T.C.
YALOVA ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DERİN PEKİŞTİRMELİ ÖĞRENME İLE OTONOM GEZGİN ROBOTLARIN DİNAMİK ORTAMLARDA NAVİGASYONU

YÜKSEK LİSANS TEZİ
Koray ÖZDEMİR
195105003

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Dr. Öğr. Üyesi Adem TUNCER

ARALIK 2021

T.C.
YALOVA ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DERİN PEKİŞTİRMELİ ÖĞRENME İLE OTONOM GEZGİN ROBOTLARIN DİNAMİK ORTAMLARDA NAVİGASYONU

YÜKSEK LİSANS TEZİ
Koray ÖZDEMİR
195105003

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Dr. Öğr. Üyesi Adem TUNCER

ARALIK 2021

ÖNSÖZ

Yüksek lisans eğitimimin her aşamasında bana destek olup yol gösteren, mesleki bilgilerini ve deneyimlerini paylaşarak katkıda bulunan sayın danışman hocam Dr. Öğr. Üyesi Adem TUNCER'e teşekkür ederim. Ayrıca benden desteğini hiçbir zaman esirgemeyen değerli eşime ve tüm aileme teşekkür ederim.

Bu tez çalışmasının desteklenmesine olanak sağlayan Yalova Üniversitesi Bilimsel Araştırma Projeleri (BAP) Koordinasyon Birimine (Proje No: 2020/YL/0014) teşekkür ederim.

Aralık 2021

Koray ÖZDEMİR
Bilgisayar Mühendisi

İÇİNDEKİLER

KISALTMALAR

CPU : Merkezi İşlem Birimi (Central Processing Unit)
DC : Doğru Akım (Direct Current)
D3QN : Düello Çift Derin Q Öğrenme **(**Dueling Double Deep Q Learning)
GA : Genetik Algoritma (Genetic Algorithm)
GA3C : Asenkron Avantaj Aktör-Eleştirmen Ağı (Asynchronous Advantage Actor-Critic Network)
GPU : Grafik İşlem Birimi (Graphics Processing Unit)
LIDAR : Işık Tespiti ve Uzaklık Tayini (Light Detection and Ranging)
OpenCV : Açık Kaynak Bilgisayarlı Görme (Open Source Computer Vision)
PC : Kişisel Bilgisayar (Personal Computer)
RGB : Kırmızı Yeşil Mavi (Red Green Blue)
RGB-D : Kırmızı Yeşil Mavi-Derinlik (Red Green Blue-Depth)
ROS : Robot İşletim Sistemi (Robot Operating System)
SBC : Tek Kartlı Bilgisayar (Single Board Computer)
SLAM : Eşzamanlı Konumlandırma ve Haritalama İşlemi (Simultaneous Localization and Mapping)
TPU : Tensör İşlem Birimi (Tensor Processing Unit)
YSA : Yapay Sinir Ağları (Artificial Neural Networks)

ÇİZELGE LİSTESİ

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

DERİN PEKİŞTİRMELİ ÖĞRENME İLE OTONOM GEZGİN ROBOTLARIN DİNAMİK ORTAMLARDA NAVİGASYONU

ÖZET

Son yıllarda teknolojinin gelişmesine bağlı olarak gezgin robotların kullanımı yaygınlaşmakta ve robotların görevlerini otonom olarak gerçekleştirmeleri ile ilgili çalışmaların da sayısı gün geçtikçe artmaktadır. Gezgin robotların hedef tanıma, navigasyon ve engelden sakınım gibi pek çok görevi otonom olarak yerine getirmesi onların üstesinden gelmesi gereken problemlerin başında yer almaktadır. Literatürde sabit engellerin bulunduğu ortamlar ile ilgili pek çok çalışma bulunmaktadır; ancak gerçek dünyada çevrede sabit engellerin yanında hareketli engeller de bulunmaktadır. Gezgin robotların insanların veya diğer hareketli engellerin bulunduğu ortamlarda görevlerini yerine getirebilmeleri oldukça önemlidir.

Pekiştirmeli öğrenme bulunulan ortamla ilgili herhangi bir ön bilgi gerektirmemesi ve insan öğrenmesine benzer bir yaklaşım göstermesi nedeniyle son yıllarda üzerinde çalışmaların yoğunlaştığı bir makine öğrenmesi yöntemidir. Pekiştirmeli öğrenmenin en sık kullanılan yöntemi Q öğrenme algoritmasıdır. Q öğrenme algoritması ile derin sinir ağlarının birleştirilmesiyle derin Q öğrenme (veya derin Q ağlar) ortaya çıkmıştır. Robotların engellerle dolu çevrelerde herhangi bir ön bilgiye ihtiyaç duymadan kendi kendilerine öğrenme gerçekleştirebilmeleri, derin Q öğrenmenin geleneksel algoritmalara göre tercih edilmesinin en önemli sebebidir.

Bu tez çalışmasında, bir gezgin robotun herhangi bir harita bilgisine ihtiyaç duymadan verilen hedefleri tanıyarak ve engellerden kaçınarak hedeflere ulaşması amaçlanmaktadır. Bu amaçla derin Q öğrenme algoritmalarından, düello çift derin Q öğrenme (D3QN) algoritmasını temel alan üç farklı ağ modeli tasarlanmıştır. Derinlik kamerasından elde edilen derinlik görüntüleri modellerin eğitim sürecinde kullanılmıştır. Gezgin robotun hedefini tanıması ve hedefe doğru yönelmesi için ağ modellerine ek bilgi olarak hedefe olan yön açısı ve mesafe bilgileri verilmiştir. Eğitim ve test süreçleri Robotik işletim sistemi (ROS) ve Gazebo benzetim ortamı üzerinde gerçekleştirilerek sistemin gerçek dünyada uygulanabilirliği gösterilmiştir. Sonuç olarak önerilen modelin gezgin robotların hedeflerini tanıyıp engellerden sakınarak hedeflerine ulaşmalarında başarılı olduğu görülmüştür.

Anahtar Kelimeler: Gezgin Robot; Pekiştirmeli Öğrenme; Derin Q Öğrenme; Engelden Sakınım

NAVIGATION OF AUTONOMOUS MOBILE ROBOTS IN DYNAMIC ENVIRONMENTS WITH DEEP REINFORCEMENT LEARNING

ABSTRACT

In recent years, depending on the development of technology, the use of mobile robots is becoming widespread and the number of studies on autonomous robots is increasing day by day. The fact that mobile robots can perform many tasks autonomously, such as target recognition, navigation and obstacle avoidance, is one of the problems that they must overcome. There are many studies in the literature on environments with static obstacles. However, in the real world, there are dynamic obstacles as well as static obstacles in the environment. It is very important that mobile robots can perform their tasks in environments with people or other dynamic obstacles.

Reinforcement learning is a method of machine learning that has been focused on in recent years, since it does not require any prior knowledge of the environment and shows a similar approach to human learning. The most commonly used algorithm of reinforcement learning is the Q learning algorithm. Deep Q learning (or Deep Q networks) has emerged by combining the Q learning algorithm with deep neural networks. In recent years, the use of deep Q learning algorithms in robot navigation has been increasing. The most important reason why deep Q learning is preferred over classical algorithms is that robots can learn on their own without the need for any prior knowledge in environments full of obstacles.

In this thesis, it is aimed for a mobile robot to reach the targets by recognizing the given targets and avoiding obstacles without the need for any map information. For this purpose, three different network models were designed based on the dueling double deep Q learning (D3QN) algorithm, one of the deep Q learning algorithms. Depth images obtained from the depth camera were used in the training step of the models. In order for the mobile robot to recognize its target and steer towards the target, the direction angle and distance information to the target are given as additional information to the network models. Training and testing steps were carried out on Robotic operating system (ROS) and Gazebo simulation environment, and the applicability of the system in the real world was demonstrated. As a result, it has been seen that the proposed model is successful in reaching the targets of mobile robots by recognizing their targets and avoiding obstacles.

Keywords: Mobile Robot; Reinforcement Learning; Deep Q Learning; Obstacle Avoidance

1. GİRİŞ

Teknoloji alanındaki hızlı gelişmelere paralel olarak yapay zekâ kavramı hayatımızın tüm alanında kendini göstermeye başlamıştır. Yapay zekâya sahip kişisel asistanlardan e-ticaret ve bankacılık işlemlerine, siber güvenlik alanından sağlık hizmetlerine kadar her alanda günlük yaşamın vazgeçilmez unsurlarından biri olarak yapay zekâ karşımıza çıkmaktadır. Yapay zekânın ne olduğunu anlayabilmek için öncelikle zekâ kavramının tanımlanması gerekmektedir. Zekâ genel bir ifadeyle düşünme, akıl yürütme, kavrama, analiz etme ve sonuç çıkarma gibi yeteneklerin tümü şeklinde tanımlanabilir. Yapay zekâ teriminin isim babası olarak John McCarty görülmektedir. 1956 yılında yaptığı tanımlamaya göre yapay zekâyı akıllı makineler ve özellikle akıllı bilgisayar programları üretme bilimi ve mühendisliği olarak ifade etmiştir [1]. Her ne kadar isim olarak tanımlanması McCarty tarafından yapılmışsa da fikir olarak ortaya atılması daha eskidir. 1943 yılında Warren S. McCulloch ve Walter Pitts tarafından yayınlanan "A Logical Calculus of the Ideas Immanent in Nervous Activity" adlı makalede Turing makinelerine benzer şekilde çalışan, nöronlar ve düğümler kullanan bir işlemci fikrinden bahsedilmiştir [2]. Yapay zekâ alanındaki bilimsel çalışmalar ilk tanımlandığı günden itibaren devam etmesine rağmen asıl hızlı ilerlemeler 2000'li yıllardan sonra gerçekleşmiştir. Donanım alanında meydana gelen gelişmeler ve özellikle işlemcilerin hesaplama kapasitelerinin hızla artması yapay zekâ alanındaki çalışmalara hem hız kazandırmış hem de ilginin artmasını sağlamıştır. Böylelikle pek çok farklı sektörde yapa zekâ uygulamaları kendilerine yer bulabilmiştir. Bu tezin temel çalışma alanını oluşturan robotik alanında da yapay zekâ uygulamaları sıklıkla kullanılmaktadır.

Günlük hayatta bazı kavramlar genel bir ifadeyle yapay zekâ olarak ifade edilse de yapay zekâ kavramını bütünsel olarak kullanmak çok doğru bir yaklaşım olmayacaktır. Her ne kadar yapay zekâ kapsayıcı bir tanım olsa da pratikte birbirinden oldukça farklılık gösteren alt dallara ve çalışma alanlarına ayrılmıştır. Bu tezin kapsamında yapay zekânın alt dallarından biri olan makine öğrenmesine ait çalışmalardan faydalanılmıştır. Makine öğrenmesi, insanların öğrenme süreçlerini taklit etmek için verileri ve çeşitli algoritmaları kullanma üzerine yoğunlaşan bir yapay zekâ alt dalı olarak tanımlanabilir. Gözetimli, gözetimsiz ve pekiştirmeli öğrenme olmak üzere alt dalları bulunmaktadır. Bu alt dallara ait tanımlar aşağıda verilmiştir [3]:

- Gözetimli Öğrenme: Öğrenme daha önceden belirlenmiş olan etiketlere sahip olan veri kümeleri üzerinden gerçekleştirilir. Eğitim sürecinin ardından uygulanan test sürecinde veri kümesinde bulunmayan farklı veriler üzerinden tahminde bulunulur ve başarı oranları çeşitli metrikler kullanılarak hesaplanır.

- Gözetimsiz Öğrenme: Gözetimli öğrenmeden temel farklılığı etiketlenmiş veri kümesi kullanılmamasıdır. Amaç veri kümesinde bulunan gizli yapıları ve ilişkileri tespit edebilmektir.

- Pekiştirmeli Öğrenme: Pekiştirmeli öğrenme, davranışçılıktan esinlenen, öznelerin bir ortamda en yüksek ödül miktarına ulaşabilmesi için hangi eylemleri yapması gerektiğiyle ilgilenen bir makine öğrenmesi yaklaşımıdır. Bu problem genelliğinden ötürü oyun kuramı, kontrol kuramı, yöneylem araştırması, bilgi kuramı, benzetim tabanlı en iyileme ve istatistik gibi diğer birçok dalda kullanılmaktadır.

Makine öğrenmesi kavramıyla birlikte sıklıkla adı geçen diğer bir kavram ise derin öğrenmedir. Kısaca insan beyninin yapısını ve işlevlerini taklit etme üzerine kurulu olan bir makine öğrenmesi alt sınıfıdır. Bir ya da birden fazla yapay sinir ağı içeren yapılar bu kapsamda değerlendirilebilir.

Bu tez kapsamında makine öğrenmesi alt dallarından olan pekiştirmeli öğrenme, gezgin robotların navigasyon problemlerinin çözümünde kullanılmıştır. Teknolojinin gelişmesiyle birlikte gezgin ve otonom robotların günlük yaşamımızda kullanımı her geçen gün artmaktadır. Son yıllarda evlerimize giren robot süpürgeler bu duruma iyi bir örnektir. Gezgin robotlardan beklenen en önemli yeteneklerden biri otonom navigasyon yeteneğidir. Daha açık bir şekilde ifade edilecek olursa, bu robotların verilen görevleri çevrelerindeki nesnelere çarpmadan ve zarar vermeden mümkün olan en kısa sürede gerçekleştirmeleri beklenmektedir [4]. Bu görevlerin daha başarılı şekilde yapılmasını hedefleyen çeşitli algoritmalar bulunmaktadır [5]. Geleneksel algoritmaların yanında derin sinir ağları gibi derin öğrenme algoritmaları da bu alanda kullanılmaktadır; ancak hem eğitim süreci için gerekli olan veri miktarının büyüklüğü hem de bu verilerin yorumlanması sürecinde yaşanan zorluklar yeni yöntem ve algoritmaların kullanılmasını zorunlu kılmıştır. Pekiştirmeli öğrenme bu yeni yöntemlerin arasında önde gelmektedir. Çevreyle ilgili bir harita bilgisine ihtiyaç duyan geleneksel algoritmaların aksine herhangi bir ön bilgi gerektirmemesi

pekiştirmeli öğrenmenin gezgin robotlar alanında kullanımını cazip kılmaktadır. Derin pekiştirmeli öğrenme kavramı ise pekiştirmeli öğrenme ve derin sinir ağlarının birlikte kullanımı ile ortaya çıkmıştır. Derin pekiştirmeli öğrenme alanında Mnih ve diğ. [6] tarafından yapılan çalışma öncü olarak gösterilebilir. Bu çalışmada 49 farklı atari oyunun oynanmasında ham görüntüler ile birlikte derin pekiştirmeli öğrenme kullanılmış ve %75 başarı elde edildiği ifade edilmiştir. Çalışmada görüntülerin işlenmesi sürecinde evrişimsel sinir ağları kullanılmıştır. Bu tez kapsamında da yapılan çalışmaya [6] benzer bir yaklaşım kullanılmıştır. Gezgin robotun üzerinde bulunan derinlik kamerasından elde edilen derin görüntüler evrişimsel sinir ağı modeli üzerinden işlenmiş ve derin pekiştirmeli öğrenme algoritmalarından biri olan düello çift derin Q öğrenme algoritması (D3QN) kullanılarak eğitilmiştir. Gezgin robotun karmaşık şekilde konumlandırılmış sabit ve hareketli engellerin bulunduğu bir çevrede verilen hedeflere herhangi bir ön bilgi verilmeksizin engellerden kaçınarak ulaşması beklenmiştir. Bu süreçte robot, insan öğrenmesine benzer şekilde önceki deneyimlerinden de faydalanmıştır.

1.1 Tezin Amacı ve Katkısı

Bu tez çalışmasında, otonom gezgin bir robotun sabit ve dinamik engellerin bulunduğu bilinmeyen bir çevrede verilen hedeflere engellerden sakınarak ulaşması hedeflenmiştir. Bu amaçla D3QN algoritması ve RGB-D kameradan alınan derinlik görüntülerinin yanında normal kameradan görüntü işleme yöntemleri kullanılarak elde edilen mesafe ve yön bilgileri birlikte kullanılmıştır. Çalışmanın katkılarından biri, gezgin robotun çevre hakkında hiçbir ön bilgiye gereksinim duymadan hedeflere ulaşabilmesidir. Hem engellerden sakınma hem de hedef tanımlama aşamalarında herhangi bir harita ve konum bilgisi kullanılmamıştır. Bu bakımdan geliştirilen modelin çevreden bağımsız bir şekilde öğrenme gerçekleştirebildiği ve bu sayede farklı çevrelerde herhangi bir ön işleme gereksinim duyulmaksızın kullanılabileceği deneysel çalışmalar sonucunda görülmüştür. Çalışmanın öne çıkan diğer bir özelliği ise literatürdeki çalışmaların çoğunluğunun aksine modelin dinamik ortamlarda eğitilip test edilmesidir. Çünkü gerçek dünya, insanlar, hayvanlar ve çeşitli hareketli engellerle doludur. Bu nedenle geliştirilen modelin gerçek dünyadakine benzer dinamik ortamlarda eğitilip test edilmesi önem arz etmektedir. Çalışmada bu durum göz önüne alınarak geliştirilen modelin gerçek dünyada kullanılacak olan gezgin robotlar üzerinde de çalıştırılabileceği sonucuna varılmıştır.

Tez çalışmasının literatüre katkıları aşağıdaki gibi sıralanabilir:

- Otonom gezgin bir robotun D3QN algoritması, derinlik görüntüleri, yön ve mesafe bilgileri kullanarak sabit ve dinamik engellerin bulunduğu bilinmeyen bir çevrede verilen hedeflere ulaşmasını sağlayan bir model önerilmiştir.
- Sabit engellerin yanında dinamik engeller de kullanılarak gerçek dünyaya benzer bir çevrede modelin eğitilip test edilmesi sağlanmıştır.
- Hedeflerin tanınması, mesafe ve yön bilgilerinin tespit edilmesi tek bir RGB kamera yardımıyla görüntü işleme yöntemleri kullanılarak yapılmış ve elde edilen veriler D3QN algoritmasının eğitimi sırasında derinlik görüntüleri ile eş zamanlı olarak kullanılmıştır.

1.2 Literatür Araştırması

Otonom gezgin robotların verilen görevleri gerçekleştirmek için bir insanın kontrol etmesine ihtiyaç duymamaları, maliyet ve güvenlik açısından sıklıkla tercih edilmelerinin en önemli nedenidir. Günümüzde kullanımları her ne kadar ağırlıklı olarak savunma sanayiinde olsa da günlük yaşamda da görülmeye başlamıştır. Evlerde kullanılmaya başlanan temizlik robotları bu duruma uygun bir örnek olarak verilebilir. Bu robotlar otonom şekilde evlerin haritasını çıkarmakta ve bu haritaya uygun şekilde temizlik görevlerini yerine getirmektedir [7]. Navigasyon yetenekleri bir otonom gezgin robotun başarılı kabul edilebilmesi için gerekli olan en önemli özelliklerdendir. Bu robotların verilen görevleri bazen tek başlarına bazen de sürü halinde [7] en kısa sürede ve engellerden kaçınarak yapabilmeleri beklenmektedir.

Navigasyon problemi üç kısımdan meydana gelmektedir. Bunlar haritalama, konumlandırma ve yol planlamadır. Haritalama, gezgin robotun üzerindeki sensörler sayesinde bilinmeyen bir çevrenin özelliklerini çıkarabilmesi ve çevreyi tanımlaması olarak ifade edilirken, konumlandırma ise robotun içinde bulunduğu çevrenin neresinde olduğunu belirleyebilmesidir. Haritalama ve konumlandırma probleminin çözülebilmesi amacıyla Kalman filtresi ve Rao-Blackwellized parçacık filtreleri gibi geleneksel yaklaşımlar kullanılmaktadır [8]. LIDAR ve RGB kameralar gibi sensörler de haritalandırma sürecinde sıklıkla tercih edilmektedir.

Yol planlama belirtilen hedefe ulaşabilme olarak tanımlanabilir. Robotlarda yol planlamasının yapılması için istatistiksel, olasılıksal ve yapay zekâ temelli

algoritmalar sıklıkla kullanılmaktadır. Yol planlaması alanında bilinen temel ve başarılı algoritmalardan biri A-yıldız (A*) algoritmasıdır. A* algoritması hedefe hızlı şekilde ulaşma konusunda başarılı sonuçlar vermektedir. Standart A* algoritmasının yanında algoritmanın geliştirilmiş versiyonları da yol planlaması alanında kullanılmaktadır. Sudhakara ve Ganapathy yaptıkları çalışmada [9] mevcut A* algoritmasında iyileştirilmeler yapmışlardır. Çalışmada sabit hedeflerin bulunduğu bilinmeyen bir ortamda robotun engellere çarpmadan hedefe ulaşması amaçlanmıştır. A* algoritmasına dönüş sayısı adlı bir parametre eklenerek robot simülasyon ortamında çalıştırılmıştır. Sonuç olarak geliştirilmiş A* algoritmasının orijinal algoritmaya göre daha iyi performans gösterdiğini belirtmişlerdir. Levy uçuşu, bulanık mantık, kalman filtresi, petri ağı gibi algoritmalar da robot sistemlerinde kullanılan algoritmalar arasında sayılabilir [10-13]. Bu algoritmalar bazen tek başlarına kullanılmış bazen de birlikte kullanılarak melez yaklaşımlar geliştirilmiştir.

Doğadaki canlıların davranışlarını taklit etme üzerine kurulu olan sürü zekâsı algoritmaları da robotlarda yol planlaması alanında önemli bir yer tutmaktadır [14]. Zhang [15] tarafından yapılan çalışmada geliştirilmiş çok amaçlı bir parçacık sürü optimizasyonu algoritması, bilinen bir çevre üzerinde engellerden ve tehlikeli noktalardan sakınarak hedefe ulaşma amacıyla kullanılmıştır. Çalışmada parçacık sürü optimizasyonu algoritması öz uyumlu mutasyon ile kullanılmış ve elde edilen sonuçlar mevcut parçacık sürü algoritmasıyla karşılaştırılmıştır.

Dewang ve diğ. [16] robotların farklı çevrelerde engellerden kaçınarak hedefe ulaşmalarını sağlayan geliştirilmiş bir parçacık sürü optimizasyonu algoritması önermişlerdir. Çalışmada robotun hedefe ve engellere olan uzaklıkları amaç fonksiyonu olarak seçilmiş ve optimizasyonu geliştirilen algoritma üzerinden yapılmıştır. Yapılan testler sonucunda önerilen yöntemin hem keşif hem de sömürü oranlarının daha başarılı olduğu ayrıca yerel minimum noktalarında takılmaların da engellendiği belirtilmiştir.

Mousavi [17] genetik algoritma (GA) ve parçacık sürü optimizasyonu kullanarak melez bir model geliştirmiştir. Çalışmada endüstride nakliye amacıyla kullanılan robotların görevleri temel alınmıştır. Elde edilen sonuçlarda geliştirilen melez modelin %79.8 oranında başarı gösterdiği, parçacık sürü optimizasyonu ve genetik algoritmanın ise sırasıyla %69.4 ve %74 oranında başarılı oldukları ifade edilmiştir.

Che ve diğ. [18] yaptıkları çalışmada bilinmeyen çevrelerde robotların hedeflerine ulaşmaları için geliştirilmiş bir karınca kolonisi algoritması önermişlerdir. Çalışmada çevre modelinin oluşturulması aşamasında ızgara yöntemi kullanılmıştır. Ayrıca karınca kolonisi algoritması parametrelerinden olan feromon güncellemesi için kurt kolonisi algoritmasından faydalanılmıştır. Böylece yerel minimumdan kaçınma ve yakınsama hızının arttırılması konusunda gelişme sağlanmıştır. Sonuç olarak önerilen algoritmanın, mevcut karınca kolonisi algoritmasına göre hedefe ulaşma süresi bakımından daha yüksek başarı sağlandığı belirtilmiştir.

Rao ve diğ. [19] tarafından yapılan çalışmada yol planlaması için gri kurt kolonisi algoritması kullanılmıştır. Çalışmada robotun farklı çevrelerde farklı boyutlardaki engellerden sakınarak hedefe ulaşması hedeflenmiştir. Simülasyon ortamında yapılan testlerle elde edilen sonuçlar olasılıksal yol haritası yöntemiyle kıyaslanarak gri kurt kolonisi algoritmasının özellikle sivri kenarlı engellerden kaçma konusunda başarılı olduğu belirtilmiştir. Ayrıca robotun hem hedefe varma süresi hem de gerçekleştirdiği manevra sayısı göz önüne alındığında etkili sonuçlar alındığı ifade edilmiştir.

HadiAbbas ve Ali [20] çevrimdışı otonom robotların yol planlamasında yapay arı kolonisi algoritmasını kullanmışlardır. Çalışmalarında 2-boyutlu üç farklı çevrede robotun başlangıç noktasından hedef noktasına engellere çarpmadan en kısa sürede ulaşması hedeflenmiştir. Sonuçlar karınca kolonisi algoritması ve parçacık sürü optimizasyonu ile karşılaştırılmıştır. Üç farklı çevrenin ikisinde algoritmaların yakın sonuçlar verdiği, birinde ise yapay arı kolonisi algoritmasının daha başarılı olduğu belirtilmiştir.

Nayyar ve diğ. [21] tarafından yapılan çalışmada Arrhenius denklemi ile geliştirilmiş yapay arı kolonisi algoritması robot yol planlaması probleminin çözümünde kullanılmıştır. Çalışmada yapay arı kolonisinin problemlerinden olan erken yakınsama ve durgunluk problemlerine çözüm bulmayı amaçlamışlardır. Yapılan testler sonucunda elde edilen bulgular standart yapay kolonisi algoritması ve parçacık sürü optimizasyonu ile karşılaştırılmıştır. Sonuç olarak önerilen algoritmanın robot yol probleminin çözümünde daha iyi performans gösterdiği belirtilmiştir.

Bu tez çalışmasında kullanılan derin Q ağlara temel olacak çalışmalar öncelikle Q öğrenmenin robot yol planlaması alanında kullanılması ile başlamıştır. Khriji ve diğ. [22] yaptıkları çalışmada gezgin robotların yol planlaması için Q öğrenme

kullanmışlardır. Çalışmada Q öğrenmenin temel sorunlarından olan tablo boyutunun aşırı artması ve ödüllerin durum uzayında çok seyrek olması problemi, bulanık mantık ve ön bilgi kullanılarak çözülmeye çalışılmıştır. Testler hem simülasyon hem de fiziksel ortamda yapılmıştır.

Pekiştirmeli öğrenme ve derin sinir ağlarının ilkelerinin birlikte kullanılmasıyla geliştirilen derin Q ağlar son yıllarda robot yol planlaması alanında sıklıkla kullanılmaya başlanmıştır. Surmann ve diğ. [23] tarafından yapılan çalışmada 2-boyutlu lazer tarayıcı ve RGB-D kamera kullanılarak bilinmeyen çevrelerde robotun hedeflere ulaşması amaçlanmıştır. Çalışmada derin Q öğrenme algoritmalarından biri olan asenkron avantaj aktör-eleştirmen ağı (GA3C) kullanılmıştır. Robot kontrol ağı yüksek hızlı ve paralel simülasyon ortamında eğitilmiştir. Modelin test edilmesi ise Turtlebot 2 robot kullanılarak fiziksel ortamda gerçekleştirilmiştir.

Chen ve diğ. [24] 3-boyutlu bir lazer tarayıcı ve derin pekiştirmeli öğrenme kullanarak fiziksel ortamda çalışan bir robot geliştirmişlerdir. Tamamen otonom bir şekilde hareket eden robot yayaların yoğunlukta olduğu alanlarda ve insan hareket hızına benzer şekilde çalıştırılmıştır. Mevcut çalışmalar daha çok eşleştirme ve insan yollarını takip etme üzerine çalışırken bu çalışmada toplum içerisinde neyin yapılıp neyin yapılmayacağı şeklinde sosyal normlar da dikkate alınmıştır.

Bae ve diğ. [25] tarafından gerçekleştirilen çalışmada çoklu robotların farklı çevrelerde uyumlu şekilde hedeflere ulaşması amaçlanmıştır. Geleneksel yaklaşımların aksine çoklu robot görevlerinde her bir robotun diğerleriyle iş birliği yaparak bağımsız şekilde hareket etmeleri hedeflenmiştir. Çalışmada robotun ortamdaki görüntü bilgilerini kullanarak durumu analiz edip gezinmesi için evrişimsel sinir ağları ve derin Q öğrenme kullanılmıştır. Elde edilen sonuçlar farklı çevrelerde A* ve D* algoritmalarıyla karşılaştırılmış ve çalışmada kullanılan yöntemin daha iyi sonuçlar verdiği belirtilmiştir.

Quan ve diğ. [26] yaptıkları çalışmada yol planlaması için derin pekiştirmeli öğrenme ve yinelemeli sinir ağlarını kullanmışlardır. Metodun etkinliğini test etmek için ızgara harita tabanlı 2-boyutlu bir çevre, 3-boyutlu simülasyon çevresi ve fiziksel ortam kullanılmıştır. Test sonuçları göz önüne alınarak önerilen algoritmanın yol bulma ve hedefe varış süresi konularında gelişme sağladığı ifade edilmiştir.

Weideman [27] tarafından yapılan çalışmada D3QN algoritması kullanan bir gezgin robot iki farklı çevre üzerinde eğitilmiş ve verilen hedeflere engellere çarpmadan ulaşması beklenmiştir. Bu çalışmada rastgele şekilde sabit engeller kullanılmıştır. Sonuç olarak geliştirilen modelin %88 oranında hedefe ulaşma başarısı gösterdiği belirtilmiştir.

Tezin bundan sonraki kısmı temel olarak iki kısımdan oluşmaktadır. 2. bölümde yapay sinir ağları, evrişimsel sinir ağları, pekiştirmeli öğrenme, Q öğrenme ve gezgin robot navigasyon problemleri gibi konular hakkında bilgilendirmeler yapılmıştır. 3. ve 4. bölümlerde ise D3QN algoritması yardımıyla bir gezgin robotun Gazebo simülasyon ortamı üzerinden eğitim süreci anlatılmış ve elde edilen test sonuçları gösterilmiştir.

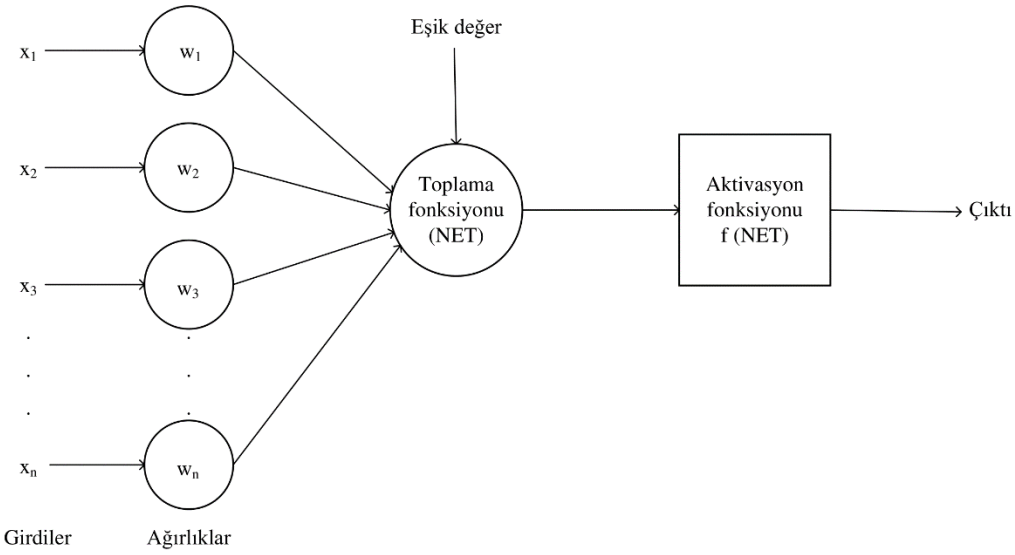
2. GENEL BİLGİLER

2.1 Yapay Sinir Ağları

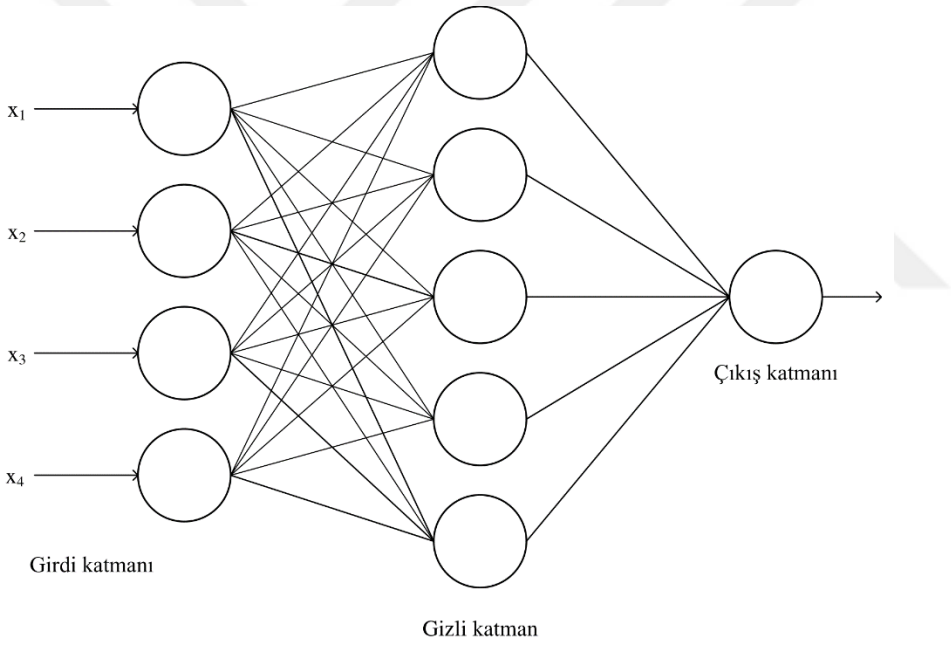
Yapay sinir ağları (YSA) insan beyninin işlevlerinin taklit edilmesiyle geliştirilmiş adaptif sistemlerdir [28]. İnsan beynindeki sinir hücrelerinden esinlenilen ve düğüm adı verilen yapılar ile onları birbirine bağlayan bağlantılardan meydana gelmektedir. Sınıflandırma, kümeleme, örüntü ilişkilendirme, tahmin, kontrol uygulamaları, optimizasyon ve araştırma problemleri gibi alanlarda kullanılmaktadır.

Tek bir sinir hücresinin matematiksel modellemesi ilk kez McCulloch ve Pitts tarafından 1943 yılında yapılmıştır [2]. 1958 yılında ise Frank Rosenblatt tarafından McCulloch ve Pitts'in sinir hücresi modeline bir öğrenme yöntemi olarak ilk algılayıcı (perseptron) geliştirilmiştir [28]. Tek katmanlı ve çok katmanlı olmak üzere iki çeşit algılayıcı modeli bulunmaktadır. Tek katmanlı algılayıcılar yalnızca doğrusal olarak ayrılabilen desenleri öğrenebilirken çok katmanlı algılayıcılar daha karmaşık durumların çözümünde kullanılan ve daha yüksek işlem gücüne ihtiyaç duyan yapılardır.

Kendi belleğine sahip olan her bir düğüm diğer düğümlerden ya da çevreden gelen girdilere sahiptir. Her bir düğümde gelen girdi ile ağırlık değerleri (w) çarpılır ve eşik değeri (b) eklenerek toplam değer elde edilir. Daha sonra ise aktivasyon fonksiyonu yardımıyla çıktı değeri üretilir. Aktivasyon fonksiyonu kullanılmasının amacı sinir hücresine doğrusal olmayan özellik kazandırmaktır. Böylece doğrusal olmayan karmaşık fonksiyonların öğrenilmesi sağlanmaktadır. En çok kullanılan aktivasyon fonksiyonları birim basamak fonksiyonu, Sigmoid, Tanh ve Rectified linear unit (ReLU) şeklindedir. En bilinen YSA tam bağlantılı katmanlı sinir ağlarıdır. Her bir sinir hücresi kendisinden önce gelen sinir hücresinin çıkış değerini girdi olarak alır. Ağ modeli bir girdi katmanı, bir çıkış katmanı ve farklı sayıda gizli katmandan oluşur. Derin YSA kavramı bu gizli katmanların birden fazla kullanılmasını ifade etmektedir. YSA'da eğitim süreci ise geri yayılım olarak adlandırılmaktadır. Geri yayılım işleminin amacı en uygun ağırlık değerlerinin bulunmasını sağlamaktır. Şekil 2.1'de bir yapay sinir ağı hücresi, şekil 2.2'de ise bir yapay sinir ağı modeli gösterilmiştir.



Şekil 2.1: Yapay sinir hücresi



Şekil 2.2: Bir yapay sinir ağı modeli

Bir yapay sinir ağının performansı aşağıdaki kriterler yönünden değerlendirilir [6]:

- Sonuçların kalitesi: Gerçek sonuçlar ve tahmin edilen sonuçlar arasındaki hata ölçüsü ile belirlenebilir.
- Genelleştirilebilme: Eğitim verisi üzerinde yüksek başarı sağlayan bir modelin benzer başarıyı test verileri üzerinden de gösterebilmesi yani farklı durumlarda da uygulanabilmesidir.

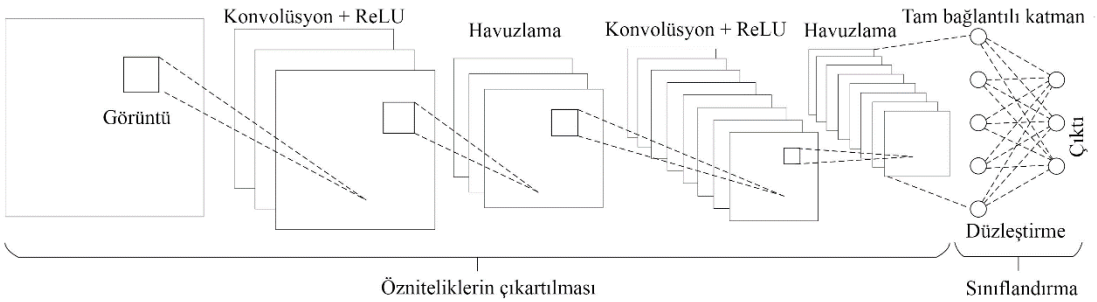
- Tüketilen kaynaklar: Modelin eğitilmesi için geçen süre ve gerekli olan hesaplama kaynakları bu kapsamdadır.

2.1.1 Evrişimsel Sinir Ağları

Evrişimsel sinir ağları çoğunlukla görüntü analizinde kullanılan bir derin sinir ağı alt sınıfıdır. Görüntü ve video tanımada, öneri sistemlerinde, görüntü sınıflandırmasında ve doğal dil işlemede sıklıkla kullanılmaktadır. Yaygın şekilde kullanılması 1998 yılında Yann LeCunn tarafından geliştirilen LeNet adlı evrişimsel sinir ağı ile başlamıştır [29]. Bu çalışmada LeNet el yazısı ve makine üretimi karakter tanımlamada kullanılmıştır. Evrişimsel sinir ağları standart yapay sinir ağlarının aksine özellikle yüksek boyutlu görüntü verilerinin sınıflandırılmasında yüksek performans göstermektedir [30]. Standart algoritmalarda manuel olarak yapılması gereken verilerden özellik çıkarımı işlemleri, evrişimsel sinir ağlarında otomatik olarak gerçekleştirilmektedir. Öte yandan standart yapay sinir ağlarına göre dezavantajı, evrişimsel sinir ağlarının yüksek işlem gücüne ihtiyaç duymasıdır. Bu sorunun üstesinden gelmek için yüksek boyutlu bellekler ve güçlü GPU'lar kullanılmaktadır. Bir evrişimsel sinir ağı farklı katmanlardan oluşmaktadır. Bunlar:

- Konvolüsyon katmanı (Convolutional layer): Görüntünün özniteliklerinin çıkarıldığı katmandır.
- Doğrusal olmayan katman (Non-linearity layer): Sistemin doğrusal niteliğinin ortadan kaldırıldığı katmandır.
- Havuzlama katmanı (Pooling layer): Uzaysal boyutun ve parametrelerin azaltıldığı katmandır.
- Düzleştirme katmanı (Flattening layer): Verilerin düzleştirilerek tam bağlantılı katman için hazırlandığı katmandır.
- Tam bağlantılı katman (Fully connected layer): Standart yapay sinir ağının kullanıldığı katmandır.

Evrişimsel sinir ağlarında aktivasyon fonksiyonu olarak genellikle ReLU kullanılmaktadır. Şekil 2.3'te örnek bir evrişimsel sinir ağı mimarisi gösterilmiştir.

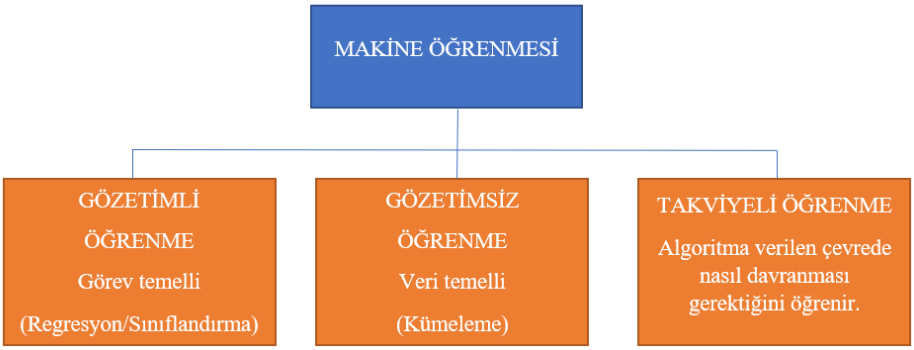


Şekil 2.3: Bir evrişimsel sinir ağı mimarisi

2.2 Pekiştirmeli Öğrenme

Derin öğrenme algoritmalarının kullanımı her ne kadar yeni olmasa da hesaplama gücündeki iyileştirmeler ve kullanılan verilerin büyüklüğü son yıllarda bu alandaki gelişmelere hız kazandırmıştır [31]. Donanım alanında merkezi işlem birimi (CPU)'nden grafik işlem birimi (GPU)'ne ve sonraki süreçte ise tensör işlem birimi (TPU)'ne geçiş olarak özetlenebilecek gelişmeler, bilgisayarların hesaplama gücünü arttırırken hesaplama süresini oldukça azaltmıştır [32,33].

Pekiştirmeli öğrenme, verilen bir çevrede farklı eylemleri gerçekleştirerek en yüksek ödülü elde etmeyi amaçlayan bir makine öğrenmesi alanıdır. Ajan adı verilen yapı tarafından önceki tecrübelerden de faydalanılarak başarılı ve başarısız denemeler yapılması sonucunda öğrenme gerçekleştirilir. Ajanın hangi eylemleri yapması gerektiği söylenmeden en uygun eylem dizisinin kendisi tarafından bulunması beklenir [34]. Pekiştirmeli öğrenme diğer makine öğrenmesi çeşitlerinden olan gözetimli ve gözetimsiz öğrenmeden farklılık göstermektedir. Şekil 2.4'te pekiştirmeli öğrenme diğer makine öğrenmesi alt dalları birlikte gösterilmiştir.



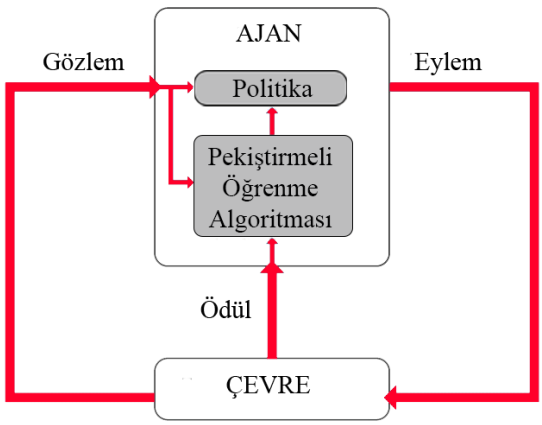
Şekil 2.4: Makine öğrenmesi alt dalları

Gözetimli öğrenmede etiketlere sahip veri setleri üzerinden eğitim gerçekleştirilir. Bu aşamada hangi eylemlerin doğru hangilerinin ise yanlış olduğu önceden

belirlenmektedir. Daha sonra ise eğitim verilerinde bulunmayan veriler üzerinden tahminde bulunulması istenir. Çözüm aranan temel problemler sınıflandırma, regresyon, nesne tespiti vb. olarak ifade edilebilir. Gözetimli öğrenmede eğitim verilerinin büyük olması başarıyı arttırmaktadır. Gözetimsiz öğrenmede ise daha önceden sağlanan etiketlenmiş veriler yoktur. Temel amaç veri kümesinde bulunan gizli yapıları ve ilişkileri tespit etmektir. Çözüm aranan problemler kümeleme, öznitelik tespiti, boyut azaltma vb. şeklinde sıralanabilir [35]. Pekiştirmeli öğrenme belirli bir veri kümesinin daha önceden sağlanmaması, ödül ve ceza sistemi üzerine kurulu olması ve ödülü maksimize etme amacı gütmesi gibi özellikleri yönünden hem gözetimli hem de gözetimsiz öğrenmeden farklılık göstermektedir. İnsan öğrenmesine benzer bir yaklaşıma sahiptir. Algoritmalar verilen bir çevrede nasıl davranmaları gerektiğini belirleyen bir politikayı öğrenmeye çalışırlar. Yapılan her eylem aynı zamanda çevreyi de etkilemektedir. Çevre ise verilen ödüller yardımıyla ajana yol gösterme görevini üstlenmektedir.

OpenAI'nın Dota 2'si, DeepMind'ın AlphaGo Zero'su [36] pekiştirmeli öğrenme kullanılarak oldukça başarılı sonuçlar göstermişlerdir. Bunun yanında pekiştirmeli öğrenme benzer şekilde Satranç, GTA 5, Doom, Pong, Mario, Pacman, Tetris ve birçok atari oyununda da [5] başarıyla uygulanmıştır.

Pekiştirmeli öğrenmenin temel bileşenleri ajan (agent), çevre (environment), politika (policy), ödül (R) ve değer fonksiyonu olarak sıralanabilir [34]. Şekil 2.5'te pekiştirmeli öğrenmenin temel bileşenleri gösterilmiştir.



Şekil 2.5: Pekiştirmeli öğrenmenin temel bileşenleri

Ajan, eylemleri gerçekleştiren algoritma şeklinde ifade edilebilir. Çevre, ajanın eylemlerde bulunduğu ortamı belirtir. Ödül, ajan tarafından gerçekleştirilen eylemlerin kısa vadede ne kadar iyi ya da kötü olduğunu gösterir. Belirli bir *S* durumunda gerçekleştirilen *A* eylemine bağlı olarak elde edilen ödül *R*, Denklem (2.1)'de verilmiştir.

$$R_S^A = E[R_{t+1}|S_t = S, A_t = A] \tag{2.1}$$

Değer fonksiyonu *V* ise ajanın başlangıçtan bitişe kadar toplamayı beklediği en büyük ödül miktarını belirtir. Matematiksel eşitliği Denklem (2.2)'de verilmiştir.

$$V_\pi(S) = E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = S] \tag{2.2}$$

Burada γ indirim faktörünü göstermektedir. 0 ve 1 değerleri arasında tanımlıdır. Gelecekte elde edilecek ödüllerin ne kadar önem arz edeceğini belirlemek amacıyla kullanılır. Eğer 0 olarak belirlenirse yalnızca ilk ödül dikkate alınır ve diğerlerinin etkisi olmaz. Eğer 1 olarak tanımlanırsa elde edilecek tüm ödüller eşit öneme sahip olacaktır.

Politika (π) ajanların eylemleri gerçekleştirmesinde rehber görevini üstlenen fonksiyonlardır. Belirtilen bir zamanda ajanın davranışlarını belirler. Pekiştirmeli öğrenmenin temel amacı problemin çözümünü sağlayan en uygun politikayı bulmaktır.

2.2.1 Markov Karar Süreci

Matematikte markov karar süreci ayrık zamanlı bir stokastik kontrol süreci olarak tanımlanır. Temelleri markov zincirlerine dayanmaktadır. Pekiştirmeli öğrenme algoritmaları markov karar süreci modelini temel olarak almışlardır. Ajanın belirli durumlarda en yüksek ödülü elde edeceği eylem dizilerinin belirlendiği sıralı karar verme problemlerinde kullanılır [37].

Markov karar süreci, durum, eylem ve ödüllerin sıralı gösterimi Denklem (2.3)'te ifade edilmiştir.

$$\ldots\ldots S_t, A_t, R_t, S_{t+1}, A_{t+1}, R_{t+1}, S_{t+2}, A_{t+2}, R_{t+2}\ldots\ldots \tag{2.3}$$

Markov karar sürecinin temel özelliği gelecek durum S_{t+1}'in geçmiş durumlara değil yalnızca mevcut durum olan S_t'ye bağlı olmasıdır. Bu durum Denklem (2.4)'de verilmiştir.

$$P\left[S_{t+1} \mid S_t\right] = P\left[S_{t+1} \mid S_1, \ldots, S_t\right] \quad (2.4)$$

S durumundan gelecek durum olan S' durumuna geçiş olasılığı ise Denklem (2.5)'te gösterilmiştir.

$$P_{SS'} = \mathbb{P}\left[S_{t+1} = S' \mid S_t = S\right] \quad (2.5)$$

2.2.2 Bellman Eşitliği

Bellman eşitliği Richard Bellman tarafından geliştirilen bir matematiksel optimizasyon yöntemidir [38]. Markov karar süreci problemlerinin çözümünde dinamik programlama yaklaşımı kullanarak hesaplama yükünün azaltılmasına katkıda bulunmuştur.

Ajanın farklı eylemler gerçekleştirerek en yüksek ödülü kazanmaya çalışması bir optimizasyon problemi olarak değerlendirilebilir. Bu durum Denklem (2.6)'da gösterilen eylem değer fonksiyonu ile tüm politikalar için ifade edilebilir.

$$Q_*(S, A) = \max_{\pi} Q_\pi(S, A) \quad (2.6)$$

Eylem değer fonksiyonu Denklem (2.7)'de gösterilen şekilde tekrarlanan formda da yazılabilir.

$$Q(S, A) = R_S^A + \gamma\, Q\left(S', A'\right) \quad (2.7)$$

Bellman eşitliği Denklem (2.8)'de gösterilmiştir.

$$Q(S, A) = R_S^A + \gamma \sum_{S' \epsilon S} P_{SS'}^A k \max_{A'} \left(Q_*\left(S', A'\right)\right) \quad (2.8)$$

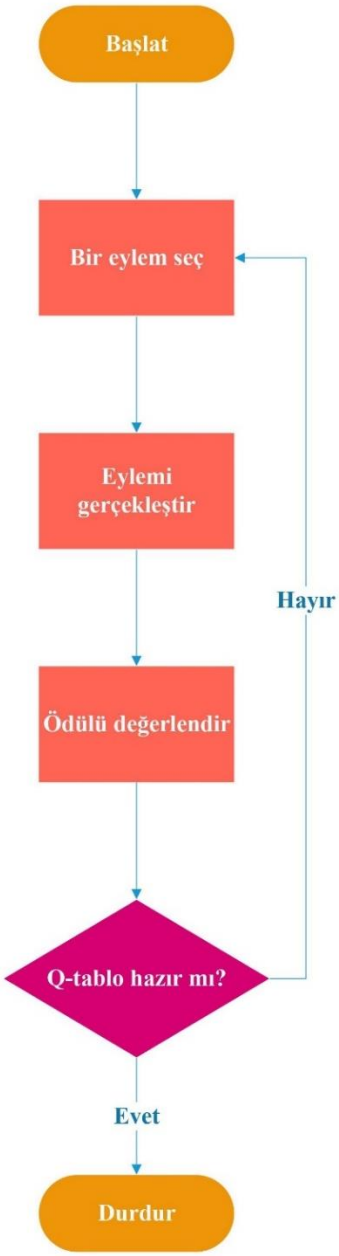
Burada R_S^A , mevcut durumda gerçekleştirilen eyleme bağlı olarak elde edilen ödülü, $P_{SS'}^A$ ise S durumundan S'durumuna geçiş olasılığını göstermektedir. R_S^A ve $P_{SS'}^A$ sırasıyla Denklem (2.9) ve Denklem (2.10)'da verilmiştir.

$$R_S^A = \mathbb{E}\left[R_{t+1} \mid S_t = S, A_t = A\right] \quad (2.9)$$

$$P_{SS'} = \mathbb{P}\left[S_{t+1} = S' \mid S_t = S, A_t = A\right] \quad (2.10)$$

2.2.3 Q Öğrenme

Q öğrenme herhangi bir çevre modeline ihtiyaç duymayan politika dışı (off policy) bir pekiştirmeli öğrenme algoritmasıdır [39]. Bellman eşitliğini temel almaktadır. Politika dışı olarak nitelenmesinin sebebi eylemleri gerçekleştirmek için kullanılan politika ve güncellenen politikanın farklı olmasıdır. Q öğrenmede temel amaç toplam ödülü maksimize eden politikayı öğrenmektir. Algoritma çalıştırıldığında Q-tablo adı verilen [eylem, durum] şeklinde bir matris oluşturulur ve başlangıç değerleri 0 olarak atanır. Şekil 2.6'da Q öğrenme algoritmasının genel işleyişi gösterilmiştir.



Şekil 2.6: Q Öğrenme algoritmasının genel işleyişi

Q-tablo başlatıldıktan sonra mevcut eylem uzayından bir eylem seçilir ve seçilen eylem uygulanır. Elde edilen ödül, miktarına göre değerlendirilir. Her iterasyonda Q değerleri güncellenir ve çözümü sağlayan en iyi politika bulunur.

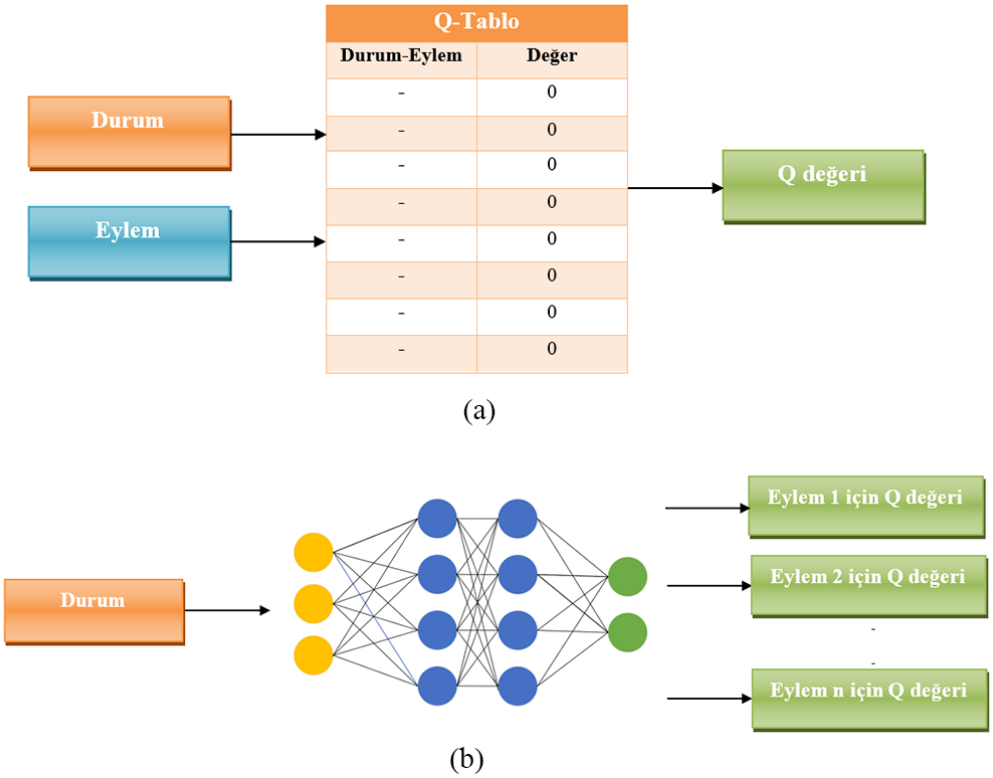
Denklem (2.11)'de Q değerlerinin güncellenme formülü gösterilmiştir.

$$Q(S,A) \leftarrow Q(S,A) + \alpha[R(S,A) + \gamma \max_{A'} Q(S',A') - Q(S,A)] \qquad (2.11)$$

Burada A ajanın gerçekleştirdiği eylemi, R elde edilen ödül miktarını, S durum ya da gözlemi, α öğrenme oranını, γ ise indirim oranını göstermektedir. İndirim oranının azalması ödüllerin değerinin zamanla azalmasına yol açmaktadır. Buna bağlı olarak daha çabuk elde edilen ödüller daha değerli hale gelmektedir.

2.2.4 Derin Q Öğrenme

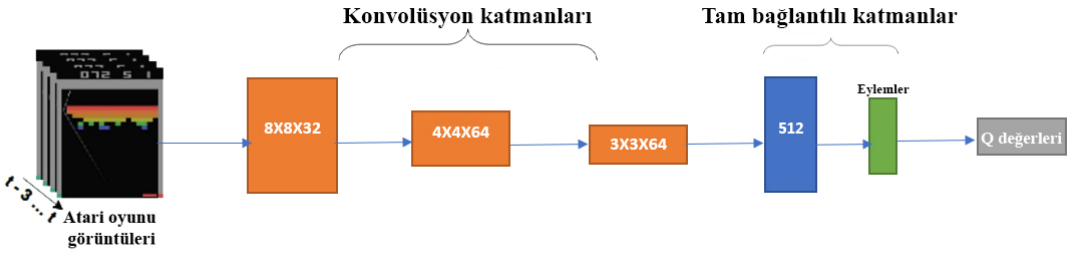
Durum/eylem çiftlerinin sayısı arttığında politika parametrelerini tabloda ifade etmek zorlaşmaktadır. Bu nedenle karmaşık ve yüksek ölçekli problemlerin çözümünde Q-tabloların kullanılması uygun değildir. Derin Q öğrenme ya da Derin Q ağlar, derin sinir ağları ve pekiştirmeli öğrenmenin birlikte kullanımı ile oluşturulmuştur. Şekil 2.7'de Q öğrenme ve derin Q öğrenmenin karşılaştırılması gösterilmiştir.



Şekil 2.7: (a) Q öğrenme ve (b) derin Q öğrenmenin karşılaştırılması

Uygun Q değerlerinin elde edilmesi için YSA'dan faydalanılmaktadır. Ayrıca deneyim tekrarı (experience replay) kullanılarak ajanın deneyimleri depolanmaktadır. Depolanan deneyimlerden tekdüze örnekler alınarak ajanın deneyimler üzerinden öğrenme gerçekleştirmesi beklenmektedir. Yapılan çalışmalarda deneyim tekrarının kararlılık sağlamada başarılı olduğu belirtilmiştir [40].

Giriş bölümünde de bahsedildiği üzere derin Q ağlar, Mnih ve diğ. [6] tarafından yapılan çalışma ile ön plana çıkmıştır. Çalışmalarında ham görüntüler kullanarak 49 adet Atari 2600 oyununu derin Q ağlar kullanarak test etmişlerdir. Durum olarak kullanılan oyun görüntüleri 80×80 boyutunda ve dört tanedir. Görüntülerin bir tanesi mevcut durum diğer üç tanesi ise önceki görüntülerdir. Çalışmanın farklı yönü evrişimsel sinir ağlarının kullanılmasıdır. Şekil 2.8'de çalışmada kullanılan evrişimsel sinir ağı modeli gösterilmiştir.



Şekil 2.8: Çalışmada [6] kullanılan derin Q ağ yapısı

Bunun dışında yukarıda bahsedilen deneyim tekrarı da çalışmanın ortaya koyduğu önemli yeniliklerdendir. Belirli bir iterasyon süresince deneyimler veri tabanına kaydedilmiştir. Deneyimler ve veri tabanı sırasıyla Denklem (2.12) ve Denklem (2.13)'te verilmiştir. Daha sonra ise eğitim aşamasında bu deneyimlerden tekdüze örnekler alınarak öğrenme gerçekleştirilmiştir. Çalışmada atari oyunlarının oynanması sürecinde %75 oranında başarı sağlandığı belirtilmiştir.

$$E_t = (S_t, A_t, R_t, S_{t+1}) \tag{2.12}$$

Burada E_t, t anında kaydedilen deneyimi, S_t durum bilgisini, A_t gerçekleştirilen eylemi, R_t ödülü, S_{t+1} ise $t+1$ anındaki durumu göstermektedir.

$$D_t = E_1, \ldots, E_t \tag{2.13}$$

Çalışmada dikkat çeken diğer bir nokta birden fazla yapay sinir ağı kullanılmasıdır. Bunun nedeni derin Q ağların gözetimli öğrenmeden farklı olarak veriler için etiket kullanmamasıdır. Her bir iterasyonda hedef sürekli değişmektedir. Bu nedenle Q ağ

ve hedef ağ olmak üzere iki farklı ağ kullanılmaktadır. Sistemin kararlılığını sağlamak için belirli bir iterasyonda Q ağ parametreleri (θ) hedef ağa kopyalanır. Farklı ağlar kullanılmaması durumunda Q öğrenme formülüne göre sürekli değişen bir hedef yakalanmaya çalışılacak ve sonuç olarak yakınsama asla gerçekleşmeyecektir.

2.2.5 Çift Derin Q Öğrenme

Derin Q ağların problemlerinden biri bazı eylemlerin olduğundan daha değerli kabul edilmesi ve bunun sonucunda belirli durumda (S) aynı eylemlerin sürekli seçilmesidir. Bu da aşırı öğrenme (overfitting) denilen duruma yol açmaktadır. Sonuç olarak en iyi politikanın öğrenilmesi olumsuz etkilenmektedir. Hasselt ve diğ. [41] yaptıkları çalışmada bu durumun önüne geçmek için çift derin Q ağları önermişlerdir. Derin Q ağlar ile ilgili yapılan çalışmaya [6] benzer şekilde önerdikleri modelin performansını atari oyunları üzerinden test etmişler ve performansta artış sağladıklarını belirtmişlerdir.

Derin Q ağlarda eylem değerlerinin hem değerlendirilmesi hem de seçilmesi aynı şekilde gerçekleştirmektedir (Denklem (2.14)).

$$\max_{\mathrm{A}} \mathrm{Q}(\mathrm{S}_{\mathrm{t+1}}, \mathrm{A}) \tag{2.14}$$

Çift derin Q ağlarda ise eylemlerin seçilmesi ve eylem değerlerinin değerlendirilmesi süreçleri birbirinden ayrı şekilde gerçekleştirilmektedir (Denklem (2.15)).

$$\mathrm{Q}(\mathrm{S}', \operatorname{argmax}_{\mathrm{A}} \mathrm{Q}(\mathrm{S}, \mathrm{A}; \theta_{\mathrm{i}}), \theta_{\mathrm{i}}) \tag{2.15}$$

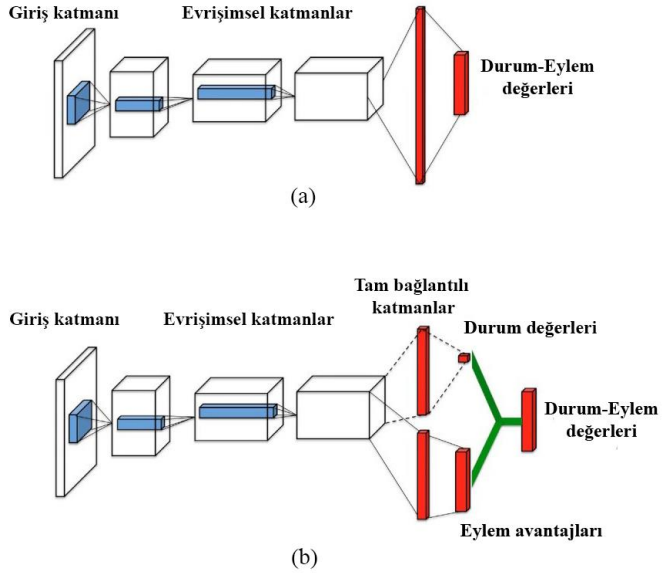
Burada θ_i değeri ağ için ağırlık değerlerini göstermektedir.

Her bir durum eylem çifti için tek bir Q değeri yerine $Q(A)$ ve $Q(B)$ şeklinde iki Q değeri kullanılmıştır. Öncelikle gelecek durum (S') için en yüksek Q değerini sağlayan eylem (A) bulunur. Daha sonra ise bu eylem (A) ikinci Q değerini $Q(B)$ bulmak için kullanılır. Son olarak ise $Q(B)$ değeri $Q(A)$ değerini güncellemek için kullanılır. Çift derin Q ağ formülü aşağıda gösterilmiştir (Denklem 2.16).

$$\mathrm{Q}^{\mathrm{A}}(\mathrm{S}, \mathrm{A}) \leftarrow \mathrm{Q}^{\mathrm{A}}(\mathrm{S}, \mathrm{A}) + \alpha[\mathrm{R} + \gamma \mathrm{Q}^{\mathrm{B}}(\mathrm{S}', \mathrm{A}) - \mathrm{Q}^{\mathrm{A}}(\mathrm{S}, \mathrm{A})] \tag{2.16}$$

2.2.6 Düello Derin Q Öğrenme

Düello derin Q öğrenme Wang ve diğ. [42] tarafından tanıtılmıştır. Derin Q öğrenme ile aynı ağ yapısını kullanır. Şekil 2.9'da derin Q öğrenme ve düello derin Q öğrenmenin karşılaştırılması gösterilmiştir.



Şekil 2.9: (a) Derin Q ağlar ve (b) düello derin Q öğrenmenin karşılaştırılması [41]

Farklılık tam bağlantılı katmanda görülmektedir. Bu katman iki bölüme ayrılmaktadır. Birinci bölümde durumun (s) değer tahmini (V) yapılır ve $V(s)$ şeklinde ifade edilir. İkinci bölümde ise belirli bir durumda (s) yapılan eylemin (a) avantaj tahmini yapılır ve $A(s,a)$ şeklinde gösterilir. Böylece bazı gereksiz eylemlerden sakınılarak performans artışı sağlanması hedeflenmiştir. Q değerleri aşağıdaki şekilde elde edilmektedir (Denklem 2.17).

$$\mathrm{Q}^{\pi}(\mathrm{s},\mathrm{a}) = \mathrm{V}^{\pi}(\mathrm{s}) + \mathrm{A}^{\pi}(\mathrm{s},\mathrm{a}) \tag{2.17}$$

2.2.7 Düello Çift Derin Q Öğrenme (D3QN)

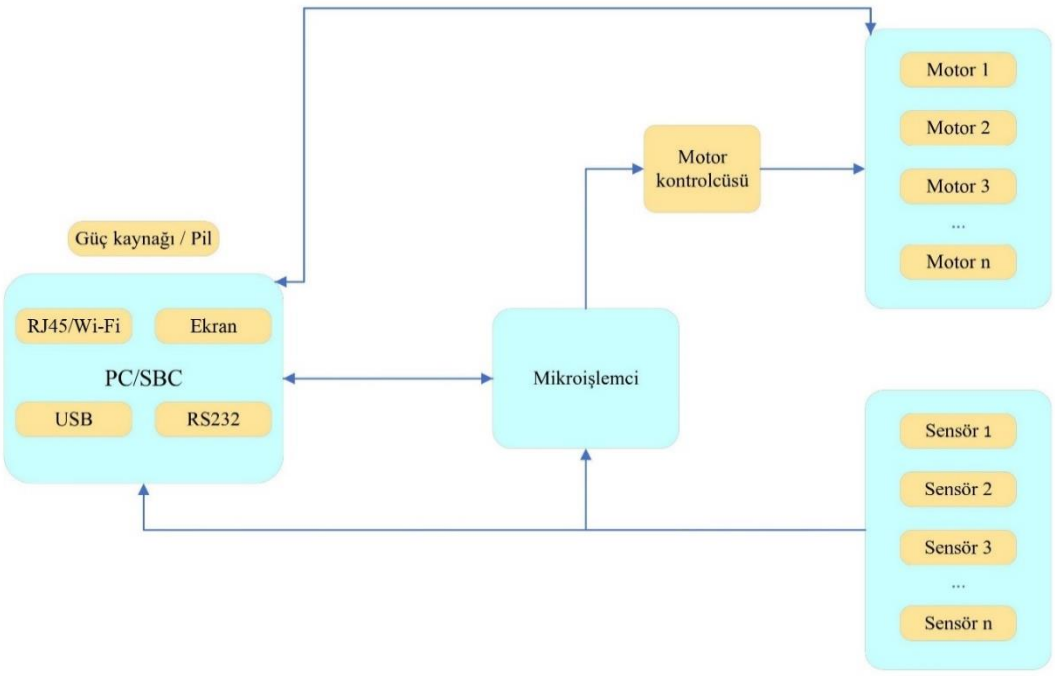
D3QN algoritması Wang ve diğ. [42] tarafından tanıtılmış, Xie ve diğ. [43] tarafından ise ilk kez robotların engellerden kaçınması probleminde kullanılmıştır. Algoritmanın yapısı çift derin Q ağ ve düello derin Q ağ algoritmalarının birlikte kullanılması üzerine kuruludur. Yapılan çalışmada [43] robotların derinlik görüntüleri kullanarak başarılı şekilde engellerden kaçınmaları amaçlanmıştır. Bu amaçla dört adet sıralı derinlik görüntüsü kullanılmıştır. Böylelikle robotun mevcut konumu ve hızı tespit

edilerek engellerden kaçınması sağlanmıştır. İki farklı doğrusal hız v (0.2, 0.4 m/s) ve beş farklı açısal hız $\omega(\frac{\pi}{6}, \frac{\pi}{12}, 0, -\frac{\pi}{12}, -\frac{\pi}{6}$ rad/s) kullanılarak yedi farklı eylem tanımlanmıştır. Robot engellere çarpması sonucunda -10 puanlık bir ceza almaktadır. Yapılan testler sonucunda D3QN algoritmasının standart derin Q öğrenme, çift derin Q öğrenme ve düello derin Q öğrenme algoritmalarına göre daha başarılı sonuçlar verdiği belirtilmiştir.

2.3 Robot İşletim Sistemi (ROS)

2.3.1 Robot ve Robot Programlama Nedir?

ROS'a giriş yapmadan, öncelikle robot ve robot programlama kavramlarından bahsedilmesi gerekmektedir. Robot, motorlardan ve sensörlerden oluşan ve bir hesaplama birimi tarafından kontrol edilen bir makine olarak tanımlanabilir. Hesaplama birimi bir mikroişlemciden ya da bilgisayardan oluşabilir. Robot programlama ise hesaplama birimi için gerekli olan programların yazılmasıdır. C/C++, Python ve Java gibi programlama dilleri bu amaçla kullanılmaktadır. Şekil 2.10'da temel bir robot diyagramı gösterilmiştir.



Şekil 2.10: Temel bir robot diyagramı [44]

Programlanabilir mikroişlemci bir kişisel bilgisayar (PC) ya da tek kartlı bilgisayar (SBC) kullanılarak programlanır. Mikroişlemci robotun ana bileşenlerini oluşturan

motorlar ve sensörlere komut vererek robotun hareketlerini gerçekleştirir. Motorlar robotun işlevine göre servo veya DC motorlardan seçilir. Sensörler ise robotun çevre ile etkileşimini sağlamaktadır. Sensörlere, ultrasonik sensörler, kızılötesi sensörler, yüksek çözünürlüklü kameralar ve lazer tarayıcılar örnek verilebilir. Sensörlerin ve motorların bazıları doğrudan bilgisayarlara bağlanabilirken bazıları yalnızca mikroişlemci üzerinden bağlanabilmektedir.

Bir robotun programlanabilmesi için kullanılacak olan programlama dilinin bazı özelliklere sahip olması gerekmektedir. Bunlar aşağıdaki şekilde sıralanabilir [44-46].

- Çoklu kullanım (Multithreading) desteği: Bir robot farklı sayıda motor ve sensörden oluşmaktadır. Bu nedenle kullanılacak olan programlama dilinin farklı görevleri gerçekleştirebilecek ve her bir görevin birbirleriyle iletişimini sağlayacak özellikte olması gerekmektedir.

- Nesne tabanlı olma: Nesne tabanlı programlama dilleri modüler yapılarıyla hem bakım hem de tekrar kullanılabilirlik açısından diğer dillere göre avantaj sağlamaktadırlar. Bu özellikleri robot programlama alanında tercih edilmelerinin nedenlerindendir.

- Düşük seviye aygıt kontrolü: Robot programlamada kullanılacak olan programlama dillerinden beklenen diğer bir özellik seri port, paralel port, usb ve giriş çıkış pinleri gibi düşük seviye aygıtlara erişim sağlayabilmeleridir.

- Geliştirme kolaylığı: Programlama dilinin hızlı şekilde yeni algoritmalar geliştirmeye imkân vermesidir. Python kullanım kolaylığı ve hızlı geliştirmeye imkân vermesi nedeniyle sıklıkla tercih edilmektedir.

- İşlemler arası iletişim kolaylığı: Bir robotun görevlerini gerçekleştirmesi için farklı sensörlerden gelen verileri ve motorları kullanması gerekmektedir. Bu nedenle farklı işlemlerin aynı anda gerçekleştirilmesi robotun doğru bir şekilde çalışması için önemlidir. Örneğin kamera üzerinden elde edilen görüntü ile farklı bir işlem yapılırken kızılötesi sensörden gelen veriler farklı bir görev için kullanılabilir. Daha önce bahsedildiği üzere bu işlemlerin aynı anda yapılabilmesi için çoklu kullanım mimarisi kullanılabilir. Diğer bir seçenek ise her bir işlem için ayrı bir program yazılması ve bu programların paralel bir şekilde çalıştırılmasıdır. Bu şekilde programlar birbirleriyle iletişim kurabilir ve veri alışverişinde bulunabilir.

- Performans: Bir robot çalışırken bazen yüksek çözünürlüklü derinlik görüntüleri gibi verilerle işlem yapılması gerekebilmektedir. Bu nedenle kullanılacak olan programlama dilinin hesaplama kaynaklarının tüketiminde başarılı olması gerekmektedir.

- Topluluk desteği: Bir programlama dilinin yalnızca performans bakımından tatmin edici olması yeterli değildir. Kullanıcıların karşılaştıkları problemlerin çözümünde ihtiyaç duydukları yardımı sağlayabilecekleri bir topluluk bulunmalıdır. Bu hem forumlar ve bloglardaki kullanıcı desteği hem de iyi bir şekilde hazırlanmış dokümanlar yardımıyla sağlanabilmektedir.

- Üçüncü parti yazılımların kullanılabilmesi: Üçüncü parti yazılım desteği yapılan işlemlerin daha kolay ve hızlı bir şekilde gerçekleştirilmesine imkân sağlamaktadır. Örneğin kamera görüntülerinin işlenmesi gerektiğinde OpenCV kütüphanesinin kullanılabilmesi işleri oldukça kolaylaştıracaktır.

- Robotik yazılım iskeleti desteği: Robot programlama işlemlerini kolaylaştırmak için geliştirilmiş olan ROS gibi yazılım iskeletleri bulunmaktadır. Bu nedenle kullanılacak olan programlama dilinin bu gibi yazılım iskeletleri ile uyumlu olması geliştirme sürecini kolaylaştıracak ve hızlandıracaktır.

Yukarıda sayılan özellikler dikkate alındığında robot programlamada en çok kullanılan programlama dillerinin C/C++, Python, Java ve C# olduğu söylenebilir. Bu tez kapsamında gerekli olan yazılımların tümü Python programlama dili kullanılarak geliştirilmiştir.

2.3.2 ROS Nedir?

ROS hem ticari olarak hem de araştırma faaliyetlerinde kullanılan açık kaynak kodlu bir robotik yazılım iskeletidir**.** ROS'un genel özellikleri aşağıdaki şekilde sıralanabilir [44-46];

- Farklı işlemler arası kolay mesaj aktarımı: ROS robot programlama sürecinde en önemli özelliklerden biri olan farklı işlemlerin aynı anda yapılmasına ve çalışan programların birbirleriyle veri alışverişine imkan sağlamaktadır. Örneğin motorların hareket edeceği yönü belirlemede kameradan gelen verilerin kullanılması bu kapsamdadır.

- İşletim sistemi benzeri özellikler: ROS gerçek bir işletim olmasa da bazı özellikleri yönünden benzer işlevlere sahiptir. Bunlar düşük seviye aygıt kontrolü, çoklu kullanım, paket yönetimi ve donanım soyutlaştırma olarak sıralanabilir. Donanım soyutlaştırma, geliştiricinin donanım kaynaklarına erişerek aygıttan bağımsız ve yüksek performanslı uygulamalar oluşturması olarak ifade edilebilir. Örneğin bir sensör için yazılan kodun yeni bir sensör kullanıldığında tekrar yazılması gerekmez. Paket yönetimi ise verilerin paketler halinde organize edilmesi ve farklı bilgisayarlara yüklenmesidir. Bir paket kaynak, ayar ve veri dosyalarını içerebilir.

- Yüksek seviyeli programlama dili desteği: ROS robot programlamada kullanılan pek çok programlama dilini desteklemektedir. Farklı diller için sağladığı istemci kütüphaneleri sayesinde geliştiriciler ROS özelliklerini kendi uygulamalarında rahatlıkla kullanabilirler. Örneğin geliştirilen bir android uygulamasında ROS özellikleri entegre edilmek istendiğinde "rosjava" istemci kütüphanesi kullanılabilir.

- Görselleştireme araçları ve simülatörler: ROS paketleri içerisinde robot uygulamaların görselleştirilmesi için faydalı araçlar bulunmaktadır. "Rviz" bunlardan biridir. Kamera ve lazer tarayıcıdan gelen verilerin görselleştirilmesi için kullanılmaktadır. Bunun yanında Gazebo gibi simülasyon programları da gerçek dünyaya benzer ortam sunarak geliştirme sürecini hızlandırmakta ve kolaylaştırmaktadır.

- Üçüncü parti kütüphanelerle uyumluluk: ROS sıklıkla kullanılan kütüphanelerle yüksek düzeyde uyumluluğa sahiptir. Örneğin görüntü işlemede kullanılan OpenCV kütüphanesi ROS uygulamalarında kullanılabilir. Böylece robotun üzerinde bulunan kameradan elde edilen veriler kolaylıkla işlenebilmekte ve görevlerin gerçekleştirilmesinde kullanılabilmektedir.

- Bilinen bazı algoritmaların kolaylıkla uygulanması: ROS üzerinde bilinen bazı algoritmaların uygulamaları hazır olarak bulunmaktadır. Böylelikle istenilen robotun prototipinin oluşturulma süreci kısalmaktadır. Bu algoritmalara örnek olarak SLAM, A* ve Dijkstra algoritmaları verilebilir.

- Prototip oluşturma kolaylığı: Hazır algoritmalar gibi bazı hazır robot paketleri de ROS'ta bulunmaktadır. Bu robotlar ihtiyaca göre düzenlenerek geliştirme süreci hızlandırılabilir.
- Topluluk desteği: ROS'un popüler olmasının sebeplerinden biri de topluluk desteğinin üst düzeyde olmasıdır. Dünyanın her yanından ROS geliştiriciler paketlerin güncellenmesine ve geliştirilmesine destek olmaktadırlar.

2.3.3 ROS Tarihçesi

ROS'un ortaya çıkışından itibaren gelişim süreci aşağıdaki şekilde özetlenebilir [47].

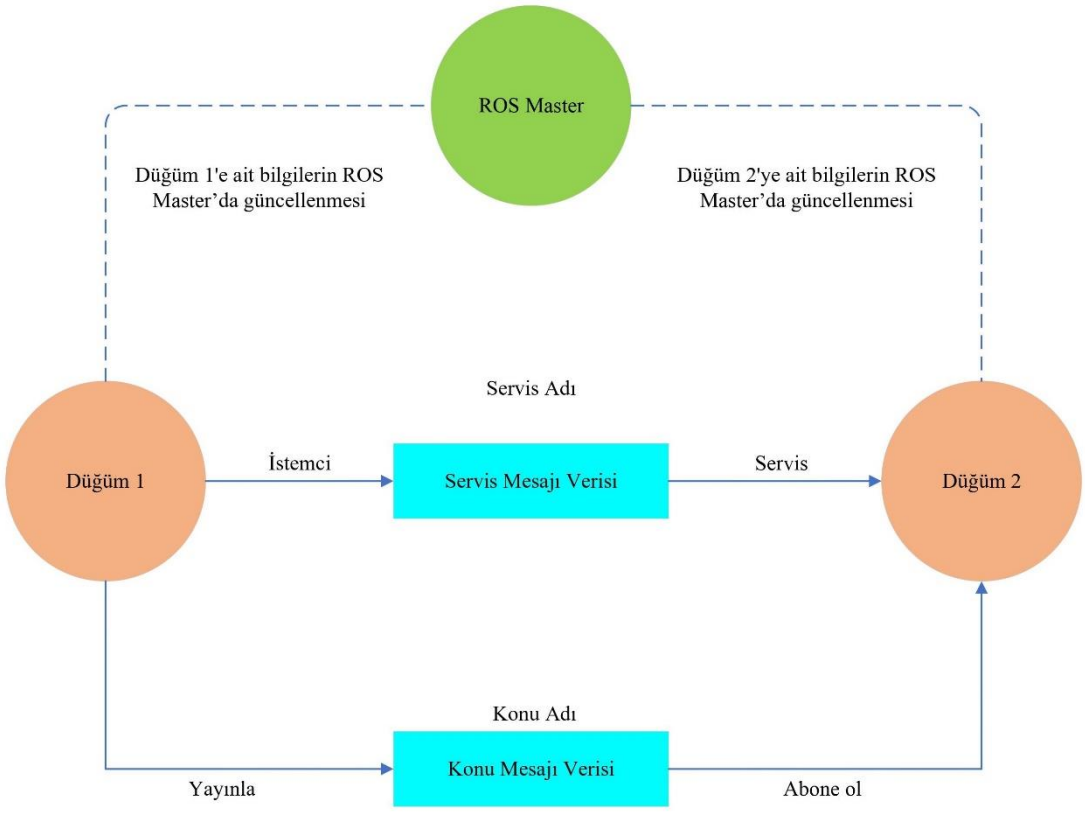
- ROS projesi 2007 yılında Stanford Üniversitesinde Morgan Quigley tarafından geliştirilmeye başlandı.
- 2007'nin sonlarına doğru Willow Garage adlı şirket projeyi üstlendi ve ROS adını verdi.
- 2009 yılında ROS 0.4 yayınlandı PR2 adlı robot geliştirildi.
- 2010 yılında ROS 1.0 ve ROS C Turtle yayınlandı.
- 2011 yılında ROS Diamondback ve ROS Electric Emys yayınlandı.
- 2012 yılında ROS Fuerte ve ROS Groovy Galapagos yayınlandı.
- 2012 yılında OSRF (Open Source Robotics Foundation) ROS projesini devraldı.
- 2013 yılında ROS Hydro Medusa yayınlandı.
- 2014 yılında ROS Indigo Igloo yayınlandı.
- 2015 yılında ROS Jade Turtle yayınlandı.
- 2016 yılında ROS Kinetic Kame yayınlandı.
- 2017 yılında ROS Lunar Loggerhead yayınlandı.
- 2018 yılında ROS Melodic Morenia yayınlandı.
- 2019 Noetic Ninjemys yayınlandı. Bu sürüm son ROS 1 sürümüdür.
- 2017 yılından itibaren ROS 1 ile eş zamanlı olarak pek çok yeni özelliğe sahip ROS 2 geliştirilmeye başlanmıştır. Güncel ROS 2 sürümü 2021 yılında yayınlanan Galactic Geochelone sürümüdür.

2.3.4 ROS Kullanmanın Avantajları

ROS kullanılmadan da robotik uygulamalar geliştirmek mümkündür; ancak bu durumda geliştiriciler her robot için kendi programlarını tekrar yazmak zorundadırlar. Bu da oldukça zaman kaybına yol açmaktadır. ROS ile bu sorun ortadan kalkmış ve tüm robotik uygulamalar için ortak bir platform oluşturulmuştur. Güncel algoritmaların hazır şekilde ROS ile gelmesi geliştirme sürecini kısaltmıştır. Bunun yanında geniş topluluk desteği ve açık kaynak yapısı tercih edilme sıklığını arttırmaktadır.

2.3.5 ROS Mimarisi

ROS temelde iki veya daha fazla programın haberleşmesini sağlayan bir yazılım iskeletidir. ROS kullanmadan da robotik uygulamalar geliştirmek mümkünse de kullanılan sensörler ve motorların sayısı arttıkça yazılım karmaşık hale gelmektedir. ROS kullanmanın getirdiği kolaylık bu kısımda kendini göstermektedir. Farklı programlar ROS üzerinden haberleşerek veri alışverişinde bulunmakta ve bu yolla uygulamanın geliştirilme süreci kolaylaşmaktadır. Şekil 2.11'de temel ROS mimarisi gösterilmiştir.



Şekil 2.11: Temel ROS mimarisi

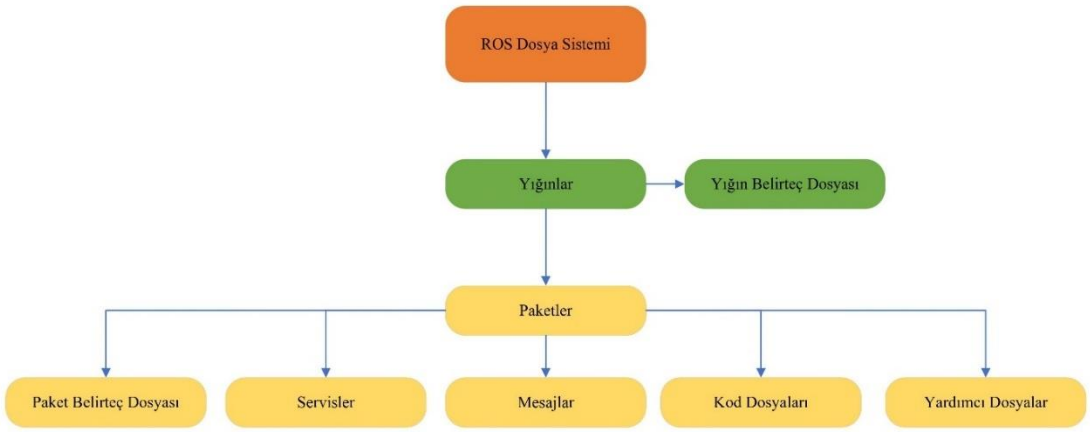
Burada Düğüm 1 ve Düğüm 2 ayrı ayrı çalışan iki programı göstermektedir. Örneğin farklı iki sensörden gelen veriler bu düğümler üzerinde işlenebilir. Hem Düğüm 1 hem de Düğüm 2'den veriler ROS Master'da bulunmaktadır. Verileri gönderen düğümlere yayıncı düğümler bu verileri kullanan düğümlere ise abone düğümler adı verilmektedir. Şekilde Düğüm 1'den gönderilen veriler ROS Master üzerinden Düğüm 2'ye gönderilmekte ve bu şekilde düğümlerin haberleşmesi sağlanmaktadır. Gönderilen tüm veriler ROS mesajları olarak ifade edilirler. ROS mesajları string, integer, float gibi ilkel veri tiplerinden oluşabileceği gibi tanımlanan farklı tiplerden de oluşabilir. Mesajların gönderildikleri yola ise ROS konuları adı verilmektedir. Her bir konu farklı bir adla tanımlanır.

2.3.6 ROS Kavramları

- ROS düğüm: ROS üzerinden haberleşen programlardır. Basit şekilde herhangi bir sensör üzerinden verileri alan veya bunları işleyen yapılar düğüm olarak tanımlanabilir. ROS master üzerinden birbirleriyle haberleşirler. Verilerin dağıtımını yapan düğümlere yayıncı düğüm, bu verileri kullanan düğümlere ise abone düğüm adı verilmektedir.
- ROS master: ROS düğümlerinin haberleşmesini sağlayan temel yapıdır. Tüm düğümlere ait verileri üzerinde tutar.
- ROS parametre sunucusu: ROS master ile çalışır. Tüm düğümlere ait parametre verilerini üzerinde tutar ve düğümlerin erişmesine imkân sağlar. "Public" olarak tanımlanan parametrelere tüm düğümler erişebilirken "Private" olarak tanımlanan parametrelere yalnızca izin verilen düğümler erişebilir.
- ROS konu: ROS mesajlarının gönderildikleri yollar olarak tanımlanabilir. Bir düğüm farklı sayıda konu yayınlayabilir ya da farklı konulara abone olabilir.
- ROS mesaj: Düğümlerin gönderdikleri verilerdir. ROS konuları aracılığıyla gönderilirler. İlkel veri tiplerinde olabilecekleri gibi farklı tiplerde de olabilirler.
- ROS servis: ROS servis ile bir düğüm diğer bir düğümden istekte bulunabilir. Servis çağrısı yapan düğüme istemci düğüm çağrıya cevap veren düğüme ise sunucu düğüm adı verilmektedir.

2.3.7 ROS Dosya Yapısı

ROS dosya sistemi temelde paketler ve yığınlardan meydana gelmiştir. Paketler servisler, mesajlar, kod dosyaları vb. yapıları içermektedir. Paketlerin bir araya gelerek oluşturduğu yapılara ise yığınlar adı verilmektedir. Şekil 2.12'de ROS dosya sistemi gösterilmiştir.

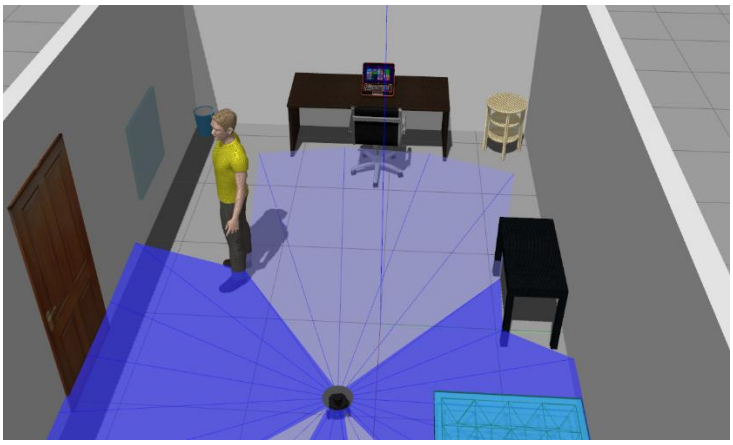


Şekil 2.12: ROS dosya sistemi

2.3.8 Gazebo Benzetim Ortamı

Gazebo açık kaynak olarak kullanıma sunulan 3-boyutlu bir robot benzetim ortamıdır. 2004 yılında robotik alanında açık kaynak yazılımlar sağlamak üzerine geliştirilen "Player Project" çatısı altında bir bileşen olarak üretilmiştir. 2011 yılında ise Willow Garage tarafından desteklenen bağımsız bir proje haline gelmiştir. 2018 yılından itibaren ise ticari amaç gütmeyen bir araştırma laboratuvarı olan "Open Robotics" tarafından desteklenmektedir [44].

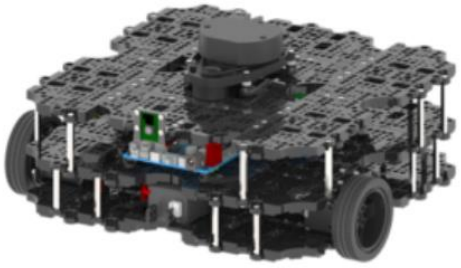
Gazebo çeşitli algoritmaları hızlı bir şekilde test etmek, robot tasarımları yapmak ve yapay zekâ uygulamaları geliştirmek için kullanılmaktadır. Robotik uygulamalarını gerçek dünya ortamında test etmek maliyet vb. sebeplerle çoğu zaman mümkün olamamaktadır. Bu gibi durumlarda Gazebo benzetim ortamı ile gerçekçi iç ve dış mekân tasarımları yapılabilmektedir. Bu ortamlarda robotların hızlı ve etkili bir şekilde test edilmesine imkân sağlanmaktadır. ROS ile geliştirilen robotik uygulamalarda yüksek uyumluluğu nedeniyle benzetim ortamı olarak çoğunlukla Gazebo tercih edilmektedir. Şekil 2.13'te Gazebo benzetim ortamı ile oluşturulmuş örnek bir iç mekân tasarımı gösterilmiştir.



Şekil 2.13: Gazebo ile oluşturulmuş bir iç mekân tasarımı

2.3.9 TurtleBot

TurtleBot, ROS ortamında kullanılan standart robot modelidir. Temeli 1967 yılında eğitsel amaçlarla geliştirilmiş bir programlama dili olan Logo'nun üzerinde çalıştırıldığı Turtle robota dayanmaktadır. Turtlebot günümüzde eğitim, araştırma, prototip oluşturma ve hobi amaçlarıyla kullanılmaktadır. Toplam üç farklı modeli vardır. TurtleBot 1 modeli 2010 yılında Willow Garage çalışanları tarafından geliştirilmiştir. 2012 yılında ise Yujin Robot şirketi tarafından TurtleBot 2 modeli tanıtılmıştır. Son olarak 2017 yılında kullanıcılardan gelen istekler doğrultusunda yeni özelliklerle donatılmış TurtleBot 3 modeli kullanılmaya başlanmıştır. TurtleBot'un tüm modelleri SLAM ve navigasyon algoritmalarının test edilmesi amacıyla ROS ortamında sıklıkla kullanılmaktadır. Şekil 2.14'te TurtleBot 3 modellerinden biri olan Waffle Pi modeli, Çizelge 2.1'de ise Waffle Pi modeline ait standart donanım özellikleri gösterilmiştir.



Şekil 2.14: Turtlebot3 Waffle Pi modeli

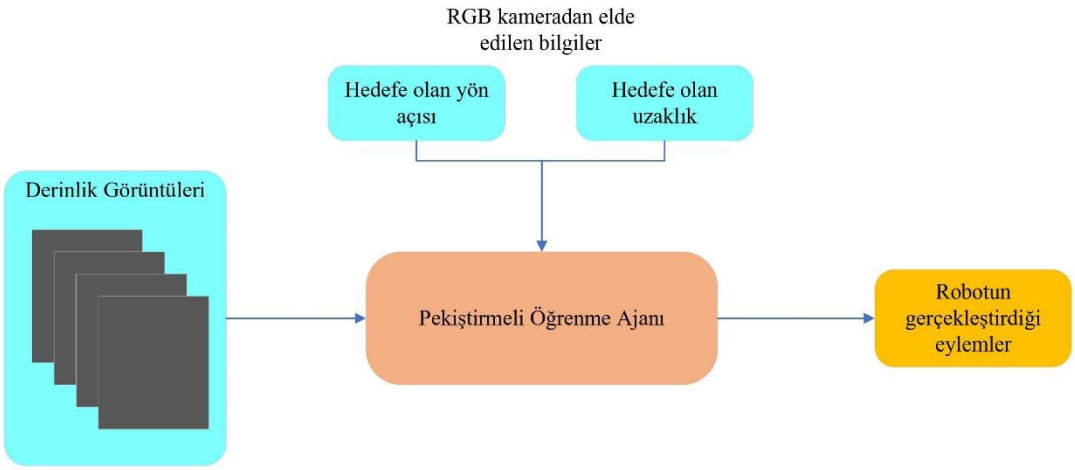
Çizelge 2.1: TurtleBot 3 Waffle Pi modeline ait standart donanım özellikleri

Donanım Adı	Özelliği
Maksimum doğrusal hız	0.26 m/s
Maksimum dönüş hızı	1.82 rad/s (104.27 deg/s)
Maksimum taşınabilir yük	30kg
Boyut	281mm x 306mm x 141mm
Ağırlık	1.8kg
SBC (Single Board Computers)	Raspberry Pi
MCU (Micro Controller Unit)	32-bit ARM Cortex®-M7 FPU (216 MHz, 462 DMIPS)
Pil	Lityum Polimer 11.1V 1800mAh / 19.98Wh 5C
PC bağlantısı	USB
LDS (Lazer uzaklık sensörü)	360 Laser Uzaklık Sensörü LDS-01
Kamera	Raspberry Pi Kamera Modülü v2.1
IMU	Jiroskop 3 Eksen, İvmeölçer 3 Eksen

3. MATERYAL VE METOT

Bu bölümde öncelikle tasarlanan sistemin genel yapısı gösterilmiştir. Daha sonra sistemin genel yapısını oluşturan derinlik görüntüleri, D3QN yapısı, durum uzayı, ödül-ceza sistemi, hedefe olan uzaklık ve yön bilgisinin elde edilme yöntemleri açıklanmıştır. Son olarak ise gezgin robotun görevleri gerçekleştireceği çevre tasarımları tanıtılmıştır.

Sistemin temelini karar verici durumunda olan pekiştirmeli öğrenme ajanı oluşturmaktadır. Pekiştirmeli öğrenme ajanı ise D3QN algoritmasından oluşmaktadır. Ağın kullandığı veriler, derinlik kamerası tarafından sağlanan derinlik görüntüleri, hedefe olan uzaklık ve yön bilgisi şeklindedir. Ajanın karar verdiği eylemler ise robotun farklı yönlerde aldığı hız değerleridir. Şekil 3.1'de sistemin genel yapısı gösterilmiştir.



Şekil 3.1: Sistemin genel yapısı

3.1 Durum Uzayı

Ajanın gelecek yüksek değerli ödülleri alabilmesi için kullanması gereken bazı bilgiler bulunmaktadır. Durum uzayı adı verilen bu bilgiler ajanın gelecek eylemlerine karar vermesi için gereklidir. Her bir adımda ajan mevcut durum bilgisini kullanarak en yüksek ödülü almaya yönelik eylemlerini gerçekleştirir. Tasarlanan sistemde durum bilgisi olarak derinlik kamerasından gelen mevcut görüntü ve önceki üç görüntünün yanında hedefe olan uzaklık ve yön bilgileri de kullanılmıştır.

3.1.1 Derinlik Görüntüleri

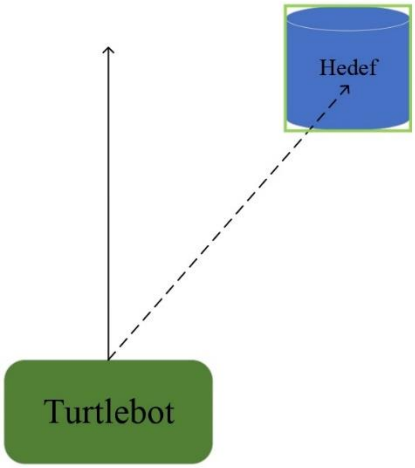
Derinlik görüntüleri robotların çevreleri hakkında 3-boyutlu geometrik bilgi edinmelerini sağlayan önemli araçlardandır. Standart RGB kameralar ile doğrudan bu bilgi elde edilememektedir; ancak RGB-D adıyla da bilinen derinlik kameraları ile doğrudan derinlik görüntüleri alınabilmektedir. LIDAR olarak tanımlanan 2-boyutlu lazer tarayıcılarla ise 3-boyutlu görünüm elde edilememektedir. Bu nedenle tasarlanan sistemde çevreye ait görüntülerin elde edilmesi için RGB-D kameradan elde edilen derinlik görüntüleri kullanılmıştır. Kullanılacak olan görüntü sayısının belirlenmesi aşamasında Mnih ve diğ. [6] tarafından yapılan çalışmadaki yaklaşıma benzer bir yaklaşım kullanılarak mevcut derinlik görüntüsünün yanında önceki üç derinlik görüntüsü de kullanılmıştır. Şekil 3.2'de derinlik kamerası ile elde edilmiş 160×128 boyutunda dört adet derinlik görüntüsü gösterilmiştir.

Şekil 3.2: Derinlik kamerası ile sırayla çekilmiş olan 160×128 boyutunda derinlik görüntüleri

Bu şekilde görüntülerin arasındaki farklar kıyaslanarak dinamik engellerin hızı konusunda bilgi edinilebilmektedir. Bu yaklaşım insanların hareketleri sırasında çevrelerindeki engellerden sakınmalarına benzer bir yaklaşımdır. İnsanlar da çevrelerindeki nesnelerin hızlarını tam olarak bilmese de tahmin edebilirler ve hareketlerini buna göre belirlerler.

3.1.2 Hedefe Yönelme Açısı

Derinlik görüntüleri engellerden kaçınma işlemi sırasında ajana gerekli bilgileri verirken hedefe ulaşmak için bazı ek bilgilere ihtiyaç duyulmaktadır. Bu bilgilerden ilki hedefe olan açının yani yön bilgisinin elde edilmesidir. Yön bilgisi tek bir RGB kamera ve OpenCV kütüphanesi kullanılarak elde edilebilmektedir. Daha sonra ajanın eğitimi sırasında ve ödül-ceza sisteminin tanımlanmasında hedefe olan açı değeri kullanılmıştır. Şekil 3.3'te Turtlebot'un hedefe olan yönelmesi ve aralarındaki θ açısı gösterilmektedir.



Şekil 3.3: Turtlebot'un hedefe olan yönelmesi ve aralarındaki θ açısı

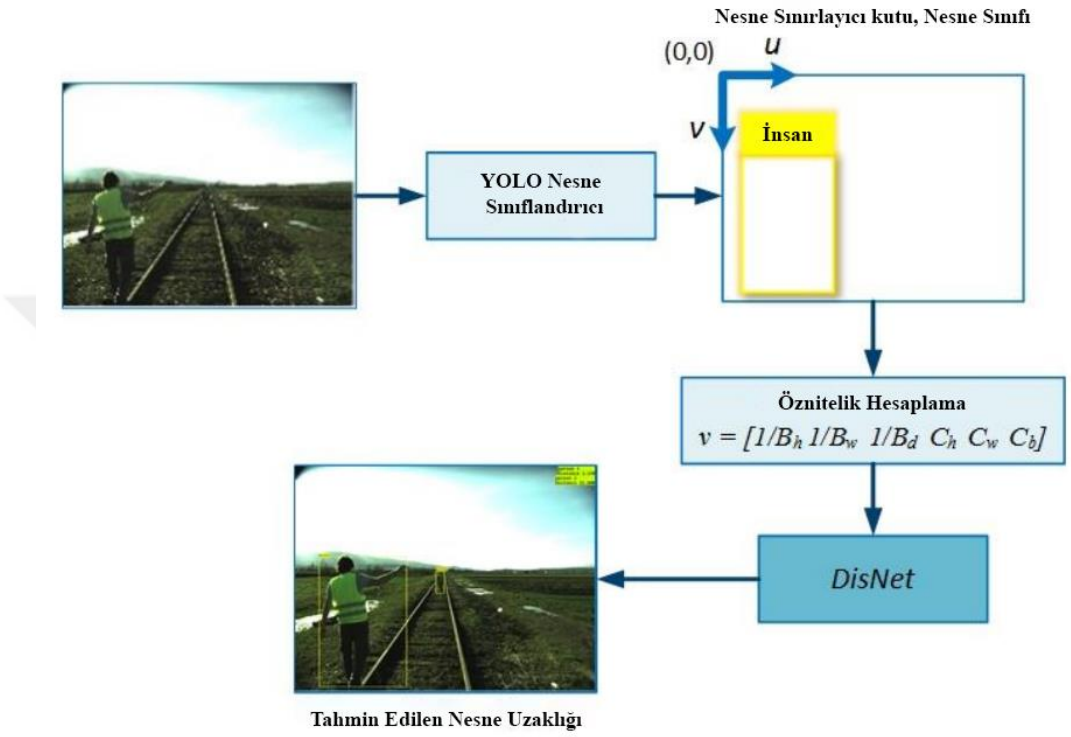
3.1.3 Hedef Tanımlama ve Uzaklık Bilgisinin Elde Edilmesi

Hedefe olan açının yanında uzaklık bilgisi de ajanın eğitilmesi sürecinde kullanılmıştır. Bu bilginin elde edilmesi sürecinde çevreye ait bir harita kullanılmadığından ve hedefin konum bilgileri bilinmediğinden farklı bir yönteme gereksinim duyulmuştur. Bu nedenle hedefin anlık olarak tespit edilmesi ve mesafe bilgisinin tahmin edilmesi sürecinde YOLOv3 tabanlı DisNet [48] kullanılmıştır. YOLOv3 kamera üzerinden nesne tespitinde kullanılan bir derin öğrenme algoritmasıdır. Açık kaynak kodlu bir ağ olan olan Darknet [49] yapısını kullanır. YOLOv3 19 evrişimsel katmandan oluşan Darknet-19 yapısını kullanırken yapılan evrişimsel ve kalıntı ağ eklemeleriyle eğitim süreci daha uzun olan ancak daha yüksek doğruluk değerleriyle sonuçlar veren Darknet-53 ağ yapısı geliştirilmiştir. Çizelge 3.1'de Darknet-53 yapısı gösterilmiştir.

Çizelge 3.1: Darknet-53 ağ yapısı

	Tip	Filtreler	Boyut/Adım	Çıktı
	Evrişimsel	32	3 × 3	256 × 256
	Evrişimsel	64	3 × 3/2	128 × 128
1 ×	Evrişimsel	32	1 × 1	
	Evrişimsel	64	3 × 3	
	Kalıntı			
	Evrişimsel	128	3 × 3/2	64 × 64
2 ×	Evrişimsel	64	1 × 1	64 × 64
	Evrişimsel	128	3 × 3	
	Kalıntı			
	Evrişimsel	256	3 × 3/2	32 × 32
8 ×	Evrişimsel	128	1 × 1	32 × 32
	Evrişimsel	256	3 × 3	
	Kalıntı			
	Evrişimsel	512	3 × 3/2	16 × 16
8 ×	Evrişimsel	256	1 × 1	16 × 16
	Evrişimsel	512	3 × 3	
	Kalıntı			
	Evrişimsel	1024	3 × 3/2	8 × 8
4 ×	Evrişimsel	512	1 × 1	8 × 8
	Evrişimsel	1024	3 × 3	
	Kalıntı			
	Ortalama havuzlama		Global	
	bağlantılı Softmax		1000	

DisNet [48] YOLOv3 yapısıyla bağlantılı üç tane gizli katman kullanan bir ağ yapısıdır. Nesnelerin tespit edilmesi ve uzaklıklarının tahmin edilmesinde kullanılmaktadır. Nesne ve mesafe tespiti yalnızca tek bir kamera kullanılarak yapılabilmektedir. Eğitim sürecinde COCO veri kümesi [50] kullanılmıştır. Şekil 3.4'te DisNet'in yapısı gösterilmiştir.



Şekil 3.4: DisNet genel yapısı [46]

Burada B_h , B_w , B_d sınırlayıcı kutunun yükseklik, genişlik ve köşegen uzunluğunu gösterirken C_h , C_w , C_d ise sınıfın ortalama yükseklik, genişlik ve köşegen değerini göstermektedir. Bu tez çalışmasında ajanın eğitilmesi sürecinde gerekli olan mesafe bilgileri DisNet kullanılarak elde edilmiştir. Mesafe bilgisi ajanın eğitilmesinin yanında ödül-ceza sisteminde yön bilgisi ile birlikte kullanılmıştır.

3.2 Eylem Uzayı

Robotun her bir adımda seçmesi gereken eylemler bulunmaktadır. Bu eylemler sonucunda farklı yönlerde hareket edebilir. TurtleBot doğrusal ve açısal olarak farklı hızlarda hareket edebilmektedir. Eylem uzayını oluşturmak için iki farklı doğrusal hız (0.2, 0.4) m/s ve beş farklı açısal hız $\left(\frac{\Pi}{4}, \frac{\Pi}{8}, 0, -\frac{\Pi}{8}, -\frac{\Pi}{4}\right)$ olmak üzere yedi farklı eylem seçilmiştir. Ajan tanımlanan yedi eylem arasından seçimini yapabilir. Eylemlerin bu

şekilde sınırlandırılmasının nedeni Q-fonksiyonunun en iyi değerlere yakınsamasını kolaylaştırmaktır. Kamera yalnızca TurtleBot'un ön tarafında bulunduğu için hareket ileri yönlü olacak şekilde sınırlandırılmıştır.

3.3 Ödül ve Ceza Sistemi

Ajanın temel amacı en yüksek ödülü elde etmektir. Gezgin robot engellere çarpınca negatif, engellerden kaçındığında ve hedefe ulaştığında ise pozitif ödül almaktadır. Bu amaçla Denklem (5.1)'de gösterilen ödül fonksiyonu tanımlanmıştır.

$$r = \begin{cases} +2 & Hedef \\ -1 & Çarpışma \\ v(\frac{\cos(\theta)+cos(10x)}{2}) - 0.01 & Diğer \end{cases} \quad (5.1)$$

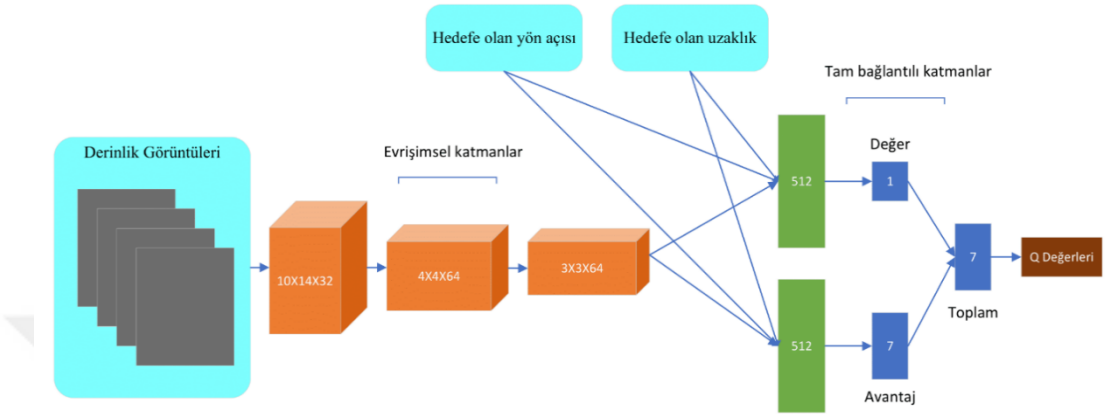
TurtleBot engellere çarpınca -1 değerinde negatif ödül alırken hedefe ulaşması durumunda ise +2 değerinde pozitif ödül almaktadır. Ayrıca hedefe yaklaştığı her durumda da ödüllendirilmektedir. Burada v gezgin robotun hızını gösterirken, θ hedefe olan açıyı, x ise hedefe olan uzaklığı göstermektedir. 0.01 değeri sabit olarak kullanılmıştır. Sabit değerin kullanılması zamana bağlı olarak elde edilen ödül miktarını azaltmaktadır. Böylelikle gezgin robot hedefe daha hızlı şekilde ulaşmaya çalışmaktadır. Bu ödüllendirme yaklaşımı Xie ve diğ. [43] tarafından yapılan çalışmaya benzer bir yaklaşım göstermektedir. Bunun nedeni yapılan çalışmada kullanılan fonksiyonun engellerden kaçınma konusunda başarılı sonuçlar vermesidir.

3.4 Ağ Mimarileri

Bu tez kapsamında eğitim ve test süreçlerinde kullanılmak üzere D3QN-A, D3QN-B ve D3QN-C olarak adlandırılan üç farklı ağ mimarisi tasarlanmıştır. Ağ mimarileri D3QN [43] yapısını temel almaktadır. Farklılık ajanın eğitilmesi için ağa verilen bilgilerin ağda kullanılma bölümleri ve ağa veriliş biçiminde kendini göstermektedir. D3QN tercih edilmesinin nedeni bu ağ yapısının engelden kaçınma konusunda başarılı sonuçlar vermesidir. Ayrıca bu ağ yapısının temelini oluşturan avantaj ve değer ikililerinin ayrı hesaplanması ajanın öğrenme sürecini hızlandırmaktadır. Ağ yapısında derinlik görüntülerinin işlendiği evrişimsel katmanların yanında tam bağlantılı katmanlar kullanılmış ve her bir eylem için Q değerleri hesaplanmıştır. Tasarlanan tüm ağ mimarilerinde kullanılan derinlik görüntüleri 80×64 boyutundadır. Bu şekilde kullanılmasının nedeni hesaplama maliyetlerini düşük tutmaktır.

3.4.1 D3QN-A

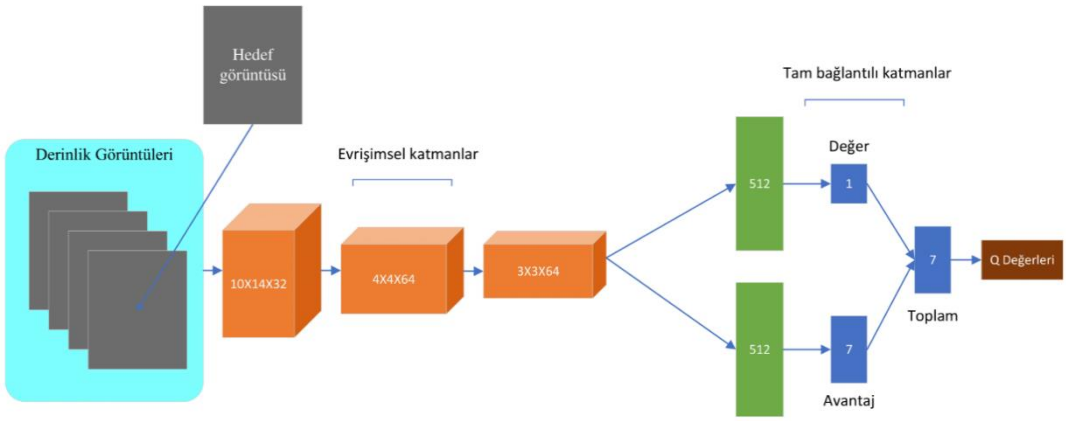
Bu ağ yapısında tam bağlantılı katmanlara ek bilgi olarak yön ve mesafe bilgileri de verilerek ajanın eğitim sürecinde kullanılmıştır. Şekil 3.5'te kullanılan ağ mimarisi gösterilmiştir.



Şekil 3.5: D3QN-A ağ mimarisi

3.4.2 D3QN-B

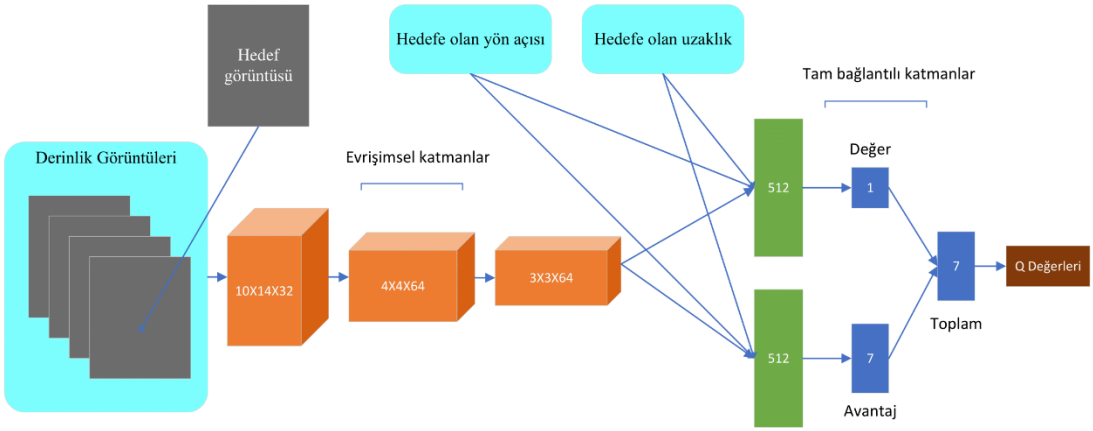
D3QN-B ağ yapısında yön açısı ve hedefe olan uzaklık değerlerini tam bağlantılı katmanlara ek bilgi olarak vermek yerine hedef görüntüsü derinlik görüntüleri ile birlikte evrişimsel katmanlarda kullanılmıştır. Hedef görüntülerinde yön açıları normalize edilmiş ve [0,1] aralığında değerler alarak her bir piksel için evrişimsel ağın eğitim sürecinde derinlik görüntüleri ile birlikte ağa verilmiştir. Bu yaklaşımın kullanılmasında AlphaGo pekiştirmeli öğrenme ajanının [51] yapısı örnek teşkil etmiştir. Şekil 3.6'da D3QN-B ağ mimarisi gösterilmiştir.



Şekil 3.6: D3QN-B ağ mimarisi

3.4.3 D3QN-C

D3QN-C ağ yapısı, D3QN-B ile benzer özelliktedir. Aralarındaki fark, D3QN-C'de tam bağlantılı katmanlara ek bilgi olarak hedefe olan uzaklık ve yön açısı değerlerinin verilmesidir. Şekil 3.7'de D3QN-C ağ mimarisi gösterilmiştir.



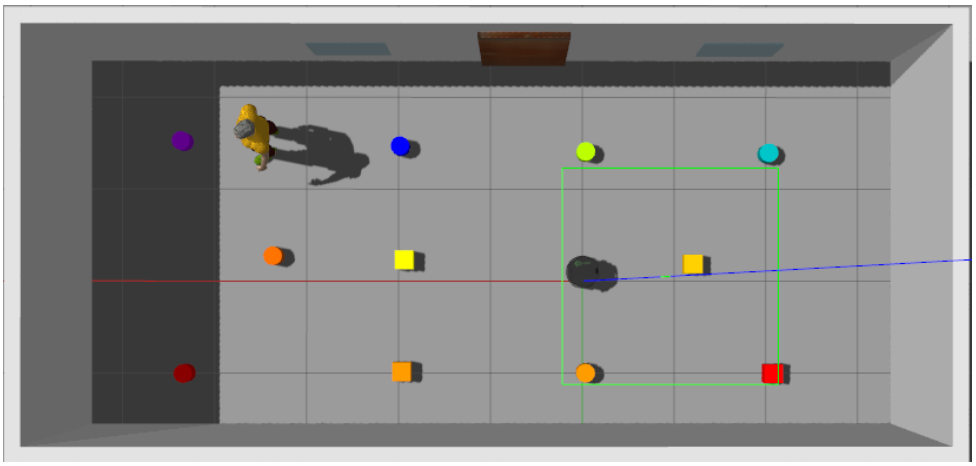
Şekil 3.7: D3QN-C ağ mimarisi

3.5 Çevre Tasarımı

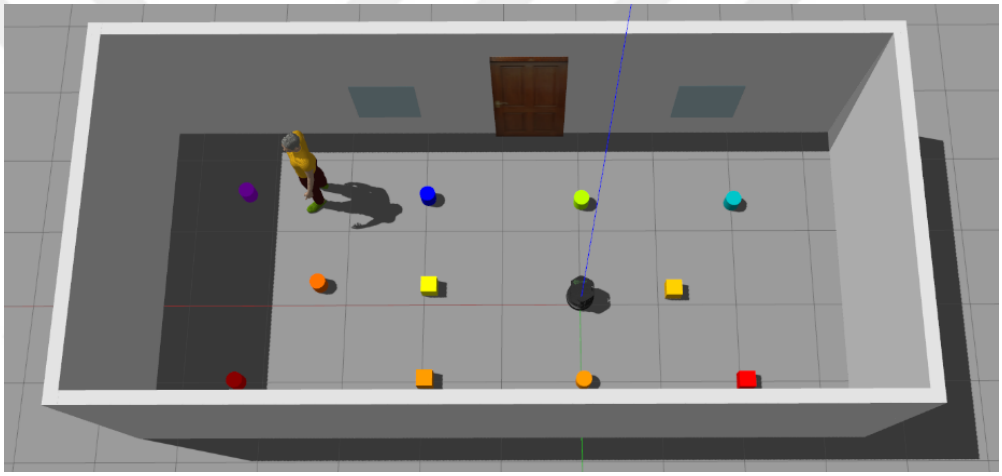
Ajanın eğitim sürecini gerçekleştirmek için Gazebo simülasyon ortamında iki farklı çevre tasarımı yapılmıştır.

3.5.1 Basit Çevre Tasarımı

Tasarlanan ilk çevre $8 \times 4\, m^2$ boyutunda ve $2\, m$ duvarlara sahip olan bir oda görünümdedir. Odanın içerisine yerleştirilmiş 11 tane silindir ve küp şeklinde engeller bulunmaktadır. Gezgin robotun engellerden kaçınarak her adımda odanın farklı bir bölümünde belirecek olan insan şeklindeki hedefe ulaşması beklenmektedir. Basit çevre tasarımındaki engeller sabit şekilde konumlandırılmış olup hareket etmemektedir. Şekil 3.8'de basit çevrenin üstten görünümü, Şekil 3.9'da ise yandan görünümü gösterilmiştir.



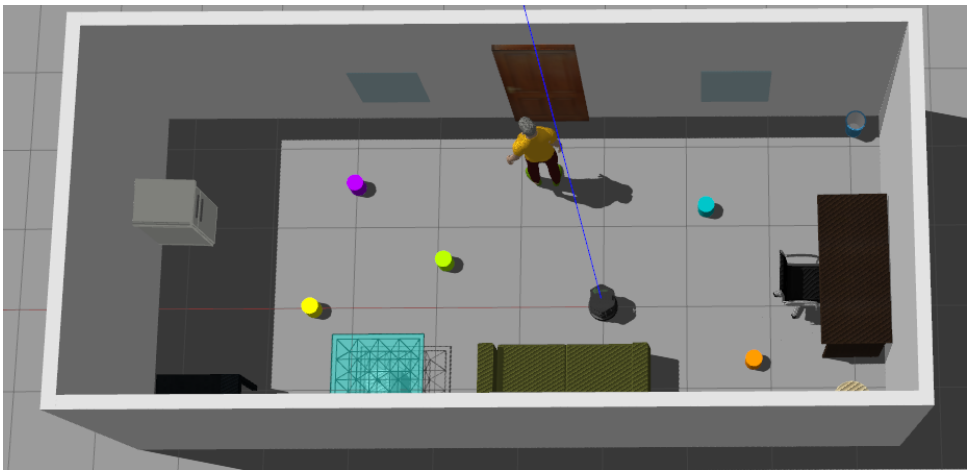
Şekil 3.8: Basit çevrenin üstten görünümü



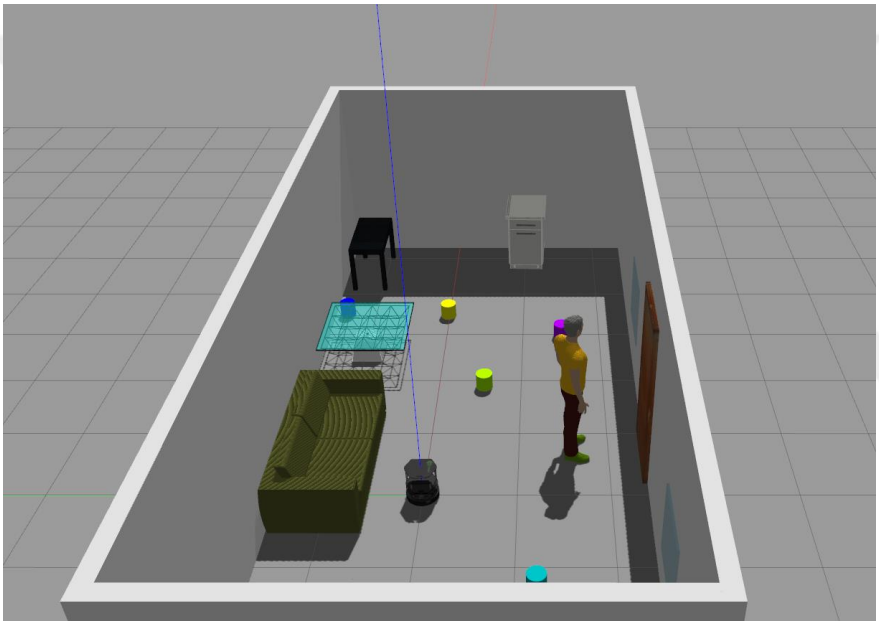
Şekil 3.9: Basit çevrenin yandan görünümü

3.5.2 Karmaşık Çevre Tasarımı

Tasarlanan ikinci çevre tasarımı ilk çevreye benzer şekilde $8 \times 4\ m^2$ boyutunda ve $2\ m$ duvarlara sahip olan bir oda görünümdedir. İlk çevrede kullanılan silindir şeklindeki engellere ek olarak ev eşyası modelleri kullanılmıştır. Buradaki en önemli farklılık bu çevrede hareketli engellerin kullanılmasıdır. Silindir şeklindeki üç engel odanın içerisinde hareket etmektedir. Bu çevrede gezgin robotun sabit ve hareketli engellerden kaçınarak insan modeline ulaşması beklenmektedir. Şekil 3.10'da karmaşık çevrenin üstten görünümü, şekil 3.11'de ise yandan görünümü gösterilmiştir.



Şekil 3.10: Karmaşık çevrenin üstten görünümü



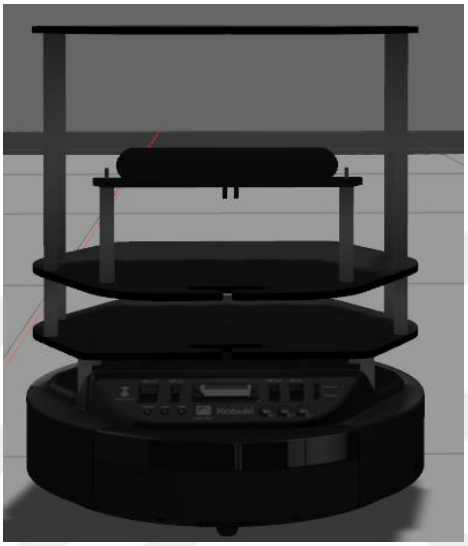
Şekil 3.11: Karmaşık çevrenin yandan görünümü

3.6 TurtleBot Modeli ve Özellikleri

Tasarlanan çevrelerde kullanılacak gezgin robot olarak TurtleBot 2 modeli tercih edilmiştir. Bu modelin tercih edilmesinin nedeni üzerinde dahili olarak Microsoft Kinect sensör bulunmasıdır. TurtleBot 3 modelinde kinect sensör olarak Intel R200 sensör kullanılabilmektedir. İncelenen çalışmalara paralel olarak Gazebo ortamında yapılan testlerde Microsoft Kinect sensörün Intel R200'e göre 3-boyutlu yüzeylerin tespitinde daha başarılı olduğu sonucuna varılmıştır. TurtleBot 3 modeline de Microsoft Kinect sensör harici olarak eklenebilmektedir; ancak uyumluluk sorunları olabileceği ve Microsoft Kinect sensörün TurtleBot 2 ile daha kararlı çalışacağı göz

önüne alınarak bu tez çalışmasında TurtleBot 2 modeli kullanılmıştır. Şekil 3.12'de üzerinde Microsoft Kinect sensör bulunan TurtleBot 2 gösterilmiştir.

Kinect sensörler derinlik görüntüleri elde etmek için kullanılan ve genellikle oyun uygulamalarında kullanılan sensörlerdir. Üzerinde bir derinlik sensörü, bir RGB kamera ve bir mikrofon bulunmaktadır. Derinlik sensörü 640×480 boyutunda görüntüler elde edebilmektedir. Mesafe olarak ise 0.4 metreden 4.5 metre uzaklıklara kadar verimli şekilde kullanılabilmektedir.



Şekil 3.12: Turtlebot 2 modeli ve Microsoft Kinect sensör

3.7 Eğitim Süreci

Ajanın eğitimi tasarlanan basit ve karmaşık çevrelerde gerçekleştirilmiştir. Süreç bölümlere ayrılmıştır. Robot, hareketine odanın ortasında başlamaktadır. Her bir bölümde hedefin konumu değiştirilmektedir. Robot hedefine ulaştığında pozitif ödül almakta, çevredeki engellere çarptığında ya da herhangi bir nesneyle arasındaki mesafe 0.5 metreden az olduğunda negatif ödül almaktadır. Pozitif veya negatif bir ödül alınmasıyla bölüm sonlanmakta ve diğer bölüme geçilmektedir. Robot eylemlerine derinlik görüntülerinin yanında, mesafe ve yön bilgilerini kullanarak karar vermekte buna bağlı olarak da ödüllendirilmektedir. Bunun yanında her bir bölümde ajanın deneyimleri depolanmakta daha sonra bu deneyimlerin içinden tekdüze örnekler alınarak D3QN algoritması kullanan ajanın eğitilmesinde kullanılmaktadır. Eğitim süreci her bir model için yaklaşık 47 saat sürmektedir. Daha iyi performans elde edebilmek için farklı hiper parametrelerle modeller üç kez çalıştırılmıştır. Sonuç olarak toplam 141 saat bir modelin eğitim sürecine ayrılmıştır.

Eğitim süreci Intel i7 11370H işlemci, 32 GB RAM ve Nvidia Geforce 1060 ekran kartı kullanan bir bilgisayar kullanılarak gerçekleştirilmiştir. ROS ve Gazebo, Linux tabanlı Ubuntu 18.04 işletim sistemi üzerinde çalıştırılarak ajanın eğitilmesi için gerekli ortamı oluşturmuştur. Gerekli olan program ise Google tarafından geliştirilen Tensorflow kütüphanesi ve OpenCV kütüphanesinden faydalanılarak Python 3.8 ile yazılmıştır. ROS versiyonu olarak ROS Melodic Morenia kullanılmıştır.

3.8 Hiper Parametreler

Benzer çalışmaların incelenmesi ve eğitim sürecinde farklı hiper parametreler ile yapılan denemeler sonucu en iyi performansın çizelge 3.2'de gösterilen hiper parametreler ile elde edildiği görülmüştür.

Çizelge 3.2: Hiper parametreler

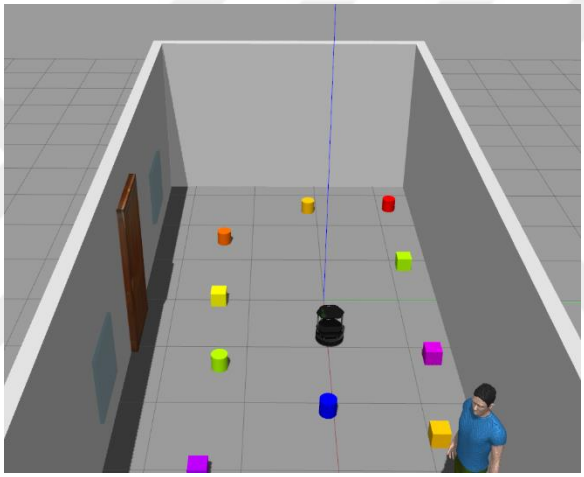
Hiper parametre	Değer
Gama (γ)	0.99
Başlangıç epsilon (ϵ) değeri	1
Son epsilon (ϵ) değeri	0.01
Keşif adımları	50.000
Deneyim tekrarı ön bellek boyutu	50.000
Yığın boyutu	8
Hedef ağ güncelleme oranı	0.001
Öğrenme oranı	0.0001

Burada γ indirim oranını göstermektedir. İndirim oranı başlangıçtaki ödüllerin gelecek ödüllere göre daha değerli olmasını sağlamaktadır. Bu şekilde daha hızlı ödül kazanma teşvik edilmektedir. Epsilon (ϵ) değeri ve keşif adımları ajanın yeni durumlar ve eylemler deneme konusunda ne kadar istekli olacağını belirlemektedir. Eğitim süreci ilerledikçe epsilon değeri azalmaktadır. Deneyim tekrarı ön bellek boyutu ajanın deneyimlerinin ne kadarının depolanacağını, yığın boyutu ise her bir bölümde bu deneyimlerden ne kadarlık örnekler alınarak kullanılacağını göstermektedir. Hedef ağ

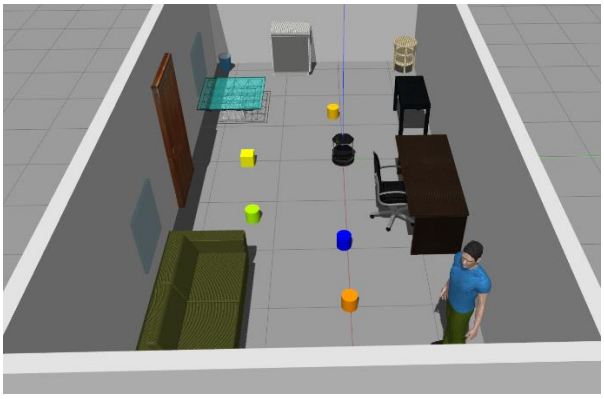
güncelleme oranı ağın hangi sıklıkta güncelleneceğini belirtmektedir. Öğrenme oranı ise optimizasyon algoritması tarafından kullanılmaktadır.

3.9 Test Ortamı

Eğitim süreci gerçekleştirildikten sonra elde edilen modeller iki farklı çevre üzerinden test edilmiştir. Test çevreleri eğitim sürecinde kullanılan çevrelere benzer özelliktedir. Aynı nesneler farklı konumlara yerleştirilerek robotun hedefe ulaşması beklenmiştir. Karmaşık test ortamında eğitim ortamından farklı olarak hareketli engellerin sayısı 4'e çıkartılmıştır. Robot ve hedef 10 farklı konuma yerleştirilerek modeller her bir test ortamı için 100'er defa test edilmiştir. Şekil 3.13'te basit test ortamı, şekil 3.14'te ise karmaşık test ortamı gösterilmiştir.



Şekil 3.13: Basit test ortamı

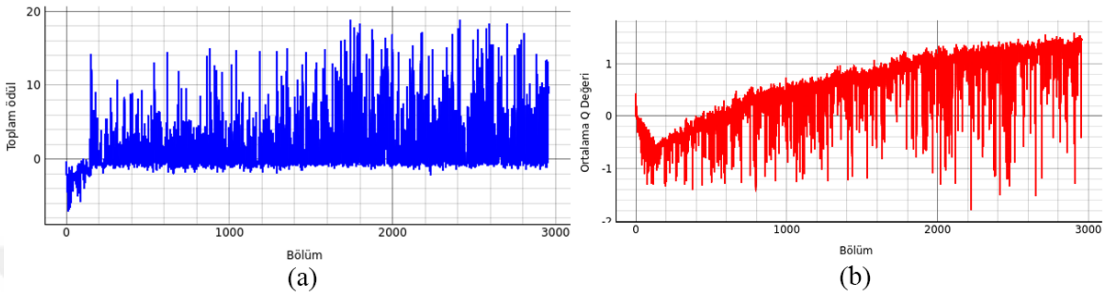


Şekil 3.14: Karmaşık test ortamı

4. DENEYSEL SONUÇLAR

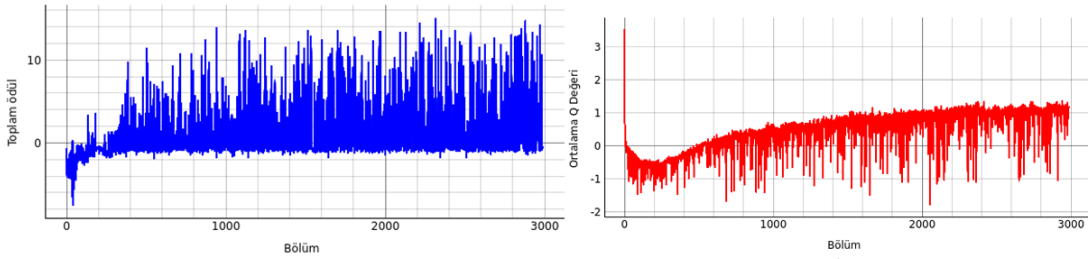
4.1 D3QN-A

D3QN-A ağ modeli basit ve karmaşık çevrelerde eğitilerek sonuçlar elde edilmiştir. Şekil 4.1'de basit çevreye ait ödül ve ortalama Q değerleri, şekil 4.2'de ise karmaşık çevreye ait ödül ve ortalama Q değerleri gösterilmiştir.



(a) (b)

Şekil 4.1: Basit çevreye ait (a) ödül ve (b) ortalama Q değerleri



(a) (b)

Şekil 4.2: Karmaşık çevreye ait (a) ödül ve (b) ortalama Q değerleri

Eğitim süreci sonunda elde edilen modeller basit ve karmaşık test ortamlarında test edilmiştir. Çizelge 4.1'de basit çevrede eğitilen modelin her iki test ortamında elde ettiği ortalama ödüller ve hedefe ulaşma başarıları gösterilmiştir. Çizelge 4.2'de ise karmaşık çevrede eğitilen modelin her iki test ortamında elde ettiği ortalama ödüller ve hedefe ulaşma başarıları gösterilmiştir.

Çizelge 4.1: Basit çevrede eğitilen modelin her iki test ortamında elde ettiği ortalama ödüller ve hedefe ulaşma başarıları

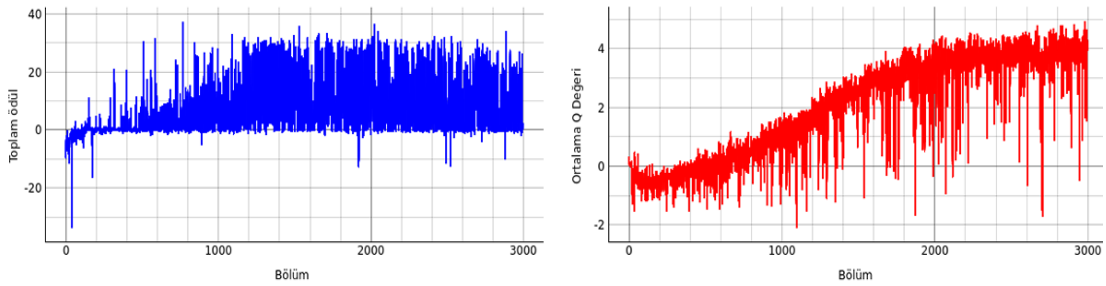
Test Ortamı	Ortalama Ödül	Hedefe Ulaşma Başarısı
Basit	16.7	%68
Karmaşık	8.9	%56

Çizelge 4.2: Karmaşık çevrede eğitilen modelin her iki test ortamında elde ettiği ortalama ödüller ve hedefe ulaşma başarıları

Test Ortamı	Ortalama Ödül	Hedefe Ulaşma Başarısı
Basit	11.5	%61
Karmaşık	14.2	%65

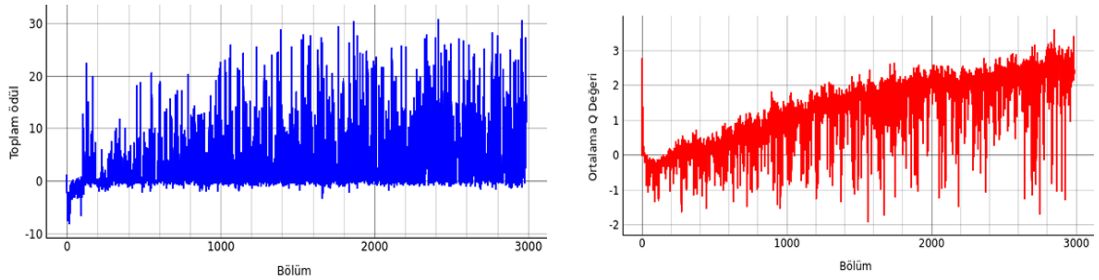
4.2 D3QN-B

D3QN-B ağ modeli basit ve karmaşık çevrelerde eğitilerek sonuçlar elde edilmiştir. Şekil 4.3'te basit çevreye ait ödül ve ortalama Q değerleri, şekil 4.4'te ise karmaşık çevreye ait ödül ve ortalama Q değerleri gösterilmiştir.



(a) (b)

Şekil 4.3: Basit çevreye ait (a) ödül ve (b) ortalama Q değerleri



(a) (b)

Şekil 4.4: Karmaşık çevreye ait (a) ödül ve (b) ortalama Q değerleri

Eğitim süreci sonunda elde edilen modeller basit ve karmaşık test ortamlarında test edilmiştir. Çizelge 4.3'te basit çevrede eğitilen modelin her iki test ortamında elde ettiği ortalama ödüller ve hedefe ulaşma başarıları gösterilmiştir. Çizelge 4.4'te ise karmaşık çevrede eğitilen modelin her iki test ortamında elde ettiği ortalama ödüller ve hedefe ulaşma başarıları gösterilmiştir.

Çizelge 4.3: Basit çevrede eğitilen modelin her iki test ortamında elde ettiği ortalama ödüller ve hedefe ulaşma başarıları

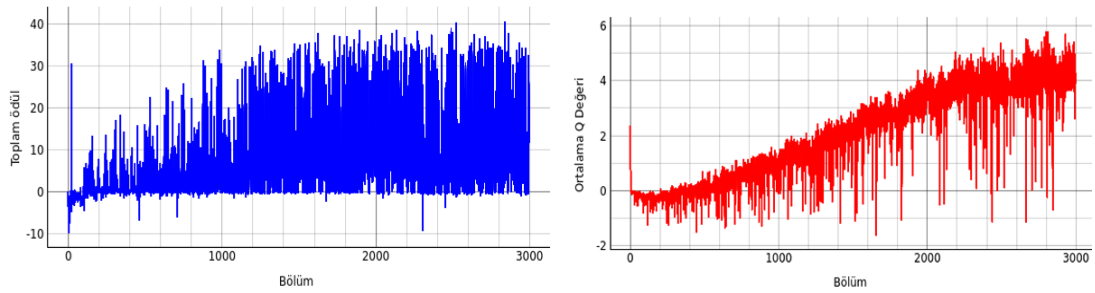
Test Ortamı	Ortalama Ödül	Hedefe Ulaşma Başarısı
Basit	24.5	%85
Karmaşık	16.6	%67

Çizelge 4.4: Karmaşık çevrede eğitilen modelin her iki test ortamında elde ettiği ortalama ödüller ve hedefe ulaşma başarıları

Test Ortamı	Ortalama Ödül	Hedefe Ulaşma Başarısı
Basit	21.3	%79
Karmaşık	22.7	%81

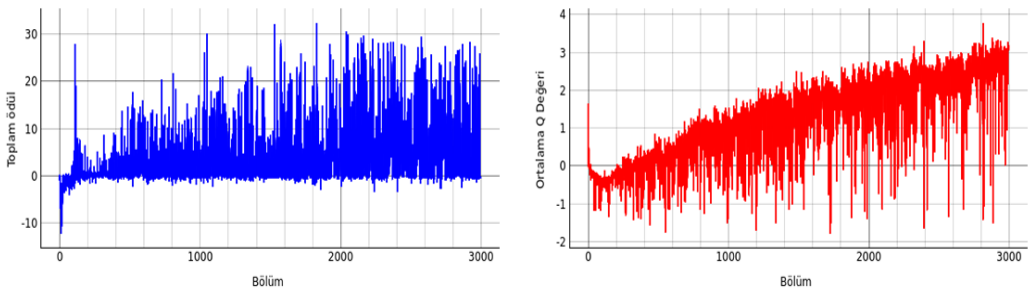
4.3 D3QN-C

D3QN-C ağ modeli basit ve karmaşık çevrelerde eğitilerek sonuçlar elde edilmiştir. Şekil 4.5'te basit çevreye ait ödül ve ortalama Q değerleri, şekil 4.6'da ise karmaşık çevreye ait ödül ve ortalama Q değerleri gösterilmiştir.



(a) (b)

Şekil 4.5: Basit çevreye ait (a) ödül ve (b) ortalama Q değerleri



(a) (b)

Şekil 4.6: Karmaşık çevreye ait (a) ödül ve (b) ortalama Q değerleri

Eğitim süreci sonunda elde edilen modeller basit ve karmaşık test ortamlarında test edilmiştir. Çizelge 4.5'te basit çevrede eğitilen modelin her iki test ortamında elde ettiği ortalama ödüller ve hedefe ulaşma başarıları gösterilmiştir. Çizelge 4.6'da ise karmaşık çevrede eğitilen modelin her iki test ortamında elde ettiği ortalama ödüller ve hedefe ulaşma başarıları gösterilmiştir.

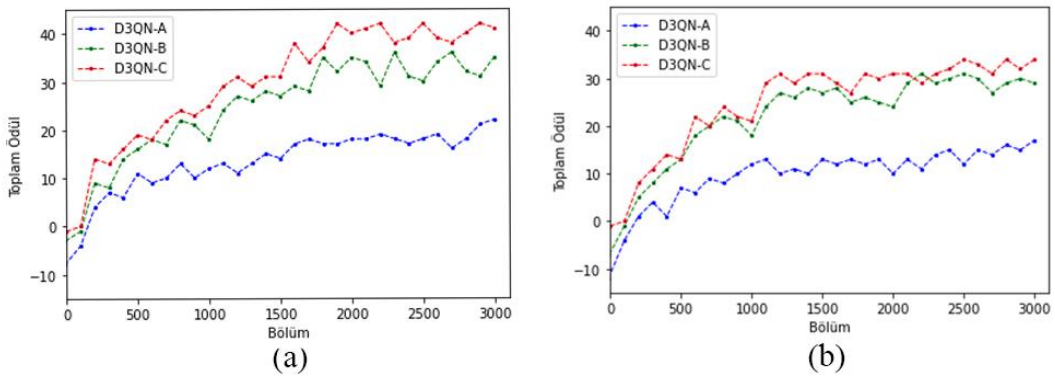
Çizelge 4.5: Basit çevrede eğitilen modelin her iki test ortamında elde ettiği ortalama ödüller ve hedefe ulaşma başarıları

Test Ortamı	Ortalama Ödül	Hedefe Ulaşma Başarısı
Basit	26.7	%89
Karmaşık	19.8	%72

Çizelge 4.6: Karmaşık çevrede eğitilen modelin her iki test ortamında elde ettiği ortalama ödüller ve hedefe ulaşma başarıları

Test Ortamı	Ortalama Ödül	Hedefe Ulaşma Başarısı
Basit	23.8	%84
Karmaşık	26.3	%87

Eğitim ve test sonuçları incelendiğinde en başarılı modelin D3QN-C modeli olduğu görülmüştür. Şekil 4.7'de basit ve karmaşık eğitim ortamlarında elde edilen ödül miktarlarına göre modellerin karşılaştırılması gösterilmiştir.

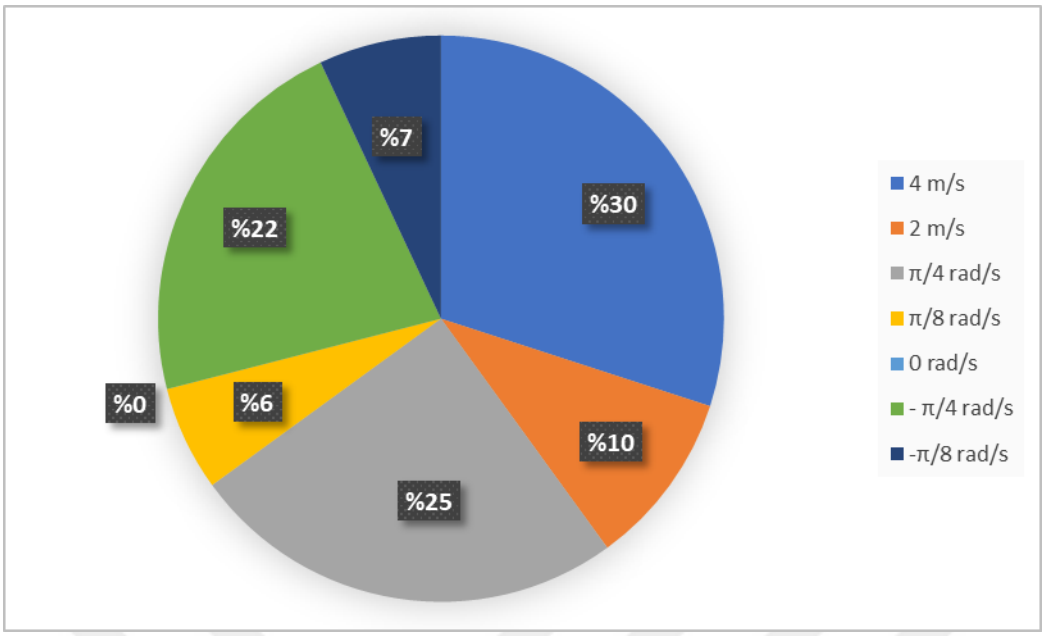


Şekil 4.7: Basit (a) ve karmaşık (b) eğitim ortamlarında elde edilen ödül miktarlarına göre modellerin karşılaştırılması

Hem basit hem de karmaşık test ortamlarında en yüksek ödül ve hedefe ulaşma yüzdeleri bu model ile elde edilmiştir. Basit eğitim çevresinde eğitilen D3QN-C modeli basit test ortamında %89, karmaşık test ortamında ise %72 hedefe ulaşma başarısı göstermiştir. Karmaşık eğitim ortamında eğitilen D3QN-C modeli ise basit ve karmaşık test ortamlarında sırasıyla %84 ve %87 başarı oranlarına ulaşmıştır. D3QN-C modelinden farklı olarak mesafe ve yön bilgisinin tam bağlantılı katmanlarda kullanılmadığı D3QN-B modeli ise basit eğitim ortamında eğitildiğinde basit ve karmaşık test ortamlarında sırasıyla %85 ve %67 oranında başarı gösterirken karmaşık eğitim ortamında eğitildiğinde ise %79 ve %81 başarı göstermiştir. Bu bakımdan mesafe ve yön bilgisinin tam bağlantılı katmanlara ek bilgi olarak verilmesinin ve bu verilerin ödül-ceza sisteminde kullanılmasının model performansı üzerinde olumlu etki yarattığı yorumu yapılabilir. En düşük değerler ise D3QN-A modeli ile elde edilmiştir. Basit eğitim ortamında eğitilen D3QN-A modeli basit ve karmaşık test ortamlarında %68 ve %56 başarı elde etmiştir. Karmaşık eğitim ortamında eğitilen D3QN-A modeli ise %61 ve %65 başarı oranlarına ulaşmıştır. Hedef görüntüsünün derinlik görüntüleri ile birlikte evrişimsel katmanlara girdi olarak verilmesinin, yön verisinin tam bağlantılı katmanlara ek bilgi olarak verilmesinden daha etkili olduğu sonucu çıkarılabilir. Bu modelde tam bağlantılı katmanlara verilen yön bilgisi ile ödül ilişkisi ağ tarafından daha geç farkedilmiştir. Bu da modelin performansını olumsuz etkilemiştir.

4.4 Eylem Analizi

Eğitim ve test süreçlerinde kullanılan modellerden en yüksek performans gösteren model olan D3QN-C modeli için eylem analizi yapılarak ajan tarafından en çok tercih edilen eylemler ve yüzdeleri belirlenmiştir. Şekil 4.8'de ajan tarafından seçilen eylemlerin tercih edilme yüzdeleri gösterilmiştir.



Şekil 4.8: Ajan tarafından seçilen eylemlerin tercih edilme yüzdeleri

Grafik incelendiğinde ajan tarafından en çok gerçekleştirilen eylemin %30 oranıyla 4 m/sn doğrusal hız eylemi olduğu görülmüştür. %10 oranıyla seçilen 2 m/sn doğrusal eylemine göre daha çok tercih edilmesinin nedeni olarak robotun hedefe en hızlı şekilde ulaşmaya teşvik edilmesi gösterilebilir. Bu nedenle robotun zorunlu durumlar dışında 4 m/s hızında hareket etmeyi seçtiği yorumu yapılabilir. Açısal hızlar incelendiğinde ise π/4 rad/s ve -π/4 rad/s eylemlerinin sırasıyla %25 ve %22 oranlarıyla tercih edildiği görülmektedir. Bu oranlar robotun daha yüksek dönme açılarıyla ilerlediğini göstermektedir. Diğer dikkat çekici bir nokta 0 rad/s eyleminin %0 oranıyla hiç tercih edilmediğinin belirlenmesidir. Yani robot asla düz gitmemekte sürekli dönerek ilerlemeyi tercih etmektedir. Derinlik kamerası ve RGB kameranın robotun yalnızca ön tarafında bulundukları dikkate alınırsa böyle bir hareketin daha geniş bir görüş açısı sağlamak için tercih edildiği düşünülebilir.

5. SONUÇLAR VE ÖNERİLER

Donanım alanındaki hızlı ilerlemelere bağlı olarak son yıllarda yapay zekâ üzerine yapılan çalışmalar hız kazanmıştır. Buna bağlı olarak hayatın her alanında yapay zekâ uygulamaları kendilerine yer bulmaktadır. Bu alanlardan biri de robotik alanıdır. Robotik alanındaki temel problemlerden biri olan navigasyon problemini çözmeye yönelik literatürde farklı algoritmalar kullanılmaktadır. Yapay zekânın alt dallarından biri olan ve özellikle bilgisayar oyunlarında başarılı sonuçlar veren derin pekiştirmeli öğrenme algoritmaları son yıllarda araştırmacıların üzerinde yoğunlaştığı alanların başında gelmektedir. Çevre hakkında herhangi bir ön bilgiye gereksinim duyulmaması ve insan öğrenmesine benzer özellik gösteren ödül-ceza sistemi bu algoritmaların robotların navigasyon problemlerinde kullanılmaları için ilgiyi arttırmıştır.

Bu tez kapsamında bilinmeyen bir çevreye bırakılan bir robotun çevresindeki sabit ve hareketli engellerden kaçınarak verilen hedefeleri tanıması ve bu hedeflere ulaşması beklenmiştir. Bu amaçla derin pekiştirmeli öğrenme algoritmalarından D3QN algoritmasını temel alan üç farklı ağ modeli tasarlanmıştır. Robotun önünde bulunan derinlik kamerasından elde edilen derinlik görüntülerinin yanında RGB kamera kullanılarak elde edilen hedefe ait yön ve mesafe bilgileri de modellerin eğitilmesi sürecinde kullanılmıştır. Hedeflerin tanınması ve mesafe bilgisinin belirlenmesi aşamasında Darknet altyapısı kullanan DisNet'ten faydalanılmıştır. Modeller basit ve karmaşık olarak adlandırılan eğitim çevreleri kullanılarak eğitilmiştir. Eğitilen modeller basit ve karmaşık test ortamlarında çalıştırılmış ve performansları elde edilen ödül miktarları ve hedefe ulaşabilme başarıları üzerinden ölçülmüştür. Sonuçlar değerlendirildiğinde en yüksek başarı oranlarına D3QN-C modeli kullanılarak ulaşıldığı görülmüştür. Bu model basit ve karmaşık test ortamlarında sırasıyla %89 ve %87 hedefe ulaşma başarısıyla diğer modellere göre yüksek başarı sağlamıştır.

Sonuç olarak D3QN algoritmasının robotların navigasyon problemlerinde başarılı bir şekilde kullanılabileceği görülmüştür. Bunun yanında güncel hedef tanılama ve mesafe bilgisi elde etme yöntemlerinin D3QN algoritması ile birlikte kullanılmasının algoritmanın başarısına olumlu katkı sağladığı da belirlenmiştir. Ayrıca kullanılan model literatürdeki çoğu çalışmanın aksine hareketli engellerin bulunduğu ortamlarda da test edilmiş ve başarılı sonuçlar vermiştir. Bu bakımdan derin pekiştirmeli öğrenme

algoritmalarının yayaların ve diğer hareketli engellerin bulunduğu ortamlarda da kullanılabileceği çalışmalara temel oluşturabilir.

Bu tez çalışmasındaki kısıtlayıcı yönlerin başında hesaplama maliyeti gelmektedir. Tasarlanan modeller 80×64 boyutundaki derinlik görüntüleri üzerinden eğitilmiştir. Eğitim sürecini oldukça uzatmasından dolayı daha yüksek çözünürlükte derinlik görüntüleri kullanılmamış ve bu nedenle modelin yüksek çözünürlüklerde nasıl performans vereceği görülememiştir. Daha yüksek donanım özelliklerine sahip cihazlarda daha yüksek çözünürlüklerde modelin performansı test edilebilir. Bunun yanında yine hesaplama maliyetleri gözetilerek model üç farklı hiper parametre kombinasyonuyla test edilebilmiştir. Farklı parametrelerle ve mümkünse meta sezgisel algoritmalar kullanılarak modelin başarısı test edilebilir; ancak bunun yüksek işlem gücü ve zaman gerektireceği açıktır. Değinilmesi gereken diğer bir konu ise modelin yalnızca simülasyon ortamı üzerinden eğitilip test edilmesidir. Gazebo ortamı her ne kadar gerçeğe oldukça yakın özellikte ortamlar sunmaya imkan verse de gerçek dünyada modelin eğitilip test edilmesi modelin güvenilirliğini arttıracaktır. Bu amaçla gerçek bir TurtleBot modeli benzer özellikte tasarlanan iç mekanlarda eğitilip test edilebilir.

KAYNAKLAR

[1] J. McCarthy, What is artificial intelligence?, (1998).

[2] W. S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, The bulletin of mathematical biophysics, 5(4) (1943) 115-133.

[3] M. Naeem, S. Rizvi, S. T. H., and A. Coronato, A gentle introduction to reinforcement learning and its application in different fields, IEEE Access, (2020).

[4] A. Tuncer, M. Yildirim, and K. Erkan, A hybrid implementation of genetic algorithm for path planning of mobile robots on FPGA, In Computer and Information Sciences III, Springer, London, pp. 459–465, 2013.

[5] B.K. Patle, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, A review: On path planning strategies for navigation of mobile robot, Defence Technology, vol. 15(4) (2019) 582–606.

[6] V. Mnih et al., Playing Atari with Deep Reinforcement Learning, In Conference on Neural Information Processing Systems, pp. 1–9, 2013.

[7] B. Das, M.S. Couceiro, P.A. Vargas, MRoCS : A new multi-robot communication system based on passive action recognition, Rob. Auton. Syst. 82 (2016) 46–60.

[8] H. D. Whyte, Simultaneous localisation and mapping (SLAM): Part I the essential algorithms, Robotics and Automation Magazine. (2006).

[9] P. Sudhakara, V. Ganapathy, Trajectory planning of a mobile robot using enhanced A-star algorithm, Indian Journal of Science and Technology. 9(41) (2016) 1-10.

[10] Y. Katada, A. Nishiguchi, K. Moriwaki, R. Watakabe, Swarm robotic network using Lévy flight in target detection problem. Artif. Life Robot. 21(3) (2016) 295–301.

[11] Z. H. Akpolat, Application of Fuzzy-sliding mode control and electronic load emulation to the robust control of motor drives, University of Nottingham, England, 1999.

[12] G. Du, P. Zhang, Robotics and Computer-Integrated Manufacturing A novel human – manipulators interface using hybrid sensors with Kalman filter and particle filter. Robot. Comput. Integr. Manuf. 38 (2016) 93–101.

[13] A. Farinelli, M.M. Raeissi, N. Marchi, N. Brooks, P. Scerri, Interacting with team oriented plans in multi-robot systems, Auton. Agent. Multi. Agent. Syst. 31(2) (2017) 332–361.

[14] N. S. Pal & S. Sharma, Robot path planning using swarm intelligence: a survey, International Journal of Computer Applications. 83(12) (2013) 5-12.

[15] D. W. Gong, J. Zhang, Y. Zhang, Multi-objective Particle Swarm Optimization for Robot Path Planning in Environment with Danger Sources. J. Comput. 6(8) (2011) 1554-1561.

[16] H. S. Dewang, P.K. Mohanty, S. Kundu, A robust path planning for mobile robot using smart particle swarm optimization. Procedia computer science, 133 (2018) 290-297.

[17] M. Maryam, J. Y. Hwa, N. M. Siti, T. Farzad, and Z. M. D. Siti, Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization. Plos One, 12(3) (2017) 1-24.

[18] H. Che, Z. Wu, R. Kang, and C. Yun, Global path planning for explosion-proof robot based on improved ant colony optimization. In 2016 Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), pp. 36-40, 2016.

[19] A. M. Rao, K. Ramji, and T. N. Kumar, Intelligent navigation of mobile robot using grey wolf colony optimization. Materials Today: Proceedings, 5(9) (2018) 19116-19125.

[20] N. H. Abbas and F. M. Ali, Path planning of an autonomous mobile robot using directed artificial bee colony algorithm. International Journal of Computer Applications, 96(11) (2014).

[21] A. Nayyar, N. G. Nguyen, R. Kumari, and S. Kumar, Robot path planning using modified artificial bee colony algorithm, In Frontiers in Intelligent Computing: Theory and Applications, pp. 25-36, 2020.

[22] L. Khriji, F. Touati, K. Benhmed, and A. Al-Yahmedi, Mobile robot navigation based on Q-learning technique. International Journal of Advanced Robotic Systems. 8(1) (2011) 4.

[23] H. Surmann, C. Jestel, R. Marchel, F. Musberg, H. Elhadj, M. Ardani, Deep Reinforcement learning for real autonomous mobile robot navigation in indoor environments. (2020).

[24] Y. F. Chen, M. Everett, M. Liu, and J. P. How, J. P, Socially aware motion planning with deep reinforcement learning, In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1343-1350, 2017.

[25] H. Bae, G. Kim, J. Kim, D. Qian, and S. Lee, Multi-robot path planning method using reinforcement learning, applied sciences, 9(15) (2019) 3057.

[26] H. Quan, Y. Li and Y. Zhang, A novel mobile robot navigation method based on deep reinforcement learning, International Journal of Advanced Robotic Systems, 17(3) (2020) 1729881420921672.

[27] R. Weideman, Robot Navigation in Cluttered Environments with Deep Reinforcement Learning, 2019.

[28] K. Mehrotra, C. K. Mohan, and S. Ranka, Elements of artificial neural networks, MIT press, 1997.

[29] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, In Proceedings of the IEEE, (1998) pages 2278–2324.

[30] G. Du, P. Zhang, Robotics and Computer-Integrated Manufacturing A novel human – manipulators interface using hybrid sensors with Kalman filter and particle filter, Robot. Comput. Integr. Manuf., 38 (2016) 93–101.

[31] I. Goodfellow, Y. Bengio , and A. Courville, Deep learning. MIT press, 2016.

[32] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, GPU Computing, Proc. IEEE, vol. 96 (2008) pp. 879–899.

[33] K. Sato, C. Young and D. Patterson, An in-depth look at Google's first Tensor Processing Unit (TPU), Google Cloud Big Data and Machine Learning Blog, 12 (2017).

[34] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed. Pearson, 2009.

[35] R. S. Sutton and A. G. Barto, Reinforcement learning : an introduction, 2nd ed. Cambridge, MA: Mit Press, 2017.

[36] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, ... , and D. Hassabis, Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815, (2017).

[37] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., 1st Edn., New York, NY, USA, 1994.

[38] R. Bellman, On the theory of dynamic programming, Proceedings of the National Academy of Sciences of the United States of America, 38(8) (1952) 716.

[39] C. J. C. H. Watkins, Learning from delayed rewards, 1989.

[40] L. J. Lin, Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine learning, 8(3-4) (1992) 293-321.

[41] H. Van Hasselt, A. Guez, and D. Silver, Deep reinforcement learning with double q-learning, In Proceedings of the AAAI conference on artificial intelligence, Vol. 30, No. 1, 2016.

[42] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, & N. Freitas, Dueling network architectures for deep reinforcement learning, In International conference on machine learning (pp. 1995-2003), 2016.

[43] L. Xie, S. Wang, A. Markham, and N. Trigoni, Towards monocular vision based obstacle avoidance through deep reinforcement learning. arXiv preprint arXiv:1706.09829, (2017).

[44] L. Joseph, Robot Operating System (ROS) for Absolute Beginners. Springer, 2018.

[45] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, ... and A. Y. Ng, A. Y. , ROS: an open-source Robot Operating System. In ICRA workshop on open source software, Vol. 3, No. 3.2 (2009), p. 5.

[46] V. DiLuoffo, W. R. Michalson, and B. Sunar, Robot Operating System 2: The need for a holistic security approach to robotic architectures, International Journal of Advanced Robotic Systems, 15(3) (2018), 1729881418770011.

[47] http://wiki.ros.org/Distributions. Automation Magazine, 2021.

[48] M. A. Haseeb, D. Ristić-Durrant, and A. Gräser, Long-range obstacle detection from a monocular camera, In Proceedings of the ACM Computer Science in Cars Symposium (CSCS), pp. 13-14, 2018, Munich, Germany.,

[49] J. Redmon, Darknet: Open Source Neural Networks in C. [Online]. Available: https://pjreddie.com/darknet/. [Accessed: 06-May-2021].,

[50] COCO dataset, https://arxiv.org/pdf/1405.0312.pdf

[51] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, ..., and D. Hassabis, Mastering the game of Go with deep neural networks and tree search, nature, 529(7587) (2016) 484-489.

ÖZGEÇMİŞ

Adı Soyadı: Koray ÖZDEMİR

Lisans: Sakarya Üniversitesi Bilgisayar ve Öğretim Teknolojileri Öğretmenliği, Ahmet Yesevi Üniversitesi Bilgisayar Mühendisliği

Yüksek Lisans: Ahmet Yesevi Üniversitesi, İngiliz Dili Eğitimi

TEZDEN TÜRETİLEN YAYINLAR

[1] K. Özdemir, A. Tuncer, Deep Reinforcement Learning Based Mobile Robot Navigation in Unknown Indoor Environments, International Conference on Interdisciplinary Applications of Artificial Intelligence (ICIDAAI), pp. 127–132, 2021, Yalova, Turkey