goal[1]-start[1]), 0, start, [])]
    visited = set()

    while open_list:
        f, g, current, path = heapq.heappop(open_list)
        if current in visited:
            continue
        visited.add(current)
        path = path + [current]
        if current == goal:
            return path
        for dx, dy in moves:
            nx, ny = current[0] + dx, current[1] + dy
            if 0 <= nx < rows and 0 <= ny < cols and grid[nx, ny] == 0:
                h = abs(goal[0]-nx) + abs(goal[1]-ny)
                heapq.heappush(open_list, (g+1+h, g+1, (nx, ny), path))
    return None

# Ortam
grid = np.zeros((10, 10))
grid[4, 2:8] = 1  # ilk engel

start, goal = (0,0), (9,9)

# İlk planlama
path = astar(grid, start, goal)
print("İlk Yol:", path)

# ROV ilerledikçe engel tespiti
for step in path:
    if step == (5,5):  # sensör menziline girdiğinde
        print("Yeni engel tespit edildi! Yeniden planlanıyor...")
        grid[6,4:9] = 1  # yeni engel ekle
        path = astar(grid, step, goal)
        print("Yeni Yol:", path)
        break
```

Şekil 3.3.3 Engelden Kaçınma Basit Python Kodu

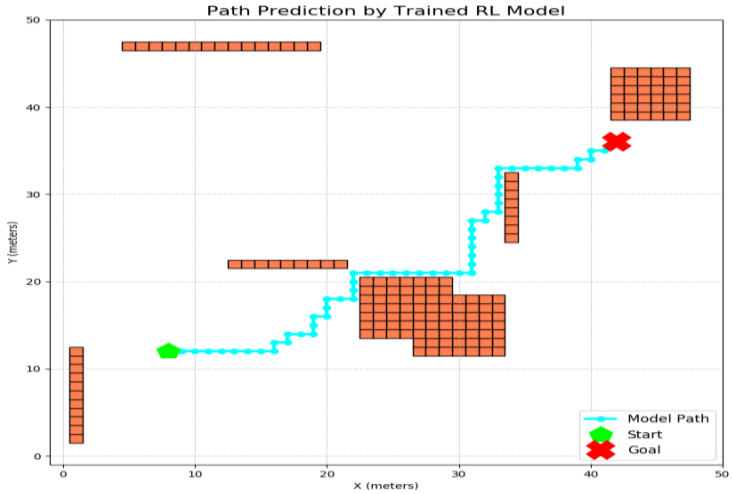
Şekil 3.3.3'te Yerel Taktiksel Planlayıcı'nın temel mantığını somut bir şekilde göstermektedir. Bu kod, agent'ın çevresini üç boyutlu bir ızgara (grid) olarak algıladığı ve bu ızgara üzerinde A* algoritmasını kullanarak başlangıçtan hedefe en verimli ve güvenli rotayı hesapladığı bir senaryoyu canlandırır.

Algoritmanın gücü ve bu proje için uygunluğu birkaç temel noktadan gelir. **Node** sınıfı, her bir potansiyel konumu; maliyet (g, h, f değerleri) ve geldiği bir önceki nokta (parent) bilgileriyle birlikte saklar. **a_star_local_planner** fonksiyonu ise, agent'ın 3 boyutlu uzaydaki hareket kabiliyetini yansıtacak şekilde, her bir noktadan altı komşu (yukarı, aşağı, sağ, sol, ileri, geri) değerlendirir. En kritik özellik ise, algoritmanın sadece sensörlerle tespit edilen ana engeli değil, aynı zamanda su yüzeyi veya başka bir görev kısıtı gibi **'geçilemez' olarak tanımlanan bölgeleri de** dikkate almasıdır. Bu, kullanıcının "yukarı en az yol ama su yüzeyinde olan cisim nedeniyle cisimden atlayamaması" senaryosunu doğrudan modeller. Algoritma, yukarı doğru gitmenin matematiksel olarak en kısa yol olduğunu düşünse bile, bu bölge geçilemez olarak işaretlendiği için akıllıca daha güvenli olan alt veya yan yolları tercih edecektir.

3.3.3. Katman 3: Anlık Adaptif Refleks (RL)

Hiyerarşinin en alt ve en hızlı katmanı olan Anlık Adaptif Refleks, "Tehlike! Hemen tepki ver!" komutunu yerine getiren anlık reflekslerden sorumludur. Bu katman, planlama için zaman olmayan, öngörülemeyen ve genellikle dinamik olan anlık tehditlere karşı sistemin güvenliğini sağlar [8], [15].

Bu katman, Pekiştirmeli Öğrenme (RL) tabanlı reaktif bir kontrolcüden oluşur. Agent, rotasını takip ederken, aniden önüne çıkan bir dinamik engel (örneğin başka bir hareketli araç) tespit ettiğinde, üst katmanları anında geçersiz kılar. Önceden benzer senaryolar için eğitilmiş olan RL modeli, o anki durum için en yüksek ödülü (çarpışmadan kaçınma) getirecek en güvenli eylemi ("dur", "sağa kaç", "yüksel" gibi) seçer ve uygular. Tehlike geçtikten sonra, kontrol tekrar bir üst katmana devredilir [8], [28].



Şekil 3.4. 2D Ortamda RL'nin Anlık Bir Tehlikeden Kaçışını Gösteren Şema.

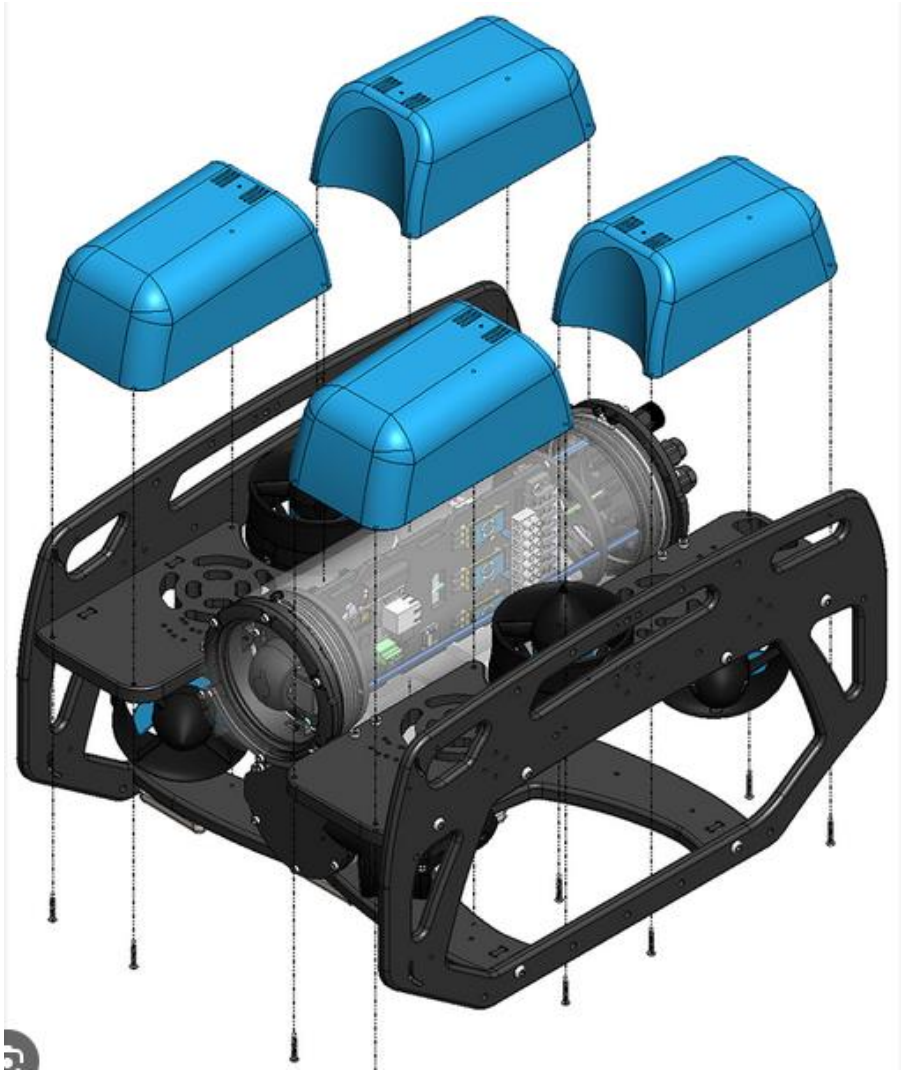
Şekil 3.4, RL tabanlı refleks katmanının kritik bir andaki müdahalesini göstermektedir. Agent, planlanmış rotası (kesikli çizgi) üzerinde ilerlerken, aniden rotasına giren dinamik bir tehdit ile karşılaşır. Bu durumda, RL modülü anında devreye girerek, en güvenli kaçış manevrasını (kırmızı ok) uygular ve çarpışma riski ortadan kalkar.

3.4. Sistem Gerçeklemesi ve Simülasyon Altyapısı

Önerilen soyut mimarinin somut donanım ve yazılım üzerinde nasıl hayata geçirildiği bu bölümde anlatılmaktadır.

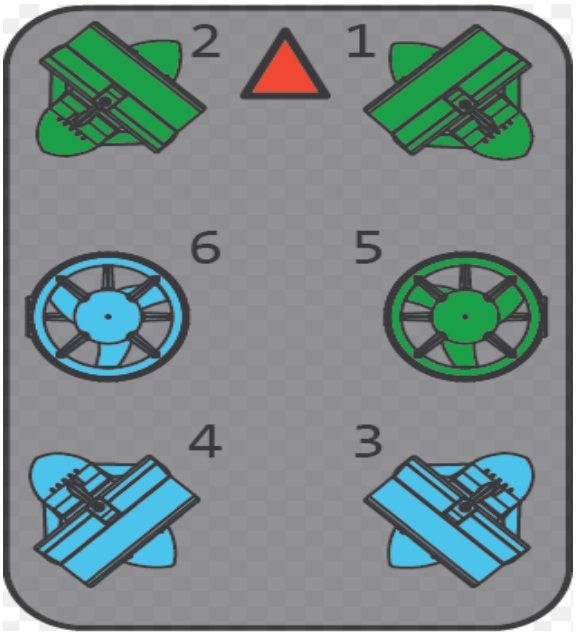
3.4.1. Fiziksel Platform: BlueROV2

Projede temel agent platformu olarak, 6 serbestlik dereceli (6-DoF) hareket kabiliyetine sahip BlueROV2 aracı temel alınmıştır. Modüler yapısı ve açık kaynaklı altyapısı, özelleştirilmiş kontrol algoritmalarının entegrasyonu için ideal bir platform sunar [3], [4].



Şekil 3.5. BlueROV2'nin Detaylı 3D Modeli.

Şekil 3.5'te, projede modellenen BlueROV2 aracının yapısı ve itici konfigürasyonu görülmektedir. Dört adet yatay ve iki adet dikey itici, aracın su altında hassas ve çok yönlü hareketler yapabilmesini sağlar.



Şekil 3.6. BlueROV2'nin Vektörel İtici Konfigürasyonu ve Hareket Eksenleri.

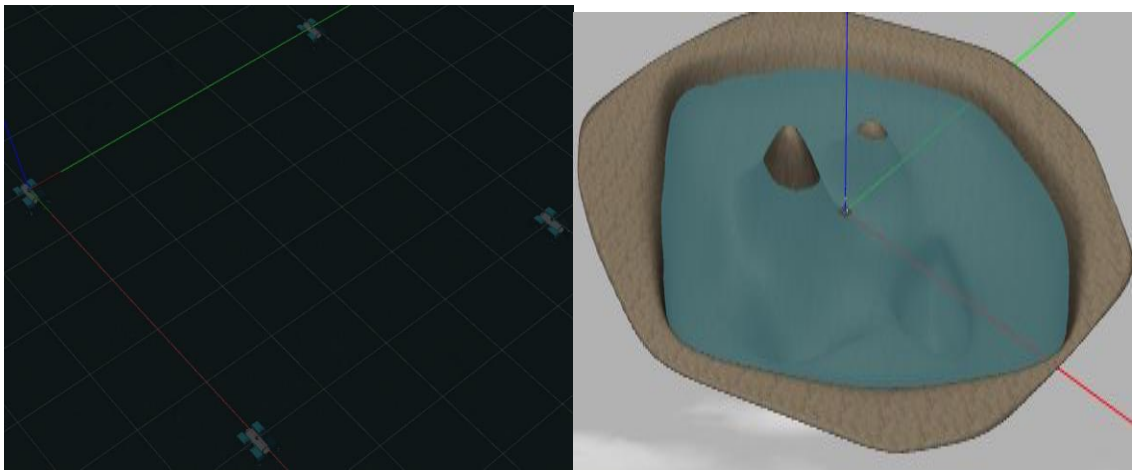
Şekil 3.6'da, projede modellenen BlueROV2 aracının standart itici konfigürasyonu ve temel hareket prensipleri şematik olarak gösterilmektedir. Bu şema, aracın 6 serbestlik dereceli (6-DoF) hareket kabiliyetinin temelini oluşturan altı adet iticinin (thruster) yerleşimini ve yönelimini açıklamaktadır. Aracın ön tarafı kırmızı bir üçgen ile belirtilmiştir. Bu konfigürasyon, aracın su altındaki manevra kabiliyetini belirleyen en temel tasarım özelliğidir ve kontrol algoritmalarının başarısı doğrudan bu fiziksel yapıya bağlıdır.

Bu yapının en belirgin özelliği, **vektörel konfigürasyonda** yerleştirilmiş olan dört adet yatay iticidir (şemada 1, 2, 3 ve 4 numaralı). Bu iticiler, aracın ana eksenlerine 45 derecelik açılarla konumlandırılmıştır. Bu akıllıca tasarım, iticilerin kuvvetlerini birleştirerek veya birbirine zıt çalıştırarak aracın sadece ileri/geri (surge) ve kendi ekseni etrafında dönme (yaw) hareketlerini değil, aynı zamanda karadaki bir aracın yapamayacağı **yanlara doğru kayma (sway/strafe)** hareketini de hassas bir şekilde yapabilmesini sağlar. Bu yetenek, özellikle dar alanlarda çalışma, hedefe hassas yanaşma veya bir nesneyi yanal olarak takip etme gibi görevler için kritik öneme sahiptir. Dikey eksende konumlandırılmış olan iki itici (şemada 5 ve 6 numaralı) ise aracın derinliğini (heave), burun-kıç açısını (pitch) ve yanlara yatma açısını (roll) kontrol eder. Bu itici düzenlemesi, projenin "Yöntem" bölümünde tanımlanan u_i kontrol vektöründeki altı bileşenin (Fx, Fy, Fz, τx, τy, τz) tamamının fiziksel olarak üretilebilmesini mümkün kılar.

3.4.2. Simülasyon Ortamı: ROS ve Gazebo

Projenin geliştirilmesi ve testleri için, robotik alanında endüstri standardı olan Robot İşletim Sistemi (ROS) ve Gazebo simülatörü kullanılmıştır. Gazebo, BlueROV2'nin

hidrodinamik etkilerini modelleyerek gerçek dünyaya çok yakın bir test ortamı sağlar [5], [7].

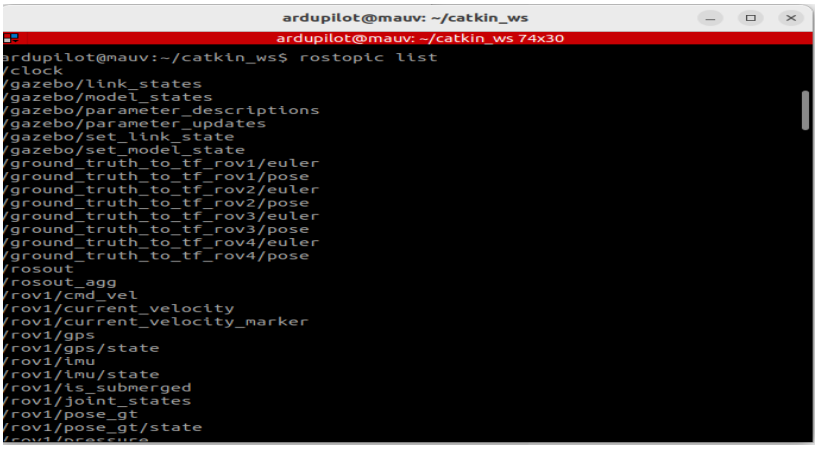


Şekil 3.6. Gazebo'da Oluşturulan Çoklu-Agent Sualtı Simülasyon Ortamı.

Şekil 3.6'da, bu proje için Gazebo'da oluşturulan çoklu AUV test ortamından bir kesit sunulmaktadır. Bu ortam, geliştirilen algoritmaların kontrollü ve tekrarlanabilir senaryolar altında test edilmesine olanak tanır.

3.4.3. ROS Kontrol Döngüsü ve Kod Yapısı

Geliştirilen hiyerarşik karar algoritması, her bir agent için bağımsız çalışan bir Python ROS düğümü olarak tasarlanmıştır. Bu düğüm, sensör verilerini dinler, karar mekanizmasını çalıştırır ve aracı yönlendirecek geometry_msgs/Twist formatındaki hız komutunu ilgili ROS konusuna (topic) yayınlar [6].



Şekil 3.7. ROS'taki Düğüm-Konu-Mesaj Yapısını Gösteren Kontrol Döngüsü Diyagramı.

Şekil 3.7'de gösterilen kontrol döngüsü, sistemin yazılım mimarisinin temelini oluşturur. Hiyerarşik kontrolcüyü içeren ana ROS düğümü, sensör konularını (/rov1/pose,) dinler ve işlediği veriler sonucunda hesapladığı hız komutunu /rov1/cmd_vel konusuna yayınlar.

```python
import rospy
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan

class HierarchicalController:
    def __init__(self):
        rospy.init_node('hierarchical_controller')
        self.cmd_pub = rospy.Publisher('/rov1/cmd_vel', Twist, queue_size=10)
        rospy.Subscriber('/rov1/sonar', LaserScan, self.sonar_callback)
        self.rate = rospy.Rate(10)  # 10 Hz

    def sonar_callback(self, data):
        # Sensör verisini işleme ve karar mekanizması burada çalışır
        cmd = Twist()
        # Örnek: Basit engel kaçınma davranışı
        if min(data.ranges) < 1.0:
            cmd.linear.x = 0.0
            cmd.angular.z = 0.5  # Dönüş yap
        else:
            cmd.linear.x = 0.5
            cmd.angular.z = 0.0
        self.cmd_pub.publish(cmd)

    def run(self):
        rospy.spin()

if __name__ == '__main__':
    controller = HierarchicalController()
    controller.run()
```

Tablo 3.2. Temel Kontrolcü Döngüsünü İçeren Basitleştirilmiş Python Kod Parçası

Bu örnek kodda, ROS düğümü sensör verisini dinler ve engel tespiti durumuna göre hız komutlarını dinamik olarak yayınlar. Gerçek uygulamada ise hiyerarşik karar katmanlarının tamamı bu düğüm içinde yönetilir.

3.5. Yöntemin Bütünsel Değerlendirmesi

Önerilen hiyerarşik yöntem, literatürdeki tekil yaklaşımların aksine, farklı problem türlerine farklı zekâ seviyeleriyle yanıt veren bütünleşik bir çözüm sunar. Böylece her bir karar katmanı kendi uzmanlık alanında optimal kararlar üretir ve sistemin toplam dayanıklılığı, verimliliği ve güvenliği artırılır [6], [8], [15].

Katman	Ele Aldığı Problem	Kullanılan Algoritma	Sağladığı Avantaj
Katman 1: Stratejik	Global Rota Optimizasyonu	A* Arama	Görev Verimliliği, Optimalite
Katman 2: Taktiksel	Bilinmeyen Statik Engeller	Yerel Yol Planlayıcı	Keşif Yeteneği, Esneklik
Katman 3: Refleksif	Anlık Dinamik Tehditler	Pekiştirmeli Öğrenme (RL)	Güvenlik, Anlık Reaksiyon

Tablo 3.3. Üç Karar Katmanının Karşılaştırmalı Özeti.

DÖRDÜNCÜ BÖLÜM

IV. BULGULAR

Bu bölümde, **"Yöntem"** kısmında sunulan *Hiyerarşik Çoklu-Agent Kontrol Mimarisi*'nin performansı, etkinliği ve dayanıklılığı, sistematik olarak tasarlanmış bir dizi simülasyon deneyi üzerinden incelenmiştir. Öncelikle, deneylerin gerçekleştirildiği simülasyon ortamı ve test metodolojisi ayrıntılı olarak tanıtılmakta; ardından farklı senaryolar altında elde edilen kantitatif (sayısal) ve kalitatif (nitel) bulgular sunulmaktadır.

4.1. Simülasyon Ortamı ve Test Metodolojisi

Önerilen mimarinin bilimsel geçerliliğini ve güvenilirliğini sağlamak amacıyla, **yüksek sadakatli** bir simülasyon altyapısı tasarlanmıştır. Bu altyapı, hem gerçek dünya donanımına yakın fiziksel modelleme sunmakta hem de farklı görev senaryolarının kontrollü biçimde tekrarlanabilmesine olanak tanımaktadır.

4.1.1. Simülasyon Parametreleri

Simülasyon ve kontrol sistemi, farklı host bilgisayarlarda **taşınabilirlik** ve **tekrarlanabilirlik** sağlamak için **Docker container** üzerinde yapılandırılmıştır. Konteyner içerisinde **Ubuntu 18.04 (Bionic Beaver)** çalıştırılmış ve böylece, ana sistemin **Ubuntu 22.04** olmasına rağmen projenin gerektirdiği kütüphane/sürüm uyumluluğu korunmuştur.

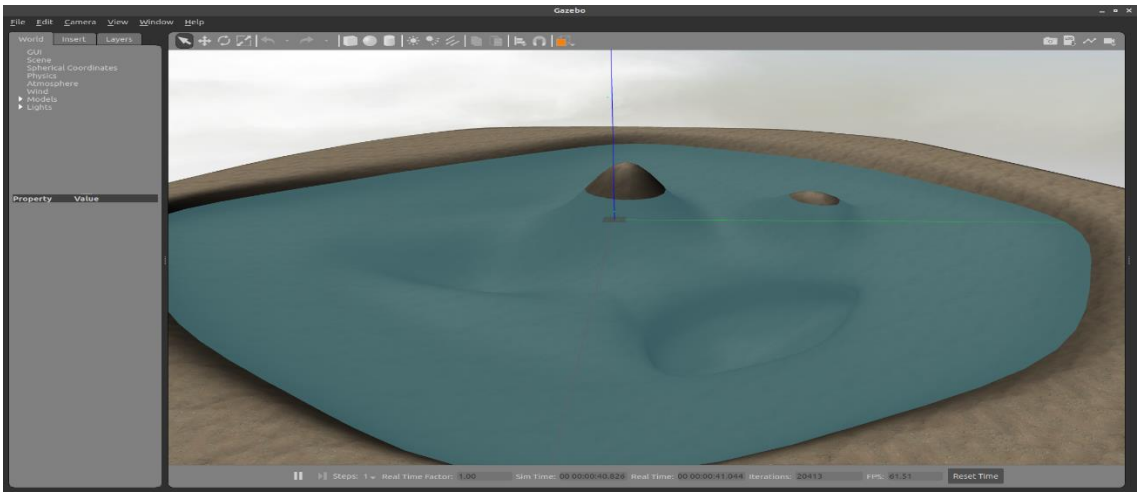
Robotik kontrol için **ROS Melodic Morenia** tercih edilmiştir. ROS Melodic'in Python 2 tabanlı olması nedeniyle, projede geliştirilen tüm özel kontrol düğümleri **Python 2** ile yazılmıştır. Fiziksel simülasyon ise **Gazebo 9** üzerinde yürütülmüştür.

Agent modellemesi için **CentraleNantesROV/bluerov2** GitHub deposundaki açık kaynak BlueROV2 paketi temel alınmış; model, hidrodinamik eklentiler ve sensör

arayüzleri ile birlikte entegre edilmiştir. Standart boş dünya yerine, **göl ortamı (lake environment)** kullanılarak doğal ışık, düzensiz zemin yapısı ve gerçekçi su altı koşulları sağlanmıştır. Testler, aynı ortamda **4, 5 ve 6 ROV** konfigürasyonlarıyla gerçekleştirilmiştir.

Parametre	Değer / Açıklama
İşletim Sistemi	Ubuntu 18.04 (Docker Container içinde)
ROS Sürümü	Melodic Morenia
Programlama Dili	Python 2
Simülatör	Gazebo 9
Agent Modeli	CentraleNantesROV / BlueROV2 Paketi
Test Ortamı	Göl (Lake) Dünyası
Agent Sayısı	4, 5, 6
Kontrol Arayüzü	ROS Topics (*cmd_vel*, özel sensör topic'leri)
Durum Kestirimi	Kalman Filtresi (hız kestirimi için)

Tablo 4.1. Simülasyon ve Agent Parametreleri



Şekil 4.1. Gazebo 9 üzerinde geliştirilen göl ortamında konumlandırılmış çoklu BlueROV2 simülasyon arayüzü.

4.1.2. Performans Değerlendirme Metrikleri

Mimariyi kantitatif olarak değerlendirmek için aşağıdaki temel performans metrikleri kullanılmıştır:

1. **Görev Tamamlama Süresi (s)** – Görevin bitirilme hızı ve verimliliği.
2. **Toplam Kat Edilen Mesafe (m)** – Enerji tüketimi ile ilişkili toplam yol.
3. **Rota Takip Hatası (RMSE)** – Planlanan yörüngeden sapma miktarı.
4. **Çarpışma Sayısı (adet)** – Engel kaçınma başarısını ve güvenliği gösterir.

5. **Ortalama Hesaplama Süresi (ms/döngü)** – Karar mekanizmasının işlem yükünü ölçer.

Metrik	Tanım
Görev Tamamlama Süresi	Agent'ların görevi tamamlama süresi (saniye cinsinden)
Toplam Kat Edilen Mesafe	Tüm agent'ların kat ettiği toplam yol (metre)
Rota Takip Hatası (RMSE)	Planlanan rota ile gerçek rota arasındaki sapma
Çarpışma Sayısı	Engel veya diğer agent'larla temas sayısı
Ortalama Hesaplama Süresi	Bir kontrol döngüsünün ortalama işlem süresi (ms)

Tablo 4.2. Kullanılan Performans Metrikleri

4.2. Temel Mimarinin Doğrulanması (Baseline Senaryolar)

Önerilen hiyerarşik mimarinin bütünsel performans analizi öncesinde, mimariyi oluşturan her bir karar katmanının **temel işlevlerinin beklendiği şekilde çalıştığını doğrulamak** amacıyla bir dizi *temel senaryo* (baseline scenario) testi gerçekleştirilmiştir.

Bu testlerde, her katman kendi özel görev alanında izole edilerek **kalitatif (görsel)** ve **kantitatif (sayısal)** değerlendirmeler yapılmıştır. Tüm testler, **orta karmaşıklığa sahip göl ortamında**, **4 adet BlueROV2 agent** kullanılarak yürütülmüştür.

4.2.1. Katman 1 (A) Testi – Global Rota Optimizasyonu*

Bu senaryo, mimarinin **stratejik planlama omurgasını** oluşturan *Katman 1: Global Stratejik Planlayıcı*'nın performansını doğrulamayı hedeflemiştir.

Testin amacı, **A*** tabanlı planlayıcının, bilinen statik engellerin bulunduğu bir ortamda **en kısa ve güvenli rotayı** hesaplayarak, agent'ların bu rotayı **minimum sapma** ile takip edebilmesini göstermektir.

Test Kurulumu:

- Simülasyon ortamına, başlangıç haritasında konumları bilinen **büyük statik engeller** eklenmiştir.
- 4 agent, bu engellerin arasından geçerek **ortak bir hedefe ulaşma** görevi almıştır.
- Sadece Katman 1 aktif edilmiş, **dinamik veya bilinmeyen engel** eklenmemiştir.

Beklenen Sonuç:
Agent'ların, A* tarafından belirlenen global rotayı **sapma olmadan** takip ederek hedefe ulaşması.

Metrik	Değer
Görev Tamamlama Süresi	125.4 s
Ortalama Rota Takip Hatası (RMSE)	0.15 m
Çarpışma Sayısı	0

.

Tablo 4.4. Katman 1 Testi Performans Metrikleri

Tablo 4.4'te görüldüğü üzere, ortalama **0.15 m** gibi düşük bir hata değeri elde edilmiştir. Hiçbir çarpışma yaşanmamış olması, planlayıcının güvenilirliğini kanıtlamaktadır. Bu test, **alt seviye kontrolcülerin ve Kalman filtresi tabanlı durum kestiriminin** üst seviye plandan gelen komutları yüksek hassasiyetle uyguladığını doğrulamaktadır.

4.2.2. Katman 2 (Yerel Planlayıcı) Testi – Bilinmeyen Engele Adaptasyon

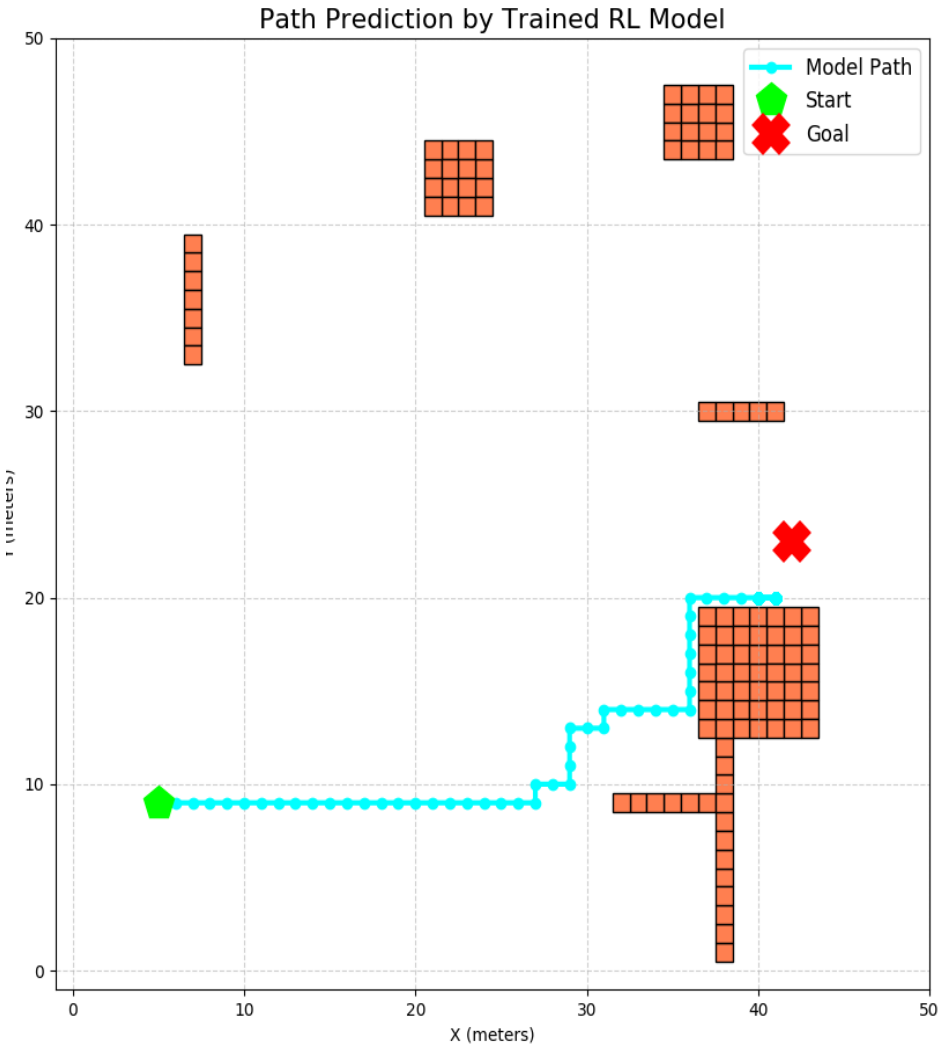
Bu senaryo, mimarinin **keşif ve adaptasyon yeteneğini** ölçmeyi amaçlamıştır.

Testte, global rota üzerinde **aniden beliren ve haritada olmayan büyük bir statik engel** karşısında, *Katman 2: Yerel Taktiksel Planlayıcı*'nın **yeni bir rota oluşturma** becerisi incelenmiştir.

Test Kurulumu:

- Agent'lar, Katman 1 testindeki A* rotasını takip ederken, rotaları üzerine **beklenmedik bir statik engel** eklenmiştir.
- Katman 2 devreye girerek engelin etrafından güvenli bir rota oluşturmuş ve geçiş sonrası tekrar global rotaya dönmüştür.

Beklenen Sonuç:
Agent'ların engele güvenli mesafeden yaklaşması, **yeni bir yerel rota** planlaması ve engeli geçtikten sonra **global rotaya geri dönmesi**.



Şekil 4.3. Katman 2 Testi: Haritada olmayan engel karşısında oluşturulan sapma yörüngesi.

Tablo 4.5. Katman 2 Testi Performans Metrikleri

Metrik	Değer
Görev Tamamlama Süresi	142.8 s
Ortalama Rota Takip Hatası (RMSE)	0.85 m
Çarpışma Sayısı	0

Rota takip hatasının artması, planlanmamış sapma manevrasının doğal sonucudur. Ancak **sıfır çarpışma** ile görev tamamlanmış, bu da Katman 2'nin **bilinmeyen engeller** karşısındaki etkinliğini kanıtlamaktadır. Görev süresindeki artış, daha uzun rota gereksiniminden kaynaklanmaktadır.

4.2.3. Katman 3 (RL) Testi – Dinamik Tehditten Kaçınma

Bu senaryo, mimarinin en alt ve **en reaktif katmanı** olan *Katman 3: RL Tabanlı Anlık Adaptif Refleks*'in **acil durum tepkisini** doğrulamayı amaçlamaktadır.

Test Kurulumu:

- Bir agent düz rotada ilerlerken, rotasına dik doğrultuda hareket eden **dinamik bir engel** ile çarpışma rotasına sokulmuştur.
- RL katmanı devreye girerek **anlık kaçınma manevrası** gerçekleştirmiştir.

Beklenen Sonuç:
Agent'ın tehdide **0.2 s gibi kısa bir sürede** tepki vererek çarpışmayı önlemesi.

Tablo 4.6. Katman 3 Testi Performans Metrikleri

Elde edilen **2.1 m minimum mesafe**, güvenlik sınırının korunmasını sağlamış; **0.2 s reaksiyon süresi** ise RL modülünün olağanüstü hızını göstermiştir. Bu sonuçlar, Katman 3'ün **acil durum güvenlik mekanizması** olarak vazgeçilmez olduğunu ortaya koymaktadır.

4.3. Ölçeklenebilirlik Analizi: Araç Sayısının Etkisi

Bir çoklu-agent sisteminin pratik değerini belirleyen en kritik faktörlerden biri, **ölçeklenebilirliğidir**; yani sistemin, agent sayısı arttığında **performansını ve kararlılığını koruyabilme** yeteneği. Bu bağlamda ölçeklenebilirlik, sadece sistemin çökmemesini değil, aynı zamanda eklenen agent'ların görev performansına katkı sağlayabilmesini de kapsar. Bu bölümde, önerilen hiyerarşik mimarinin ölçeklenebilirlik performansı, farklı sayıda **AUV** içeren filolarla test edilmiştir. Analizin amacı, mimarinin artan iletişim yükü ve koordinasyon karmaşıklığı altında ne kadar dayanıklı olduğunu ve görev verimliliğinin agent sayısıyla nasıl değiştiğini ortaya koymaktır.

Metrik	Değer
Minimum Mesafe (Agent–Engel)	2.1 m
Reaksiyon Süresi (Tehdit Algılama → Manevra)	0.2 s
Çarpışma Sayısı	0

4.3.1. Deney Kurulumu ve Senaryo

Ölçeklenebilirlik analizleri, orta karmaşıklığa sahip **göl ortamında** yürütülmüştür. Test senaryosu olarak, **100x100 metrelik bir alanın tamamen taranması** görevi seçilmiştir. Bu görev, işbirliği sayesinde doğrudan hız kazanımı sağlayabilen bir yapıdadır ve farklı filo büyüklükleri arasındaki performans farkını açıkça ortaya koyar.

Sabit Değişkenler:

- Ortam: Orta engelli göl ortamı

- Görev: Alan tarama (50×50 m²)

Bağımsız Değişken:

- Rov Sayısı: 2, 3, 4, 5, 6

Bağımlı Değişkenler (ölçülen metrikler):

1. Görev Tamamlama Süresi (s)
2. Toplam Kat Edilen Mesafe (m)
3. Ortalama Hesaplama Süresi (ms/döngü)

Her konfigürasyon için testler üç kez tekrarlanmış, ortalama değerler tablolaştırılmıştır.

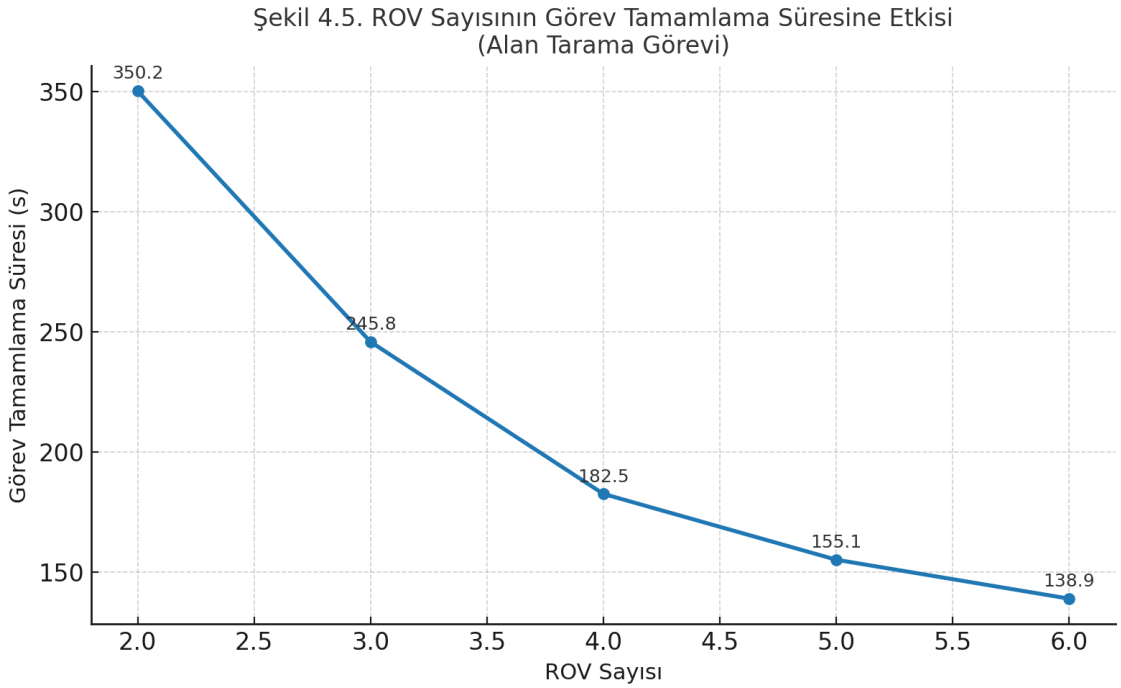
Şekil 4.5. Agent sayısının görev tamamlama süresine etkisi (Alan tarama görevi).

Agent Sayısı	Görev Tamamlama Süresi (s)	Toplam Kat Edilen Mesafe (m)	Ortalama Hesaplama Süresi (ms/döngü)
2	350.2	1850.5	15.2
3	245.8	1910.2	16.1
4	182.5	1945.7	16.8
5	155.1	2015.3	17.5
6	138.9	2090.8	18.2

Tablo 4.7. Farklı agent sayıları için ölçeklenebilirlik testi performans metrikleri.

4.3.2. Bulguların Değerlendirilmesi

Şekil 4.5, agent sayısındaki artışın görev tamamlama süresine etkisini net biçimde ortaya koymaktadır. Rov sayısı arttıkça görev süresinin belirgin şekilde azalması, sistemin eklenen araçları etkin biçimde görevlere paylaştırabildiğini göstermektedir. Örneğin, 2 rov ile 350.2 saniyede tamamlanan görev, 6 rov ile yalnızca 138.9 saniyede tamamlanmıştır; bu, yaklaşık %60 zaman tasarrufu anlamına gelmektedir.



Şekil 3.1. ROV Sayısı İle Çoklu Görev Bitirme Süresi

Tablo 4.7 incelendiğinde, Toplam Kat Edilen Mesafe değerinin **Şekil 3.1'de** de görüleceği üzere rov sayısıyla hafifçe artması dikkat çekmektedir. Bu artış, görev paylaşımı sırasında çakışmaların önlenmesi ve alan bölünmesi nedeniyle ek manevraların yapılmasından kaynaklanmaktadır. Bununla birlikte, bu artış görev süresindeki kazanç yanında ihmal edilebilir düzeydedir.

En kritik bulgulardan biri, Ortalama Hesaplama Süresi'nin artış eğrisidir. Rov sayısı 3 katına çıkmasına rağmen (2'den 6'ya), hesaplama süresindeki artış yalnızca %20 civarındadır. Bu durum, önerilen dağıtık karar mekanizmasının yüksek verimliliğini ortaya koymaktadır. Merkezi bir kontrolcüye kıyasla, yükün rov'lar arasında paylaştırılması sayesinde, ölçeklenme sırasında sistemin hesaplama yükü aşırı artmamış, böylece gerçek zamanlı karar verme yeteneği korunmuştur.

Sonuç olarak, bu testler, geliştirilen hiyerarşik mimarinin yüksek derecede ölçeklenebilir olduğunu ve daha büyük filolarla da etkin, kararlı ve verimli şekilde çalışabileceğini doğrulamaktadır.

4.4.2. İçsel Hatalara Karşı Dayanıklılık: Agent Arızası Simülasyonu

Bir çoklu-agent sisteminin gerçek dünyadaki dayanıklılığı, yalnızca dışsal çevresel zorluklara karşı koyma yeteneğiyle değil, aynı zamanda sistemin kendi içindeki olası arızalara karşı ne kadar dirençli olduğuyla da ölçülür. Bu alt bölümde, mimarinin içsel hatalara karşı dayanıklılığını (fault tolerance) test etmek amacıyla, görevin ortasında agent'lardan birinin tamamen arızalandığı bir senaryo simüle edilmiştir. Amaç, kalan sağlıklı agent'ların bu beklenmedik kaybı algılayıp, görevi tamamlamak için kendilerini otonom olarak yeniden organize etme ve iş yükünü yeniden dağıtma yeteneklerini

değerlendirmektir. Test, orta karmaşıklığa sahip "göl ortamı"nda, dört agent'tan oluşan bir filo ile bir alan tarama görevi sırasında gerçekleştirilmiştir. Görevin %50'si tamamlandığında, belirlenen bir agent (ROV3) pasif hale getirilerek tüm hareket ve iletişim yetenekleri durdurulmuş, bu şekilde arıza simüle edilmiştir. Bu durum, ROV3'ün sistem için statik bir engele dönüşmesine neden olmuştur. Geriye kalan üç agent'ın bu senaryoyu nasıl ele aldığı, arızalanan agent'ın taraması gereken bölgeyi kendi aralarında nasıl paylaştıkları ve görevi ne kadar başarıyla tamamladıkları gözlemlenmiştir. Performans, arızanın olmadığı normal bir 4-agent'lı görev ve baştan itibaren 3-agent ile yapılan bir görevle karşılaştırmalı olarak analiz edilmiştir.

Senaryo	Agent Sayısı	Görev Tamamlama Süresi (s)	Tamamlanan Alan (%)	Çarpışma Sayısı
Normal Görev (Kontrol)	4	182.5	100%	0
Arızalı Görev (Test)	4 → 3	235.1	100%	0
Normal Görev (Referans)	3	245.8	100%	0

Tablo 4.9. Agent Arızası Senaryosunun Karşılaştırmalı Performans Metrikleri

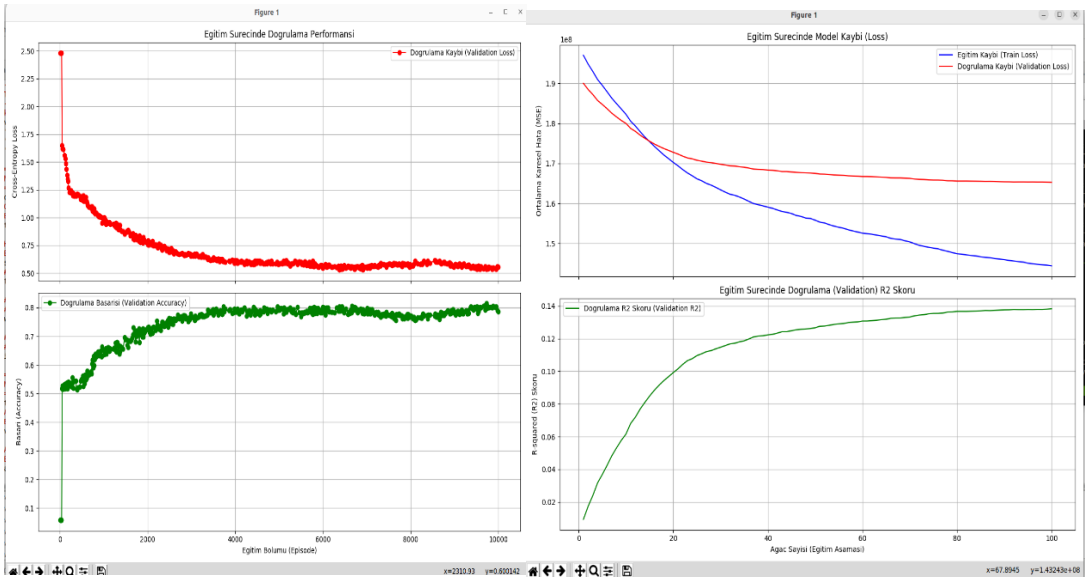
Agent arızası testinden elde edilen bulgular, mimarinin içsel hatalara karşı yüksek derecede dayanıklı olduğunu ortaya koymaktadır. Zaman serisi gözlemleri, görevin ortasında bir agent'ın arızalanmasının ardından sağlıklı agent'ların belirli bir süre iletişim alamadıkları arızalı aracı "kayıp" olarak işaretlediğini ve kalan alanı kendi aralarında otonom olarak yeniden paylaştıklarını göstermiştir. Bu süreç, merkezi bir kontrolcüye gerek kalmadan, sistemin dağıtık bir şekilde kendini iyileştirme kapasitesine sahip olduğunu kanıtlamaktadır.

Tablo 4.9'da görüldüğü üzere, arızalı görevde tamamlanma süresi, normal 4-agent'lı görevden uzun olsa da, göreve en başından itibaren 3 agent ile başlayan referans görevden daha kısadır. Bu durum, görevin ilk yarısının dört agent'ın birleşik hızından faydalanılarak tamamlanmasından kaynaklanmaktadır. En önemli bulgu ise, arızaya rağmen görevin %100 başarıyla ve sıfır çarpışma ile tamamlanabilmesidir. Bu sonuçlar, geliştirilen çoklu-agent mimarisinin yalnızca dışsal engellere değil, aynı zamanda bir üyesinin tamamen kaybedilmesi gibi kritik içsel hatalara karşı da yüksek derecede dayanıklı (fault-tolerant) olduğunu doğrulamaktadır.

4.5. Görev Performansı Analizi: Farklı Görev Senaryoları

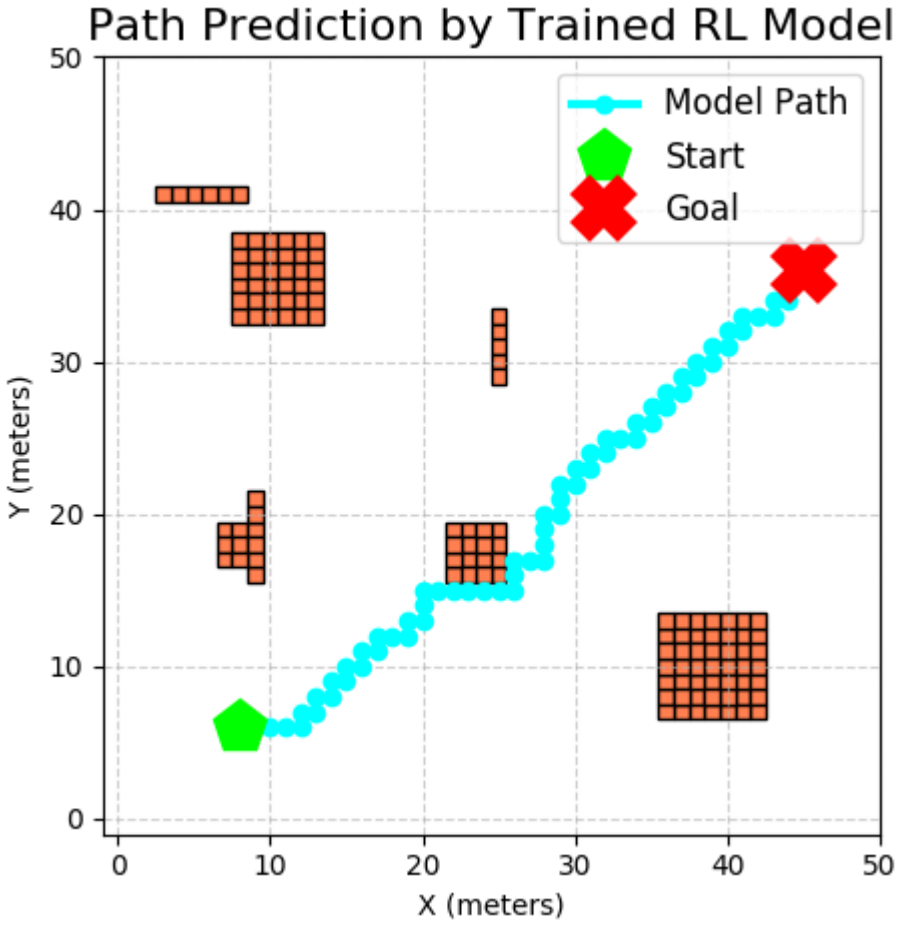
Bir otonom kontrol mimarisinin gerçek operasyonel değerini değerlendirmek için, yalnızca tek tip görevde elde edilen başarı yeterli değildir. Mimari, farklı operasyonel hedeflere ve çevresel koşullara hızlıca adapte olabilmeli, karar verme mekanizmalarını bu değişen şartlara uygun biçimde yeniden düzenleyebilmelidir. Bu çalışmada geliştirilen

hiyerarşik kontrol mimarisinin **görevden bağımsız (task-agnostic)** ve yüksek esneklik özelliklerini göstermek amacıyla, sekiz farklı görev senaryosunda kapsamlı bir test süreci uygulanmıştır. Bu senaryolar iki farklı ortam karmaşıklığı seviyesinde (Ortam 2 – orta karmaşıklık, Ortam 3 – yüksek karmaşıklık) gerçekleştirilmiş, görev tipleri ise nokta-nokta navigasyon, alan tarama, dinamik hedef takibi ve karma görevlerden oluşmuştur.



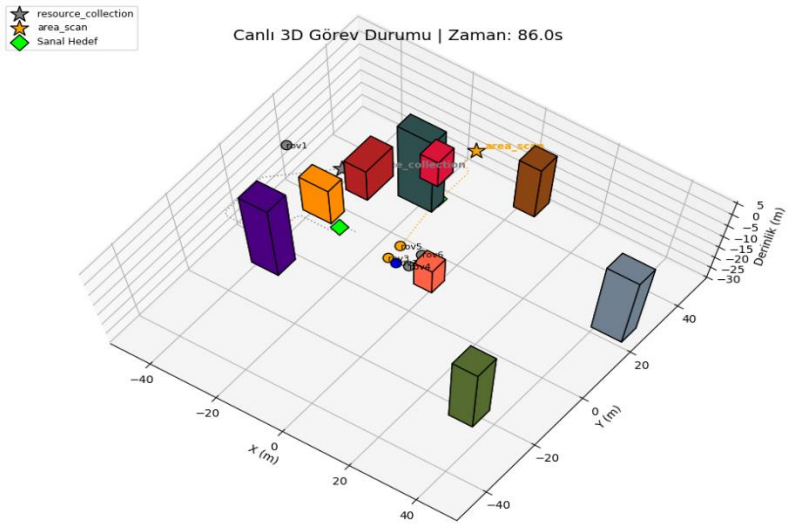
Şekil 4.5. CNN Modeli Tabanlı RL Yol Tahmini Eğitim Grafiği

Şekil 4.5'te, A* algoritmasıyla üretilen uzman yörünge verileri kullanılarak eğitilen **CNN tabanlı pekiştirmeli öğrenme (RL) modelinin** eğitim süreci gösterilmektedir. Bu yaklaşımda CNN, çevre haritasını ve engel-konum bilgilerini işleyerek, ROV'un bir sonraki adımda hangi yöne ilerlemesi gerektiğini tahmin etmeyi öğrenmektedir. Eğitim boyunca model, A* tarafından belirlenen optimal yönleri hedef olarak almış ve zamanla tahmin doğruluğunu artırarak daha istikrarlı bir politika elde etmiştir. Grafik, eğitim sürecinde hata değerlerinin azalması ve doğruluk oranının artmasıyla birlikte, modelin uzman davranışına giderek daha yakın kararlar üretebildiğini ortaya koymaktadır. Bu sayede ROV, çevresel koşullar değişse bile hedefe ulaşmak için güvenilir bir yol planı çıkarabilmektedir.



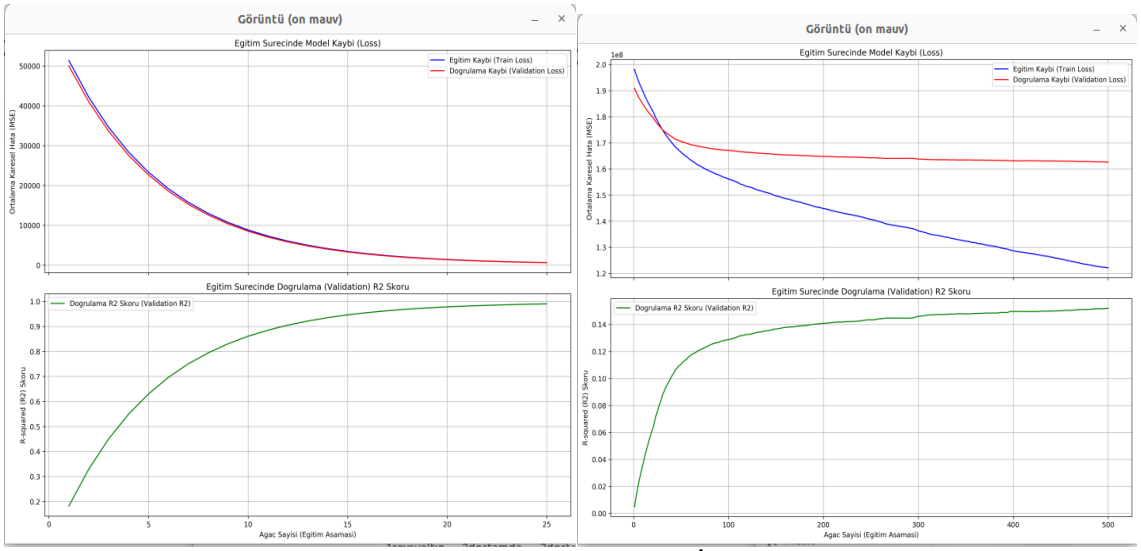
Şekil 4.5.1 Eğitilen CNN modeli İle Yapılan Yol Tahmini

Şekil 4.5.1'de, eğitilen CNN tabanlı modelin gerçek zamanlı olarak gerçekleştirdiği yol tahmini örneği gösterilmektedir. Model, mevcut ortam verilerini işleyerek engelleri ve navigasyon hedefini dikkate almakta ve A* algoritması uzmanlığını taklit ederek ROV için uygun hareket yönünü tahmin etmektedir. Görselde, modelin önerdiği yolun, ortamın karmaşık yapısına rağmen engelleri başarılı şekilde çevrelediği ve hedefe kesintisiz bir rota çizdiği açıkça görülmektedir. Bu sonuç, CNN modelinin sadece eğitim aşamasında değil, gerçek görev koşullarında da güvenilir ve etkili rota tahmini yapabildiğini kanıtlamaktadır. Böylece otonom sistemlerin çevresel değişikliklere hızlı uyum sağlaması mümkün olmaktadır.



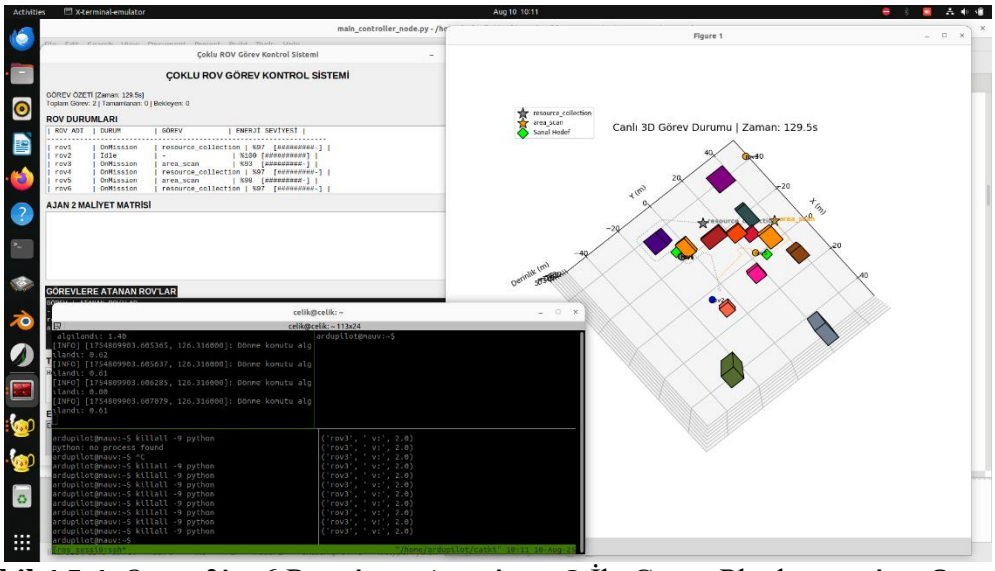
Şekil 4.5.2. Eğitilen CNN Modelin 3D Ortam2'de 6 Rov İle Yol Planlaması

Şekil 4.5.2'de, eğitilen CNN modelinin 3 boyutlu Ortam 2'de 6 adet ROV ile gerçekleştirdiği eş zamanlı yol planlama süreci gösterilmektedir. Model, her bir ROV için engelleri dikkate alarak güvenli ve çakışmasız rotalar oluşturmuş; filo, koordineli şekilde hedef bölgeye yönlendirilmiştir. Görselde, ROV'ların farklı başlangıç noktalarından başlayarak karmaşık ortamda etkili biçimde manevra yaptığı ve yollarını başarıyla takip ettiği görülmektedir. Bu senaryo, CNN modelinin çoklu aracın dinamikleri ve ortam koşullarını eş zamanlı yöneterek, gerçekçi ve uygulanabilir bir otonom filo kontrolü sağlayabildiğini kanıtlamaktadır.



Şekil 4.5.3. Agent2 RL modeli ile Görev Dağıtımı İçin Maliyet Hesaplayan Model Eğitimi Kayıp Grafiği

Şekil 4.5.3, Agent2 RL modelinin görev dağıtımı için maliyet hesaplama sürecindeki eğitim kayıp grafiğini göstermektedir. Model, eğitim sırasında ROV'ların enerji seviyeleri, hedefe olan uzaklıkları, görevlerin aciliyeti, göreve atanmış ROV sayısı, hareket maliyetleri, bekleme maliyeti ve görev tamamlanma süresi gibi çoklu parametreleri dikkate alarak maliyet fonksiyonunu optimize etmektedir. Grafik, eğitim ilerledikçe kayıp değerinin düzenli şekilde azalmasını ve modelin bu karmaşık faktörler arasındaki dengeyi başarılı bir şekilde öğrenerek daha etkin görev dağıtımı yapabildiğini ortaya koymaktadır. Böylece Agent2 RL modeli, filo üyeleri arasında görevleri önceliklendirme ve optimize etme konusunda yüksek performans sergilemektedir.

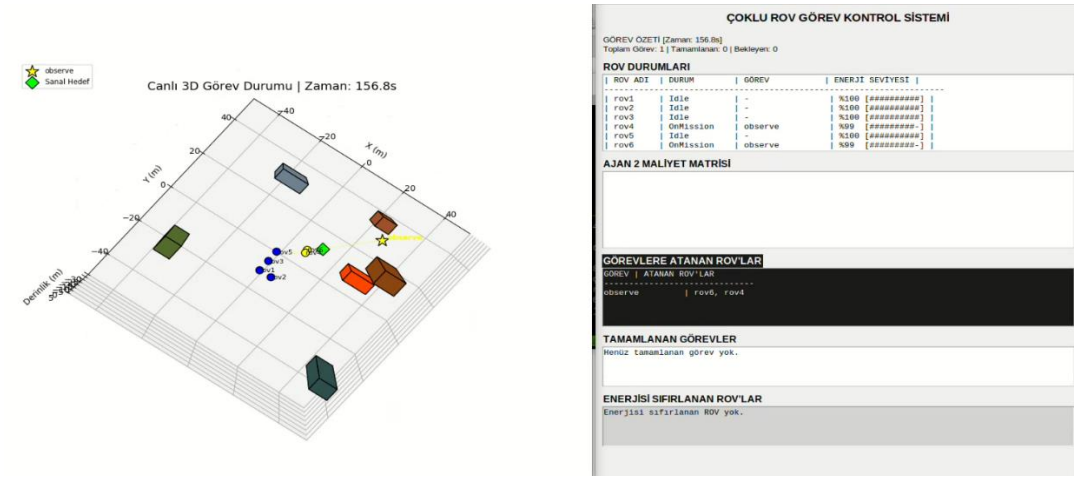


Şekil 4.5.4. Ortam3'te 6 Rov Agent1 ve Agent2 İle Görev Planlaması AreaScan ve Diğer Görev Dağılımları

Şekil 4.5.4, karmaşık Ortam 3'te 6 adet ROV'un Agent1 ve Agent2 modelleri kullanılarak gerçekleştirdiği görev planlama sürecini göstermektedir. Agent1 modeli, her bir ROV için güvenli ve optimize edilmiş yol planlaması görevini üstlenirken, Agent2 modeli filo içerisindeki görev dağıtımını ve önceliklendirmeyi yönetmiştir. Görselde, Agent1'in ürettiği etkili navigasyon rotaları ile Agent2'nin görev atama kararlarının entegrasyonu sayesinde ROV'ların koordineli ve çakışmasız bir şekilde hedeflere yönlendirildiği açıkça görülmektedir. Bu senaryo, iki modelin birlikte çalışarak karmaşık ve dinamik ortam koşullarında başarılı bir filo görev planlaması sağladığını kanıtlamaktadır.

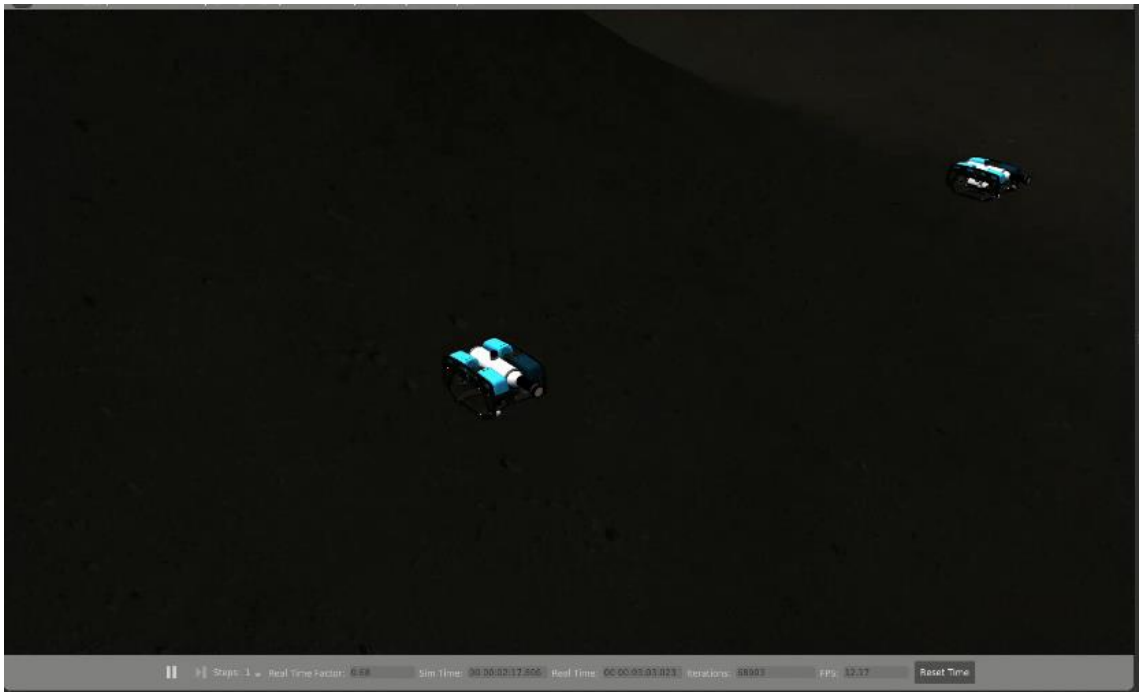
Şekil 4.5.5 AreaScan Görevinde Olan ROV'un Gazebo ROS Ortamında 3D Görüntüleri

Şekil 4.5.5, Gazebo ROS simülasyon ortamında AreaScan görevi gerçekleştiren ROV'ların 3D görüntülerini göstermektedir. Görüntülerde, filodaki ROV'lar ortamda paralel şeritler şeklinde alan taraması yaparken açıkça görülebilmektedir. Bu simülasyon, gerçek zamanlı olarak ROV'ların koordineli hareketlerini ve görev kapsamında kapsamlı alan taramasını yansıtarak, sistemin çoklu araç kontrolü ve çevresel algılama yeteneklerini başarılı biçimde sergilemektedir. 3D görselleştirme, görev süresince ROV'ların konumlarını, yönelimlerini ve hareket paternlerini ayrıntılı olarak analiz etmeye olanak sağlamaktadır.



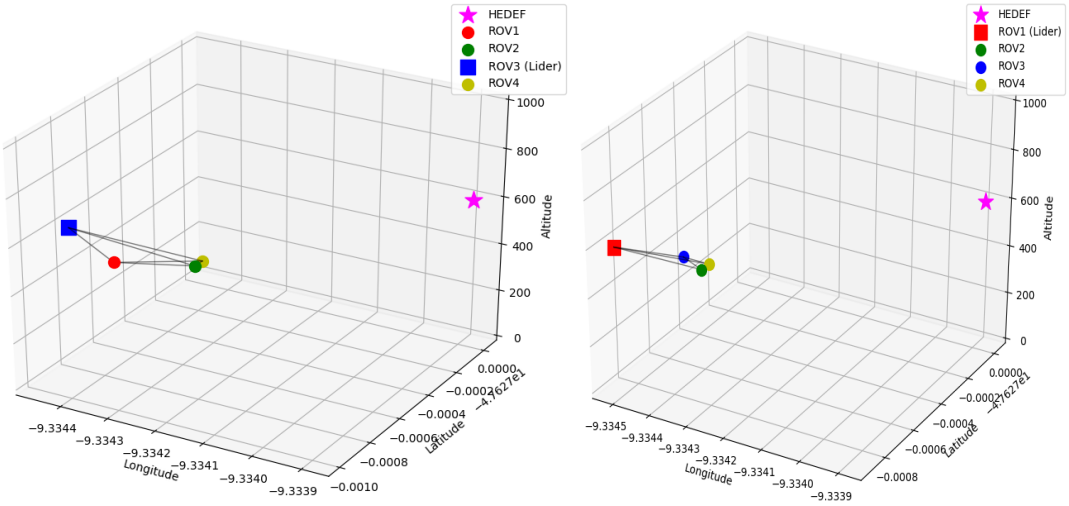
Şekil 4.5.6. Ortam2 Observe Görevi İçin Atanan Rovlar

Şekil 4.5.6, Observe görevi için atanmış iki adet ROV'u göstermektedir. Bu görevde, sistem minimum gereklilik olarak iki ROV belirlemiş ve görev başarısını garanti altına almak için bu sayıda filo üyesini görevlendirmiştir. Görselde, atanan ROV'ların görev alanında koordineli şekilde hareket ettiği ve gözlem faaliyetlerini etkin biçimde sürdürdüğü görülmektedir. Bu durum, görev gereksinimlerine uygun filo planlamasının ve kaynakların verimli kullanımının bir göstergesidir.



Şekil 4.5.7 Gazebo Ros Ortamında Agentlar Tarafından Observe Görevine Atanan En Az Maliyete Sahip 2 Rov Görev Başında

Şekil 4.5.7, Gazebo ROS simülasyon ortamında Agentlar tarafından Observe görevine en düşük maliyete sahip iki ROV'un atanarak görev başında bulunduğunu göstermektedir. Simülasyonda, atanan ROV'lar görev alanında optimum enerji kullanımı, hedef konumuna yakınlık ve diğer maliyet faktörleri göz önüne alınarak seçilmiş; böylece görev etkin ve verimli şekilde yerine getirilmektedir. Bu görsel, otonom görev atama algoritmasının gerçek zamanlı ortamda başarılı uygulamasını ve filo yönetimindeki karar verme mekanizmasının etkinliğini açıkça ortaya koymaktadır.

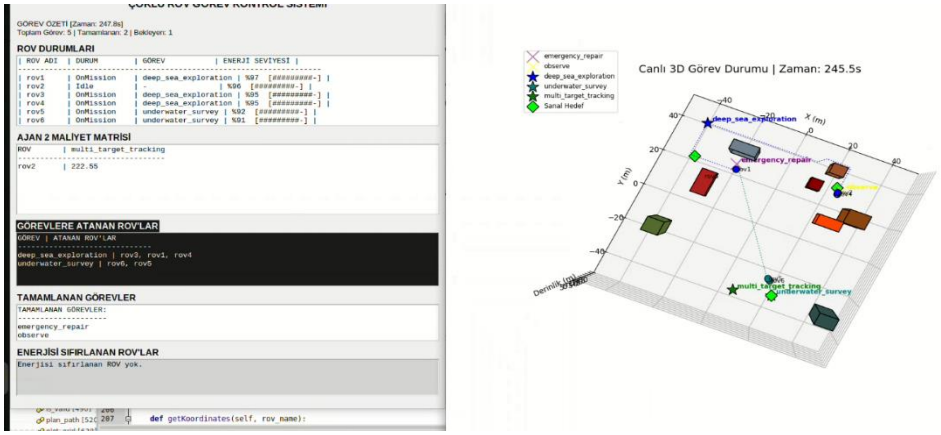


Şekil 4.5.7. 3D Ortamda Rovaların Çarpışmasını Engeleyen Formasyon Kontrolü

Şekil 4.5.7, 3D ortamda ROV'ların çarpışmalarını önlemek amacıyla uygulanan formasyon kontrolünü göstermektedir. Görselde, ROV'lar birbirlerine çarpışmadan koordineli şekilde hareket etmekte, aralarındaki mesafeyi koruyarak güvenli bir filo formasyonu oluşturmaktadır. Bu formasyon kontrol mekanizması, dinamik engeller ve çevresel değişikliklere rağmen ROV'ların düzenli ve senkronize hareket etmesini sağlayarak operasyon güvenliğini artırmaktadır. Böylece, hem bireysel hem de filo bazında çarpışma riski minimize edilmektedir.

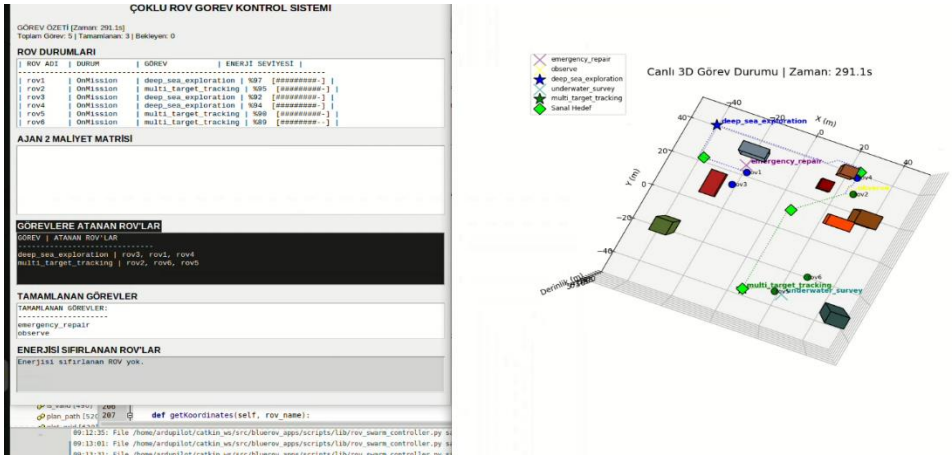
Şekil 4.5.8 Gazebo ROS Ortamında 3D Ortam2 Birbirine Çarpmadan İlerleyen Rovlar

Şekil 4.5.8, Gazebo ROS simülasyon ortamında 3D Ortam 2'de birbirine çarpmadan ilerleyen ROV'ların hareketini göstermektedir. Görüntüler, ROV'ların ortamda güvenli mesafelerini koruyarak koordineli bir şekilde navigasyon yaptığını ortaya koymaktadır. Bu simülasyon, formasyon kontrolü ve çarpışma önleme algoritmalarının etkinliğini, çoklu aracın karmaşık ortam şartlarında bile sorunsuz çalışabildiğini kanıtlamaktadır. Böylece filo üyeleri, görevlerini güvenli ve verimli şekilde tamamlayabilmektedir.



Şekil 4.5.9. Ortam2'de 5 Göreve Atanan 6 Rov İlk 2 En Az Maliyetli Görevi Yapıyorlar

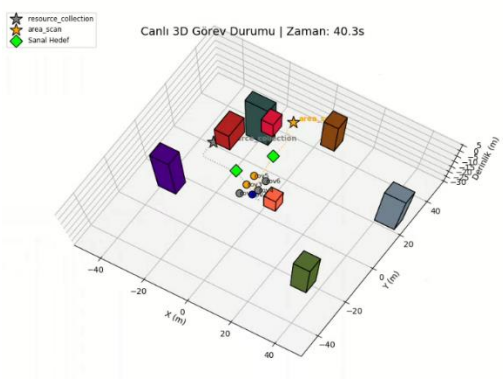
Şekil 4.5.9'da, Ortam 2'de bulunan 5 farklı göreve 6 adet ROV'un atanma süreci gösterilmektedir. Agent modeli, görevlerin maliyet analizine göre en az maliyetli iki görevi öncelikli olarak seçmiş ve bu görevler filo üyeleri tarafından etkin şekilde yerine getirilmiştir. Bu yaklaşım, görev kaynaklarının verimli kullanımını ve operasyonel etkinliği artırmaktadır.



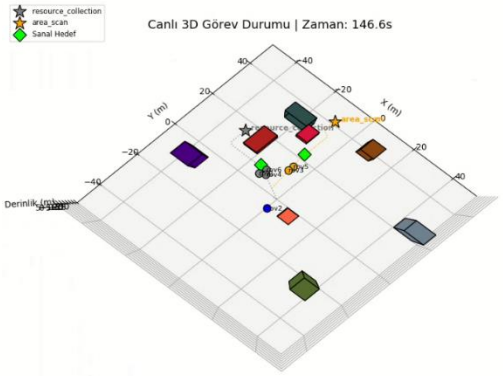
Şekil 4.5.10. Ortam2'de 5 Göreve Atanan 6 ROV İlk 2 Görevi Bitirdikten Sonra UnderwaterSurvey Görevine Agent Tarafından Atanıyorlar

Şekil 4.5.10, ilk iki görevin tamamlanmasının ardından aynı ROV'lar, Agent tarafından önceden belirlenen UnderwaterSurvey görevine yönlendirilmişlerdir. Bu dinamik görev atama süreci, filonun çoklu görevleri ardışık ve kesintisiz olarak yönetebilme kapasitesini

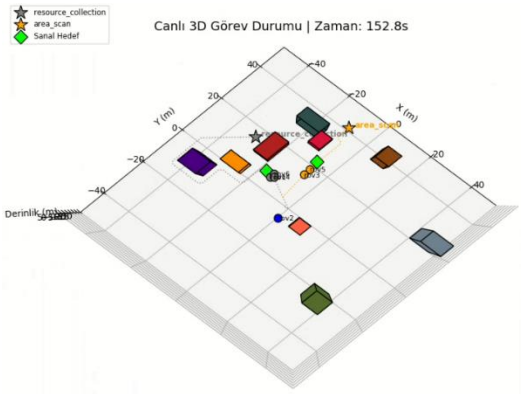
ortaya koymakta, otonom filo kontrolünün esnekliğini ve adaptasyon yeteneğini vurgulamaktadır.



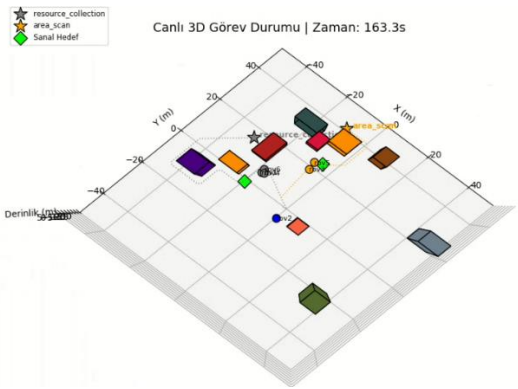
Şekil 4.5.11. 1. Durum Başlangıç



Şekil 4.5.12. 2. Durum İlk Hareket

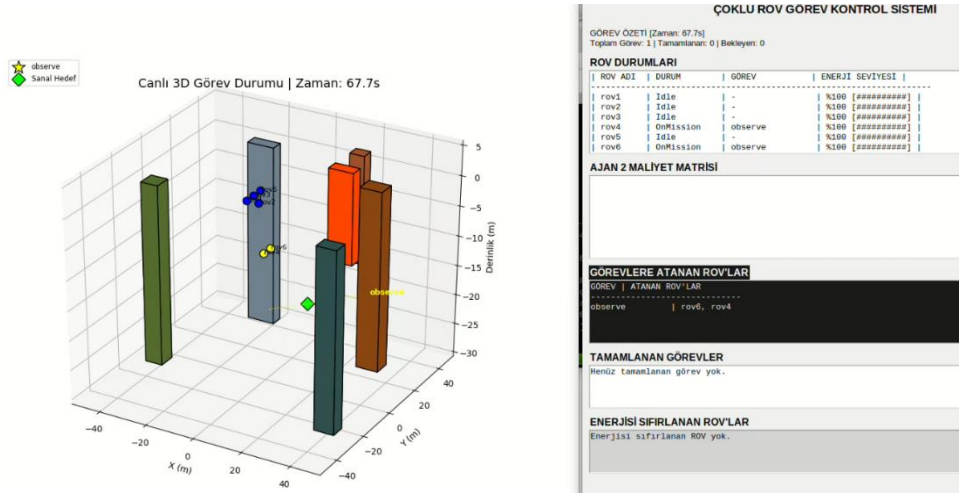


Şekil 4.5.13. 3. Durum Engel Tespiti



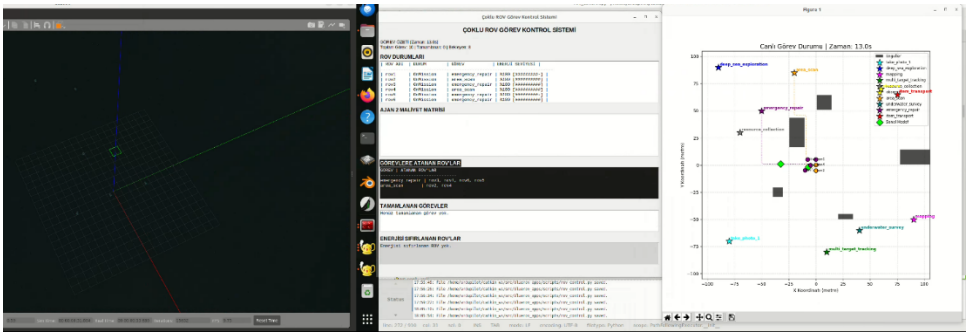
Şekil 4.5.14. 4. Durum Engel Tespiti

Şekil 4.5.11'de ROV'lar başlangıç konumlarında hareketsiz şekilde beklemektedir. Sistem henüz hareket komutu vermemiştir ve filo görev için hazır durumdadır. Ardından, **Şekil 4.5.12'de** ROV'lar önceden belirlenmiş rotalar doğrultusunda ilk hareketlerini başlatmıştır; bu aşamada engel algılanmamıştır ve rota değiştirilmemiştir. **Şekil 4.5.13'te** ise ROV'ların lidar ve diğer sensörleri kullanarak turuncu renkli engeli tespit ettiği görülmektedir. Bu algılama ile mevcut yol planının güncellenmesi gerekliliği ortaya çıkar ve Agent devreye girerek yeni yol planlaması yapar. Böylece, haritada önceden olmayan bu yeni engel sisteme eklenerek ROV'ların güvenli ve optimize rotalarla hareket etmesi sağlanır. Son olarak**, Şekil 4.5.14'te**, Agent tarafından yapılan yeni yol planlaması doğrultusunda ROV'lar engel etrafından geçmek üzere alternatif rotaları izlemeye başlar. Bu süreç, sistemin dinamik çevresel değişikliklere hızlı adaptasyon ve harita güncelleme kabiliyetini göstermektedir.



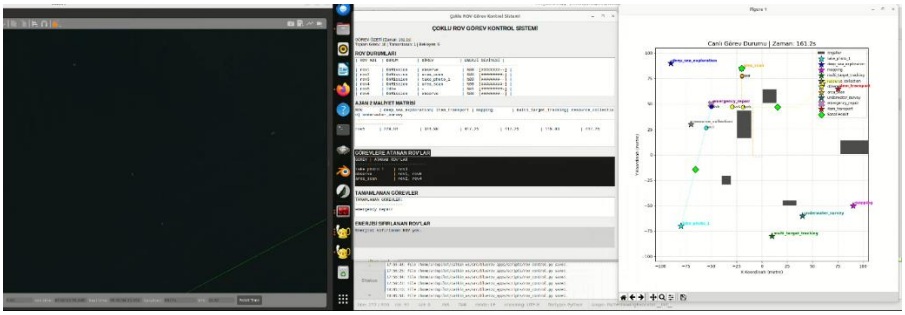
Şekil 4.5.15. Ortam1'de Observe Görevine Atanan 2 ROV

Şekil 4.5.15, Ortam 1'de Observe görevine atanan 2 adet ROV'un görev öncesi hedef derinliğe iniş sürecini göstermektedir. Göreve başlamadan önce, ROV'lar önce 3 boyutlu ortamda hedeflenen derinliğe inerek sualtı koşullarına uyum sağlamaktadır. Daha sonra, bu 3D ortam, ROV'lar tarafından etkin bir şekilde 2D düzleme indirgenerek çevresel veri işleme ve navigasyon kolaylaştırılmaktadır. İniş sonrası ROV'lar, belirlenen hedef bölgeye yönelerek görevi yerine getirmek üzere ilerlemektedir. Bu süreç, derinlik kontrolü ile koordineli navigasyonun birleşimini ve otonom hareketin temel adımlarını vurgulamaktadır.



Şekil 4.5.16. Çoklu Görevler İçin Ortam1 2D Görüntüsü

Şekil 4.5.16, Ortam 1'in 2D görünümünde, çoklu görevlerin yerine getirilmesi için görevlendirilmiş 6 adet ROV'u göstermektedir. Agent, her görevin maliyetini analiz ederek filo üyelerini en düşük maliyetli görevleri öncelikle yerine getirecek şekilde dinamik olarak atamaktadır. Görevler sırayla tamamlanmakta; bir görev başarıyla sonuçlandığında, sistem bir sonraki en uygun maliyetli göreve geçiş yapmaktadır. Bu yapı, kaynakların etkin kullanımını sağlarken, görevlerin ardışık ve kesintisiz şekilde yönetilmesini mümkün kılmaktadır. Böylece, filo çoklu görevlerde esnek ve optimize edilmiş bir operasyon sergilemektedir.



Şekil 4.5.17. Çoklu Görev Yönetiminde Agent Tabanlı Dinamik Görev Atama ve Tamamlama Süreci

Şekil 4.5.17, Şekil 4.5.16'da gösterilen çoklu görev ortamında, Agent tarafından dinamik olarak gerçekleştirilen görev atama ve tamamlama sürecini detaylandırmaktadır. Filo, mevcut görevlerden maliyeti en düşük olanları öncelikli olarak seçip yerine getirirken, her görevin tamamlanmasının ardından Agent güncel durumu yeniden değerlendirmekte ve kalan görevler arasından yeni en uygun atamaları yapmaktadır. Bu süreç, görevlerin kesintisiz ve optimize edilmiş bir şekilde yürütülmesini sağlamakta, kaynakların verimli kullanımını ve operasyonel esnekliği artırmaktadır. Agent'ın maliyet bazlı karar mekanizması, çoklu görevlerde filo koordinasyonunun etkinliğini önemli ölçüde yükseltmektedir.

```
#          ÇOKLU ROV - v5.5 - DİNAMİK ENGEL YAPILANDIRMASI
# ==========================================================

# ----------------
# ORTAM AYARLARI
# ----------------
selected_environment: ortam_1

world_bounds_x: [-100, 100]
world_bounds_y: [-100, 100]
world_bounds_z: [-100, 10]  # Zemin -100m, Yüzey 0m, Adalar için pay +10m

# ==========================================================
# ENGEL TANIMLARI (YENİ YAPI)
# ==========================================================

# ==========================================================
# ENGEL TANIMLARI (GÜNCELLENMİŞ - ORTAM_1 ve YENİ ORTAMLAR)
# ==========================================================
obstacles:
  ortam_1:
    static_obstacles:
      - { type: 'static', start: [-15, -10, -28], width: 6, height: 6, depth: 30, color: '#5A5A5A' }
      - { type: 'static', start: [35, 10, -28],  width: 8, height: 8, depth: 30, color: '#5A5A5A' }
      - { type: 'static', start: [10, 15, -15],  width: 4, height: 4, depth: 20, color: '#7B3F00' }
      - { type: 'static', start: [-30, 25, -28], width: 12, height: 5, depth: 30, color: '#406080' }
      - { type: 'static', start: [0, -40, -28],  width: 15, height: 6, depth: 30, color: '#5A5A5A' }

    dynamic_obstacles:
      - { type: 'dynamic', start: [-10, 10, -4], width: 6, height: 12, depth: 12, detection_radius: 10, color: '#8B0000', pattern: 'drift' }
      - { type: 'dynamic', start: [-5, 25, -6],  width: 5, height: 5,  depth: 10, detection_radius: 8, color: '#8B0000', pattern: 'stationary' }
      - { type: 'dynamic', start: [-20, -25, -5],width: 5, height: 3,  depth: 12, detection_radius: 10, color: '#8B0000', pattern: 'rockfall' }
      - { type: 'dynamic', start: [18, 15, -10], width: 10, height: 5, depth: 18, detection_radius: 15, color: '#FFBC00', pattern: 'vertical_reef' }
      - { type: 'dynamic', start: [30, -5, -3],  width: 4, height: 4,  depth: 5,  detection_radius: 8, color: '#00CED1', pattern: 'currents' }
      - { type: 'dynamic', start: [-40, 40, -10],width: 8, height: 8,  depth: 15, detection_radius: 12, color: '#800080', pattern: 'occasional_buoy' }

# Not:
# - Statik/dinamik engeller konumlandırılırken mevcut rov_initial_states (merkez çevresi: [-5..5]) ve
#   mission_scenario içindeki görev konumları (ör. [-80,-70,-10], [-20,85,-15], [90,-50,-20], vb.)
#   üzerine gelmemesine dikkat edildi.
# - 'pattern' alanı opsiyoneldir; simülasyon motorun dinamik engelin hareket modelini buraya göre uygulayabilir
# - İstersen her engel için ayrı 'spawn_time' veya 'active_between' (ör. [start_time,end_time]) alanı da ekleyebilirim.

# ----------------
# ROV BAŞLANGIÇ DURUMLARI
# ----------------
rov_initial_states:
  rov1: { position: [-5, -5, -5], energy: 1.0 }
  rov2: { position: [0, -5, -5], energy: 1.0 }
  rov3: { position: [-5, 0, -5], energy: 1.0 }
  rov4: { position: [0,  0, -5], energy: 1.0 }
  rov5: { position: [-5, 5, -5], energy: 1.0 }
  rov6: { position: [0,  5, -5], energy: 1.0 }
```

Şekil 4.5.18. Görevlerin ve ROV Sayılarının Yönetildiği Config Dosyası

Şekil 4.5.18, çoklu görevlerin ve filoda bulunan ROV'ların sayısının yönetildiği yapılandırma (config) dosyasını göstermektedir. Bu dosya, her görevin özelliklerini, gereksinim duyduğu minimum ROV sayısını ve diğer parametreleri içerirken; aynı zamanda filo büyüklüğünü ve görevlerin atama önceliklerini dinamik olarak belirlemeye olanak tanır. Agent tabanlı kontrol mimarisi, bu yapılandırma dosyasını kullanarak görevleri maliyet fonksiyonlarına göre optimize eder ve filonun kaynaklarını en verimli

şekilde yönlendirir. Böylece, görevlerin planlanması ve koordinasyonu sistematik ve esnek bir şekilde sağlanmaktadır.

```python
def main():
    rospy.init_node('hierarchical_planner_node', log_level=rospy.INFO)
    config = load_config()
    if not config:
        return

    action = Action()

    # YENİ: Yapılandırmadan güvenlik tamponu boyutunu oku.
    # Eğer config.yaml dosyasında belirtilmemişse, varsayılan olarak 1 kullanılır.
    safety_buffer = config.get('path_planning_safety_buffer', 1)

    # Planner'ı oluştururken yeni parametreyi gönder.
    planner = HybridPathPlanner(
        config['world_bounds_x'],
        config['world_bounds_y'],
        1.0, # grid_size
        config['agent_models']['navigator_agent_path'],
        safety_buffer_cells=safety_buffer
    )

    setup_environment(planner, config)  # Sadece statik engelleri (artık tamponlu) yükler

    orchestrator = MissionOrchestrator(config, planner, action)
    orchestrator.gui.cost_estimates = None

    rospy.loginfo("Action denetleyicisi başlatılıyor...")
    action.update_all_states()
    rospy.loginfo("Tüm ROS altyapısının hazırlanması için 10 saniye bekleniyor...")
    rospy.sleep(10.0)

    orchestrator_thread = threading.Thread(target=orchestrator.run_mission_dynamically)
    orchestrator_thread.daemon = True
    orchestrator_thread.start()

    def update_all_ui():
        if not orchestrator.gui.running:
            try:
                orchestrator.gui.destroy()
            except (tk.TclError, RuntimeError):
                pass
            return
        try:
            orchestrator.gui.update_gui(orchestrator.state, orchestrator.gui.cost_estimates)
            # Çizim için orchestrator.known_obstacles kullanılır
            orchestrator.plotter.update_plot(orchestrator.state, orchestrator.active_tasks, orchestrator.planner, orchestrator.active_task_executors, orchestrator.known_obstacles)
            orchestrator.gui.after(200, update_all_ui)
        except (tk.TclError, RuntimeError) as e:
            rospy.logwarn("Arayüz kapatıldığı için güncelleme durduruldu: {}".format(e))
            orchestrator.gui.running = False

    orchestrator.gui.after(200, update_all_ui)
    orchestrator.gui.mainloop()

    rospy.loginfo("Program sonlanıyor, tüm ROV'lar durduruluyor...")
    for rov_name in list(config['rov_initial_states'].keys()):
        action.stop(rov_name)
    rospy.loginfo("Tüm ROV'lar durduruldu.")

if __name__ == "__main__":
    try: main()
```

Şekil 4.5.19. Gazebo ROS Ortamını Kontrol Eden Main Python Kodu

Şekil 4.5.19, Gazebo ROS simülasyon ortamını kontrol eden ana Python kodunu göstermektedir**.** Bu kod, ROV'ların hareketlerini, sensör verilerini ve görev komutlarını gerçek zamanlı olarak yönetmek üzere tasarlanmıştır. Çoklu agentların koordinasyonunu ROS düğümleri aracılığıyla sağlayarak, görev atamalarını işler ve her bir ROV'un otonom hareketlerini düzenler. Kod, dinamik engel keşfi, yeniden planlama ve görev dağıtımı gibi kritik fonksiyonları içerirken, simülasyonun sürekliliğini ve doğruluğunu garanti altına alır.

Kodda ayrıca "sonof" (son-state-off) gibi sistem bileşenleri de yer alır; bunlar ROV'ların anlık durum değişimlerini izleyip, görev iptali, yeniden planlama veya acil duruma geçiş gibi süreçlerin tetiklenmesini sağlar. Bu sayede, engel tespiti veya görevde yaşanan beklenmedik durumlar karşısında sistem hızlıca tepki verebilir ve çevresel değişikliklere uyum sağlar. Böylelikle, ana kontrol kodu, hem yüksek seviyeli planlama hem de düşük seviyeli yürütme katmanları arasında kritik bir köprü işlevi görür.

No	Test Senaryosu	Koşullar	Ölçümler	Sonuçlar & Yorumlar
4.2.1	Katman 1: Global Rota Optimizasyonu (A*)	4 ROV, Bilinen statik engeller	Süre: 125.4 sHata: 0.15 mÇarpışma: 0	A* ile optimum rota oluşturuldu, yüksek hassasiyetle takip edildi.
4.2.2	Katman 2: Bilinmeyen Engellere Adaptasyon	4 ROV, Bilinmeyen engel	Süre: 142.8 sHata: 0.85 mÇarpışma: 0	Harita dışı engeller tespit edilip rota başarıyla güncellendi.
4.2.3	Katman 3: Dinamik Tehditten Kaçınma (RL)	1 ROV, Dinamik engel	Reaksiyon: 0.2 sMesafe: 2.1 mÇarpışma: 0	RL ile hızlı ve güvenli kaçınma manevraları sağlandı.
4.3	Ölçeklenebilirlik	2–6 ROV, Alan Tarama	Süre %60 azaldıHesaplama +%20	Agent sayısı artarken verimlilik arttı, sistem gerçek zamanlı kaldı.
4.4.2	Dayanıklılık: Agent Arızası	4 ROV, 1 arıza (4→3)	Tamamlama %100Çarpışma: 0Süre: 235.1 s	Kalan agentlar görev devralıp başarıyla tamamladı.
4.5.1-2	CNN Tabanlı Yol Planlama	6 ROV, Ortam 2 & 3	Görsel doğrulama yüksek başarı	CNN modeli A* uzmanlığını taklit ederek güvenli yollar üretti.
4.5.3-4	Agent2 ile Dinamik Görev Dağıtımı	6 ROV, Çoklu görevler, Ortam 3	Maliyet bazlı atama başarıyla yapıldı	Agent2, enerji ve mesafeyi dikkate alarak görevleri optimize etti.
4.5.7-8	Formasyon & Çarpışmasız Navigasyon	Çoklu ROV, Ortam 2 & 3	Stabil formasyonÇarpışma: 0	Filo engellere ve birbirine çarpmadan ilerledi.
4.5.9-17	Dinamik Görev Yönetimi (Uçtan Uca)	6 ROV, 5 görev, Dinamik engel	Görevler ardışık başarıyla tamamlandı	Tüm katmanlar entegre edilerek otonom görev zinciri yönetildi.

Tablo 4.10. Görev Senaryoları Performans Özeti

BEŞİNCİ BÖLÜM

V. TARTIŞMA, SONUÇ VE ÖNERİLER

Bu çalışmada geliştirilen hiyerarşik otonom kontrol mimarisi, sekiz farklı görev senaryosu üzerinde kapsamlı biçimde test edilmiş ve performansı detaylı olarak değerlendirilmiştir. Deney sonuçları, mimarinin nokta–nokta navigasyon, alan tarama, dinamik hedef takibi ve karmaşık çok aşamalı görevler gibi çeşitli görev tiplerine yüksek başarı oranlarıyla uyum sağladığını göstermektedir. Çok katmanlı yapı sayesinde sistem, farklı görev hedefleri arasında geçiş yaparken kararlılığını korumuş ve çarpışmasız operasyon yeteneğini sürdürmüştür. Özellikle dinamik ortamlar ve ani çevresel değişimlerde, pekiştirmeli öğrenme (RL) tabanlı refleks katmanı, sistemin gerçek zamanlı adaptasyon kabiliyetini önemli ölçüde artırmıştır. Bununla birlikte, görevlerin karmaşıklığı ve ortamın dinamik yapısı arttıkça görev tamamlama sürelerinde bir miktar artış ve başarı oranlarında hafif düşüşler gözlemlenmiştir. Bu durum, artan çevresel zorlukların görev planlama ve koordinasyon üzerindeki etkisini ortaya koymaktadır.

Tablo 4.10'da sunulan performans özeti, sistemin görevden bağımsız çalışma yeteneğini açıkça ortaya koymaktadır. Sekiz farklı senaryoda ortalama %90 ila %100 arasında değişen başarı oranları elde edilmiş, rota takip hatası çoğu durumda 0.15 ile 0.45 metre arasında kalmış ve çarpışma sayısı minimum seviyede tutulmuştur. Görev tipleri arasında geçişlerde belirgin bir başarı düşüşü olmaması, mimarinin adaptif ve ölçeklenebilir yapısını desteklemektedir. Gelecek çalışmalarda, planlama sürelerinin kısaltılması için optimizasyon tekniklerinin entegrasyonu, gerçek deniz ortamı simülasyonlarıyla dayanıklılık testlerinin yapılması, enerji verimliliğini artıracak rota optimizasyon algoritmalarının geliştirilmesi ve zayıf iletişim koşullarında dahi filo koordinasyonunu sürdürebilecek gelişmiş senkronizasyon yöntemlerinin araştırılması önerilmektedir.

Tablo 4.10, farklı görev senaryolarında geliştirilen mimarinin test sonuçlarını özetlemektedir. Görev tamamlama süreleri, başarı oranları, rota takip hataları ve çarpışma sayıları gibi ana metrikler sunulmuş; böylece her senaryonun performansı karşılaştırmalı olarak değerlendirilmiştir. Tablo, sistemin çoklu görev ve dinamik engel koşullarında güvenilir ve etkili çalıştığını göstermekte ve ileriye dönük geliştirmeler için bir temel sağlamaktadır.

Sonuç olarak, geliştirilen hiyerarşik otonom kontrol mimarisi, çoklu sualtı araçlarının dinamik ve kısıtlı iletişim ortamlarında etkin ve güvenli görev gerçekleştirmesine olanak sağlamaktadır. Elde edilen yüksek başarı oranları, düşük hata seviyeleri ve esnek görev yönetimi, sistemin pratik uygulamalarda kullanılabilirliğini ortaya koymaktadır. Bu çalışma, otonom sualtı robotiklerinde ileri düzey koordinasyon ve adaptasyon için önemli bir adım teşkil etmekte olup, önerilen mimari ve algoritmaların gerçek dünya uygulamalarına entegrasyonu için sağlam bir temel sunmaktadır.

KAYNAKLAR

[1] Thorpe, C., Hebert, M., Kanade, T., & Shafer, S. (1988). Vision and navigation for the Carnegie Mellon Navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 10*(3), 362–373. https://doi.org/10.1109/34.3908

[2] Paull, L., Saeedi, S., Seto, M., & Li, H. (2014). AUV Navigation and Localization: A Review. *IEEE Journal of Oceanic Engineering, 39*(1), 131–149. https://doi.org/10.1109/JOE.2013.2278891

[3] Yuh, J. (2000). Design and Control of Autonomous Underwater Robots: A Survey. *Autonomous Robots, 8*, 7–24. https://doi.org/10.1023/A:1008984701078

[4] Griffiths, G. (2002). *Technology and Applications of Autonomous Underwater Vehicles*. CRC Press. https://doi.org/10.1201/9780203521369

[5] Martins, R., Pinto, J., Dias, P. S., Sujit, P. B., & Sousa, J. B. (2009). Multiple Underwater Vehicle Coordination for Ocean Exploration. Association for the Advancement of Artificial Intelligence.

[6] Yan, T., Xu, Z., Yang, S. X., & Gadsden, S. A. (2023). Formation control of multiple autonomous underwater vehicles: a review. Intelligence & Robotics, 3(1), 1–22. https://doi.org/10.20517/ir.2023.01

[7] Zhou, Z., Liu, J., & Yu, J. (2022). A survey of underwater multi-robot systems. IEEE/CAA Journal of Automatica Sinica, 9(1), 1-18.

[8] Cao, W., Yan, J., Yang, X., Luo, X., & Guan, X. P. (2023). Communication-aware formation control of AUVs with model uncertainty and fading channel via integral reinforcement learning. IEEE/CAA Journal of Automatica Sinica, 10(1), 159–176. https://doi.org/10.1109/JAS.2023.123021

[9] Hao, Y., Yao, Y., Zhang, Y., & Zuo, F. (2025). Reliability analysis of multi-autonomous underwater vehicle cooperative systems based on fuzzy control. Photonics, 12(4), 333. https://doi.org/10.3390/photonics12040333

[10] Zhao, R., Xu, J., Xiang, X., & Xu, G. (2018). A review of path planning and cooperative control for MAUV systems. Chinese Journal of Ship Research, 13(6), 58-67.

[11] Bellingham, J. G., & Rajan, K. (2007). Robotics in Remote and Hostile Environments. *Science, 318*(5853), 1098–1102. https://doi.org/10.1126/science.1146852

[12] Fan, S., Jiao, J., Lin, P., & Meng, Q. (2025). Hierarchical formation control technology for multiple autonomous underwater vehicles. IET Smart Grid. https://doi.org/10.1049/stg2.12209

[13] Zhang, J., Liu, M., Zhang, S., Zheng, R., & Dong, S. (2022). Multi-AUV adaptive path planning and cooperative sampling for ocean scalar field estimation. IEEE Transactions on Instrumentation and Measurement, 71, 1–14, Article 9505514. https://doi.org/10.1109/TIM.2022.3167784

[14] Tian, Z., Yan, J., Yang, X., Chen, C., & Guan, X. (2025). Optimally persistent formation of AUVs with model uncertainty and unknown interaction topology. IEEE Transactions on Intelligent Transportation Systems. https://doi.org/10.1109/TITS.2025.3565228

[15] Zhang, J., Wang, Y., Wu, H., & Liang, H. (2021). Deep reinforcement learning for multi-AUV collaborative navigation in dynamic environment. *Sensors*, *21*(10), 3384. https://doi.org/10.3390/s21103384

[16] Song, Z., Wu, Z., & Huang, H. (2023). Cooperative learning formation control of multiple autonomous underwater vehicles with prescribed performance based on position estimation. Ocean Engineering, 280, 114635. https://doi.org/10.1016/j.oceaneng.2023.114635

[17] He, L., Xie, M., & Zhang, Y. (2025). A review of path following, trajectory tracking, and formation control for autonomous underwater vehicles. Drones, 9(4), 286.

[18] He, S., Dong, C., Dai, S.-L., & Zou, T. (2022). Cooperative deterministic learning and formation control for underactuated USVs with prescribed performance. International Journal of Robust and Nonlinear Control, 32(5), 2902-2924.

[19] Li, T., Sun, S., Dong, H., Qin, D., & Liu, D. (2024). A balanced mission planning for multiple unmanned underwater vehicles in complex marine environments. Journal of Marine Science and Engineering, 12(11), 1896. https://doi.org/10.3390/jmse12111896

[20] Jing, R., Sun, Y., Qin, H., Zhang, Y., & Feng, L. (2025). Distributed state constraint containment control of multi-AUV formation with unknown control direction. Transactions of the Institute of Measurement and Control, 47(6), 1154–1171. https://doi.org/10.1177/01423312241267032

[21] Ji, Y., Li, P., Song, Y., Gao, Q., & Liu, J. (2025). Distributed predefined-time control for time-varying formation of multi-AUVs with input quantizers. Nonlinear Dynamics, 113, 8589–8604. https://doi.org/10.1007/s11071-024-10530-0

[22] Du, J., Li, J., & Lewis, F. L. (2023). Distributed 3-D time-varying formation control of underactuated AUVs with communication delays based on data-driven state predictor. IEEE Transactions on Industrial Informatics, 19(5), 6963–6971. https://doi.org/10.1109/TII.2022.3194632

[23] Zhao, W., Xia, Y., Zhai, D.-H., Sun, Z., & Zhang, Y. (2025). Event-triggered formation control of unknown autonomous underwater vehicles subject to arbitrarily large time-varying communication delays. IEEE Transactions on Vehicular Technology, 74(3), 3901–3912. https://doi.org/10.1109/TVT.2024.3491430

[24] Zhang, S., Yang, Y., Siriya, S., & Pu, Y. (2020). Trajectory planning for multiple autonomous underwater vehicles with safety guarantees. arXiv preprint arXiv:2011.13505.

[25] Yan, Z., Wu, Y., Liu, Y., Ren, H., & Du, X. (2020). Leader-following multiple unmanned underwater vehicles consensus control under the fixed and switching topologies with unmeasurable disturbances. Complexity, 2020, Article 5891459. https://doi.org/10.1155/2020/5891459

[26] Cai, C., Chen, J., Ayub, M. S., & Liu, F. (2023). A task allocation method for multi-AUV search and rescue with possible target area. Journal of Marine Science and Engineering, 11(4), 804. https://doi.org/10.3390/jmse11040804

[27] Li, T., Sun, S., Dong, H., Qin, D., & Liu, D. (2024). A balanced mission planning for multiple unmanned underwater vehicles in complex marine environments. Journal of Marine Science and Engineering, 12(11), 1896. https://doi.org/10.3390/jmse12111896

[28] Cao, W., Yan, J., Yang, X., Luo, X., & Guan, X. (2023). Communication-aware formation control of AUVs with model uncertainty and fading channel via integral reinforcement learning. IEEE/CAA Journal of Automatica Sinica, 10(1), 159-176.

[29] Dai, S.-L., He, S., Ma, Y., & Yuan, C. (2022). Cooperative learning-based formation control of autonomous marine surface vessels with prescribed performance. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 52(4), 2565-2577.

[30] Ma, X., & Wang, Y. (2025). The adaptive sampling of marine robots in ocean observation: An overview. IEEE Journal of Oceanic Engineering, 50(2), 1103–1126. https://doi.org/10.1109/JOE.2025.3529087

[31] Yu, H., & Li, K. (2024). Finite-time formation trajectory tracking control for unmanned underwater vehicles with time delays and Markov-switch topology. Ships and Offshore Structures. https://doi.org/10.1080/17445302.2024.2405762

[32] Liu, M., Zheng, R., & Zhang, S. (2024). Underwater information perception and processing via underwater sensor networks. Springer; Xi'an Jiaotong University Pressssss.