T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

SİMÜLASYON ORTAMINDA DERİN PEKİŞTİRMELİ ÖĞRENME İLE ROBOT KONTROLÜ

YÜKSEK LİSANS TEZİ

HÜSEYİN PULLU

TEZ DANIŞMANI

PROF. DR. CİHAN KARAKUZU

BİLECİK, 2024

10597855

T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

SİMÜLASYON ORTAMINDA DERİN PEKİŞTİRMELİ ÖĞRENME İLE ROBOT KONTROLÜ

YÜKSEK LİSANS TEZİ

HÜSEYİN PULLU

TEZ DANIŞMANI

PROF. DR. CİHAN KARAKUZU

BİLECİK, 2024

10597855

BEYAN

"Simülasyon Ortamında Derin Pekiştirmeli Öğrenme İle Robot Kontrolü" adlı yüksek lisans tezi hazırlık ve yazımı sırasında bilimsel araştırma ve etik kurallarına uyduğumu, başkalarının eserlerinden yararlandığım bölümlerde bilimsel kurallara uygun olarak atıfta bulunduğumu, kullandığım verilerde herhangi bir tahrifat yapmadığımı, tezin herhangi bir kısmının Bilecik Şeyh Edebali Üniversitesi veya başka bir üniversitede başka bir tez çalışması olarak sunulmadığını, aksinin tespit edileceği muhtemel durumlarda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

<table>
<tr><td colspan="4">Bu çalışmanın,
Bilimsel Araştırma Projeleri (BAP), TÜBİTAK veya benzeri kuruluşlarca desteklenmesi durumunda; projenin ve destekleyen kurumun adı proje numarası ile birlikte, ETİK KURUL onayı alınması durumunda ise ETİK KURUL tarih karar ve sayı bilgilerinin beyan edilmesi gerekmektedir.</td></tr>
<tr><td>DESTEK ALINMIŞTIR</td><td></td><td>DESTEK ALINMAMIŞTIR</td><td>X</td></tr>
<tr><td colspan="4">Destek alındı ise;</td></tr>
<tr><td colspan="4">Destekleyen kurum;</td></tr>
<tr><td colspan="2">Desteğin Türü</td><td colspan="2">Proje Numarası</td></tr>
<tr><td colspan="2">1- BAP (Bilimsel Araştırma Projesi)</td><td colspan="2"></td></tr>
<tr><td colspan="2">2- TÜBİTAK</td><td colspan="2"></td></tr>
<tr><td colspan="2">Diğer;…….....
……..</td><td colspan="2"></td></tr>
<tr><td colspan="4">ETİK KURUL onayı var ise;</td></tr>
<tr><td colspan="2">ETİK KURUL karar tarih/sayı:</td><td colspan="2">……/……
……</td></tr>
</table>

Hüseyin PULLU

..... /….. / 2024

……

ÖN SÖZ

Bu tez çalışmamı sahiplenerek her aşamasında desteğini eksik etmeyen, bilgi ve tecrübesi ile her daim yol gösteren değerli hocam Sayın Prof. Dr. Cihan KARAKUZU'ya katkı ve emekleri için teşekkürlerimi ve saygılarımı sunarım.

Savunma sınavı sırasında değerli jüri üyelerine ve danışman hocam Sayın Prof. Dr. Cihan KARAKUZU'ya çalışmamın son haline gelmesindeki değerli katkıları adına teşekkürlerimi ve saygılarımı sunarım.

Hayatımın her aşamasında yanımda olan, sonsuz sabrı ve hoşgörüsü ile bana destek olan sevgili eşime teşekkür ederim. Bu çalışmayı, kıymetli evlatlarım Ali Çağan'a ve Meva'ya ithaf ediyorum.

Hüseyin PULLU

2024

ÖZET

SİMÜLASYON ORTAMINDA DERİN PEKİŞTİRMELİ ÖĞRENME İLE ROBOT KONTROLÜ

Makine öğreniminin bir dalı olan Pekiştirmeli Öğrenme, bir ortamda ödülleri en üst düzeye çıkarmak için eylemleri seçmeye dayalı öğrenme yöntemidir. Doğada çokça gözlemlenen bu davranışsal öğrenme şekli, hakkında bilgi sahibi olunmayan ortamlarda otonom görevler için başarılı sonuçlar verir. Pekiştirmeli Öğrenme yöntemlerinden, Derin Pekiştirmeli Öğrenme ile otonom görevler için gezgin robotları eğitebiliriz.
Gezgin robotlar, eş zamanlı konum belirleme ve haritalama (SLAM) algoritmaları ile ortam haritasını çıkararak yol ve hareket planlama görevlerini yapabilirler. Ancak haritası çıkarılmamış ortamlarda hareket planlamak ve yapmak zorlu bir görevdir. Çalışmamızda, bu zorlukları aşmak için Derin Pekiştirmeli Öğrenme yöntemleri kullanılarak gezgin robotlar ile bilinmeyen ortamlarda belirlenen görevleri yerine getirmek amaçlanmıştır. Robot İşletim Sistemi ve benzetim ortamı olarak sıklıkla kullanılan, 3 boyutlu modeller ile çalışabilen Gazebo benzetim ortamı ile gerekli ortam oluşturulmuş ve TurtleBot3 gezgin robotu ile engellere çarpmadan bu ortamda belirlenen hedeflere ulaşmak amaçlanmıştır. Çalışmamızda, Pekiştirmeli Öğrenmenin sıklıkla kullanılan yöntemi Q Öğrenmenin sınırlamalarından dolayı, karmaşık problemler için bellek yerine sinir ağlarını kullanan Derin Pekiştirmeli Öğrenme yöntemlerinden Derin Q Ağları tercih edilmiştir. Gazebo'da oluşturulan ortamda TurtleBot3 gezgin robotu ile eğitim yapılmıştır. Pekiştirmeli öğrenmenin temel özelliklerinden, açgözlü yaklaşım politikasına göre başlangıç aşamasında rastgele yapılan hareketler yerini ilerleyen aşamalarda sinir ağı modelinin karar verdiği hareketlere bırakarak hedef noktayı art arda defalarca bulabilen ve bölüm boyunca topladığı ödülü artırabilen bir model oluşturulmuş ve bu modelin başarımı grafiklerle gösterilmiştir.

Anahtar Kelimeler: Derin pekiştirmeli öğrenme, otonom hareket, gezgin robot, robot işletim sistemi, Gazebo benzetim ortamı

ABSTRACT

ROBOT CONTROL WITH DEEP REINFORCEMENT LEARNING IN SIMULATION ENVIRONMENT

Reinforcement Learning, a branch of machine learning, is a method of learning based on choosing actions to maximize rewards in an environment. This form of behavioral learning, which is widely observed in nature, gives successful results for autonomous tasks in environments where there is no information about. We can train mobile robots for autonomous tasks with Deep Reinforcement Learning, one of the Reinforcement Learning methods. Mobile robots can perform path and movement planning tasks by mapping the environment with simultaneous positioning and mapping (SLAM) algorithms. However, operating in uncharted environments is a challenging task. In our study, we aimed to overcome these difficulties by using Deep Reinforcement Learning methods to perform specified tasks in unknown environments with mobile robots. The necessary environment was created with the Robot Operating System and the Gazebo simulation environment, which can work with the 3D models frequently used on simulation platform, and the TurtleBot3 mobile robot aimed to reach the determined goals in this environment without hitting obstacles. Due to the limitations of Q Learning, the frequently used method of reinforcement learning, Deep Q Networks, one of the Deep Reinforcement Learning methods that use neural networks instead of memory for complex problems, were preferred in our study. Training was conducted with the TurtleBot3 mobile robot in the environment formed in Gazebo. According to the greedy approach policy, which is one of the basic features of reinforcement learning, a model that can find the target point many times and increase the reward it collects throughout the episode, by leaving the movements made randomly at the beginning to the movements decided by the neural network model in the later stages, has been constituted and performance of this model has been showed with graphics.

Keywords: Deep reinforcement learning, autonomous navigation, mobile robot, robot operation system, Gazebo simulation environment.

İÇİNDEKİLER

Sayfa

TABLOLAR LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

GRAFİKLER LİSTESİ

Sayfa

KISALTMALAR VE SİMGELER LİSTESİ

CNN : Evrişimli Sinir Ağı (Convolutional Neural Network)

DP : Dinamik Programlama

DPÖ : Derin Pekiştirmeli Öğrenme (Deep Reinforcement Learning)

DNN : Derin Sinir Ağları (Deep Neural Networks)

DQN : Derin Q Ağları (Deep Q Networks)

IMU : Ataletsel Ölçü Birimi (Inertial Measurement Unit)

LIDAR : Lazer Algılama ve Uzaklık Belirleme (Light Detection and Ranging)

MC : Monte Carlo Yöntemi

MKS : Markov Karar Süreçleri (Markov Decision Process)

PÖ : Pekiştirmeli Öğrenme (Reinforcement Learning)

ROS : Robot İşletim Sistemi (Robot Operation System)

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

ZFÖ : Zamansal Fark Öğrenimi (Temporal Difference Learning)

$\boldsymbol{\alpha}$: Öğrenme oranı **(**Alfa**)**

$\boldsymbol{\gamma}$: İndirim oranı (Gama)

$\boldsymbol{\varepsilon}$: Keşif olasılığı (Epsilon)

$\boldsymbol{r}$: Ödül (Reward)

$\boldsymbol{s}$: Durum (State)

$\boldsymbol{a}$: Eylem (Action)

1. GİRİŞ

Gezgin robotlar, SLAM algoritmaları ile ortam haritasını çıkararak yol ve hareket planlama görevlerini yapabilirler. Yol planlama, bir robotu başlangıç konumundan hedef konuma hareket ettirirken, robotlar ve çevredeki engeller arasında çarpışmayı önlemek için gerçekleştirilen görevdir (Karakuzu ve Babuska, 2014,2021). Ancak haritası çıkarılmamış ortamlarda hareket yapmak zorlu bir görevdir. Çalışmamızda, bu zorluğu aşmak için Derin Pekiştirmeli Öğrenme (DPÖ) kullanılarak, bilinmeyen bir ortamda, belirli görevleri gezgin robot ile otonom olarak yerine getirmek amaçlanmıştır.

Pekiştirmeli Öğrenme (PÖ), makine öğrenimi yöntemlerinden biridir. PÖ'de hareketi yapan unsur (ajan), çevre ile etkileşime girerek ortamı keşfeder ve mümkün olan en yüksek toplam ödülü elde etmek amacıyla bu etkileşimleri değerlendirir. PÖ, canlıların çevreden olumlu veya olumsuz geri bildirim alarak öğrenmek için kullandıkları deneme yanılma yöntemini taklit eder.

Robot teknolojileri alanında PÖ, mühendisliği zor ve karmaşık olan davranışların tasarımı için bir çerçeve sunar (Zamora vd., 2016). Bu alandaki zorluk, kullanılan makine öğrenimi tekniklerini doğrulamak ve gerçek bir senaryoda uygulamak için ortam oluşturmaktır. Çalışmamızda bu problemin üstesinden gelmek için, robotik uygulamalarında yazılım geliştirme amacıyla kullanılan Robot İşletim Sistemi (ROS) ve 3 boyutlu modelleme ve benzetim aracı olan Gazebo ile gerekli ortam oluşturulmuştur.

TurtleBot3 gezgin robotunun Gazebo benzetim ortamında engellerden kaçınarak hedef noktaları bulmasını sağlayacak bir denetim modeli DPÖ yöntemlerinden Derin Q Ağları (DQN) ile eğitim yapılarak elde edilmiştir. PÖ'nin iki temel unsuru ajan ve çevredir. DQN ajanı, TurtleBot3'ü engellere çarpmadan hedef noktaya ulaştırmak için, hedefe yaklaştıkça pozitif bir ödül değeri, hedef noktadan uzaklaştıkça da negatif bir ödül değeri yani ceza verir. Burada ajanın amacı birikimli ödül değerini en yükseğe çıkarmaktır.

Çalışmanın içeriği bölümlere ayrılarak şu şekilde düzenlenmiştir. Bölüm 2'de, kullanılan yöntemler hakkında bilgi verilmiştir. Bölüm 3'te ROS, Gazebo, DQN bağlantısından ve uygulama da kullanılan çerçeve yapıdan bahsedilmiştir. Bölüm 4'de DQN modeli ile yapılan çalışmanın parametreleri ve eğitilen modelin sonuçları paylaşılarak analiz edilmiştir. Son olarak, Bölüm 5'de eğitilen model ile yapılan test işleminin sonuçları paylaşılarak yorumlanmıştır.

2. ÖN BİLGİ

2.1. Literatür Araştırması

Çıkarım yapılacak giriş verileri ya da etiketlenmiş eğitim verisi bulunmayan problemlerin PÖ yöntemleri ile çözümü yapılabilmektedir. Son yıllarda Türkiye'de yapılan tezler incelendiğinde PÖ ile robot kontrolü, otonom sürüş, finans yönetimi, enerji sistemleri optimizasyonu, ağ trafik optimizasyonu vb. gibi alanlarda çalışmalar yapıldığı görülmektedir. Farklı alanlarda birçok karar verme probleminin PÖ ile çözümünün yapılabildiği anlaşılmaktadır. Özellikle otonom kontrol problemlerinde DPÖ yöntemlerinin kullanımı dikkati çekmektedir. DPÖ yöntemleri, ilk olarak Derin Q Ağları (Mnih vd., 2013) ile gerçek anlamda başarısını kanıtlayarak Arcade ortamındaki bazı oyunlarda insan düzeyinde, bazılarında ise insan düzeyinin de üzerinde başarı göstermiştir.

DPÖ algoritmalarının başarısı, gerçek dünya ortamındaki kontrol problemlerinin, yüksek boyutlu durum ve eylem uzayında politika güdümlü ajanların optimize edilmesiyle doğal olarak çözülebileceğini kanıtlamaktadır. Görüş kontrol sistemlerinde durum uzayları genellikle ham görüntülerden elde edilen görüntü özellikleriyle temsil edilir (Wang vd., 2018). Otonom sürüşe ulaşmak için araştırmacılar hem sensörlerden hem de görüş algoritmalarından gelen bilgilerden yararlanmaya çalışmaktadırlar. Otonom sürüşe ulaşmanın basit bir yolu, Lidar ve Atalet Ölçüm Birimi (IMU) gibi hassas ve sağlam donanımlar ve sensörler kullanarak ortam bilgilerini yakalamaktır (Wang vd., 2018). Otonom sürüş politikasının fiziksel hasar olmadan öğrenilmesi için çok sayıda benzetim ortamı tasarlanmıştır. Literatürde yapılan çalışmalardan bazıları incelenerek bulgular aşağıda verilmiştir.

Wang ve diğerleri, otonom sürüş ile PÖ arasındaki boşluğu kapatmak amacıyla, ajanı Açık Yarış Arabası Simülatöründe (TORCS) eğitmek için Derin Deterministik Politika Eğimi (DDPG) algoritmasını kullanmışlardır. Özellikle TORCS'dan uygun sensör bilgisini girdi olarak seçmiş ve eylem alanlarını sürekli alanda tanımlamışlardır. Ajanın simülatörde hızlı hareket edebildiğini ve güvenli sürüşü sağlayabildiğini göstermişlerdir (Wang vd., 2018).

Sung ve diğerleri, TurtleBot3 gezgin robotu ile Gazebo benzetim ortamında DQN algoritmasını kullanarak, gezgin robot navigasyonu için modelden bağımsız PÖ yöntemlerini incelemişlerdir (Sung vd., 2018).

Alvarez, TurtleBot3 gezgin robotu ile Gazebo benzetim ortamında otonom sürüş için LIDAR ve IMU sensörü verilerini kullanmış ve ileriye dönük parametrelere sahip DQN modelinin standart DQN modelinden daha iyi öğrenebileceğini iddia etmiştir (Alvarez, 2019).

Tsai ve diğerleri, TurtleBot3 gezgin robotu ile Gazebo benzetim ortamında otonom sürüş için LIDAR ve IMU sensörü verilerini kullanmışlardır. DPÖ yöntemlerinden Derin Deterministik Politika Eğitimi (DDPG) ile Derin Q Ağları (DQN) yöntemlerini kullanarak sonuçları karşılaştırmışlardır (Tsai vd., 2022).

Bu çalışmasının amacı, DPÖ yöntemleri ile gezgin robotun benzetim ortamında otonom hedef bulma görevlerini yapmasını sağlamaktır. TurtleBot3 gezgin robotu ile Gazebo benzetim ortamında, LIDAR ve IMU sensörü verileri kullanılarak rastgele yer değiştiren hedefleri otonom olarak bulma görevi gerçekleştirilmeye çalışılmıştır.

2.2. Otonom Sürüş

Otonom sürüş, bilgisayarlı görme ve kontrol sistemlerinde aktif bir araştırma alanıdır (Memon vd., 2016). Otonom sürüş için, ortamlardan gelen durum alanları ve girdi görüntüleri, dinamik olarak değişebilen ve öngörülemeyen şekilde davranabilen, arka plan ve nesneler içerir. Bunlar, nesne tespiti, sahneyi anlama, derinlik tahmini gibi birçok zor görme görevini içerir. Daha da önemlisi denetlenen robotun ya da ajanın bu tür zor senaryolarda nesnelere çarpmamak ve güvende kalmak için doğru ve hızlı hareket etmesi gerekir.

DPÖ algoritmasının başarısı; gerçek dünya ortamındaki denetim problemlerinin yüksek boyutlu durum ve eylem uzayında, politika güdümlü temsilcilerin optimize edilmesiyle çözülebileceğini kanıtlamaktadır. Görüş denetim sistemlerinde durum uzayları, genellikle ham görüntülerden elde edilen görüntü özellikleriyle temsil edilir. Otonom sürüşe ulaşmanın basit bir yolu, Lidar ve IMU gibi hassas ve sağlam donanımlar ve sensörler kullanarak ortam bilgilerini yakalamaktır. Bu donanım sistemleri, 3D bilgileri hassas bir şekilde yeniden yapılandırabilir ve ardından PÖ'yi kullanarak aracın çarpışma olmadan akıllı navigasyon elde etmesine yardımcı olabilir.

Araştırmacılar otonom sürüşe ulaşmak için hem sensörlerden hem de görüntü işleme algoritmalarından gelen bilgilerden yararlanmaya çalışıyorlar. Akıllı yol bulma politikasının fiziksel hasar olmadan öğrenilmesi için gerçek dünya ortamını taklit edebilen benzetim ortamlarından yararlanılır (Wang vd., 2018).

2.3. Robot İşletim Sistemi

Robot İşletim Sistemi (ROS), geliştirici ile robot arasındaki iletişimi sağlayan ve bu iletişimi standartlaştıran açık kaynak kodlu bir ara yüz çerçevesidir. Robot sistemleri için donanım ile yazılım katmanının ayrılması, donanım seviyesinde cihaz kontrolü, paket yönetimi,

uç noktalar arasında mesajlaşmanın sağlanması konularını robot sistemleri için gerçekleştirir (ROS Introduction, 2023). ROS, isminde geçse de aslında bir işletim sistemi değildir.

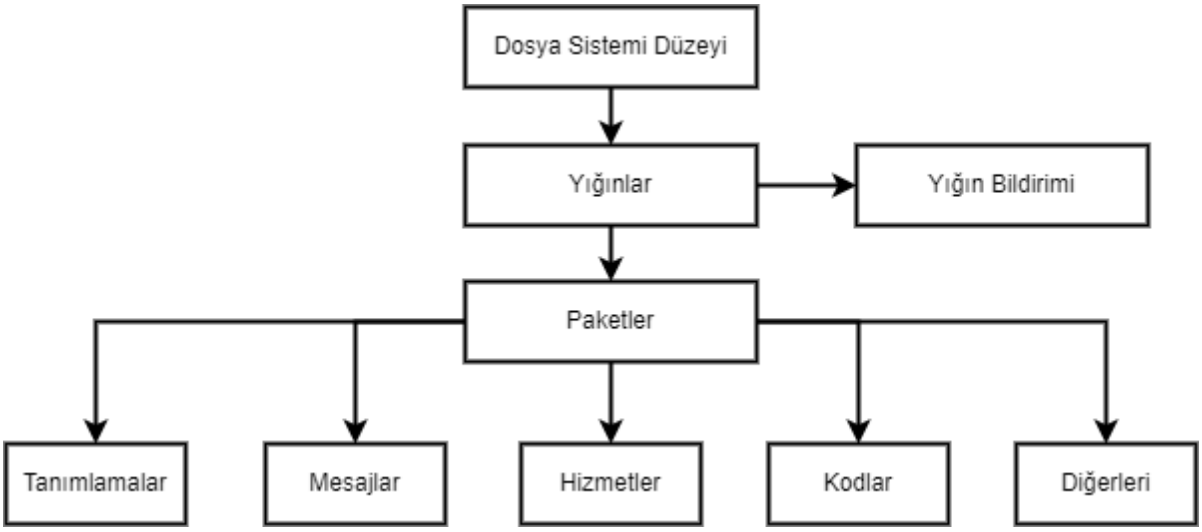
ROS ilk olarak 2007 yılından kısa bir süre önce Standford AI Robot desteğinde Standford Üniversitesinde başlamış daha sonra çalışmalara 2007 yılından itibaren Willow Garage adlı teknoloji şirketi üzerinden devam edilmiştir. ROS'un hedefi, herhangi bir robotta kullanabileceğiniz robotik yazılım geliştirme için standart oluşturmaktır. Bu, robotik projelerinde kullanılan çerçeveyi yeniden kullanılabilir ve farklı geliştiriciler için anlaşılabilir hale getirir.

ROS çerçevesinde Python, C++ ve Lisp dilleri ile uygulama geliştirilebilir. Linux Ubuntu üzerine tasarlanmıştır. ROS2 versiyonuyla Ubuntu, Debian, MacOS, Windows gibi farklı işletim sistemlerini de desteklemektedir. Uygulama da genellikle robot gövdesine entegre edilmiş Raspberry Pi gibi bir mini bilgisayar üzerine Ubuntu ve gerekli ROS paketleri kurularak çalıştırılır.

ROS ile donanım yazılımdan tamamen soyutlanabilir. Sadece ROS API'sini bilerek robot programlanabilir. ROS API, belirli bir robotun donanımına, yani sensörlere ve eyleyicilere erişim sağlamak için sağladığı konuların (topic), hizmetlerin (service), eylem sunucularının (action servers) ve mesajların (messages) bütünüdür. Modern robot üreticileri kullanıma hazır ROS API'leri yayınlarlar. Eğer robotunuz ROS'u desteklemiyorsa, robotu ROS'a dönüştürmek gerekir. ROSify'a geçmek, robotunuzu ROS ile çalışacak şekilde uyarlamak anlamına gelir. Bir robotu ROSify etmek için sensör verilerini sağlayan elektroniklerle nasıl iletişim kurulacağı veya robotun motorlarına vb. donanımlara nasıl erişileceği bilinmelidir (The Construct, 2023).

2.3.1. ROS dosya sistemi

ROS açık kaynaklı bir geliştirme ve dağıtıma sahip olduğundan yapısı paketlerden oluşmaktadır. Paketler de daha alt birimlerden oluşur. Şekil 2.1'de ROS dosya sistemi yapısı görülmektedir (Martinez ve Fernandez, 2013).



Şekil 2.1. ROS dosya sistemi yapısı

Paketler: Paketler ROS'un atomik seviyesini oluşturur. Bir paket, ROS içerisinde program oluşturmak için gereken minimum yapıya sahiptir.

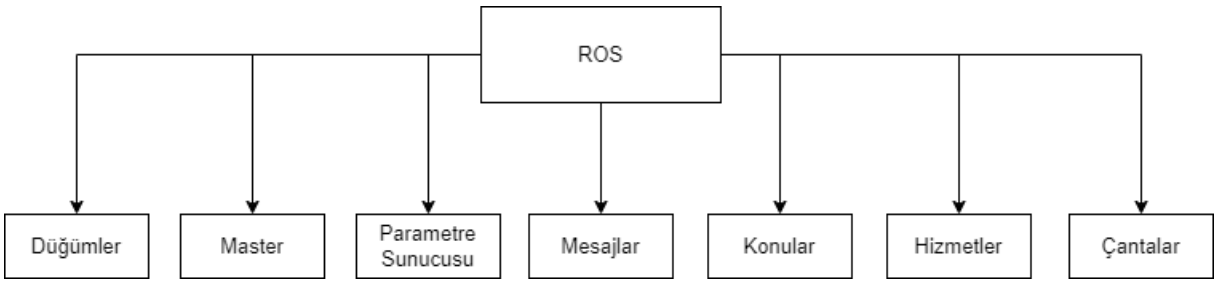
Yığınlar: Ortak bir hedef için çalışan işlevlere sahip birkaç paket yığını oluşturur. Örneğin gezinme yığını gibi.

Mesajlar: Düğümler arasında haberleşme mesajlar ile sağlanır. Örneğin motorları kontrol eden düğüme hareket bilgisi mesajlar ile iletilir.

Hizmetler: ROS'da ki hizmetler için istek ve yanıt veri yapılarını tanımlar

2.3.2. ROS işlevsel yapısı

ROS, tüm süreçlerin birbirine bağlı olduğu bir ağ oluşturur. Şekil 2.2'de bu yapı sembolize edilmiştir. Sistemde ki herhangi bir düğüm bu ağa erişebilir, diğer düğümlerle etkileşime girebilir, gönderdikleri bilgileri görebilir ve ağa veri aktarabilir (Martinez ve Fernandez, 2013).



Şekil 2.2. ROS işlevsel yapısı

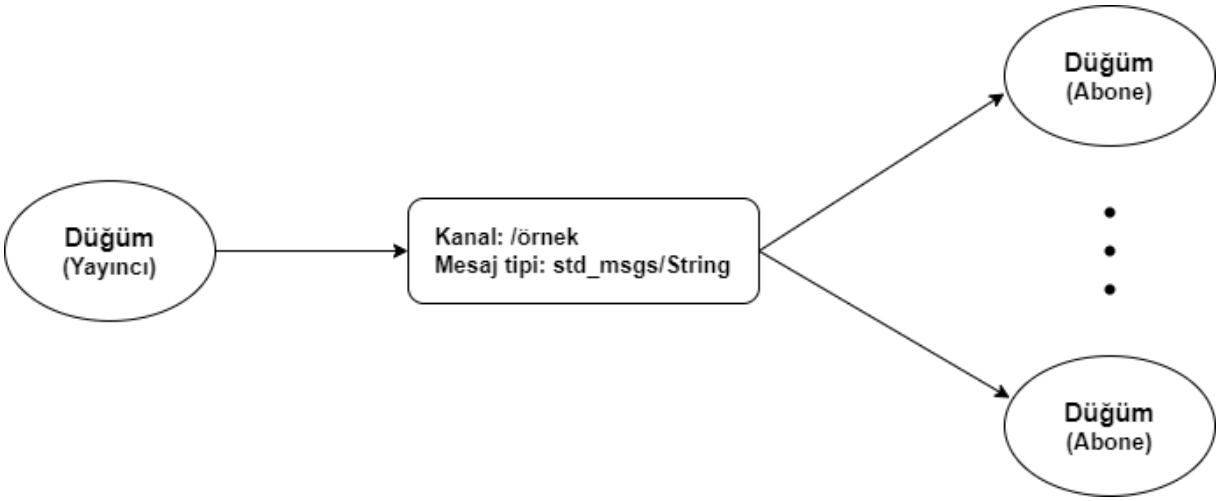
Düğümler: İşlem yapan birimler düğümlerdir. Düğümler ROS ağına (master) bağlıdır. Bu ağ sayesinde düğümler arasında veri alışverişi yapılabilir. Örneğin LIDAR verilerini yayınlayan bir düğüm, motorları kontrol ederek hareketi sağlayan bir düğüm tanımlanabilir.

Master: Düğümler arasında veri alışverişi ana (master) düğüm tarafından yürütülmektedir. Master ROS ağını yönetir. Bu düğüm ROS sistemindeki düğümleri adlandırma ve kayıt hizmeti sağlamaktadır (ROS Introduction, 2023).

Parametre Sunucusu: Verilerin merkezi bir konumda anahtarlar kullanılarak saklanmasına olanak tanır.

Mesajlar: Düğümler birbirleriyle mesajlar ile iletişim kurar.

Konular: Düğümler birbirleri ile konu (topic) adı verilen dijital kanallar üzerinden mesajlar ile haberleşebilir. ROS haberleşme sisteminde, yayıncı - abone (publish-subscribe) sistem modeli kullanılmaktadır. Tek yönlü veri akışı vardır. Şekil 2.3'de konular üzerinden mesaj aktarımı sembolize edilmiştir.



Şekil 2.3. Düğümler arasında konular üzerinden mesaj aktarımı

Hizmetler: Bir istek ve yanıt sistemiyle düğümlerin çift yönlü haberleşmesini sağlar.

Çantalar: ROS mesaj verilerini kaydetmek ve oynatmak için kullanılır. Örneğin LIDAR data verileri gibi.

2.3.3. Roscore

ROS merkezi bir sistemdir, düğümler için her zaman bir ana düğüme ihtiyaç vardır ve diğer düğümlerden önce çalıştırılmalıdır. Ana düğüm "roscore" olarak isimlendirilir. Roscore

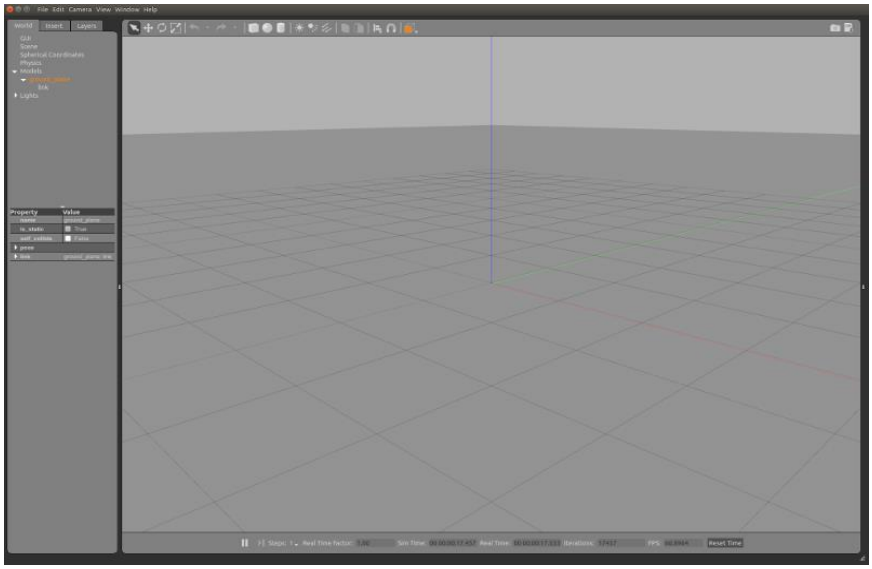
düğümlerin birbirlerine mesaj iletebilmeleri için bağlantı bilgilerini sağlar. *roscore* komutu ile ana düğüm başlatılır ve çalışma boyunca kapatılmamalıdır. Her düğüm, yayınladığı mesaj akışlarının ve abone olmak istediği akışların ayrıntılarını kaydetmek için başlangıçta roscore'a bağlanır. Yeni bir düğüm ortaya çıktığında, roscore ona, aynı mesaj konularını yayınlayan ve bunlara abone olan diğer düğümlerle doğrudan eşler arası bağlantı kurması için ihtiyaç duyduğu bilgiyi sağlar (Quigley vd., 2015).

2.4. Benzetim Ortamı

Gazebo, karmaşık iç ve dış ortamlardaki robot varlıklarını doğru ve verimli bir şekilde benzetme yeteneğine sahip, 3 boyutlu dinamik benzetim ortamıdır. Oyun motorlarına benzer olmasına rağmen Gazebo, hem kullanıcılar hem de programlar için çok daha yüksek doğrulukta fizik simülasyonu, sensör paketi ve arayüzler sunar (Gazebo Tutorial, 2023a).

Gazebo, tek veya çoklu robot sistemlerini gerçekçi bir benzetim ortamında modellemek, test etmek ve görselleştirmek için kullanılır (Koenig ve Howard, 2004). ROS çerçevesi ile senkron çalışır ve kullanımı kolaydır. Yapay zeka ve robotik araştırmacıları tarafından sıklıkla tercih edilir.

Gazebo, fizik tabanlı benzetim motoru ile robotik alanında gerçekçi bir deneyim sunar (Koenig ve Howard, 2004). Robotik sistemlerin gerçek dünya ortamında nasıl davranacağını tahmin etmeyi sağlar. Etkileyici ve görsel anlamda zengin bir benzetim sunmak için güçlü grafik ve görselleştirme motoruna sahiptir. Çeşitli sensör türlerini entegre etme ve benzetme yeteneğine sahiptir. Kameralar, hızölçerler, lazer algılama ve uzaklık belirleme sensörü (LIDAR) ve diğer sensörler gibi farklı algılayıcıları kullanarak, robotların ortamı algılama ve tepki verme davranışlarını değerlendirmek mümkündür. Gerçek dünya ortamlarını taklit etmek için benzetim ve kontrol arasında eşleme sağlar. Gerçek robotların kontrol algoritmalarını benzetim ortamında test etmek için kullanışlıdır. Aynı anda birden fazla robotun benzetimini destekler. Şekil 2.4'de boş bir Gazebo benzetim ortamı görülmektedir.



Şekil 2.4. Gazebo benzetim ortamı

Gazebo'nun temel özellikleri şunlardır:

- Çoklu fizik motorları,
- Robot modelleri ve ortamlarından oluşan zengin bir kütüphane,
- Çok çeşitli sensörler,
- Kullanışlı, programlı ve grafiksel arayüzler

Gazebo'nun tipik kullanımları şunları içerir:

- Robotik algoritmaları test etmek,
- Robot tasarlamak,
- Gerçekçi senaryolarla regresyon testi yapmak

2.5. Gezgin Robot

Gezgin robotların temel özelliği, robotun ortamda serbestçe hareket etmesini sağlayan bir gezme tabanının bulunmasıdır. Manipülatörlerin aksine bu tür robotlar çoğunlukla kapsamlı, otonom hareket yeteneklerinin gerekli olduğu servis uygulamalarında kullanılır. Mekanik açıdan bakıldığında gezgin robot, hareket sistemi ile donatılmış bir veya daha fazla katı gövdeden oluşur (Siciliano vd., 2009).

TurtleBot3, taşınabilirliği ve düşük maliyeti ile öne çıkan robot platformudur (Amsters ve Slaets, 2020). ROS platformuyla uyumludur ve birçok farklı uygulama alanında kullanılabilir. Özellikle robotik sistemler ile yapay zeka alanında öğrenim ve araştırma faaliyetleri için tasarlanmıştır. Çeşitli aksesuarları ve sensörleri ile özelleştirilebilir. TurtleBot3 Burger modeli, LIDAR algılayıcısı ve ataletsel ölçüm birimi (IMU) ile çevre ve konum

bilgilerini yayınlar (TurtleBot3, 2023). IMU, ivmeölçerler, jiroskoplar ve bazen manyetometrelerin kombinasyonunu kullanarak bir cismin özgül kuvvetini, açısal hızını ve bazen cismin yönelimini ölçen ve raporlayan elektronik bir cihazdır (IMU, 2023). LIDAR, lazer darbeleri kullanılarak bir nesne veya bir yüzeyin uzaklığını anlamaya yarayan teknolojidir (LIDAR, 2023). Şekil 2.5'de TurtleBot3 Burger gezgin robotu görülmektedir.

Şekil 2.5. Turtlebot3 Burger robotu

2.6. Makine Öğrenmesi

Makine öğrenimi, verilerdeki kalıpları tespit edebilen ve bunları bazı görevleri gerçekleştirmek için kullanabilen otomatik yöntemler sağlar (Christopher, 2006; Murphy, 2012). Üç tür makine öğrenimi görevi düşünülebilir (François-Lavet vd., 2018):

Denetimli öğrenme: Etiketlenmiş eğitim verilerinden bir sınıflandırma veya regresyon çıkarma görevidir.

Denetimsiz öğrenme: Etiketlenmiş yanıtlar olmadan giriş verilerinden oluşan veri kümelerinden çıkarımlar yapma görevidir.

Pekiştirmeli Öğrenme: Kümülatif ödülleri en üst düzeye çıkarmak için temsilcilerin bir ortamda eylem dizilerini nasıl gerçekleştirmesi gerektiğini öğrenme görevidir.

Bu makine öğrenimi görevlerini çözmek için fonksiyon yaklaşım araçları fikri, makine öğreniminin merkezinde yer alır. Birçok farklı türde fonksiyon yaklaşıklayıcısı mevcuttur: Doğrusal Modeller, SVM'ler, Karar Ağacı, Gauss süreçleri, Derin Öğrenme vb.

2.7. Derin Öğrenme

Son yıllarda, özellikle derin öğrenme alanındaki gelişmeler nedeniyle makine öğrenimi, zaman serileri, görüntüler ve videolar gibi yüksek boyutlu verilerden öğrenmede çarpıcı

gelişmeler kaydetti. Bu iyileştirmeler aşağıdaki hususlara bağlanabilir (François-Lavet vd., 2018):

• GPU'ların ve dağıtılmış hesaplamanın kullanımıyla hesaplama gücünün katlanarak artması,

• Derin Öğrenme'deki metodolojik atılımlar,

• Tensorflow gibi büyüyen bir yazılım ekosistemi ve ImageNet gibi veri kümeleri.

Tüm bu yönler tamamlayıcıdır ve son yıllarda Derin Öğrenme'nin gelişimi için verimli bir döngüye yol açmıştır.

2.8. Pekiştirmeli Öğrenme

PÖ, amaca ulaşmak için yapılması gerekenleri öğrenen bir makine öğrenmesi yöntemidir (Sutton ve Barto, 2018). PÖ'de ajan (agent) adı verilen öğrenen unsur karşılaştığı durumlarda bir eylem gerçekleştirir ve bunun karşılığında da sayısal bir ödül değeri alır. Ajanın amacı bu ödül değerini maksimuma çıkartmaktır. Ajan çevre ile etkileşime girerek deneme yanılma yöntemiyle çevreyi keşfeder. PÖ'nün problemi çözme yöntemi keşif (exploration) ve sömürüye (exploitation) dayanır. Ajanın edindiği tecrübeye dayalı eylemlerine sömürü, rastgele yaptığı eylemlere keşif denir. Ajan, keşif ve sömürüye dayalı bir denge üzerinde çalışır.

PÖ, model bilinmediğinde en uygun eylem politikasının nasıl öğrenileceği üzerine kurulu bir makine öğrenmesi yöntemidir. Ajan, uygun bir algoritma aracılığıyla optimal bir politika üretmek üzere işlenebilecek bilgiyi elde etmek için çevresi ile doğrudan etkileşime girmelidir (Kaelbling vd., 1996). PÖ gibi karar verme durumlarını modellemek için Markov Karar Süreçleri matematiksel bir çerçeve sağlar.

2.8.1. Markov Karar Süreci

Markov Karar Süreci (MKS), Rus matematikçi Andrey Andreyevich Markov tarafından ortaya atılmış olan stokastik süreçlere ilişkin çok yaygın uygulama alanı bulunan bir kavramdır. MKS, Markov Zincirinin bir uzantısıdır.

Stokastik planlama problemleri, robot denetiminin öğrenilmesi ve oyun oynama problemleri gibi birçok problem MKS ile başarıyla modellenmiştir (Otterlo ve Wiering, 2012). Yapay zekâ planlaması, yöneylem araştırması alanında yapılan araştırmalar (Winston, 1991) ve denetim kuramı (Bertsekas, 1996) arasında pek çok bağlantı vardır, çünkü bu alanlarda sıralı karar verme üzerine yapılan çoğu çalışma MKS'lerin örnekleri olarak görülebilir.

Bazen bir problemin tespiti, bir alanın geliştirilmesinde çözüm yöntemlerinin keşfedilmesinden daha önemli bir adımdır (Sutton, 1997). MKS zihinsel süreçlerin modellenmesinde problemi çözüme götüren temel çerçeve durumundadır.

MKS'ler, stokastik optimal denetim çalışmasından doğmuştur (Bellman, 1957) ve o zamandan beri bu alandaki temel problem olarak kalmıştır. 1980'lerde ve 1990'larda, tam olarak bilinmeyen MKS'ler, PÖ için doğal bir problem formülasyonu olarak yavaş yavaş kabul edilmiştir (Witten, 1977; Watkins vd., 1989; Sutton ve Barto, 2018). Robotik, yapay yaşam ve evrimsel yöntemler MKS terimleriyle iyi bir şekilde ifade edilmiştir (Sutton, 1997).

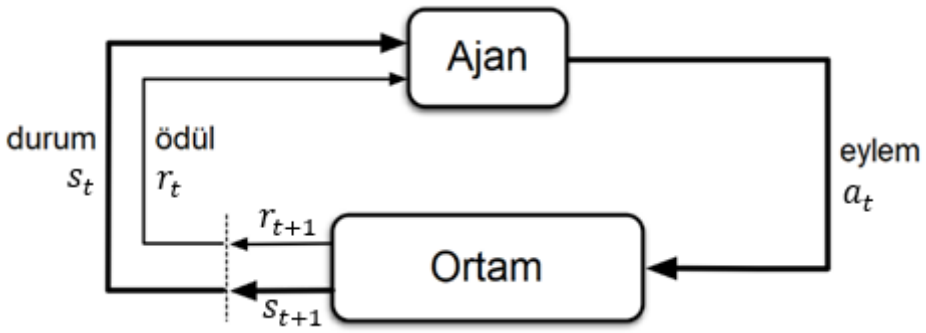
MKS, belirsizlik altında sıralı karar almaya yönelik bir modeldir. Hem mevcut kararların kısa vadeli sonuçlarını hem de gelecekte ki karar alma fırsatlarını dikkate alır (Kallenberg, 2011).

Durum geçiş fonksiyonu, ortamın bir sonraki durumunu, mevcut durumun ve ajanın eyleminin bir fonksiyonu olarak; olasılıksal olarak belirtir. Ödül işlevi, mevcut durum ve eylemin bir fonksiyonu olarak beklenen anlık ödülü belirtir. Durum geçişleri önceki çevre durumlarından veya ajan eylemlerinden bağımsızsa model Markov'dur. MKS modellerine referanslar (Bellman, 1957; Bertsekas, 1987; Howard, 1960; Puterman, 1994) (Kaelbling vd., 1996) olarak verilebilir.

2.8.2. Ajan-çevre arayüzü

Bir MKS, Şekil 2.6'da ki gibi etkileşim halinde olan bir ajan ve onun ortamından oluşur. Ajan öğrenen ve karar veren yapıdır. Ayrık zaman adımları dizisinin her birinde (t = 1, 2, 3, …) ajan ortamın durumunu s_t algılar ve bir eylem a_t seçerek hareketini yapar. Eyleme yanıt olarak, ortam yeni bir duruma s_{t+1} dönüşür ve skaler bir ödül r_{t+1} döndürür. Bu süreç aşağıdaki gösterim ile sembolize edilebilir.

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$



Şekil 2.6. Markov karar süreçlerinde ajan ve çevre etkileşimi (Sutton ve Barto, 2018)

Sonlu MKS'lerde durumlar ve eylemler sonlu kümelerden seçilir. Bu durumda çevre, bir (a) eylemi verildiğinde, bir (s) durumundan bir sonraki (s') durumuna olası her geçiş için, geçiş olasılıkları $P(s, a, s')$ ve beklenen ödüller $R(s, a, s')$ ile sembolize edilir.

Dikkat edilirse s_t (durum) ve r_t (ödül) sadece bir önceki durum ve eyleme bağlıdır. Bu etkileşimin geçmiş durumlardan sadece bir önceki duruma bağlı olması, durumların geleceğe etki edebilecek tüm geçmiş ajan-çevre etkileşimlerini kapsamasındandır. Durumların bu özelliğine Markov Özelliği denir (Sutton ve Barto, 2018). Bunun pratik uygulamalarda çok büyük faydası vardır. MKS ile modelleme yaptığımızda, anlık karar veren algoritmalar geliştirirken geçmişe yönelik bilgi depolamadığımız için büyük bellek alanları kullanımına gerek kalmaz. Robot yol planlama, hedef takibi, nesne takibi gibi gerçek zamanlı gömülü ortamlarda çalışan algoritmalarda; MKS çok etkin bir şekilde kullanılır.

Ajan her durumda eylem uzayından bir eylem seçer $(a \in A)$. Bu seçim belli bir politikaya göre olur. Buna ajan politikası denir ve π_t ile gösterilir; burada $\pi_t(s, a)$, $s_t = s$ durumunda $a_t = a$ olasılığıdır. Ajanın amacı uzun vadede aldığı kümülatif ödül miktarını maksimuma çıkarmaktır (Sutton ve Barto, 2018: 53). Kümülatif ödül getirisi denklem (2.1)'deki gibi ifade edilebilir.

$$G_t = R_t + R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \qquad (2.1)$$

Ödül hesaplanırken ihtiyaç duyulan ek bir kavram ise indirim faktörü (discount factor) dür. İndirim oranı kullanılarak azaltılan ödüllerin toplamı, indirimli getiriyi tanımlamaktadır. İndirim katsayısı eklenen kümülatif ödül getirisi, denklem (2.2)'de açıkça görülmektedir. İndirim oranı γ (gama) ile gösterilmekte olup, 0 ile 1 arasında bir değer almaktadır (Sutton ve Barto, 2018: 55).

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \qquad (2.2)$$

İndirim oranı, gelecekteki ödüllerin karar anındaki önemini belirlemektedir. 0 olması durumunda ajan o andaki en yüksek ödülü alabilecek eylemi seçme eğilimindedir, parametre 1′e yaklaştığı ölçüde ise gelecekteki ödülleri çok daha güçlü bir şekilde hesaba katarak eylemlerini seçmektedir. İlk başlarda alınacak ödüllerin gerçekleşme olasılığı daha yüksektir, çünkü uzun vadeli gelecekteki ödüllerden daha öngörülebilirdir. Bu oranın belirlenmesi ajanın öğrenimini doğrudan etkilemektedir. Genellikle 0,95 gibi 1'e yakın bir değer seçilerek uzun vadeli yüksek getiri hedeflenir.

2.8.3. Politika ve karar verme

PÖ'de amaç beklenen toplam ödülü en üst düzeye çıkarmaktır. Peki, bunu sağlayacak eylemleri seçebilen PÖ ajanını nasıl oluşturacağız?

Modelsiz PÖ yöntemleri için iki ana yaklaşım benimsenmiştir: Politika Tabanlı Yöntemler ve Değer Tabanlı Yöntemler. Politika Tabanlı Yöntem'de (Şekil 2.7 (a)), deterministtik ortamlarda doğrudan ajana, mevcut durum göz önüne alındığında hangi eylemi gerçekleştireceği öğretilir. Stokastik ortamlarda eylem seçimi olasılık değerleri olarak ifade edilir. Değer Tabanlı Yöntem'de (Şekil 2.7 (b)), ajana durum değerlerini öğrenmek ve ardından daha değerli durumları takip eden eylemleri seçmek öğretilir.

Şekil 2.7. (a) Politika tabanlı yöntem (sol), (b) değer tabanlı yöntem (sağ)

2.8.4. Politikalar ve değer fonksiyonları

PÖ algoritmalarının neredeyse tamamı, durumların ajan için ne kadar önemli olduğunu gösteren değer fonksiyonu tahminini kullanır. Değer fonksiyonları, hangi eylemlerin seçileceğini olasılıksal olarak belirleyen politikalar vasıtası ile hesaplanmaktadır. Politika (π), durum-eylem eşlemesini $\pi(a|s)$ ifade eder. Mevcut durumda seçilecek eylem kararını belirler.

Politikaya bağlı olarak atılan adımlar ile elde edilen durumlar ve ödüllerden, durum değer ve eylem değer fonksiyonları tahmin edilmektedir. Durum değer fonksiyonu denklem (2.3)'de eylem değer fonksiyonu ise denklem (2.4)'de gösterilmiştir (Sutton ve Barto, 2018).

$$v_\pi(s) = E_\pi[G_t|s_t = s] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|s_t = s\right] \tag{2.3}$$

$$q_\pi(s|a) = E_\pi[G_t|s_t = s, a_t = a] = E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|s_t = s, a_t = a\right] \tag{2.4}$$

Durum-değer fonksiyonu s durumunda iken π politikası takip edildiğinde s durumunun beklenen değerini verir. Eylem-değer fonksiyonu ise s durumunda iken a eylemi π politikası kullanılarak seçildiğinde durum-eylem çiftinin beklenen değerini verir. İdealde π politikasına bağlı olarak erişilen herhangi bir durumdaki getiri, durum değer fonksiyonunun tahminine; seçilen eylem sonrasında hesaplanan getiri de, eylem değer fonksiyonunun tahminlemesine eşit olmalıdır.

Denklem (2.5), Bellman eşitliğidir (Bellman, 1957). (s) durumunun değeri ve (s)'den erişilebilecek durumların değeri arasındaki ilişkiyi ifade eder. Bellman denklemi tüm olasılıkların ortalamasını alır ve her birini gerçekleşme olasılığına göre ağırlıklandırır. Başlangıç durumunun değerinin beklenen sonraki durumun (indirimli) değerine ve yol boyunca beklenen ödüle eşit olması gerektiğini belirtir (Sutton ve Barto, 2018).

$$\sum_{a} \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V_{\pi}(s')] \tag{2.5}$$

2.8.5. Optimal politika ve optimal değer fonksiyonları

En uygun politika, en uygun durum-değer ve eylem-değer fonksiyonlarına sahip olmaktadır. En uygun politika $\pi*$ ile, en uygun durum-değer fonksiyonu $v*$ ile, en uygun eylem-değer fonksiyonu ise $q*$ ile gösterilmektedir. $v*$ durum-değer fonksiyonu için beklenen getiri hesabı eşitlik (2.6)'da verilen Bellman uygunluk eşitliğidir. Aynı eşitlik $q*$ eylem-değer fonksiyonu için de geçerlidir ve eşitlik (2.7)'de verilmiştir.

$$v_*(s) = \max_{a} \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')] \tag{2.6}$$

$$q_*(s,a) = \max_{a} \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_{a'} q_*(s',a')\right] \tag{2.7}$$

Bellman uygunluk eşitlikleri ile en uygun değer fonksiyonlarını $v*$ veya $q*$ ifadelerini bulduğumuz zaman kolayca en uygun politikayı elde edebiliriz (Sutton ve Barto, 2018; yzai, 2023).

2.8.6. Markov karar süreçlerini çözmek

Sonlu Markov karar problemlerini çözmeye yönelik üç temel yöntem, Dinamik Programlama, Monte Carlo Yöntemi ve Zamansal Fark Öğrenimidir (Sutton ve Barto, 2018). Her yöntem sınıfının güçlü ve zayıf yönleri vardır.

Dinamik Programlama (DP) yöntemleri matematiksel olarak iyi gelişmiştir ancak ortamın tam ve doğru bir modelini gerektirir. Klasik DP algoritmaları, büyük hesaplama maliyeti nedeniyle PÖ'de sınırlı bir kullanıma sahiptir. Ayrıca DP yöntemi sonraki olayların olasılık dağılımlarına ihtiyacı olduğu için uygulaması kolay değildir. DP için öncelikle tüm ihtimallerin hesaplanması gerekmektedir.

Monte Carlo Yöntemi (MC) bir model gerektirmez ve kavramsal olarak basittir ancak adım adım artırımlı hesaplama için pek uygun değildir. MC'da tahmini güncellemek için bölüm sonunda elde edilen toplam ödülü (G) beklemeniz gerekir. Bazı PÖ problemlerinde bölümler çok uzun olabilir ve tahmin güncellemesini yapmak için bölüm sonunu beklemek algoritmayı verimsiz hale getirir. Bunu, denklem (2.8)'de verilen MC'nun tahmin güncelleme kuralında açıkça görebiliriz.

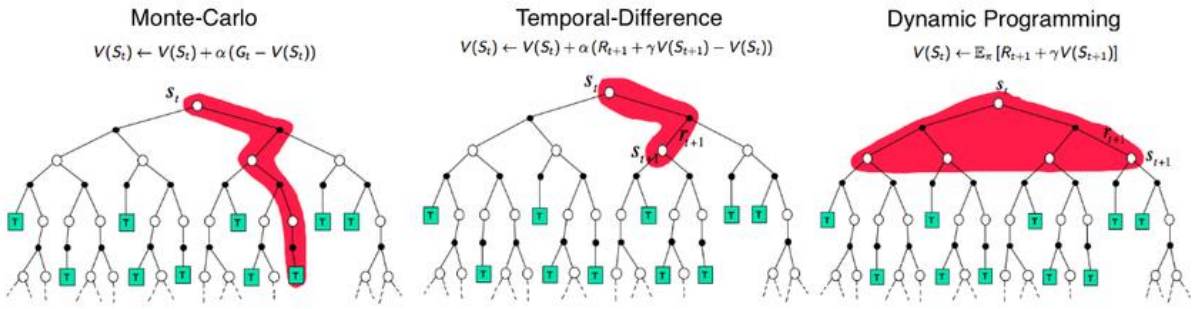
$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \tag{2.8}$$

Zamansal Fark Öğrenimi (ZFÖ) yöntemleri hiçbir model gerektirmez ve tamamen artırımlıdır ancak analiz edilmesi daha karmaşıktır. ZFÖ'de sadece bir sonraki ödülü kullanarak tahmin güncellenir. Bunu denklem (2.9)'da verilen ZFÖ'nün tahmin güncelleme kuralında açıkça görebiliriz.

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{2.9}$$

ZFÖ, ortamın modeline ihtiyaç duymamaktadır. Bu özelliği DP'ya göre en önemli üstünlüğüdür. MC yöntemlerine karşı en güçlü yanı ise bölüm sonunu beklemeden tahminlerini güncelleyebilmesidir. ZFÖ yöntemleri, herhangi bir sabit politika için yeteri kadar küçük adım boyu kullanılarak en iyi politikaya yakınsar. Ayrıca pratikte zamansal fark öğrenmesi yöntemleri genellikle MC yöntemlerinden daha hızlı yakınsar (Sutton ve Barto, 2018; yzai, 2023).

Durum değeri fonksiyonları için MC, ZFÖ ve DP'nın arama alanlarının karşılaştırılması Şekil 2.8'de görülmektedir (David Silver, 2023).



Şekil 2.8. Durum değer fonksiyonları için yöntemlerin karşılaştırılması

PÖ'de ilgilenilen problemlerde en uygun politika ve değer fonksiyonlarına ulaşmak çok büyük hesaplama maliyetleri oluşturabilir. Ortam dinamiklerinin doğru ve tam bir modeline sahip olunsa dahi, Bellman uygunluk eşitliği ile en uygun politikayı hesaplamak genellikle mümkün olmaz. Hesaplama maliyeti ile beraber, hafıza da önemli bir kısıtlama olmaktadır. Değer fonksiyonlarının, politikaların ve modellerin yaklaşımlarını oluşturmak çok büyük miktarlarda bellek gerektirebilmektedir.

Bu sebeple problemlerde, belli hata payları ile yaklaşıklıklar kullanılarak çözüme gidilir. İşe yarar bir politikanın, en uygun politika olmamakla beraber belli bir kriter içerisinde seyretmesi, yeterli bir yaklaşımı sağlayabilir.

PÖ'nün çevrimiçi (online) yapısı; daha az sıklıkla karşılaşılan durumlara daha az efor sarf ederek, sıkça karşılaşılan durumlarda daha iyi kararlar alması için en uygun politikaya yaklaşım yaptırmaya imkan sağlar. Bu, PÖ'nün, MKS'leri yaklaşık olarak çözen diğer yaklaşımlardan ayıran önemli bir özelliğidir.

2.8.7. Zamansal fark öğrenimi (ZFÖ)

ZFÖ (Sutton, 1988), ardışık zamanlarda yapılan eylemlerin değerlerinin farkıdır. DP ve MC yöntemlerinin birleşimidir. MC yöntemindeki gibi ortamı temsil eden bir model olmadan doğrudan ham deneyimlerden yararlanır. DP yöntemindeki gibi nihai sonucu beklemeden tahminleri, kısmen diğer öğrenilmiş tahminlere dayalı olarak günceller. ZFÖ durum değerini denklem (2.10)'da görülen fonksiyon ile günceller.

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \qquad (2.10)$$

ZFÖ yöntemlerinin yalnızca bir sonraki zaman adımına kadar beklemesi gerekir. $t+1$ anında hemen bir hedef oluşturur ve gözlemlenen R_{t+1} ödülünü ve $V(S_{t+1})$ tahminini kullanarak yararlı bir güncelleme yapar. En basit ZFÖ yöntemi, güncellemeyi S_{t+1}'e geçiş ve

R_{t+1} almanın hemen ardından yapar. ZFÖ güncellemesinin hedefi ise $R_{t+1} + \gamma V(S_{t+1})$'dir. Bu ZFÖ yöntemine TD(0) veya tek adımlı ZFÖ adı verilir.

Zamansal fark kavramı sayesinde $t+1$ zamanında bulunacağımız durumun ve yapacağımız eylemin kalitesini hesaplarız. Denklemde kullanılan (α) öğrenme oranını temsil eder. Bu, modelin öğrenme hızını ayarlar.

2.8.8. Q-Öğrenme

PÖ'deki ilk atılımlardan biri geciktirilmiş ya da geç gelen ödüllerden öğrenme kavramı (Watkins vd., 1989) ve buradan da Q-Öğrenme (Watkins ve Dayan, 1992) olarak bilinen politika dışı bir ZFÖ algoritmasının geliştirilmesidir (Sutton ve Barto, 2018).

Q-Öğrenme, politikasız bir denetim algoritmasıdır. $V(s)$ durum değerleri yerine $Q(s,a)$ durum eylem çiftinin getirisi (kalitesi) ile ilgilenir. Temel fikir, ödüllere ve Q fonksiyonuna dayanarak eylemler için Q-değerlerini aşamalı olarak tahmin etmektir (Otterlo ve Wiering, 2012: 29). Denklem (2.11) Q-Öğrenme için tahmin güncelleme kuralıdır ve Q fonksiyonu olarak isimlendirilir. Bu formülde görebileceğiniz gibi eylem seçimi politikaya dayalı olmadığından, Q tablosuna bakılarak, ayrık eylem kümesinden gelecek durum için en yüksek getiriyi veren eylem $(Q(s_{t+1}, a))$ seçilir. Q tablosu, durum eylem çiftlerinin Q değerlerini tutan iki boyutlu matrisi temsil eder. Ajan ortam üzerinde aksiyonlar yaparak Q tablosunu günceller.

$$Q^{yeni}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right) \qquad (2.11)$$

Q fonksiyonuna göre, ajan a_t eylemini kullanarak ortamda s_t durumundan s_{t+1}'e geçerken r_{t+1} ödülünü alır. Güncelleme, s_t durumunun ve a_t eyleminin Q değeri üzerinde gerçekleşir.

α: Öğrenme oranı (Alfa), $(0 < \alpha \leq 1)$.

γ: İndirim faktörü (Gama) gelecekte kazanılması muhtemel ödüllerin önemini belirler.

r_{t+1}: s_{t+1} durumunda a_{t+1} eylemi ile alınacak ödül.

$Q(s_t, a_t)$: Mevcut durumun getirisi.

$\gamma \max_a Q(s_{t+1}, a)$: Sonraki durumdan alınabilecek indirgenmiş en yüksek getiri.

$\left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a)\right)$: Beklenen getiri.

(Watkins ve Dayan, 1992)'da Q-Öğrenme yaklaşımının problemlerde optimum sonuca yakınsama garantisi olduğu gösterilmiştir.

Şekil 2.9. Q-Öğrenme algoritmasının akış şemasını gösterir. Q-Öğrenme sözde kodu ise Algoritma 1'de verilmiştir.

Şekil 2.9. Q-Öğrenme algoritması akış şeması

Algoritma 1. Q-Öğrenme (Watkins ve Dayan, 1992)

İndirim faktörü γ ve öğrenme oranı α'yı tanımla
Q değerlerini rastgele başlat, Q(s,a)=0, ∀ s ∈ S, ∀ a ∈ A
for1 bölüm = 1, M **do**
    Ortamı başlangıç konumuna döndür ve durum değerini al, $s = s_1$
    for2 adım = 1, T **do**
        Keşif stratejisine dayalı olarak bir eylem seç, a ∈ A(s)
        a eylemini uygula, yeni durum, s' ve ödül, r'yi gözlemle
        $Q(s,a) \coloneqq Q(s,a) + \alpha \left[r' + \gamma \max_a Q(s',a) - Q(s,a) \right]$
        s ≔ s'
        s, son durum olana kadar devam et
    end for2
end for1

2.8.9. Sömürü ve keşif

Sömürü ve keşif, PÖ ajanının eylem seçim politikası için kullanılan bir yöntemdir. Sömürü her zaman ajanın mevcut bilgisine göre getirisi en yüksek olan eylemi seçmesidir. Keşif ise ajanın ortam hakkında bilgi toplamak için rastgele eylemde bulunmasıdır. Sömürüye aynı

zamanda açgözlü (greedy) yaklaşım adı verilir. Eğer birden fazla açgözlü eylem varsa aralarında rastgele seçim yapılır. Açgözlü politika (greedy policy), var olan bilgiyi günceller ve iyileştirir ancak ortam hakkında daha fazla bilgi almamızı ve ortamı keşfetmemizi engeller. Keşif ise eylem uzayından seçilen rastgele eylemler ile yeni durumlar keşfetmeye olanak tanır. Bu yeni durumlar iyi de olabilir kötü de.

Ortamı keşfetmek ve daha iyi bir ödül mekanizmasını açığa çıkarmak için açgözlü politikaya olasılıksal bir değer (epsilon (ε)) ekleyip, olasılık gerçekleştiğinde rastgele hareket etmek hedeflenmektedir. Bu politikaya ε-açgözlü politika (epsilon greedy policy) (Sutton ve Barto, 2018) denilmektedir.

$$eylem\ seçimi \begin{cases} \max_a Q(s,a) & 1-\varepsilon\ , olasılıkla \\ rastgele\ eylem, a\ \epsilon\ A & \varepsilon\ , olasılıkla \end{cases} \tag{2.12}$$

Eşitlik (2.12)'de verilen politikaya göre ε keşif olasılığıdır. Ajanın eylemlerinde ne kadar risk alacağını belirler. Ajan ε olasılıkla rastgele, $1-\varepsilon$ olasılıkla ise açgözlü politikaya göre hareket eder. Epsilon değeri eğitim başlangıcında belirlenir.

2.9. Derin Pekiştirmeli Öğrenme (DPÖ)

Pratik karar verme problemlerinde MKS çerçevelerinin büyük bir kısmı çok yüksek boyutludur. Bir kameradan gelen görüntüler veya robotlardan alınan ham sensör verileri geleneksel PÖ algoritmalarıyla çözülemeyecek kadar yüksek veri içerir. DPÖ algoritmaları, bu tür MKS'leri çözmek için genellikle politikayı temsil eden $\pi(a|s)$ derin sinir ağları (DNN)'nı kullanır.

DPÖ'nün ana fikri, Q fonksiyonunu temsil etmek için DNN kullanmak ve bu sinir ağını toplam ödülü tahmin edecek şekilde eğitmektir. PÖ'yü sinir ağlarıyla birleştirmeye yönelik bazı girişimler, kararsız öğrenme nedeniyle büyük ölçüde başarısız olmuştu. Mnih ve arkadaşları bu kararsızlıkları gidermek için DQN (Mnih vd., 2013) (Mnih vd., 2015) algoritmalarını tanıttılar. Bunlar DPÖ alanında geniş çapta başarılı olan ilk algoritmalardır. DQN, kararsız öğrenme sorununa karşı ajanın tüm deneyimlerini tekrar hafızası da denilen bir bellekte depolar ve eğitim sürecinde çeşitli ve ilişkisiz eğitim verileri sağlamak için bu deneyimlerden yararlanır.

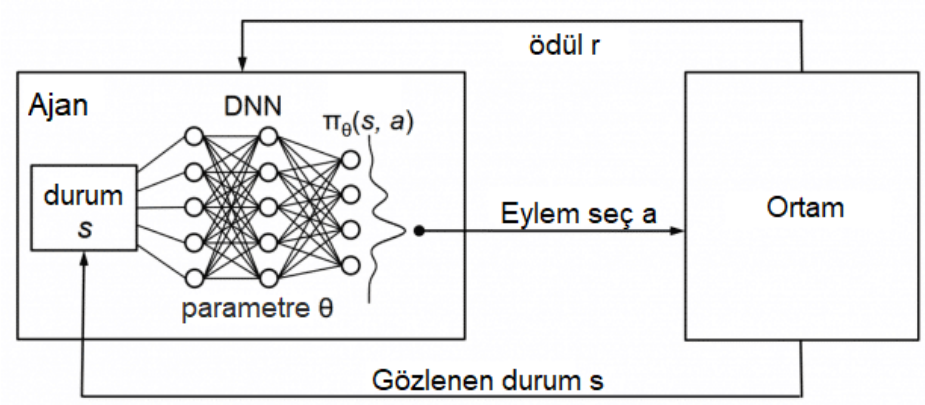
Q-Öğrenme ve DQN'de en yüksek getiriyi sağlayan eylemlerin seçilmesi aşırı iyimser değer tahminlerine yol açar fikrini ortaya atan Van Hasselt ve ark. açgözlü politikayı çevrimiçi ağa göre değerlendiren, ancak değerini tahmin etmek için hedef ağı kullanan Çift Derin Q Ağları'nı (Double DQN veya DDQN) (V. Hasselt vd., 2016) önermişlerdir.

2.9.1. Derin Q ağları

Mnih ve arkadaşları, "*Playing atari with deep reinforcement learning*" başlıklı makalelerinde (Mnih vd., 2013) PÖ'yi kullanarak denetim politikalarını doğrudan yüksek boyutlu duyusal girdiden başarılı bir şekilde öğrenmek için ilk Derin Öğrenme modelini sundular.

Derin Q Ağları (Mnih vd., 2013) modeli, Q-öğrenmeyi görüntüler gibi uzamsal veri dizilerini işlemek için uzmanlaşmış çok katmanlı derin bir yapay sinir ağı (YSA) olan CNN ile birleştirir. Burada CNN, optimum eylem-değer fonksiyonuna yaklaşmak için kullanılır. Modelin girdisi ham pikseller ve çıktısı gelecekteki ödülleri tahmin etmek için değer fonksiyonu olarak kullanılan CNN'dir. Model, Arcade Öğrenme Ortamındaki yedi Atari 2600 oyununa, mimarisi veya öğrenme algoritmasında hiçbir ayarlama yapmadan uygulanmıştır. Amaç mümkün olduğu kadar çok sayıda oyunu başarıyla oynamayı öğrenebilen tek bir sinir ağı ajanı oluşturmaktır. Model, ekran görüntüsü, ödül, terminal sinyali ve olası eylem uzayı ile eğitilerek altı oyunda önceki tüm yaklaşımlardan daha iyi başarım ve üçünde ise insan düzeyinin de üzerinde başarım elde etmiştir.

Şekil 2.10'da DNN ile temsil edilen eylem politikasına dayalı PÖ şeması görülmektedir.



Şekil 2.10. DNN ile temsil edilen politikayla PÖ (Mao vd., 2016)

Pek çok PÖ algoritmasının arkasındaki temel fikir, Bellman denklemini yinelemeli bir güncelleme aracı olarak kullanarak eylem değeri fonksiyonunu tahmin etmektir. Optimal eylem-değer fonksiyonu $(Q^*(s;a))$ elde edilebilecek maksimum beklenen getiri olarak tanımlanır ve bunu tahmin etmek için bir fonksiyon yaklaşım aracının kullanılması yaygındır. $Q(s;a;\theta) \approx Q^*(s;a)$, θ (teta) ağırlıklarına sahip bir sinir ağı fonksiyon yaklaşıklayıcısına Q ağı adı verilmiştir.

Q ağının eğitimi için geriye yayılan optimize edici öğrenme oranına zaten sahip olacağından, Q-Öğrenme'de kullanılan eylem-değer fonksiyonundaki (denklem (2.11)) öğrenme oranına artık gerek yoktur. Öğrenme oranı kaldırılınca, optimum eylem-değer fonksiyonu, denklem (2.13) elde edilir. Bu değer yineleme algoritması, optimum eylem-değer fonksiyonuna yakınsar.

$$Q^*(s,a) = E_{s'\sim\varepsilon}\left[r + \gamma \max_{a'} Q^*(s',a') \,|s,a\,\right] \tag{2.13}$$

Q ağı, her i yinelemesinde değişen bir dizi kayıp fonksiyonu $L_i(\theta_i)$ en aza indirilerek eğitilebilir. Denklem (2.14)'deki kayıp fonksiyonu, hedef değer (y_i) ile tahmin edilen değerin yani $Q(s,a;\theta_i)$'nın farklarının karesidir.

$$L_i(\theta_i) = E_{s,a\sim p(.)}\left[\left(y_i - Q(s,a;\theta_i)\right)^2\right] \tag{2.14}$$

Burada $y_i = E_{s'\sim\varepsilon}[r + \gamma Q(s',a')|s,a\,]$ hedef değerdir ve $\rho(s,a)$ davranış dağılımı olarak adlandırdığımız s durumlarında, a eylemleri üzerindeki bir olasılık dağılımıdır.

Q ağı, deneyim tekrarı (Lin, 1993) olarak bilinen teknik kullanılarak eğitilir. Buna göre ajanın deneyimleri $e_t = (s_t; a_t; r_t; s_{t+1})$, her zaman adımında birçok bölüm boyunca tekrar hafızasında toplanır $D = e_1, \ldots, e_n$. Algoritmanın iç döngüsü sırasında, Q-öğrenme güncellemeleri veya mini toplu güncellemeleri, depolanan tekrar hafızasından rastgele alınan $e\sim D$ deneyim numuneleri ile yapılır.

Deneyim tekrarını gerçekleştirdikten sonra ajan, açgözlü eylem politikasına (ε greedy) göre bir eylem seçer ve uygular. Buna göre ajan ε ihtimalle rastgele bir eylem seçer. $1-\varepsilon$ ihtimalle ise eylem kararını Q ağı verir.

DQN algoritmasının sözde kodu, Algoritma 2'de sunulmuştur. Algoritmada "terminal" sinyali bölümü sonlandırmak için kullanılır. Oyunda ajan yanarsa ya da zaman aşımında terminal değeri "true" olur.

Algoritma 2. Deneyim yeniden oynatma ile DQN (Mnih vd., 2013)

Yeniden oynatma belleği D'yi N kapasite değeri kadar tanımla
Q ağını rastgele ağırlıklarla başlat
for1 bölüm = 1, M **do**
  Ortamı başlangıç konumuna döndür ve durum değerini al, $s_t = s_1$
  for2 t = 1, T **do**
    a_t eylemini, ε olasılığınca rastgele ya da $a_t = max_a Q^*(s_t, a; \theta)$ hesabı ile seç
    a_t eylemini yap ve r_t, s_{t+1} ve *terminal* değerlerini al
    D 'ye $(s_t, a_t, r_t, s_{t+1}, terminal_t)$ verilerini ekle
    D 'den rastgele mini toplu yığın örneği al $(s_j, a_j, r_j, s_{j+1}, terminal_j)$

$$y_j = \begin{cases} r_j & , terminal_{t+1} = true\ ise \\ r_j + \gamma\ max_{a'}\ Q(s_{j+1}, a'; \theta) & , terminal_{t+1} = false\ ise \end{cases}$$

    $L_j = (y_j - Q(s_j, a_j; \theta))^2$ kayıp fonksiyonu ile θ ağırlıklarını güncelle
  end for2
end for1

Mnih ve arkadaşları, "*Human-level control through deep reinforcement learning*" başlıklı makaleleri ile (Mnih vd., 2015) DQN algoritmasını geliştirdiler. Bu çalışma ile önceki algoritmaya (Mnih vd., 2013) göre daha başarılı sonuçlar elde edildiği belirtilmiştir. Yeni algoritma, Algoritma 3'de sunulmuştur.

Buna göre ağırlıkları farklı, Q ağı (θ_i) ve hedef Q ağı (θ_i^-) olmak üzere iki farklı sinir ağı kullanılmıştır. Hedef ağın ağırlıkları belirlenen bir "C" katsayısı adımında Q ağının parametreleri ile güncellenir $(\theta_i^- = \theta_i)$. Yinelemedeki Q-öğrenme güncellemesi denklem (2.14)'deki kayıp fonksiyonunu kullanır.

$$L_i(\theta_i) = E_{(s,a,r,s')\sim U(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right] \qquad (2.14)$$

Algoritma 3. Deneyim yeniden oynatma ile DQN (Mnih vd., 2015)

Yeniden oynatma belleği D'yi N kapasitesiyle tanımla
Eylem-değer fonksiyonu Q'yu rastgele ağırlıklarla başlat, θ
Hedef eylem-değer fonksiyonu $\hat{Q}$'yu θ ağırlıkları ile başlat, $\theta^{-} = \theta$
for1 bölüm = 1, M **do**
  Ortamı başlangıç konumuna döndür ve durum değerini al, $s_t = s_1$
  for2 t = 1, T **do**
    a_t eylemini, ε olasılığınca rastgele ya da $a_t = argmax_a Q(s_t, a; \theta)$ hesabı ile seç
    a_t eylemini yap ve r_t, s_{t+1} ve *terminal* değerlerini al
    D 'ye $(s_t, a_t, r_t, s_{t+1}, terminal_t)$ verilerini ekle
    D 'den rastgele mini toplu yığın örneği al $(s_j, a_j, r_j, s_{j+1}, terminal_j)$

$$y_j = \begin{cases} r_j & , terminal_{t+1} = true\ ise \\ r_j + \gamma\ max_{a'}\ \hat{Q}(s_{j+1}, a'; \theta^{-}) & , terminal_{t+1} = false\ ise \end{cases}$$

    $L_j = (y_j - Q(s_j, a_j; \theta))^2$ kayıp fonksiyonu ile θ ağırlıklarını güncelle
    Her C adımda hedef eylem-değer fonksiyonunu güncelle $\hat{Q} = Q$
  end for2
end for1

2.9.2. Çift derin Q ağları (DDQN)

Van Hasselt ve arkadaşları, "*Deep Reinforcement Learning with Double Q-Learning*" başlıklı makalesinde (V. Hasselt vd., 2016) DQN (Mnih vd., 2015) algoritmasına iyileştirme önermişlerdir. Buna göre Q-Öğrenme'deki maksimum getirili eylem seçiminin aşırı iyimser değer tahminlerine yol açtığı, DQN algoritmasının bazı oyunlarda önemli ölçüde fazla tahminlerden muztarip olduğu ifade edilmiştir. Tablo yapısında sunulan Çift Q-öğrenme (Hasselt, 2010) algoritmasının arkasındaki fikrin, DNN'ler de dahil olmak üzere büyük ölçekli fonksiyon yaklaşımıyla çalışacak şekilde genelleştirilebileceği ve Çift Derin Q Ağları (DDQN) algoritmasının varsayıldığı gibi yalnızca gözlemlenen aşırı tahminleri azaltmakla kalmayıp, aynı zamanda birçok oyunda çok daha iyi sonuca yol açacağı da ifade edilmiştir.

Çift Q-Öğrenme yönteminin amacı hedefteki maksimum işlemi, eylem seçimi ve eylem değerlendirmesine ayrıştırarak fazla tahminleri azaltmaktır. DQN mimarisindeki hedef ağ, ikinci değer fonksiyonu için kullanılır. Buna göre açgözlü politikayı çevrimiçi ağa ($Q(s, a; \theta)$) göre değerlendirmek, ancak değerini tahmin etmek için hedef ağı ($Q(s, a; \theta^{-})$) kullanmak önerilmiştir. Hem Çift Q-öğrenme hem de DQN'in kullanılmasıyla, ortaya çıkan algoritmaya Çift DQN (DDQN) adı verilmiştir. Güncellemesi DQN ile aynıdır ancak hedef eylem-değer fonksiyonu Y_t^{DQN} 'i denklem (2.16)'da görülen fonksiyon ile değiştirir:

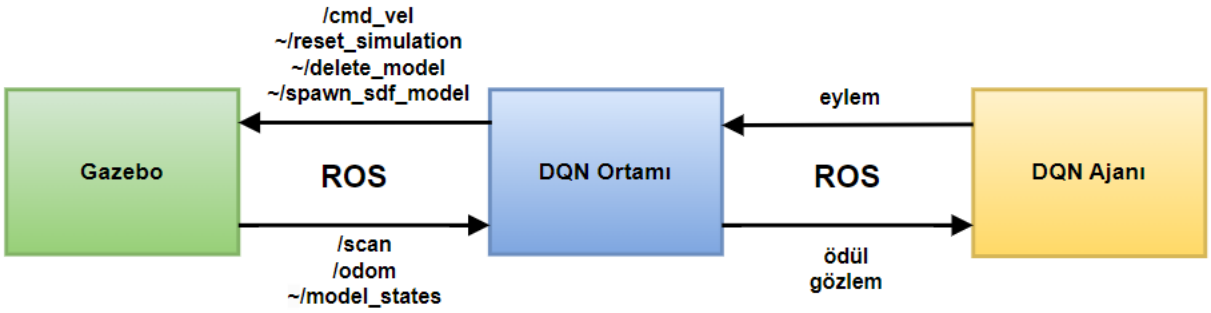
$$Y_t^{DDQN} = R_{t+1} + \gamma Q\left(S_{t+1}, \underset{a'}{\text{argmax}}\, Q(S_{t+1}, a'; \theta_t)\,; \theta_t^{-}\right) \qquad (2.16)$$

3. TEKNOLOJİ VE YÖNTEM

Bu bölümde, çalışmamızda kullandığımız ROS, Gazebo ve DQN teknolojilerinin özelliklerinden ve bunların arasındaki iletişim mimarisinden bahsedilmiştir. TurtleBot3 robotu DQN ajanı tarafından kontrol edilerek Gazebo ile oluşturulan ortamda keşif yapar, aynı zamanda Q ağı eğitilerek TurtleBot3 deneyimlerinden giderek daha fazla faydalanır. Ajan ve ortam arasındaki iletişim ROS çerçevesi ile gerçekleştirilir.

3.1. Uygulama Modeli

Gazebo, ROS ve DQN ajanı arasındaki ilişkiyi gösteren şema Şekil 3.1'de sunulmaktadır. Bu mimari, ROS ortamında tanımlanan düğümlerin, aralarında bilgi aktarımı sağlayan konular aracılığıyla etkileşime girdiği bir yapıdır. İlk olarak, Gazebo'nun iletişim ara yüzü ana düğüme bağlanır. Daha sonra DQN ajanı, kontrol stratejisine dayalı eylem talimatlarını "/cmd_vel" konusu üzerinden "/gazebo" ya geri bildirim olarak iletir. Gazebo TurtleBot3'ü çalıştırarak eylemi uygular ve TurtleBot3'ün tüm durumlarını ve ortamı günceller (Tsai vd., 2022). DQN Ortamı ise yapılan eylem sonrası ajanın alacağı ödülü ve yeni durum verilerini döndürür.



Şekil 3.1. Uygulamada kullanılan yöntemler ve ortamlar arası iletişim mimarisi

ROS'da çalıştırılabilen her birim, düğüm olarak tanımlanır. Düğümler arası veri iletimi konular üzerinden mesajlar ile yapılır. Örneğin bir robotu hareket ettirmek için, "/cmd_vel" konusu üzerinden "twist" mesajı ile doğrusal ve açısal hız mesajları motorları denetleyen düğüme iletilir. Hız mesajlarını alan düğüm aracılığı ile robot istenilen hareketi yapar.

3.2. ROS Konuları

ROS çerçevesinde çalışan Gazebo benzetim ortamında, DQN ajanı olan, gezgin robotu denetlemek için ilgili düğümlere konular aracılığı ile mesajlar gönderilmelidir. Konular düğümler arasında mesajların iletilmesi için mesaj taşıyan kanallar olarak düşünülebilir. Mevcut tüm konuların listesini almak için "rostopic list" komutu kullanılır. Yayınlanan

konulara ait mesajları dinlemek için “rostopic echo” komutu kullanılır. Örneğin LIDAR datalarını okumak için: “rostopic echo /scan” komutu yazılır. Aşağıda Şekil 3.1 üzerinde verilen ve çalışmada kullanılan ROS konularının açıklamaları kısaca verilmiştir.

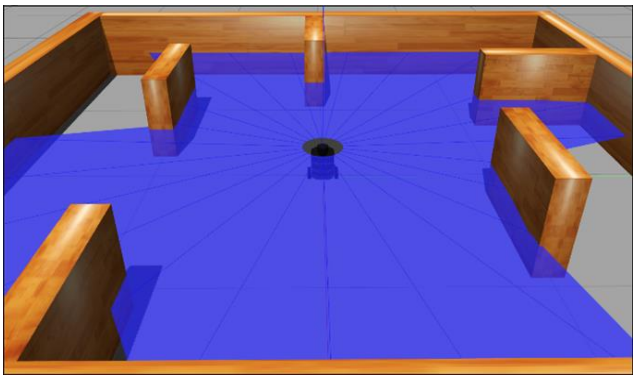
gazebo/model_states: Simülasyondaki tüm modellerin durum bilgilerini yayınlar.

/scan: LIDAR verileri bu kanal üzerinden yayınlanır. TurtleBot3 Burger modelinde 360 derece LIDAR sensörü bulunur (TurtleBot3, 2023). LIDAR sensörü verileri ortam hakkında bilgi sağlar. TurtleBot3 Burger’da LIDAR sensörünün algılama mesafeleri yaklaşık olarak minimum 12cm, maksimum 3,5m’dir.

/cmd_vel: Bu konu üzerinden “Twist” mesaj türünde açısal ve doğrusal hız mesajları yayınlanır. Bu konuya abone olan robot hız mesajlarını aldığı zaman ilgili hareketi yapar.

/odom: Bu konu üzerinden, odometri bilgilerini içeren mesajlar iletilir. Odometri, robotun hareketine bağlı olarak pozunun ve hızının yerel olarak doğru bir tahminini sağlar. Odometri bilgisi IMU sensörü, LIDAR, kilometre sayacı vb. gibi çeşitli kaynaklardan elde edilebilir.

Şekil 3.2’de sanal ortamda LIDAR kullanarak otonom olarak gezinmeyi öğrenen TurtleBot3 robotu görülmektedir.



Şekil 3.2. Sanal bir ortamda gezinen TurtleBot3

3.3. Hizmetler

ROS konuları tek yönlü mesaj iletimi yapar. İstek / yanıt biçiminde iki yönlü mesaj iletimi için hizmetler (servisler) kullanılır. Çalışmada kullanılan hizmetler aşağıda kısaca verilmiştir.

gazebo/reset_simulation: ROS çerçevesinde Gazebo benzetim ortamını başlangıç durumuna döndürmek (sıfırlamak) için kullanılan servistir. Yüklenen Gazebo dünyasını, robotların konumları, hızları ve durumlarını başlangıç değerlerine geri döndürür. Çalışmamızda

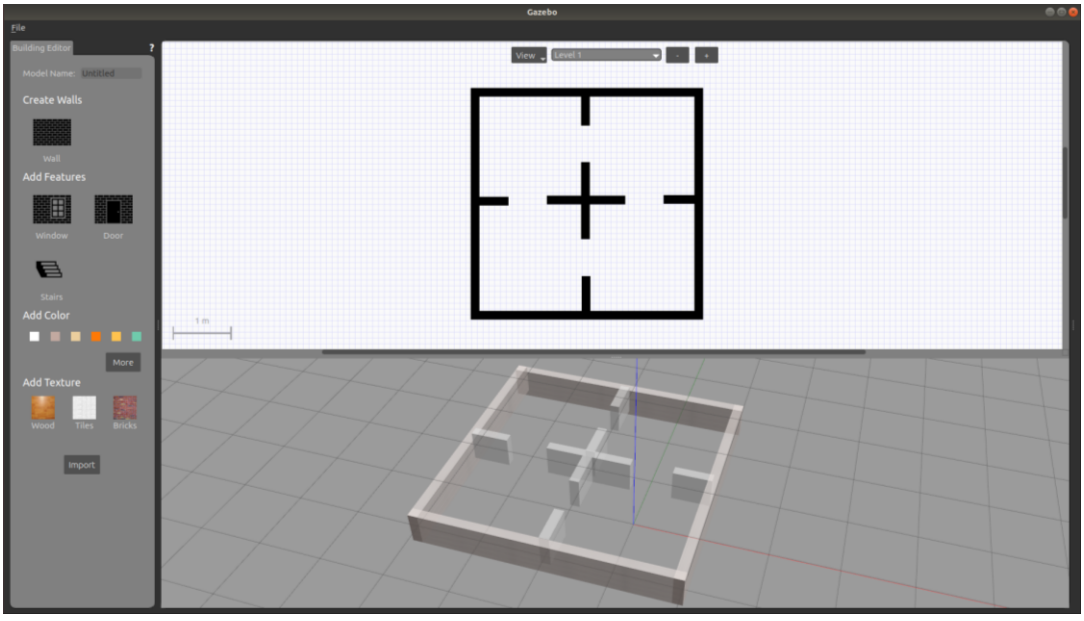
eğitim sürecinde TurtleBot3 bir engele çarparsa olumsuz bir ödül alır ve bölüm sonlandırılarak Gazebo ortamı sıfırlanır.

gazebo/spawn_sdf_model: Gazebo Simülasyon Açıklama Formatında (SDF) yazılmış bir model oluşturmak için bu hizmet kullanılır. Model uygulama sırasında dinamik olarak oluşturulabilir. Uygulama da hedef noktanın yerini göstermek için kullanılmıştır.

gazebo/delete_model: Bu hizmet bir modeli dinamik olarak benzetim ortamından silmek için kullanılır. Uygulama da hedef noktanın yerini değiştirirken eskisini silmek için kullanılmıştır.

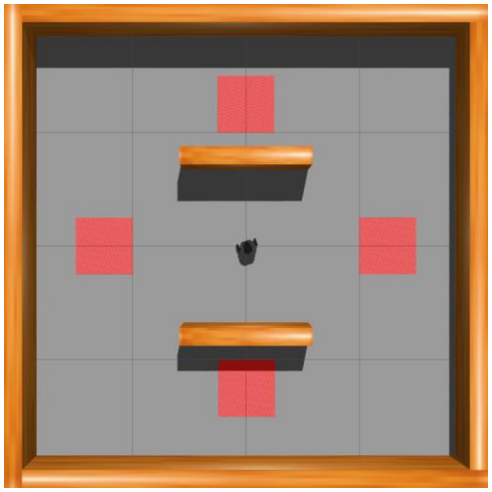
3.4. Benzetim Ortamı

Gazebo'da yapı editörü ile istenilen benzetim ortamı tasarlanabilir. Şekil 3.3'de yapı editörünün ekran görüntüsü görülmektedir. Sıfırdan bir ortam oluşturulabilir veya uygun formatta mevcut bir görüntü kullanılarak ortam içe aktarılabilir. Bu görüntü örneğin bir binanın 2 boyutlu lazer taraması olabilir (Gazebo Tutorial, 2023b). Sol panelde malzemeler ve özelliklerin seçilebildiği bölüm, üstte iki boyutlu çizim alanı, altta ise çizimin üç boyutlu benzetim hali görülmektedir.



Şekil 3.3. Gazebo yapı editörünün ekran görüntüsü

Uygulamamızda eğitim ve test işlemleri için Şekil 3.4'de görülen ortam oluşturulmuştur. Ortama dört farklı hedef nokta yerleştirilmiştir. Hedef noktalar yine Şekil 3.4'de görülmektedir.



Şekil 3.4. Çalışma için oluşturulan benzetim ortamı "GazeboStage1"

3.5. Ajan ve Çevre

Amacımız, robotu bulunduğu konumdan hedef noktaya minimum çabayla ulaştırmak için en uygun politikayı öğrenen DQN ajanı elde etmektir. Ajan bulunduğu ortamı ε-açgözlü stratejisi ile keşfeder. PÖ'de durum uzayı, eylem uzayı ve ödül sistemi en temel unsurlardır. Uygun bir ödül sistemi ile ajan kendi kendine öğrenecektir. Şekil 3.5'de ajanın çalışma süreci öbek şema ile gösterilmiştir (Alvarez, 2019). Ajan'ın amacı, TurtleBot3'ü engellerden kaçınarak hedef noktalara ulaştırmaktır. Bu süreç şu adımlar ile özetlenebilir.

• Ajan, eylemi Q ağından aldığı Q değerlerine göre seçer ve ortama yansıtır.

• Ajan, bir ortamda elde ettiği deneyimi tekrar hafızasında saklar.

• Ajan, saklanan verileri tekrar hafızasından rastgele örnekleyerek Q ağını eğitir.

Şekil 3.5. Ajanın çalışma süreci

3.5.1. Durum uzayı

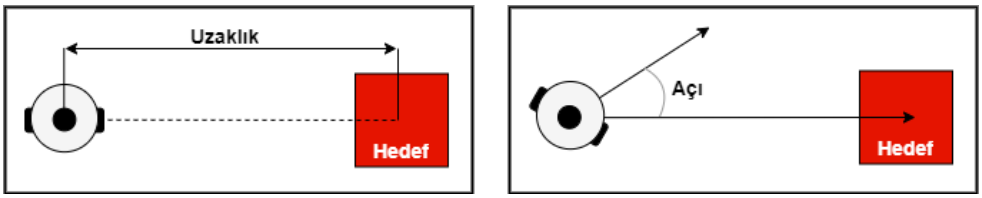
Durum (state), çevrenin bir gözlemidir ve mevcut durumun temsilidir. Durum verisi ajan için çok önemlidir. Durum, Q ağına beslenir ve Q değerleri hesaplanarak eylem seçimi yapılır. Modelimizde durum 28 veriden oluşur. Bu verilerin 24 tanesi Lazer Mesafe Sensörü (LDS) değeridir. Diğerleri ise hedefe olan açı, hedefle arasında ki mesafe, engel ile arasındaki mesafe ve engel ile arasındaki açıdır. Bu verilerin matematiksel bir yaklaşım ile ifadesi eşitlik (3.1)'de görsel şekli de Şekil 3.6'da gösterilmiştir (ROBOTIS-GIT, 2022).

Durum (28) = {LDS (24), Hedefe Açı (1), Hedefe Mesafe (1), Engele Mesafe (1), Engele Açı (1)} (3.1)

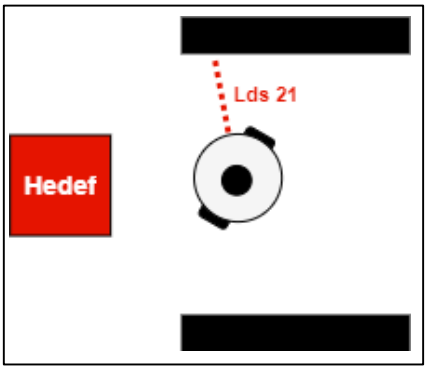
Lidar Data Sinyali (1)	Lidar Data Sinyali (2)	Lidar Data Sinyali (3)	Lidar Data Sinyali (4)	Lidar Data Sinyali (5)	Lidar Data Sinyali (6)	Lidar Data Sinyali (7)	Lidar Data Sinyali (8)	Lidar Data Sinyali (9)	Lidar Data Sinyali (10)	Lidar Data Sinyali (11)	Lidar Data Sinyali (12)	Lidar Data Sinyali (13)	Lidar Data Sinyali (14)	Lidar Data Sinyali (15)	Lidar Data Sinyali (16)	Lidar Data Sinyali (17)	Lidar Data Sinyali (18)	Lidar Data Sinyali (19)	Lidar Data Sinyali (20)	Lidar Data Sinyali (21)	Lidar Data Sinyali (22)	Lidar Data Sinyali (23)	Lidar Data Sinyali (24)	Hedefe Açı	Hedefe Mesafe	Engele Mesafe	Engele Açı
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28

Şekil 3.6. Durum uzayı

Hedefe açı robotun doğrultusu ile hedef nokta arasındaki açıdır. Hedefe mesafe robot ile hedef nokta arasındaki mesafedir. Engele mesafe, LDS den alınan değerlerin en küçüğüdür. Engele olan açı ise en yakın engeli gören LDS numarasıdır. Bu metrikler Şekil 3.7 ve Şekil 3.8'de resmedilmiştir.



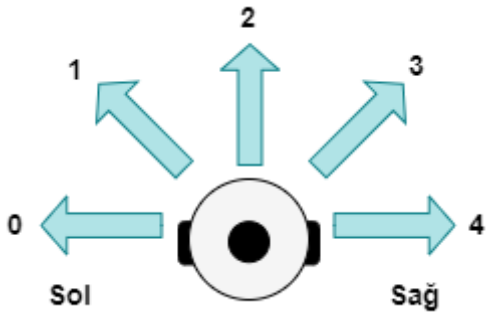
Şekil 3.7. Robot ile hedef nokta arasındaki mesafe (sol) ve açı (sağ)



Şekil 3.8. Robot ile en yakın engel arasındaki mesafe ile hedefi gören LDS numarası

3.5.2. Eylem uzayı

Robotun, durum verisine bağlı olarak hareket edeceği beş eylemi (action) vardır. Eylemler Şekil 3.9'da resmedilmiştir. Beş farklı eylemde sırasıyla açısal hızlar -1.5, -0.75, 0, 0.75 ve 1.5 rad/sn'dir. Bu çalışmada robotun sabit doğrusal hızı 0.15m/s'dir ve açısal hızı Q ağı tarafından belirlenir. Q ağının maksimum Q değeri eylem seçimini belirler.



Şekil 3.9. Eylem uzayı

3.5.3. Ödül sistemi

Ödül sistemi öğrenme için çok önemlidir. Robot her eyleminde ortamdan ödül (reward) alır. Bu, olumlu ya da olumsuz ödül olabilir. Robot hedefi her bulduğunda 4 farklı hedef noktadan biri rastgele seçilerek yeni hedef nokta olur ve yeni hedefi aramaya devam eder. Turtlebot3 hedefi bulduğunda 200 ödül puanı, engele çarparsa -150 ceza puanı alır. 14 cm ve altındaki LIDAR verisi çarpışma olarak değerlendirilir. Ayrıca robotumuz hedef nokta doğrultusunda, hedefe yaklaştığı her adımda küçük bir ödül puanı alır. Hedeften uzaklaştıkça veya farklı doğrultularda yaptığı hareketlerde küçük bir ceza puanı alır. Amaç bölüm boyunca toplanan ödülü en yükseğe çıkarmaktır. Her bölüm (episode), TurtleBot3 bir engele çarptığında veya 500 adım sayısından sonra sona erdirilir.

3.6. Kullanılan Yazılım, Kütüphane ve Donanımlar

Uygulama da kullanılan yazılımlar ve sürüm numaraları aşağıdaki şekildedir:

- Ubuntu 18.04 LTS
- ROS Melodic
- Gazebo v9.0.0
- Python 2.7
- Anaconda 4.9.2
- Tensorflow 1.15.0
- Keras 2.2.4
- Rospkg

Donanım özellikleri:

- Intel Core i7 4500U işlemci
- 8GB DDR3 RAM
- NVIDIA GeForce GT 730M ekran kartı

4. GEZGİN ROBOT İÇİN DERİN PEKİŞTİRMELİ ÖĞRENME UYGULAMASI

Benzetim senaryoları için, Şekil 3.4'deki "GazeboStage1" ortamı oluşturulmuş ve üç farklı DPÖ algoritması (Mnih vd. 2013, Mnih vd. 2015 ve V. Hasselt 2016) ile oluşturulan ajanlar bu ortamda eğitilmiştir. Sonuçları da beşinci bölümde paylaşılmıştır. Eğitim sürecinde DPÖ ajanı için bazı parametrelerin belirlenmesi gerekir. Bunlar:

Öğrenme oranı (α): Derin sinir ağının eğitimi için kullanılır. 0'a yakın değerler robotun yavaş öğrenmesini; 1'e yakın yüksek değerler, öğrenmenin hızlı olacağı anlamına gelir. Ancak hızlı öğrenme de optimum noktaya yakınsama olmayabilir. Öğrenme oranı α = 0.001 olarak seçilmiştir.

İndirim faktörü (γ): 0 değeri ajanın yalnızca mevcut ödülleri dikkate almasını sağlar. Gelecek durumlardan ödül beklentisi olamadığı anlamını taşır. 1'e yaklaşan bir faktör, ajanın uzun vadeli yüksek bir ödül için çabalaması anlamına gelir. İndirim faktörü γ = 0.95 olarak seçilmiştir.

Keşif sabiti (ε): 0,99 gibi yüksek bir değer ayarlamak, eylemlerin %99'unu stokastik yapacaktır. Keşif aşamasında yüksek bir ε değeri ile başlar, bölümler ilerledikçe epsilon azaltma tekniği ile 0'a yaklaştırırız. Bu sayede ajan öğrendikçe deneyimlerden daha fazla faydalanırız. Başlangıçta ε = 1 olarak belirlenmiştir. Minimum ε değeri olan 0,01'e ulaşana kadar her bölümde $\varepsilon * 0{,}995$ azaltma tekniği kullanılmıştır.

4.1. Q Ağı

(Mnih vd. 2013, 2015)'deki makalelerde oyunların ekran görüntüsü ile beslenen CNN kullanılmıştır. Çalışmamızda CNN yerine yapay sinir ağı tercih edilmiştir. Giriş verileri eşitlik 3.1'de gösterilen, TurtleBot3 Burger modelinin LIDAR ve IMU sensörlerinden alınan mesafe ve konuma bağlı verilerdir.

MKS modeline göre ajan bulunulan durumda bir eylem seçerek, eylemi uygular ve buna karşılık çevre, yeni durum verisi ile ödül değerlerini verir. Bu yapıya göre bulunulan durum verisi Q ağına beslenir. Buna karşılık ağ, eylem adedi kadar çıkış değeri üretir. Çıkış değerleri Q değerlerimizdir. Ajan en yüksek değeri veren çıkışa ait eylemi uygular.

Bu yapıya göre eşitlik (3.1)'de gösterilen durum verisi için Q ağı 28 girişe ve eylem sayısı olan 5 çıkışa sahiptir. Ayrıca 64 nörondan oluşan iki gizli katman bulunmaktadır.

4.2. Deneyim Tekrarı

Q ağı, deneyim tekrarı olarak bilinen teknik kullanılarak eğitilir. Buna göre ajanın deneyimleri her zaman adımında birçok bölüm boyunca tekrar hafızasında toplanır $D = e_1, \dots, e_n$. Algoritmanın iç döngüsü sırasında, sayısı başlangıçta belirlenen rastgele toplu örnek alınarak Q ağı bu verilerle eğitilir. Tekrar belleğine atılan bir adımdaki veri örneği Tablo 4.1'de görülmektedir.

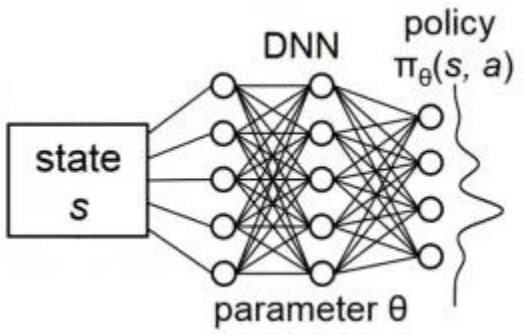
Tablo 4.1. Bir adımdaki deneyimlerin tekrar belleğine atılması

State	Action	Reward	Next State	Terminal
[0.70423466, 0.74005955, 0.81091094, 2.53835011, 2.08871937, 1.86942446, 1.85200751, 1.97293305, 2.28401041, 0.8246907, 0.69811845, 0.66576654, 0.63827717, 0.70856136, 0.83682972, 2.27525067, 1.97955251, 1.84015298, 1.88248456, 2.07542706, 2.51872468, 0.82420069, 0.72306871, 0.69062769, -0.03, 1.25, 0.64, 12.]	3	2.5	[0.70505613, 0.73852164, 0.82384688, 2.54709363, 2.08225608, 1.89276588, 1.84579492, 1.97996962, 2.26392102, 0.83209527, 0.69971853, 0.65699041, 0.65621364, 0.68580681, 0.83433872, 2.26205015, 1.96482849, 1.85515583, 1.89609575, 2.09213305, 2.52850461, 0.81666934, 0.72897488, 0.69993395, 1.57, 1.25, 0.66, 12.]	False

4.3. Eylem Seçimi

Ajan eylem seçimini, açgözlü yaklaşım stratejisini dikkate alarak Q ağı ile yapar. Başlarda daha çok rastgele eylemler ile ortam keşfedilir. Bölümler ilerledikçe daha çok Q ağının

ürettiği Q değerlerine göre seçim yapılır. Mevcut durum ile beslenen Q ağı eylem sayısı kadar çıkış üretir. Çıkışlar PÖ'deki Q değerleridir. Çıkışlardan en yüksek değerli olana ait eylem seçilir. Bu süreç Şekil 4.1'de resmedilmiştir.



Şekil 4.1. Q ağı ile eylem seçimi

4.4. Q fonksiyonu

PÖ'de gelecekteki maksimum yaklaşık getiriyi hesaplamak için Q fonksiyonu kullanılır. Q fonksiyonu bir durumda yapılan eylemin Q değerini verir. DQN'de Q ağı tarafından üretilen Q değeri, mevcut durumda yapılan eylemin kalitesini verir. DQN algoritmasında, Q ağını eğitmek ve optimum değerlere ulaştırmak için çıkış değerini, Q fonksiyonu ile tahmin edilen beklenen maksimum getiri değerine yaklaştırmaya çalışırız.

4.5. Algoritma

DPÖ algoritma süreci Şekil 4.2'deki akış şeması ile gösterilmiştir. Akış şemasında kullanılan terimlerin açıklaması aşağıda listelenmiştir.

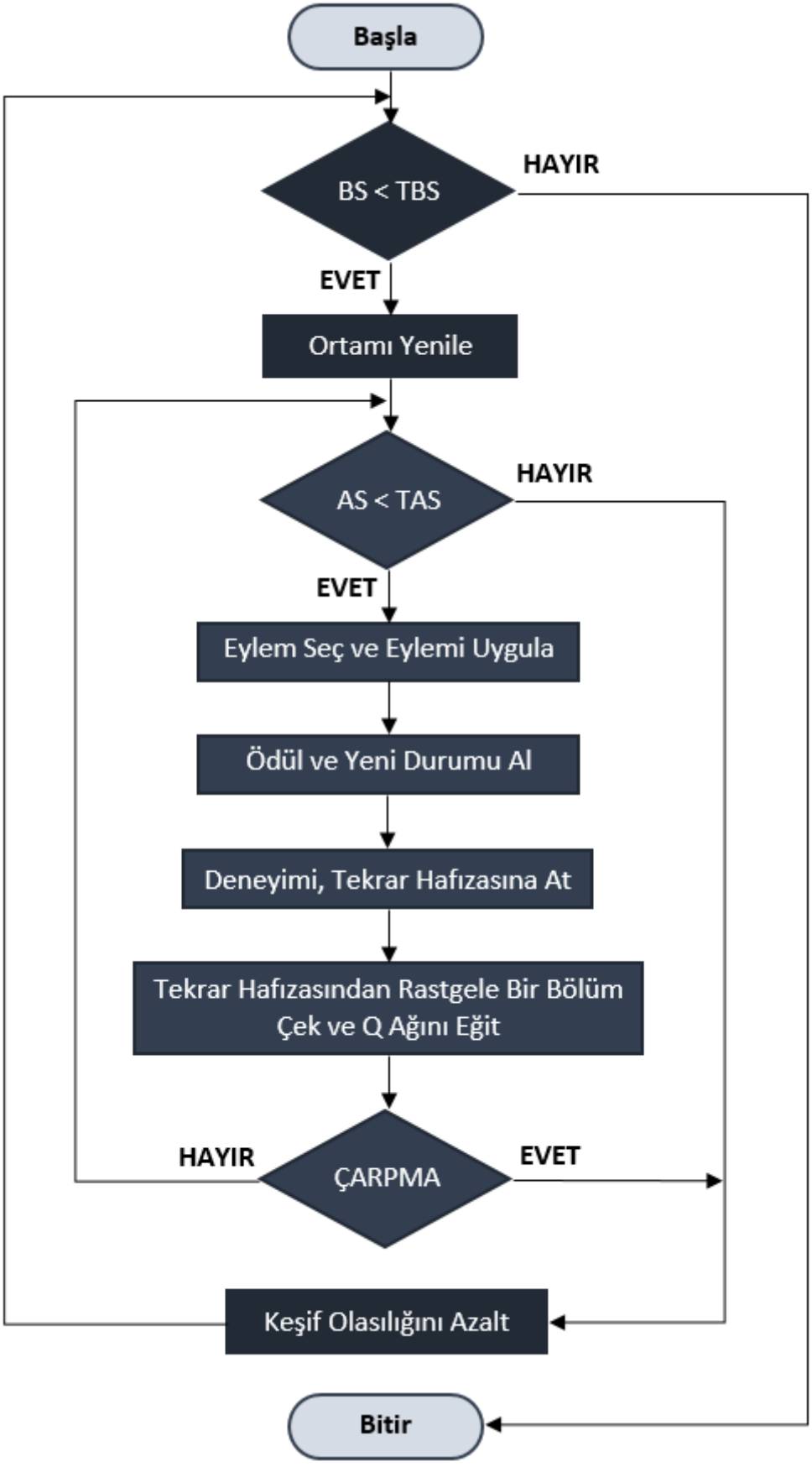
BS: Bölüm sayısı

TBS: Toplam bölüm sayısı

AS: Adım sayısı

TAS: Toplam adım sayısı

ÇARPMA: Çarpışma durumunda, bölümü sonlandırır.



Şekil 4.2. Uygulamada kullanılan DPÖ akış şeması

4.6. Q Ağının Eğitimi

Q ağı 28 giriş, 5 çıkıştan ve 64 nöronlu iki gizli katmandan oluşan bir yapay sinir ağıdır. Bu ağ her adımda (Mnih vd. 2013) makalesine göre denklem (4.1)'de görülen hedef değer fonksiyonu ve denklem (4.2)'de görülen kayıp fonksiyonuna göre eğitilir. denklem (4.2)'deki kayıp fonksiyonu diğer modeller (Mnih vd. 2015) ve (V. Hasselt vd., 2016) içinde aynıdır. Terminal değeri çarpışma ya da zaman aşımı durumunda "true", diğer durumlarda "false" değerini alır.

$$y = \begin{cases} r & ,terminal_{t+1}\ true\ ise \\ r + \gamma \max_{a'} Q(s', a'; \theta) & ,terminal_{t+1}\ false\ ise \end{cases} \quad (4.1)$$

$$L = (y - Q(s, a; \theta))^2 \quad (4.2)$$

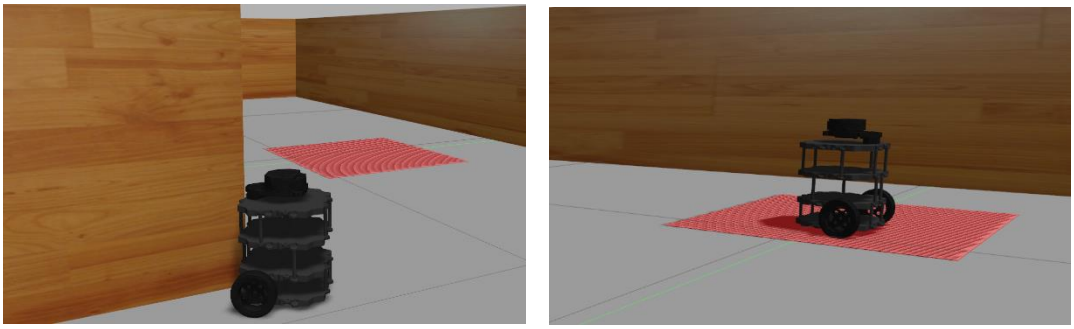
(Mnih vd. 2015) makalesine göre eylem seçimi için Q eylem-değer fonksiyonu ($Q(s_t, a; \theta)$) ve optimal hedef değerlerin tahmini için hedef eylem-değer fonksiyonu ($\hat{Q}(s_{j+1}, a'; \theta^-)$) kullanılır. Q ağı ($\theta$) her adımda eğitilir ve belirlenen bir 'C' adım sayısında Q ağının ağırlıkları hedef ağa aktarılır. Denklem (4.3) hedef değer fonksiyonudur. Tahmin için hedef ağın kullanıldığına (θ^-) dikkat edilmelidir.

$$y = \begin{cases} r & ,terminal_{t+1}\ true\ ise \\ r + \gamma \max_{a'} Q(s', a'; \theta^-) & ,terminal_{t+1}\ false\ ise \end{cases} \quad (4.3)$$

(V. Hasselt vd., 2016) makalesine göre eylem seçimi çevrimiçi ağa ($Q(s, a; \theta)$) göre yapılır. Optimal değer tahmini için hedef ağı ($Q(s, a; \theta^-)$) kullanılır. Hedef değer fonksiyonu denklem (4.4)'de görülmektedir.

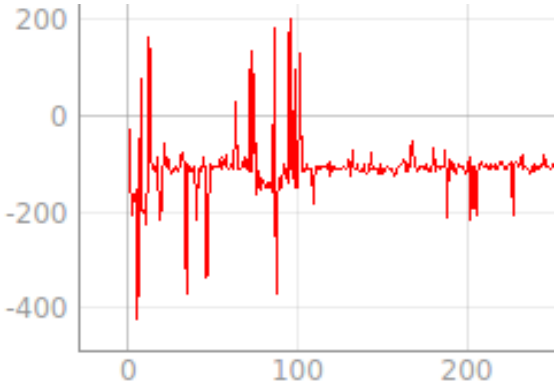
$$y = \begin{cases} r & ,terminal_{t+1} = true\ ise \\ r + \gamma Q\left(s', \operatorname{argmax}_{a'} Q(s', a'; \theta_t); \theta_t^-\right) & ,terminal_{t+1} = false\ ise \end{cases} \quad (4.4)$$

Q ağının eğitimi sürecinde ajan ortamı keşfetmek için başlarda daha çok rastgele hareketler yapar. Bu aşamada birçok defa engelle çarpışma yaşanır ve ajan olumsuz ödül alır. Ödül sistemine göre ajan; çarpışmalarda -150 ceza puanı, hedefi bulduğunda ise 200 ödül puanı alır. Ayrıca hedefe yaklaştığı her adımda küçük bir ödül puanı uzaklaştığı her adımda ise küçük bir ceza puanı alır. Ajanın amacı bölüm boyunca kümülatif ödül miktarını artırmaktır. Şekil 4.3'de eğitim sürecinde ajanın engele çarptığı ve hedef noktayı bulduğu sahnelerden örnekler paylaşılmıştır.



Şekil 4.3. TurtleBot3'ün engele çarpma anı (sol) ve hedefi bulma anı (sağ)

Eğitimin ilerleyen aşamalarında Q ağının ağırlıklarının en uygun noktaya yaklaşması ve toplanan ödül miktarının artması beklenir. Grafik 4.1'de toplam ödül değişiminden bir örnek sunulmuştur. Bu grafik, eğitimin ilk 250 bölümüne aittir. Grafikte dikey eksen toplam ödül değerini yatay eksen ise bölüm sayısını ifade eder. Genelde toplam ödül miktarının sıfırın altında kaldığı görülmektedir. Keşif aşaması için bu normaldir. Sıfırın altında kalan değerlerden ajanın hedeften uzaklaştığı ve engele çarptığı anlamını çıkarabiliriz. Pozitif değerlerde ise ajanın hedefe yaklaştığı ve hedefi bulduğu bölümleri görebiliriz. Ajanın hedefe yaklaşması ve bir defa bulması sonucunda toplayacağı ödül miktarının 200'ün üzerinde olmasını bekleriz. Ajan bölüm boyunca artarda hedef noktayı bularak toplam ödül miktarını artırabilir.

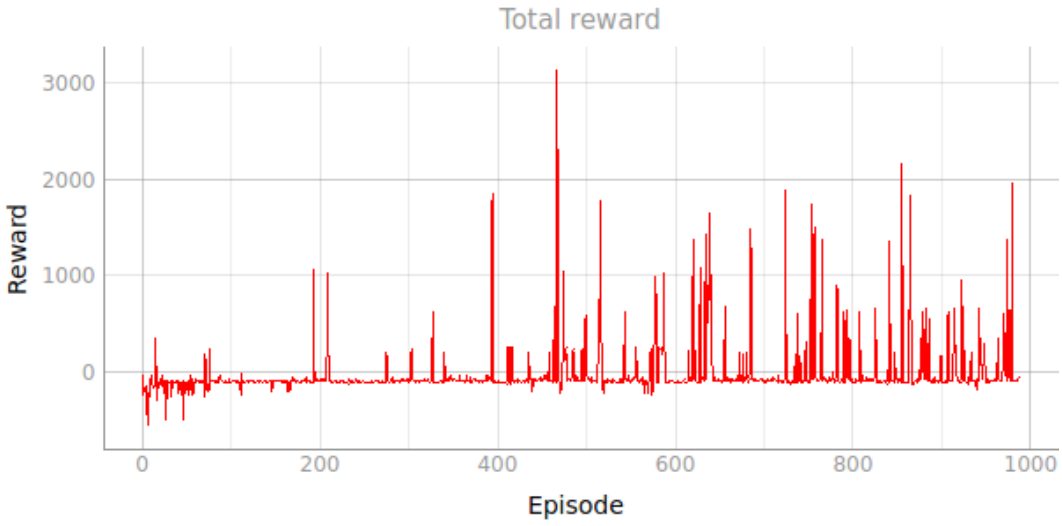


Grafik 4.1. Eğitim sürecinin başlarında toplam ödül değişiminden bir örnek

Grafik 4.2, 4.3 ve 4.4'de (Mnih vd. 2013,2015) ve (V. Hasselt vd., 2016) modelleri kullanılarak GazeboStage1 ortamında yapılan eğitim sürecinde elde edilen bölüm başına birikmiş ödül değişimleri gösterilmektedir. Grafiklerde dikey eksen toplam ödül değerini, yatay eksen ise bölüm sayısını göstermektedir.

Oluşturulan ortamda modeller ile yapılan eğitimler sonucunda, bölümler ilerledikçe robotun hedef noktaları bulmayı öğrendiği gözlemlenmiştir. (Mnih vd. 2013) ile yapılan eğitimin grafiğinde birikmiş ödül değerlerinin diğerlerine göre daha iyi olduğu görülmektedir. Eğitilen modeller ile yapılan test sonuçları da Bölüm 5'de paylaşılmıştır.

Grafik 4.2. GazeboStage1 ortamında (Mnih vd. 2013) modelinin toplam ödül değişimi



Grafik 4.3. GazeboStage1 ortamında (Mnih vd. 2015) modelinin toplam ödül değişimi

Grafik 4.4. GazeboStage1 ortamında (V. Hasselt vd., 2016) modelinin toplam ödül değişimi

5. SONUÇLAR

Bu çalışmada, benzetim ortamında DPÖ ile gezgin robotlarda otonom hareket elde edilmesi üzerine çalışılmıştır. Makine öğrenmesi yöntemlerinden PÖ, defalarca deneme yanılma tekrarı gerektirir. Öğrenmeyi gerçek dünya ortamında gerçekleştirmek çok zordur, hatta imkânsızdır. Bu sebeple iyi bir ajan eğitmek ve eğitilen modeli gerçek ortama taşımak için benzetim ortamına ihtiyaç vardır.

Benzetim çalışmasında, DPÖ yöntemi ile eğitim yapmak ve doğrulamak için ROS'un Gazebo benzetim ortamı ile TurtleBot3 Burger gezgin robotu kullanılmıştır. Gazebo'da eğitim ve test için Şekil 3.4'de görülen GazeboStage1 ortamı oluşturulmuştur. TurtleBot3'ün kilometre sayacı ve IMU sensörü tarafından toplanan pozisyon ve mevcut konum bilgisi ile LIDAR sensörü tarafından taranan benzetim ortamı bilgileri kullanılarak DQN ajanı süreç boyunca çevre ile etkileşime girer ve Q ağı eğitilir. Eğitilen Q ağı ile aynı ortamda test işlemleri gerçekleştirilmiştir.

Kurgulanan yapıda gezgin robotumuz ortamda engellere çarpmadan rastgele yer değiştiren hedef noktaları bularak bölüm boyunca toplam ödül miktarını artırmaya çalışır. Engele çarparsa ya da 500 adım sonunda bölüm sonlandırılır. Robot, hedef noktayı buldukça 200 ödül puanı; engele çarparsa -150 ceza puanı alır. Ayrıca robotumuz hedef nokta doğrultusunda, hedefe yaklaştığı her adımda küçük bir ödül puanı alır. Hedeften uzaklaştıkça veya farklı doğrultularda yaptığı hareketlerde küçük bir ceza puanı alır. Amaç bölüm boyunca toplanan ödülü en yükseğe çıkarmaktır.

Üç farklı model ile eğitilen gezgin robotumuz test işlemlerinde, bir bölümde engellere çarpmadan artarda defalarca hedef noktayı bulmayı başarmıştır. Eğitim sonrası GazeboStage1 ortamında yapılan test sonuçları Grafik 5.1, 5.2 ve 5.3'de verilmiştir. Grafiklerde dikey eksen toplam ödül değerini, yatay eksen ise bölüm sayısını göstermektedir. Grafiklerden de görüleceği üzere birikmiş ödül değeri ilk bölümlerde bile pozitif değerlerdedir. Bu, ajanın eğitildiği ve ortam koşullarına uygun eylem seçilerek robotun uygun bir gezi seyri izlediğini göstermektedir.

Grafik 5.1, (Mnih vd. 2013) modeli ile yapılan test sürecini göstermektedir. Bölüm başına toplanan ödülün ortalama 2500 puan düzeyinde olduğu görülmektedir. Ödül sistemini dikkate alarak grafik incelendiğinde, robotun başarılı bir şekilde hedef noktaları artarda defalarca bulduğu anlaşılmaktadır. Bu model, eylem seçimi ve optimal değer tahmini için tek bir Q ağı kullanır. Diğer modellerin test grafiklerine bakarak, bu modelin birikimli ödül

değerlerinin daha yüksek, toplam ödül değişiminin de diğer modellere göre daha az olduğu ve daha kararlı sonuç verdiği görülmektedir. Ayrıca bölüm başına hedef noktaları daha çok bulduğu da anlaşılmaktadır.

Grafik 5.1. (Mnih vd. 2013) modelinin test sürecindeki toplam ödül değişimi

Grafik 5.2, (Mnih vd. 2015) modeli ile yapılan test sürecini göstermektedir. Grafik incelendiğinde bölüm boyunca toplanan ödüllerin ortalaması yaklaşık 2000 puan düzeyinde olduğu görülmektedir. Bu değerlerden hedef noktaların artarda defalarca bulunduğu anlaşılmaktadır. Bu modelde eylem seçimi ve optimal değer tahmini için iki farklı Q ağı kullanılır. Her "C" adım sayısında Q ağının ağırlıkları hedef Q ağına aktarılır.

Grafik 5.2. (Mnih vd. 2015) modelinin test sürecindeki toplam ödül değişimi

Grafik 5.3, (V. Hasselt vd., 2016) modeli ile yapılan test sürecini göstermektedir. Grafik incelendiğinde bölüm boyunca toplanan ödüllerin bazı bölümlerde 3000 puana kadar çıktığı ancak bazı bölümlerde engele çarparak sıfır puan toplandığı anlaşılmaktadır. Makalede bahsedilenin aksine uygulamamızda bu modelin daha kararsız sonuçlar ürettiği gözlemlenmiştir.

Grafik 5.3. (V. Hasselt vd., 2016) Modelinin Test Sürecindeki Toplam Ödül Değişimi

DPÖ yöntemi kullanılarak, TurtleBot3 gezgin robotu ile Gazebo benzetim ortamında yapılan çalışma sonucunda, gezgin robot bilinmeyen ortamdaki hedef noktaları otonom olarak artarda bulmayı başarmıştır. Benzetim ortamında eğitilen modelin gerçek dünyada kullanımı için çeşitli ayarlamalar yapmak gerekecektir. Ayrıca hedef noktaları belli olmayan, hedefin de ajan tarafından bulunduğu bir senaryo gerçek dünya problemlerine daha uygundur. Bir sonra ki araştırma konusu bu olabilir.

Bu çalışmanın kodlarına ve (Mnih vd. 2013) modeli ile bir ajanın eğitimine ve eğitilmiş ajanın benzetim ortamında test edilirken çekilmiş videolarına "https://github.com/computeachp/TurtleBot3-Navigation-by-DQN" bağından erişilebilir.

KAYNAKÇA

Amsters, R., & Slaets, P. (2020). Turtlebot 3 as a robotics education platform. In *Robotics in Education: Current Research and Innovations* 10 (pp. 170-181). Springer International Publishing.

Bellman, R. (1957) A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6, 679-684.

Bertsekas, D. P. (1987). Dynamic Programming: Deterministic and Stochastic Models. *Prentice-Hall, Englewood Cliffs, NJ.*

Bertsekas, D. P. (1996). Dynamic programming and optimal control. *Journal of the Operational Research Society*, 47(6), 833-833.

Christopher, M. B. (2006). Pattern recognition and machine learning. *Information science and statistics New York: Springer*.

David Silver (2023). *David Silver, UCL Course on RL, Lecture 4: Model-Free Prediction* [Erişim: 15.10.2023, https://www.davidsilver.uk/wp-content/uploads/2020/03/MC-TD.pdf]

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. (2018). An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning, 11*(3-4), 219-354.

Garrido Alvarez, L. (2019). DQN: Deep Q-Learning for Autonomous Navigation, *Kansas State University Undergraduate Research Conference*.

Gazebo Tutorial (2023a). *Beginner: Overview, What is Gazebo?*. [Erişim: 16.10.2023, https://classic.gazebosim.org/tutorials?cat=guided_b&tut=guided_b1]

Gazebo Tutorial (2023b). *Building Editor, Overview*. [Erişim: 18.10.2023, https://classic.gazebosim.org/tutorials?cat=build_world&tut=building_editor]

Hasselt, H. (2010). Double Q-learning. *Advances in neural information processing systems, 23*.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. The MIT Press, Cambridge, MA.

IMU (2023). *Inertial measurement unit, From Wikipedia, the free encyclopedia*. [Erişim: 13.10.2023, https://en.wikipedia.org/wiki/Inertial_measurement_unit]

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4, 237-285.

Kallenberg, L. (2011). Markov decision processes. *Lecture Notes. University of Leiden*, *428*.

Karakuzu, C., & Babuska, R., (2014). On-line Path Planing for Mobile Robots Based on Basic Heuristic Guidance. *The 15th Workshop of the EURO Working Group* "*EU/ME: The Metaheuristic Community*" (pp.147-152). İstanbul, Turkey

Karakuzu, C., & Babuska, R., (2021). On-line Path Planning for Swarm of Mobile Robots Based on Particle Swarm Optimization. *Bseu Journal Ofengineerıng Researchand Technology*, vol.2, no.2, 7-19.

Koenig, N., & Howard, A. (2004). Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)* (Vol. 3, pp. 2149-2154).

LIDAR (2023). *Light Detection and Ranging, From Wikipedia, the free encyclopedia*. [Erişim: 13.10.2023, https://tr.wikipedia.org/wiki/LIDAR]

Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM workshop on hot topics in networks* (pp. 50-56).

Martinez, A., & Fernndez, E. (2013). *Learning ROS for robotics programming*. Packt Publishing.

Memon, Q., Ahmed, M., Ali, S., Memon, A. R., & Shah, W. (2016). Self-driving and driver relaxing vehicle. In *2016 2nd international conference on robotics and artificial intelligence (ICRAI)* (pp. 170-174).

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529-533.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT Press.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY.

Quigley, M., Gerkey, B., & Smart, W. D. (2015). *Programming Robots with ROS: a practical introduction to the Robot Operating System*. O'Reilly Media. Inc.

ROBOTIS Official GitHub (ROBOTIS-GIT), (2022). *TurtleBot3 Machine Learning repository*. [Erişim: 21.12.2022, https://github.com/ROBOTIS-GIT/turtlebot3_machine_learning].

ROS Introduction, (2023). *What is ROS? From Wikipedia, the free encyclopedia*. [Erişim: 10.10.2023, https://wiki.ros.org/ROS/Introduction]

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics modelling, planning and control* (s. 10). Springer London.

Sung, T. T., Kim, C., Lee, K., & Sohn, C. B. (2018). Exploring Navigation using Deep Reinforcement Learning. *International Journal of Applied Engineering Research*, 13(19), 14447-14450.

Sutton, R. S. (1997). On the significance of Markov decision processes. In *Artificial Neural Networks—ICANN'97: 7th International Conference Lausanne, Switzerland, October 8–10, 1997 Proceeedings 7* (pp. 273-282). Springer Berlin Heidelberg.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3, 9-44.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

The Construct, The Beginner's Guide to ROS (The Construct) (2023). *What is the robot ROS API?*. [Erişim: 15.10.2023, https://www.theconstructsim.com/about-ros-robot-operating-system/]

Tsai, J., Chang, C. C., Ou, Y. C., Sieh, B. H., & Ooi, Y. M. (2022). Autonomous driving control based on the perception of a lidar sensor and odometer. *Applied Sciences*, 12(15), 7775.

Robotis E-Manual TurtleBot3 (TurtleBot3) (2023). *TurtleBot3 Burger Specifications*. [Erişim: 12.10.2023, https://emanual.robotis.com/docs/en/platform/turtlebot3/features/]

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 30, No. 1).

Van Otterlo, M., & Wiering, M. (2012). Reinforcement Learning and Markov Decision Processes. In *Reinforcement Learning: State-of-the-Art* (pp. 3-42). Berlin, Heidelberg: Springer Berlin Heidelberg.

Wang, S., Jia, D., & Weng, X. (2018). Deep reinforcement learning for autonomous driving. *arXiv preprint arXiv:1811.11329*.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England.

Watkins, C. J., & Dayan, P. (1992). Technical Note Q-learning. *Machine Learning*, 8, 279-292.

Winston, W. L. (1991). *Operations research: applications and algorithms*. Brooks/Cole.

Witten, I. H. (1977). Exploring, modelling and controlling discrete sequential environments. *International Journal of Man-Machine Studies*, 9:715–735.

Yapay Zeka Araştırma İnsiyatifi (yzai), (2023). Pekiştirmeli Öğrenme - Bölüm 3: Sonlu Markov Karar Süreçleri [Erişim: 10.09.2023, https://yz-ai.github.io/blog/pekistirmeli-ogrenme/sonlu-markov-karar-surecleri-bolum-3#politikalar-ve-de%C4%9Fer-fonksiyonlar%C4%B1-policies-and-value-functions-1]

Zamora, I., Lopez, N. G., Vilches, V. M., & Cordero, A. H. (2016). Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using Ros and Gazebo. *arXiv preprint arXiv:1608.05742*.