**T.C.**
**FIRAT ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

# İNSANSI ROBOTUN ENGELLERDEN KAÇINARAK YÜRÜMESİ İÇİN DERİN PEKİŞTİRMELİ ÖĞRENME ALGORİTMALARININ GELİŞTİRİLMESİ

**Nuri Köksal VAROL**

Yüksek Lisans Tezi

MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Mekatronik Mühendisliği Bilim Dalı

NİSAN 2022

T.C.
Fırat Üniversitesi
Fen Bilimleri Enstitüsü

Mekatronik Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

# İNSANSI ROBOTUN ENGELLERDEN KAÇINARAK YÜRÜMESİ İÇİN DERİN PEKİŞTİRMELİ ÖĞRENME ALGORİTMALARININ GELİŞTİRİLMESİ

Tez Yazarı
**Nuri Köksal VAROL**

Danışman
Prof. Dr. Ayşegül UÇAR

NİSAN 2022
ELAZIĞ

**T.C.**
**FIRAT ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

Mekatronik Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

| | |
|---|---|
| Başlığı: | İnsansı Robotun Engellerden Kaçınarak Yürümesi İçin Derin Pekiştirmeli Öğrenme Algoritmalarının Geliştirilmesi |
| Yazarı: | Nuri Köksal VAROL |
| İlk Teslim Tarihi: | 24.12.2021 |
| Savunma Tarihi: | 28.04.2022 |

# TEZ ONAYI

Fırat Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına göre hazırlanan bu tez aşağıda imzaları bulunan jüri üyeleri tarafından değerlendirilmiş ve akademik dinleyicilere açık yapılan savunma sonucunda OYBİRLİĞİ ile kabul edilmiştir.

| | | *İmza* |
|---|---|---|
| Danışman: | Prof. Dr. Ayşegül UÇAR<br>Fırat Üniversitesi, Mühendislik Fakültesi, Mekatronik Mühendisliği | Onayladım |
| Başkan: | Prof. Dr. Fatih TALU<br>İnönü Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği | Onayladım |
| Üye: | Prof. Dr. Ömür AYDOĞMUŞ<br>Fırat Üniversitesi, Teknoloji Fakültesi, Mekatronik Mühendisliği | Onayladım |

Bu tez, Enstitü Yönetim Kurulunun ...../...../20..... tarihli toplantısında tescillenmiştir.

*İmza*

Prof. Dr. Kürşat Esat ALYAMAÇ
Enstitü Müdürü

# BEYAN

Fırat Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım "İnsansı Robotun Engellerden Kaçınarak Yürümesi İçin Derin Pekiştirmeli Öğrenme Algoritmalarının Geliştirilmesi" Başlıklı Yüksek Lisans Tezimin içindeki bütün bilgilerin doğru olduğunu, bilgilerin üretilmesi ve sunulmasında bilimsel etik kurallarına uygun davrandığımı, kullandığım bütün kaynakları atıf yaparak belirttiğimi, maddi ve manevi desteği olan tüm kurum/kuruluş ve kişileri belirttiğimi, burada sunduğum veri ve bilgileri unvan almak amacıyla daha önce hiçbir şekilde kullanmadığımı beyan ederim.

28.04.2022

Nuri Köksal VAROL

# Önsöz

Bilim ve teknolojinin ilerlemesi ile insansı robotların ortaya çıkması, çeşitli soruları akıllara getirmiştir. İnsan gibi hareket edip, insan gibi düşünebilen robotlar yapılabilir mi diye akıllarda hayaller kurmaya ve nasıl yapılabilir sorusunu düşünmeye zorlayan günümüzde, yapay zekâ algoritmalarının gelişmesi ile insansı robota verilen bazı görevler, daha yüksek başarı ile yerine getirilmiştir. Sanal dünyanın popüler olması sonucunda veri artışları gözlemlenmiş ve daha iyi düşünebilen sistemleri ortaya çıkarabilmek için kayıt altına alınmıştır. Zamandan tasarruf etmek, daha ekonomik ürünler çıkarmak ve teknolojiye yön vermek için artan veri, bir amaca uygun olarak işlenerek kontrol altına alınmaya çalışılmaktadır. Bu sayede geleceğe yönelik tahminde bulunabilecek, karar verebilecek, mevcut koşullara göre kendine görev oluşturabilecek sistemler oluşmaktadır.

İnsansı robot, bulunduğu ortamın koşullarının bilinmesi durumunda bazı sensörlerin yardımı ile engelden kaçınarak yürümesi üzerine birçok çalışma vardır. Bu tezde, ortamdaki engellerin ve hedefin bilinmeyen bir konumda olması durumunda, insansı robotun, derin öğrenme yöntemleri ile her ihtimalin gözetilip, engellerden kaçması ve hedefe ulaşması amaçlanmıştır. Mevcut ortam 4mx4m boyutunda olup tarama alanı 16 m^2 olarak sınırlandırılmıştır. Bu boyutlar, eğitim süresinin artıp azalmasına sebep olana, mevcut donanım, malzeme, geliştirilen algoritmalar ve zamana bağlı olarak değiştirilebilir. Değişen koşullara uyum sağlayabilecek şekilde programlamaya imkân sunan derin Pekiştirmeli Öğrenme yöntemleri sayesinde, sensörler ve tasarlanan algoritmalar ile elde edilen verilerin değerlendirilmesi sonucunda hedefe ulaşılmıştır.

Yüksek lisans sürecimde, yaşanılan her olaya karşı neler yapılabileceğini öğreten, yönlendirmeleri ile bu tezin oluşmasını sağlayan danışmanım Prof. Dr. Ayşegül Uçar'a teşekkür ederim.

Tezin gerçek sahipleri olan, velinimetlerim, annem ve babama, bana sundukları imkânlar ve destekler ile tezime yaptıkları katkılar için şükranlarımı sunuyorum.

Bu tez çalışması, TÜBİTAK tarafından 117E589 numaralı proje ile desteklenmiştir. Teşekkür ederim.

**Nuri Köksal VAROL**

ELAZIĞ, 2022

# İÇİNDEKİLER

# ÖZET

İnsansı Robotun Engellerden Kaçınarak Yürümesi İçin Derin Pekiştirmeli Öğrenme Algoritmalarının Geliştirilmesi

**Nuri Köksal VAROL**

Yüksek Lisans Tezi

FIRAT ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

Mekatronik Mühendisliği Anabilim Dalı

Nisan 2022, Sayfa: xi + 52

İnsansı robotların gerçekleştirecekleri operasyonlar çevresel faktörlerin etkisi ile kritik derecede zor bir görev olarak kabul edilir. Bu tez, insansı robotların operasyonel kabiliyetlerini artırmaya yönelik olarak, çeşitli engellerin ve duvarların bulunduğu ortamlarda hedefe ulaşmayı sağlamak için olası bir kontrol yaklaşımını açıklamaktadır. İnsansı robot, gördüğü çevre elemanlarından alınan bilgiler, mevcut konum ve yöne göre derin pekiştirmeli öğrenme yöntemleri kullanılarak eğitilmiştir. Eğitimin gerçekleştiği ortamlarda, robota, bir veya birden fazla hedef verilmiştir. Tasarlanan kontrol yaklaşımları, Robot İşletim Sistemi (ROS) ve Webots benzetim ortamlarında test edilmiştir.

**Anahtar Kelimeler:** Derin Öğrenme 1, İnsansı Robot 2, Mekatronik 3, Pekiştirmeli Öğrenme 4, Robotik 5, Robot İşletim Sistemi (ROS) 6, Yapay Zekâ 7

# ABSTRACT


Development of Deep Reinforcement Learning Algorithms for Humanoid Robot to Walk by Avoiding Obstacles

Nuri Köksal VAROL

Master's Thesis

FIRAT UNIVERSITY
Graduate School of Natural and Applied Sciences

Department of Mechatronics Engineering

April 2022, Pages: xi + 52

The operations to be performed by humanoid robots are considered to be a critically difficult task due to the impact of environmental factors. This thesis describes a possible control approach to increase the operational capabilities of humanoid robots to ensure target achievement in environments with various obstacles and walls. The humanoid robot has been trained using deep reinforcement learning methods based on the information received from the environmental elements it sees, the current location and direction. In the environments where the training takes place, one or more targets are given to the robot. The designed control approaches have been tested in Robot Operating System (ROS) and Webots simulation environments.



**Keywords:** Artificial Intelligence 1, Deep Learning 2, Humanoid Robot 3, Mechatronics 4, Reinforcement Learning 5, Robotics 6, Robot Operating System (ROS) 7

# ŞEKİLLER LİSTESİ

Sayfa

# TABLOLAR LİSTESİ

# KISALTMALAR

## Kısaltmalar

| | |
|---|---|
| BSD | : Berkeley Yazılım Dağıtımı (Berkeley Software Distribution) |
| ESA | : Evrişimsel Sinir Ağları |
| DH | : Denavit-Hartenberg |
| DP | : Dinamik Programlama |
| DPÖ | : Derin Pekiştirmeli Öğrenme |
| DQA | : Derin Q Ağı |
| GİB | : Grafik İşlem Birimi (Graphics Processing Unit) |
| MKS | : Markov Karar Süreci |
| PÖ | : Pekiştirmeli Öğrenme |
| RGB | : Kırmızı-Yeşil-Mavi (Red-Green-Blue) |
| ROS | : Robot Operating System |
| URDF | : Birleşik Robot Açıklama Formatı (Unified Robot Description Format) |
| YSA | : Yapay Sinir Ağı |

# 1. GİRİŞ

Matematik ve mekaniğin birleşmesi ile ortaya çıkan robotların, yapay zekânın gelişmesi ile bazı alanlarda, insandan daha güçlü ve daha kabiliyetli olması için çalışılmaktadır. Alan Turing, 1950 yılında makinalar düşünebilir mi diye bir tez ortaya koymuş ve makine zekâsı ile ilgili bir test oluşturmuştur. Bir makinenin zekâya sahip olabilmesi için insan olduğuna ikna edebilmesi gerektiğini vurgulamıştır [1].

Yapay zekâ terimi, 1955 yılında, ilk kez, John McChorty tarafından kullanılmış, 1957 yılında Yapay Sinir Ağlarının (YSA'ların) temeli olan perseptron, Frank Rosenblott tarafından tanımlanmıştır. 1986 yılında geriye yayılım algoritması ile Yapay Sinir Ağlarının çok hızlı bir gelişim göstermesi ile daha insansı sistemler için zemin hazırlanmış oldu. 1998 yılında Yann LeCun tarafından, Evrişimsel Sinir Ağları (ESA'lar) ile rakamların sınıflandırılması ve sayısallaştırılması başarıldı. 2009 yılına kadar gelişim göstermeye devam eden yapay zekâda oluşan veri eksikliği sonucunda, 167 ülkenin iş birliği ile dünyanın en büyük görüntü veri kümesi olan IMAGENET ücretsiz olarak herkesin kullanımına açıldı. Donanımların gelişmesi ile daha da hazlı gelişen yapay zekâ ile 2011 yılında sanal asistan SİRİ tanıtılarak kendini gösterdi. Bu gelişmeler sonucunda giderek insana benzer sistemler, robotlar üzerinde gerçekleştirilmeye başlayınca, kendi kendine düşünebilen ve belirli görevleri başarmayı amaçlayan daha insansı robotlar ortaya çıkmış oldu.

1941 yılında, Isaac Asimov "Robot" kelimesinden "Robotik" kelimesini türeterek ilk kez kullandı. 1953 yılında Grey Walter, robot bir kaplumbağa geliştirdi. Oval şekilli bu kaplumbağanın hareket etmesi ve yön değiştirmesi iki motorla sağlanıyordu. Bu kaplumbağa robot, ışık detektörleri ile ışığı algılayıp, ışık şiddetine bağlı olarak ışık kaynağına doğru yöneliyordu. 2000 yılında Honda, yeni humanoid robotu Asimo'u dünyaya tanıttı [2]. Boston Dynamics tarafından 2013 yılında tanıtılan Atlas ile dengede kalabilme ve hareket serbestliği yönünden daha insansı tasarlandığı ve programlandığı görülmektedir.

Bu tezde, insansı robot olan robotis-op2 ile Robot İşletim Sistemi (ROS) ve Webots ortamında Derin Q Ağı (DQA) kullanarak, hedefe ulaşmasını sağlamak için uygulamalar yapılmıştır. Robotis-OP2, 20 aktüatöre sahip bir insansı robottur. İnsansı eylemlerin yapılabilmesi için çeşitli özelliklere sahiptir. Üzerinde bulunan dâhili bilgisayar, kamera, aktüatörler ve çeşitli sensörler robotun insansı özellikleri kazandırılmasına imkân tanımaktadır.

Robotun programlamasında kullanılan ROS, robot yazılımı yazmak için esnek bir çerçeve, donanım soyutlama, düşük seviye cihaz kontrolü, yaygın olarak kullanılan işlevlerin uygulanması, işlemler arasında mesaj geçişi ve paket yönetimi dâhil olmak üzere, bir işletim sisteminden bekleyeceğiniz hizmetleri sağlar [3]. ROS, birçok insan tarafından bakımı yapılan açık kaynak kodlu bir kaynaktır. Bu, ROS'u gerçekten esnek ve kullanıcının ihtiyaçlarına uyarlanabilir hale getirir. ROS, kodların yeniden kullanımını teşvik eder, böylece robotik geliştiricileri ve bilim

insanlarının her zaman tekerleği yeniden icat etmeleri gerekmez. Kodlar, depolardan hazır alınıp, geliştirilip, tekrardan yayınlanabilir [4]. DQA uygulanmadan önce, gerekli alt yapıyı hazırlayabilmek için, hedefe giden en uygun yolu, eylemleri ve durumları saptamak gerekmektedir. Bu yüzden, oluşturulan bir ortamda, robotun çevreyi tanıyabilmesi için gmapping paketi ile ortamın haritası, robotun klavye ile tüm çevreyi dolaşmasının ardından çıkarılmış, amcl ve movebase paketleri ile hedefe giden en kısa yol saptanmakla beraber hedefe hareket gerçekleştirilmiş, Rviz ve Gazebo ortamlarında, görsel olarak test edilmiştir. Bu paketler ile robotun çevreyi tanıması sağlanmıştır. Programlama dili olarak C++ ve Python kullanılmıştır.

Linux işletim sistemi, açık kaynak kodlu olup ROS gibi programlara uygun ortamı sunabilmektedir. Çalışma ortamları oluşturulup ayrı ayrı projeler yapılabilir ve proje için özel gereksinimler mevcut çalışma ortamına özel tanımlanabilir. Açık kaynak kodlu sistemler olduğu için kullanılan programların versiyon ve sürümleri, sürekli değişmektedir. Çalışma alanlarına özel sürüm ve versiyonlar yüklenerek, sistemdeki gelişmelerin, eski projelere etkisi azaltılmış olup, çalışır vaziyette kalması sağlanabilmektedir.

Robotun eğitilmesi için kullanılan derin öğrenme, karar verebilmeyi ifade eder. Bilinen yapay zekâ yöntemlerinden farkı, çoklu fonksiyonlar kullanabilmesidir. Fonksiyonlar, katmanlı olduğundan, programda bir işlem yükü meydana gelir. Bu problem, kullanılan cihaza zarar verebilir veya cihazın gücü yetmeyebilir. Aynı zamanda, çok fazla zaman kaybı oluşur. Derin öğrenme, daha fazla katman bulundurması bakımından daha fazla işlem yükü oluşturur. Cihazlarda bulunan Grafik İşlem Birimi (GPU-Grafik İşlem Birimi (Graphics Processing Unit)) modülü sayesinde işlemcinin daha odacıklı hale gelmesi sağlanmıştır. Bu modül, aynı anda yüzlerce katmanla işlem yapmayı kolaylaştırır. Derin öğrenme, diğer yapay zekâ yöntemlerinden daha fazla katmanlı yapısıyla, daha az hata oluşturur ve daha yüksek başarım elde eder. GPU modül masraflı bir ürün olduğundan, herkesin rahatlıkla derin öğrenme yapabilmesi için çeşitli yardımcı ortamlar tasarlanmıştır. Google Colab gibi ortamlarda, Google'ın ana sunucusundan GPU, kullanıma açılmış ve ücretsiz olarak kullanıma sunulmuştur. Keras, Tensorflow vb. gibi, matematiksel işlemleri, bilgisayarlı görmeye dayalı birçok kodu içinde barındıran kütüphaneler kullanılır. Her işletim platformunda çalışır. Derin öğrenmede, en çok kullanılan programlama dili Python' dur. Bu sayede oluşturulan çalışma ortamlarına yönlendirmeler yapılarak, daha az hata ile karşılaşılması sağlanmış olur.

Robotun eğitiminde Google Colab kullanılarak, insansı robotun, Webots simülasyon programında hazırlanan ortamdan aldığı görüntüler ile elde edilen engel ve hedef bilgileri, derin öğrenme için oluşturulan modele giriş olarak verilip, ileri, sağ ve sol olmak üzere, 3 çıkış bilgisi elde edilmiştir. Eğitim yüzde 70 başarı göstermiştir. Eğitim sonucunda, her durumda elde edilen eylem çifti olan Q tablosu, ele edilmiştir. Webots ortamında robotun engellere ve hedefe olan açı ve uzaklık bilgileri kontrol altındadır. Bu ortamda, hazır fonksiyonlar ile hedefe hareket gerçekleştirilmiştir.

Ortamlardan alınan görüntüler, OpenCV kütüphanesi içindeki çeşitli filtreler ve maskeler kullanılarak işlenmiş, engeller ve hedef tespit edilmiştir. 0 ve 1'den oluşan 300x300 boyutundaki resimler 90000 piksellik bir işlem yükü oluşturmaktadır. ROS'ta oluşturulan bir ortamda alınan görüntüler üzerinde engeller ve hedef, yuvarlak daireler içerisinde belirtilmiş ve daha anlaşılır görsel testler elde edilmiştir.

# 2. MARKOV KARAR SÜRECİ

Markov Karar Süreci (MKS), Pekiştirmeli Öğrenme (PÖ) problemini çözmek için matematiksel bir çerçeve sunmaktadır. Hemen hemen tüm PÖ problemleri MKS olarak modellenebilir. MKS, çeşitli optimizasyon problemlerini çözmek için yaygın olarak kullanılmaktadır [5].

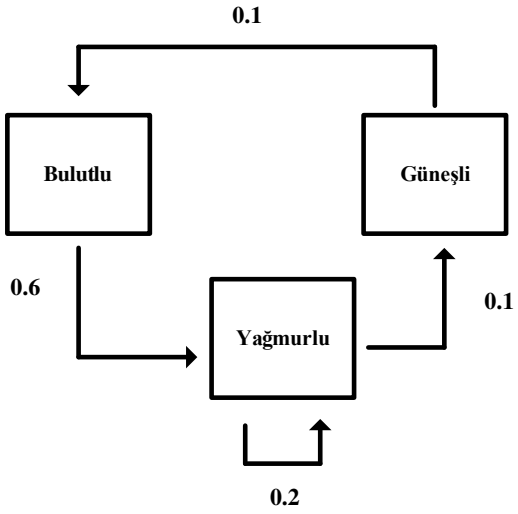
## 2.1. Markov Zinciri ve Markov Süreci

Markov, geleceğin yalnızca günümüze bağlı olduğunu, geçmişe bağlı olmadığını belirtir. Markov zinciri, yalnızca bir sonraki durumu kestirim etmek için önceki durumlara bağlayan ve sadece mevcut duruma bağlı olan bir modeldir. Gelecek, geçmişten şartlı olarak bağımsızdır. Örneğin, mevcut durumun bulutlu olduğu biliniyorsa, bir sonraki durumun yağmurlu olabileceği kestirim edilebilir. Ancak, Markov özelliği, tüm işlemler için geçerli değildir. Örneğin, zar atma (bir sonraki durum), zar üzerinde ne olursa olsun (mevcut durum) önceki sayıya bağlı değildir. Bir durumdan diğerine geçmeye, geçiş denir ve olasılığı geçiş olasılığı olarak adlandırılır. Geçiş olasılıkları, Tablo 2.1'de gösterildiği gibi formüle edebilir ve buna Markov tablosu denir [6].

**Tablo 2.1.** Markov tablosunun durum diyagram gösterimi [6]

| **Mevcut durum** | **Sonraki durum** | **Geçiş olasılığı** |
|---|---|---|
| Bulutlu | Yağmurlu | 0.6 |
| Yağmurlu | Yağmurlu | 0.2 |
| Güneşli | Bulutlu | 0.1 |
| Yağmurlu | Güneşli | 0.1 |

Markov zinciri, geçiş olasılığını gösteren bir durum diyagramı şeklinde temsil edilebilir. Şekil 2.1'de markov zincirinin geçiş olasılıklarına dayalı olarak gösterimi görülmektedir.




**Şekil 2.1.** Markov zincirinin geçiş olasılıklarına dayalı olarak gösterimi [7]

## 2.2. Markov Karar Süreci

MKS, Markov zincirinin bir uzantısıdır. Karar verme durumlarını modellemek için matematiksel bir çerçeve sağlar [8].

MKS beş önemli elemanla temsil edilir:

- Öğrenme etkeninin (insansı robot) içinde olabileceği durumlar kümesi (S)
- Bir durumdan diğerine geçmek için bir öğrenme etkeni (insansı robot) tarafından gerçekleştirilebilecek eylemler kümesi (A)
- Bazı eylemleri gerçekleştirerek S durumundan, Ś durumuna geçme olasılığı $P_{s\acute{s}}^{a}$
- Öğrenme etkeni (insansı robot) tarafından S durumundan, Ś durumuna geçmek için elde edilen bir ödül olasılığı $R_{s\acute{s}}^{a}$
- Anlık ve gelecekteki ödüllerin önemini kontrol eden bir indirim faktörü (γ) [7].

### 2.2.1. Ödül Sistemi

Bir PÖ ortamında, bir öğrenme etken (insansı robot) bir eylem gerçekleştirerek ortam ile etkileşime girer ve S durumundan, Ś durumuna geçer. Gerçekleştirdiği eyleme dayanarak, bir ödül alır. Bir ödül, sayısal bir değer ile ifade edilir. Örneğin bu tezde, iyi bir eylem için + 200 ve kötü bir eylem için -200 puan verilmiştir. Tasarlanan ortamda, iyi bir eylem, insansı robotun engele veya duvarlara çarpmayacak şekilde hareket ettiği yer olup, kötü bir eylem, robotun hareket ettiği, engele veya duvarlara çarptığı durumdur.

Robot, ani ödüller yerine ortamdan aldığı toplam ödül miktarını (kümülatif ödül) ençoklamaya çalışır. Öğrenme etkeninin (insansı robot), ortamdan aldığı toplam ödül tutarına dönüt denir. Böylece, öğrenme etkeni tarafından alınan toplam ödül (dönüt) miktarı:

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_T \quad (2.1)$$

olarak ifade edilir. Burada; $r_{t+1}$, bir durumdan diğerine geçmek için $a_0$ eylemini gerçekleştirirken, $t_0$ aşamasında bir öğrenme etkeni tarafından alınan ödüldür. $r_{t+2}$, bir durumdan diğerine geçmek için bir eylem gerçekleştirirken, $t_1$ aşamasında bir öğrenme etkeni tarafından alınan ödüldür. Benzer şekilde, $r_T$, S durumundan, Ś durumuna geçmek için bir eylemi gerçekleştirirken, T aşamasında zamana bağlı olarak, öğrenme etkeni tarafından alınan ödüldür.

### 2.2.2. Süreksiz ve Sürekli Görevler

Süreksiz görevler, uçbirim durumuna (son) sahip olan görevlerdir. PÖ'de bölümler, başlangıçtan son durumuna kadar öğrenme etkeni (insansı robot)-ortam etkileşimleri olarak kabul edilir.

Örneğin bu tezde, engel ve duvarların bulunduğu ortamda, insansı robot bir başlangıç konumuna yerleştirilir (başlangıç hali) ve hedefe gidinceye kadar hareket eder (son hali). Buna bir bölüm denir. Hedefe varıldığında, görev biter. Başka bölümlerde, başka görevler verilebilir. Yani, engel ve duvarların pozisyonları değiştirilip robota yeni bir hedef verilebilir veya robotun pozisyonu değiştirilebilir. Diğer bir deyişle, her bölüm diğerinden bağımsızdır. Ortamdaki nesnelerin pozisyonları ile ilgili tüm değişiklikler yeni bir süreksiz görevi oluşturur.

Sürekli bir görevde, bir uçbirim durumu yoktur. Sürekli görevler asla bitmez. Örneğin, bir kişisel yardım robotunun uçbirim durumu yoktur.

### 2.2.3. İndirim Faktörü

Bir öğrenme etkeninin hedefi, dönütü en üst düzeye çıkarmaktır. Bir süreksiz görev için, dönütü $R_t = r_{t+1} + r_{t+2} + \cdots + r_T$ olarak tanımlayabiliriz, burada T bölümün son halidir ve $R_t$ dönütü ençoklamaya çalışır.

Sürekli bir görev için herhangi bir durum olmadığından $R_t = r_{t+1} + r_{t+2} + \cdots$ olarak tanımlanır.

Dönüt, bir indirim faktörü ile aşağıdaki gibi yeniden tanımlanır:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \quad (2.2)$$

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (2.3)$$

İndirim faktörü, gelecekteki ve anlık ödüllere ne kadar önem verileceğine karar verir. İndirim faktörünün değeri 0 ila 1 arasındadır. 0 indirim faktörü, anlık ödülün daha önemli olduğu anlamına gelirken, 1 indirim faktörü, gelecekteki ödüllerin anında kazanılan ödüllerden daha önemli olduğu anlamına gelir.

0 indirim faktörü ile yalnızca ani kazanımları göz önünde bulundurarak asla öğrenemez; Benzer şekilde, indirim faktörü, gelecek ödül yoluna gidilerek öğrenilir. Dolayısıyla, indirim oranının en iyi değeri 0,2 ile 0,8 arasında kullanılır [7].

Kullanım durumuna bağlı olarak, anlık ödüllere ve gelecekteki ödüllere önem verilir. Bazı durumlarda, gelecekteki ödüller anlık ödüllerden daha fazla istenilir ve bunun tersi de geçerlidir.

### 2.2.4. Politika İşlevi

Π ile gösterilir. Politika işlevi, durumlardan eylemlere eşlemeyi gösterir ve $\pi(s)$: $S \rightarrow A$ ile ifade edilir. Temel olarak, bir politika işlevi her durumda hangi eylemin gerçekleştirileceğini gösterir. Hedefimiz, her durumda uygulanacak doğru eylemi belirleyen en iyi politikayı bulmaktır. Bu yapıldığı zaman ödül ençoklanmış olur [7].

### 2.2.5. Durum Değeri İşlevi

Değer işlevi, öğrenme etkenini bir Π politikası ile özel bir durumda olmasının ne kadar iyi olduğunu belirtir. Bir değer işlevi genellikle V(s) ile gösterilir. Bir politikayı takip eden durumun değerini gösterir.

$$V^{\pi}(s) = E_{\pi}[R_t \vee s_t = s] \tag{2.4}$$

ile gösterilir. Burada; E, Π politikasına göre S durumundan başlayarak, beklenilen dönütü ifade eder. Denklem (2.3)'deki $R_t$ değeri, denklem (2.4)'de yerine yazılırsa:

$$V^{\pi}(s) = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \vee s_t = s] \tag{2.5}$$

Burada durum değer işlevinin, seçilen politikaya bağlı olarak değişimi açıkça anlatılmıştır.

İki durumun olduğu ve bu durumların her ikisinin de politikaya uyduğu varsayılmıştır. Bu iki durum değerine göre, öğrenme etkeninin politikayı izleyen durumda ne kadar iyi olduğu anlatılmıştır. Değer ne kadar büyük olursa, durum o kadar iyi olur:

### 2.2.6. Durum Eylem Değeri İşlevi (Q işlevi)

Bir durum-eylem değeri işlevi, Q işlevi olarak adlandırılır. Bir öğrenme etkeninin, π politikalı bir durumda, özel bir eylemi uygulamasının ne kadar iyi olduğunu ifade eder. Q işlevi, Q(s) ile ifade edilir ve π politikasını takip eden bir durumda, gerçekleştirilen her eylemin değerlerini gösterir.

Q işlevi şu şekilde tanımlanır:

$$Q^{\pi}(s,a) = E_{\pi}[R_t \vee (s_t) = s, a_t = a] \quad (2.6)$$

Bu π politikasına göre, s durumunda gerçekleştirilen a eylemi için elde edilen dönütü verir. Denklem (2.6)'deki $R_t$ değeri yerine, denklem (2.3) yazılırsa:

$$Q^{\pi}(s,a) = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|s_t = s, a_t = a] \quad (2.7)$$

elde edilir.

Değer işlevi, bir durumun iyiliğini belirtirken, Q işlevi bir durumdaki bir eylemin iyiliğini belirtir. Tablo 2.2'te, durum değer işlevi için bir Q tablosu verilmiştir.

**Tablo 2.2.** Q tablosuna göre Durum-Eylem ilişkisi ve değeri

| **Durum** | **Eylem** | **Değer** |
|---|---|---|
| Durum 1 | Eylem 1 | 0.03 |
| Durum 1 | Eylem 2 | 0.02 |
| Durum 2 | Eylem 1 | 0.5 |
| Durum 2 | Eylem 2 | 0.9 |

Q tablosu, tüm olası durum eylem çiftlerinin değerini gösterir. Dolayısıyla, bu tabloya bakarak, değeri yüksek olduğu için, 1. durumda 1. eylemi ve 2. durumda 2. eylemi gerçekleştirmenin, değeri daha iyi bir sonuca götürebileceği görülmektedir.
Değer işlevi V(S) veya Q işlevi, Q(S,a) kısaca; değer tablosu ve Q tablosu olarak kullanılır.

### 2.2.7. Bellman Denklemi ve Eniyileme

Amerikalı matematikçi Richard Bellman'ın adını taşıyan Bellman denklemi, MKS'yi çözmek için kullanılır. PÖ'de her yerde bulunur. MKS'yi çözmek demek, bu aslında en iyi politikaları ve değer işlevlerini bulmak demektir. Farklı politikalara göre birçok farklı değer işlevi olabilir. En iyi değer işlevi $V^{\pi}(s)$, diğer tüm değer işlevleriyle karşılaştırıldığında en yüksek değeri veren değerdir:

$$V^{*}(s) = max_{\pi} V^{\pi}(s) \quad (2.8)$$

Benzer şekilde, en iyi politika, en iyi değer işlevi ile sonuçlanan politikadır. En iyi değer işlevi $V^{*}(s)$, diğer tüm değer işlevleriyle karşılaştırıldığında en büyük dönüt ile sonuçlanır, yani daha yüksek bir değere sahip olduğu için Q işlevinin en büyük değeri olacaktır. Böylece, en iyi değer işlevi Q işlevi en büyük alınarak aşağıdaki şekilde hesaplanır:

$$V^*(s) = max_a Q^\pi(s,a) \tag{2.9}$$

Değer işlevi için Bellman denklemi:

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{ś} P^a_{sś} [R^a_{sś} + \gamma V^\pi(ś)] \tag{2.10}$$

Bir durumun değeri ile ardışık hali arasındaki değer, bütün olasılıkların ortalaması arasındaki özyinelemeli ilişkiyi gösterir.
Benzer şekilde, Q işlevi için Bellman denklemi aşağıdaki gibi gösterilir:

$$Q^\pi(s,a) = \sum_{ś} P^a_{sś} [R^a_{sś} + \gamma \sum_{á} Q^\pi(ś,á)] \tag{2.11}$$

(2.9)'deki denkleme, (2.11)'deki denklemi yerine yazılırsa:

$$V^*(s) = max_a \sum_{ś} P^a_{sś} [R^a_{sś} + \gamma \sum_{á} Q^\pi(ś,á)] \tag{2.12}$$

Yukarıdaki denkleme, Bellman'ın optimizasyon denklemi denir.

### 2.2.8. Bellman Denkleminin Değer ve Q İşlevleri İçin Türetilmesi

$P^a_{sś}$, bir durumdan diğerine geçme olasılığı olarak tanımlanır.
Bir a eylemi gerçekleştirilirken:

$$P^a_{sś} = pr(s_{t+1} = ś | s_t = s, a_t = a) \tag{2.13}$$

$R^a_{sś}$, bir durumdan diğerine geçerek alınan muhtemel ödül olarak tanımlanır.
Bir a eylemi gerçekleştirilirken:

$$R^a_{sś} = E(R_{t+1} | s_t = s, s_{t+1} = ś, a_t = a) \tag{2.14}$$

$$= \gamma E_\pi [\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \vee s_{t+1} = ś] \tag{2.15}$$

Değer işlevi aşağıdaki gibi temsil edilir:

$$V^\pi(s) = E_\pi[R_t | s_t = s] \tag{2.16}$$

$$V^\pi(s) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots | s_t = s] \tag{2.17}$$

İlk ödülü alarak değer işlevi yeniden yazılırsa:

$$V^\pi(s) = E_\pi[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \vee s_t = s] \tag{2.18}$$

Değer işlevindeki beklentiler ortamda mevcut ise, politikasıyla bir eylem gerçekleştirerek beklenen getiriyi belirler. Bu nedenle, tüm olası eylemleri ve ödülleri aşağıdaki gibi özetleyerek beklenti açıkça yeniden yazılabilir:

$$E_\pi[r_{t+1}|s_t = s] = \sum_a \pi(s,a) \sum_{ś} P^a_{sś} R^a_{sś} \quad (2.19)$$

(2.15)'deki $R^a_{sś}$ değeri, (2.19)'da yerine yazılırsa:

$$\sum_a \pi(s,a) \sum_{ś} P^a_{sś} \gamma E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \vee s_{t+1} = ś] \quad (2.20)$$

Benzer şekilde, $r_{t+1}$ değerini, denklem (2.3)'ün yerine aşağıdaki gibi yazarsak:

$$E_\pi[\gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \vee s_{t+1} = s] \quad (2.21)$$

Böylece, hedef denklemi şöyle olur:

$$E_\pi[\gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \vee s_{t+1} = s] = \sum_a \pi(s,a) \sum_{ś} P^a_{sś} \gamma E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \vee s_{t+1} = ś] \quad (2.22)$$

Şimdi, değer işlevindeki (2.18) denklemi (2.21)'deki denkleme eşitlenirse:

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{ś} P^a_{sś} \left[R^a_{sś} + \gamma E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \vee s_{t+1} = ś]\right] \quad (2.23)$$

$E_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \vee s_{t+1} = ś]$ yerine $V^\pi(ś)$, denklem (2.18)'de yerine yazılırsa, son değer işlevi aşağıdaki gibi olur:

$$Q^\pi(s,a) = \sum_{ś} P^a_{sś} [R^a_{sś} + \gamma \sum_{á} Q^\pi(ś, á)] \quad (2.12)$$

Hem değer işlevi hem de Q işlevi için bir Bellman denklemi olduğuna göre, en iyi politikalar hesaplanabilir.

### 2.2.9. Dinamik Programlama

Dinamik Programlama (DP), karmaşık problemleri çözmek için kullanılan bir yöntemdir. DP'de, karmaşık problemleri birer birer çözmek yerine, sorun basit alt problemlere ayırılır, sonra her alt problem için çözüm hesaplanır ve saklanır. Aynı alt problem ortaya çıkarsa, yeniden hesaplanamaz, bunun yerine önceden hesaplanmış çözüm kullanılır. Bu nedenle DP, hesaplama süresini büyük ölçüde en aza indirir. Bilgisayar bilimi, matematik, biyoenformatik, vb. dahil olmak üzere çok çeşitli alanlarda uygulamalara sahiptir.

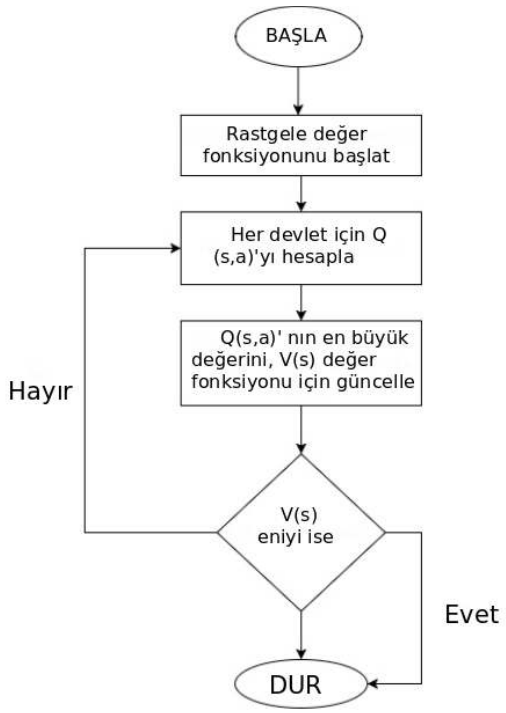
Bellman denklemi, iki güçlü algoritma kullanarak çözülür:

• Değer yineleme

• Politika yineleme

**Değer Yineleme**

Değer yinelemesinde rastgele bir değer işlevi ile başlanır. Rastgele değer işlevi en iyi bir işlev olmayabilir, bu yüzden en iyi değer işlevini bulana kadar yinelemeli olarak yeni geliştirilmiş bir değer işlevi aranır. En iyi değer işlevini bulduktan sonra, kolayca en iyi bir politika türetilebilir. Şekil 2.2'de değer yinelemenin akış diyagramı gösterilmektedir.




**Şekil 2.2.** Değer yenileme akış diyagramı [7]

Değer yinelemesine dâhil olan adımlar aşağıdaki gibidir:

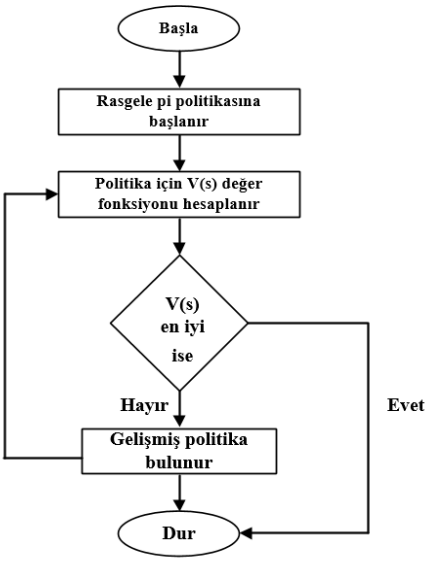
- İlk olarak, her durum için rastgele değer olan rastgele değer işlevi başlatılır.
- Daha sonra, Q(s, a) durumunun tüm çiftleri için, Q işlevi hesaplanır.
- Daha sonra, değer işlevi, Q(s, a) değerindeki maksimum değerle güncellenir.
- Değer işlevindeki değişiklik çok küçük olana kadar bu adımlar tekrar edilir.

**Politika Yineleme**

Değer yinelemenin aksine, politika yineleme için rastgele politika ile başlanır, sonra politikanın değer işlevi bulunur; eğer değer işlevi en iyi değilse, o zaman yeni geliştirilmiş politika bulunur. En iyi politikayı bulana kadar bu işlem tekrarlanır.

Politika yinelemede iki adım vardır:

1. Politika değerlendirme: Rastgele kestirim edilen bir politikanın değer işlevinin değerlendirilmesi.
2. Politika iyileştirme: Değer işlevini değerlendirdikten sonra, en iyi değilse, yeni bir iyileştirilmiş politikanın bulunması için gereken akış diyagramı, Şekil 2.3'te gösterilmiştir:




**Şekil 2.3.** Politika yenileme akış diyagramı [7]

Politika yinelemesine dâhil olan adımlar şunlardır:

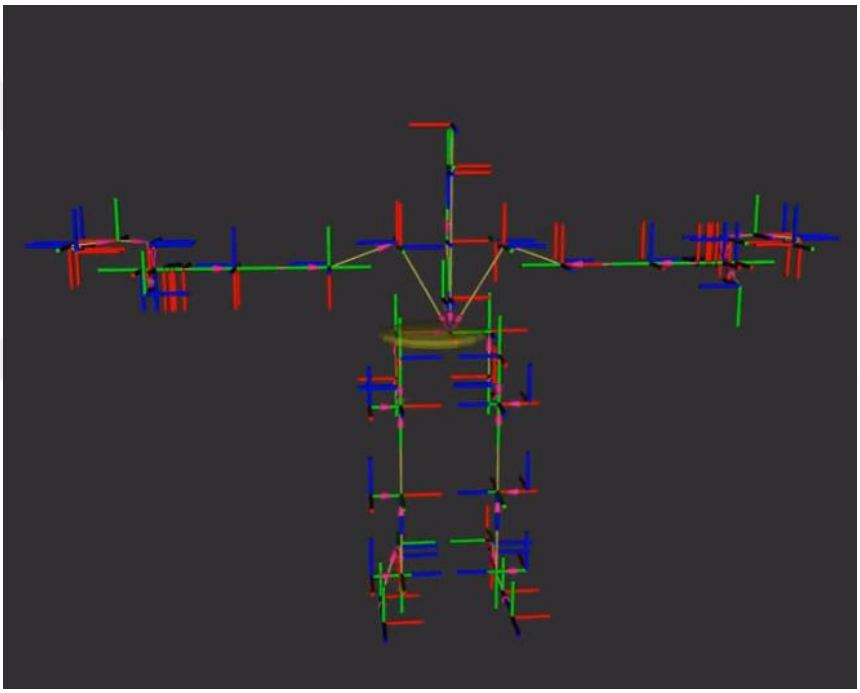
1. İlk olarak, bazı rasgele politikalar başlatılır.
2. Daha sonra bu rastgele politika için değer işlevi bulunur ve politikanın en iyi olduğu kontrol edilir.
3. Eğer uygun değilse, politika geliştirme adımı ile yeni bir geliştirilmiş politika bulunur.
4. En iyi politika elde edilene kadar bu adımlar tekrar edilir.

# 3. İLERİ VE TERS KİNEMATİK HESAPLAMALARI

Bu çalışmada, robotun hareketlerini kontrol etmek için robotun ileri ve ters kinematik hesaplamaları yapılmıştır.

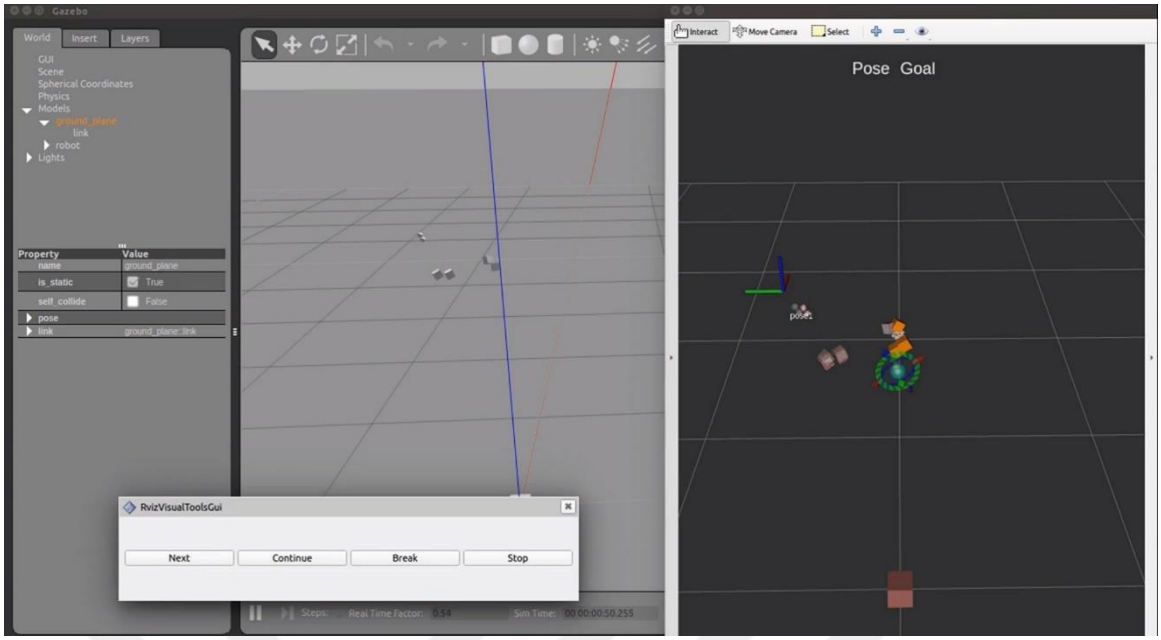
Bir kinematik kontrolörün amacı, zamanın fonksiyonu olarak konum ve hız profilleri ile tanımlanan bir yörüngeyi takip etmek demektir [9].

Harekete etki eden eklem-bağlantı çiftlerinin, koordinatlarına, boyutlarına, şekline ve hareket serbestliğine göre DH parametreleri çıkarılarak incelenmelidir. Parametreler çıkarılırken, URDF dosyasından faydalanılmalıdır. URDF dosyasından elde edilebilecek olan eklem-bağlantı çiftlerinin yönleri ve hareket ekseni dâhil olmak üzere gösterilebilecek dönüşüm iskelet haritası, Şekil 3.1'de gösterilmektedir.



**Şekil 3.1.** İnsansı robotun dönüşüm iskelet haritası gösterimi

Hareket etmeyi sağlayan bir bacaktaki eklem-bağlantı çiftlerinin kontrolünü pekiştirmek için hazırlanan URDF dosyasının Rviz ve Gazebo programları ile görseli, Şekil 3.2'de gösterilmektedir.




**Şekil 3.2.** Temsili bir bacağın eklem-bağlantı çiftlerinin Rviz ve Gazebo'da gösterilmesi [10]

Şekil 3.2'de hazırlanan temsili bacağın hareketini, sanal ortamda kontrol edebilmek için hesaplanan DH parametreleri, Tablo 3.1'de gösterilmektedir.

**Tablo 3.1.** DH parametreleri

| **Link** | θ | α | a | d |
|---|---|---|---|---|
| 1 | 0 | 0 | $a_1 = 1.2673$ | $d_1 = 0.1199$ |
| 2 | $\theta_2$ | -90 | $a_2 = 0.0312$ | $d_2 = 0.0482$ |
| 3 | $\theta_3 + 90$ | 90 | $a_3 = 0.041$ | $d_3 = 0.03$ |
| 4 | $\theta_4$ | -90 | $a_4 = 0.0312$ | $d_4 = 0.208$ |
| 5 | $\theta_5$ | 90 | $a_5 = 0.0317$ | $d_5 = 0.0334$ |
| 6 | $\theta_6$ | -90 | 0 | $d_6 = 0.1617$ |

Parametreler ayarlandığına göre rotasyon matrisleri ile birlikte homojen dönüşüm matrisleri hesaplanabilir. Rotasyon matrisleri, eklemin hareket serbestliğini ve açı değerini bulmamıza yarar. Bu sayede eklemleri kontrol etmemiz için gereklidir. Aşağıda rotasyon matrisleri görülmektedir.

$$R_1^0 = [C\theta_1 \; -S\theta_1 \; 0 \; S\theta_1 \; C\theta_1 \; 0 \; 0 \; 0 \; 1\,] * [1\,0\,0\,0\,1\,0\,0\,0\,1\,] = [C\theta_1 \; -S\theta_1 \; 0 \; S\theta_1 \; C\theta_1 \; 0 \; 0 \; 0 \; 1\,]$$

$$R_2^1 = [C\theta_2 \; -S\theta_2 \; 0 \; S\theta_2 \; C\theta_2 \; 0 \; 0 \; 0 \; 1\,] * [1\,0\,0\,0\,0\,1\,0\,-1\,0\,] = [C\theta_2 \; 0 \; -S\theta_2 \; S\theta_2 \; 0 \; C\theta_2 \; 0 \; -1\,0\,]$$

$$R_3^2 = [C\theta_3\ -S\theta_3\ 0\ S\theta_3\ C\theta_3\ 0\ 0\ 0\ 1\ ] * [1\ 0\ 0\ 0\ 0\ \ -1\ 0\ 1\ 0\ ] = [C\theta_3\ 0\ S\theta_3\ S\theta_3\ 0\ -C\theta_3\ 0\ 1\ 0\ ]$$

$$R_4^3 = [C\theta_4\ -S\theta_4\ 0\ S\theta_4\ C\theta_4\ 0\ 0\ 0\ 1\ ] * [1\ 0\ 0\ 0\ 0\ 1\ 0 - 1\ 0\ ] = [C\theta_4\ 0\ -S\theta_4\ S\theta_4\ 0\ C\theta_4\ 0\ \ -1\ 0\ ]$$

$$R_3^2 = [C\theta_5\ -S\theta_5\ 0\ S\theta_5\ C\theta_5\ 0\ 0\ 0\ 1\ ] * [1\ 0\ 0\ 0\ 0 - 1\ 0\ 1\ 0\ ] = [C\theta_5\ 0\ S\theta_5\ S\theta_5\ 0\ -C\theta_5\ 0\ 1\ 0\ ]$$

$$R_6^5 = [C\theta_6\ -S\theta_6\ 0\ S\theta_6\ C\theta_6\ 0\ 0\ 0\ 1\ ] * [1\ 0\ 0\ 0\ 0\ 1\ 0 - 1\ 0\ ] = [C\theta_6\ 0\ -S\theta_6\ S\theta_6\ 0\ C\theta_6\ 0 - 1\ 0\ ]$$

Homojen dönüşüm matrisinin genel gösterimi ve hesaplanan denklemler [11].

$$H_1^0 = [R_1^0\ d_1^0;\ 0\ 0\ 0\ 1\ ]$$

$$H_1^0 = [C\theta_1\ -S\theta_1\ 0\ 126.73;\ S\theta_1\ C\theta_1\ 0\ 0;\ 0\ 0\ 1\ 11.89; 0\ 0\ 0\ 1\ ]$$

$$H_2^1 = [C\theta_2\ 0\ -S\theta_2\ 3.12;\ S\theta_2\ 0\ C\theta_2\ 0;\ 0 - 1\ 0\ 4.82; 0\ 0\ 0\ 1\ ]$$

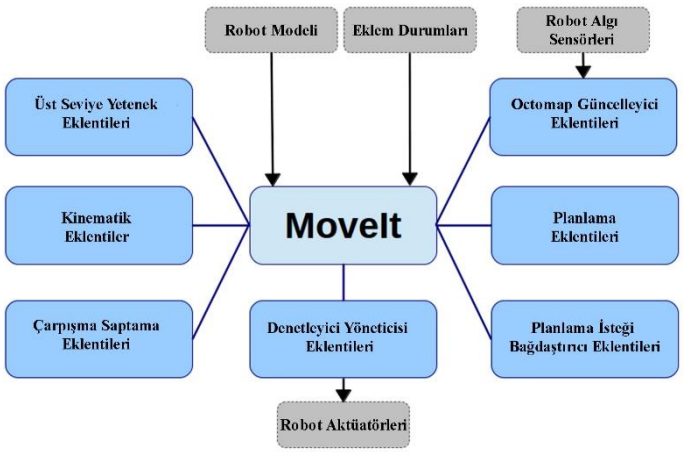
$$H_3^2 = [C\theta_3\ 0\ S\theta_3\ 4.1;\ S\theta_3\ 0 - C\theta_3\ 0;\ 0\ 1\ 0\ 3; 0\ 0\ 0\ 1]$$

$$H_4^3 = [C\theta_4\ 0\ -S\theta_4\ 3.12;\ S\theta_4\ 0\ C\theta_4\ 0;\ 0 - 1\ 0\ 20.8; 0\ 0\ 0\ 1]$$

$$H_5^4 = [C\theta_5\ 0\ S\theta_5\ 3.17;\ S\theta_5\ 0 - C\theta_5\ 0;\ 0\ 1\ 0\ 3.34; 0\ 0\ 0\ 1]$$

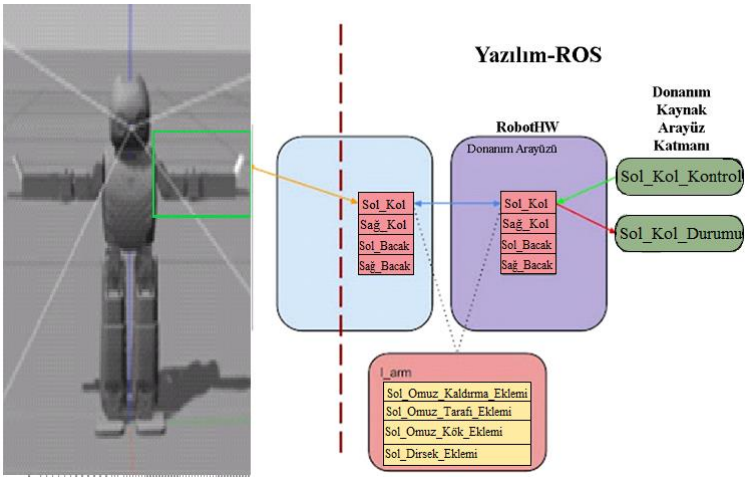
$$H_6^5 = [C\theta_6\ 0\ -S\theta_6\ 0;\ S\theta_6\ 0\ C\theta_6\ 0;\ 0 - 1\ 0\ 14.17; 0\ 0\ 0\ 1]$$

Rotasyon matrisi, Matlab ortamında hesaplanmış ve temsili bacağın kontrolü sağlanmıştır. MoveIT paketinin bize sunduğu gelişmiş bir arayüz ile oluşturduğumuz URDF dosyasındaki verilerden yola çıkarak, bir rehber sunar. Bu sayede, bazı matematiksel yükten kurtulmuş oluruz. Bu arayüzde, kinematik eklentiler, robot-nesne çarpışma eklentileri, bazı yetenek eklentileri, planlama eklentileri gibi kolaylık sağlayan avantajlar bulunur. Elde ettiğimiz eklentilerde, ayarlama yapılarak eylemlerin gerçekleştirileceği ortam hazırlanmış olur. Şekil 3.3'te, MoveIT ile eklenti bağlantıları görülmektedir.



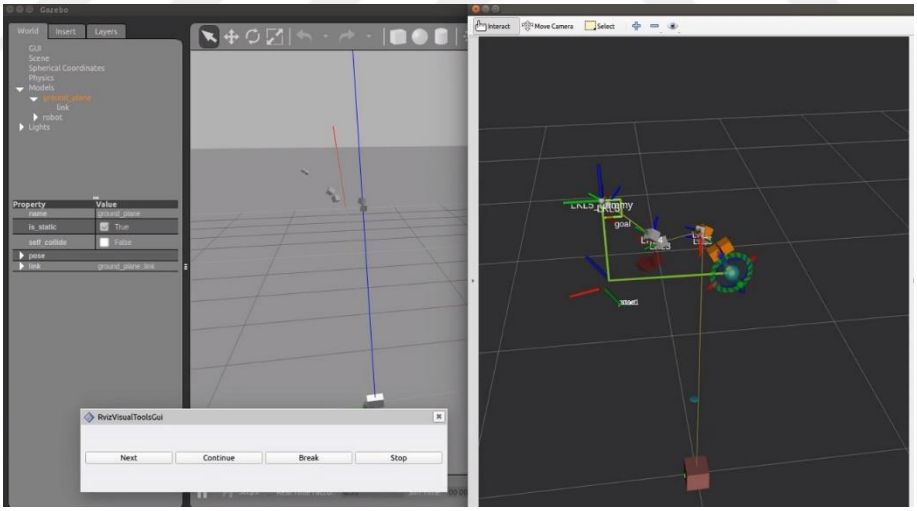

**Şekil 3.3.** MoveIT eklenti bağlantıları [11]

URDF dosyasının, temsili görselleştirilmiş hali, Şekil 3.4'te gösterilmiştir.



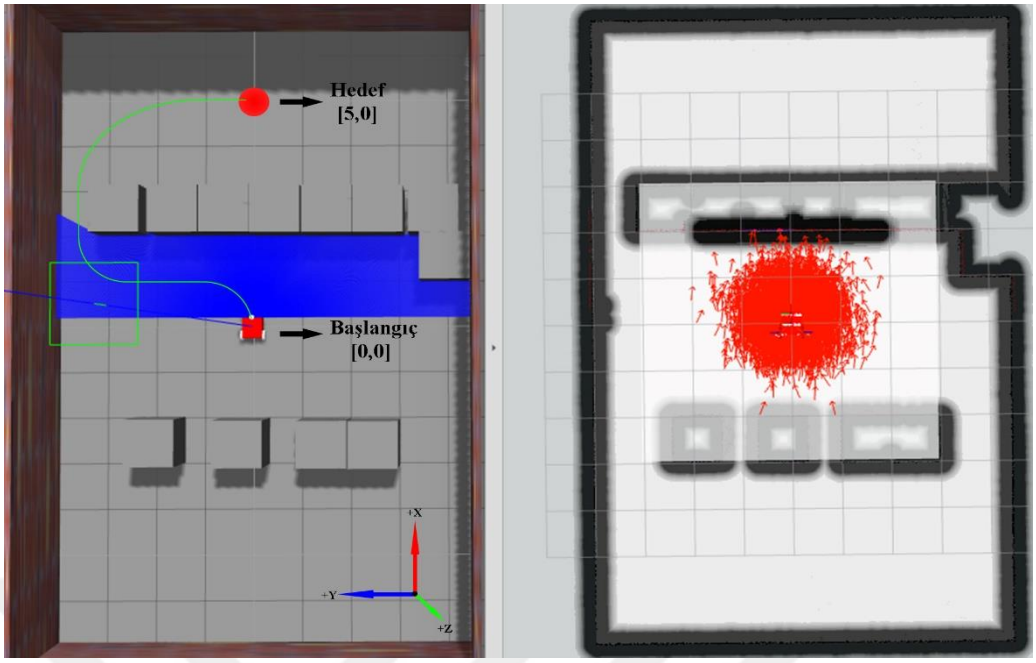

**Şekil 3.4.** Gerçek robotun, sanal hali [11]

MoveIT işlemi yapıldıktan sonra oluşturulan temsili bacağa, belirli bir hareket yaptırılarak yörünge çizdirilmiştir. Robotun yürümesi için gerekli olan ön hazırlık yapılmış ve insansı robotların yürümesinin nasıl yapılacağına dair bilgi sahibi olunmuştur. Şekil 3.5'te temsili bacağa, belirli bir uzunlukta yol güzergahı hazırlandıktan sonra uygun pozisyona getirilmiş ve karesel hareket yapmak suretiyle yörünge çizdirilmiştir. Hareket ilk önce devamlı ve daha sonra sonlu olarak gerçekleştirilmiştir.




**Şekil 3.5.** Rviz ve Gazebo ortamlarında, aynı anda yörünge kontrolü

Rviz ve Gazebo kullanılarak, ortamda herhangi bir yörünge çizmeden önce kullanılan gmapping, amcl gibi ortam haritasının çıkarılmasını sağlayan launch parametreleri ile oluşturulan engellerin ve dar geçitlerin bulunduğu bir ortamda, temsili bir robotu, klavye yardımı ile kontrol ederek ortamın haritası çıkarılmıştır. Harita çıkartıldıktan sonraki ortam, Şekil 3.6'de gösterilmektedir [12].



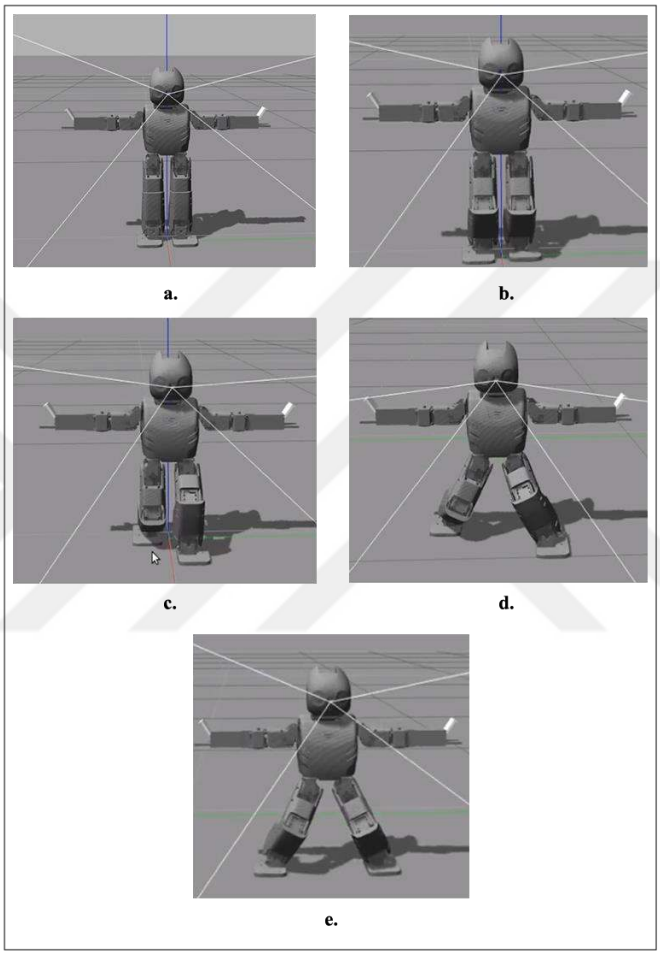

**Şekil 3.6.** Rviz ve Gazebo kullanılarak haritalanmış ortamda, hedefe hareket

Bütün bu çalışmalardan sonra insansı robot Robotis-OP2'nin ileri, geri, yan ve çapraz hareketlerini gerçekleştirebilmek için gerekli alt yapı oluşmuş bulunmaktadır. Robotta bulunan, 4'ü sol bacakta ve 4'ü sağ bacakta olmak üzere 8 adet aktüatörün kontrolünü sağlamak için belirli denklemler tasarlamak gerekmektedir. 8 eklemin mevcut pozisyondaki değerleri kontrol altına alınarak bir sinüs eğrisi şeklinde hareket ettirebilmek için yapılan hesaplamalar aşağıdaki denklemlerde görülmektedir.

$$y = \text{Eklem\_Pozisyon\_Değeri} + \text{Ölçek} * \sin(\text{in\_offset} + \text{in\_scale} * x)$$

Denklemde x, 0 ile 1 arasında giderek artarak, eklemlere verilen parametrelerden bir tanesini ifade etmektedir. x için i/n şeklinde hesaplanan, 50 adet değer verilerek 1 adım döngüsü oluşturulacaktır. Ölçek, radyan cinsinden olup adım aralığını ifade eder. in_scale, 3,14 olarak radyan cinsinden gereklidir. Eklemlere etki eden 2 değer vardır. Bunlar, Ölçek ve Eklem_Pozisyon_Değeri'dir. Bunların temel kodlaması yapıldıktan sonra hareket için, yukarı denklemdeki bilinmeyen değerlerin atanacağı fonksiyonlarla alınan değerlere hız bilgileri eklenir. Elde edilen 8 adet aktüatöre ait 8 adet eklemin her birine ayrı ayrı hız değerleri ile diğer hesaplanan bilgiler eklenmiş olur. Kullanıcının, x ve y eksenindeki hız isteği 0,5 metre/saniye olacak şekilde belirlenmiştir ve bu değer istenildiğinde değiştirilebilir. Yüksek değerler, gürültülere yol açar. Eklenen hız bilgileri ile kullanıcı tarafından verilen hız değerleri çarpılır ve eklem_açı_hız değeri bulunur. Bu değer (x*2-1) ile çarpılır. Bu çarpım sayesinde, eklem_açı_hız değeri, artıp daha sonra azalan değerler alır. Eklemler başlangıç değerinden, yine başlangıç değerine ulaşabilen bir formülle artmış ve azalmış olur. Tüm bu hesaplamalar, robotun sol veya sağ bacağındaki 4 adet aktüatör için

hesaplanır. Daha sonra – ile çarpılarak diğer 4 adet aktüatör değerleri hesaplanır. Bu sayede, 2 bacakta bulunan 8 adet aktüatör, aynı anda bir adım için elde edilen 50 farklı açı değerleri, ters-simetrik hareket yaparak yürümeyi gerçekleşmiş olur. Oluşturulan kodla görselleştirilen, robot temel hareketleri, Şekil 3.7'de görülmektedir.




**Şekil 3.7.** Robot temel hareketleri

Şekil 3.7a'da, robotun ilk pozisyonu, Şekil 3.7b'de robotun hareket edebilecek pozisyona geçmiş hali, Şekil 3.7c'de ileri hareket yaparken, Şekil 3.7d'de çapraz-ileri hareket yaparken ve Şekil 3.7e'de yan hareket yaparken resmedilmiş görselleri görülmektedir.

# 4. DERİN Q AĞI

Derin Q Ağı (DQA), en popüler ve yaygın olarak kullanılan Derin Pekiştirmeli Öğrenme (DPÖ) algoritmalarından biridir. Bu algoritma, Google'ın DeepMind ekibindeki araştırmacılar tarafından önerilmiş ve herhangi bir oyunu oynarken, sadece oyun ekranını giriş olarak alıp, insan düzeyinde sonuçlar elde edilmiştir.

Durum eylemi değeri işlevi olarak da adlandırılan bir Q işlevi, s durumunda, s için a eyleminin ne kadar iyi olduğunu belirtir. Bu nedenle, her durumda tüm olası eylemlerin değerini Q tablosu adı verilen bir tabloda depolar. En iyi eylem olarak, durumdaki en büyük değere sahip olan seçilir.

Bu çalışmada, sürekli olarak durum içeren ve sınırlı eylemi olan ortamlar örnek olarak verilmiştir. Fakat sürekli durumlardan sınırlı bir sayıda eylem çiftini ele alarak küçük bir Q tablosu oluşturulmuştur. En iyi Q değerini bulmak için tüm olası durum-eylem çiftleri arasında kapsamlı bir araştırma yapılmıştır. Çok fazla sayıda durumun bulunduğu ve her durumda denenecek çok fazla eylemin olduğu bir ortam her durumdaki bütün eylemleri gözden geçirmek zaman alıcıdır. Daha iyi bir yaklaşım, Q işlevini $Q(s,a;\theta) \approx Q(s,a)$ gibi bir parametreyle yakınlaştırmak istenir. Her durumdaki olası tüm eylemlerde, Q değerini yaklaşık olarak belirlemek için ağırlıkları olan bir sinir ağı kullanılabilir. Q işlevine yaklaşmak için sinir ağları kullandığı durumda, bunu bir Q ağı olarak adlandırabiliriz. Bu durumda ilk önemli soru ortaya çıkar. Q öğrenme güncelleme kuralı için:

$$Q(s,a) = Q(s,a) + \propto \left(r + \gamma maxQ(\acute{s}\acute{a}) - Q(s,a)\right) \tag{4.1}$$

$[r + \gamma maxQ(\acute{s}a)]$ hedef değer ve Q(s, a) öngörülen değerdir. Q öğrenmede, doğru bir politika öğrenerek bu değeri en aza indirme hedeflenir.

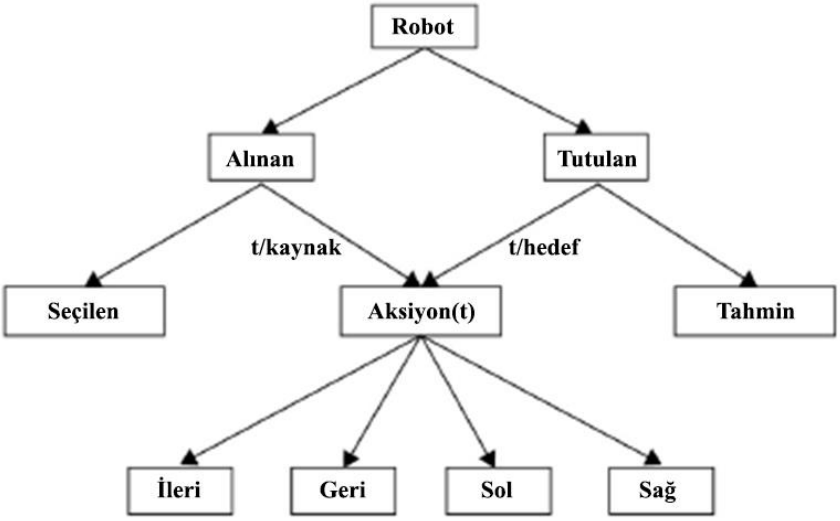
Benzer şekilde, DQA'da kayıp işlevini hedef ve öngörülen değer arasındaki kare fark olarak tanımlanır ve ayrıca ağırlıkları güncellenerek kaybı enazlanmaya çalışılır.

$$\theta: Kayıp = \left(y_i - Q(s,a;\theta)\right)^2 \tag{4.2}$$

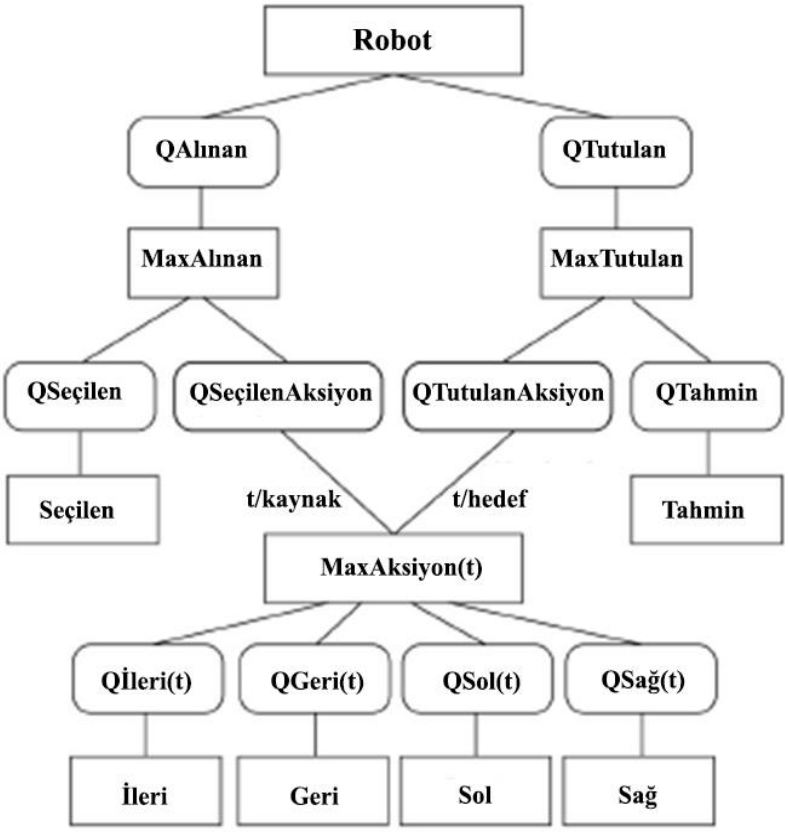
$$y_i = r + \gamma maxQ(\acute{s},\acute{a};\theta) \tag{4.3}$$

Ağırlıkları günceller ve gradyan iniş yöntemi ile kayıp en aza indirilir. Özet olarak, DQA'da, bir Q işlevine yaklaşmak için, işlev belirleyicileri olarak sinir ağlarını kullanır ve hatalar gradyan iniş yöntemi ile en aza indirilir [13].

Şekil 4.1. ve Şekil 4.2'de, Q öğrenme olmadan ve Q öğrenmeli durum eylem haritaları gösterilmektedir.




**Şekil 4.1.** Q öğrenme olmadan durum-eylem

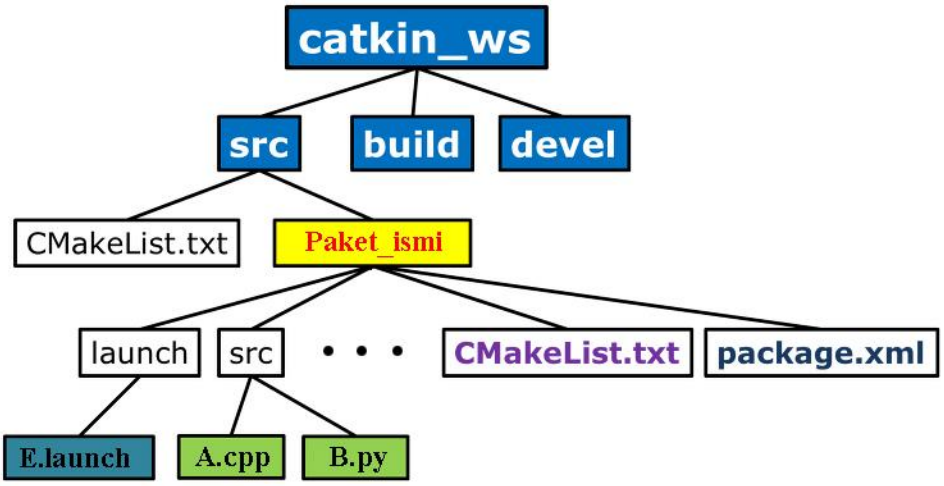



**Şekil 4.2.** Q öğrenmeli durum-eylem

## 4.1. Veri İşleme

Veri işleme, anlamlı bilgiler üretmek için veri öğelerinin toplanması ve manipüle edilmesidir. Bu konu bilgi işlemenin, "bir gözlemci tarafından tespit edilebilen herhangi bir şekilde bilginin değişimi" bir alt kümesi olarak düşünülmektedir [14].

Bu tezde, veriyi işleyebilmek için Numpy, Pandas, Matplotlib, Keras ve Tensorflow kütüphaneleri ve Linux açık kaynak kodlu işletim sistemi olan Ubuntu 16.04 kullanılmıştır. Bu kütüphaneleri yüklemek için kısaca Anaconda kurulabilir. Anaconda programı, gerekli tüm kütüphaneleri içerisinde barındırmaktadır. Sürüm farklılıklarından dolayı veya eksikliklerden kaynaklı olası hatalarda, gerekli kütüphaneler ayrı ayrı paketler olarak kurulabilir. Linux'ta, açık kaynak kodlu olmasından dolayı sürekli bir veri giriş çıkışı olmaktadır. Bu olay sorunlara yol açmaktadır. Çalışan bir program bir hafta sonra çalışamaz hale gelebilmektedir. Bu sorunu çözmek için; her projenin gereksinimlerine özel çalışma ortamı kurulabilir. Bu sayede güncellemelerden ve diğer projeler için kullanılacak olan farklı sürümlerdeki paketlerden etkilenmemiş olur. Örnek olarak bu projede hem Python 2.7 kullanılmakta hem de Python 3.9 kullanılmaktadır. Eğitim aşamasında sonuçların görselleşmesi için Python 2.7 kullanılırken, diğer tüm eğitim aşamaları için Python 3.9 kullanılmaktadır. Örnek bir çalışma alanı yapısı, Şekil 4.3'te gösterilmektedir.

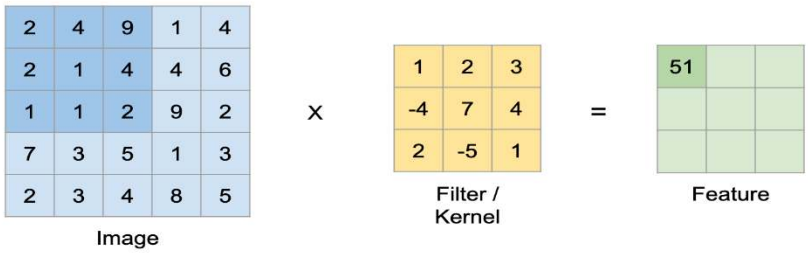



**Şekil 4.3.** Linux'da oluşturulan tipik çalışma ortamı [4]

Çalışma yapılacak projelerimiz src klasörünün içerisine kurulur. Projenin bulunduğu pakette, launch, src, Script, Word, Rviz, Doc gibi klasörler bulunmaktadır. Bu klasörler ile tüm klasörler arasındaki bağlantıları, CMakeList.txt ve package.xml belgelerini düzenleyerek

sağlayabiliriz. Hem Python hem de C++ gibi programlama dilleri ile yapılan projeleri çalıştırabiliriz [15].

Eğitim için gerekli verileri işlemek için kullanılan evrişim işlemi ile giriş değerleri daha belirgin hale getirilebilir ve daha az boyut kaplayarak hem zamandan hem de işlem yükünden kurtulmak için işlem yapılmış olunur. Şekil 4.4'te, temsili bir evrişim şekli görülmektedir.




**Şekil 4.4.** Evrişim işlemi örneği

Evrişim işlemi parametreleri farklı başarılar elde etmemize sebep olabilir. Elde edilen giriş değerlerimiz oluşturulan YSA ile işlenerek çeşitli öznitelikleri ortaya çıkarır. YSA, katmanlardan ve nöronlardan oluşur. Bir nörona girdiler ile birlikte bunlarla öğrenmeyi kontrol edebileceğimiz ağırlıklar atanarak hesaplama yapılır. Ayrıca, bir bias değeri verilir. Tüm bu parametreler ile uygun bir aktivasyon fonksiyonu uygulanarak çıkış elde edilir.

### 4.1.1. Pandas, Matplotlib ve Tensorflow kütüphaneleri

Pandas, veri işlemesi ve analizi için Python programlama dilinde yazılmış olan bir yazılım kütüphanesidir. Bu kütüphane temel olarak zaman etiketli serileri ve sayısal tabloları işlemek için bir veri yapısı oluşturur ve bu şekilde çeşitli işlemler, bu veri yapısı üzerinde gerçekleştirilebilir olur [16].

Kütüphane özellikleri

- İndeksli DataFrame (veri iskeleti) objeleri ile veri işlemesi yapabilmek.
- Hafızadaki veya farklı türlerde bulunan veriyi okuyabilmek ve yazabilmek için araçlar sağlamak.
- Veri sıralama ve bütünleşik kayıp veri senaryolarına karşı esnek imkânlar sunması.
- Veri setlerinin, tekrardan boyutlandırılması veya döndürülmesi.
- Etiket bazlı dilimleme, özel indeksleme ve büyük veri setlerini ayrıştırma özelliği.
- Veri iskeletine sütun ekleme veya var olan sütunu çıkarma.

- Veri gruplama özelliği ile ayırma-uygulama-birleştirme uygulamalarının yapılabilmesi.
- Veri setlerinin birleştirilmesi ve birbirine eklenmesi.
- Hiyerarşik eksenleri indeksleme özelliğiyle birlikte çok boyutlu veriden, daha az boyutlu veri elde edilebilmesi.
- Zaman serisi özelliği: Zaman aralığı oluşturma ve sıklık çevrimleri yapma, hareketli aralık istatistik fonksiyonları, tarih öteleme ve geciktirme.
- Veri filtrelemesi yapabilmek.

Kütüphane performans konusunda yüksek derecede eniyilenmiştir [17].

NumPy, Python programlama dili için büyük, çok boyutlu dizileri ve matrisleri destekleyen, bu diziler üzerinde çalışacak üst düzey matematiksel işlevler ekleyen bir kitaplıktır. NumPy'ın atası Numeric, ilk olarak Jim Hugunin tarafından diğer birkaç geliştiricinin katkılarıyla oluşturuldu. 2005 yılında Travis Oliphant, Numarray'ın özelliklerini kapsamlı değişikliklerle Numeric'e dâhil ederek NumPy'yı yarattı. NumPy açık kaynaklı bir yazılımdır ve birçok katkıda bulunanlara sahiptir [18].

Matplotlib, Python programlama dili ve sayısal matematik uzantısı NumPy için bir çizim kitaplığıdır.

Matplotlib, John D. Hunter tarafından yazılmıştır. O zamandan beri aktif bir geliştirme topluluğuna sahiptir ve BSD tarzı bir lisans altında dağıtılmaktadır. Michael Droettboom, John Hunter Ağustos 2012'de ölmeden kısa bir süre önce matplotlib'in baş geliştiricisi olarak aday gösterildi ve Thomas Caswell da kabul edildi.

TensorFlow, makine öğrenimi için ücretsiz ve açık kaynaklı bir yazılım kütüphanesidir. Bir dizi görevde kullanılabilir, ancak derin sinir ağlarının eğitimi ve çıkarımına özel olarak odaklanmaktadır. Derin öğrenme modelleri oluşturmak için yaygın olarak kullanılır ve makine öğrenmesinin bir alt kümesidir. Birçok farklı platformda paylaşılabilen ve yürütülebilen veri akış grafikleri kullanır. Tensör demek, çok boyutlu bir dizi demektir. Bu yüzden TensorFlow, tam anlamıyla hesaplama grafiğinde çok boyutlu dizilerin (tensörler) akışıdır [19].

Tensorflow, veri akışına ve türevlenebilir programlamaya dayalı sembolik bir matematik kitaplığıdır. Google'da hem araştırma hem de üretim için kullanılmaktadır. Google Brain ekibi tarafından Google'ın iç işlerinde kullanımı için geliştirilmiştir. 2015 yılında Apache License 2.0 sürümü altında piyasaya sürülmüştür [20].

## 4.2. Derin Q Ağı Mimarisi

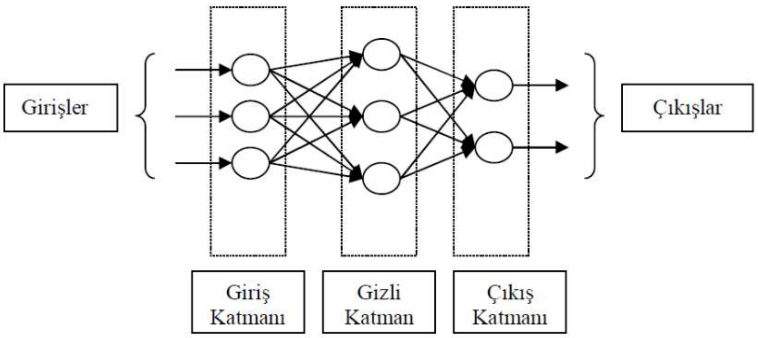
DQA mimarisi, DeepMind ekibi tarafından tanımlanmıştır. DQA, herhangi bir duruma göre herhangi bir eylemin ne kadar iyi olduğunu saptamaya yarayan veya benzeri iyileme çalışmaları yapabilen, bazı metotlar uygulanabilmektedir.

### 4.2.1. Sinir Ağları

Sinir sistemi, çevreden alınan bilgiye karşılık bir tepki oluşturulmasını sağlayan sistemdir. Örneğin masada duran kalemi elimizle tutup kaldırmak istediğimizde ona dokunuruz ve yeteri kadar sıkarız. Eğer çok fazla sıkarsak kalem zarar görebilir ya da yeteri kadar sıkmazsak da bu sefer kalemi kaldıramayız, elimizden kayar. İşte tam burada sinir ağlarımız görev alır. Önce kalemi ne kadar sıktığımız bilgisi sinir ağlarımıza gider ve belli bir değere (kalemin elimize uyguladığı basınç) gelene kadarda kaleme uyguladığımız kuvveti artırırız. Başka bir örnek verecek olursak odamızda otururken ellerimizde havayı hissetmeyiz ama hareket halindeki bir arabanın camından elimizi çıkardığımızda havayı(rüzgârı) hissedebiliriz. Bu örnekte havanın elimize uyguladığı basıncı hissederiz çünkü basınç istenilen değere ya da fazlasına ulaşmıştır. Bu örnekteki, hissetmemiz için gereken minimum basınç değeri kişiden kişiye değişiklik göstermektedir [21].

### 4.2.2. Yapay Sinir Ağları

Yapay sinir ağları (YSA) bir makine öğrenmesi yöntemidir. YSA'da amaç insan sinir sistemini taklit ederek karar verebilen ve yorum yapabilen makineler oluşturmaktır [22]. Şekil 4.5'te, bir yapay sinir ağı gösterilmektedir:

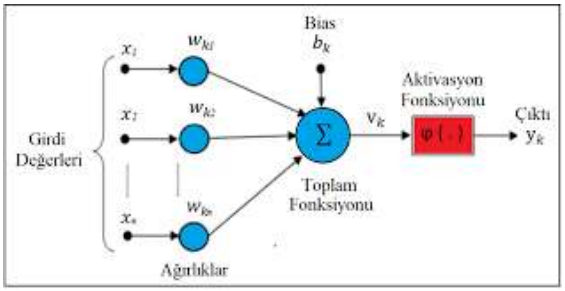



**Şekil 4.5.** YSA katmanları [22]

Gördüğünüz gibi yapay sinir ağı sistemi üç katmandan oluşmaktadır. Giriş katmanında herhangi bir hesaplama gerçekleşmez. Bu katmanda sinir ağlarına verilerin giriş sağlanır. Göreceğiniz üzere hücreler birbirlerine bağlıdır, bu bağlantılar bizim sinir sistemimizdeki sinapsları temsil etmektedir. Bu bağlantılara konunun devamında ağırlık denilmektedir. Görüntüye uygulanacak evrişim işlemi, özellikleri ayıklamak amacı ile evrişimli katmanda gerçekleştirilir, daha sonra da özellik haritaları, boyutların azaltıldığı havuzlama katmanına geçirilir. Kullanım durumuna bağlı olarak herhangi bir sayıda evrişim ve havuzlama katmanı eklenebilir. Bundan

sonra, görüntüyü sınıflandıran, tamamen bağlı bir katman olarak bilinen ve sonunda gizli bir katman bulunan bir sinir ağı eklenebilir.[23]

Şekil 4.6'da bir yapay sinir hücresi gösterilmektedir:




**Şekil 4.6.** Yapay sinir hücresi [22]

Gördüğünüz gibi, hücremize m adet veri (girdiler X ile gösterilir) giriyor. Sonra giren veriler ağırlıklarla çarpılıyor. Daha sonra tüm veriler toplanıyor ve bias değeri ekleniyor, bu sayede net girdi elde ediyoruz. Net girdi, aktivasyon fonksiyonumuzdan geçiyor ve bir çıktı (çıktılar Y ile gösterilir) elde etmiş oluyoruz. Başka bir deyişle, y çıkışını kestirim etmek için üç tane x1, x2 ve x3 girişimiz var. Bu girdiler w1, w2 ve w3 ağırlıkları ile çarpılır ve birlikte toplanır, yani x1.w1 + x2.w2 + x3.w3 şeklinde hesaplama yapılır. Girdileri ağırlıklarla çarpılır. Çünkü tüm girdiler y çıktısının hesaplanmasında önem derecesi olarak eşit değildir. Çıktının hesaplanmasında x2'nin diğer iki girişe göre daha önemli olduğunu varsayalım. Sonra, diğer iki ağırlıktan ziyade w2'ye yüksek bir değer atanır. Ağırlıklar, girişlerle çarpılmasından dolayı, x2 diğer iki girişten daha yüksek bir değere sahip olacaktır. Girdiler ağırlıklar ile çarptıktan sonra özetlenir ve "önyargı b" denilen bir değer eklenir. Yani, z = (x1.w1 + x2.w2 + x3.w3) + b, yani: z doğrusal regresyon denklemine benzer. Bu sadece düz bir çizginin denklemidir. z = mx + b. M ağırlıklar (katsayılar) olduğunda, x girdidir ve b ise biastır [24].

### 4.2.3. Evrişimsel Sinir Ağları

ConvNet olarak da bilinen ESA, özel bir sinir ağı türüdür ve görüntü işlemede yaygın olarak kullanılmaktadır. Bir ESA'nın uygulaması, kendi kendine hareket eden arabalardan, fotoğraflarınızdaki arkadaşlarınızın otomatik olarak etiketlenmesine kadar tüm alanlarda kullanılabilir. ESA'lar, görüntüyü tanımak için uzamsal bilgileri kullanır.[25]

DQA'nın ilk katmanı evrişimli ağdır ve ağın girişi için robotun gördüğü ortamdan bir görüntü karesi veya çevresel algı sensörlerinden elde edilen değerler alınır. Bu saf görüntüler ortamdaki durumu anlamak için evrişim katmanına verilir. Ancak saf karelerde, 256 renk değerine sahip, RGB halde bulunan, 300 x 300 piksel olacaktır ve ham pikseller, doğrudan beslenirse net bir şekilde çok fazla hesaplama ve hafıza alacaktır. Bu nedenle, boyut ölçüleri indirilir ve RGB

değerleri griye dönüştürülür ve önceden işlenmiş bu görüntü, evrişimli katmanlara girdi olarak beslenir. Evrişimli katman, görüntüdeki farklı nesneler arasındaki uzamsal ilişkiyi tanımlayarak anlamaya çalışır. Örnek bir ağ yapısı, iki evrişim katmanı ve onu izleyen ReLU aktivasyon işlevli tam bağlı katman olabilir. Havuzlama katmanı eklenebilir.

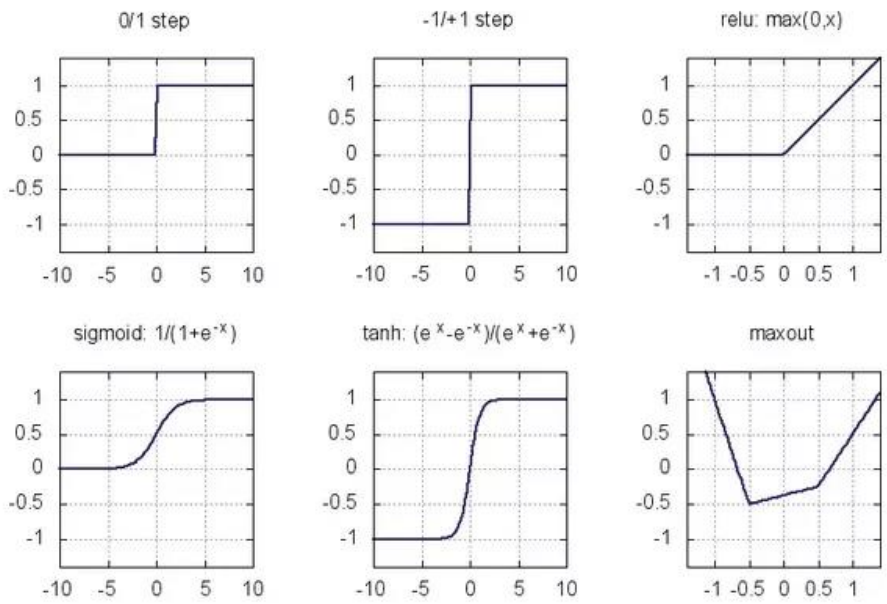
Havuzlama katmanı, nesne algılamada, sınıflandırma gibi görevler uygulandığı zaman faydalıdır. Bu durumlarda görüntü içinde nesnenin pozisyonu ile ilgili bilgi gerekli değildir. Örneğin, görüntüde bir köpek olup olmadığını sınıflandırmak istendiği zaman, yalnızca bir köpeğin görüntüde olup olmadığı kontrol edilir ve köpeğin nerede olduğu kontrol edilmez. Bu durumda, köpeğin durumuna bakılmaksızın görüntüyü sınıflandırmak için bir havuz katmanı kullanılır.

Q değeri için, alınan görüntü ve bir eylem, DQA'ya giriş olarak verilirse, ağ çıkışı Q değeri olacaktır. Ancak bir durumda pek çok eylem olacağı için tek bir eylem seçimi gerekecektir. Ayrıca, her bir eylem için bir ileri geçiş ile birçok durum olacak ve bu da hesaplama açısından maliyetli olacaktır. Bu nedenle, ağ girişine alınan görüntü, giriş olarak tek başına verilir ve durumdaki tüm olası eylemler için Q değerleri alınır. Çıkış katmanındaki birimlerin sayısı, görüntüdeki durum olarak ayarlanır. DQA, bir görüntü için beslenilerek, görüntü durumlarındaki tüm eylemler için Q değeri bulunur.

Durumun, Q değerlerini elde etmek için sadece mevcut görüntüler kullanılmaz. Ayrıca son dört görüntüde göz önünde bulundurulur [26].

### 4.2.4. Aktivasyon fonksiyonları

Seçilen aktivasyon fonksiyonu, probleme uyumlu olmalıdır. Eğer 2 farklı çıkış isteniyorsa, aktivasyon fonksiyonu olarak step fonksiyonu kullanılmalıdır. Bu fonksiyon, evet veya hayır gibi, 0 veya 1 gibi cevaplar vererek problemi çözmeye çalışır. Örnek olarak, cinsiyet belirlemede 0 değeri erkeği, 1 değeri kızı ifade ederek işlem yapılabilir. Eğitim sonucunda ise işlenmiş verilere göre 0 veya 1 değerinden herhangi bir çıkışı, modelin başarım oranına göre doğru seçebilmektedir. Bazı aktivasyon fonksiyonlarının yapısı ve şekli, Şekil 4.7'de gösterilmektedir.




**Şekil 4.7.** Aktivasyon fonksiyonlarından bazıları [27]

Bu tezde, çıkış 3 veya 5 adet olarak elde edilmek istendiği için ReLU aktivasyon fonksiyonu kullanılmıştır. Bu fonksiyon ile çeşitli problemler ile ortaya çıkan negatif değerler sıfırlanarak görmezden gelinmiş ve pozitif değerlere y=x doğru fonksiyonu uygulanarak 3 veya 5 değer elde edilmiştir. DQA Kullanılarak Eğitim İşlemi başlığı altında detaylı anlatılmıştır.

### 4.2.5. Tecrübe Tekrarı

PÖ ortamlarında, bir a eylemi gerçekleştirilerek ve bir ödül alınarak bir durumdan diğer durumlara geçilir. Bu geçiş bilgisi <s, a, r, s> olarak "tecrübe tekrarı" veya "tekrar belleği" olarak adlandırılan bellekte saklanır. Bu geçişlere, öğrenme etkeninin deneyimi denir.

Deneyim tekrarının ana fikrinde, DQA'nı, son geçişler ile eğitmek yerine, tekrar belleğinden örneklenen geçişlerle eğitilmesi amaçlanmıştır. Öğrenme etkeninin deneyimleri teker teker ilişkilendirilir, bu nedenle tekrar belleğinden rastgele bir eğitim örneği grubunu seçmek, öğrenme etkeninin deneyimi arasındaki korelasyonu azaltır ve öğrenme etkeninin çok çeşitli deneyimlerden daha iyi öğrenmesine yardımcı olur.

YSA'lar ilişkili tecrübeye karşı aşırı öğrenme gösterir. Bu nedenle tekrar belleğinden rastgele bir deneyim grubu seçerek, aşırı öğrenme azaltılır. Tecrübeyi örneklemek için tek tip örnekleme kullanılabilir. Deneyim tekrarını, bir liste yerine bir kuyruk olarak düşünebiliriz. Tekrar belleği, son tecrübelerin sadece sabitlenmiş sayısını depolar. Bu nedenle, içine yeni bir bilgi geldiği zaman, eskisi silinir [28].

### 4.2.6. Hedef Ağ

Kayıp işlevinde, hedef ile öngörülen değer arasındaki kare farkı hesaplanır:

$$Kayıp = \left(r + \gamma max_{\acute{a}} Q(\acute{s}, \acute{a}; \theta) - Q(s, a; \theta)\right)^2 \quad (4.4)$$

Hedef değeri ve öngörülen değeri hesaplamak için aynı Q işlevi kullanılır. Yukarıdaki denklemde hem hedef Q, hem de öngörülen Q için aynı ağırlıkların kullanıldığı görülebilir. Aynı ağ ile öngörülen değer ve hedef değer hesapladığından, bu ikisi arasında çok fazla sapma olabilir. Bu sorunu önlemek için, hedef ağ adı verilen ayrı bir ağ kullanılır. Böylece kayıp işlevi:

$$Kayıp = \left(r + \gamma max_{\acute{a}} Q(\acute{s}, \acute{a}; \theta') - Q(s, a; \theta)\right)^2 \quad (4.5)$$

Hedef Q parametresindeki, $\theta$ yerine $\theta'$ olduğu görülmektedir. Q değerlerini elde etmek için kullanılan gerçek Q ağı, gradyan iniş yöntemi kullanılarak $\theta$ doğru ağırlıklarını öğrenir. Hedef ağ birkaç zaman adımı için dondurulur ve ardından ağırlıklar, gerçek Q ağından kopyalanarak, hedef ağ ağırlıklarına güncellenir. Hedef ağı bir süre dondurmak ve daha sonra ağırlıkları gerçek Q ağ ağırlıkları ile güncellemek eğitimi kararlı hale getirir.

### 4.2.7. Kırpma Ödülleri

Robotun herhangi bir durumda göstereceği eylemi kontrol altına alabilmek için kırpma ödülleri atanır. Ödül atama her duruma göre değişir. Bazı durumlarda, hedefe ulaştığında +200, engele veya duvara çarptığında -200 gibi ödüller atanır. Diğer durumlarda ise robotun, hedefe yönelik yaptığı her açı değerine göre bulunulan durum-eylem çiftine özel ödüller verilir.

$$\text{Ödül} = \left[1 - 4 * \left|0.5 - (0.25 + 0.5 * açı \,\%\, (\tfrac{2*\pi}{\pi})\right| [aksiyon] * 5\right] * 2^{\frac{Mevcut\_uzaklık}{Hedefe\_uzaklık}} \quad (4.6)$$

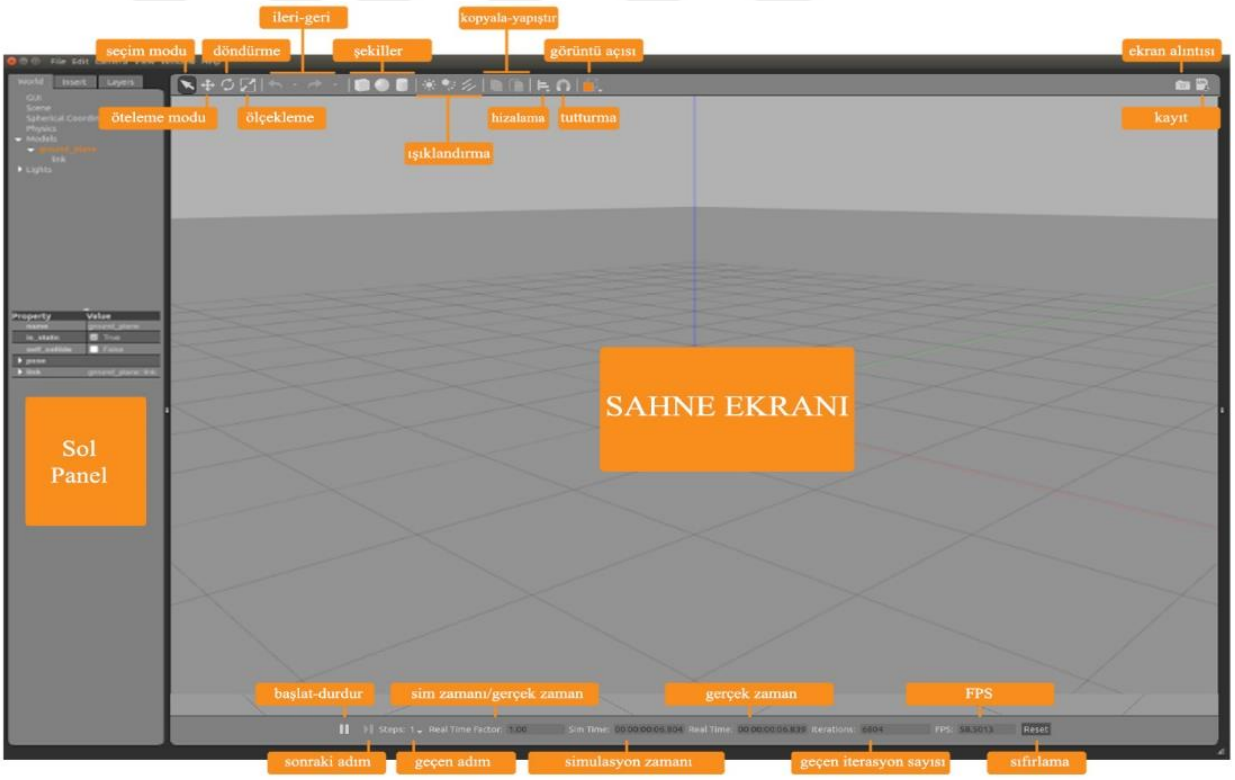
$$açı = \frac{-\pi}{4} + durum[-4] + \frac{\pi}{8} * i + \frac{\pi}{2} \quad (4.7)$$

# 5. DENEYSEL ÇALIŞMALAR

Bu tezde, Robotis-OP2'nin DQA ile engelden kaçarak hedefe gidebilmesi, en uygun metodu bularak ve metoda uygun bir ortam tasarlayarak yapılmaktadır. Bundan dolayı birçok ortamda farklı metotlar kullanılarak çalışmalar yapılmıştır.

## 5.1. Simülasyon programları

Tezde, benzetim ortamı olarak Gazebo ve Webots ortamları kullanılmıştır. ROS benzetim programının, görselleştirme arayüzü olarak kullanılan Gazebo, geliştirilen kodları ve tasarımsal şekilleri görselleştirerek, gerçek ortama benzetilerek tasarlanan ortamı, daha iyi analiz etmemizi sağlar. Şekil 5.1'de, Gazebo'ya ait ekran görüntüsü ve sağladığı bazı kolaylıklar gösterilmektedir.




**Şekil 5.1.** Gazebo ortamı üzerindeki bazı kolaylıkların gösterimi

Şekil 5.1'de, kütüphanesi kurulu ajanın (insansı robot), yakınlaşma, uzaklaşma, öteleme ve dönme hareketini yapabilmemizi sağlayan mouse ile kontrol yapabildiğimiz, gerçek ortamdaki nesneleri birebir oluşturmamıza, duvarlar eklememize, çeşitli engeller tasarlamamıza imkân sunan arayüz görünmektedir. Bu program ile oluşturulan şekiller, aşağıda gösterilmektedir [28].

Webots ortamı, yapı olarak Gazebo ortamının gereksinimlerinden çok farklıdır. Çoğu dosyaların arasındaki bağlantılar yapılmış vaziyette olup, fazladan bir işlem yapmadan, kodlara müdahale edilebilen bir arayüz ile kullanıcı için kolaylık sağlamaktadır [29].

## 5.2. Robotun Hesapsal Kontrolü

Tezde insansı robotun engelleri ve hedefi bulabilmesi için ilk uygulama olarak robotun hesapsal kontrolü yapılmıştır. Üç farklı ortam oluşturulmuş ve her ortamda konumları bilinen engeller ve hedef bulunmaktadır.

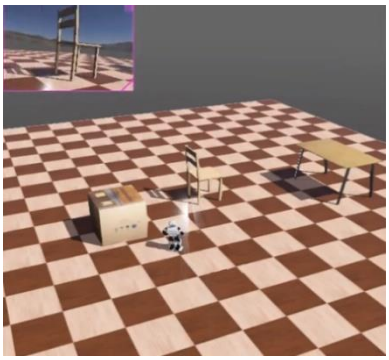
Deney 1'de robot, yerleri bilinen engellerden kaçarak, konumu bilinen hedefe doğru hareket etmektedir. Robotun engellerden nasıl kaçacağı ve hedefe ne şekilde hareket edeceği hesaplanarak hedefe gitmesi sağlanmıştır. Robot gideceği güzergahı seçerken, engele çarpmadan hedefe en kısa yoldan ne şekilde gideceğini hesaplayarak gitmektedir.

Deney 2'de robot nesnelerin yerlerini bilmemekte fakat görüntü verisinden faydalanarak engel ve hedefi ayırt etmeye çalışmaktadır. Nesnenin engel veya hedef olmasına göre hangi yöne gidileceği hesaplanmaktadır.

Deney 3'te ise önceki deneyden farklı olarak sadece ortamda değişiklik yapılmış, engeller sarı renkli toplar ile hedef ise kırmızı renkli top ile tanımlanmıştır.

**Deney 1: Gerçek Nesnelerle Oluşturulan Ortam**

10x10m boyutunda bir alan içerisine yerleştirilmiş 3 engel ile insansı robotun bulunduğu, rasgele konumlara engellerin yerleştirildiği ve güney-batı köşesindeki başlangıç noktasından, tüm engelleri geçmek sureti ile kuzey-doğu köşesindeki hedefe hareket ederek ulaşması için tasarlanmış Ortam-1, Şekil 5.2'de görülmektedir.



**Şekil 5.2.** Webots ortamında hazırlanan Ortam-1

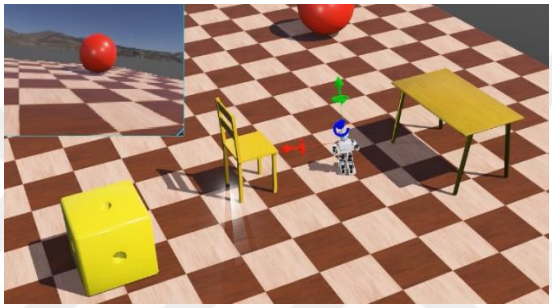
Bu ortam için tasarlanan algoritmada, engellerin ve hedefin yeri belirli olmak sureti ile kodlama yapılmış, robotun başlangıç noktasından, hedefe doğru olan eğimi hesaplanarak, hedefe

dönerek yürüyen bir şekilde hareket eden, engellerin kapsama alanına girince, bulunduğu konumdan hedefe paralel olarak hareket eden ve engellerin kapsama alanından çıktıktan sonra yeniden hedefe dönerek hareketine başlayan, hedefe ulaştığında, duran bir akış diyagramı oluşturulmuştur.

Ortamda bulunan 3 farklı engel ile hedef noktası, robotun sola dönmesi, sağa dönmesi ve ileri gitmesi, durum-eylem çiftleri oluşturmak için gereklidir. Ortamdaki engeller ve hedef noktası, durumları ve robotun yaptığı üç yön hareketi, gerçekleştirilecek eylemi ifade etmektedir.

**Deney 2: Renkleri Belirginleştirilmiş Nesnelerle Oluşturulan Ortam**

Ortam-1'de bazı sıkıntılar oluşmuştur. Bunlar, görüntü işlemeyi zorlaştıran problemlerdir. Bundan dolayı, aynı ortamda gerekli değişiklikler yapıldıktan sonra, bazı problemler çözülmüştür. Ortam-2, Şekil 5.3'te görülmektedir.



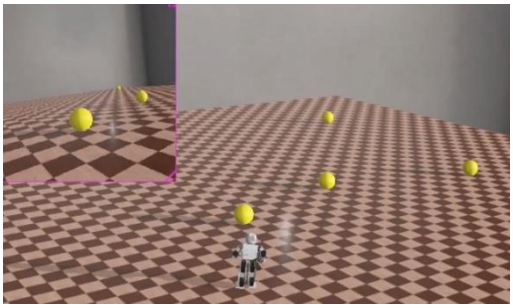
**Şekil 5.3.** Webots ortamında hazırlanan Ortam-2

Bu ortamda, engellerin ve hedefin, kamera yardımı ile saptanabilmesi için sarı ve kırmızı renkte olmasına karar verilmiştir. Bu sayede robot, engelleri ve hedefi, tespit edebilecek olmasından dolayı belirli sınırlar verilerek, engellere çarpmadan, hedefe gitmesi sağlanmıştır. Hedef noktasının gözlemlenebilmesi için kırmızı bir top konulmuştur.

Durumlar ve eylemler, Ortam-1'deki gibidir.

**Deney 3: Gerçek Nesnelerin Toplarla Temsil Edildiği Ortam**

Ortam-2'de oluşan sıkıntıları ortadan kaldırmak için yeni bir ortam oluşturulmuştur. Bu ortamda engellerin, belirli bir şeklinin olmasına ve özdeş olmasına karar verilmiştir. Şekil 5.4'te, Ortam-3 görülmektedir.



**Şekil 5.4.** Webots ortamında hazırlanan Ortam-3

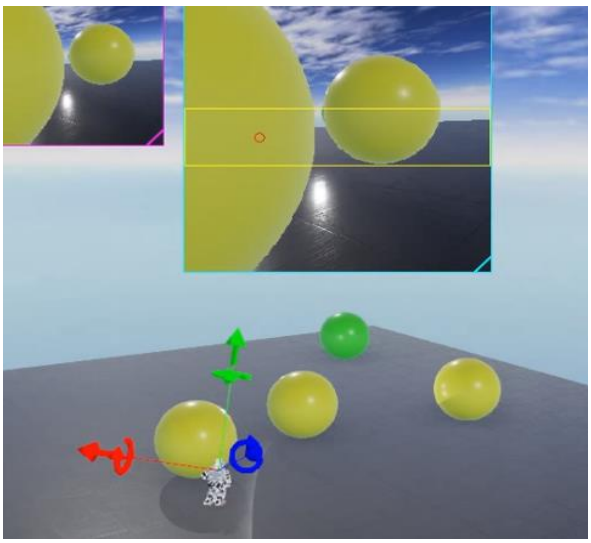
Özdeş olarak ayarlanan engeller sayesinde, görüntü işleme yapılarak saptanan engeller ile hedefe olan uzaklıklar hesaplanmıştır. Robottan alınan görüntüler ile engellerin alanlarının değişimi gözlemlenmiş ve bu değişimin karşılık geldiği uzaklık değeri bulunarak, engellere olan uzaklıklar ve hedefe olan uzaklık saptanmıştır. Engeller ve hedefin özdeş olması ile alınan görüntülerde, hesaplanan alanlar karşılaştırılarak, en yakın engel tespit edilmiş ve robot, belirli bir uzaklığa kadar engele yakınlaştığında, farklı bir eyleme geçecek şekilde programlanmıştır. Eylem sayısı 3'tür. Bunlar; sağa dönme, sola dönme ve ileri gitme eylemleridir.

Durumlar ve eylemler, Ortam-1'deki gibidir.

## 5.3. Engel-Hedef Tespiti ve Hareket Yörüngesi Hesaplama

Tasarlanan ortamda, görüntüden elde edilen verilerle robotun hareket etmesi için çalışma yapılmıştır. Elde edilen görüntüler ile yakınlık-uzaklık ölçülerek, yakındaki nesne engel ise yönünü değiştirerek çarpmamaya ve hedefe ilerlemeye devam etmektedir. Eğer nesne hedefi simgeliyor ise hedefi merkezine alacak şekilde hedefe yaklaşmaya çalışmaktadır.

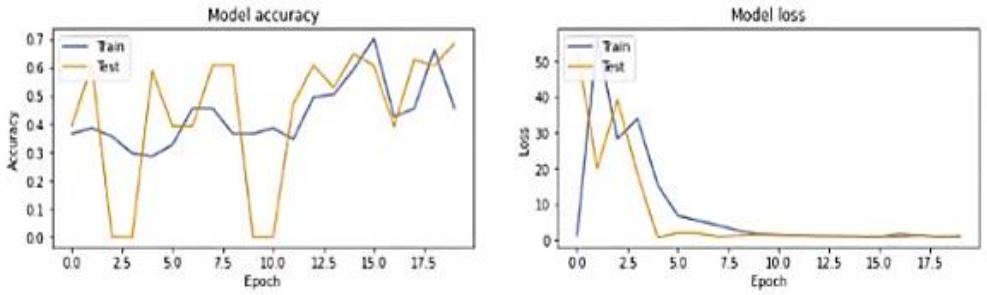
**Deney 5: Hedef ve Engel Tespitinin Yapıldığı Ortam**



**Şekil 5.5.** Webots ortamında hazırlanan Ortam-3.2

Şekil 5.5'te gösterilmiş olan ortam, Webots benzetim programı üzerinde tasarlanmıştır. Ortam-1'deki gibi dört durum ve üç eylem mevcuttur.

Ortama yerleştirilen üç engel ile hedef noktasının bulunduğu dört durum, robotun gerçekleştirebildiği sol, sağ ve ileri yön hareketleri olmak üzere üç eylemin olduğu sistemde yapılan eğitim sonuçları, Şekil 5.6'de gösterilmektedir.

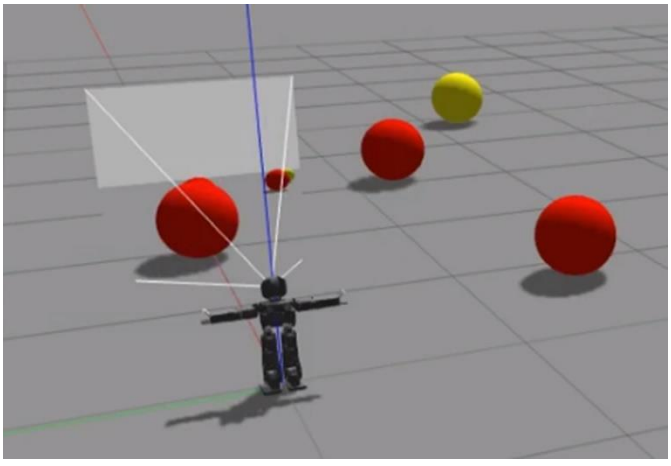



**Şekil 5.6.** Derin öğrenme ile elde edilen doğruluk ve kayıp sonuçları

Eğitim işlemi, İntel (R) Xeon (R) CPU 2.30 GHz CPU işlemci ile toplamda bir buçuk saatte gerçekleştirilmiştir. İlk 50 epochta kayıp değerlerinin yüksek olması, istenilen eğitim değerleri ile test edilen görüntü verilerinin öğrenilmeye çalışıldığı alanı göstermektedir. 112 epochtan sonra istenilen değer ile test verilerinin eşitlendiği görülmektedir. Böylece, oluşturulan eğitim modeli doğru sonuçlar vermeye başlamış ve öğrenme gerçekleşmiştir.

**Deney 6: Hedef ve Engel Tespitinin Farklı Yapıldığı Ortam**

Gazebo'da oluşturulan Ortam-4, Şekil 5.7'de gösterilmektedir.



**Şekil 5.7.** Gazebo ortamında hazırlanan Ortam-4

Bu ortamda, engellerin ve hedefin, kamera yardımı ile saptanabilmesi için sarı ve kırmızı renkte olmasına karar verilmiştir. Bu sayede robot, engelleri ve hedefi, tespit edebilecek olmasından dolayı belirli sınırlar verilerek, engellere çarpmadan, hedefe gitmesi sağlanmıştır. Hedef noktasının gözlemlenebilmesi için kırmızı bir top konulmuştur.

Ortam-4, Ortam-1'de bulunan dört durum ile birlikte hızlı sola dönme, normal sola dönme, ileri gitme, normal sağa dönme ve hızlı sağa dönme olmak üzere beş eyleme sahiptir. Yukarıdaki ortamlara göre daha fazla hareket yolu çıkartabilen bu sistem ile daha iyi bir eğitim yapılabilmektedir. Eylem sayısının fazla olması ile aşağıdaki ortamlarda uygulanacak eğitim için elde edilecek veri arttırılmış ve DQA için ön hazırlık yapılmıştır. Kameradan alınan görüntüler renklere göre filtrelenmiştir. Kırmızı renk için maske oluşturulmuştur. Aynı zamanda görüntüde birden çok olan ve özdeş olan engellerin yakınlık durumu hesaplanmıştır. Bu sayede robot, sadece kendi görüş açısında ve en yakında olan engele göre bir eylem içerisine girecektir. Alınan resmin orta noktası robotun bulunduğu yönü ifade ettiği için görüntülerden, engelin robota olan açısı hesaplanabilmektedir. 300x300 piksel büyüklüğünde alınan görüntülerin orta noktası (150,150), merkez noktadır. Eğer engel, merkez noktanın solunda ise robot sağa dönme eylemi gerçekleştirmektedir. Tersi durumda geçerlidir. Robotun gördüğü alanın açı değeri 105 derecedir. Bu değerle hesaplama yaparsak sağa veya sola dönerken kaç derece dönmesi gerektiğini hesaplayabiliriz.

$$\text{açı} = \frac{(\text{MerkezNokta} - \text{EngelMerkezNokta})}{105}$$

Açı, negatif değer alırsa engel solda demektir. Robot sağa doğru, açı değeri kadar dönme yaparak ilerler. Açı değeri, pozitif ise sola dönme eylemi açı değeri kadar gerçekleştirilir. Bu işlem, engellerden herhangi birisine yeterince yakın olduğunda devreye girer. Bu ortamda hedef ve engellerin konumları belirlidir. Öğrenme işlemi için veri elde edebilmek amacı ile bazı matematiksel denklemlere yar verilmiştir. Robot normal durumda, kendi koordinatları ile hedef koordinatları arasındaki bağıntılardan, hedefin bulunduğu yöne dönme eğilimindedir.

$$\text{A} = \frac{(\text{RobotKonumX} + \text{HedefKonumX})}{(\text{RobotKonumZ} + \text{HedefKonumZ})}$$

$$\text{B} = \arctan(A)$$

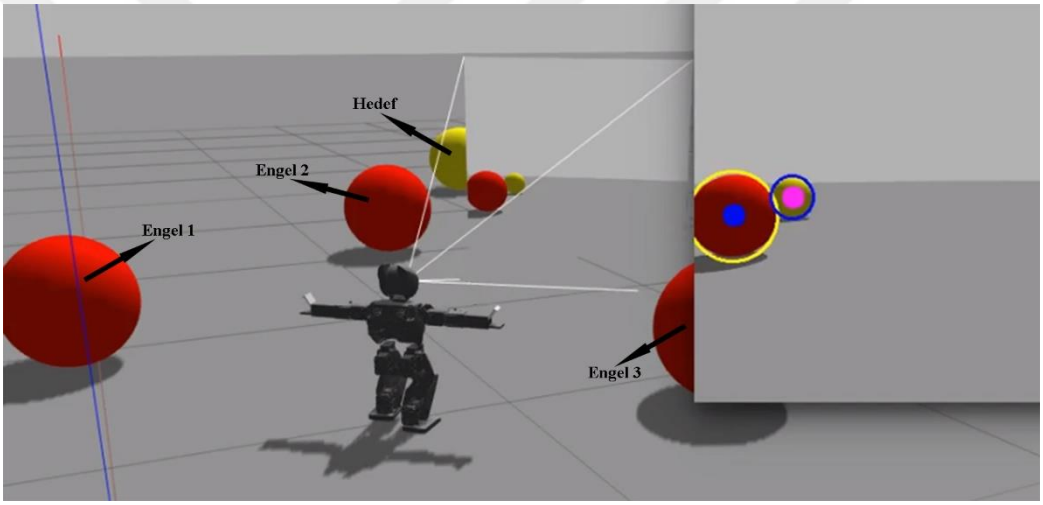
$$\text{Açı} = -3.14 + \text{B}$$

Burada hesaplanan açı değeri, robotun herhangi bir konumdayken hedefe gidebilmesi için dönmesi gereken açı miktarını göstermektedir. Eğer robot hedefe doğru gidiyorsa açı değeri 0 elde edilir. +- 10 tolerans ile hesaplama yapılmıştır. Çünkü insansı robotun dümdüz gidebilmesi zordur. Bu ortamda 3 eylem bulunmaktadır. Bunlar, ileri, sağa dönme ve sola dönme eylemleridir.

Görüntü işleme için OpenCV Kütüphanesi kullanılmıştır. İlk önce resim, işlem yükünü azaltmak için [0,1] arasındaki değerlerden oluşan grileştirmeye maruz bırakılır. Daha sonra,

bulunmak istenen renk değerleri verilerek, engeller ve hedef bulunabilir veya bulunmak istenen rengin, 3 katman için en düşük ve en yüksek değerleri belirlenerek, HSV formatında belirlenebilir. İhtiyaca göre tersleme (255'ten çıkarma) yapılabilir. Bu ortamda BGR'den, HSV'ye geçiş sağlanmış ve maskelenmiştir. Oluşturulan maskede oluşan gürültüler temizlenmiş ve konturlar çıkarılmıştır. Bulunan konturların alanları hesaplanmış ve engeller ve hedef özdeş olduğu için en büyük alana sahip olan konturun, en yakın engel olduğu belirlenmiş ve sadece o kontur üzerinde işlem yapılmıştır. Hedef konturu, sürekli gözlem altında tutulmuştur.

Robot, Gazebo ve Webots ortamında oluşturulan, 7 farklı ortamda, engellerden kaçınarak hedefe yürüyebilecek çevre elemanları ve hedef noktası belirtilmiştir. Görüntü işleme yardımı ile hareket eden robot ile örnek çevre, Şekil 5.8'de gösterilmektedir.



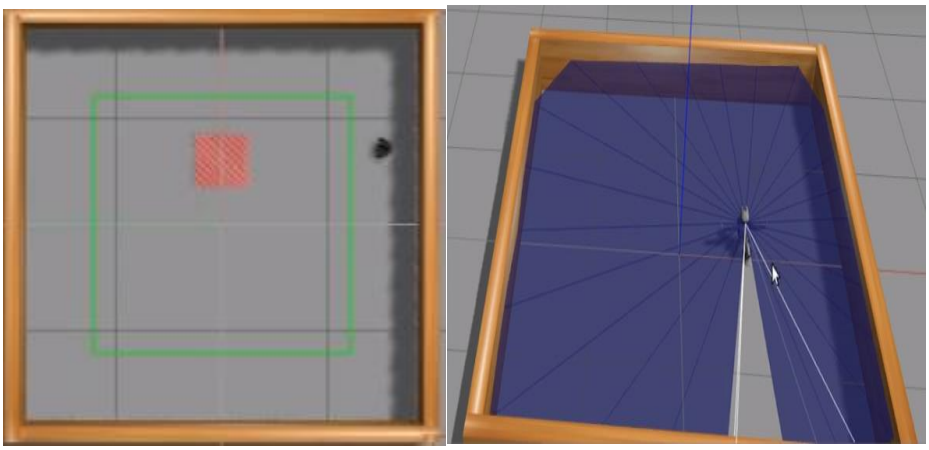

**Şekil 5.8.** Gazebo ortamında oluşturulmuş ortam

## 5.4. Derin Q Ağı ile Eğitim İşlemi

Yapılan çalışmalar ile bu tez süresi boyunca yapılan en ideal ortam ve eğitim işlemi için elde edilecek verileri bulmamızı sağlayan metotlar saptanarak DQA uygulanmaya başlanmıştır.

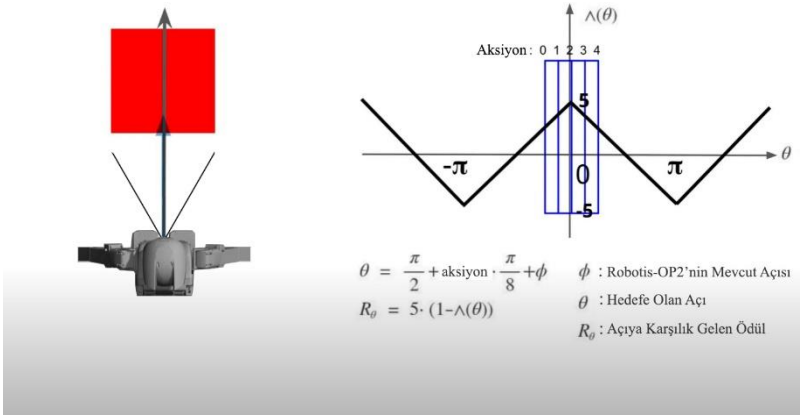
**Deney 7: Sürekli Durumda Duvarlarla Çevrili Alanda Oluşturulan Engelsiz Ortam**

Tasarlanan tüm ortamlardan elde edilen deneyimler ile daha basit ve anlaşılır bir Ortam-5, DQA için gerekli olan altyapı düşünülerek tasarlanmıştır. Bu ortam tasarlanırken, işlem yükünün azaltılması için 10x10m olan ortamın boyutları küçültülerek 4x4m boyutunda bir alan hazırlanmıştır. Alanın çevresine duvar örülmüştür. Bu sayede robotun üzerinde var olan sensörler yardımı ile robotun konumu sürekli kontrol altında tutulmuştur. İlk etapta alan içerisine bir engel konulmamıştır. Ama alanın çevresine örülen duvarlar engel olarak görülmektedir. Oluşturulan ortamın farklı açılardan alınmış iki görseli, Şekil 5.9'de görülmektedir.

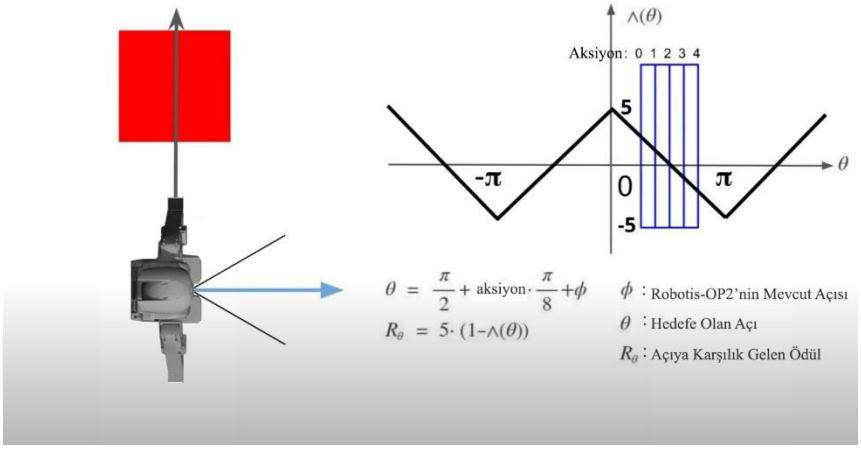


**Şekil 5.9.** Gazebo ortamında hazırlanan Ortam-5'in, farklı açılardan görüntüsü

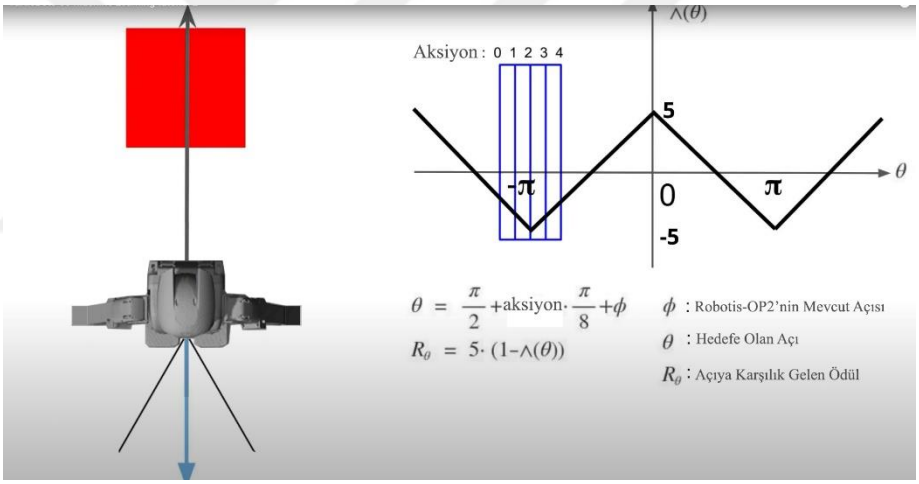
Şekil 5.8'de görülen ortamda robot, duvarlara çarpmadan, kırmızı kare ile belirlenen hedefe hareket ederek, DQA için gerekli altyapıyı kullanacak şekilde, kontrolörü tasarlanmıştır. Bu ortamda başlangıç noktası, alanın tam ortası olan (0,0) merkez noktasıdır. Robota 3 eylem vermek yerine, 5 eylem verilerek yeni bir kodlamaya geçilmiştir. Tasarlanan kod, Pekiştirmeli Öğrenme yöntemleri uygulanacak şekilde kodlanmıştır. Robot merkez noktasından hedefe hareket etmek için sürekli bir eylem denemektedir. Robotun bulunduğu her durum için 5 eylem değeri hesaplanıp kayıt altına alınmıştır. Robotun duvarlara çarpması durumunda -200 puan verilmiş, hedefe gitmesi durumunda +200 puan verilmiştir. Aynı zamanda robotun bulunduğu döngü içerisinde, her eylem için verilen bir puanlama sistemi tasarlanmıştır. Bu sistem, Şekil 5.10'da Şekil 5.11'de ve Şekil 5.12'te görülmektedir.




**Şekil 5.10.** Robotun, hedefe doğru hareket ederken, 5 eylem değeri için elde edilecek açı ödülleri



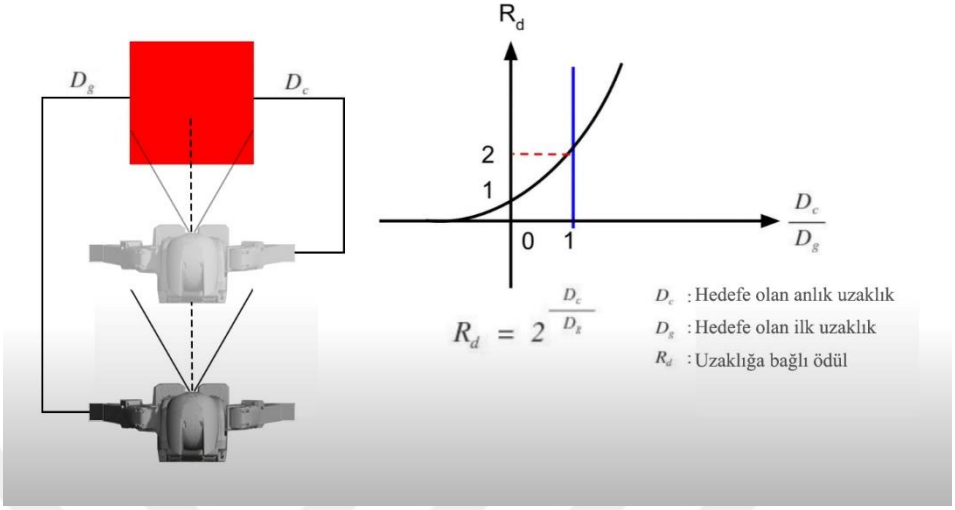

**Şekil 5.11.** Robotun, hedefe 90 derece açı ile hareketi sonucunda, 5 eylem değeri için elde edilecek açı ödülleri




**Şekil 5.12.** Robotun, hedeften 180 derece açı ile uzaklaşması sonucunda, 5 eylem değeri için elde edilecek açı ödülleri

Bu 5 eylem, sağa hızlı dönme, sağa dönme, ileri gitme, sola dönme ve sola hızlı dönme olarak adlandırılıp, 5 eyleme tekabül eden açı değerleri ise sıra ile 45, 67.5, 90, 112.5, 135 derece açılarıdır. Robot sürekli ileri gitmektedir. Bu yüzden, durup, sola veya sağa dönüş yapmamaktadır. Döngü, robot çarpıncaya kadar veya hedefe ulaşıncaya kadar sürmektedir. İnsansı robot, her döngü için merkez noktasından başlamaktadır. Döngüler içerisinde hesaplana tüm ödül değerleri, her durum ve o duruma ait 5 eylem değeri ödülleri olarak kaydedilmektedir. Son 4 durum incelenerek en iyi ödüle sahip yeni durum tahmin edilmektedir. Son durumdan 3 önceki durum geçmiş durumu, 2 önceki durum mevcut durumu ve diğeri ise gelecek durumu ifade etmektedir. Döngü içerisinde, verilen yığın boyutuna göre durum eylem çiftinin değerlerini toplar ve Q öğrenme yardımı ile en

yüksek ödüle sahip olan durum-eylem çiftlerini seçerek, ağırlıklarını günceller. Bütün bu işlemler olurken uzaklığa bağlı olarak eğitime katkı sağlanabilmesi için geliştirilen ödül sistemi, Şekil 5.13'te gösterilmektedir.




**Şekil 5.13.** Robotun, hedefe olan ilk uzaklık ile döngü içerisinde bulunduğu konuma göre olacağı ödül değeri

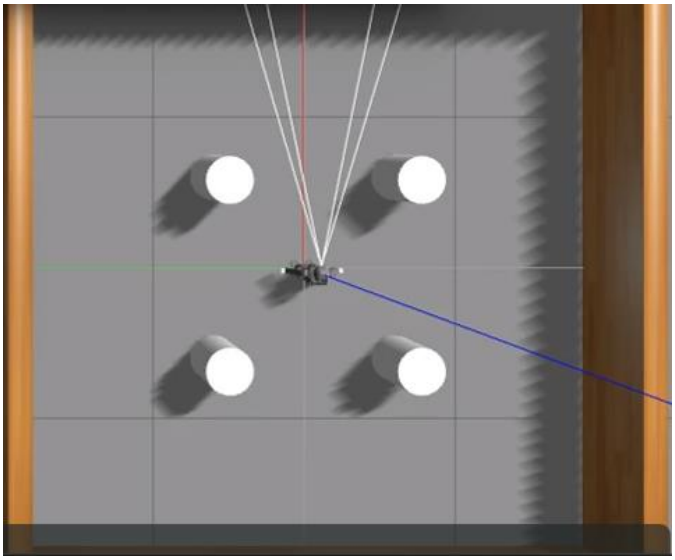
Robot, hedeften uzaklaştıkça küçülen, yakınlaştıkça büyüyen ödüller elde etmemiz için üslü sayı ile oluşturulan fonksiyon sayesinde, daha iyi bir öğrenme gerçekleştirilir.

Robot her çarptığında aynı hedefe tekrar tekrar merkez noktadan harekete başlayarak gitmeye çalışır. Hedefe ulaşınca ise 3 farklı eğitim modeli uygulanmıştır. 1. modelde robot hedefe gidince durur ve döngü sıfırlanmaz. Bunun yerine yeni bir hedef belirlenir ve robot eski hedefin bulunduğu konumdan aynı döngü içerisinde ağırlıklarını güncelleyerek hareket etmeye devam eder. 2. modelde ise robot hedefe ulaşınca robot merkez noktasında yeniden atanır ve aynı hedefe tekrar gidebilmesi için hareket etmeye devam eder. Ağırlıklar sürekli güncellenir. Robot engele çarparsa her şey sıfırlanır ve robot yeniden merkez noktasından aynı hedefe hareket etmeye çalışarak kendini eğitmeye devam eder. 3. modelde, robot hedefe ulaşınca, yeni bir hedef atanır ve robotta merkez noktasına taşınarak tekrar tekrar öğrenmeye devam eder. Bu tez kapsamında, bahsi geçen bu 3 modelde uygulanmıştır [10].

Bu ortamda, robot beş farklı açıda eylem gerçekleştirebilir. Etrafını çevreleyen duvarlar ile hedef noktası, durumları ifade etmektedir. Durum sayısını azaltarak daha az durumda, daha fazla eylemi gerçekleştiren ve daha hızlı sonuç elde eden bir sistem oluşturulmuştur.

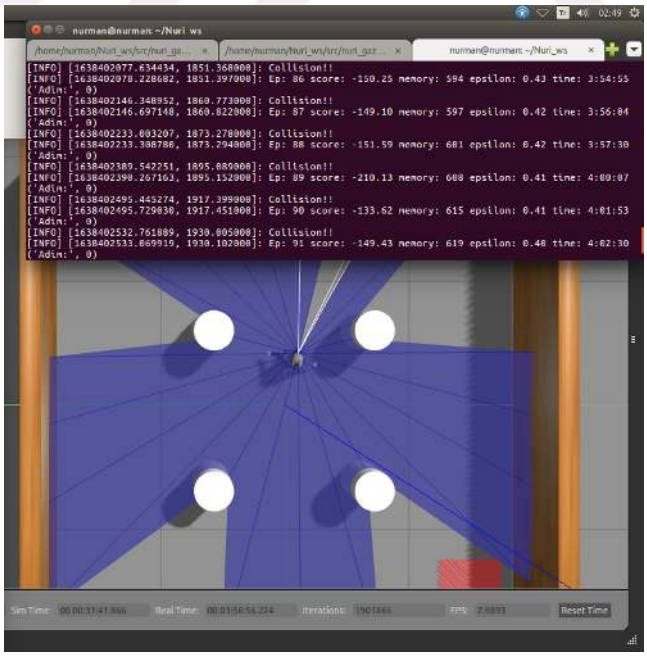
**Deney 8: Ortam-5'e Engel Eklenerek Oluşturulan Ortam**

Ortam-5' de uygulanan tüm sisteme duvarlar ile 4 engel eklenerek, eğitilmesi daha uzun sürecek olan ortam tasarlanmıştır. Şekil 5.14'te tasarlanan ortam görülmektedir.



**Şekil 5.14.** Gazebo ortamında hazırlanan ortam-6'in kuş bakışı görüntüsü

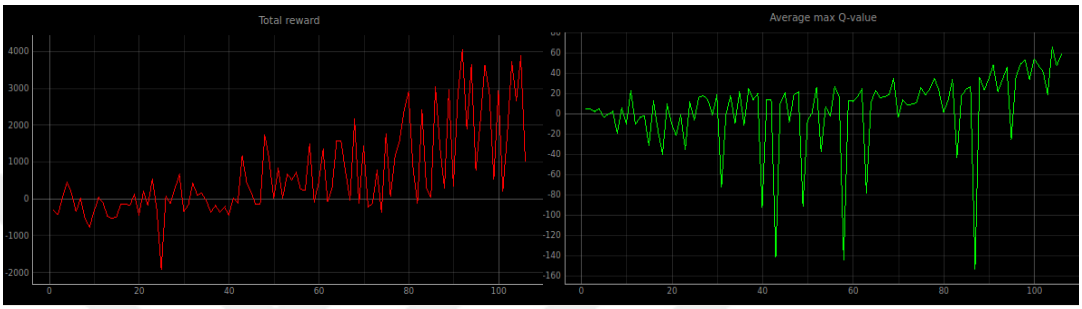
Bu ortamda, Ortam-5'te uygulanan sistem ve modeller uygulanmıştır. Eğitim işlemi gerçekleşirken elde edilen başarım grafiği ile Q tablosu grafiği, anlık ve toplam ödül değeri, anlık olarak 3 saniyede bir rastgele seçilen eylemler, Şekil 5.15'te görülmektedir.




**Şekil 5.15.** DQA uygulanan Ortam-6

Şekil 5.15'teki kesit, eğitimin ilk başlarında öğrenemediği görülen robotun, yavaş yavaş öğrenmeye başladığını ve çalışan programın, seçilen durum-eylem çiftlerini daha doğru seçmeye çalıştığı görülmektedir.

Şekil 5.14'teki ortama, konumu belli olmayan 4 engel yerleştirilmiştir. Robotun çevredeki engellere, duvarlara ve hedefe olan uzaklık bilgileri toplanarak DQA uygulanmıştır. Elde edilen grafikler, Şekil 5.16'de gösterilmiştir.



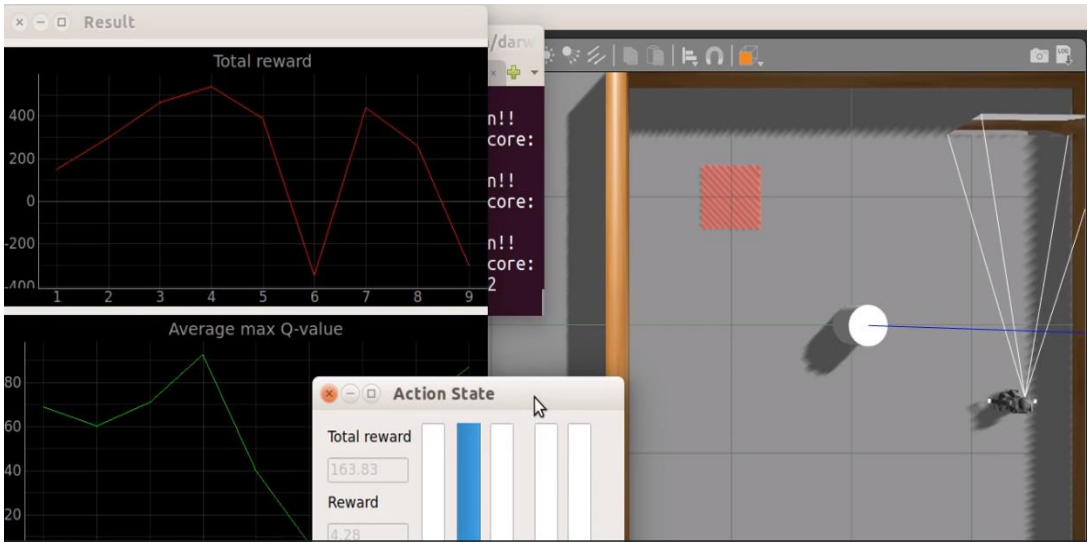

**Şekil 5.16.** DQA ile elde edilen ödül ve Q-değerleri

Şekil 5.16'deki eğitim grafikleri, 1.Nesil İntel i5 1.66 GHz CPU işlemci ile elde edilmiştir. İlk 50 epochta, oluşturulan eğitim modeli, bulunduğu durumlara karşı yanlış yönelmeler yaptığı için, oluşturulan ödül denkleminde verilen ceza puanları fazla olduğu görülmektedir. Alınan ödül, alınan cezadan azdır. Bu, modelin öğrenmeye çalıştığını gösterir. Daha sonraki epochlarda alınan ödül artmaya başladığı için model öğrenmektedir. Alınan ceza puanlarının, alınan ödüllerden daha az olduğunu ifade eder. Yani, hangi durumda hangi eylemi gerçekleştireceğini öğrenmiştir. Elde edilen Q değerleri incelendiğinde, ilk 50 epochta, %50 oranında negatif değerler aldığı görülmüştür. Daha sonraki epochlarda, aldığı negatif değerler, %10'lara kadar düşmüştür. Yani, bir duruma giderken, uygulaması gereken eylemleri, doğru şekilde seçmeye başlamış demektir.

Bu ortamda durum sayısı artırılmıştır. Bunun sonucunda, daha yavaş ve geç öğrenen bir ortam olduğu gözlemlenmiştir. Ortam-5'te uygulanan ödül denklemleri kullanılmış, ödül ve ceza puanları artırılmıştır. Bunun sonucunda ise gecikme süresi gittikçe artmaya başlamıştır. Robot, elde edilen ceza puanlarının bulunduğu durum eylem çiftleri arasında, en iyiyi aramaya başlamıştır.

**Deney 9: Süreksiz Durumda Tek Engel Bulunan Ortam**

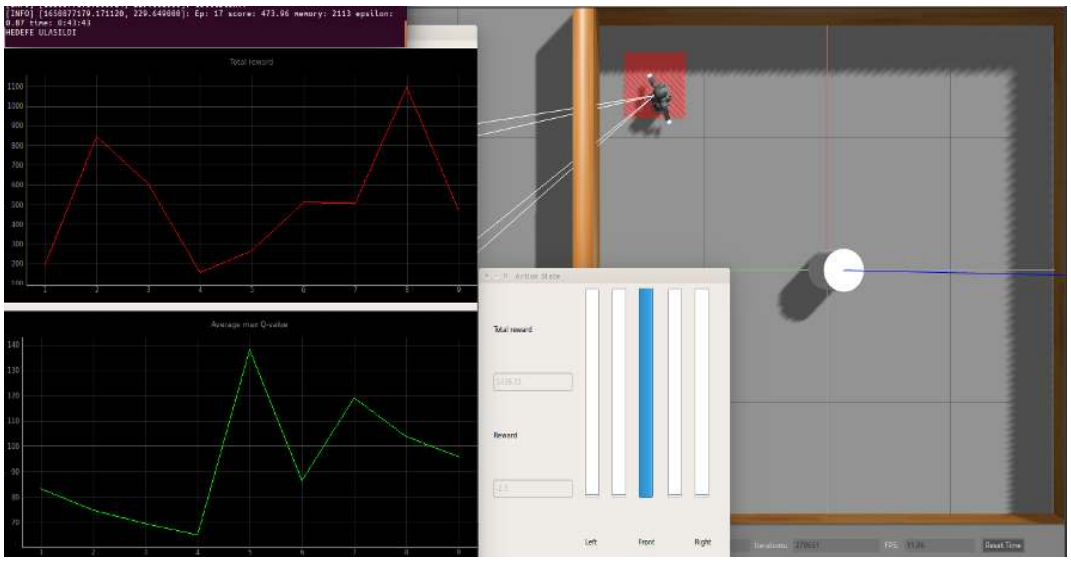
Şekil 5.17'da tezi kapsayacak şekilde hazırlanmış bir ortamda, başlangıç noktasından hedefe giderken, derin pekiştirme yöntemleri kullanılarak, insansı robotun verilen görevi gerçekleştirmeye çalıştığı gözlemlenebilmektedir.




**Şekil 5.17.** Yeni ortamda uygulanan DQA ile elde edilen sonuçlar

Tezde insansı robotun engelleri ve hedefi bulabilmesi için DQA Algoritması kullanılmıştır. Bilgisayar ile görü alanlarında, gerçek hayattaki nesneleri tanımada, bu nesnelerden bir eylem almada ve bunlara göre bazı olaylar gerçekleştirmede kullanılır.

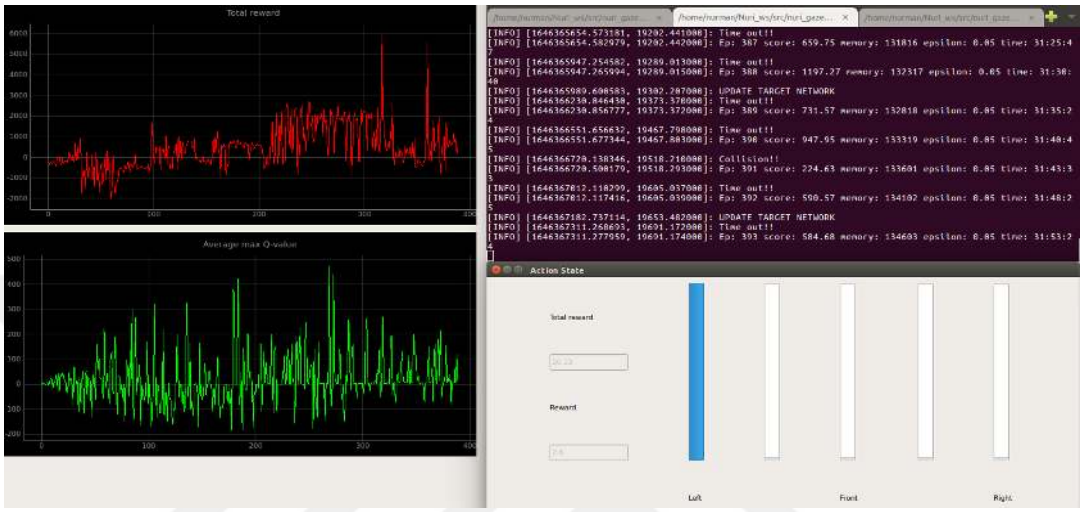
Eğitim için üç farklı model uygulanmıştır. Birinci modelde, robot sürekli aynı hedefe giderek eğitilmektedir. 1.Nesil İntel i5 1.66 GHz CPU işlemci ile elde edilen grafikler Şekil 5.18'te gösterilmiştir.



**Şekil 5.18.** Birici model için elde edilen ödül, Q-değerleri ile eğitimden bir kesit

Robot her durum eylem çiftinde başlangıç noktasına geri döner ve hedefe hareket etmeye başlar. Eğitim, kayıp değeri 0.05'ten az olduğunda ve ödül artmaya devam ettiğinde eğitim başarı ile gerçekleştirilmiş olur.

İkinci modelde, hedefe ulaşıldığında, aynı döngü (epoch) devam ederken, yeni bir hedef belirlenmiş ve daha iyi öğrenme için çoklu hedef sistemi uygulanmıştır. Bu eğitimde, 1.Nesil İntel i5 1.66 GHz CPU işlemci ile otuz iki saatte elde edilmiş grafikler Şekil 5.19'de gösterilmiştir.




**Şekil 5.19.** DQA ile ikinci model için elde edilen, toplam ödül,anlık ödül ve Q-değerleri

İlk 200 epochta, %30 oranında negatif ödül alırken ve ortalama Q değerleri incelendiğinde, %60 oranında yanlış eylemler seçildiği görülmektedir. Daha sonraki epochlarda %80 oranında pozitif ödül ile %70 oranında, bulunan duruma karşı doğru eylemler seçildiği, Şekil 5.19'de görülmektedir.

Üçüncü modelde, üç eylem ve beş durum içerisine hapsedilecek şekilde hem işlem yükü azaltılmaya çalışılmış hem de eğitim süresi azaltılmaya çalışılmıştır. Tüm modeller, simülasyon ortamında başarı ile programlanmıştır. Benzer sonuçlar elde edilmiştir.

Birinci durumda; robot hedefe giderken, oluşan sürekli durumda 5 sn. aralıklarla alınan 100 adet görüntü ile bağımsız verimiz elde edildi. Google Colab ortamında bize sunulan CPU desteği ile derin öğrenme yöntemi kullanılarak %70 başarı elde etmiş model oluşturuldu. Bu model için 300x300 boyutunda alınan görüntülerden, test ve eğitim verileri elde edildi. 300x300'lük her bir piksel yan yana olacak şekilde vektör haline getirilip 90000 giriş elde edildi. İşlem yükünü azaltmak için küçültme işlemi yapılabilir ve piksel değerlerini 0 ve 1 olarak çevirme işlemi yaparak, daha yüksek başarı elde edilebilir. Girişe, 3x3'lük bir filtre uygulayıp evrişim işleminden geçirdikten sonra daha belirgin değerler oluşmaya başlar. Boyutlarda değişim olabilir. Evrişim formülü:

$$y[a, b] = \sum_j \sum_i x[i, j] * h[a - i, b - j] \quad (5.1)$$

Çıkış matris boyutu:

$$y_{mxm} \rightarrow m = n - f + 1 \quad (5.2)$$

m, çıkış matrisi boyutudur. n, giriş matrisinin boyutu ve f'de filtre boyutumuzdur. x, giriş matrisi olmak üzere h matrisi, elimizde bulunan 3x3'lük filtreyi ifade eder. Fakat boyut değişimine maruz kalmadan hesaplama yaparak işlem yükü azaltılmış olur. Bunun için piksel ekleme yöntemi kullanılır. Piksel ekleme ile çıkış matris boyutu formülü:

$$y_{mxm} \rightarrow m = n + 2p - f + 1 \quad (5.3)$$

Piksel ekleme miktarı p ile gösterilir. Amaç, giriş boyutu ile çıkış boyutunu eşitlemektir. Eklenilecek piksel adedi, filtreye uygun olarak belirlenir. Piksel ekleme yöntemi ile eklenen piksellere 0 veya giriş matrisinin komşu olduğu eklenen piksellere kopyalama işlemi ile daha iyi evrişim işlemi yapılır. Piksel ekleme:

$$p = \frac{(f-1)}{2} \quad (5.4)$$

ile hesaplanır. Eşit boyutlarda çıkış elde etikten sonra ortaklama veya havuzlama yöntemi ile boyut azaltma işlemi uygulanır. Çünkü, boyut işlem yüküdür. Boyut azaltarak hassasiyeti ve boyutu azaltmış oluruz. Eğitimde en büyük ortaklama ile işlem yapılmıştır. Uygulanacak filtre boyutu 2x2 olup, adım kaydırma 1 olarak belirlenmiştir. Böylece girişteki her 4 pikselden en büyük piksel değeri alınıp boyut 4'te 1oranında küçültülmüş olur. Aşırı uydurmayı önleyebilmek için oluşan nöronların 4'te 1'ini unutma işlemi yapıyoruz. Elde edilen tüm nöronları 1 satır halinde vektöre dönüştürüyoruz. Daha sonra YSA ile tam bağlantı sistemi kuruluyor. Boyut düşürülüyor. En son katmanda çıkış olarak, ilk başta belirtildiği gibi ileri, sağa dönme ve sola dönme eylemlerini ifade eden 3 çıkış değeri elde ediliyor. Dolayısı ile 100 görsel ile eğittiğimiz modeli kullanarak, farklı görseller girişe verilince çıkış olarak 3 eylemden hangisinin gerçekleştirileceği %70 başarı ile elde edilmiş oluyor. Aşağıda eğitim parametreleri ile sonuç bilgileri bulunmaktadır.

Bu eğitimde, girişe verilen 100 adet görüntünün her biri için ayrı ayrı çıkışlarının bulunduğu Q tablosu oluşturulmuştur. Oluşturulan Q tablosunun ilk 10 resim için elde edilen değerleri, Tablo 5.2'de gösterilmiştir. Tablo 5.1'de ise gerçek eylemleri ifade eden çıkışlar gösterilmektedir.

**Tablo 5.1.** Eğitim için verilen gerçek çıkışlar

| Gerçek Çıkış | İleri | Sağ | Sol |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 2 | 1 | 0 | 0 |
| 3 | 1 | 0 | 0 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 0 |
| 6 | 1 | 0 | 0 |
| 7 | 1 | 0 | 0 |
| 8 | 1 | 0 | 0 |
| 9 | 1 | 0 | 0 |
| 10 | 1 | 0 | 0 |

**Tablo 5.2.** Elde edilen Q tablosunun ilk 10 tanesi

| Q Tablosu | İleri | Sağ | Sol |
|---|---|---|---|
| 0 | 0.64356935 | 0.18055163 | 0.17587905 |
| 1 | 0.44480312 | 0.18606915 | 0.36912766 |
| 2 | 0.6824496 | 0.15535331 | 0.16219707 |
| 3 | 0.48314962 | 0.17440721 | 0.34244323 |
| 4 | 0.46986023 | 0.21005628 | 0.32008347 |
| 5 | 0.41232565 | 0.17149583 | 0.4161785 |
| 6 | 0.41047442 | 0.21104737 | 0.3784783 |
| 7 | 0.6519757 | 0.15530123 | 0.19272314 |
| 8 | 0.42363912 | 0.19495754 | 0.38140333 |
| 9 | 0.44562575 | 0.21937442 | 0.33499986 |
| 10 | 0.5836137 | 0.21352573 | 0.20286049 |

Açı değeri robotun bulunduğu yöne göre önceki durumu göze alarak hesaplanan açı olmakla birlikte i değer ise her durum için verilebilecek 5 eylemi ifade etmektedir. Her eylem için

açı ayrı ayrı hesaplanır ve 5 açı değeri bulunur. Mevcut uzaklık, robotun harekete başladığı ilk andan, hareketin sonlandığı ana kadar değişebilen, hedefe olan uzaklığıdır. Hedefe uzaklık ise robotun ilk anda bulunduğu konumdan hedefe olan uzaklığıdır.

Öğrenme işlevi, 20000 döngü içerisine hapsedilmiştir. Bu değer değiştirilebilir. Her döngü, hedefe ulaşınca veya herhangi bir nesneye çarpınca sonlanır. Döngü içerisinde birçok durum ve eylem çifti elde edilir. Elde edilen (s,a) çiftlerinden en yüksek başarıma sahip olanlar kayıt altında tutulur. Q öğrenme bu işi yapar. Eğitim tamamlandığında oluşturulan Q tablosundan en iyi durum-eylem çiftleri seçilerek hedefe en kısa yoldan ulaşılmış olur. Durum [-4], önceki durumu ifade ederken, durum [-3], mevcut durumu ifade eder. Ödül için oluşturulan denklemler ile ortama uyumluluk bakamından başarılı olduğu düşünülmüştür. Denklemler geliştirilebilir. Elde edilen çıkışlar, Tablo 5.3'te görülmektedir.

**Tablo 5.3.** İlk 10 giriş için elde edilen QMax değerleri

| QMax | İleri | Sağ | Sol |
|---|---|---|---|
| 0 | 1 | 0.18055163 | 0.17587905 |
| 1 | 1 | 0.18606915 | 0.36912766 |
| 2 | 1 | 0.15535331 | 0.16219707 |
| 3 | 1 | 0.17440721 | 0.34244323 |
| 4 | 1 | 0.21005628 | 0.32008347 |
| 5 | 0.41232565 | 0.17149583 | 1 |
| 6 | 1 | 0.21104737 | 0.3784783 |
| 7 | 1 | 0.15530123 | 0.19272314 |
| 8 | 1 | 0.19495754 | 0.38140333 |
| 9 | 1 | 0.21937442 | 0.33499986 |
| 10 | 1 | 0.21352573 | 0.20286049 |

Daha sonra yapılan eğitim, daha gelişmiş bir kodlama ile yapılmıştır. Bunun için, bulunan hatayı yayarak geri dönüşüm yapılıp, nöronlara verilen ağlar güncellenerek daha iyi bir öğrenmeye çalışıldı. Pekiştirmeli Öğrenmeye uygun ortam hazırlanıp, daha iyi bir öğrenme için gerekli olan parametrelerde kontrol edilerek eğitim gerçekleştirildi. DQA için sürekli ve süreksiz sistemde ve farklı ortamlarda engelden kaçarak hedefe giden robot hedefe ve engelle yakınlık durumlarına göre ödül denklemleri uygulanarak, öğrenme eğitimini pekiştirmiş oldu. Sürekli sistemde, her hedefe ulaştığında tasarlanan ortamın içinden rastgele bir şekilde yeni hedef atanarak, tüm ortam öğrenilmeye çalışıldı. Süreksiz sistemde ise yeri belirsiz engeller ve hedefi verilen ödül denklemlerinden gelen puanlar ile bulmaya çalışarak hem engelden kaçıldı hem de hedefe giderek

eğitim bitirildi. Sürekli sistem, süreksiz sisteme göre daha uzun eğitim süresine sahiptir. Eğitim bittiğinde belirlenen hedefe gitmesi için gereken başarı oranı süreksiz sisteme göre daha yüksektir.

# 6. SONUÇLAR

Bu tez, insansı robotun engellerden kaçınarak hedefe yürümesi için, yeni geliştirilmiş algoritmalar içermektedir. Robotun, belirtilen bir hedefe, kendi kendine öğrenerek gidebilmesi için yapay zekâ algoritmalarının, var olan robot sistemine adapte edilmesi amaçlanmıştır. İlk olarak robotun sistemi, Linux işletim sistemine uyarlanmıştır. Çalışma alanı oluşturulup, hareket etme, DQA eğitimi, hedef alanı gibi gerekli alt yapılar için algoritmalar hazırlanmıştır. Engelleri ve hedefi saptayabilmek için görüntü işleme yardımı ile nesne algılama yapılmış ve aynı işlem için ortamdaki tüm nesnelerin konumları kontrol edilerek, robot sürekli ortamda hedefe doğru hareket ettirilmiştir. Çeşitli ortamlar hazırlanmış ve farklı modeller oluşturulmuştur. Robot modeli üzerine lazer sensör eklenmiş ve eğitim için gerekli veriler elde edilmiştir. Elde edilen veriler, derin öğrenme yöntemi kullanılarak eğitilmiş, denetimli şekilde öğrenme gerçekleşmiştir. Bu eğitimde, %70 başarı elde edilmiştir. Daha sonra, daha yüksek ödül elde edebilen eğitim modelleri oluşturulmuştur.

Bu tezde uygulanan teknikler ve tasarlanan denklemler gelişime açıktır. DQA için tasarlanan ödül denklemleri değiştirilerek daha iyi öğrenme sağlanabilir. Ortamda oluşturulan tasarım değiştirilerek daha iyi bir öğrenme elde edilebilir. Robotun gördüğü alanları, görüntü işleme metotları ile daha belirgin şekilde saptamakla ve görüntü boyutlarının küçültülmesi ile işlem yükünü azaltarak daha iyi bir eğitim gerçekleştirilebilir. Oluşturulan robotun hareketlerinde, bazı problemler gözlemlenmiştir. Bu problem, Webots ortamında hazır kütüphane kullanılarak oluşturulmuş Robotis-OP2 ile Gazebo ortamında oluşturduğum yürüme algoritması arasındaki farktan kaynaklandığı tespit edilmiştir. Webots ortamındaki robotun dönüşleri, keskin, hızlı ve ivme kazanmadan gerçekleştirilirken, Gazebo ortamında oluşturduğum kütüphanede, robotun dönüşleri gerçekleşirken bir ivme kazandığı ve bu ivme problemi sebebi ile hareketten 2 sn. sonrasına kadar devam eden hareket, başka eylemlerle birleştiğinden hesapta olmayan yeni eylemler oluşmaktadır. Mesela, sola dönme devam ettikten sonra ileri eylemi verildiğinde, robotun sola dönmeden ivme kazandığı için sola dönmeli ileri hareket adında yeni bir eylemin oluşması sonucunda, eğitimde, kalite düşmekte ve döngü sayısını artırarak fazladan iş yükü oluşturmaktadır. Yine robotun Webots'ta hazır olarak bulunan ama Gazebo'da, robotun daha istikrarlı hareket edebilmesi ve düşünce kalkabilmesi için tasarlanması gereken algoritmalar bulunmaktadır. Böylece daha iyi eğitim başarısı elde edilebilir olduğu apaçık ortadadır.

Webots'ta, daha dengeli Robotis-OP2'nin kontrolörü olmasına rağmen, Gazebo için bu tezde yeni bir kontrolör tasarlanmıştır. Bu sayede, her ne kadar Webots'ta hazır olan kontrolörden daha dengesiz kontrolör olsa da hazır olan kütüphanenin sınırlamalarına maruz kalmadan ve farklı programlarda kullanılabilecek şekilde kullanıma imkân veren bir kontrolör elde edilmiştir.

Aynı minvalde, Webots'ta yapılan çalışmalar [30], [31]'den farklı olarak hem Webots ortamı hem de ROS Gazebo ortamı kullanılmış ve insansı robotun engelden kaçarak hedefe gitmesi sağlanmıştır.

Bu tezde, 7 farklı ortam oluşturulmuştur. 1., 2. ve 3. ortamda 3 engel ile bir hedef belirtilerek toplamda 4 durum belirtilmiş ve 3 eylem gerçekleştirilmiştir. 1., 2. ve 3. durum, engele yaklaşma ve uzaklaşma durumudur. 4.durum, hedefe yaklaşma ve uzaklaşma durumudur. Robotun sağ, sol ve ileri yön hareketleri, 3 eylemi ifade etmektedir. Hangi durumlarda hangi eylemlerin gerçekleşmesi gerektiği, tasarlanan ortama göre değişkenlik gösterir. Oluşturulan algoritma ile durum eylem verileri doğru şekilde elde edilmiştir. Daha kaliteli verilerin elde dilmesi için farklı ortamların oluşturulması gerekmektedir. Bundan dolayı yeni bir ortam oluşturulmuştur. Kısmen daha iyi bir model oluşmuştur. 1.ortamdan farklı olarak engellere daha belirgin renkler verilmiş ve hedef için kırmızı bir top yerleştirilmiştir. Bu sayede kamera ile daha iyi saptama yapılarak, durum eylem çiftleri elde edilmeye çalışılmıştır. 3.ortam tekrardan yenilenerek daha sade ve basit bir ortam tasarlanmıştır. Oluşturulan 4.ortamda 1.ortamdaki gibi 4 durum ve farklı olarak 5 eylem mevcuttur. Eylem sayısı artırılarak daha fazla veri elde edilmiştir. Hedef noktası ile engellerin renkleri farklı seçilmiştir. Bu sayede işlem yükü azalmıştır. DQA için ön hazırlık tamamlanmıştır. Tasarlanan 6.ortamda çalışma alanı 4x4'e düşürülüp, alan duvarlarla çevrilmiştir. Bu sayede işlem yükü azalmıştır. Sürekli ortam için hedef sürekli değişkenlik gösterir. Alanın içinde rastgele birçok hedef noktası belirlenir ve rastgele bir şekilde hedef öğrenilmeye çalışılır. Model eğitildiğinde, tüm alanda, istenilen noktaya gidebilme sağlanmış olur. Ödül denklemleri ile duruma karşı gösterilecek doğru eylem, daha belirginleştirilir. Böylece, eğitim sistemi geliştirilmiştir. 7.ortamda, 6.ortamdaki altyapı kullanılarak 4 farklı engel yerleştirilmiştir. Engeller sabittir. Robotun, engelleri aşarak hedefe gitmesi amaçlanmıştır. Tüm bu deneylerden sonra, yeni bir ortam tasarlanmıştır. Ortama, 4 adet duvar engeli ve merkezde ise dairesel engel konulmuştur. Kamera işlem yükünden dolayı kaldırılmış ve yerine lazer sensor kullanılmıştır. Bu sayede işlem yükü çok düşmüştür. Robotun, derin öğrenme algoritmaları kullanarak, başlangıç noktası olarak belirlenen yerden, tüm engelleri aşıp hedef noktasına ulaşması sağlanmıştır. Son ortamda, 24 adet lazer sensor değeri ile 1 adet durum ve 1 adet eylem değeri olmak üzere elde edilen 26 eğitim verisi elde edilmiştir. Bu veriler ile eğitilen model başarılı şekilde hedefe gitmiştir.

Bu tezde, tartışılacak konular ile üzerine çalışılması gereken bulgulardan birisi; ROS Gazebo programı ile görselleştirdiğim kütüphanenin eksiklikleri ile alakalıdır. Robot hareketi için tasarlanan kodlar, simetrik olarak tasarlandığı için robotun kollarının en ufak hareketinde, robotun dengesinin bozulmasına ve düşmesine sebep olmaktadır. Bu problemlerin çözümü, robotik ile alakalıdır. Daha iyi eğitim için, daha iyi bir robotik çalışmasının yapılması gerekmektedir.

Gelecekte, 3D yazıcı ile üretilecek olan insansı robotun, Raspberry Pi 4 Model B (ROS Master) kullanılarak, aynı algoritma ile karmaşık koşullarının oluşturulduğu hem sanal hem de gerçek ortamlarda, eğitimine devam edilecektir.

# ÖNERİLER

Robotis-OP2'nin üzerinde bulunan imu sensörden yardım alınarak, dengesi bozulan robotun tekrar dengeye gelebilmesi için, robotun üzerinde bulunan 20 aktüatörün, hangi açı değerlerini alması gerektiği ile alakalı robotik ağırlıklı bir çalışmanın olması, Robotis-OP2'nin, daha insansı bir robot olması için gerekli görülmüştür. Bu sayede, robot daha dengeli yürüme yetisine sahip olacaktır.

Derin Pekiştirmeli Öğrenme yöntemlerinin, daha çok açık kaynak destekli sistemlerde çalışılıyor olması sonucunda, bu tezin konusu ile paralel olarak çok fazla çalışma mevcut olduğu için, etik kurallara ayrıca uyulması gerekmektedir.

Tezde uygulanan görüntü işleme yöntemlerinin geliştirilmesi ile daha iyi eğitim sonuçları elde edilebilir.

# KAYNAKLAR


[1] Saygin, A.P.; Roberts, Gary; Beber, Grace (2008), "*Comments on "Computing Machinery and Intelligence" by Alan Turing*", in Epstein, R.; Roberts, G.; Poland, G. (eds.), Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer, Dordrecht, Netherlands: Springer, Bibcode:2009pttt.book.....E, doi:10.1007/978-1-4020-6710-5

[2] Momentexpo, Erişim: 24-12-2021, https://www.moment-expo.com/tr/dergiler/23/nostalji/moden-gunumuze-robot-tarihi/

[3] Erlerobotics, Erişim: 24-12-2021, https://erlerobotics.gitbooks.io/erle-robotics-erle-brain-a-linux-brain-for-drones/content/en/ros/ROS.html

[4] ROS Wiki, Erişim: 24-12-2021, https://wiki.ros.org/navigation?distro=kinetic

[5] Lieberman, Hillier, "Marcov Decision Processes", Erişim: 24-12-2021, MKS Harvard lecture materials:

http://am121.seas.harvard.edu/site/wp-content/uploads/2011/03/MarkovDecisionProcesses-HillierLieberman.pdf

[6] Riedmiller, M., 2005. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In: European Conference on Machine Learning. Springer, pp. 317–328

[7] Ravichandiran, S. (2018). Hands-On Reinforcement Learning with Python: Master reinforcement and deep reinforcement learning using OpenAI Gym and TensorFlow. Packt Publishing Ltd

[8] Oh, J., Guo, X., Lee, H., Lewis, R. L., Singh, S., 2015. Action-conditional video prediction using deep networks in atari games. In: Advances in Neural Information Processing Systems. pp. 2863–2871

[9] Siegwart, Roland and Chli, Margarita, Rufli, Martin, "*Mobile Robot Kinematics Autonomous Mobile Robots*", Spring 2017, ETH Zürich, ASL(Autonomous Systems Lab)

[10] MoveIT, Erişim: 24-12-2021, https://moveit.picnik.ai/foxy/doc/getting_started.html

[11] GazeboSim, Erişim: 24-12-2021, https://gazebosim.org/tutorials?cat=guided_b&tut=guided_b2

[12] Zhang, Anfan, "*Chapter3, Forward Kinematics: The Denavit-Hartenberg Convention*"

[13] Van Hasselt, H., Guez, A. and Silver, D., 2015. Deep reinforcement learning with double Q-learning. CoRR, abs/1509.06461

[14] Illıngworth, Valerie (1997). A dictionary of computing. İnternet Archive. Oxford; New York: Oxford Universty Press. ISBN 987-0-19-280046-6

[15] ROSWiki, Erişim: 24-12-2021, http://wiki.ros.org/turtlebot/Tutorials/indigo/Turtlebot%20Installation

[16] Pandas, "*License – Package overview – pandas 1.0.0 documentation*". 28 Ocak 2020. 14 Şubat 2012 tarihinde kaynağından arşivlendi. Erişim tarihi: 30 Ocak 2020

[17] Wes McKinney (2011). "*pandas: a Foundational Python Library for Data Analysis and Statistics*"

[18] Harris, Charles R.; Millman, K. Jarrod; van der Walt, Stéfan J.; Gommers, Ralf; Virtanen, Pauli; Cournapeau, David; Wieser, Eric; Taylor, Julian; Berg, Sebastian; Smith, Nathaniel J.; Kern, Robert (17 Eylül 2020). "*Array programming with NumPy*". Nature (İngilizce). 585 (7825): 357-362. doi:10.1038/s41586-020-2649-2. ISSN 0028-0836

[19] TensorFlow official website, Erişim: 24-12-2021, https://www.tensorflow.org/

[20] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, Google Brain "*TensorFlow: A System for Large-Scale Machine Learning"*, November 2–4, 2016, Savannah, GA, USA ISBN 978-1-931971-33-1

[21] Mavi, Arda, Erişim: 24-12-2021, https://www.ardamavi.com/2017/07/sinir-aglari.html

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

[23] Karpathy, Andrej, "*CS231n Winter 2016: Lecture1: Introduction and Historical Context*", Erişim: 24-12-2021, https://www.youtube.com/watch?v=NfnWJUyUJYU&ab_channel=AndrejKarpathy

[24] Haykin, S. (2008). Neural Networks and Learning Machines: A Comprehensive Foundation. Pearson Education, Singapor

[25] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., 1989. "*Backpropagation applied to handwritten zip code recognition*.", Neural computation 1 (4), 541–551

[26] 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

[27] Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581

[28] Lample, Guillaume and Chaplot, Devendra Singh, "*Playing FPS Games with Deep Reinforcement Learning*", Erişim: 24-12-2021, https://arxiv.org/pdf/1609.05521.pdf

[29] Ertunç, Suna; Hitit,Yılmazer,Zeynep; Sert, Dilara; "*Endüstri 4.0 Uygulamaları, Mevcut Durumu ve Kimya Mühendisliğindeki Yeri*", 4. Tehlikeli Kimyasalların Yönetimi ve Proses Güvenliği, Nisan 2019, Ankara Üniversitesi

[30] M. Kirtas, K. Tsampazis, N. Passalis, A. Tefas, 2020. Webots: Symbiosis Between Virtual and Real Mobile Robots, Artificial İntelligence Applications and İnnovations, Thessaloniki, Greece

[31] P. Tiritiris, N. Passalis, A. Tefas, 2021. Temporal Difference Rewards for End-to-end Vision-based Active Robot Tracking using Deep Reinforcement Learning, Thessaloniki, Greece

# ÖZGEÇMİŞ

**Nuri Köksal VAROL**

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]