DERİN PEKİŞTİRMELİ ÖĞRENME KULLANARAK İNSANSI ROBOTLAR İÇİN İTME KURTARMA KONTROL SİSTEMİNİN GELİŞTİRİLMESİ

EMRAH ASLAN

MART 2023

DİYARBAKIR

DİCLE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DERİN PEKİŞTİRMELİ ÖĞRENME KULLANARAK İNSANSI ROBOTLAR İÇİN İTME KURTARMA KONTROL SİSTEMİNİN GELİŞTİRİLMESİ

EMRAH ASLAN

DİCLE ÜNİVERSİTESİ LİSANSÜSTÜ EĞİTİM-ÖĞRETİM VE SINAV YÖNETMELİĞİNİN BİR PARÇASI OLARAK
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANA BİLİM DALINDA
DOKTORA TEZİ
OLARAK HAZIRLANMIŞTIR

MART 2023

DİYARBAKIR

DERİN PEKİŞTİRMELİ ÖĞRENME KULLANARAK İNSANSI ROBOTLAR İÇİN İTME KURTARMA KONTROL SİSTEMİNİN GELİŞTİRİLMESİ

Emrah ASLAN tarafından Dicle Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği'nin bir parçası olarak hazırlanan bu çalışma, aşağıda bilgileri yazılı jüri üyeleri tarafından değerlendirilerek **Elektrik Elektronik Mühendisliği Ana Bilim Dalı**'nda **Doktora Tezi** olarak kabul edilmiştir.

Prof. Dr. Neslihan DALKILIÇ
Müdür, **Fen Bilimleri Enstitüsü** _____

Dr. Öğr. Üyesi Muhammet Ali ARSERİM
Danışman, **Elektrik Elektronik Mühendisliği Bölümü, Dicle Üniversitesi** _____

Prof. Dr. Ayşegül UÇAR
İkinci Danışman, **Mekatronik Mühendisliği Bölümü, Fırat Üniversitesi** _____

Sınav Jürisi:

Prof. Dr. Bilal ALATAŞ **(*)**
Yazılım Mühendisliği Bölümü, Fırat Üniversitesi _____

Dr. Öğr. Üyesi Muhammet Ali ARSERİM **(**)**
Elektrik Elektronik Mühendisliği Bölümü, Dicle Üniversitesi _____

Doç. Dr. Bilal GÜMÜŞ
Elektrik Elektronik Mühendisliği Bölümü, Dicle Üniversitesi _____

Doç. Dr. Özkan ATAN
Elektrik Elektronik Mühendisliği Bölümü, Van Yüzüncü Yıl Üniversitesi _____

Dr. Öğr. Üyesi Mesut HÜSEYİNOĞLU
Makina Mühendisliği Bölümü, Dicle Üniversitesi _____

ONAY

Savunma Tarihi: 02 / 03 / 2023

(*) Sınav Jürisi kısmının birinci satırına Jüri Başkanının bilgilerini yazınız.
(**) Sınav Jürisi kısmının ikinci satırına Tez Danışmanının bilgilerini yazınız.

Aileme…

Dicle Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırlanan bu tez çalışmasında yer alan tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu beyan ederim. Ayrıca, bahse konu bu kural ve ilkelerin gerektirdiği üzere, bu çalışmada özgün olmayan tüm bilimsel içerikleri kurallara uygun biçimde alıntılayıp kaynak gösterdiğimi beyan ederim. Beyanımla çelişen herhangi bir delil bulunduğu takdirde tüm sorumluluğu üstleneceğimi kabul ederim.

Ad, Soyad: Emrah ASLAN

İmza: …….

TEŞEKKÜR

Öncelikle tez çalışmamın fikir aşamasından son aşamasına kadar ki süreçte bilgisi, birikimi ve tecrübesiyle bana rehberlik eden danışman hocam Dr. Öğr. Üyesi Muhammet Ali ARSERİM'e ve ikinci danışman hocam Prof. Dr. Ayşegül UÇAR'a çok teşekkür ederim. Tez çalışmam süresince hazırladığım raporları değerlendirerek çalışmamın daha nitelikli hale gelmesini sağlayan tez izleme jüri üyelerim Doç. Dr. Bilal GÜMÜŞ ve Dr. Öğr. Üyesi Mesut HÜSEYİNOĞLU'na teşekkür ederim.

Doktora sürecimde beni destekleyen Prof. Dr. Sedat İLHAN'a, Dr. Öğr. Üyesi Orhan ARPA'ya, Dr. Emre ERKAN'a ve Müh. Ali BİÇER'e teşekkür ederim. Tez çalışmama yapmış olduğu olumlu eleştiri ve yönlendirmelerinden dolayı Dr. Öğr. Üyesi Yıldırım ÖZÜPAK'a teşekkür ederim.

Çalışmalarım süresince her türlü fedakârlığı gösteren eşim Atike Esra ASLAN'a, kızım Zümra Hilal'e teşekkürü bir borç bilirim. Son olarak hayatımın her aşamasında verdikleri sonsuz desteklerden dolayı anneme, babama, ablama ve kardeşlerime sonsuz teşekkürlerimi ve minnetlerimi sunarım.

İÇİNDEKİLER

TEŞEKKÜR v

İÇİNDEKİLER vi

ŞEKİLLER LİSTESİ viii

TABLOLAR LİSTESİ xi

SİMGELER VE KISALTMALAR LİSTESİ xii

ÖZET xiv

ABSTRACT xv

1. GİRİŞ 1

1.1 Robotların Tarihsel Gelişimi 3

1.2 Literatür Araştırması 6

1.3 Tez Amacı 10

1.4 Tezin İçeriği 11

2. İKİ AYAKLI İNSANSI ROBOTLAR 13

3. ROBOTİK SİMÜLATÖRLER 22

3.1 V-Rep 23

3.2 Webots 25

3.3 Gazebo 27

3.4 Robot İşletim Sistemi 28

3.5 MobileSim 29

3.6 MATLAB 30

4. İTME-KURTARMA KONTROLÖRLERİ 32

4.1 İnsansı Yürüme 32

4.2 İnsansı Robotlarda Denge Stratejileri 35
4.2.1 Ayak bileği stratejisi 36
4.2.2 Kalça stratejisi 36
4.2.3 Adım stratejisi 38

5. KONTROL YÖNTEMLERİ 39

5.1 Yapay Sinir Ağları 40

5.2 Derin Pekiştirmeli Öğrenme 44

5.3 Derin Pekiştirmeli Öğrenme Algoritmaları 46
5.3.1 Q öğrenme 46
5.3.2 Derin q ağı 48
5.3.3 Çift derin q ağı 52

5.4 PID Kontrolör 55

5.5 Model Öngörülü Kontrol 57

6. TEZİN UYGULAMA AŞAMASI 61

6.1 Yeni Proje Oluşturma 61

6.2 Projeye Kontrolör Ekleme 65

6.3 Dış Kuvvet Uygulama 69

6.4 Kontrolörlerin Programlanması 73

7. BULGULAR VE TARTIŞMA 80

8. SONUÇLAR 90

KAYNAKLAR 92

ÖZGEÇMİŞ 97

ŞEKİLLER LİSTESİ

Şekil 1.1 a) Hero'nun buhar motoru b) Hero'nun rüzgar enerjisiyle çalışan otomatı [48] 4
Şekil 1.2 El-Cezeri'nin filli su saati [49] 4
Şekil 1.3 Leonardo da Vinci'nin sürücüsüz arabası [50] 5
Şekil 1.4 Honda tarafından geliştirilen insansı robot Asimo [52] 6
Şekil 2.1 Waseda Üniversitesi tarafından geliştirilmiş olan Wabot-1 [4] 13
Şekil 2.2 Sabancı Üniversitesi tarafından geliştirilen Suralp [45] 14
Şekil 2.3 Boston Dynamics tarafından geliştirilmiş olan Atlas [51] 15
Şekil 2.4 Honda Motor tarafından geliştirilmiş olan Asimo [52] 16
Şekil 2.5 Robotis firması tarafından üretilen Thormang3 [53] 17
Şekil 2.6 Sony firması tarafından üretilen Qrio 18
Şekil 2.7 SoftBank Robotics firması tarafından üretilen Nao 18
Şekil 2.8 Robotis firması tarafından üretilen Robotis-OP2 19
Şekil 2.9 Robotis firması tarafından üretilen Robotis-OP3 20
Şekil 3.1 V-Rep Simülâtör ekran görüntüsü örneği 24
Şekil 3.2 Webots Simülâtör ekran görüntüsü örneği 25
Şekil 3.3 Gazebo Simülâtör ekran görüntüsü örneği 27
Şekil 3.4 MobileSim Simülâtör ekran görüntüsü örneği 29
Şekil 3.5 MATLAB Simulinkte robot programlama ekran görüntüsü örneği 31
Şekil 4.1 İnsansı hareket referans düzlemi 33
Şekil 4.2 İnsansı robotlarda yürüme döngüsünün önden görünümü 34
Şekil 4.3 Üç temel dengeleme stratejisi. Yeşil nokta kütle merkezini, pembe nokta basınç merkezini ve mavi ok yer reaksiyon kuvvetini temsil eder. a) Ayak bileği stratejisi b) Kalça stratejisi c) Adım sratejisi [1] 35
Şekil 4.4 a) Ayak bileği stratejisi b) Kalça stratejisi c) Adım stratejisi 37
Şekil 5.1 Açık çevrim kontrol sistemi blok diyagramı 39
Şekil 5.2 Kapalı çevrim kontrol sistemi blok diyagramı 39
Şekil 5.3 Biyolojik nöron hücresinin yapısı 41
Şekil 5.4 Yapay sinir ağı nöron yapısı 42
Şekil 5.5 Yapay sinir ağı modeli 43
Şekil 5.6 Pekiştirmeli öğrenme blok diyagramı 45
Şekil 5.7 Q öğrenme akış şeması 47

Şekil 5.8 a) Q öğrenme algoritması b) DQA algoritması karşılaştırılması 48
Şekil 5.9 DQA akış diyagramı 49
Şekil 5.10 ÇDQA akış diyagramı 53
Şekil 5.11 PID kontrolör blok diyagramı 55
Şekil 5.12 MÖK kontrolör blok diyagramı 57
Şekil 6.1 Webots ortamında yeni proje dosyası oluşturma-1 61
Şekil 6.2 Webots ortamında yeni proje dosyası oluşturma-2 62
Şekil 6.3 Webots ortamında yeni proje dosyası oluşturma-3 62
Şekil 6.4 Webots ortamında yeni proje dosyası oluşturma-4 63
Şekil 6.5 Webots ortamında yeni proje dosyası oluşturma-5 63
Şekil 6.6 Webots ortamına zemin ekleme 64
Şekil 6.7 Webots simülâtör ortamı 64
Şekil 6.8 Webots simülâtör ortamında Robotis-OP2 65
Şekil 6.9 Webots ortamında yeni kontrolör oluşturma-1 66
Şekil 6.10 Webots ortamında yeni kontrolör oluşturma-2 66
Şekil 6.11 Webots ortamında yeni kontrolör oluşturma-3 67
Şekil 6.12 Webots ortamında yeni kontrolör oluşturma-4 67
Şekil 6.13 Webots ortamında nesneye kontrolör ekleme-1 68
Şekil 6.14 Webots ortamında nesneye kontrolör ekleme-2 68
Şekil 6.15 Robota manuel olarak itme kuvveti uygulanması 69
Şekil 6.16 Robota fizik eklentisi eklemek-1 70
Şekil 6.17 Robota fizik eklentisi eklemek-2 70
Şekil 6.18 Robota fizik eklentisi eklemek-3 71
Şekil 6.19 Robota fizik eklentisi eklemek-4 71
Şekil 6.20 Robota fizik eklentisi eklemek-5 72
Şekil 6.21 Robota uygulanan kuvvet grafiği 74
Şekil 6.22 Oluşturulan yapay sinir ağı mimarisinin yapısı 74
Şekil 6.23 İtme-Kurtarma kontrollü insansı robotun hareket akış diyagramı 75
Şekil 6.24 Robot penceresi 78
Şekil 6.25 Programı gönder simgesi 78
Şekil 6.26 Programı kaldır simgesi 79
Şekil 6.27 Uzaktan kumanda simgesi 79
Şekil 6.28 Durdurma simgesi 79

Şekil 7.1 a) Arkadan itme b) İtme sonucu aldığı pozisyon c) Denge pozisyonu 80
Şekil 7.2 a) Önden itme b) İtme sonucu aldığı pozisyon c) Denge pozisyonu 81
Şekil 7.3 DQA algoritma sonuçları 82
Şekil 7.4 ÇDQA algoritma sonuçları 83
Şekil 7.5 DQA ve ÇDQA algoritma sonuçların karşılaştırılması 83
Şekil 7.6 PD kontrol sonuçları 84
Şekil 7.7 MÖK kontrol sonuçları 85
Şekil 7.8 PD ve MÖK kontrol sonuçlarının karşılaştırılması 85
Şekil 7.9 Gerçek dünya test görüntüsü 87
Şekil 7.10 Arkadan uygulanan kuvvetten sonra ayak bileğinin pozisyonu 87
Şekil 7.11 Önden uygulanan kuvvetten sonra ayak bileğinin pozisyonu 88
Şekil 7.12 Arkadan uygulanan farklı kuvvetlere göre ayak bileği tarafından üretilen tork değerleri 88
Şekil 7.13 Önden uygulanan farklı kuvvetlere göre ayak bileği tarafından üretilen tork değerleri 89

TABLOLAR LİSTESİ

Tablo 5.1 Q öğrenme algoritması 48
Tablo 5.2 DQA algoritması 50
Tablo 5.3 ÇDQA algoritması 54
Tablo 6.1 DQA ve ÇDQA algoritması parametreleri 76
Tablo 6.2 DQA ve ÇDQA algoritması yapay sinir ağı özellikleri 76
Tablo 6.3 PD kontrolör parametreleri 77
Tablo 7.1 Algoritma bölüm sayıları 80
Tablo 7.2 Algoritma sonuçlarının karşılaştırılması 86

SİMGELER VE KISALTMALAR LİSTESİ

Simge	Açıklama
m	Kütle
g	Yerçekimi
ẍ	Gövdenin Doğrusal Hızı
z_0	Kütle Merkezinin Yüksekliği
ε	Epsilon Açıkgözlü Parametresi
Q	Q-Değeri
Q^E	Q Değerlendirme Ağı
Q^T	Q Hedef Ağı
K_p	Oransal Kazanç Katsayısı
K_i	İntegral Kazanç Sabiti
K_d	Türev Kazanç Sabiti
F	Kuvvet
f	Kuvvetin İşareti
T	Tork
t	Torkun İşareti
S	Durum
S'	Sonraki Durum
A	Eylem
R	Ödül
P	Konum
γ	İndirim Oranı

Kısaltma	Açıklama
A2C	Avantaj Aktör-Kritik
A3C	Asenkron Avantaj Aktör-Kritik
CMP	Merkezi Moment Ekseni
COM	Kütle Merkezi
COP	Basınç Merkezi
CP	Yakalama Noktası
DOF	Serbestlik Derecesi
DÖ	Derin Öğrenme
DPÖ	Derin Pekiştirmeli Öğrenme
DQA	Derin Q-Ağı
ÇDQA	Çift Derin Q-Ağı
D3QA	Düello Çift Derin Q-Ağı
BDQA	Bulanık Derin Q-Ağı
M.Ö.	Milattan Önce
MÖK	Model Öngörülü Kontrol - Model Predictive Control (MPC)
M.S.	Milattan Sonra
PID	Oransal, İntegral, Türev (Proportional, Integral, Derivate)
PÖ	Pekiştirmeli Öğrenme
PPO	Yaklaşık Politika Optimizasyonu
DDB	Doğrultulmuş Doğrusal Birim (Rectified Linear Unit-ReLU)
RİS	Robot İşletim Sistemi
YSA	Yapay Sinir Ağları

ÖZET

DERİN PEKİŞTİRMELİ ÖĞRENME KULLANARAK İNSANSI ROBOTLAR İÇİN İTME KURTARMA KONTROL SİSTEMİNİN GELİŞTİRİLMESİ

Aslan, Emrah
Doktora, Elektrik Elektronik Mühendisliği Bölümü
Danışman: Dr. Öğr. Üyesi Muhammet Ali Arserim
İkinci Danışman: Prof. Dr. Ayşegül Uçar
Mart 2023, 114 sayfa

Bu tezin amacı, iki ayaklı insansı bir robot için bir insanın eylemlerini taklit edebilecek tamamen bağımsız bir itme-kurtarma kontrol sistemi tasarlamak ve robota uygulamaktır. Bu çalışmada, dış kuvvetlerden ve itmelerden etkilenen iki ayaklı insansı robotların itme-kurtarma problemine odaklanılmıştır. İnsansı robotlar denge açısından yapısal olarak kararsız olduklarından dolayı bu problem robotlarda önemli sorun olarak ortaya çıkmaktadır. Robotlarda, itme-kurtarma kontrolörleri ayak bileği, kalça ve adım olmak üzere 3 stratejiden oluşmaktadır. Bu stratejiler, insanların denge bozukluğu durumlarında gösterdikleri biyomekanik tepkilerdir. Bu tezde, insansı robotların ayakta dururken ya da yürürken dengede kalabilmesi ve dış kuvvetlerden kaynaklanabilecek denge bozukluklarının önlenmesi için aktif bir denge kontrolü sunulmuştur. Buna ilişkin yapılan çalışmada hem simülâsyon hem de gerçek dünya testleri yapılmıştır. Çalışmanın simülâsyon testleri Webots ortamında 3 boyutlu modeller ile gerçekleştirilmiştir. Gerçek dünya testleri ise Robotis-OP2 insansı robot üzerinde yapılmıştır. Robot üzerinde bulunan sensörlerden jiroskop, ivmeölçer ve motor verileri kaydedilip, robota harici itme kuvveti uygulanmıştır. Kaydedilen bu veriler ve ayak bileği stratejisi kullanılarak robotun dengesi sağlanmıştır. Bunun için klasik kontrol yöntemi olarak PD kontrolör ve tahmine dayalı olan Model Öngörülü Kontrol (MÖK) yöntemi ile de robotun kontrolü sağlanmıştır. Bununla birlikte robotun tamamen otonom hale getirilebilmesi için Derin Pekiştirmeli Öğrenme (DPÖ) algoritmalarından Derin Q Ağı (DQA) ve Çift Derin Q Ağı (ÇDQA) yöntemleri uygulanmıştır. Bu uygulamalarda robota hem önden hem arkadan kuvvetler uygulanmıştır. Simülâsyon ortamında yapılan test çalışmalarında robotun her iki durumda da gelen itmelere karşı ayakta kaldığı gözlemlenmiştir. Uygulanan dört farklı kontrol yönteminden en iyi sonuçları, ÇDQA algoritması vermiştir. Gerçek ortam testlerinde alınan sonuçlar simülâsyon sonuçlarına paralellik göstermiştir.

Anahtar Kelimeler: İtme-Kurtarma, İnsansı Robot, Robotis-OP2, Derin Pekiştirmeli Öğrenme (DPÖ)

ABSTRACT

DEVELOPMENT OF PUSH-RECOVERY CONTROL SYSTEM FOR HUMANOID ROBOTS USING DEEP REINFORCEMENT LEARNING

Aslan, Emrah
PhD of Science in Department of Electrical and Electronics Engineering
Supervisor: Dr. Öğr. Üyesi Muhammet Ali Arserim
Co-Supervisor: Prof. Dr. Ayşegül Uçar
March 2023, 114 pages

The aim of this thesis is to design and implement a completely independent push-and-rescue control system for a bipedal humanoid robot that can imitate the actions of a human. In this study, the thrust-recovery problem of bipedal humanoid robots affected by external forces and thrusts is focused. Since humanoid robots are structurally unstable in terms of balance, this problem emerges as an important problem in robots. In robots, push-rescue controllers consist of 3 strategies: ankle, hip and step. These strategies are biomechanical responses that people show in cases of balance disorder. In this thesis, an active balance control is presented in order to keep the humanoid robots in balance while standing or walking and to prevent balance disorders that may be caused by external forces. In this study, both simulation and real-world tests were conducted. The simulation tests of the study were carried out with 3D models in the Webots environment. Real-world tests were conducted on the Robotis-OP2 humanoid robot. The gyroscope, accelerometer and motor data from the sensors on the robot were recorded and external thrust was applied to the robot. The balance of the robot was ensured by using these recorded data and the ankle strategy. For this, the control of the robot is provided with the PD controller as the classical control method and the Model Predictive Control (MPC) method, which is based on prediction. In addition, Deep Q Network (DQN) and Double Deep Q Network (DDQN) methods from Deep Reinforcement Learning (DRL) algorithms have been applied in order to make the robot fully autonomous. In these applications, both front and rear forces were applied to the robot. In the test studies carried out in the simulation environment, it has been observed that the robot survives the pushes in both cases. The DDQN algorithm gave the best results from the four different control methods applied. The results obtained in the real environment tests showed parallelism with the simulation results.

Keywords: Push-Recovery, Humanoid Robot, Robotis-OP2, Deep Reinforcement Learning

1. GİRİŞ

Antik çağlardan günümüze kadar insan ve hayvan hareketleri araştırmacılar için önemli konular arasında yer almıştır. Bu sebepten birçok araştırmacı insan ve hayvan hareketlerini taklit eden makinalar üzerinde çalışmıştır.

Teknolojik gelişmeler ve insan hayatını kolaylaştırma çabaları robotlar üzerine olan çalışmaları hızlandırmıştır. Bu robotlar, insanlar için tehlikeli olan işlerde veya insan gücünün yetersiz kaldığı yerlerde kullanmak üzere geliştirilmişlerdir. Robotik çalışmalarında hedeflenen durum insan gücüne gerek kalmadan verilen görevi yerine getiren ve düşünebilen robotlar üretmektir. Fakat robotlar bazı insan hareketlerini taklit etme ve insan gibi düşünme konusunda yetersiz kalmaktadırlar.

Son yıllarda yapılan robotik çalışmalarda, gelişmiş sensörler ve yapay zekâ teknolojileri kullanılarak insan hareketlerini taklit etme konusunda daha başarılı insansı robotlar geliştirilmiştir. Bu robotlar, savunma sanayi, mühendislik uygulamaları, endüstri, tıp ve psikoloji gibi birçok alanda kullanılmaktadır. İnsansı robotlar temel olarak iki grupta incelenmektedir. Bunlardan birincisi, iki ayaklı insansı robotlardır. Diğeri ise gezgin insansı robotlardır. İki ayaklı insansı robotlar; bacaklar, kollar, gövde ve baş kısımlarından oluşmaktadır. Bu robotlarda genellikle yürüme, koşma gibi insan hareketlerini taklit etme üzerine araştırmalar yapılmaktadır. Gezgin robotlar ise üst gövde kısmı insana benzeyen alt kısmı ise tekerlekli ya da paletli olan robotlardır. Gezgin robotlar genellikle mobil işlemler ve servis yapma gibi işlerde kullanılmaktadır.

İki ayaklı insansı robotlar insanlara benzemeleri ve onlar gibi gerçek ortamda hareket edebilmeleri yönüyle diğer robotlara kıyasla ön plana çıkmaktadırlar. İki ayaklı insansı robotların tekerlekli ve paletli robotlara göre gerçek ortamlarda hareket kabiliyetlerinin fazla olması en temel avantajları olarak gösterilmektedir. Engellerin üzerinden geçme, merdiven çıkabilme gibi hareketleri yapmaları tekerlekli ve paletli robotlara göre onları daha cazip hale getirmektedir.

İki ayaklı insansı robotların, geleneksel tekerlekli ve paletli robotlara göre bazı dezavantajlı olduğu durumlarda bulunmaktadır. Bunların en başında hareket ve denge

problemleri yer almaktadır. İki ayaklı insansı robotların dengede kalabilmesi zorlu bir problemdir. Bu robotlar esnek bir vücut yapısına sahip olmadığı için dengelerini sağlamakta problem yaşamaktadırlar. Özellikle beklenmeyen itmelerde ve engebeli arazilerde yürürken dış kuvvetlere karşı kararsızdırlar. Buna karşın, geleneksel tekerlekli robotlar insansı robotlara göre daha kararlı ve dengelidir [3].

İnsansı robot çalışmalarının ana hedefi gerçek ortamlarda çalışabilen robotlar geliştirmektir. Bundan dolayı robotlarda insansı hareketleri geliştirmek için dış kuvvetlerden kurtarma üzerine çalışmalar yapılmaktadır. Bu robotların kendi denge problemlerinin yanı sıra aldıkları darbeler, zeminden kaynaklı düzensizlikler de dış itme olarak nitelendirilmektedir. İnsansı robotlar dış itmelerden çabuk etkilenip, dengelerini kaybedip düşebilmektedirler. Bu nedenle insansı robotlarda itme geri kazanımı kontrolü çok önemlidir. Literatürde bu problem iki bacaklı yürüme (bipedal locomotion) problemi olarak adlandırılmıştır. İnsansı robotlarda yürüme problemini çözmek için birçok yöntem önerilmiştir. Önerilen bu yöntemlerden bazıları geleneksel kontrol sistemleri, bazıları bulanık kontrol sistemleri, bazıları da YSA yöntemlerinden oluşmaktadır.

İnsansı robotların, gerçek ortamlarda başarılı bir yürüyüş yapabilmeleri için aktif ve hızlı bir şekilde karar vermeleri gerekmektedir. Bu robotlar iki temel sebepten dolayı yürüyüşlerinde başarısız olmaktadır. Birincisi, zeminde bulunan engel ya da yokuşlar, ikincisi ise dışarıdan gelen itme kuvvetleridir. Bu sorunları ortadan kaldırmak için robotun denge pozisyonu, sensörlerden gelen veriler ile çok hızlı bir şekilde hesaplanmalıdır. Hesaplanan sonuçlar aktüatörlere iletilerek robotun dengesinin sağlanması gerekmektedir.

İki ayaklı insansı robotların modellenmesinde genel olarak temel bazı zorluklar bulunmaktadır. Bunlardan biri yüksek serbestlik derecelerine sahip sistemler olmalarıdır. Serbestlik derecesi arttıkça kontrol edilmesi gereken aktüatör sayısı artmaktadır. Bir diğer temel zorluk ise değişken yapıda olan robot kinematiğidir. İnsansı robotun hareketine bağlı olarak kinematiği sürekli değişir. Robot tek destek ve çift destek noktasında olduğu zaman denge noktası değişmektedir. Robotun destek noktası tespit edilip ona uygun bir denge pozisyonuna geçmesi sağlanmalıdır. Bir diğer

zorluk ise dış kuvvetlerdir. Bu zorluk robotun kendisinden kaynaklı olmadığı için ne zaman ve nereden geldiği belli olmadığından kontrol etmesi daha zordur.

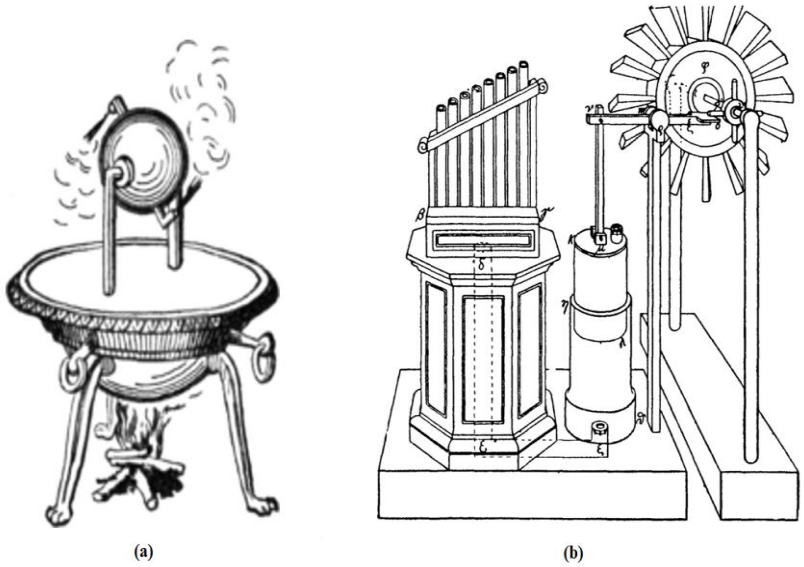
İki ayaklı insansı robotlarda dengeli bir şekilde yürüme alt ve üst vücut hareketleri ile sağlanmaktadır. Alt vücut hareketlerinde robotların dengeli bir yürüyüş ve koşma sorunları incelenir. Üst vücut hareketleri ise insansı robotların kol ve vücut hareketlerini kontrol ederek daha dengeli ve çevreye daha duyarlı olmasını sağlar. Bu metotlar ile insansı robotlar düşmeden kurtarılabilir. Alt vücut hareketlerinin içerdiği itme-kurtarma kontrol stratejileri; ayak bileği stratejisi, kalça stratejisi ve adım stratejisinden oluşur. Bu stratejiler gerçek hayatta insanların denge bozukluklarında ayakta durabilmek için verdiği tepkilerin matematiksel ifadesidir. Ayak bileği stratejisinde, ayak bileği ekleminin tork değeri kontrol edilerek denge sağlamaktadır. Kalça stratejisinde, gövdenin açısal ivmesi kullanılarak vücut dengede tutulmaktadır. Adım stratejisinde ise, destek noktasının bozulma yönünde konumunu değiştirerek vücudu dengelemektedir.

Düzgün bir yürüyüş için bu stratejiler teorik olarak kullanılsa da insansı robotlar üzerinde fiziksel olarak uygulaması çok kolay değildir. Bu üç strateji kullanılarak robotların nasıl hareket ettiği konusunda literatürde az çalışma bulunmaktadır. İnsansı robotlarda itme geri kazanım kontrolü maliyetini en aza indirmek için makine öğrenmesi yaklaşımı kullanılmalıdır.

1.1 Robotların Tarihsel Gelişimi

Robot fikri insanlık tarihinde çok eskiye dayanmaktadır. Günümüzdeki elektrik enerjisi ile çalışan robotlar gibi olmasa da M.Ö. yapılan mekanik otomat robotlara rastlamak mümkündür.

M.Ö. 10 - M.S. 70 yılları arasında yaşadığı düşünülen İskenderiyeli Hero otomat çalışmaları ile bilinmektedir. İskenderiyeli Hero; otomatik kapı açma düzeneği, rüzgâr enerjisiyle çalışan makinalar ve su otomatları geliştirmiştir [48]. Şekil 1.1'de İskenderiyeli Hero'nun buhar motoru ve rüzgâr enerjisi ile çalışan otomatı verilmiştir.



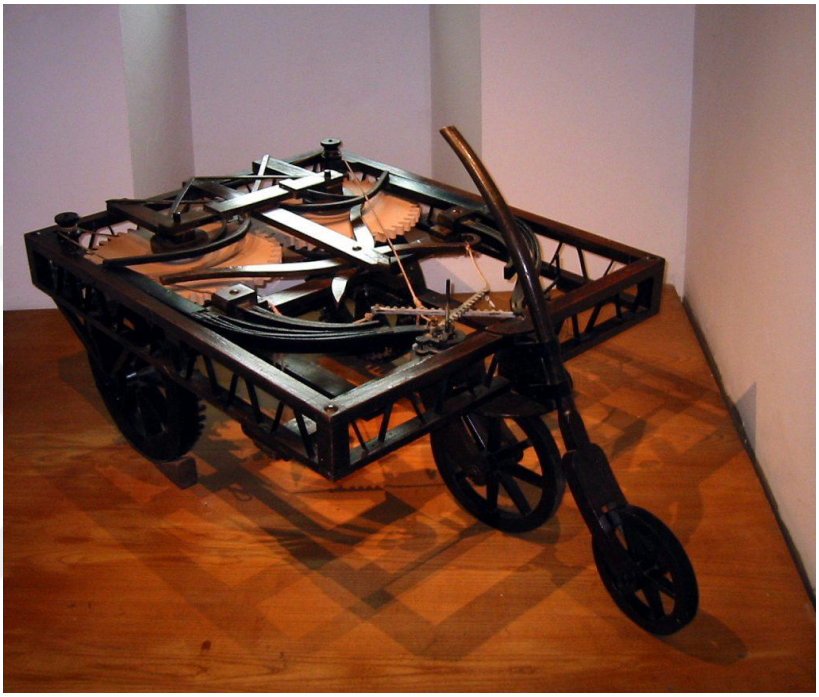
Şekil 1.1 a) Hero'nun buhar motoru b) Hero'nun rüzgar enerjisiyle çalışan otomatı[48]

Robotiğin babası olarak kabul edilen aslen Cizreli olup Diyarbakır'da yaşamını sürdüren El-Cezerî otomatlar ile alakalı "Kitab-ül Hiyel" isimli eseri yazmıştır. Bu eserinde icat ettiği birçok robot düzeneğini anlatmıştır. El-Cezerî yapmış olduğu sistemlerde devrinin çok ötesinde mühendislik işleri gerçekleştirmiştir [49]. Şekil 1.2'de El-Cezerî'nin filli su saati gösterilmiştir.



Şekil 1.2 El-Cezeri'nin filli su saati [49]

El-Cezerî'nin çalışmalarından etkilenen Leonardo da Vinci çeşitli mekanik robotlar üzerine çalışmalar gerçekleştirmiştir. Leonardo da Vinci "Carro Semovente" adını verdiği sürücüsüz araç projesini çizimlerinde göstermiştir. Şekil 1.3'de Leonardo da Vinci'nin sürücüsüz arabası gösterilmiştir. Ayrıca helikopter, robot şövalye gibi robotlar üzerinde çalışmalar gerçekleştirmiştir [50].



Şekil 1.3 Leonardo da Vinci'nin sürücüsüz arabası [50]

El-Cezerî'nin yazmış olduğu eserler 19. ve 20. yy. da birçok ülkede ders kitabı olarak kullanılmıştır. 20. yy. da robotik teori, fikir olmaktan çıkıp elektrik ile çalışan robotlar geliştirilmeye başlanmıştır. Bu robotlar sabit ve yalnızca kendi eksenlerinde hareket eden makinalardır.

1930'larda havacılık sektöründe önemli bir dönüm noktası olan otomatik pilot sistemi tasarlanmıştır. Programlanabilir püskürtmeli boyama makinası geliştirilmiştir. 1940'lı yıllarda ilk radar teknolojisi geliştirilmiştir. Aynı yıllarda ilk elektronik bilgisayar olan ENIAC geliştirilmiştir. İlerleyen yıllarda uzaktan kumanda edilebilen robot kolu tasarlanmıştır. Programlanabilir robotlar geliştirilmiştir. 1960'lı yıllara gelindiğinde ticari amaçlı robotlar geliştirilmiş ve pazarlanmıştır. Bilgisayar destekli altı eklemli yapay kol üretilmiştir. 1970'li yıllarda robot üretimi ve pazarlaması daha da yaygınlaşmıştır. 1980'li yıllarda Honda firması Asimo projesini duyurmuştur. Şekil

1.4’de Honda firmasının geliştirmiş olduğu insansı robot Asimo gösterilmiştir. Robotların sanayide etkin bir şekilde kullanılması robotların akademik anlamda çalışmalarına ilham olmuştur. 1990’lı yıllarda yapay zekâ ile robot çalışmaları hız kazanmıştır. 2000’li yıllarda insansı robotlar üzerine çalışmalar artmıştır. Akademik olarak insan hareketlerini taklit eden robotlar üzerine çalışmalar artarak devam etmektedir [34, 52].

Teknolojinin gelişmesiyle birlikte sanayi, tıp, psikoloji, endüstri gibi alanlarda robotlar karşımıza daha çok çıkmaya başlamıştır. Son yıllarda özellikle insan hareketlerini taklit eden ve insan gibi düşünen robotlar üzerine çalışmalar gerçekleştirilmektedir. Günümüzdeki çalışmalarda yapay zekâ, bulanık mantık ve derin pekiştirmeli öğrenme yöntemleri tercih edilmektedir.



Şekil 1.4 Honda tarafından geliştirilen insansı robot Asimo [52]

1.2 Literatür Araştırması

İnsansı robotların denge ve itme-kurtarma problemleri üzerine son yıllarda birçok araştırma gerçekleştirilmiştir. İnsansı robotların dengede kalması ile ilgili literatür

çalışmaları incelendiğinde çok çeşitli çalışmalar gözümüze çarpmaktadır. Bu araştırmalarda birçok farklı kontrol yöntemi kullanılmıştır. Literatüre bakıldığında klasik kontrol yöntemleri, tahmine dayalı kontrol yöntemleri, yapay zekâ, bulanık mantık, derin öğrenme, akıllı kontrol yöntemleri gibi birçok farklı yöntem kullanılmış ve kullanılmaya devam etmekte olduğu görülmektedir. Bu çalışmalardan bazıları aşağıda kısaca ifade edilmiştir.

Huang ve arkadaşları, insansı robotlarda oluşan dengesizliği azaltmak üzerine çalışmışlardır. İnsanlar, dışardan gelen itmeye ters bir kuvvetle cevap verip dengeyi korumaktadırlar. Bu stratejiden faydalanarak insansı robotlar için dışardan gelen itmenin sonucu denge noktasından uzaklaşma hareketinin neden olduğu dengesizliği azaltan dirençli bir kontrol sistemi önermişlerdir. Bu sistem için sanal kütle modeli önerilmiştir. Önerilen model ile beklenmedik dış kuvvetlerden ve eğimli zeminden kaynaklı denge bozukluklarına karşı bir çözüm sunulmuştur [30].

Bir başka çalışmada Li ve arkadaşları, bir insansı robotun doğrusal ters sarkaç modelini oluşturmak için Bulanık Derin Q Ağı (BDQA) algoritması kullanmışlardır. Oluşturdukları kontrolör ile gerçek zamanlı bir yürüyüş modeli tasarlamışlardır. BDQA, insansı robotun yürüyüş modelini düzeltmesi ile birlikte denge problemini de çözmektedir. Yapılan deneylerde robotun, BDQA ile yürüyüş sırasında zamanında geri bildirimleri aldığını ve yürüyüşünü bağımsız dengeli olarak yaptığı tespit edilmiştir [29].

2021 yılında Yang ve arkadaşlarının yapmış olduğu çalışmalarında insansı robotlarda değişken yükseklikteki ters sarkaç modeline dayalı itme kurtarma problemine odaklanmışlardır. Hamilton-Jacobi erişilebilirlik analizine dayalı bir yöntem ile değişken ters sarkacın sıfır adım noktası belirlenmiştir. Robotun destek noktasını yakalayamayacağı durumlar için dayanak adım konumunun belirlenmesini sağlayan bir yöntem tasarlamışlardır. Uygulanan stratejinin konum kontrollü insansı robotlarda kullanılması için, ilişkili diferansiyel ters kinematik tabanlı kontrolör kullanmışlardır. Geliştirilen kontrolör sayesinde klasik ters sarkaç modeline kıyasla denge için daha iyi sonuçlar alındığı gözlemlenmiştir [26].

Schuller ve arkadaşları 2021 yılında yaptıkları çalışmada sistemin dönme dinamiklerinden faydalanarak insansı robotlar için itme kurtarma algoritması sunmuşlardır. Önerdikleri yöntemde insansı robot gelen dış itmeyi gidermek ve dengelemek için, itmenin büyüklüğüne ve yönüne dayalı olarak merkezcil momentum üretmektedir. Merkezcil momenti arttırıp azaltarak robotun denge konumuna gelmesini sağlamışlardır [27].

Messesan ve arkadaşları, adım stratejisi ve ayak bileği stratejisini kullanarak denge bozulmalarına karşı bir kontrolör oluşturmuşlardır. Anlık olarak dengeyi sağlamak için hem tek destek hem de çift destek aşamalarında algoritma aktif olarak çalışmaktadır. Yaptıkları kontrolör ile robotun denge problemine çözüm geliştirmişlerdir [28].

Chiang ve Wang, insansı robotların engebeli ve yokuşlu zeminlerde dengeli yürüyüş için bir duruş kontrol sistemi tasarlamışlardır. İvmeölçer ve jiroskoptan aldıkları verileri tamamlayıcı filtre kullanarak birleştirmişlerdir. Filtrelenen bu veriler kullanarak bulanık kontrolör aracılığı ile robotun denge problemine çözüm bulmuşlardır [25].

Melo ve arkadaşları, mevcut yürüme motorlarına Sıfır Moment Noktası kavramına dayanan kontrol teorisi ile insansı robotların itmeye karşı dengede kalmasını sağlayan bir kontrol sistemi geliştirmişlerdir. Çalışmalarında derin sinir ağı algoritmalarından Yaklaşık Politika Optimizasyonu (PPO) algoritmasını kullanmışlardır [23].

Kim ve Seo 2019 yılında yapmış oldukları çalışmalarında Lineer Ters Sarkaç Modeli kullanarak analizlerini gerçekleştirmişlerdir. Yüksek seviyeli itme-kurtarma stratejilerini derin pekiştirmeli öğrenme ile Derin Q Ağı (DQA) algoritmasında eğitimlerini uygulamışlardır [17].

Hosseinmemar ve arkadaşları, zayıf servo motor kullanan insansı motorların itmelere karşı dengede kalmasını sağlamak için ivmeölçer ve jiroskoptan aldığı verilerle kapalı döngü geribildirim kontrol yöntemini kullanmışlardır. İtmelerden kurtulmak için üç farklı deney gerçekleştirmişlerdir. Bunlar; yalnızca jiroskop, yalnızca ivmeölçer ve her

iki sensörün birleşimi kullanılarak yapılmıştır. Sonuçlara göre iki sensör birleşimi ile oluşturulan dengeleme yöntemi daha iyi bir performans göstermiştir [21].

Maulana arkadaşları ile birlikte, insansı robotlarda dengeleme sisteminde kullanılan eğim sensörü bilgilerinden alınan veriler üzerindeki gürültüyü azaltma yeteneğine sahip bir sistem tasarlamışlardır. Jiroskop ve ivmeölçerden alınan veriler tamamlayıcı filtreden geçirilerek eğim açısı tespit edilmiştir. Robot, tamamlayıcı filtrenin çıktısına bağlı olarak dengeleme eylemini gerçekleştirmiştir. Tespit edilen eğim açısı düşmesine neden olacak konumda olduğunda ana işlemciye haber verir ve servo motorlar denge konumuna geçecek şekilde ayarlanır [24].

Yang, Li ve Komura yapmış oldukları çalışmalarında, itme gibi denge bozukluklarını kontrol politikalarını kullanarak derin pekiştirmeli öğrenme yoluyla insansı robotların denge problemlerine bir çözüm sunmuşlardır [20].

Ashtiani arkadaşları ile ayak bileği ve kalça stratejisinin birleşimini kullanılmışlardır. Basınç Merkezi ve Merkezi Moment Ekeni stratejileri tork kontrollü insansı robotlar için kullanılmaktadır. Fakat geleneksel insansı robotlar konum kontrollüdür. Bu çalışmada konum kontrollü insansı robota tork sensörlerine ihtiyaç duyulmadan itme kurtarma stratejilerini kullanarak bir geri bildirim kontrol sistem tasarlayıp dengeyi sağlamışlardır [7].

Ghassemi ve arkadaşları kararsız yapıda olan insansı robotların itme-kurtarma problemi üzerinde çalışmıştır. Konum kontrolü olan bir robota tork kontrolü yaklaşımları uygulanmıştır. Robotu kontrol etmek için PD kontrolörü kullanılmıştır [22].

Yi çalışmasında, iki ayaklı insansı robotların yürüyüşlerinde, dengede kalmak için sağlam geri beslemeli kontrol yöntemi kullanmışlardır. Çeşitli sensör ve motorlardan alınan veriler Derin Pekiştirmeli Öğrenme (DPÖ) algoritmaları kullanılarak robotun kararlılığı optimize edilmiştir [18].

Stephens 2007 yılında, insansı robotlarda itme kurtarma stratejileri üzerine çalışmıştır. İtmelerden kurtarmak için üç farklı itme kurtarma stratejisi sunmuştur. İnsansı robotların; ayak bileklerini, kalça eklemlerini ve destek adım stratejilerini kullanarak dengede kalacaklarını göstermiştir [1].

1.3 Tez Amacı

Elektronik ve yazılım teknolojilerindeki gelişmeler sonucunda insanların günlük yaşamını kolaylaştıran en uygun mekanizmalar olarak robotlar daha işlevsel hale gelmiştir. Hayatı kolaylaştırmak için insansı robot tasarımlarına ilgi her geçen gün artmaktadır. Fakat insansı robotlarda karşılaşılan bazı problemler vardır. Bu problemlerden en önemlisi dengedir. İnsansı robotların dengesini koruması zorlu bir problem olarak karşımıza çıkmaktadır. Bir insansı robot yapısal olarak dengesizdir, özellikle öngörülemeyen çarpışmaların varlığında ve engebeli arazide yürürken dış bozulmalara karşı kararsızdır.

Bu çalışmada insansı robotlarda dış tepkilere karşı itme geri kazanımı kontrolü olan bir sistem önerilmiştir. Bu kontrolü gerçekleştirmek için yüksek seviyeli itme kurtarma kontrolörü ve düşük seviyeli itme kurtarma kontrolörü kullanılmaktadır. Düşük seviyeli itme kurtarma kontrolörleri, ayak bileği stratejisi, kalça stratejisi ve adım stratejisinden oluşur. Bu stratejiler, insanların denge bozukluğu ve tedirginlik durumlarında denge kontrolünü sağlayıp bununla nasıl başa çıktıklarının biyomekanik bir analizidir. Yüksek seviyeli itme kurtarma kontrolörleri sensörlerden aldıkları verileri kullanarak hangi düşük seviyeli itme kurtarma stratejilerinin kullanılacağına karar vermektedir. Ana işlemci, alınan sensör verilerinden açısal hız ve doğrusal ivme verisini ölçerek tek bir modülde toplar ve bu verileri yüksek seviyeli itme kurtarma kontrolörüne iletir.

İnsansı robotların dengesini sağlaması yapacağı üst düzey hareketlere, yürüyüş biçimine, yüzeyin eğimine, çevredeki engellere karşı göstereceği tepkilerden dolayı karmaşık ve zor bir görevdir. Bu araştırmada; insansı robotların ayakta dururken ve yürürken dengede durması ve dış kuvvetler sebebiyle oluşacak denge bozukluklarını önlemesini sağlayan aktif dengeleme ve itmeden kurtarma sorunlarına odaklanması amaçlanmıştır.

Bu tez çalışmasında, hem gerçek dünya testleri hem de simülâsyon testleri yapılmıştır. Çalışmanın simülâsyon testleri Webots ortamında 3 boyutlu model olarak gerçekleştirilmiştir. Gerçek dünya testlerinde ise Robotis-OP2 insansı robotu üzerinde yapılmıştır. Robotis-OP2 modern endüstriyel robotlara göre daha ucuz servo motorlardan üretilmiştir. Bu yüzden dengede durması ve itme geri kazınım özellikleri azdır. Testlerde kullanılan insansı robota farklı güçlerde itmeler uygulanarak dengesini bulması için kapalı döngü kontrol uygulanması yapılmıştır. Bu araştırmanın amacı, bir insanın eylemlerini taklit edebilecek tamamen özerk bir kapalı döngü itme kurtarma kontrol sistemi tasarlamak ve uygulamaktır. Harici bir itme uygulandıktan sonra sistemin dengesini bulup yörüngesine girmesi sağlanmaktadır. İnsansı robotu kontrol etmek için PD ve Model Öngörülü Kontrol (MÖK) sistemleri kullanılmıştır. Ayrıca insansı robotu tamamen otonom hale getirmek için de Derin Pekiştirmeli Öğrenme (DPÖ) algoritmaları uygulanmıştır. DPÖ yöntemlerinden farklı algoritmalar kullanılmış ve sonuçlar karşılaştırılarak en iyi sonucu veren algoritmanın bulunması amaçlanmıştır.

1.4 Tezin İçeriği

Tezin birinci bölümünde robotik çalışmaların tarihçesinden ve insansı robotlar üzerine literatürde yapılmış çalışmalardan bahsedilmiştir. Ayrıca tezin amacı ortaya konmuştur.

İkinci bölümde; insansı robotlardan ve genel özelliklerinden bahsedilmiştir. Üçüncü bölümde; robotik simülâtörler hakkında bilgi verilmiştir. En çok kullanılan robotik simülâtörlere değinilmiştir.

Dördüncü bölümde; insansı robotlarda yürüme modeli anlatılmıştır. Robota dışardan gelecek tepkiler sonucunda denge pozisyonuna gelebilmesi için kullandığı itme-kurtarma kontrolörleri açıklanmıştır. Beşinci bölümde klasik kontrol ve yapay sinir ağı kontrol yöntemlerinden bahsedilmiştir.

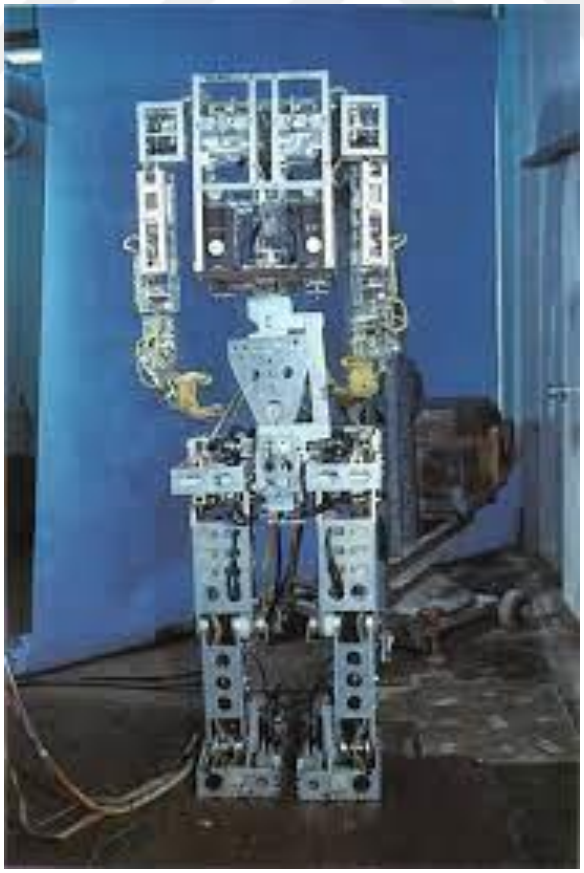
Altıncı bölümde tezin uygulama aşamasına yer verilmiştir. Yedinci bölümde bulgular ve tartışma kısmı yer almaktadır.

Son bölümde ise yapılan çalışmanın literatüre olan katkısı sunulmuş ve sonraki çalışmalarda yapılabileceklerden bahsedilmiştir.

2. İKİ AYAKLI İNSANSI ROBOTLAR

Bilim dünyasında iki ayaklı insansı robotlar sürekli dile getirilse de ilk insansı robot yapma girişimleri 1960'lı yılların sonlarına denk gelmektedir. Son yıllarda robotların gelişmesi ile insan-robot arasındaki etkileşimler hızla artmıştır. İnsansı robotların çeşitli insan faaliyetlerini yapabilmeleri bilim insanlarının bu konuya olan ilgisini arttırmıştır. Üniversiteler, devlet kurumları ve özel sektörden birçok araştırmacı-yatırımcı insansı robotlar üzerine çalışmalar sürdürmektedir. Bu robotlar savunma, mühendislik, tıp, eğlence gibi birçok alanda kullanılmaktadır [4, 45, 46, 47].

İnsansı robotlar ile ilgili dengede kalma, düşünebilme, karar verme, konuşma gibi özelliklerinin geliştirilmesi üzerine daha çok araştırma yapılmaktadır. İki ayaklı olarak tasarlanan insansı robotlar insanlar gibi engebeli zeminler ve yokuşlarda yürüyebilmektedir. Boston Dynamics, Honda, Sony, Robotis, Aldebaran Robotics gibi firmalar insansı robotlar üzerine AR-GE faaliyetleri yürütmektedirler.



Şekil 2.1 Waseda Üniversitesi tarafından geliştirilmiş olan Wabot-1[4]

İki bacak ile yürüme çalışmalarında ilk başarılı çalışma Waseda Üniversitesi'nde robotik araştırmacılar tarafından gerçekleştirilmiştir [4]. İlk tam ölçekli insansı robot ise 1973 yılında tamamlanmıştır. Wabot-1 olarak adlandırılan bu robot yürüme ve yön değiştirebilme kabiliyetlerine sahip olarak geliştirilmiştir. Şekil 2.1'de Wabot-1 isimli insansı robot gösterilmiştir.

Ülkemizde yapılan insansı robot çalışmalarında, Sabancı Üniversitesi tarafından yürütülen ve TÜBİTAK tarafından desteklenen proje ile Suralp insansı robotu geliştirilmiştir. Suralp insansı robot projesi 2009 yılında başarı ile tamamlanmıştır. Şekil 2.2'de Suralp insansı robot verilmiştir.



Şekil 2.2 Sabancı Üniversitesi tarafından geliştirilen Suralp [45]

Suralp; bacaklar, kollar, boyun, baş ve gövde olarak 29 serbestlik derecesine sahip olarak tasarlanmıştır. Robot her bir bacağında 6 her bir kolunda 7, kafasında 2, kalçasında 1 olmak üzere toplam 29 ekleme sahiptir. Düz ve eğimli yollarda yürüyebilme, yük taşıyabilme ve görüntü algılama gibi özelliklere sahip olarak gerçekleştirilmiştir. Suralp insansı robotu 1.64 m uzunluğunda ve 101 kg ağırlığında üretilmiştir [45].

Boston Dynamics tarafından üretilen iki ayaklı insansı robot Atlas 2013 yılında kamuoyuna açıklanmıştır. Amerika Birleşik Devletleri'ne bağlı Savunma İleri Araştırma Projeleri Ajansı (DARPA) finansman desteği ve gözetiminde geliştirilmiştir. 1,5 m uzunluğunda 89 kg ağırlığında 1,5 m/s hıza sahip olarak üretilmektedir. Atlas insansı robotu arama ve kurtarma faaliyetlerinde kullanılmak için tasarlanmıştır. 11 kg yük taşıma kabiliyeti olan 28 serbestlik derecesine sahip gelişmiş bir robot teknolojisi olarak programlanmıştır. Atlas üzerinde her kol ve bacak için 6, geri eklemler için 3 ve boyun ekleminde 1 tane olmak üzere 28 hidrolik aktüatör bulunmaktadır [51]. Şekil 2.3'te Atlas isimli insansı robot gösterilmiştir.

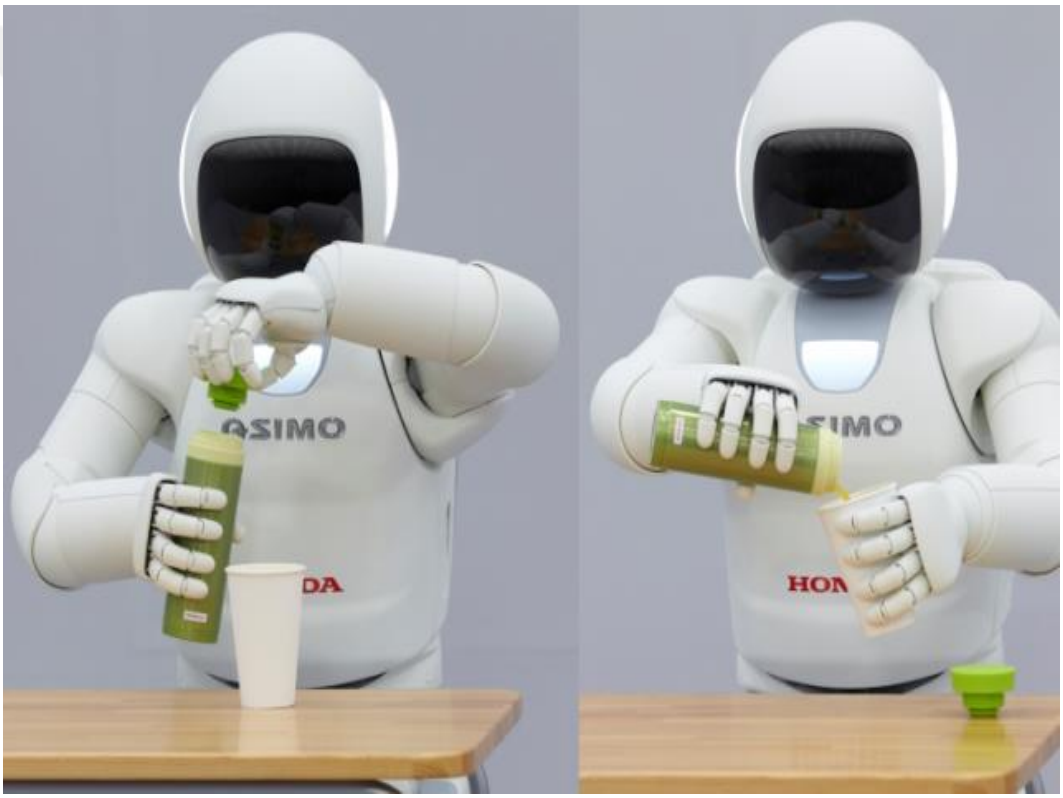


Şekil 2.3 Boston Dynamics tarafından geliştirilmiş olan Atlas [51]

Bu robot, engebeli arazi koşullarında yürürken, merdiven çıkarken ve dışardan gelen tepkilere cevap verirken dengede kalmak için güçlü bir algoritma ile üretilmiştir. Üzerindeki güçlü hidrolik aktüatörler ile diğer birçok robotun yapamayacağı üstün yeteneklere sahiptir. Boston Dynamics firması Atlası ticari olarak satmamaktadır.

Firma bu projeyi kendi bünyesinde geliştirdiği bir araştırma platformu olarak tanımlamaktadır. Günümüzde tasarlanmış ve kamuoyuna açıklanmış en gelişmiş iki ayaklı insansı robot olarak Atlas gösterilmektedir.

Honda Motor 1986 yılından beri insansı robot çalışmaları yürütmektedir. Firmanın üretmiş olduğu 11. robot serisi Asimo olarak adlandırılmıştır. Asimo, 2000 yılında iki ayak üzerinde hareket eden robot olarak kamuoyuna tanıtılmıştır. Yıllar içerisinde geliştirilerek değişik yeni modelleri üretilmiştir. En güncel model 2011 yılında duyurulan 48 kg ağırlığa 130 cm uzunluğa sahip modelidir [52].



Şekil 2.4 Honda Motor tarafından geliştirilmiş olan Asimo [52]

Yapay zekâ teknolojisini başarılı bir şekilde kullanan Asimo yüz tanıma, çevre tanıma, sesleri ayırt etme, hareketli nesneleri algılama, engellerden geçme gibi insan hareketlerini taklit edebilmektedir. 2000 yılında ilk tanıtıldığı modelinde 26 serbestlik derecesine sahipken 2011 modelinde 57 serbestlik derecesine sahiptir. Bacaklar için 6x2, kollar için 7x2, eller için 13x2, kafa için 3, kalça için 2 serbestlik derecesi bulunmaktadır. Ellerine eklenen eklemler sayesinde insanların el hareketlerini

kolaylıkla taklit edebilmektedir. Asimo yıllık olarak kiralanabilmektedir [52]. Şekil 2.4'de Asimo isimli insansı robot gösterilmiştir.

Robotis firması tarafından geliştirilmiş olan Thormang3 araştırma ve eğitim faaliyetleri için kullanılan bir insansı robottur. Gelişmiş hesap gücü, sofistike sensörler, dinamik hareket kabiliyeti ve yüksek yük taşıma yetenekleri ile ön plana çıkmaktadır. 42 kg ağırlığa, 137,5 cm uzunluğa ve 29 serbestlik derecesine sahiptir. Linux işletim sistemi üzerinde ROS platformunda çalışmakta ve C/C++ dilleri ile programlanabilmektedir. Şekil 2.5'te Thormang3 insansı robotu verilmiştir.



Şekil 2.5 Robotis firması tarafından üretilen Thormang3

Qrio, Sony tarafından eğlence sektörü için geliştirilen 2003 yılında duyurulan iki ayaklı insansı robot platformudur. Dans etme, engellerin üzerinden geçme, zıplana, koşma, konuşma gibi özellikleriyle Qrio ön plana çıkmaktadır. Şekil 2.6'de Qrio gösterilmiştir. 6,5 kg ağırlığı ve 58 cm boyu ile küçük bir insansı robottur. 38 serbestlik derecesine sahip, 3 eksenli gyroscope, 2 eksenli accelerometer, kamera, mikrofon ve her ayakta dört kuvvet sensörü bulunmaktadır [53].



Şekil 2.6 Sony firması tarafından üretilen Qrio

Aldebaran Robotics firması tarafından 2004 yılında orta ölçekli programlanabilir insansı robot olarak Nao geliştirilmiştir. Daha sonra Aldebaran Robotics ismi SoftBank Robotics olarak değiştirilmiştir. Akademik araştırmalar ve eğitimler için sıkça kullanılan bir insansı robot platformudur.



Şekil 2.7 SoftBank Robotics firması tarafından üretilen Nao

Dünya çapında 600'den fazla üniversite ve araştırma merkezi Nao insansı robotu kullanmaktadır. Nao insansı robotun görüntüsü Şekil 2.7'de verilmiştir. 5,5 kg ağırlık, 58 cm uzunluğa ve 25 adet fırçasız DC motora sahiptir. Türkçe dâhil 20 dilde konuşma tanıma ve diyalog içermektedir. Nesne tanıma, konuşma, hareket algılama, yürüme, engelleri tanıma gibi birçok fonksiyonu barındırmaktadır. Açık kaynak kodlu ve istenilen işler için programlanabilir bir yapıya sahiptir. Python, C, C++, Urbi ve .Net programlama dilleri kullanılarak Nao programlanabilmektedir.

Robotis-OP2, Robotis firması tarafından geliştirilmiş olan eğitim ve araştırma amaçlı insansı robottur. Robotis-OP2 insanların fiziksel yapısına göre tasarlanmış, gelişmiş hesaplama kabiliyetine sahip ve açık kaynak kodlu bir robot ortamıdır. Bu robot kafa, gövde, kollar ve bacaklar olarak üretilmiştir.



Şekil 2.8 Robotis firması tarafından üretilen Robotis-OP2

Robot üzerinde kafada 2, her kolda 3 ve her bacakta 6 olmak üzere toplam 20 serbestlik derecesi (DOF) bulunmaktadır. Robotis-OP2 platformu içerisinde 20 motor, 20 pozisyon sensörü, 3 eksenli gyroscope, 3 eksenli accelerometer, kamera, hoparlör, usb

giriş, mini HDMI, ethernet girişi ve ledler yer almaktadır. Her bir eklem için Dynamixel MX-28T servo motorlar kullanılmıştır. Intel Atom N2600@1.6 GHz çift çekirdekli işlemci bulundurmaktadır ve 4 GB'a kadar DDR3 Ram desteği vardır. Bu robotun ayak tabanı dikdörtgen olarak tasarlanmıştır. Bu tasarım robotun daha iyi dengede durmasına yardımcı olmaktadır. Robotis-OP2 45,5 cm yüksekliğe ve 3,0 kg ağırlığa sahiptir. Robotun görüntüsü Şekil 2.8 da gösterilmektedir. Robotis-OP2 Linux tabanlı işletim sistemlerinin yanı sıra Windows tabanlı işletim sistemlerini de desteklemektedir. Birçok simülâtörde Python, Java, C, C++ gibi programlama dilleri kullanılarak programlanabilmektedir. Donanımları sayesinde birçok robotik uygulamayı gerçekleştirmeye imkân sunan eğitim ve araştırma tabanlı bir insansı robot platformudur.

Robotis firması tarafından eğitim ve araştırmalar için geliştirilmiş olan bir diğer insansı robot Robotis-OP3'tür. Robotis-OP3, Robotis-OP2'ye göre daha üstün özelliklere sahiptir. Şekil 2.9'da Robotis-OP3 gösterilmektedir.



Şekil 2.9 Robotis firması tarafından üretilen Robotis-OP3

Robotis-OP3 içerisinde kafada 2, her kolda 3 ve her bacakta 6 olmak üzere toplam 20 serbestlik derecesi bulunmaktadır. Her bir eklem için servo motorlar XM430-W350 kullanılmıştır. Ram kapasitesi 32 GB'a kadar arttırılabilmekte ve hafıza olarak SSD bellekler kullanılmaktadır. Robotis-OP3 ileri hesap yapabilme gücüne sahip işlemcileri ve gelişmiş sensörleri sayesinde AR-GE ve eğitim alanlarındaki çalışmalar için uygun bir insansı robot platformudur.

3. ROBOTİK SİMÜLATÖRLER

İnsanlar ve hayvanlarda büyüme ile birlikte öğrenmede gerçekleşmektedir. Bu canlılarda vücut ve beyin büyüdükçe buna paralel olarak emeklemek, ayakta durmak, yürümek, koşmak, zıplamak, yemek yeme gibi hareketler öğrenilmektedir. Fakat robotlarda bu şekilde ilerlememektedir. Robotlarda gelişmiş bir altyapı bulunmaktadır. Üzerlerindeki sensör ve motorlar sayesinde vücudunu verimli şekilde kullanma becerisi sonradan öğretilmektedir.

Robotlarda en önemli zorluklardan biri gerçek dünya testleridir. Robot hareketlerini yönlendiren yapay zekâ sistemleri insan hareketlerini taklit edebilmesi için çok uzun eğitimler gerektirmektedir. Robotları gerçek dünyada test ederken bir beceriyi kazandırmak için parkurlarda binlerce kez veya yıllarca test etme ihtiyacı doğabilmektedir. Bir robot gerçek dünya testi esnasında zarar gördükten sonra düzenli olarak onarılması gerekebilmektedir. Her testten sonra robotta oluşacak zararı onarmak hem maliyeti arttırmakta, hem de eğitimi yavaşlatmaktadır.

Gerçek dünya testlerine alternatif yöntem simülâsyon üzerinde işlemleri gerçekleştirmektir. Birçok bilim alanında olduğu gibi robotikte de zamandan ve maliyetten tasarruf etmek için simülâtörler kullanılmaktadır. Simülâsyon yazılımları sayesinde robota yüklenecek programların gerçek robottan bağımsız sanal ortamda test etme imkânı sunulmaktadır. Simülâsyon yazılımları, robotun 3-B sanal bir sürümü bilgisayar ortamında hızlı bir şekilde ve gerçek dünyada oluşturulacak fiziksel altyapı maliyeti olmadan eğitimi gerçekleştirilebilmektedir.

Simülâsyon ortamında geliştirilen uygulama değişiklik yapılmadan gerçek robota yüklenebilmektedir. Örneğin; Webots simülâsyon ortamında Robotis-OP2 insansı robotu üzerinde geliştirilmiş program, gerçek robota yüklenip çalıştırılabilmektedir.

Robotik simülâtörlerde, robotlar ve çalışma ortamları üç boyutlu olarak modellenebilmektedir. Bu şekilde üç boyutlu modelleme imkânı sayesinde gerçek ortama yakın bir şekilde test etme imkânı sunulmaktadır. Simülâtörlerde gerçek robot üzerinde bulunan sensör ve motorlar birebir aynı özelliklerde tasarlanarak oluşturulmaktadır. Robotik simülâtörler kullanıcı dostu uygulamalar olarak

tasarlanmaktadır. Robot programlama esnasında kolaylık olması açısından birden fazla programlama dili ile uygulama geliştirmeye olanak sağlamaktadırlar.

Robot programlamasında kullanılan simülâsyon programları seçiminde kullanılacak robotun simülâsyon tarafından desteklenmesini göz önünde bulundurulması gerekmektedir. Simülâsyon programları ile mevcut bir robotu geliştirme uygulamaları yapılacağı gibi robotu sıfırdan tasarlayıp programlama imkânı da sunmaktadır.

Robot geliştirme ya da programlama için birçok robotik simülâtör bulunmaktadır. Bu simülâtörlerden bazıları açık kaynak kodlu bazıları ise lisanslı robotik simülâtörlerdir. Açık kaynak kodlu simülâtörlere Gazebo, ARS, Breve, Moby, OpenHRP3, MORSE, SimRobot gibi yazılımlar örnek verilebilir. Lisanslı robotik simülâtörlere Webots, V-REP, Microsoft Robotics, Developer Studio, Workspace5 gibi yazılımlar örnek verilebilir.

Simülâsyon yazılımları zaman ve maliyet açısından kolaylıklar sunmasına rağmen, bunlar sadece gerçek dünyanın bir tahminidir. Bu yazılımlar gerçek dünya testlerine olan ihtiyacı tamamen ortadan kaldırmazlar. Simülâsyonlarda oluşturulamayacak ortamlar veya beklenmeyen durumlar için gerçek dünya testlerinin yapılması gerekmektedir.

3.1 V-Rep

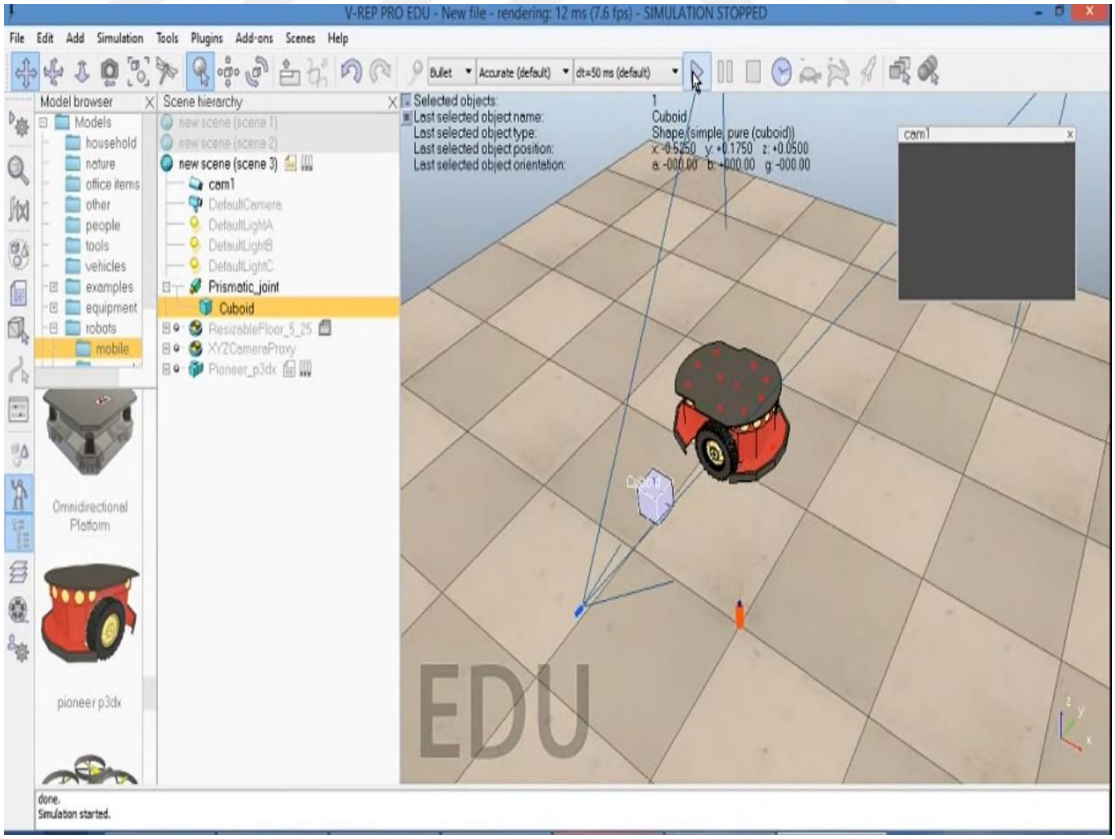
V-Rep (Virtual Robot Experimentation Platform) bir robot sistemini fiziksel ve yazılım olarak modelini gerçekleştirildiği sanal deney platformudur. V-Rep programı Coppelia Robotics firması tarafından genel amaçlı robot simülâsyonu olarak geliştirilmiş lisanslı bir üründür.

Bu simülâtör Windows, Linux ve MAC işletim sistemlerinde çalışmaktadır. V-Rep simülâtörü programcının, C/C++, Java, Python, Matlab, Lua, Urbi ve Octave gibi yedi farklı dil ile programlama yapmasına olanak sağlamaktadır. Kullanıcı dostu arayüzü sayesinde simülâtörü kolay kullanma, hızlı prototip çıkartma ve kodlanmasına olanak sağlamaktadır. V-Rep simülâtörü üç boyutlu tasarımlara izin verdiği için gerçek dünyaya benzer ortamlar oluşturulmasına izin vermektedir. V-Rep ile karada, denizde

ve havada hareket eden robotlar simüle edilebilmektedir. V-Rep'i oluşturan üç ana işlevi şöyle sıralanabilir;

- Sahne nesneleri
- Hesaplama modülleri
- Kontrol mekanizmaları

V-Rep simülâtörü içerisinde; şekiller, eklemler, mesafe sensörleri, görüntü sensörleri, kuvvet ve tork sensörleri, grafikler, kameralar, ışıklar, yollar, referans objeler, kesiciler gibi nesneler yer almaktadır. Bu sahne farklı nesneleri ve sensörleri (örneğin; ivmeölçer, jiroskop, GPS, Tork vb.) kullanılarak çeşitli işlevsel robotların modellenmesine olanak sağlamaktadır. V-Rep simülâtöründe oluşturulan çevreye sürükleyerek kolayca eklenilebilen geniş sensör ve robot model kütüphaneleri bulunmaktadır. Bu model kütüphaneleri kullanılacak projeye göre özelleştirilebilir olması V-Rep'i araştırmacılar için cazip hale getirmektedir. Şekil 3.1'de V-Rep simülâtörünün arayüz ekran görüntüsü örneği verilmiştir [47].



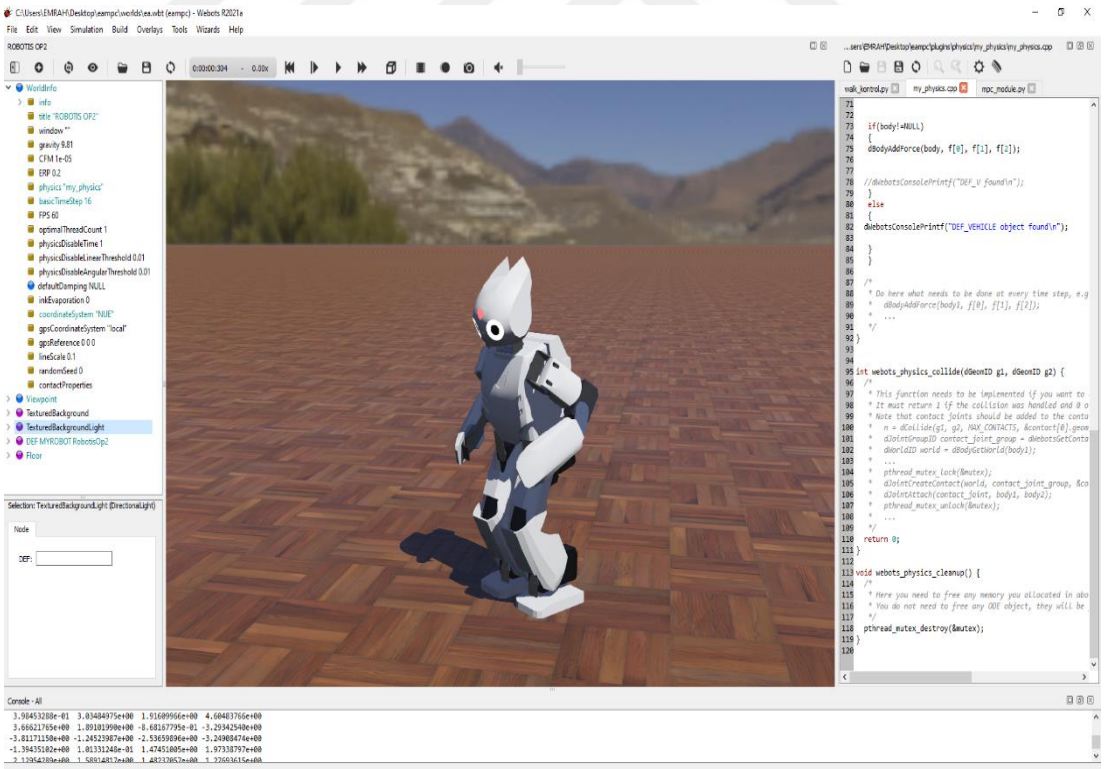
Şekil 3.1 V-Rep Simülâtör ekran görüntüsü örneği [54]

Kullanıcılar V-Rep simülâtörü ile çalıştıkları robotlar üzerinde bulunan çeşitli sensörlerden verileri okuyup, bu verileri işleyebilirler. Ayrıca yapılan uygulamaları görsel olarak sergilenme isteğine karşı simülâsyonun video kayıtları oluşturulabilmektedir.

V-Rep içerisinde bulunan hazır robot kütüphaneleri ve sensörler kullanılabileceği gibi, araştırmacı tasarım araçları sekmesinden kendi işlevsel robotunu oluşturup gerekli sensörleri robota ekleyebilmektedir. Simülâsyon esnasında robota ait veriler grafikler ile dışarıya aktarılabilmektedir. Örneğin; bir robotun bir eklemindeki tork değeri anlık olarak grafikler aracılığı ile dışarıya aktarılabilmektedir.

3.2 Webots

Webots, Cyberbotics firması tarafından geliştirilmiş ücretsiz bir robot geliştirme simülâtörüdür. Khpera Simülâtor açık kaynak kodu temel alınarak geliştirilmiştir.



Şekil 3.2 Webots Simülâtör ekran görüntüsü örneği

Robotları modellemek, programlamak ve simülek etme için kullanılan bir geliştirme ortamıdır. 3D tasarım yapmaya olanak sağlayan Webots akademik çalışmalarda

kullanılmaktadır. Nesnelerin kütle, eklemler, sürtünme, şekil, doku gibi fiziki özellikleri gerçek dünyaya yakın olarak tasarlama imkânı sunan bir simülâsyon yazılımıdır. Webots sade arayüzü sayesinde basit ve kullanışlı bir simülâtördür. Şekil 3.2 de Webots simülâtörünün arayüz ekran görüntüsü örneği verilmiştir.

İçerdiği kütüphaneler ile birçok farklı robotun programlanmasına ve geliştirilmesine olanak sağlamaktadır. Webots ortamında sensör verilerini öğrenmek ve işlemek oldukça basittir. Robot üzerinde yer alan ivmeölçer, jiroskop, kuvvet/tork sensörü, basınç/dokunma sensörü, ışık sensörü, kamera, GPS gibi sensörlerden verileri okuyup hızlı bir şekilde işleyebilmektedir. Kütüphaneleri sayesinde sensörlerden aldığı verileri işleyebilen birçok fonksiyonu içerisinde barındırmaktadır.

Webots içerisinde yer alan robotların gerçek dünyada da çalışacak şekilde programlanmasına olanak sağlamaktadır. Webots ile dört ayaklı, iki ayaklı, tekerlekli, uçan, yüzebilen ve sabit robotlar programlanıp simüle edilebilmektedir. Gerçek dünyada veriler gürültülüdür. Webots içerdiği fizik kütüphanelerindeki sensörler aracılığı ile araştırmacılara simülâsyon ortamında istenilen gürültülerin eklenmesine olanak sağlamaktadır. Örneğin; bir fizik kütüphanesi ile robota dışardan bir kuvvet uygulanabilir.

Robot kontrollerini gerçekleştirmek için C/C++, Java, Python gibi programlama dillerini kendi editörü içerisinde derleyebilmektedir. Ayrıca geliştirilen MATLAB arayüzü sayesinde kodları burada derlemeye de imkân sunmaktadır.

Webots ile aynı ortamda birden fazla robotun programlanıp simülâsyonu yapılabilmektedir. Kütüphaneleri içerisinde hazır bulunan robotların dışında araştırmacı kendi robotunu tasarlayabilmektedir. Tasarladığı bu robota ihtiyaç duyduğu tüm sensörleri entegre edebilmektedir.

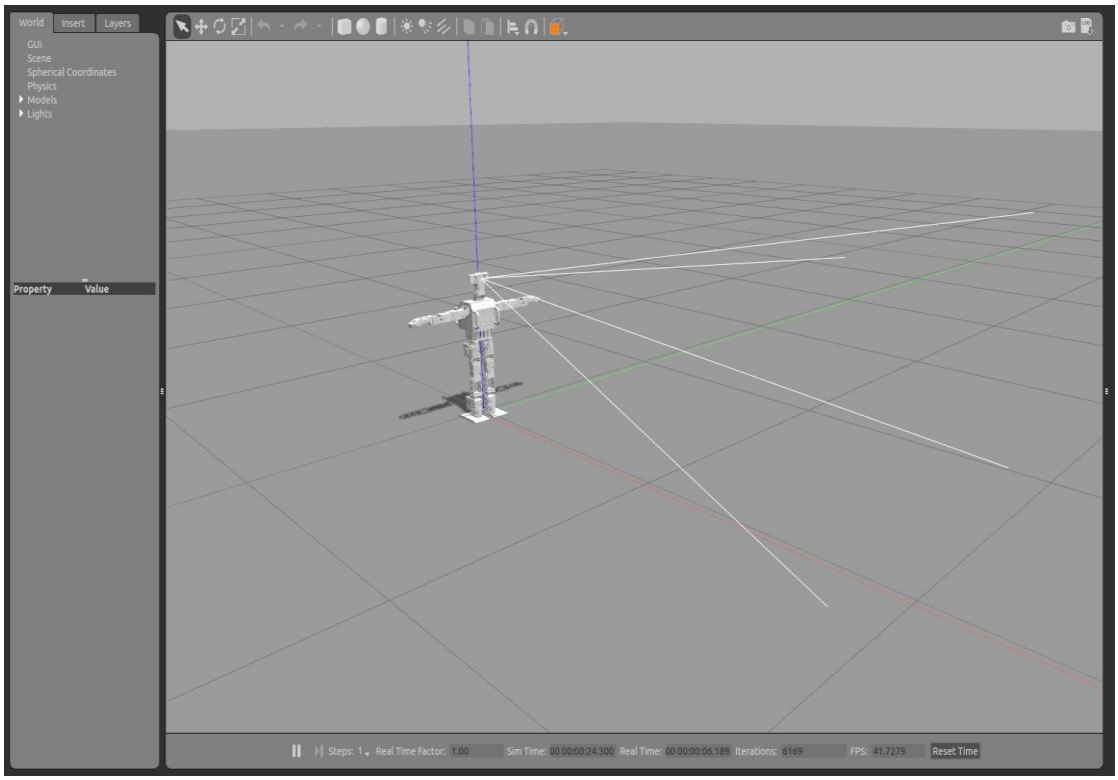
Webots'un işletim sisteminden bağımsız olarak çalışması geliştiricilere kolaylık sağlamaktadır. Uygulaması gerçekleştirilen simülâsyonun görselleştirilip veya video ile kaydı oluşturulabilmektedir. Webots içerdiği kütüphaneler, güçlü sensör ve eklenti

desteği, mimarisi, basit arayüzü, işlevselliği ve kendi içinde kodları derleyebilme özelliği ile diğer simülâsyon yazılımlarına göre oldukça kullanışlıdır.

3.3 Gazebo

Gazebo yazılımı robotik uygulamalar geliştirmeye imkân sunan üç boyutlu dinamik bir simülâsyon programıdır. Ticari bir amaç gütmeyen Open Source Robotics Foundation(OSRF) tarafından açık kaynak kodlu bir yazılım projesi olarak geliştirilmektedir. Açık kaynak kodlu olması sebebiyle Gazebo'yu farklı amaçlarla geliştirmek için çeşitli araçlar geliştiren geniş bir topluluk vardır.

Gazebo yüksek kalitede grafikler ve arayüzler kullandığından gerçek dünyaya oldukça benzer üç boyutlu tasarımlar gerçekleştirmeye olanak sunmaktadır. Yapılacak simülâsyon ortamını gerçek dünyada ihtiyacımız olan probleme göre tasarlamamız mümkündür. Gazebo ile gerçeğe yakın iç ve dış ortam tasarımları yapılabilmektedir. Gerçek dünya ortamına benzer ortamlar oluşturmak robotları daha doğru ve verimli bir şekilde simüle etmemizi sağlamaktadır. Şekil 3.3 de Gazebo simülâtörünün arayüz ekran görüntüsü örneği verilmiştir.



Şekil 3.3 Gazebo Simülâtör ekran görüntüsü örneği

Gazebo ile aynı anda birden fazla robot simüle edilebilmektedir. Gazebo bize kamera, ivmeölçer, mesafe sensörü, GPS, LİDAR gibi birçok sensör verileri ile simülâsyonu gerçekleştirebilmektedir. Gazebo çok yönlü bir simülâtördür ve önemli ölçüde değiştirilebilirlik sağlayan eklenti desteğine sahiptir.

Gazebo çeşitli robot programlarını oluşturma ve test etme açısından oldukça hızlıdır. Aynı zamanda robot tasarlamak ve yapay zekâ uygulamaları gerçekleştirmek açısından da oldukça kullanışlı bir simülâtördür.

Simülâsyon ortamlarında veriler gürültüsüz olarak hesaplanmaktadır. Fakat gerçek dünyada veriler genellikle gürültülüdür. Uygulamaları gerçek dünyada test etmeden önce simülâsyon ortamında yapılacak testlerde bu durum göz önünde bulundurularak testlerin gerçekleştirilmesi gerekmektedir. Gazebo araştırmacılara sensörler aracılığı ile fiziksel ortamda karşılaşabileceği gürültüleri ekleme imkânı sunmaktadır. Böylece simülâsyon ortamı ve gerçek dünya arasındaki farklılıkları en aza indirip gerçekçi bir simülâsyon gerçekleştirilebilmektedir.

Ayrıca Gazebo Robot İşletim Sistemi ile uyumlu çalışmaktadır. Gazebo ve RİS arasındaki bağlantı uygulama programlama arabirimi aracılığı ile sağlanmaktadır.

3.4 Robot İşletim Sistemi

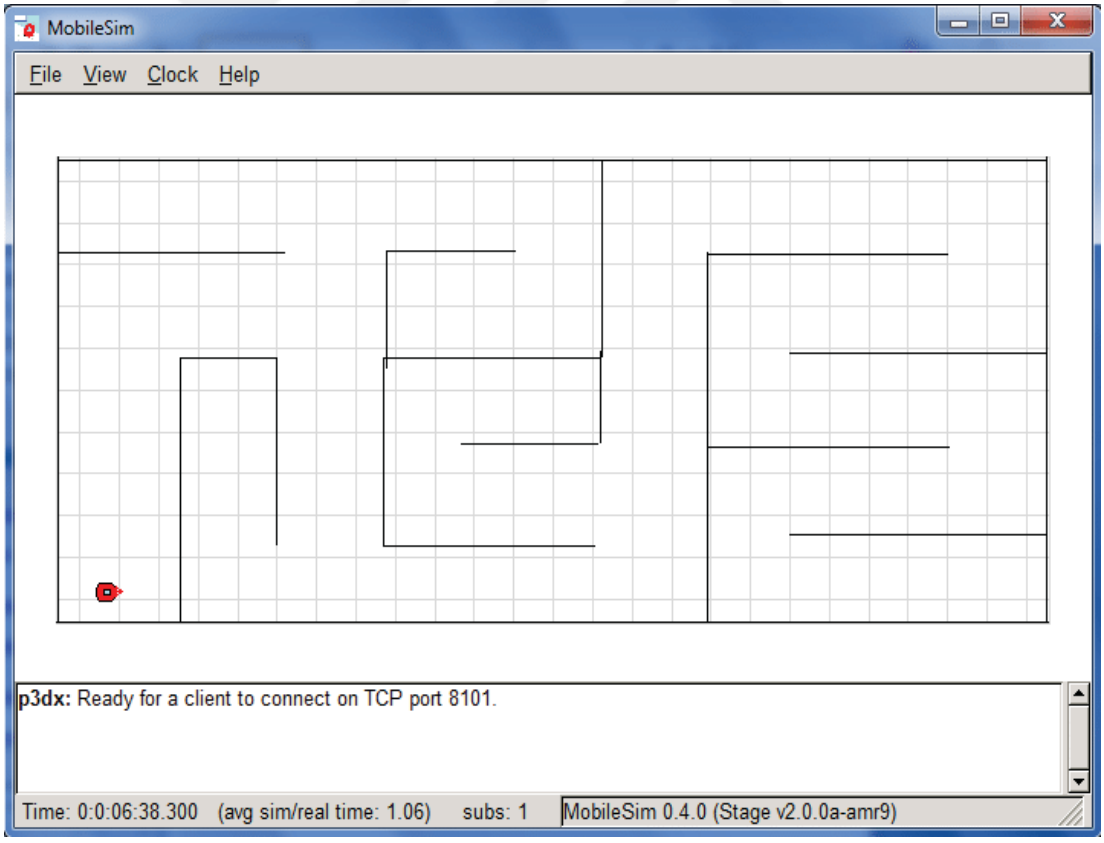
Robot İşletim Sistemi (RİS) robotları kontrol etmeye yarayan açık kaynak kodlu yazılım iskeletidir. Stanford Üniversitesinde ilk çalışmalara başlandıktan sonra Willow Garage adlı şirket tarafından geliştirilmiştir. İsminde işletim sistemi ifadesi geçse de aslında RİS bir işletim sistemi değildir. RİS yapısını kullanabilmek için Linux tabanlı bir işletim sistemi kullanmak gerekmektedir. C/C++, Java, Python, Lua, Lisp gibi birçok programlama dilini desteklemektedir. İçerisinde yer alan standart kütüphaneleri dışında robot geliştiricileri tarafından oluşturulmuş birçok kütüphaneyi de bünyesine katarak robot araştırma ve geliştirme çalışmalarında standart haline gelmektedir.

RİS yapmak istenilen bir uygulamada kodlar üzerinde küçük değişiklikler yaparak robota istenilen işi yaptırmayı hedeflemektedir. Robota bir işi yaptırmak için yükle

kullan şeklinde hazır kod yapıları oluşturulmak istenmektedir. RİS yazılan kodun her robot için yeniden yazılması yerine bir sefer yazılan kodun benzer tüm robotlarda çalıştırılabilmesini hedeflemektedir. Örneğin; bir robotun herhangi bir motorunun konum bilgisini aldığımız kod ile başka bir robotun motorunun konum bilgisine erişebilmek gibi standart bir yapı oluşturulmaktadır. RİS'in asıl amacı programcı ve robot arasında bir standart oluşturmaktır. RİS, Gazebo ile tam uyumlu çalışmaktadır.

3.5 MobileSim

Active Media şirketi tarafından açık kaynak kodlu olarak mobil robot geliştirme ve simüle etme yazılımı olarak geliştirilmiştir. MobileSim simülâtörü, Pionner robotlar için geliştirilmiştir. ARIA arayüzü ile TCP üzerinden robot kontrolü ve haberleşmesi sağlanmaktadır. ARIA Linux ve Windows işletim sistemlerinde çalışan C++ ile geliştirilmiş, açık kaynak koda sahip bir kütüphanedir.



Şekil 3.4 MobileSim Simülâtör ekran görüntüsü örneği [55]

Simülâsyon ortamındaki engelleri çizgiler ile tanımlanmaktadır. Bu çizgileri harita temelli ücretsiz bir yazılım olan MapperBasic ile oluşturulabilmektedir. MapperBasic

ile oluşturulan ortam .map uzantılı dosya olarak kaydedilir. Kaydedilen bu dosya MobileSim simülâtörü ile açılmaktadır. Şekil 3.4'te MobileSim simülâtörünün arayüz ekran görüntüsü örneği verilmiştir.

MobileSim LİDAR, GPS, jiroskop, ivmeölçer, mesafe sensörü gibi birçok sensör verilerini işleyebilmektedir. Yapılan simülâsyon içerisinde aynı anda birden çok robot simüle edilebilmektedir.

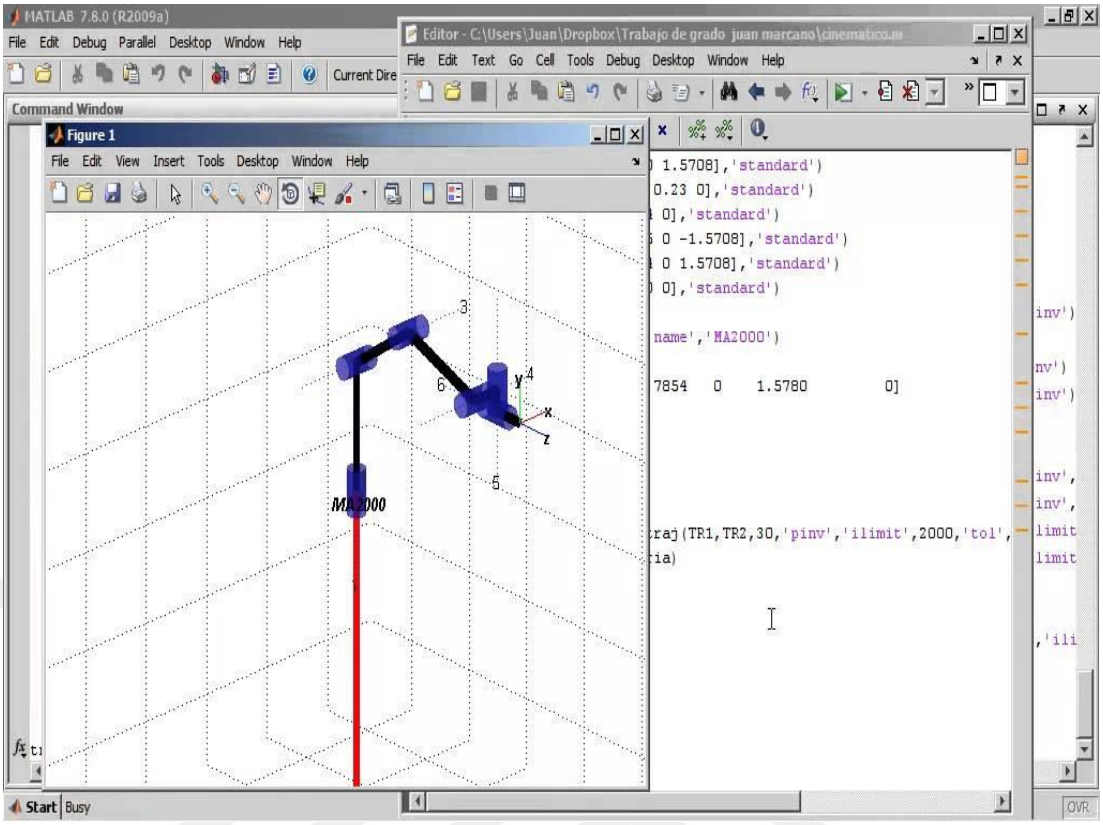
3.6 MATLAB

MATLAB robotik kodlama için kullanıcıya çeşitli hazır algoritmalar ve işlevler sağlamaktadır. MATLAB üzerinden robotlar derin öğrenme algoritmaları kullanılarak hızlı bir şekilde simüle edilebilmektedir.

MATLAB Simulink, robot programlamak için Model Tabanlı Tasarım ile modelleme ve simülâsyonu kullanabilmek için önceden oluşturulmuş bloklar sunmaktadır. Ayrıca MATLAB içerisinde bulunan Robotics System Toolbox, Navigation Toolbox, ROS Toolbox ile gelişmiş birçok robot aracı kullanılarak simülâsyonlar oluşturulmaktadır.

MATLAB Robotics System Toolbox, mobil robotları tasarlamak, simüle ve test etmek için araçlar ve algortimalar sağlamaktadır. Yol planlama, çarpışma kontrolü, haritalama, hareket kontrolü gibi birçok hazır algoritmayı içinde barındırır. MATLAB Navigation Toolbox haritalama işlemleri, iki ve üç boyutlu tasarımları gerçekleştirme işlemlerini gerçekleştir. IMU, GPS, mesafe sensörü, ivmeölçer, jiroskop gibi sensör verilerine ulaşmak için kullanılan bir araçtır. Otomatik sürüş, robot ve tüketici elektroniği uygulamalarında kullanılmaktadır.

MATLAB RİS ile uyumlu çalışmaktadır. Simülink içerisinde bulunan ROS blokları sayesinde kullanıcı sensör verilerine ulaşıp işlem gerçekleştirebilmektedir. Birçok hazır RİS bloğu barındırdığından çok az kod ile birçok işlem gerçekleştirilebilmektedir. Şekil 3.5'te MATLAB Simulink ile gerçekleştirilmiş simülâsyon ekran görüntüsü örneği verilmiştir.



Şekil 3.5 MATLAB Simulinkte robot programlama ekran görüntüsü örneği [56]

MATLAB aracılığı ile prototip oluşturma, konsept modellerini test etme, robot programlama gibi robot simülâsyon işlemleri oluşturulmaktadır. MATLAB Simulink üzerinden bir RİS ağı oluşturularak RİS bloklarına erişim sağlanıp simülâsyon yapılmaktadır.

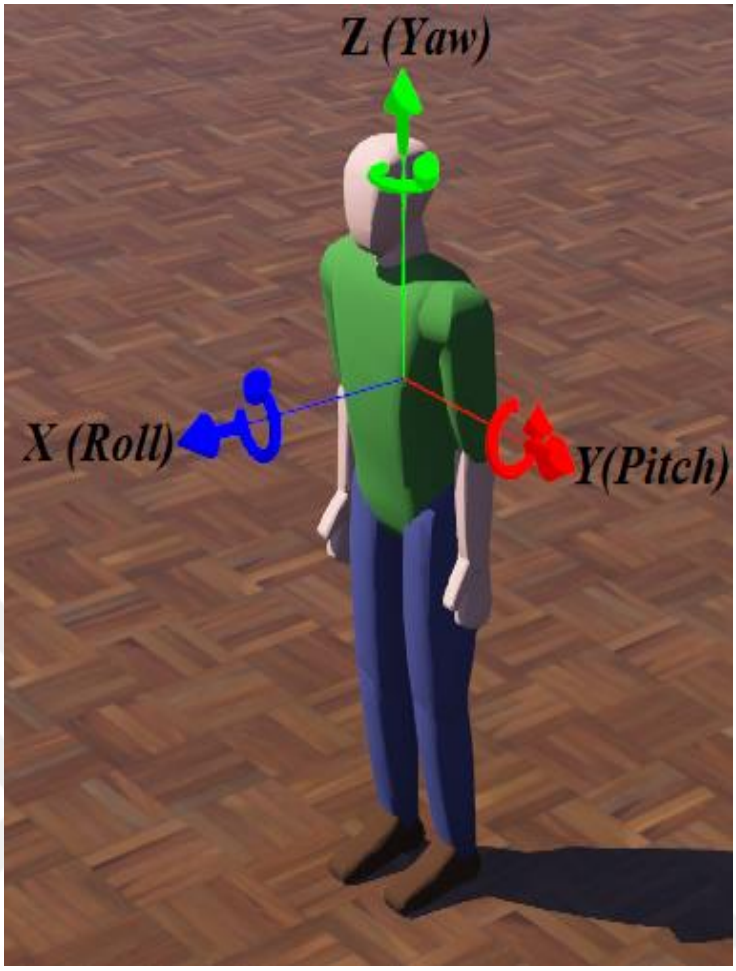
4. İTME-KURTARMA KONTROLÖRLERİ

İnsansı robotlar insana benzerler ve insan hareketlerini taklit etmeye çalışırlar. İnsansı robotlar üzerine yapılan çalışmalarda gelen dış itmelere karşı dengede kalabilme problemi en güncel ve zorlu konuların başında gelmektedir. Bir robot yürürken veya ayakta durduğu zaman dışardan gelen anormal her tepki itme olarak algılanmaktadır. Robota uygulanan itme sonucunda robotun denge ritmi bozulmaktadır. Robotun dengesini kaybetmesi düşmesine sebep olmaktadır. Dışardan gelen tepkiler dışında zeminden kaynaklı olan engellerde robotun denge problemi yaşamasına sebep olacağından bunlarda dış itme olarak algılanmaktadır. Robotların dışardan gelen itmelere karşı dengesini koruması istenmektedir. Araştırmacılar itme-kurtarma kontrolörleri ile robotları dengede tutmaya çalışmaktadır. İnsansı robotların dengede kalabilmesi insanların dengede kalması ile benzerdir. Bu yüzden insan yürüyüşünün matematiği iyi analiz edilmeli ve robotlara uygulanmalıdır.

4.1 İnsansı Yürüme

İnsansı robotlarda yürüme birçok alana hitap eden bir araştırma konusudur. İnsansı robotların yürüyüşünü analiz edebilmek için öncelikle insan yürüyüşünün temellerinin nasıl olduğu iyi anlaşılmalıdır. İnsansı yürüyüş için 3 boyutlu bir düzlemin tanımlanması gerekmektedir. Şekil 4.1'de insan hareketlerinin analizini kolaylaştıran üç temel referans düzlem ve bunlara karşılık gelen dönüş yönleri gösterilmiştir. X ekseni robotun ileri/geri yönü, Y ekseni robotun sağ/sol yönü ve Z ekseni robotun yukarı/aşağı yönüdür. YZ düzlemi roll, XZ düzlemi pitch ve XY düzlemi yaw dönüş açıları olarak adlandırılır.

İki ayaklı bir robotun adım atarak ilerlemesi yürüyüş olarak tanımlanır. Yürüyüş esnasında iki farklı destek noktası oluşmaktadır. Bunlar tek destek ve çift destek noktası olarak adlandırılmaktadır. Tek bir bacak üzerinde tüm robot vücut ağırlığını dengede olduğu faz tek destek noktası olarak adlandırılır. İki bacak yerde iken tüm robot vücut ağırlığının bu iki bacak tarafından dengelendiği faz ise çift destek noktası olarak tanımlanmaktadır.

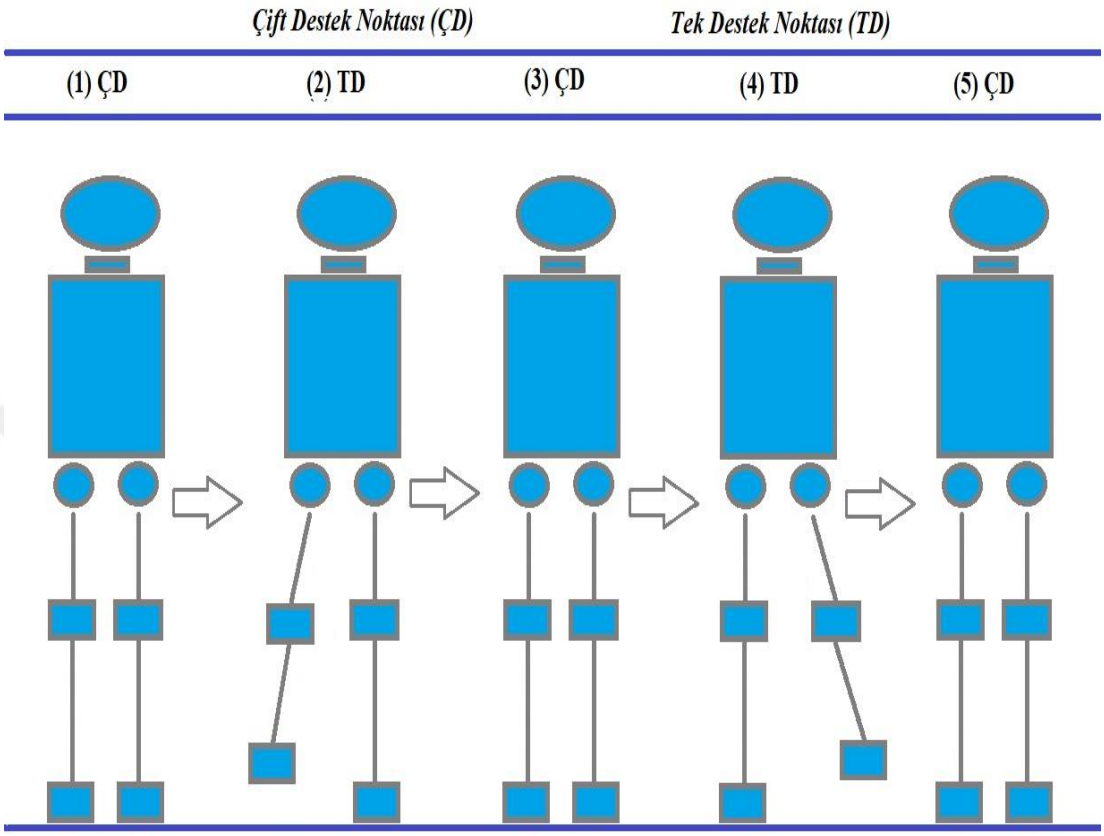


Şekil 4.1 İnsansı hareket referans düzlemi

İnsansı robotlarda yürüme esnasında denge noktası robotun ayakta kalmasını sağlamaktadır. Yürüyüş hareketinin belli bir periyot ile devam etmesi yürüme döngüsünü oluşturur. Yürüme döngüsü beş aşamaya ayrılabilir. Yürüme döngüsünün her aşamasında insansı robotun önden görünümü Şekil 4.2'de gösterilmiştir. Birinci aşamada, robot yürümeye başlamadan önce çift destek noktasındadır. İkinci aşamada, robot sol ayağını kaldırıp tek destek noktasına geçer. Üçüncü aşamada sol ayağını ilerletip zemine bırakır ve çift destek noktasına tekrar geçer. Dördüncü aşamada, sağ ayağını kaldırıp tek destek noktasına geçer. Son olarak beşinci aşamada, sağ ayağını ilerletip zemine bırakır ve çift destek noktasına geçerek yürüme döngüsünü tamamlar. Stabil bir yürüyüş gerçekleştirmek için yürüyüş döngüsündeki aşamalar gerçekleştirilmelidir.

Gelişmiş motorlara sahip olan robotlarda denge problemini aşmak ucuz endüstriyel motorlara göre daha kolaydır. Fakat bu motorlar robotun maliyetini çok fazla

arttırmaktadır. AR-GE amaçlı kullanılan ucuz endüstriyel servo motor kullanan insansı robotlarda denge problemi sık karşılaşılan bir durumdur.



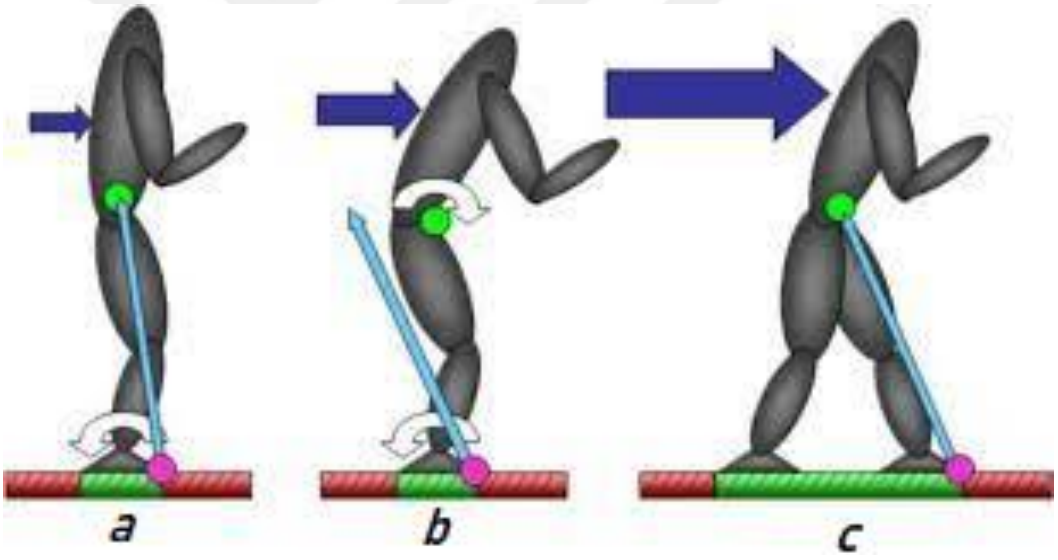
Şekil 4.2 İnsansı robotlarda yürüme döngüsünün önden görünümü

İnsansı robotlarda gelen dış itmeden kurtarmak için iki temel itme-kurtarma kontrolörü bulunmaktadır. Bunlar, düşük seviyeli itme-kurtarma kontrolörleri ve yüksek seviyeli itme-kurtarma kontrolörleridir. Düşük seviyeli itme kurtarma kontrolörleri, ayak bileği stratejisi, kalça stratejisi ve adım stratejisinden oluşur. Bu stratejiler insanların denge bozukluğu ve tedirginlik durumlarında denge kontrolünü sağlayıp bununla nasıl başa çıktıklarının biyomekanik bir analizidir. Yüksek seviyeli itme kurtarma kontrolörleri sensörlerden aldıkları verileri kullanarak hangi düşük seviyeli itme kurtarma stratejilerinin kullanılacağına karar verir. Ana işlemci alınan sensör verilerinden açısal hız ve doğrusal ivme verisini ölçerek tek bir modülde toplar ve bu verileri yüksek seviyeli itme kurtarma kontrolörüne iletir.

4.2 İnsansı Robotlarda Denge Stratejileri

İnsansı robotların en büyük denge sorunlarından olan dış kuvvetler ve engebeli arazilerde yürüme için birçok çözüm sunulmuştur. Mevcut yürüme algoritmalarının çoğu, zemine önceden tanımlanmış statik bir açıyla ve ayakları ona paralel olacak şekilde düz yüzeylerde yürümek üzere tasarlanmıştır. Bu nedenle, yürüme sırasında insansı bir robota yapılan her harici itme ile robotun düşme olasılığı yükselir. Bu sorunun üstesinden gelmek için birçok farklı strateji geliştirilmiştir. Bu stratejiler arasında üç temel yöntem ayak bileği, kalça ve adım stratejileridir.

Ayak bileği stratejisi; Kütle Merkezini (Center of Mass-COM) değiştirerek robotun ayak bileklerini dengelemek için kullanan Basınç Merkezi (Center of Pressure-COP) stratejisidir [2,5,6].



Şekil 4.3 Üç temel dengeleme stratejisi. Yeşil nokta kütle merkezini, pembe nokta basınç merkezini ve mavi ok yer reaksiyon kuvvetini temsil eder. a) Ayak bileği stratejisi b) Kalça stratejisi c) Adım stratejisi [1]

Kalça stratejisi; Robotu dengede tutabilmek için hem ayak bileklerini hem de kalçalarını kullanan, vücudun pozisyonunu değiştirmesine ve gövdeyi sallayarak açısal bir momentum oluşturmasına izin veren Merkezi Moment Ekseni (Centroidal Moment Pivot-CMP) stratejisidir [1].

Adım stratejisi; Dış kuvvetleri uygun bir yönde bir veya daha fazla adım atarak denge konumuna geçiren Yakalama Noktası (Capture Point-CP) stratejisidir [11]. İtme-kurtarma stratejileri Şekil 4.3'te verilmiştir.

4.2.1 Ayak bileği stratejisi

Ayak bileği stratejisi, robotun kütle merkezini kontrol eder. Basınç merkezi stratejisi olarak da bilinir. Mekanik olarak bu strateji, robot vücudunu ayak bileği eklemleri etrafında döndürerek dengeyi sağlamasından ibarettir. Bu stratejide diğer eklemler sabittir. Kütle merkezini dengede tutmak için ayak bileği eklemlerine kontrol torku uygulanır. Böylece robot uygulanan tork ile ters çevrilmiş sarkaç gibi davranır. Bu stratejide; ayak bileği torkunu kontrol etmek için ya ayak bileğindeki sıfır moment noktası kontrol edilir, ya da ayak bileğinde bulunan servonun hedef açısı düzeltilerek yapılabilir. Ayak bileği stratejisi 4.1'de verilmiştir.

$$\ddot{x} = \frac{g}{z_0}\left(x - \frac{\tau_{ayak_bileği}}{mg}\right) \tag{4.1}$$

Burada; $\tau_{ayak_bileği}$ tork değerini, m doğrusal ters sarkaç modelinin kütlesini, g yerçekimini ve z_0 kütle merkezinin yüksekliğini ifade etmektedir. x, kütle merkezinin mevcut destek noktasından yatay mesafesidir. Şekil 4.4.a'da ayak bileği stratejisine dayalı model verilmiştir.

4.2.2 Kalça stratejisi

Kalça stratejisinde, robot dengede kalabilmek için gövdesini kullanarak itme kuvvetine karşı ters yönlü bir kuvvet oluşturur. Diğer bir ifadeyle, robotun kütle merkezinin hareketine karşı koymak için kalçada bulunan servo motorunun açısal ivmesini kullanan bir itme-kurtarma kontrolörüdür. Kalça stratejisi Merkezi Moment Ekseni stratejisi olarak da bilinir. Bu metot ile robotun kalça kısmında bulunan servo motorlar gelen itmeye uygun bir tork üreterek dengede kalmaya çalışır. İnsanların gelen itmelere karşı kalça eklemlerini kullanarak dengede kalmaya çalışmalarına benzer bir yol izlerler. Bu strateji ayak bileği stratejisine göre daha etkili bir yöntemdir. Kalça stratejisini matematiksel model denklemi:

$$\ddot{x} = \frac{g}{z_0}\left(x - \frac{\tau_{kalça}}{mg}\right) \tag{4.2}$$

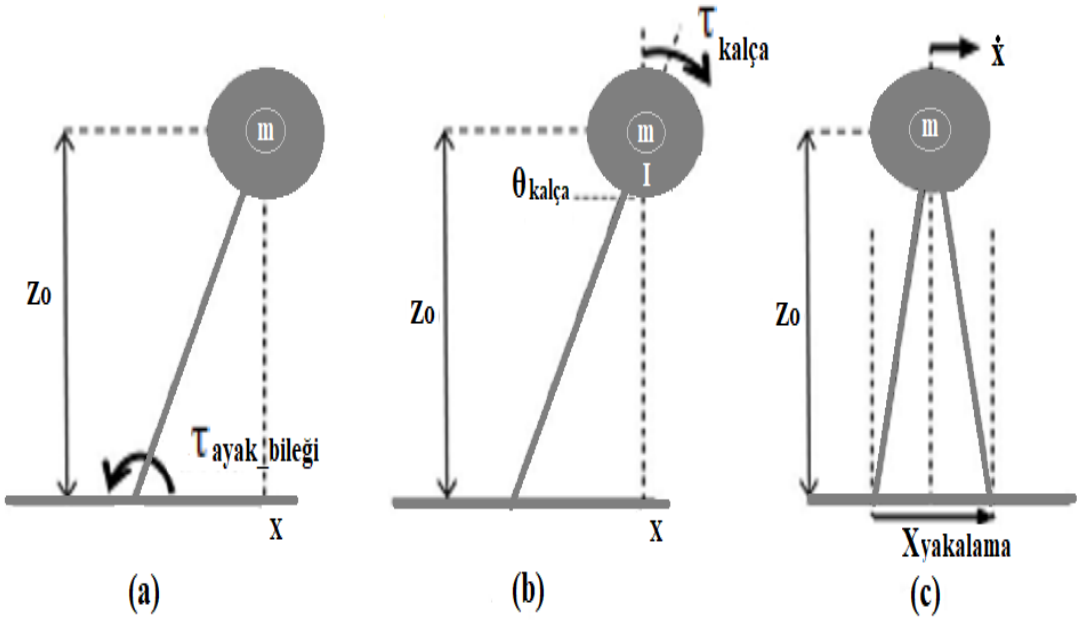
$$\tau_{kalça} = I\theta \tag{4.3}$$

gibidir. Burada; $\tau_{kalça}$ gövdeye uygulanan tork değerini, m modelinin kütlesini, g yerçekimini,z_0 kütle merkezinin yüksekliğini ve I dönme ataletini ifade etmektedir. x, kütle merkezinin mevcut destek noktasından yatay mesafesidir. Şekil 4.4.b'de kalça stratejisine dayalı model verilmiştir. İnsansı robotlar eksenleri etraflarında dönemez, bu yüzden bang-bang kontrol tekniği kullanılmıştır. Bang-bang kontrol yöntemi $\tau_{kalça}(t)$ fonksiyonu için önce T_{R1} periyodu için maksimum torku uygular. Ardından T_{R2} periyodu için maksimum negatif torku uygular. Bu teorinin matematiksel denklemi aşağıdaki gibidir:

$$\tau_{kalça}(t) = \tau_{max}u(t) - 2\tau_{max}u(t - T_{R1}) + \tau_{max}u(t - T_{R2}) \tag{4.4}$$

$$\theta(T_{R2}) = \theta_{max} \tag{4.5}$$

Burada; τ_{max} eklemin uygulayabileceği maksimum tork, T_{R1} gövdenin hızlanmasının durduğu zaman, T_{R2} gövdenin durduğu zaman, u(t) birim adım fonksiyonu ve θ_{max} maksimum eklem açısını ifade etmektedir. Bu kontrolör, kalça servosunun konumunu neredeyse hedef açısına ulaşana kadar maksimum tork (τ_{max}) uygulayarak çalışmasını sağlar. Sistemi durdurmak için ise tersi yönde tork uygulanır. Hedef konuma ulaştıktan sonra, kalça açısının değeri başlangıç değerine geri getirilmelidir.



Şekil 4.4 a) Ayak bileği stratejisi b) Kalça stratejisi c) Adım stratejisi

4.2.3 Adım stratejisi

Robota uygulanan itmenin tork değeri arttıkça, robotun dengede durması zorlaşır. Dışardan gelen itme robotun dayanacağı maksimum tork değerini aştığı zaman ayak bileği ve kalça stratejileri yetersiz kalır ve robotun dengede kalması için adım atması gerekir. Robota gelen itmeye karşı destek noktasına adım atarak yapılan bu hamle adım stratejisi olarak adlandırılmaktadır. Robotun düşmeden dengede kalabilmesi için atacağı en iyi adım uzunluğu, denge noktası içinde olmalıdır. Adım stratejisinde, itmenin geldiği yöne doğru destek adımı atılarak denge sağlanır. Bu strateji aynı zamanda yakalama adımı olarak da bilinir. Yakalama noktası, insan hareketlerinde de çok kullanılan bir kavram olup, robotun kendini tamamen durdurması ya da dengede kalması için atmış olduğu destek adımıdır. Şekil 4.4.c'da adım stratejisine dayalı model verilmiştir. Dengede olmayan bir robotun yakalama noktası gövde hızıyla orantılıdır ve şu şekilde hesaplanır:

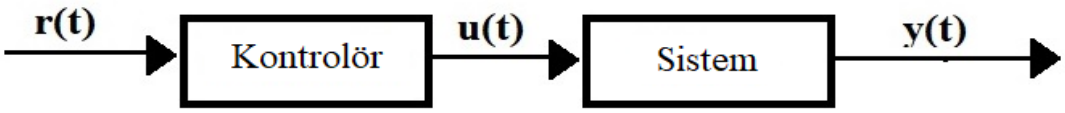
$$X_{yakalama} = \dot{x}\sqrt{\frac{z0}{g}} \tag{4.6}$$

Burada; $X_{yakalama}$ yakalama noktası, $\dot{x}$ gövdenin doğrusal hızıdır. Adım stratejisi, yakalama noktasının robotun gövdesinin doğrusal hızıyla orantılı olduğunu ve robotu itmeden-kurtarmak için atılması gereken adımın uzaklığını göstermektedir.

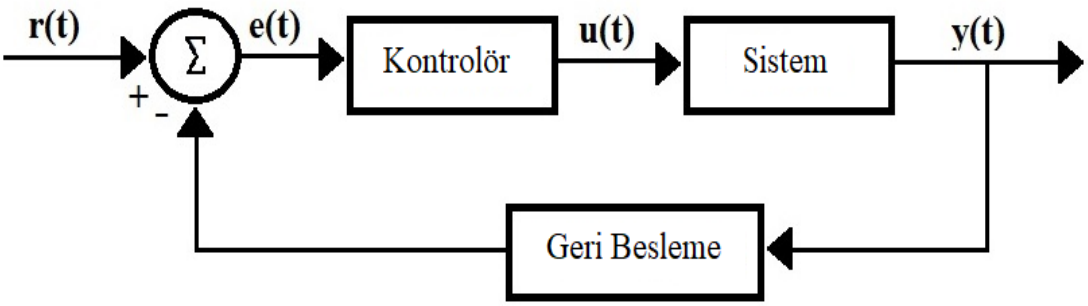
5. KONTROL YÖNTEMLERİ

Teknolojik gelişmeler ilerledikçe buna paralel olarak sistemlerin kontrol edilme gereksinimi oluşmuştur. Kontrol sistemlerinin temel amacı, sistem içerisinde yer alan değişken ve durumları kullanarak sistemin uyumlu bir şekilde çalışmasını sağlamaktır. Sistem belli bir amaç ve işlevi olan uyum içerisinde görevi yerine getirmeye çalışan parçalar bütünüdür. Sistemlerin belirli bir yöntem kullanılarak istenilen zaman ve sonuç doğrultusunda kararlı hale gelmesi kontrol sayesinde gerçekleşmektedir. Kontrol sistemlerinde, sistem girişlerine uygulanan sinyal kontrolörde işlenir ve sistemin yapısına uygun bir çıkış sinyali üretir.

Açık çevrim ve kapalı çevrim olmak üzere temelde iki çeşit kontrol sistemi bulunmaktadır. Açık çevrim kontrol sistemlerinde uygulanan sinyal sistem tarafından kontrol edildikten sonra sisteme geri bildirim vermeksizin çıkış sinyalini verir. Kapalı çevrim kontrol sistemleri ise çıkış sinyali tekrar sisteme geri besleme olarak uygulanmaktadır. Kapalı çevrim kontrol sistemleri geri beslemeli kontrol sistemleri olarak da bilinmektedir. Açık çevirim kontrol sistemi blok diyagramı Şekil 5.1'de kapalı çevrim kontrol sistemi blok diyagramı Şekil 5.2'de verilmiştir.



Şekil 5.1 Açık çevrim kontrol sistemi blok diyagramı



Şekil 5.2 Kapalı çevrim kontrol sistemi blok diyagramı

Bir sistemin verimini arttırmak ve hatayı en aza indirmek için sistemlerde kontrol mekanizması kullanılır. Geçmişten günümüze kadar endüstride yaygın olarak klasik kontrolörlerden PI, PD ve PID kullanılmaktadır. Bu kontrolörler parametre ayarlayabilme ve basit yapıları dolayısıyla sıkça tercih edilmektedir. Teknolojinin gelişmesi ve bu kontrolörlerin yetersiz kaldığı durumlar için yeni kontrol yöntemleri geliştirilmiştir.

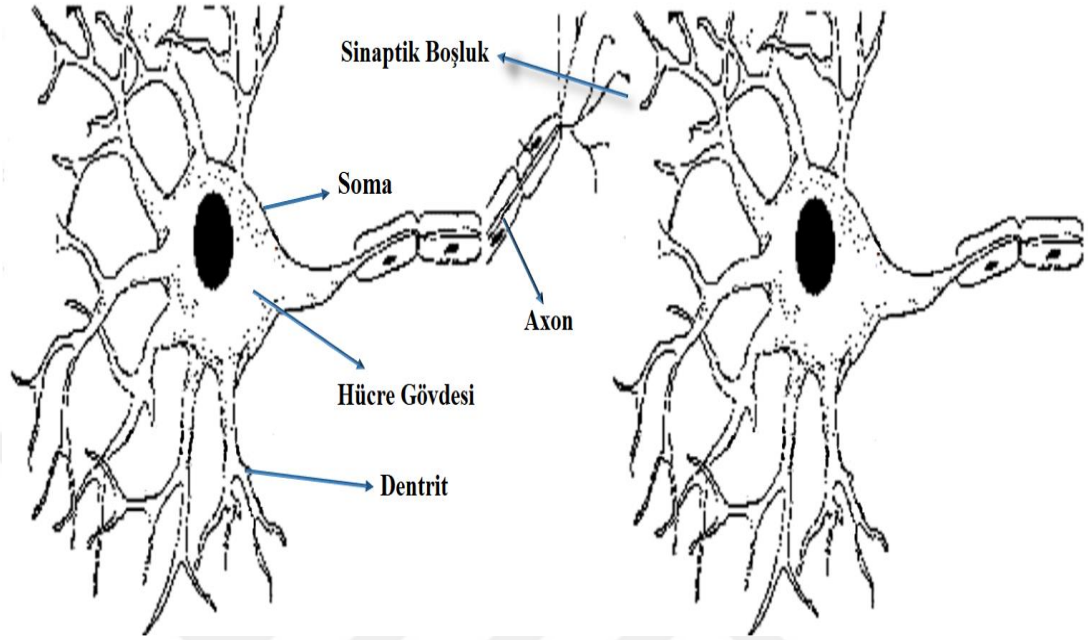
Kontrol edilecek sistemlerin ve uygulama alanlarının artması araştırmacıları klasik kontrol yöntemlerin dışında yenilikler aramaya itmiştir. Böylece modern kontrol yöntemlerinin gelişmiştir. Günümüzde robotik, savunma sistemleri, bilgisayar, otomotiv, havacılık, elektronik, ziraat, eğitim, güç sistemleri gibi birçok farklı alanda kontrol sistemleri kullanılmaktadır. Modern kontrol sistemlerinin ortaya çıkması ile sistemlerin performanslarında iyileşmeler gerçekleşmiştir. Modern kontrol yöntemleri içerisinde makine öğrenmesi, uyarlamalı kontrol, bulanık mantık, model öngörülü kontrol, derin pekiştirmeli öğrenme, yinelemeli öğrenme gibi kontrol yöntemleri bulunmaktadır.

5.1 Yapay Sinir Ağları

YSA'lar; insan beynini modelleyerek taklit eden, makine öğrenmesinde yaygın olarak kullanılan yöntemlerden biridir. Bu model biyolojik sinir sistemindeki sinir hücrelerini (nöron) temel almaktadır. YSA'larda insan sinir sistemindeki nöronlar gibi yapay nöronlar tanımlanmıştır. Bu nöronlar birbirleriyle ilişkili olacak şekilde tasarlanmıştır. YSA'lardaki yapay nöronlar birbirleri bağlantılı olduklarından veriler arasında bağlantı kurma, hafızada saklama, öğrenme gibi faaliyetleri kolaylıkla yapabilmektedir. YSA'lar ile kümeleme, sınıflandırma, tahmin, görüntü ve ses tanıma, ilişkilendirme, optimizasyon gibi birçok uygulama kontrolü gerçekleştirilebilmektedir. Bilgisayar, elektrik, elektronik, haberleşme, tıp, finans, matematik gibi alanlarda sıkça kullanılan bir model olarak karşımıza çıkmaktadır.

YSA'lar insanların öğrenme tekniğinden faydalanarak modellenmiştir. Canlı sistemlerde bulunan nöronlar, biyolojik sinir ağlarımızın en temel elemanlarıdır. Gelen sinyali çekirdeğe ileten kısım dentrit olarak isimlendirilir. Soma, gelen uyartıyı toplayan kısımdır. Axon ise uyartımı diğer hücrelere dağıtır. Sinaptik boşluklar ise

uyartımların diğer sinir hücrelerine iletilmesinde axon ve dentrit arasında rol alır. Biyolojik nöron yapısı Şekil 5.3'te verilmiştir.



Şekil 5.3 Biyolojik nöron hücresinin yapısı

Canlı sistemlerde bulunan sinir ağlarındaki gibi yapay sinir ağlarında da nöronlar bulunmaktadır. YSA'lardaki bu nöronlara düğüm de denilmektedir. Yapay sinir ağlarında nöronlar 5 temel elemandan oluşmaktadır:

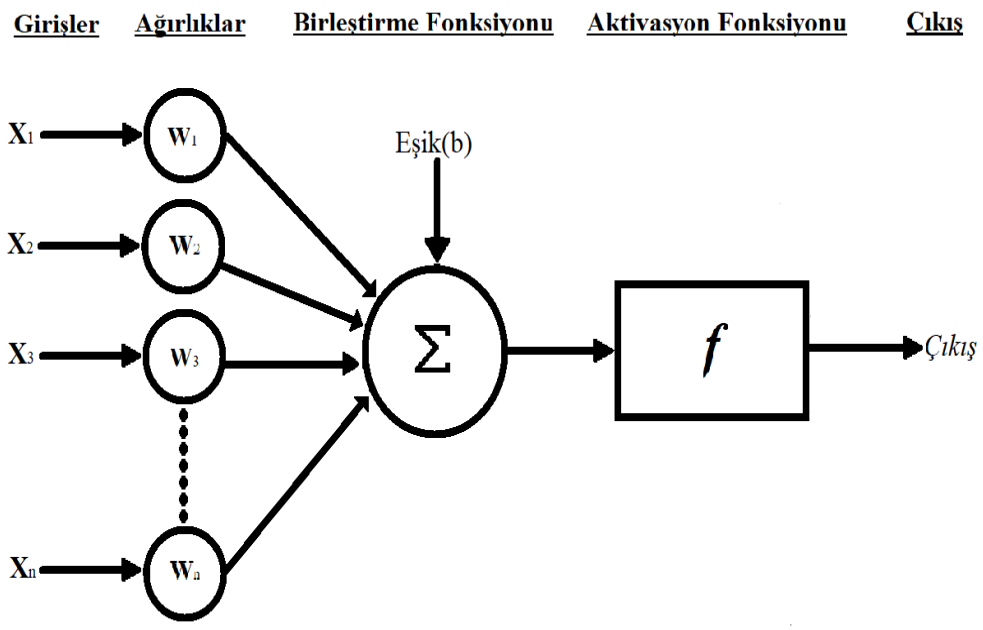
- Giriş
- Ağırlık
- Birleştirme Fonksiyonu
- Aktivasyon Fonksiyonu
- Çıkış

Girişler, dışardan gelen bilgilerdir. Her düğüm diğer düğümlerden veya dış dünyadan gelen girdi değerlerine sahiptir. Ağırlık, bir nörona gelen bilginin önem derecesini gösteren bir kavramdır. Ağırlıklar bir nöron üzerindeki etkiyi belirlemekte kullanılmaktadır. Her girişin farklı ağırlık değeri bulunmaktadır. Düğümlere gelen girdiler ağırlık değeri (w) ile çarpılır. Birleştirme fonksiyonu ile nöronun net girişi hesaplanır. Birleştirme fonksiyonu kullanılarak çarpma işlemine eşik değeri (b) eklenir ve toplam değer bulunur. Buradaki eşik değeri nörona eğitilebilir sabit bir değer sağlamaktadır. Aktivasyon fonksiyonu, gelen girişleri işleyerek çıkışı belirlemek için

kullanılır. Birleştirme fonksiyonu kullanıldıktan sonra çıkan bu toplam değere bir aktivasyon fonksiyonu uygulanır ve çıkış değeri üretilir. Burada aktivasyon fonksiyonu kullanılmasındaki amaç, nörona lineer olmayan özellik eklemektir. Burada lineer olmamayı nöron çıkışına (5.1)'deki gibi aktarılmalıdır.

$$y = f\left(\sum_{i=1}^{n} W_i x_i + b\right) \tag{5.1}$$

YSA'larda en çok Sigmoid, Rectified Linear Unit (ReLU), Tanh aktivasyon fonksiyonları kullanılmaktadır. Yapay sinir ağlarında nöron yapısı Şekil 5.4'te verilmiştir.



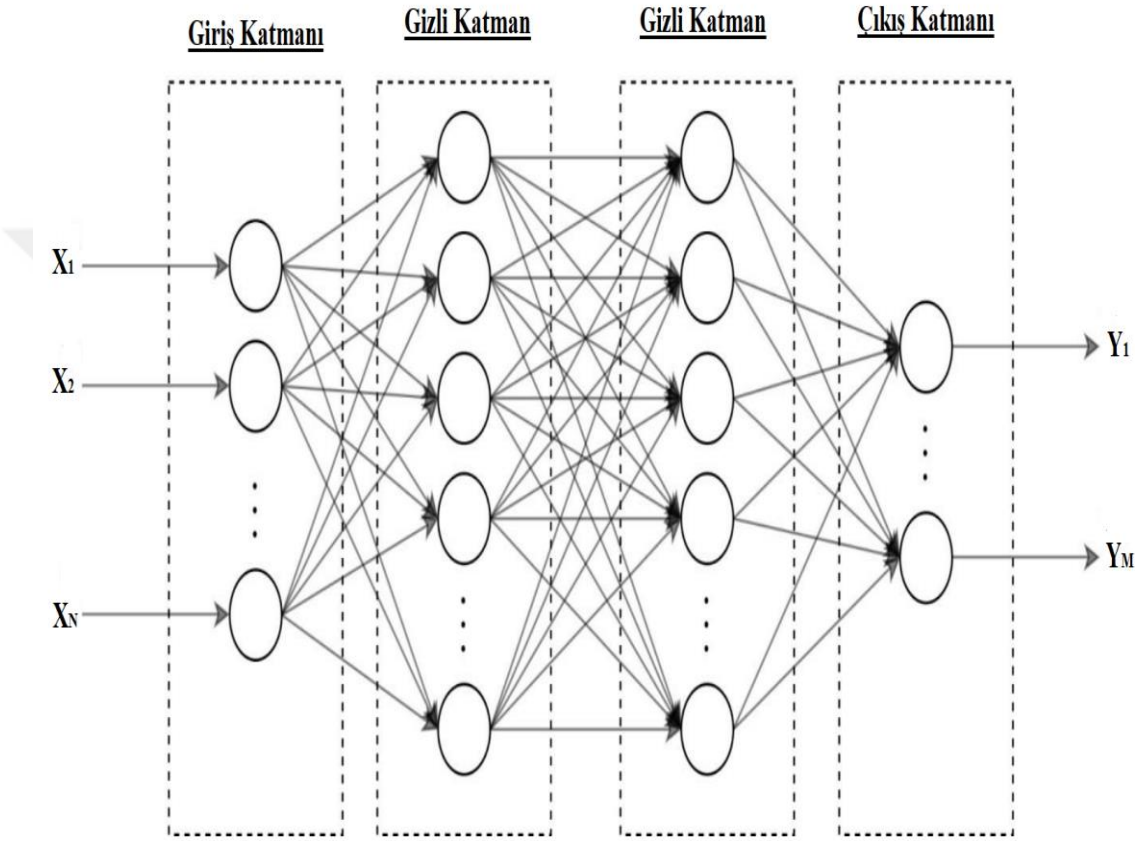
Şekil 5.4 Yapay sinir ağı nöron yapısı

Yapay sinir ağları katmanlı bir yapıya sahiptir. Yapay sinir ağlarında 3 temel katman bulunmaktadır. Bunlar,

- Giriş katmanı
- Gizli katman
- Çıkış katmanıdır.

Giriş katmanı sayesinde, sinir ağına dış dünyadan verilerin girilmesi sağlanır. Bir katmanda bulunan nöronun çıkışı sonraki katmanda ağırlık ile çarpılıp eşik değeri

eklenerek giriş olarak verilir. Gizli katman nöronların yer aldığı katmandır. Çıkış katmanı ise veriyi işledikten sonra dış dünyaya aktarılmasını sağlayan katmandır. Şekil 5.5'de yapay sinir ağı modeli verilmiştir. Eğer bir yapay sinir ağında tek bir gizli katman var ise bu sığ (shallow) yapay sinir ağıdır. Birden fazla gizli katmanı bulunan yapay sinir ağları derin (deep) yapay sinir ağı olarak adlandırılmaktadır. Yapay sinir ağları derin öğrenme çalışmalarının temelini oluşturmaktadır.



Şekil 5.5 Yapay sinir ağı modeli

Çok katmanlı bir YSA'da her bir nöron bir sonraki katmanda bulunan nöronlarla bağlantılıdır. Aynı katmanda bulunan nöronlar arasında bağlantı bulunmamaktadır. Derinlik kavramı YSA'nın içerisinde bulunan gizli katman sayısı ile bağlantılıdır. Bir YSA'da ne kadar çok gizli katman varsa derinliği o kadar artmaktadır. Çok katmanlı bir YSA'da öğrenme her katmandaki ağırlıkların değiştirilmesi ile oluşur. Bundan dolayı ağırlıklar en uygun değerler olarak belirlenmelidir. Bir YSA'nın öğrenmesi uygun ağırlık değerleri ile gerçekleşmektedir.

5.2 Derin Pekiştirmeli Öğrenme

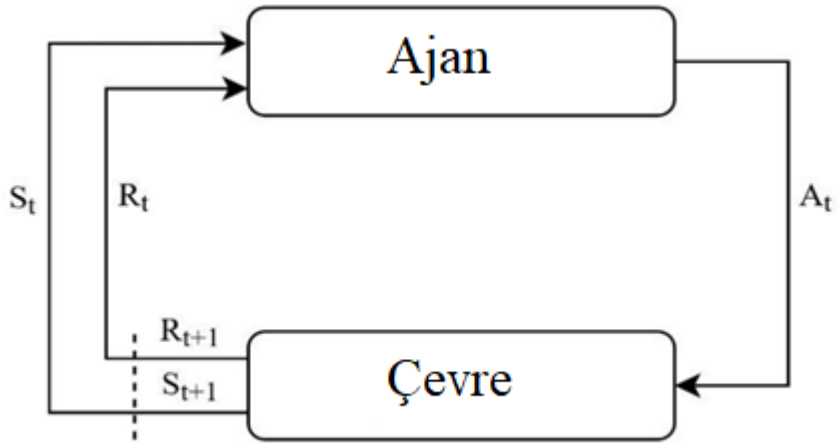
Derin öğrenme, üç ya da daha fazla katmandan oluşan yapay sinir ağı olan makine öğrenmesinin bir yaklaşımıdır. Derin öğrenme, büyük verileri giriş olarak kullanıp işledikten sonra anlamlı bir bilgi üretmektedir. Derin öğrenme algoritmalarını makine öğrenmesinde bulunan algoritmalardan ayıran yönü, büyük miktardaki veriyi hızlı bir şekilde işleyebilme özelliğidir. Bu özelliği dolayısıyla derin öğrenme algoritmaları için yüksek işlem hızı olan donanımlara ihtiyaç duyulmaktadır. Makine öğrenmesi gözetimli öğrenme, gözetimsiz öğrenme ve pekiştirmeli (takviyeli) öğrenme olarak sınıflandırılır.

Gözetimli öğrenmede, eğitim etiketlenmiş veri setleri ile gerçekleştirilmektedir. Burada hangi eylemin doğru hangisinin yanlış olduğu önceden belirlenmiştir. Bu bilgiler doğrultusunda olmayan veriler üzerinde tahmin yürüterek işlemini yürütür. Gözetimli öğrenmede veri setinin büyüklüğü eğitimin başarı oranını arttırmaktadır. Gözetimsiz öğrenmede etiketlenmiş veri bulunmamaktadır. Burada temel amaç veriler arasındaki gizli ilişkileri bularak kümeleme ve veri azaltma işlemlerini gerçekleştirmektir.

Pekiştirmeli öğrenme, bir ortamda farklı eylemler gerçekleştirerek en yüksek ödülü almayı hedefleyen bir makine öğrenmesi yöntemidir. Pekiştirmeli öğrenme, dinamik bir ortamla deneme-yanılma etkileşimi yoluyla öğrenme yeteneğine sahip bir modelidir. Girişlere uygun, sistemin hesapladığı çıkışların doğru veya yanlış olarak nitelendirilmesine dayanır. Pekiştirmeli öğrenmede diğer iki yöntemde olduğu gibi bir veri kümesi bulunmamaktadır. Burada sistem ödül ve ceza sistemi üzerine kuruludur. Pekiştirmeli öğrenmede amaç ödülü en yüksek düzeye çıkarabilmektir. Bu öğrenme süreci, öğrenme sonlandırılana kadar ödülü maksimize etmeye odaklanarak özyinelemeli olarak gerçekleşir.

Pekiştirmeli öğrenme ajan, çevre, durum, eylem (aksiyon) ve ödül bileşenlerinden oluşmaktadır. Şekil 5.6'da pekiştirmeli öğrenme blok diyagramı verilmiştir. Ajan, çevrede eylemlerde bulunan yapıyı ifade etmektedir. Çevre, ajanın içerisinde bulunduğu ortamdır. Çevre ajanın etkileşime girebileceği ve öğrenmeyi gerçekleştirebileceği bir yapıda olmalıdır. Eylem, ajan tarafından yapılan faaliyet ve

kararlar olarak nitelendirilir. Aksiyon olarak da isimlendirilir. A veya a ile ifade edilir. Durum, ajanın çevre içerisindeki bilgisini saklar. S veya s ile ifade edilir. Ödül ise ajanın çevrede yapmış olduğu eylem sonucunda almış olduğu geribildirimdir. Ödül yapılmış olan eylemin sonucuna göre negatif veya pozitif değerli olabilir. Ödül R veya r ile ifade edilmektedir. Pekiştirmeli öğrenmede temel amaç mevcut probleme çözüm sunacak en uygun politikayı bulmaktır.



Şekil 5.6 Pekiştirmeli öğrenme blok diyagramı

Pekiştirmeli öğrenmede ajan önceki tecrübelerinden faydalanarak yaptığı eylemler sonucunda başarılı ya da başarısız olarak öğrenme işlemini gerçekleştirmektedir. Ajanın hangi eylemi yapması gerektiği söylenmeden kendisi yapacağı eylemi belirlemektedir. Şekil 5.6'da R_t ajanın çevreden kazandığı ödülü, A_t ajanın eylemlerini ifade etmektedir. S_t ise ajanın çevredeki durumunu göstermektedir. Blok diyagramda da görüldüğü gibi ajan bir t zamanında S_t durumunda iken A_t eylemini gerçekleştirerek R_{t+1} ödülünü alır ve S_{t+1} durumuna geçer.

Pekiştirmeli öğrenme algoritmaları model olarak Markov karar sürecini temel almaktadır. Markov karar modeli ayrık zamanlı bir karar sürecidir. Bu modelde ajanın belli bir durumda en yüksek ödülü alacağı eylem dizilerinin oluşturulması için olan problemlerde kullanılmaktadır. Modeldeki karar sürecinde eylem, durum ve ödüllerin sıralı gösterimi (5.2)'de verilmiştir. Markov karar sürecinde sonraki durum olan S_{t+1} bir tek şu anki durum olan S_t'ye bağlıdır.

$$\ldots S_t, A_t, R_t, S_{t+1}, A_{t+1}, R_{t+1}, S_{t+2}, A_{t+2}, R_{t+2}, S_{t+3}, A_{t+3}, R_{t+3}, \ldots \quad (5.2)$$

Belman eşitliği, Markov karar süreci problemlerinin dinamik programlama ile çözümlerinde hesaplama yükünün azaltılmasına katkı sağlayan denklemdir. Burada ajan farklı eylemler gerçekleştirerek en yüksek ödülü kazanarak problemi optimize etmektedir. Belman eşitliği (5.3)'te verilmiştir.

$$Q(S,A) = R_S^A + \gamma \sum_{s' \in s} P_{SS'}^A \; Q(S,A) \tag{5.3}$$

Burada; R_S^A mevcut durumda yapılan eyleme bağlı kazanılan ödülü ifade etmektedir. S durumundan S' duruma geçiş olasılığı $P_{SS'}^A$ ile ifade edilmiştir.

5.3 Derin Pekiştirmeli Öğrenme Algoritmaları

Derin pekiştirmeli öğrenme farklı algoritmalar içermektedir. Bu algoritmalar ajanın en yüksek ödülü alması için tasarlanmış yaklaşımlardır. Bu algoritmalardan bazıları ayrık zamanlı bazıları sürekli zamanlıdır. Q Öğrenme, SARSA, DQA, ÇDQA, D3QA, PPO, A2C, A3C, TRPO, NAF, DDPG gibi birçok derin öğrenme algoritması bulunmaktadır. Bu bölümde Q öğrenme, Derin Q Ağı (DQA), Çift Derin Q Ağı'ndan (ÇDQA) oluşan önemli pekiştirmeli öğrenme algoritmaları açıklanmıştır.

5.3.1 Q öğrenme

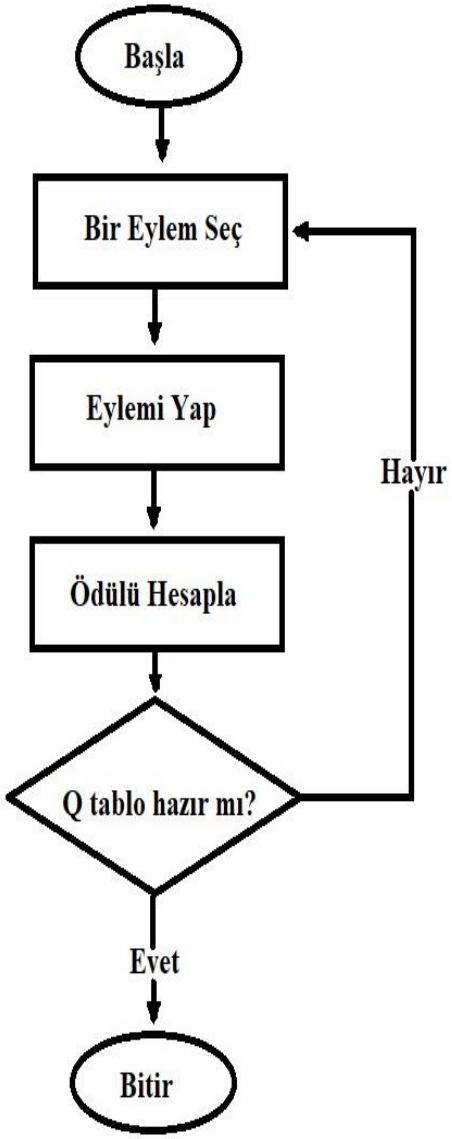
Pekiştirmeli öğrenmede en popüler algoritmalardan biridir. Q öğrenme çevre modeline ihtiyaç duymayan ayrık zamanlı politika dışı (off-policy) bir öğrenme yöntemidir. Belman eşitliği temeline dayanmaktadır. Politika dışı olması bir eylemi gerçekleştirmek için kullandığı politika ve güncellediği politikanın birbirinden farklı olmasıdır. Bu algoritmadaki amaç, en yüksek ödülü veren politikayı öğrenmektir.

Bu algoritma çalıştığında Q tablosu adı verilen bir eylem durum bilgilerini içinde bulunduran matris oluşturulur. Q tablosunun başlangıç değerleri sıfır olarak belirlenir. Q tablosunda başlangıçta sıfırlar bulunduğu için ajan ilk aşamada rastgele hareketler gerçekleştirecektir. İlk ödül bulunana kadar ajan rastgele hareket etmeye devam edecektir. Ajan tarafından gerçekleştirilen tüm eylemler ve durumlar tabloya kaydedilir. Q öğrenme algoritmasının akış şeması Şekil 5.7'de gösterilmiştir.

Q tablosu oluşturulduktan sonra eylem uzayından bir eylem seçilip uygulanır. Eylemin sonucuna göre ödül miktarı hesaplanır. Her eylem gerçekleştirildiğinde Q tablo değerleri güncellenir. En yüksek ödülün alındığı politikalar en iyi çözüm yolu olarak belirlenir. Q öğrenme algoritmasının eylem ve durumlara göre Q tablosunun güncelleme formülü (5.4)'te verilmiştir.

$$Q(S,A) = \alpha[R(S,A) + \gamma maxQ(S^{\iota}, A^{\iota})] \quad (5.4)$$

Denklemde yer alan A ajanın yapmış olduğu eylem, R eylem sonunda elde edilen ödülün değeri, S durumu, α öğrenme oranını, γ indirim oranını göstermektedir. γ değerinin azalması ödülün zamanla azalması anlamına gelmektedir. Tablo 5.1'de Q öğrenme algoritması verilmiştir.



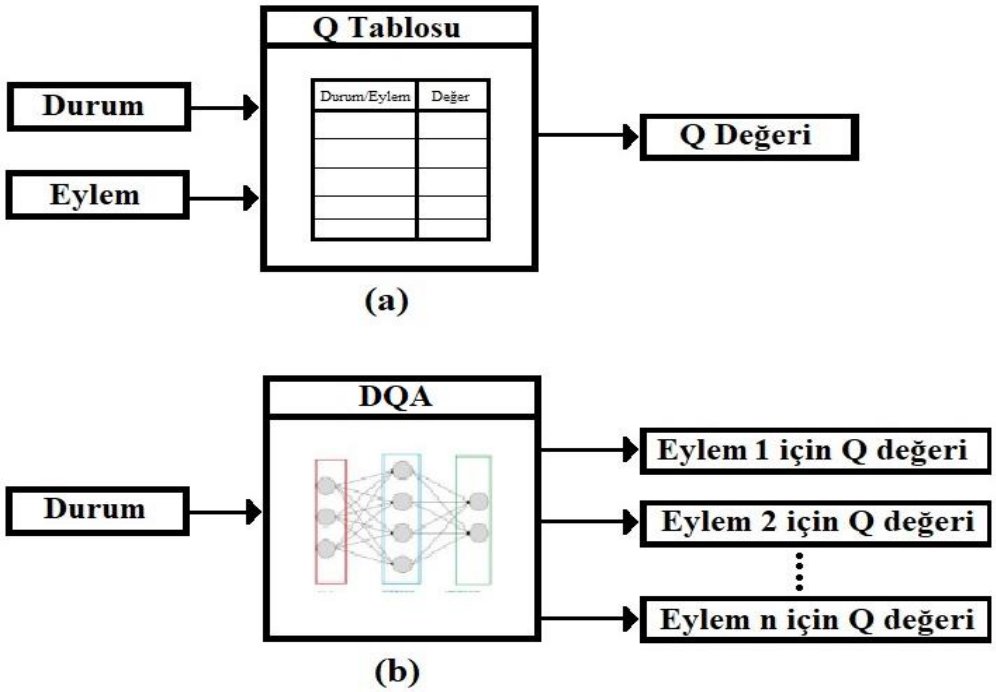
Şekil 5.7 Q öğrenme akış şeması

Tablo 5.1 Q öğrenme algoritması

Q öğrenme algoritması	
1:	Yeniden oynatma belleğini başlat(M)
2:	Ağ ağırlıklarını başlat
3:	**for** bölüm=1 **to** M **do**
4:	s0 başlangıç durumunu elde etmek için çevre ile etkileşim kurun
5:	Ajanın başlangıç durumu(A), Ajanın S durumu için yapacağı eylem(A)
6:	Rastgele bir A uygula
7:	**for** t=0 to T **do**
8:	A eylemini rastgele ve R ve S^1 kaydet
9:	$Q(S,A) = \alpha[R(S,A) + \gamma maxQ(S^\iota, A^\iota)]$ Q tablosunu güncelle
10:	S ← S^1
11:	**end for**
12:	**end for**

5.3.2 Derin q ağı

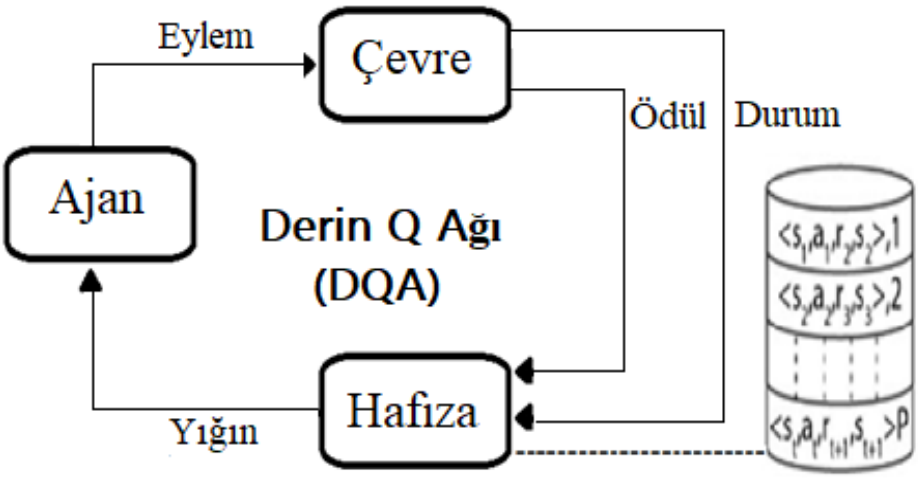
Derin Q Ağı (DQA), Q öğrenmede yapay sinir ağları ile kullanılan bir derin öğrenme algoritmasıdır. Derin Q ağı algoritması sürekli zamanlı politika dışı bir öğrenme yöntemidir.



Şekil 5.8 a) Q öğrenme algoritması b) DQA algoritması karşılaştırılması

Q öğrenmenin durum-eylem parametrelerinin tabloya kaydedilmesi prensibi ile çalıştığı anlatılmıştır. Durum-eylem parametrelerinin sayısı arttığında bunu tablo ile ifade etmek güçleşecektir. Karmaşık ve çok ölçekli durumlarda Q tablolarını kullanmak elverişli olmayacaktır. Bu tür durumlarda DQA kullanılmaktadır. Q-öğrenmede ajanın hafıza yapısı Q tablolarından oluşur. DQA de ise ajanın hafızası evrişimli sinir ağıdır. Şekil 5.8'de Q öğrenme ve DQA karşılaştırılması verilmiştir.

DQA sinir ağının girdisini durumlar, çıktısını ajanın gerçekleştirdiği eylemler oluşturmaktadır. DQA algoritmasında her bir keşifte elde edilen örnekler (s,a,r,s^1) hafızada tutulmaktadır. Buradaki s durum, a eylem, r ödül, s^1 ise sonraki durumdur. DQA algoritmasında öncelikle s durumunda a eylemi gerçekleştirilir. Gerçekleştirilen bu eylem sonucunda bir r ödülü kazanılır ve s^1 sonraki duruma geçilir. DQA algoritması deneyim tekrarı özelliğini kullanarak ajanın deneyimlerini hafızada tutmaktadır. Hafızada tutulan bu örneklemler alınarak ajanın deneyimleri üzerinden öğrenmesi sağlanmaktadır. DQA algoritmasının akış diyagramı Şekil 5.9'da gösterilmiştir.



Şekil 5.9 DQA akış diyagramı

Bu algoritmanın en önemli özelliği hatırlatma ve yeniden oynatma fonksiyonlarıdır. Ajan öğrenmiş olduğu deneyimleri belli bir süreden sonra unutma eğilimindedir. Ajanın yeniden eğitimi için öğrenmiş olduğu deneyimlere ihtiyaç duyulacaktır. DQA algoritması ile ajanın deneyimleri hafızada tutulur. Öğrenme esnasında bu veriler

rastgele örnekler ile ajanın eğitimi için kullanılır. Tablo 5.2'de DQA algoritması verilmiştir.

Tablo 5.2 DQA algoritması

DQA algoritması	
1:	Yeniden oynatma belleğini başlat(M)
2:	Ağ ağırlıklarını başlat
3:	**for** bölüm=1 **to** M **do**
4:	s0 başlangıç durumunu elde etmek için çevre ile etkileşim kurun
5:	**for** t=0 to T **do**
6:	Rastgele bir a_t eylemlerini elde etmek için ε stratejisini kullan
7:	a_t eylemi gerçekleştir ve s_t+1 durumuna ilerle ve mevcut ödül r_t'yi hesapla.
8:	(st,at,st+1,rt) değerlerini M'ye kaydet
9:	$L(\theta) = [Q(s,a) - (r + \gamma maxQ(s^\iota, a^\iota)]^2$ kayıp değerini hesapla.
10:	**end for**
11:	**end for**

Algoritmada ajanın tahmininin gerçek hedeften ne kadar uzakta olduğunu veren değer kayıp olarak ifade edilmektedir. Burada tahmini, hedeften çıkararak kayıp değeri bulunur. DQA algoritmasında kayıp fonksiyonunu (5.5)'de tanımlanmıştır.

$$L(\theta) = [Q(s,a) - (r + \gamma maxQ(s^\iota, a^\iota)]^2 \quad (5.5)$$

Burada a ajanın yapmış olduğu eylemi, r eylem sonunda elde edilen ödülün değerini, s durumu, γ indirim oranını göstermektedir.

Yıkıcı bir girdi uygulandığında, sistemin zaman içinde yaklaşık denge noktasına yaklaşması durumunda asimptotik olarak kararlı olduğu söylenir. Çalışmalarında, Fan ve ark. Q-yineleme algoritmasının basitleştirilmiş bir sürümünü inceleyerek DQA algoritmasının yakınsamasını oluşturmayı önermişlerdir. Bu algoritma ile DQA arasındaki fark, durum-eylem-ödül-durum kümelerini depolamak için yinelenen arabellek M olmamasıdır. Q-yineleme algoritmasının yakınsamasını daha iyi anlamak için, önce bazı önemli sinir ağları tanımlanmalıdır [57-58]. DDB ağları şu şekilde tanımlanır:

$$W_{L,s,dj} = \left\{ \text{f NN with DDB: } \max_{l \in \{0, \ldots, L+1\}} \|\vec{\theta}_l\|_\infty \leq 1, \sum_{l=0}^{L+1} \|\vec{\theta}_l\|_0 \leq s, \max_{i \in \{d_0, \ldots, d_{L+1}\}} \|f_i\|_\infty \leq V \right\}. \quad (5.6)$$

Burada L, gizli katmanların sayısıdır, s seyrekliktir, dj, j-th katmanının genişliğidir, V ∈ (0, ∞) ve $\vec{\theta}_l$, l katmanının tüm ağırlıkları ve önyargılarıdır. Burada, maksimum ağırlığı 1 ve önemli nöronların sayısını s olarak sınırlıyoruz. Hölder fonksiyonları,

$$W_{L,s,dj} = \left\{ f: C \to R : \sum_{\alpha:|\alpha|<\beta} \|\partial^\alpha f\|_\infty + \sum_{\alpha:|\alpha|_1=\lfloor\beta\rfloor} \sup_{x,y \in C \atop x \neq y} \frac{|\partial^\alpha f(x) - \partial^\alpha(y)|}{\|x-y\|_\infty^{\beta-\lfloor\beta\rfloor}} \leq K \right\}. \quad (5.7)$$

Burada β, K > 0 $\partial^\alpha = \prod_{i=1}^{n} \partial^{\alpha i}$. G ile q Hölder fonksiyonlarının tüm geçerli kompozisyonlarını göstereceğiz. Hölder fonksiyonları $\widehat{G_q}$ ve genişletilmiş seyrek DDB ağları $\widehat{W_{L,s,dJ}}$olarak tanımlanır:

$$\widehat{G_q} = \{f: S\ X\ A : \forall \alpha \in A, f(.,\alpha) \in G_q\}. \quad (5.8)$$

$$\widehat{W_{L,s,dJ}} = \{f: S\ X\ A : \forall \alpha \in A, f(.,\alpha) \in W_{L,s,dj}\}. \quad (5.9)$$

Q-yineleme algoritmasının geçerli olması için iki varsayım yapılmıştır. Bunlar;

- Her f ∈ $W_{L,s,dj}$, sahip olunmalı,

$$r(s,a) + \gamma E\left[\max_{a' \in A} Q(S',a) | S' \sim P(s,a)\right] \in Gq \quad (5.10)$$

- ν1, ν2 ∈ D[S × A] iki mutlak sürekli olasılık ölçüsü ve μ = (μ1,· · · , μi ,· · · ·) markov karar süreci için bir politika olsun. Dentone $P_T^\mu V_1 = P^{\mu_T} \ldots P^{\mu_1 V_1}$ (S0, A0) ∼ ν1'e sahip markov karar sürecinin {(Si, Ai)}i=1,...,T durumlarının olasılık dağılımı.

π_t, Q-yineleme algoritmasının t adımındaki açgözlü politikası olsun. Öyle bir C ∈ R^+ sabiti vardır ki,

$$\|Q^* - Q^{\pi_t}\|_{1,\nu} \leq \frac{C}{(1-\gamma)^2} \emptyset(\nu,\sigma)\gamma|A|(\ln n)^{1+\frac{2\xi_n(\alpha-1)}{2}} + \frac{4\gamma^{t+1}R_{max}}{(1-\gamma)^2}. \quad (5.11)$$

α aşağıdaki denklem ile hesaplanır,

$$\alpha = max_j = 1, \dots, q \frac{t_j}{2\beta_j \prod_{l=1}^{j+1} min(1, \beta_l) + t_j} \ . \quad (5.12)$$

Burada; n örneklem boyutu, β_j q Hölder fonksiyonlarının bileşiminde j-th Hölder fonksiyonlarını tanımlayan sabit, tj bir j-th Hölder fonksiyonunun bağlı olduğu ve ξ 'nin karşıladığı maksimum giriş değişkeni sayısıdır.

$$max\left\{\sum_{j=1}^{q}(t_j + \beta_j + 1)^{3+t_j}, \sum_{j=1}^{q}\ln(t_j + \beta_j), \max_{j \in \{1, \dots q\}} P_j \right\} \leq \tilde{C}(\log n)^{\frac{\xi-1}{2}} \ . \quad (5.13)$$

Burada iki terim şu anlama gelir: birincisi, istatistiksel bir hatadır,

$$\frac{C}{(1-\gamma)^2} \emptyset\ (v, \sigma)\gamma|A|(\ln n)^{1+\frac{2\xi_n(\alpha-1)}{2}} \ , \quad (5.14)$$

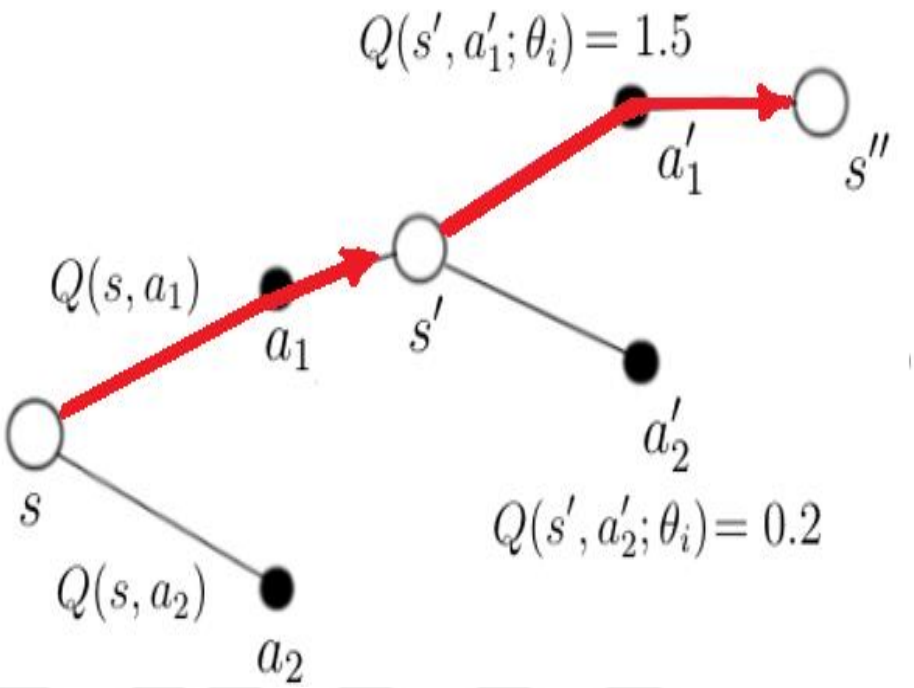
ikincisi, algoritmik bir hatadır,

$$\frac{4\gamma^{t+1}R_{max}}{(1-\gamma)^2} \ . \quad (5.15)$$

5.3.3 Çift derin q ağı

DQA algoritmasında bazı eylemler daha değerli olduğu kabul edilir. Bu tür durumlarda sürekli aynı eylemler seçilir. Belirli durumlarda sürekli aynı eylemlerin seçilmesi aşırı öğrenme durumu denilen overfitting olayına sebep olmaktadır. Bu durumda ajanın en iyi politikayı öğrenmesi olumsuz etkilenir. Aşırı öğrenme probleminin önüne geçmek için Çift Derin Q Ağları (ÇDQA) geliştirilmiştir.

ÇDQA'ları özdeş iki sinir ağı modeli kullanır. Bu sinir ağından biri, DQA'nın yaptığı gibi deneyim tekrarı sırasında öğrenir, diğeri ise ilk modelin son bölümünün bir kopyasıdır. Burada Q değeri ikinci modelle hesaplanır. Şekil 5.10'da ÇDQA algoritmasının akış diyagramı gösterilmiştir.



Şekil 5.10 ÇDQA akış diyagramı

DQA algoritmasında eylemin seçilmesi ve değerlendirilmesi aynı yöntemle gerçekleştirilir. Bu işlem (5.16)'da gösterildiği gibidir.

$$max_A Q(S_{t+1}, A) \tag{5.16}$$

ÇDQA algoritmasında eylemin seçilmesi ve değerlendirilmesi işlemleri birbirinden ayrı şekilde gerçekleştirilir. ÇDQA algoritmasında bu işlemler (5.17)'de gösterilmiştir.

$$Q(S^{\iota}, argmax_A Q(S, A;\ \theta_i), \theta_i) \tag{5.17}$$

Denklemde yer alan θ_i değeri ağırlık değerini göstermektedir. ÇDQA algoritmasında her bir eylem için Q(A) ve Q(B) olmak üzere iki farklı Q değeri hesaplanır. Bu algoritmada öncelikle sonraki durum olan S^{ι} için en yüksek Q değerini veren eylem tespit edilir. Ardından bu eylem ikinci Q değerini bulmak için kullanılır. Bu işlemlerden sonra Q(B) değeri ile Q(A) değeri güncellenir. ÇDQA algoritmasında kayıp fonksiyonunu (5.18)'de tanımlanmıştır.

$$\mathrm{L}(\theta) = [r_{i+1} + \gamma Q^T(s_{t+1}. argmax Q^E(s_{t+1}, a^{\iota})] \tag{5.18}$$

Ana modelden en yüksek Q değerinin indeksi bulunur ve bu indeks ikinci modelden eylemi elde etmek için kullanılmaktadır. Q^E ve Q^T, sırasıyla değerlendirme ağını ve hedef ağını temsil etmektedir. ÇDQA algoritmasının Tablo 5.3'de verilmiştir.

Tablo 5.3 ÇDQA algoritması

ÇDQA algoritması	
1:	Yeniden oynatma belleğini başlat(M)
2:	Ağ ağırlıklarını başlat
3:	Hedef ağı $Q^{T}(\theta)= Q^{E}(\theta)$ olarak başlat
4:	Hedef ağ eşitleme sayacı(C)
5:	**for** bölüm=1 **to** M **do**
6:	s0 başlangıç durumunu elde etmek için çevre ile etkileşim kurun
7:	**for** t=0 to T **do**
8:	Rastgele bir a_t eylemlerini elde etmek için ε stratejisini kullan
9:	a_t eylemi gerçekleştir ve s_t+1 durumuna ilerle ve mevcut ödül r_t'yi hesapla.
10:	(st,at,st+1,rt) değerlerini M'ye kaydet
11:	L(θ) $= r_{i+1} + \gamma Q^{T}(s_{t+1}. argmax Q^{E}(s_{t+1}, a^{\iota})]$ kayıp değerini hesapla.
12:	**if** mod(T,C)==0 **then**
13:	$Q^{T}(\theta)= Q^{E}(\theta)$
14:	**end if**
15:	**end for**
16:	**end for**

Bu algoritmanın yararı, gerçek Q(s, a) değerlerinin daha iyi bir tahminidir. Bazı tarafsız tahminler göz önüne alındığında, $Q_t(s,a)$ şöyle gösterilmiştir:

$$\sum_{a}\left(\mathrm{Q_t(s,a)} - \mathrm{V^*(s)}\right) = 0, \quad \frac{1}{\mathrm{N}}\sum_{\alpha}\left(\mathrm{Q_t(s,a)} - \mathrm{V^*(s)}\right)^2 = \mathrm{k} > 0 \ , \tag{5.19}$$

$Q_t(s,a)$ tahminlerinin katı bir alt sınırı vardır,

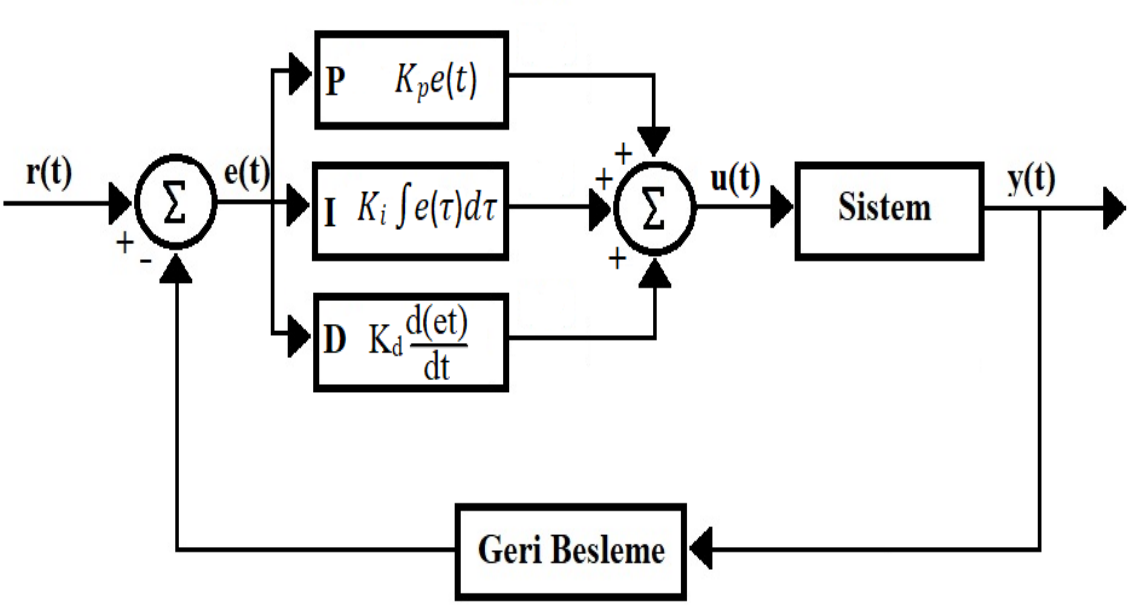
$$\max_{\alpha}\mathrm{Q_t(s,a)} - \mathrm{V^*(s)} \geq \sqrt{\frac{\mathrm{k}}{\mathrm{N}-1}} \ , \tag{5.20}$$

ÇDQA tahminleri sıfır değerli bir alt sınıra sahipken (5.21)'de verildiği gibi hesaplanır.

$$\left| Q_t^A\left(s,\ argmax_a Q_t^B(s,a)\right) - V^*(s) \geq 0 \right| . \tag{5.21}$$

5.4 PID Kontrolör

PID, oransal, integral ve türev elemanlarını toplamıyla oluşan bir kontrolördür. PID kontrolör, giriş sinyali ve geri besleme ile girişe gönderilen sinyali karşılaştırarak hatayı hesaplayan bir kontrolördür. PID kontrolörde amaç zaman içerisinde oluşan hatayı en aza indirmektir. Hatayı minimize etmek için PID parametrelerinin iyi bir şekilde ayarlanması gerekmektedir. Sistemin ihtiyacına göre P, I, PI, PD, PID kontrolör olarak ayarlanabilmektedir. PID kontrolör uzun zamandan beri yaygın olarak kullanılan bir kontrolördür. Basit ve anlaşılır yapısı sayesinde endüstride birçok sistemin kontrolünde tercih edilmektedir. PID kontrolörün blok diyagramı Şekil 5.11'de verilmiştir.



Şekil 5.11 PID kontrolör blok diyagramı

PID kontrolörler hatanın sıfır olabilmesi için üç parametre kullanmaktadır. Bu parametreler K_p, K_i, K_d şeklindedir. Şekilde ifade edilen r(t) referans sinyali, e(t) hata fonksiyonu, u(t) sistemin girişi ve y(t) sistemin çıkışıdır. Blok diyagramında da görüldüğü gibi hata fonksiyonu (5.22)'da verilmiştir.

$$e(t) = r(t) - y(t) \tag{5.22}$$

Sistemin hata değeri, r(t) referans giriş değerinden y(t) sistemin çıkış değeri çıkarılarak bulunur. Sistemlerde e(t) fonksiyonunun sıfır olması istenmektedir. PID kontrolör matematiksel olarak oransal integral ve türevin toplamından oluşmaktadır. (5.23)'da PID kontrolörün matematiksel ifadesi verilmiştir.

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \quad (5.23)$$

(5.23)'te yer alan K_p oransal kazanç katsayısı, K_i integral kazanç sabiti, K_d türev kazanç sabitidir. Denklemde yer alan t zamanı ifade etmektedir. e(t) fonksiyonu ise hata fonksiyonudur. PID kontrolördeki oransal terimi, tüm geçişlerin kazanç faktörü yoluyla hata sinyali ile orantılı olarak kontrol edilmesini sağlar. İntegral terimi düşük frekanslar için integratör ayarıyla kararlı durum hatalarını azaltır. Türev terimi ise yüksek frekanslı farklılaştırıcı telafisi yoluyla geçici yanıtı iyileştirir. Bir PID kontrolörde girişe uygulanan sinyal bu üç terime bağlı olarak çıktı üretir.

Oransal kontrolör, sistemin kontrol işaretinin çıkışa sabit bir oran ile aktarılmasını sağlamaktadır. Oransal kontrolörde hata bir K_p katsayısı ile çarpılarak oluşturulur. Oransal kontrol (5.24)'de verilmiştir.

$$u(t) = K_p e(t) \quad (5.24)$$

İntegral kontrolör, hatayı sıfır değerine ulaştırıncaya kadar çıkış değerini sürekli arttırıp azaltır. İşlem bu şekilde belli bir süre devam ettirilince hata sıfır olmaktadır. T_i katsayısı integral zaman sabitidir. İntegral ifadesinin oransal ifadeye ulaşması için geçen süreyi ifade etmektedir. İntegral kontrolör (5.25)'de verilmiştir.

$$I = \frac{K}{T_i} \int_0^t e(t)dt \quad (5.25)$$

Türev kontrolör, türev elemanının zamana bağlı değerlerinin değişimi ile ilgilidir. Sistemde gerçekleşen değişimden kaynaklı, istenilen değerin üstüne çıkmayı engellemek için sistemi yavaşlatmayı amaçlamaktadır. (5.26)'te türev kontrolör eşitliği verilmiştir. Denklemde yer alan T_d türev zaman sabitidir.

$$u(t) = K\left(e(t) + T_d \frac{de(t)}{dt}\right) \quad (5.26)$$

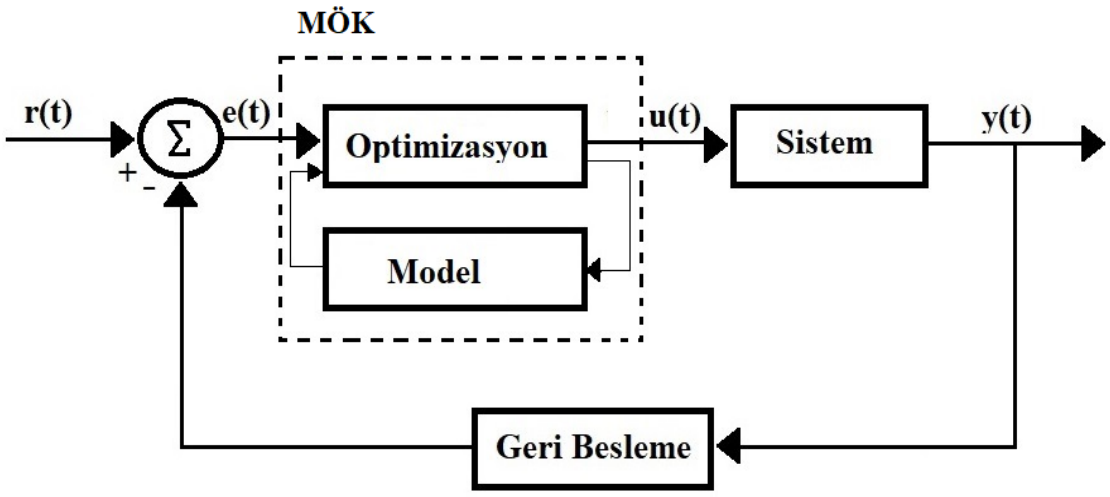
PID kontrolörün düzgün sonuçlar vermesi için katsayıların düzgün olarak seçilmesi gerekmektedir. Bu katsayıların hesaplanması için farklı yöntemler bulunmaktadır. Wang-Juan-Chan, Chien Hroners Reswik, Ziegler-Nichols gibi klasik yöntemler parametrelerin hesaplanmasında kullanılmaktadır [59].

5.5 Model Öngörülü Kontrol

Model Öngörülü Kontrol (MÖK) sistemin gelecekteki hareketini tahmin etmek için bir süreç model kullanan gelişmiş bir kontrol yöntemidir. MÖK kontrolörün temel amacı, sistemin sonraki hareketini sınırlı zaman içerisinde tahmin etmek ve sistem kısıtlarını karşılayarak en az maliyetle en uygun kontrol girdisini hesaplamaktır. Oluşturulan sistemde tahmini hesaba katan MÖK yöntemi, kısıtlı bir optimizasyon problemini çözerek optimal bir çıkış oluşturur.

MÖK, gelecekteki olayları öngörme yeteneğine sahiptir ve buna göre kontrol önlemleri alabilir. PID kontrolörleri bu tahmin yeteneğine sahip değildir. MÖK, özel olarak tasarlanmış analog devrelerle daha hızlı yanıt süreleri elde etmeye yönelik araştırmalar olmasına rağmen, neredeyse evrensel olarak bir dijital kontrol olarak uygulanmaktadır. MÖK, geleneksel geri besleme kontrol sistemleri tarafından kontrol edilemeyen sistemleri bile kontrol edebilir.

PID kontrolörler yalnızca geçmiş ve mevcut sistem davranışlarını dikkate alırlar. Klasik bir kapalı çevrim sistem blok diyagramı Şekil 5.2 de verilmiştir. Tahmine dayalı olarak çalışan MÖK kontrolör sapmaları öngörerek gelecekteki davranışları tahmin edip davranışlarını belirler. MÖK kontrolör blok diyagramı Şekil 5.12'de verilmiştir.



Şekil 5.12 MÖK kontrolör blok diyagramı

MÖK kontrolör başlangıç durumundan optimal bir yörünge çıkarmak için bir yörünge optimizasyon tekniği kullanır. Model ve gerçek sistem biraz farklı olacağından, gerçek

sistem tahmin edilen yörüngeden sapacaktır. Bu nedenle, tahmin edilen optimum yörüngeyi güncellemek için yörünge optimizasyonu periyodik olarak çevrimiçi olarak yeniden çalıştırmaktadır. MÖK'ün PID'ye göre avantajı gelecekteki kısıtlamaların dikkate almasıdır. Algoritma geleceğe giden bir yörüngeyi optimize ettiğinden, gelecekteki kısıtlamaları ihlal etmekten kaçınmak için uygun eylemleri tahmin edebilmektedir.

Bir dizi kontrol sinyali uygulandığında; $\overline{U}$ doğrusal bir model, $X_{t+1} = AX_t + BU_t$ durum dizisi, $\overline{X}$ bir sonraki N zaman adımı,

$$\overline{X} = \overline{A}X_t + \overline{BU}\,, \tag{5.27}$$

(5.27) mevcut durum olan X_t ile eşleşen basit bir fonksiyondur. $\overline{X}$ bir yörünge $\overline{U}$ yörünge üzerindeki eylemlerdir.

$$\overline{X} = (X_{t+1}^T, \dots, X_{t+N}^T)^T\,, \tag{5.28}$$

$$\overline{U} = (U_{t+1}^T, \dots, U_{t+N-1}^T)^T\,. \tag{5.29}$$

$\overline{A}$ ve $\overline{B}$ matris formları sırasıyla (5.30) ve (5.31)'de verilmiştir.

$$\overline{A} = \begin{bmatrix} A \\ A^2 \\ . \\ . \\ . \\ A^{N-1} \\ A^N \end{bmatrix}, \tag{5.30}$$

$$\overline{B} = \begin{bmatrix} A & 0 & \dots & \dots & 0 \\ AB & B & 0 & \dots & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ A^{N-2}B & A^{N-3}B & \dots & B & 0 \\ A^{N-1}B & A^{N-2}B & \dots & AB & B \end{bmatrix}. \tag{5.31}$$

En iyi yörüngeyi belirlemek için bir maliyet fonksiyonu oluşturulur. Maliyet işlevi, yörüngeyi puanlayan skaler bir işlevdir. MÖK kontrolörler tipik olarak $\overline{X}$ ve $\overline{U}$ öğeleri kullanarak maliyet fonksiyonunun hesaplanması (5.32)'de verilmiştir.

$$J = \frac{1}{2}\,\overline{X}^{T}\overline{Q}\,\overline{X} + \frac{1}{2}\,\overline{U}^{T}\overline{R}\,\overline{U}. \tag{5.32}$$

(5.32)'de belirtilen denklemin içine (5.27)'deki denklem eklenerek maliyet $\overline{U}$'nun bir fonksiyonu olarak (5.33)'de gösterilmiştir.

$$J = \frac{1}{2}\,\overline{U}^{T}\left(\overline{R} + \overline{B}^{T}\overline{Q}\,\overline{B}\right)\overline{U} + X_t^{T}\overline{A}^{T}\overline{Q}\,\overline{B}\,\overline{U} + \frac{1}{2}X_t^{T}\,\overline{A}^{T}\overline{Q}\,\overline{A}\,X_t. \tag{5.33}$$

(5.33)'ü basit bir formda göstermek gerekirse;

$$J = \frac{1}{2}\,\overline{U}^{T}H\overline{U} + f^{T}\overline{U} + J_0 \tag{5.34}$$

Burada yer alan H, f^{T}, J_0 sırasıyla şöyledir:

$$H = \overline{R} + \overline{B}^{T}\overline{Q}\,\overline{B}, \tag{5.35}$$

$$f^{T}, = X_t^{T}\overline{A}^{T}\overline{Q}\,\overline{B}\,, \tag{5.36}$$

$$J_0 = \frac{1}{2}X_t^{T}\,\overline{A}^{T}\overline{Q}\,\overline{A}\,X_t. \tag{5.37}$$

J_0 teriminin $\overline{U}$'ya göre sabit olduğu ve minimum J'nin konumu üzerinde hiçbir etkisinin olmadığına dikkat edilmelidir. Kısıtlamalar olmadan en uygun yörünge bulunabilir. (5.34)'de yer alan $\overline{U}$ yerine, sonucu sıfıra ayarlamak ve en uygun yörüngeyi çözmek için $\overline{U}^{*}$:

$$\overline{U}^{*} = -H^{-1}f\,. \tag{5.38}$$

Bununla birlikte, kısıtlamalar genellikle sisteme eşitlik kısıtlamaları şeklinde uygulanır,

$$C_{eq}\overline{U} = b_{eq}\,. \tag{5.39}$$

Veya eşitsizlik kısıtlamaları (5.40)'daki gibi ifade edilir:

$$C_{in}\overline{U} \leq b_{in}\,. \tag{5.40}$$

Formun durum kısıtlamaları (5.41)'de ifade edilmiştir.

$$C_{in}\overline{X} \leq b_{in} . \quad (5.41)$$

Ayrıca (5.27) kullanılarak $\overline{U}$ fonksiyonlarına dönüştürülerek de uygulanabilir.

$$C_{in}\overline{B}\,\overline{U} \leq b_{in} - C_{in}\overline{A}\,X_t. \quad (5.42)$$

Kısıtlamaların varlığında, bir programlama çözücü gereklidir. Bu çözücüler, kısıtlamaları eşzamanlı olarak karşılayan en uygun sonucu bulmak için daha basit alt problemleri yinelemeli olarak çözmektedirler.

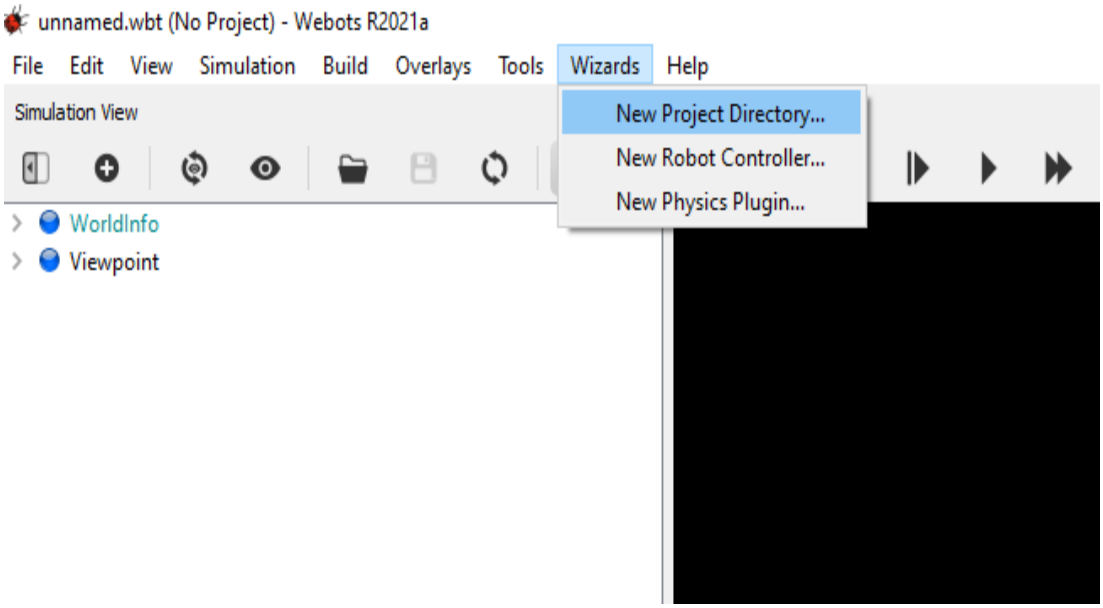
6. TEZİN UYGULAMA AŞAMASI

Bu bölümde iki ayaklı insansı robotlara gelen itmelere karşı dengede durabilmelerini sağlayan kontrol yöntemleri uygulanmıştır. Önceki bölümlerde verilen bilgiler doğrultusunda tasarladığımız sistemin genel yapısı çıkarılmıştır. Yapılan uygulamada hem simülâsyon hem de gerçek dünya testleri gerçekleştirilmiştir.

Simülâsyon ortamı olarak Webots robot geliştirme simülâtör aracı kullanılmıştır. Testlerin gerçekleştirdiği insansı robot Robotis-OP2'dir. Robotu eğitmek için PD kontrolör, MÖK kontrol yöntemi ve derin pekiştirmeli öğrenme algoritmalarından DQA ve ÇDQA yöntemleri kullanılmıştır. Uygulamaları gerçekleştirmek için Python ve C++ programlama dilleri kullanılmıştır.

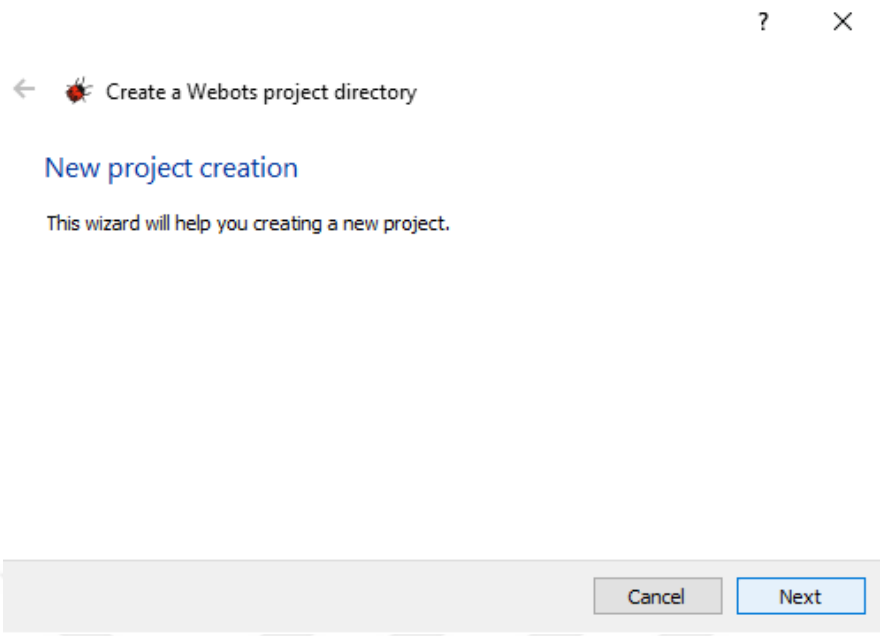
6.1 Yeni Proje Oluşturma

Öncelikle Webots simülâtöründe gerçek dünyaya benzer bir çevre oluşturulmalı ve Robotis-OP2 çevreye entegre edilmesi gerekmektedir. Yeni bir proje oluşturmak için Webots programını açtıktan sonra menü sekmesinde bulunan Wizards->New Project Directory yolu takip edilir.



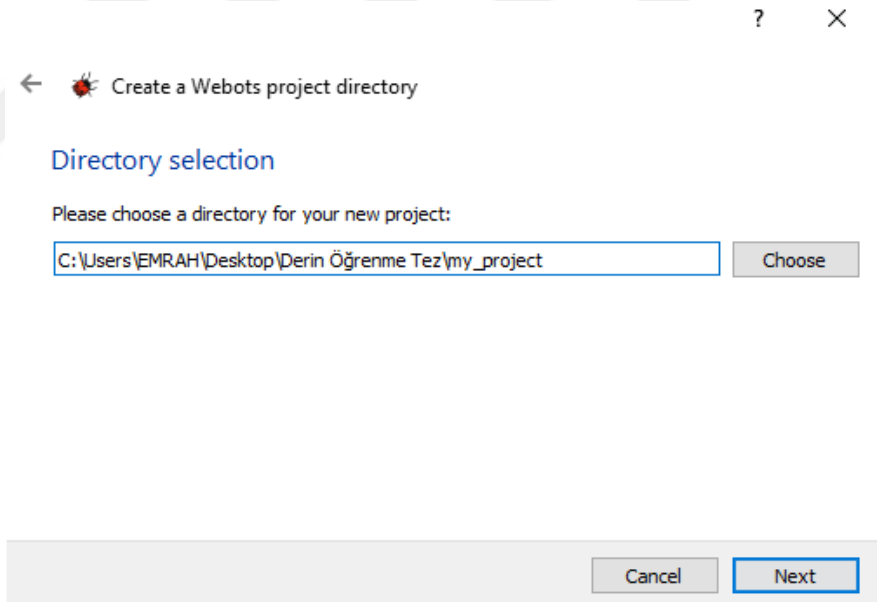
Şekil 6.1 Webots ortamında yeni proje dosyası oluşturma-1

Ardından açılan pencerede Next tuşuna basılır.



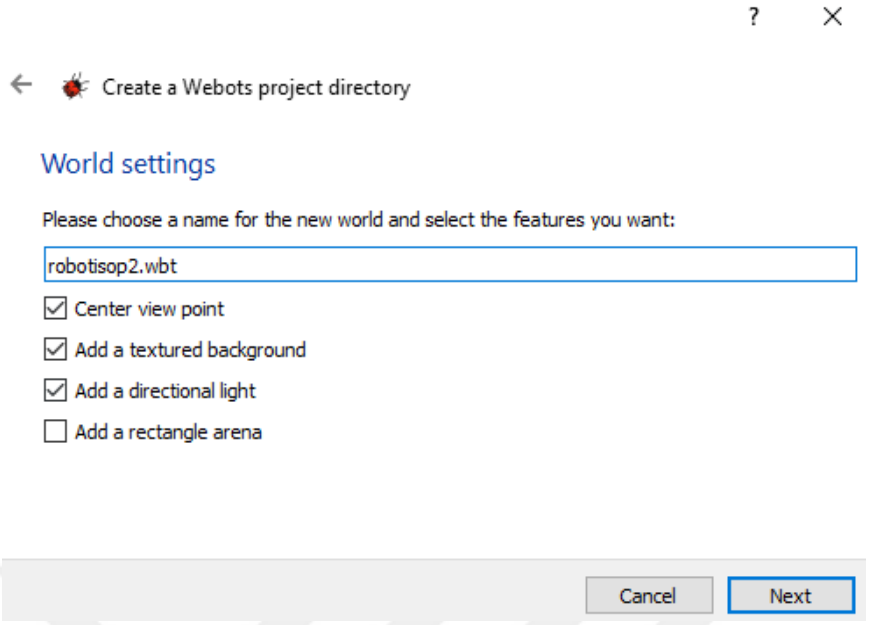
Şekil 6.2 Webots ortamında yeni proje dosyası oluşturma-2

Açılan pencerede dosya konumu belirtilip Next tuşuna basılır.



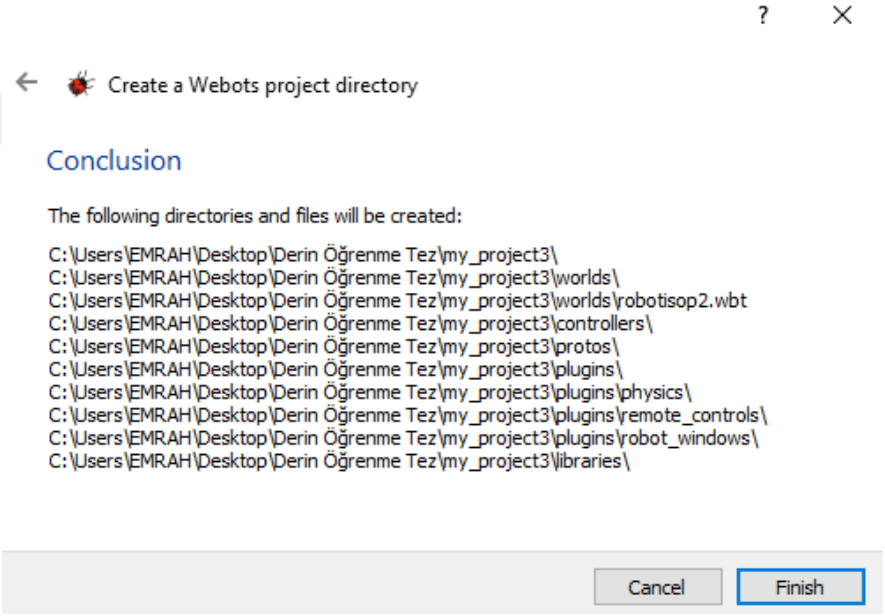
Şekil 6.3 Webots ortamında yeni proje dosyası oluşturma-3

Projeye boş bir Webots ortamı eklemek gerekmektedir. Burada projeye eklenen ortama bir isim verip '.wbt' uzantılı olarak kaydedilmelidir. Gelen ekranda alt kısımda bulunan 'Add a rectangle area' kutucuğu seçilirse boş bir kare alan proje ile birlikte oluşturulacaktır. Bu alanı seçme zorunluluğu bulunmamaktadır. Ardından Next tuşuna basılır.



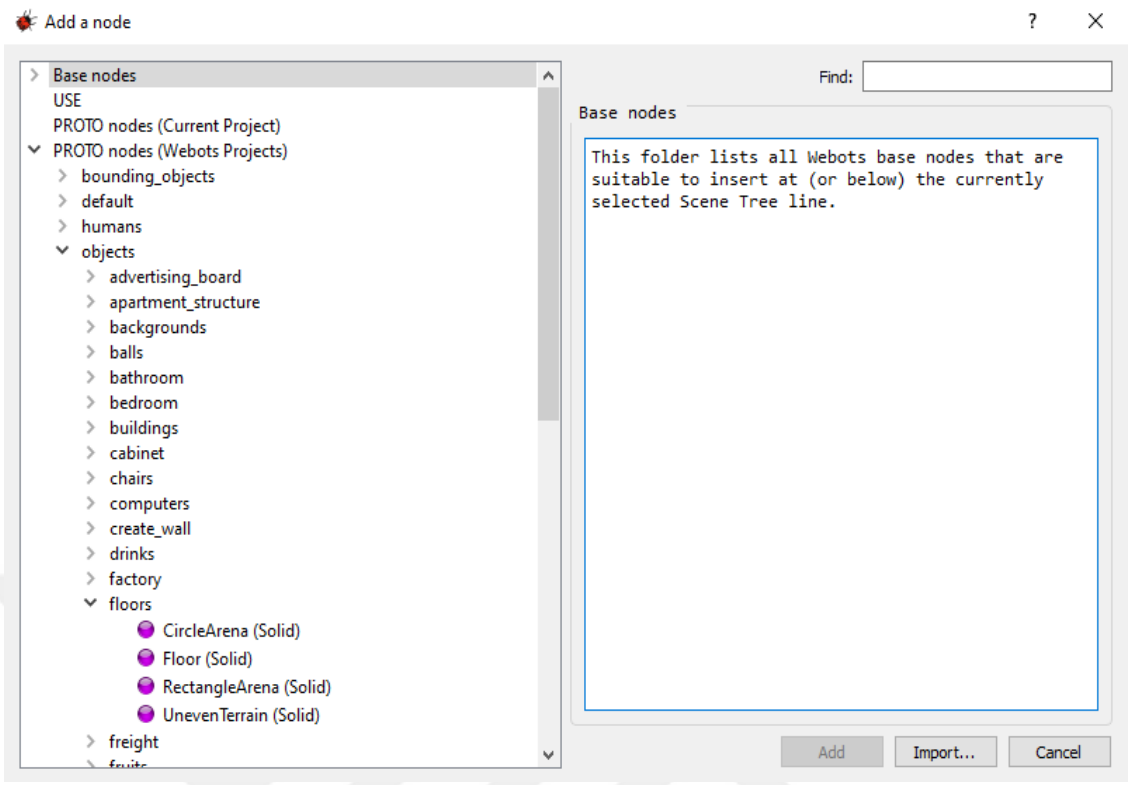
Şekil 6.4 Webots ortamında yeni proje dosyası oluşturma-4

Son olarak çıkan pencerede Finish tuşuna basarak yeni proje oluşturulur.



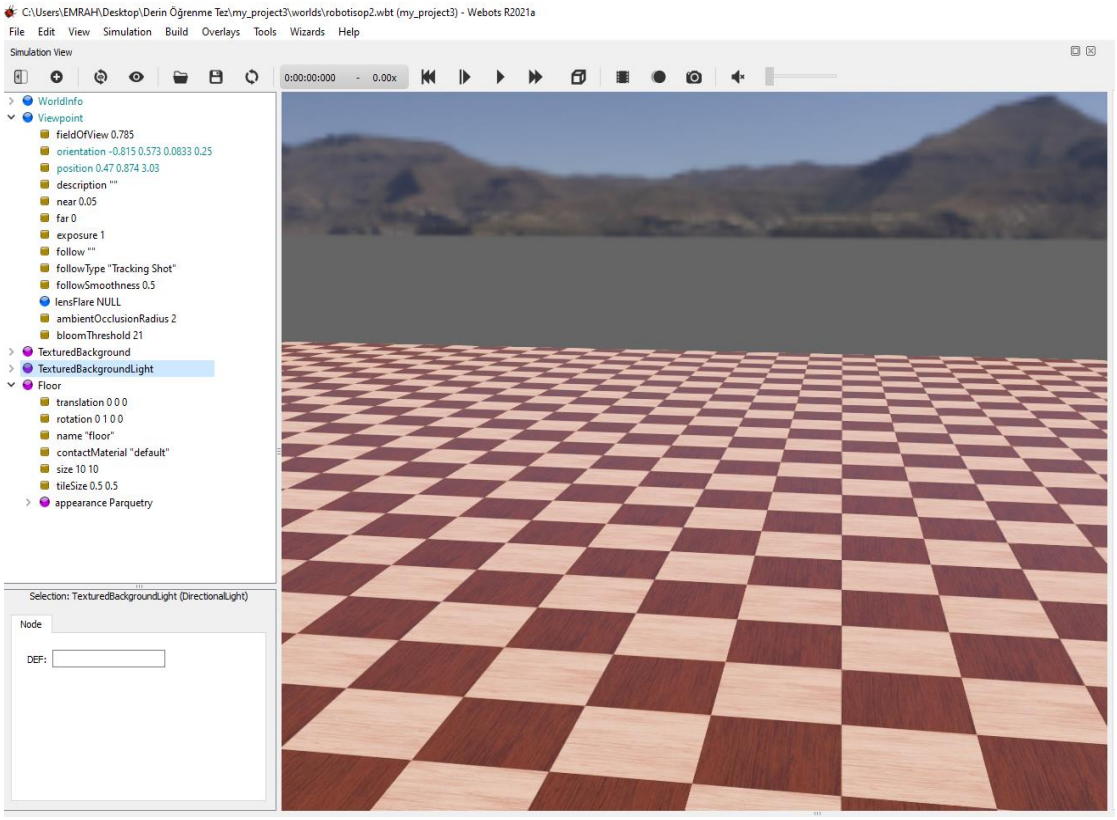
Şekil 6.5 Webots ortamında yeni proje dosyası oluşturma-5

Proje oluşturulduğunda ‘Add a rectangle area’ kutucuğunu seçilmediyse uygun bir zemin alanı oluşturmak gerekmektedir. Bunun için Webots ana ekranının menü kısmının hemen altında bulunan (+) tuşuna basıp bir zemin seçmek gerekmektedir. Sırasıyla PROTO nodes (WebotsProjects)->objects-> floors seçenekleri seçilerek istenilen zemin alanı projeye eklenir.



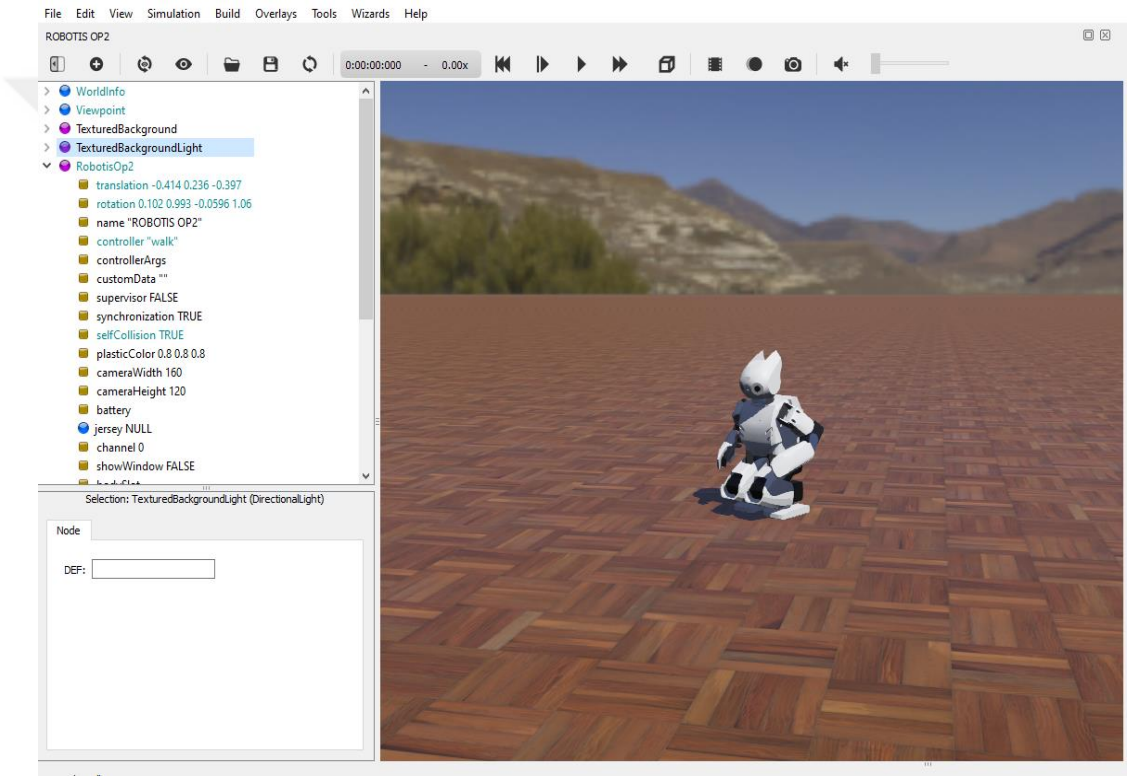
Şekil 6.6 Webots ortamına zemin ekleme

Projeye zemin eklendikten sonra ekran görüntüsü Şekil 6.7 de görüldüğü gibi olacaktır.



Şekil 6.7 Webots simülâtör ortamı

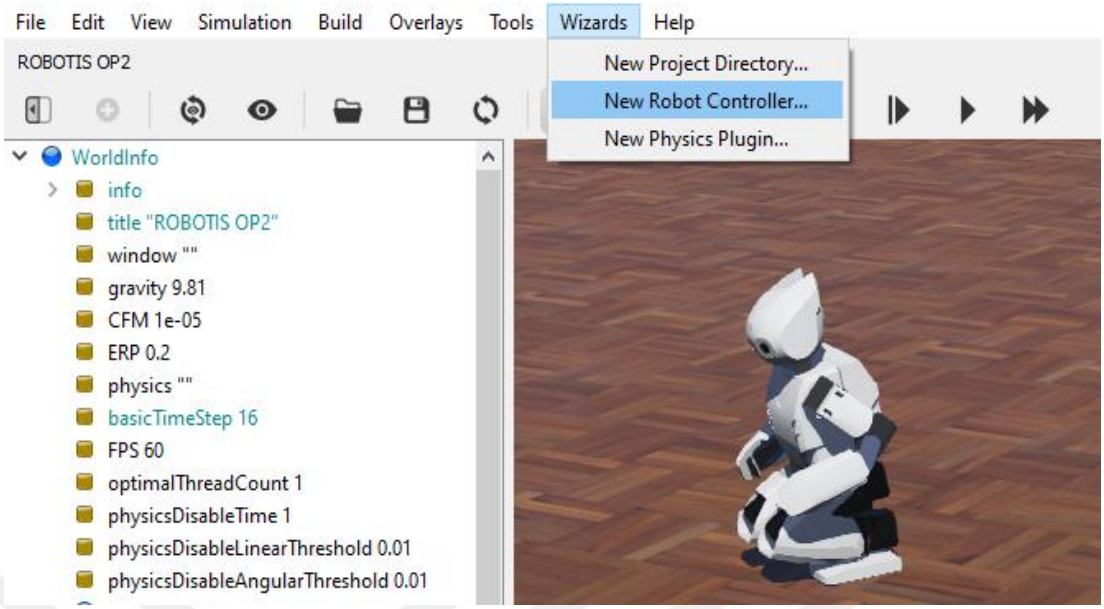
Sol tarafta bulunan sekmede projeye eklenen tüm nesneler görülmektedir. Sol alt kısımda kalan ‘selection’ kısmında seçilen nesnelerin özellikleri veya aldıkları değerler görülmektedir. Sağ kısımda kalan bölüm Webots ortamıdır. Bu ortama istenilen nesneleri ekleyerek gerçek dünya görünümüne yakın ortamlar oluşturulabilir. Yapılacak projeye uygun olarak ortamlar, nesneler eklenmelidir. PROTO nodes (WebotsProjects)->robots->robotis->darwin-op->RobotisOp2 yolu izlenerek Robotis-OP2 projeye dâhil edilir. Zemin ayarı yapılıp, Robotis-OP2 projeye dâhil edildikten sonraki görüntü Şekil 6.8’de verilmiştir.



Şekil 6.8 Webots simülâtör ortamında Robotis OP2

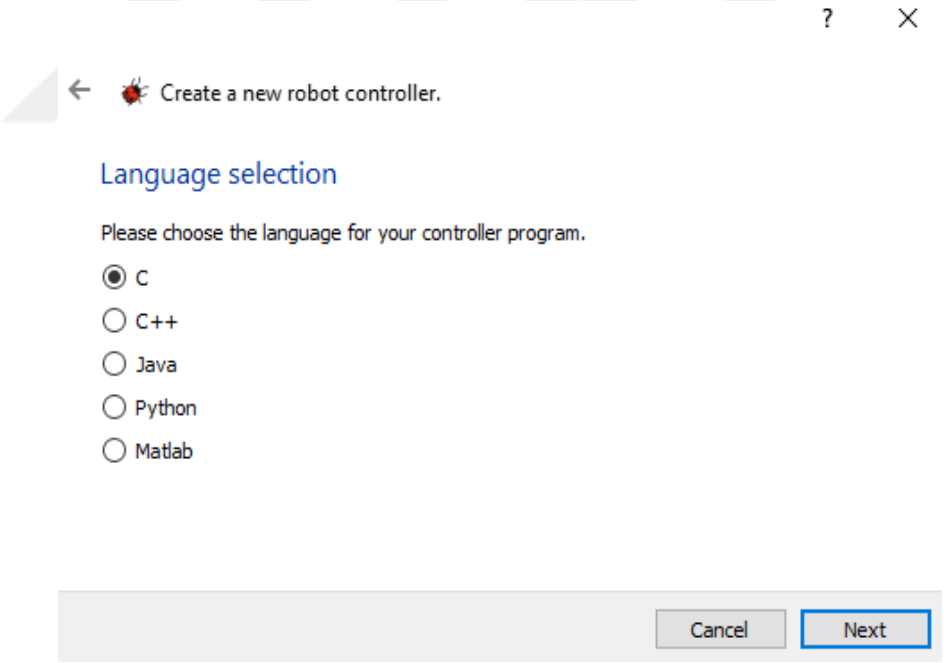
6.2 Projeye Kontrolör Ekleme

Webots ortamına gerekli nesneler eklendikten sonra yapılması gereken bir kontrol dosyası oluşturmaktır. Bu kontrol dosyası sayesinde eklenen nesnelere istenen işlevler yaptırılabilmektedir. Kontrol dosyası oluşturmak için menü sekmesinde bulunan Wizards->New Robot Controller yolu takip edilir. Böylece yeni bir robot kontrolü oluşturulur. Gelen ekranda Next tuşuna basıp ilerlenir.



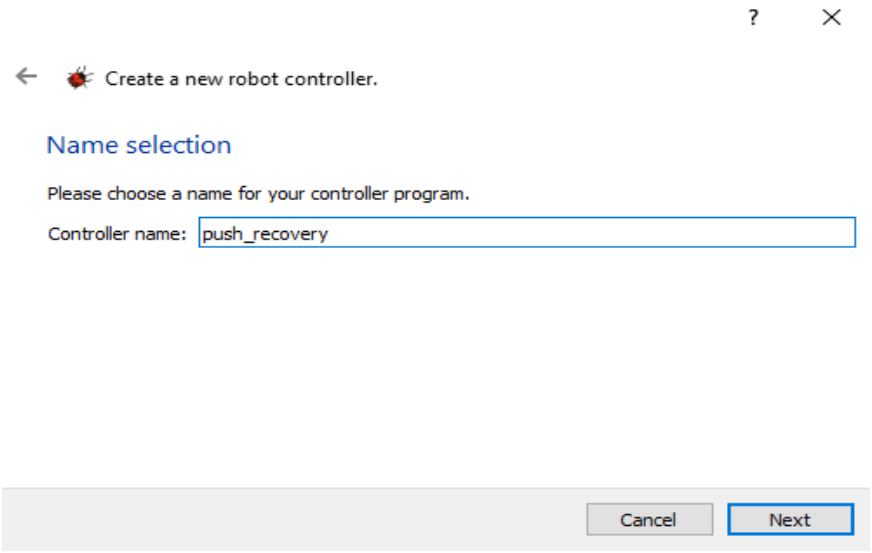
Şekil 6.9 Webots ortamında yeni kontrolör oluşturma-1

Burada robot hangi programlama dili ile kodlanacaksa o dili seçmek gerekmektedir. Programlanacak dil seçtikten sonra Next tuşu ile bir sonraki aşamaya geçilmelidir.



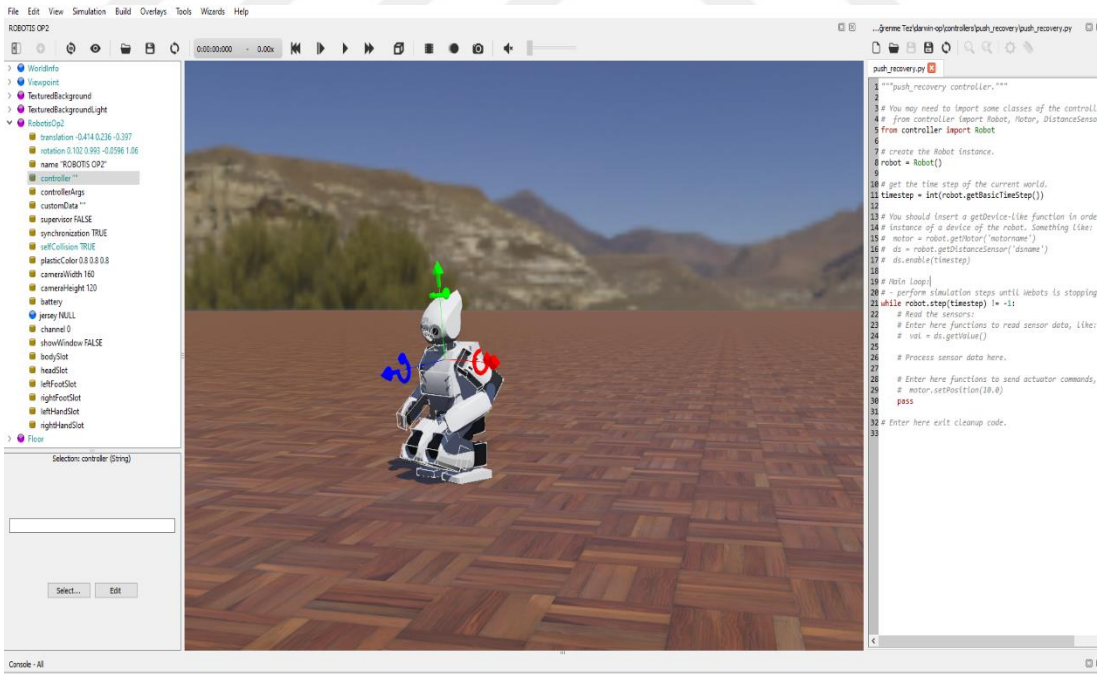
Şekil 6.10 Webots ortamında yeni kontrolör oluşturma-2

Ardından gelen pencerede kontrolöre bir isim verip Next tuşu ile sonraki pencereye geçilmelidir. Son olarak Finish tuşuna basarak projeye kontrolör eklenilmektedir.



Şekil 6.11 Webots ortamında yeni kontrolör oluşturma-3

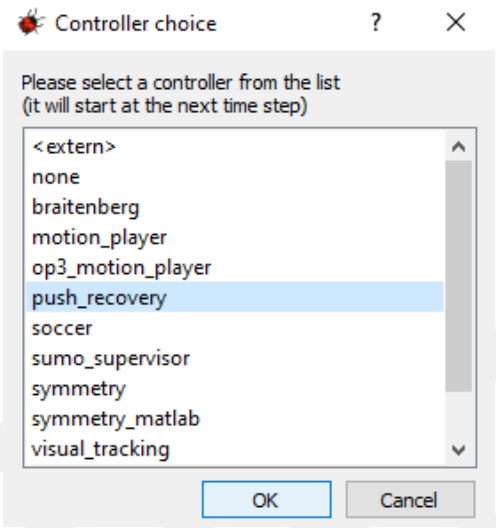
Projeye kontrolör eklendikten sonra ekran görüntüsü Şekil 6.12'de verilmiştir. Görüntünün sağ tarafında kodlamanın yapılacağı text editör alanı bulunmaktadır. Kodlama yapıldıktan sonra kontrolörü kaydetmek için text editör kısmının üstünde bulunan kaydet tuşu ile kontrolör kaydedilir.



Şekil 6.12 Webots ortamında yeni kontrolör oluşturma-4

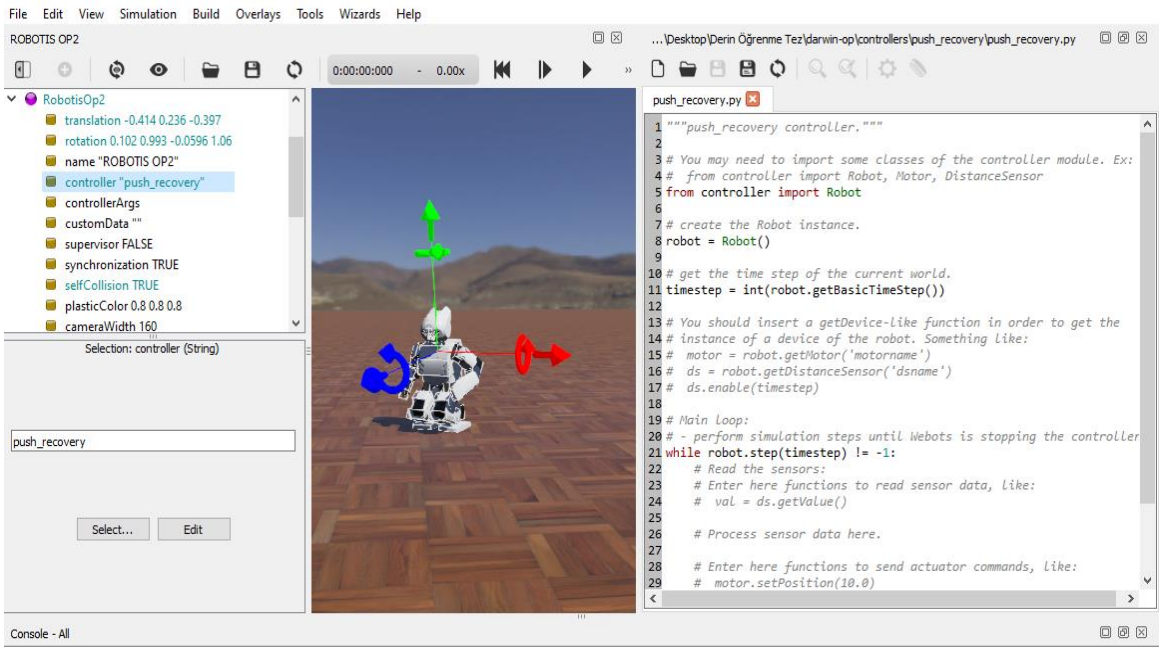
Kontrolörü kaydettikten sonra sol tarafta bulunan sekmede eklenen robota bu kontrolörü tanıtmak gerekmektedir. Bir proje içerisinde birden çok nesne bulunabilir.

Her bir nesnenin yapacağı işlem farklı olacağından her biri için farklı bir kontrolör yazılması gerekmektedir. Önceki bölümde eklenen Robotis-OP2 nesnesi sol sekmede seçilir. ‘Controller’ yazan kısma tıklanır. Sol alt sekmede ‘selection’ kısmında görünen ‘Select’ butonuna tıklanır. Açılan pencerede nesne için olan kontrolör seçilip ‘OK’ butonuna basılır.



Şekil 6.13 Webots ortamında nesneye kontrolör ekleme-1

Kontrolörü nesneye ekledikten sonra sol tarafta seçilen nesnenin ‘controller’ özelliğine seçilen kontrolör eklenmiş olmaktadır. Şekil 6.14’te robota eklenen kontrolör gösterilmiştir.



Şekil 6.14 Webots ortamında nesneye kontrolör ekleme-2

6.3 Dış Kuvvet Uygulama

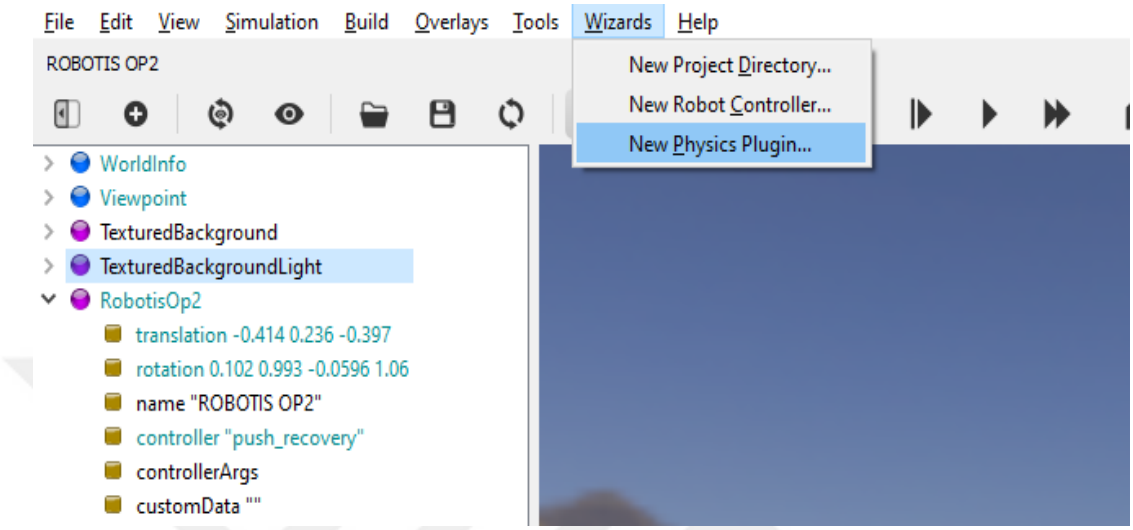
Bu çalışmada insansı robotlara dışardan gelen itme kuvvetine karşı robotun dengede konumuna gelmesi sağlanmıştır. Webots simülâsyon ortamında bir nesneye itme kuvveti uygulamanın iki farklı yöntemi bulunmaktadır. Bu yöntemlerden biri manuel, diğeri otomatik olarak ayarlanabilmektedir.

İlk yöntemde, simülâsyon çalışırken itme kuvveti uygulamak istenilen nesneyi seçerek klavyeden 'Alt' tuşuna ve aynı zamanda farenin sol tuşuna basarak kuvvet uygulayıp istenilen yönde fareyi ileri hareket ettirmektir. Fare hareketi ne kadar uzağa doğru çekilirse robota uygulanan kuvvetin büyüklüğü de o kadar artmaktadır. Şekil 6.15'te robota manuel olarak dışardan itme kuvveti uygulanmasını göstermektedir. Manuel olarak uygulanan itme kuvvetinin bir dezavantajı bulunmaktadır. Kullanıcı simülâsyon çalışırken belli aralıklar ile bu işlemi sürekli tekrarlamalıdır. Robotların DPÖ algoritmaları ile eğitilmesi uzun zamanlar alabilmektedir. Eğitim süresi boyunca sürekli kullanıcının manuel olarak itme kuvveti uygulaması pek mümkün görünmemektedir.



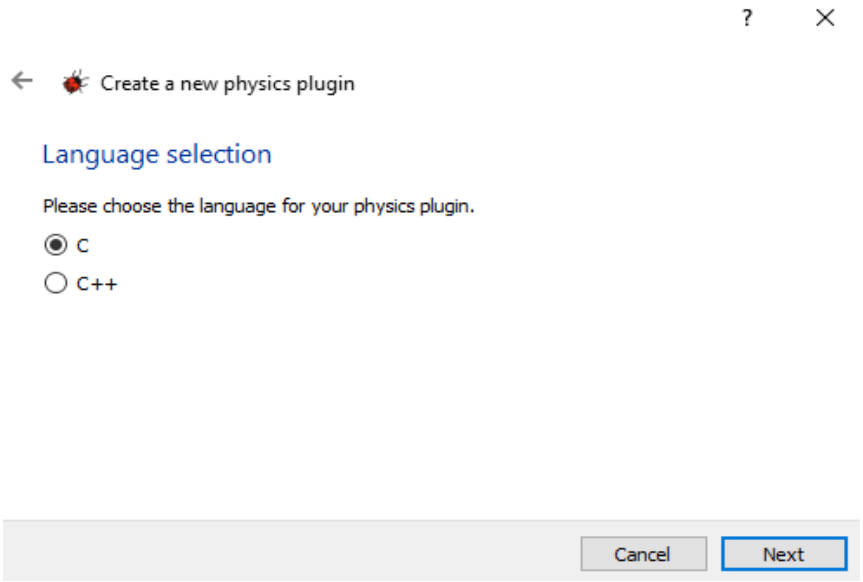
Şekil 6.15 Robota manuel olarak itme kuvveti uygulanması

İkinci yöntem ise bir fizik dosyası oluşturularak programlama yardımı ile belirlenen aralıklarda robota dışardan itme kuvvetinin uygulanmasıdır. Bu yöntemi uygulamak için ilk önce Wizards->New Physics Plugin yolu izlenir. Ardından gelen pencerede Next tuşu ile ilerlenir.



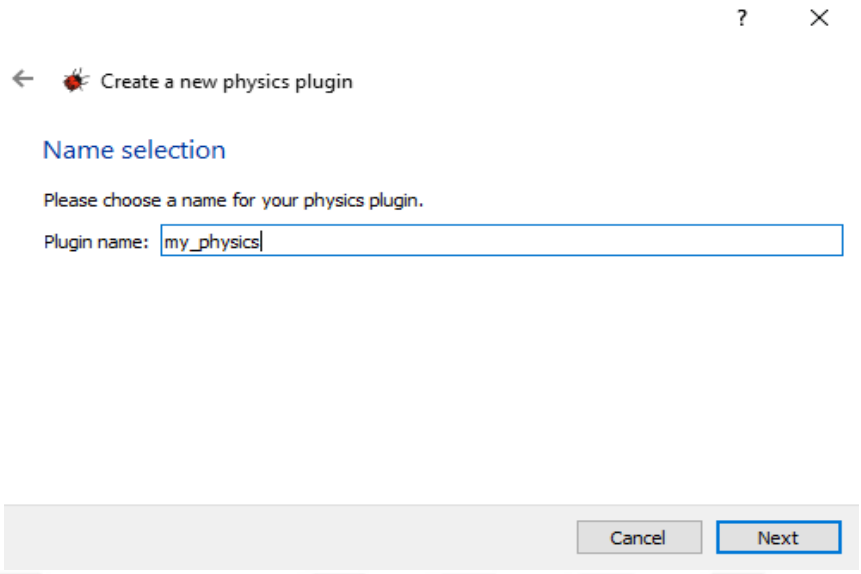
Şekil 6.16 Robota fizik eklentisi eklemek-1

Açılan pencerede eklenmek istenen yeni fizik dosyası için C veya C++ programlama dillerinden birini seçilir. Fizik dosyası için Webots iki farklı programlama dili desteği sunmaktadır.



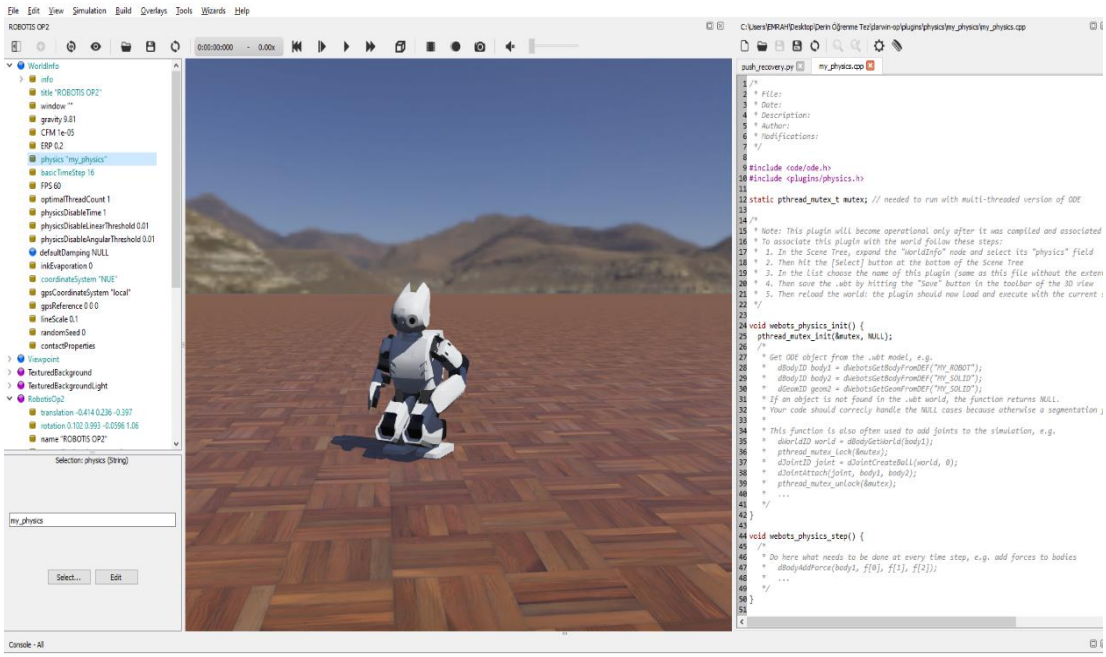
Şekil 6.17 Robota fizik eklentisi eklemek-2

Ardından gelen pencerede fizik dosyasına bir isim verilip Next tuşu ile ilerlenir. Gelen ekranda Finish tuşuna basıp dosya oluşturulur.



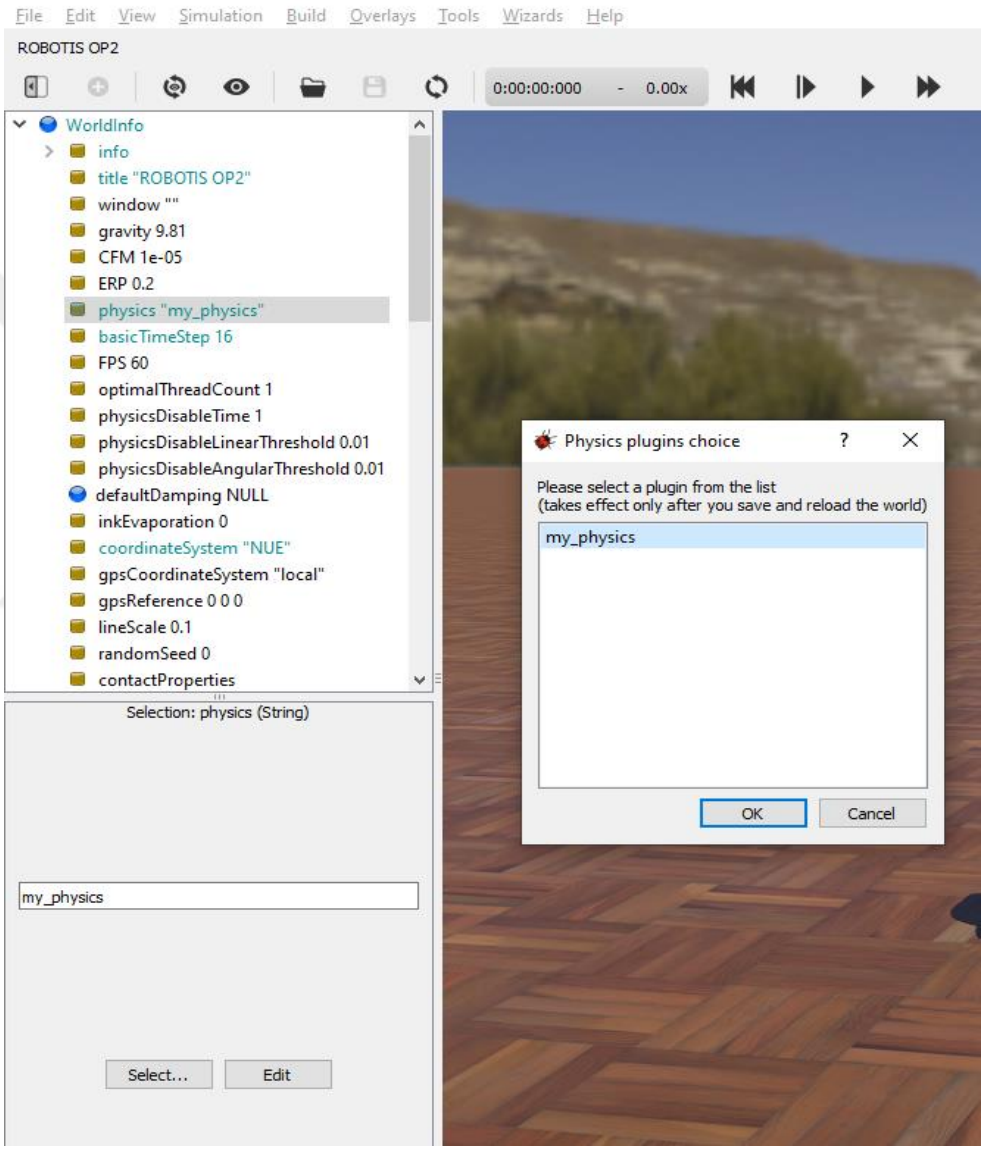
Şekil 6.18 Robota fizik eklentisi eklemek-3

Fizik dosyasının eklenmiş hali Şekil 6.19'da verilmiştir. Burada '.cpp' uzantılı fizik dosyası içerisinde gerekli kodlama işlemi yapılır. Kodlama yapılırken hangi nesneye kuvvet uygulanacaksa o nesnenin ismi doğru bir şekilde belirtilmelidir. Örneğin; eklenen Robotis-OP2 nesnesine sol taraftaki sekmede tıkladığında 'DEF' olarak belirtilen bölümde robota bir isim tanımlaması yapılmalıdır. Uygulamada buradaki DEF alanına 'MYROBOT' ismi ile robot tanımlanmıştır.



Şekil 6.19 Robota fizik eklentisi eklemek-4

Sonrasında sol sekmede üst kısımda bulunan ‘WorldInfo’ kısmına tıklayıp altında yer alan ‘physics’ özelliğine oluşturulan fizik dosyası eklenilir. ‘Physics’ özelliği seçili iken sol alt tarafta bulunan kısımda ‘Select’ butonu ile istenilen fizik dosyası seçilir. ‘OK’ butonuna basılıp işlemler kaydedilir. Sonrasında uygulamanın üst bölümünde yer alan kaydet butonu ile tüm uygulamayı kaydedip ortamı yeniden yüklemek gerekmektedir.



Şekil 6.20 Robota fizik eklentisi eklemek-5

Böylece fizik dosyası oluşturulan sanal ortamda aktifleşmiş olmaktadır. İstenilen nesnelere belirlenen aralıklarda ve yönlerde dışardan kuvvet uygulaması otomatik olarak gerçekleştirilmektedir. Oluşturulan fizik dosyası ile robotun kütlesi, ağırlık merkezi, motor kuvvetleri gibi verilerine erişerek robota uygulanan gerçekçi kuvvet

hesaplanabilmektedir. Uygulamada derin pekiştirmeli öğrenme algoritmaları kullandığı için ikinci yöntem olan otomatik olarak kuvvet uygulama yöntemini uygulanmıştır.

6.4 Kontrolörlerin Programlanması

Webots ortamında gerekli çevre, fizik ve kontrolörler oluşturulduktan sonra eklenen kontrolörlerin derin pekiştirmeli öğrenme algoritmalarına uygun kodlanması gerekmektedir.

Robotun sensörlerinden alınan ivme, gyro, tork ve konum gibi bilgileri anlık olarak kaydedilmiştir. Kaydedilen bu veriler ile robotun gövdesinin denge pozisyon aralığında olup olmadığı kontrol edilmiştir. Simülâsyon içerisinden robota kuvvet uygulamak için bir fizik kütüphanesi oluşturularak robot ile bağlantısı gerçekleştirilmiştir.

Robota rastgele ön ve arka taraftan kuvvetler uygulanmıştır. Uygulanan kuvvetler sürekli kontrol edilerek, denge konumu dışına çıktığı zaman ayak bileğine müdahale ederek robot denge pozisyonuna getirilmiştir. Proje itme kurtarma stratejilerinden ayak bileği stratejisi referans alınarak gerçekleştirilmiştir. Robotun eğitim sürecinde klasik kontrol yöntemi olarak PD, tahmine dayalı kontrol yöntemi olarak MÖK, derin pekiştirmeli öğrenme olarak ise DQA ve ÇDQA algoritmaları kullanılmıştır. Algoritmadaki durumlar (6.1)'deki gibi S değişkeni ile gösterilmiştir.

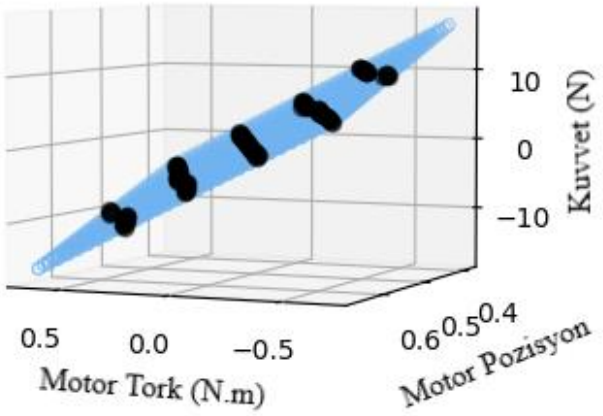
$$S = (\ tY\ddot{o}n, mutlak(bilek_{tork}), ayak_bile\breve{g}i_poz, kY\ddot{o}n, mutlak(kuvvet)) \quad (6.1)$$

Burada; yYön torkun işareti, $,mutlak(bilek_{tork})$ torkun mutlak değerini, ayak_bileği_poz ayak bileğinin pozisyonunu, kYön uygulanan itmenin işaretini ve mutlak(kuvvet) uygulanan itmenin büyüklüğünün mutlak değerini verir. Uygulanan itmenin değerini hesaplamak için ayak bileğinin tork ve pozisyon bilgilerini kaydedip çoklu doğrusal regresyon ile (6.2)'de verilmiştir.

$$F = 20{,}9255 - (16{,}0364 * T) - (40{,}5437 * P) \quad (6.2)$$

Burada; F uygulanan itme kuvvetinin büyüklüğü, T ayak bileğinin torku, P ise ayak bileğinin konum bilgisidir. Rastgele alınmış 120 pozisyon ve tork bilgilerinin çoklu

doğrusal regresyon grafiği Şekil 6.21'de verilmiştir. R-squared, istatistiksel ölçme metriği bir olarak elde edilmiştir.

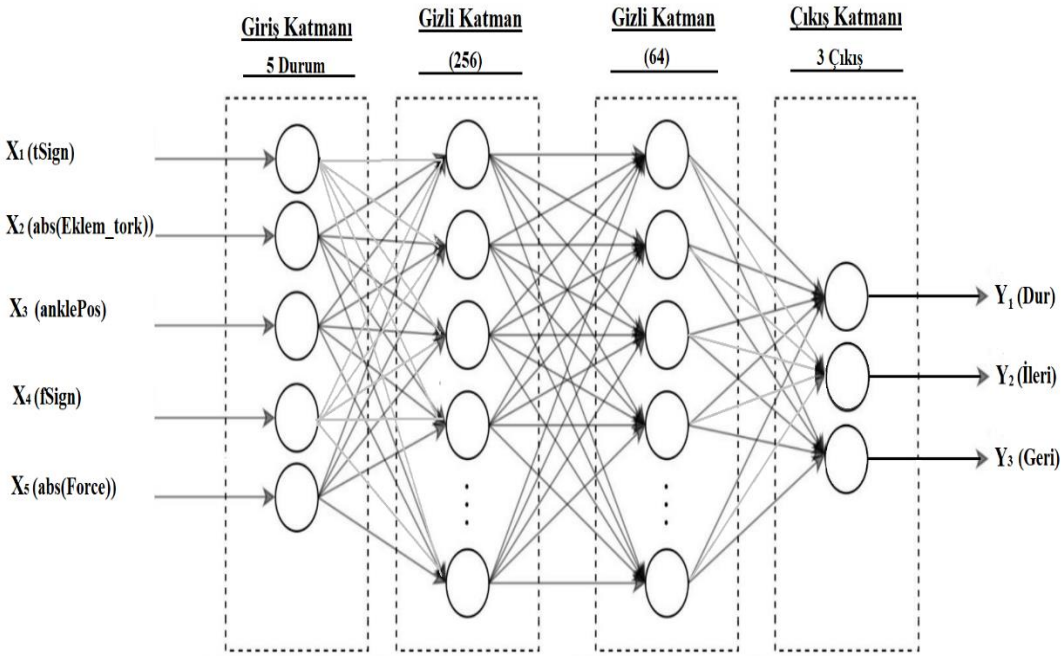


Şekil 6.21 Robota uygulanan kuvvet grafiği

Robotun gelen itmeye karşı uygulayacağı eylemler üç tanedir. Bunlar sırasıyla;

- Dur (0),
- ileri hareket (1),
- geri harekettir (2).

Oluşturulan modelin mimarisi Şekil 6.22'de gösterilmiştir.

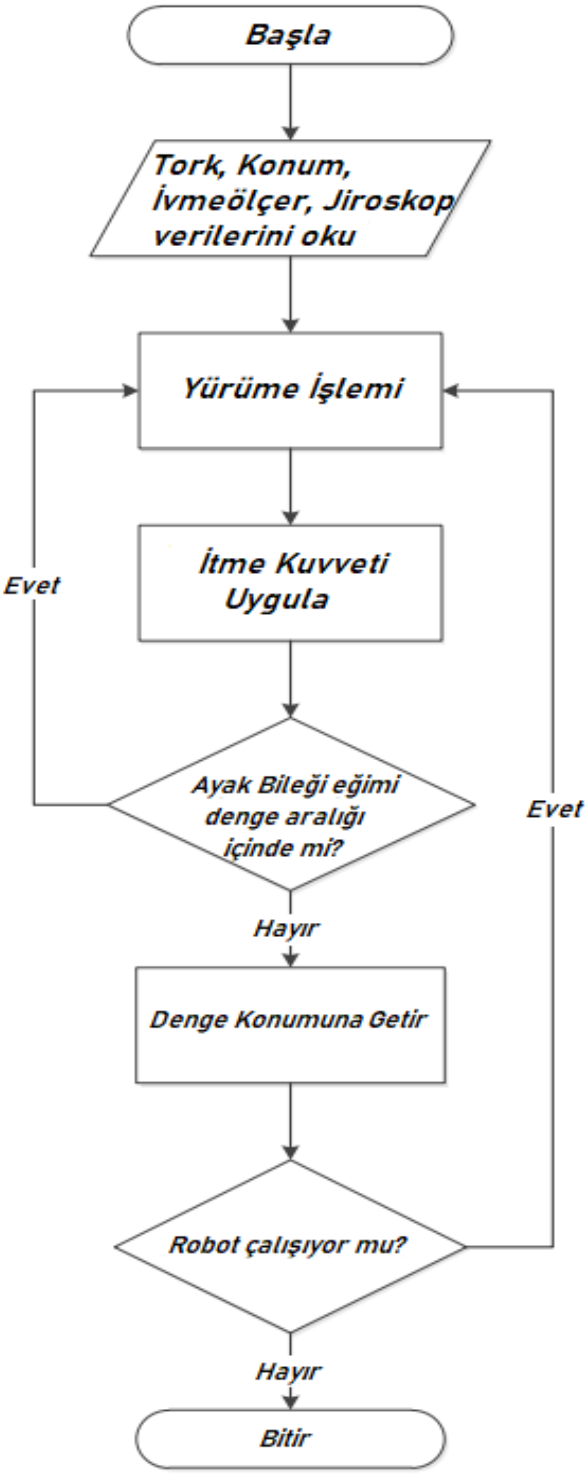


Şekil 6.22 Oluşturulan yapay sinir ağı mimarisinin yapısı

Robotun yaptığı harekete göre kazanacağı ödül fonksiyonu R olarak ifade edilir. R formülü (6.3)'te verilmiştir.

$$R = 0{,}25 - T^2 \tag{6.3}$$

İnsansı robotun denge hareket akış şeması Şekil 6.23'te verilmiştir.



Şekil 6.23 İtme-Kurtarma kontrollü insansı robotun hareket akış diyagramı

Oluşturulan DQA ve ÇDQA algoritmalarında kullanılan parametreler ayrıntılı olarak Tablo 6.1'de verilmiştir.

Tablo 6.1 DQA ve ÇDQA algoritması parametreleri

Parametreler	Değerler
Toplam Bölüm Sayısı	7500
Hafıza	2000
Gamma	0,95
Epsilon Greedy Başlangıç Değeri	1,0
Minimum Epsilon Greedy	0,005
Epsilon Greedy Azaltma Oranı	0,995
Batch Boyutu	64
Öğrenme Başlama	100
Eylem Sayısı	3
Durum Sayısı	5
Maksimum Adım Sayısı	30

Kullanılan her iki derin pekiştirmeli öğrenme algoritmasında da aktivasyon fonksiyonu olarak giriş ve gizli katmanlarda DBB kullanılmıştır. Çıkış katmanında ise aktivasyon fonksiyonu lineer olarak belirlenmiştir. DQA ve ÇDQA algoritmalarının yapay sinir ağı özellikleri Tablo 6.2'de verilmiştir.

Tablo 6.2 DQA ve ÇDQA algoritması yapay sinir ağı özellikleri

Parametreler	Değerler	Aktivasyon Fonksiyonu
Giriş Katmanı	5	DBB
Gizli Katman 1	256	DBB
Gizli Katman 2	64	DBB
Çıktı Katmanı	3	DBB
Optimizer	RMSprop Algorithm	
Kayıp	Mean Squared Error(mse)	
Öğrenme Oranı	0,00025	
Rho	0,95	
Epsilon	0,01	

Modelin geliştirilmesi sırasında kullanılan bazı kütüphaneler ve çerçeveler şu şekilde sıralanabilir.

- Nump
- Datetime
- Controller
- Math
- Csv
- Pandas
- Tensorflow
- Keras

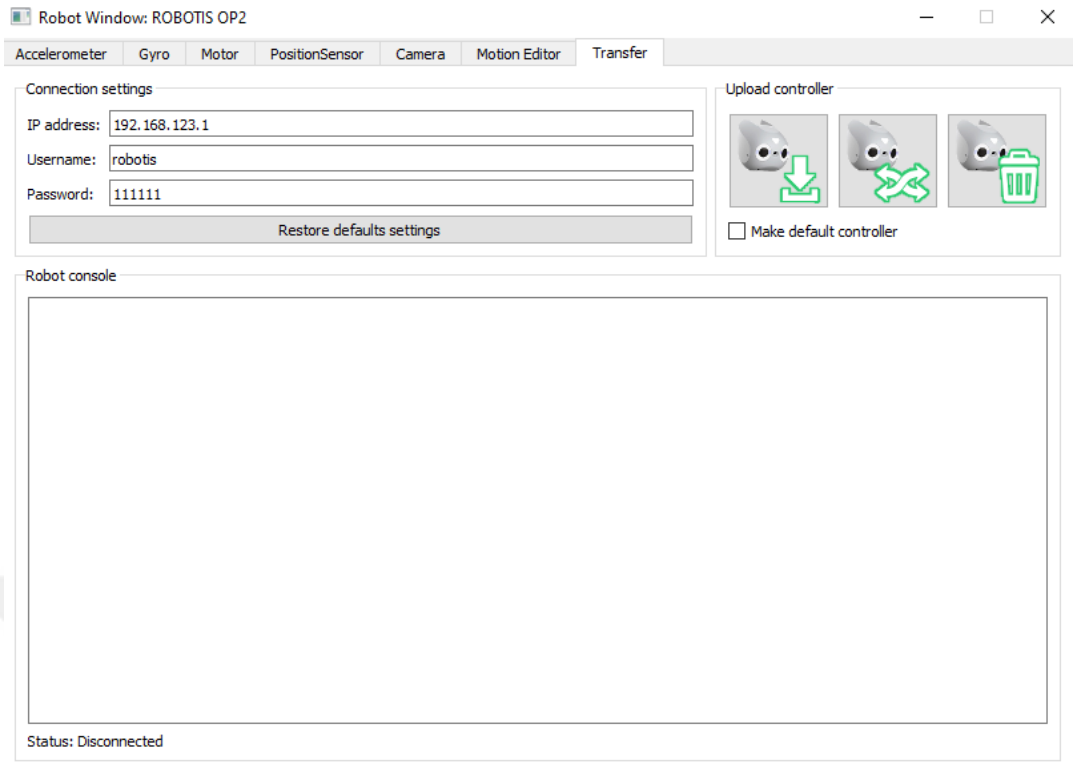
Kullanılan tüm kütüphane ve çerçeveler yapay zekâ ve makine öğrenmesi uygulamalarında sıklıkla kullanılan ücretsiz ve açık erişime sahiptir. Yapay zekâ uygulamalarında özellikle Keras ve Tenserflow kütüphaneleri kullanılmaktadır.

Derin öğrenme algoritmalarının dışında klasik yöntemlerden PD kontrolör kullanılarak aynı uygulama gerçekleştirilmiştir. Klasik kontrolör için kullanılan parametre değerleri Tablo 6.3'de verilmiştir. Burada I değeri 0 olarak alınmıştır. Bu yüzden kontrolör PD kontrolör olarak geçmektedir.

Tablo 6.3 PD kontrolör parametreleri

Parametreler	Değerler
P	0,5
I	0
D	0,01
Örnekleme Zamanı	0,00001
Eylem Sayısı	3
Durum Sayısı	5
Maksimum Adım Sayısı	30

Yazılan kodun gerçek ortamda bulunan robota aktarılması için robot ile bilgisayar arasında bağlantı kurulmalıdır. Webots içerisinde programlayarak kontrol edilen robot üzerine çift tıklayarak Robot Penceresi (Robot Window) açılmalıdır.



Şekil 6.24 Robot penceresi

Bağlantı kurabilmek için robotun IP adresini 'IP address' kısmına girmek gerekmektedir. Eğer ethernet kablosu ile bağlantı gerçekleştiriliyorsa 192.168.123.1 olarak IP adresi girilmelidir. Eğer wifi bağlantısı kullanılacak ise ipconfig komutu ile robotun IP adresi tespit edilip girilmelidir. Kullanıcı adı için robotis ve şifre için 111111 girilerek bağlantı gerçekleştirilir. Kodu gerçek robotta test etmek için Şekil 6.25'te verilen gönder simgesine tıklanmalıdır.



Şekil 6.25 Programı gönder simgesi

Bu şekilde bağlantı Webots gerçek robota bağlanabilecektir. Eğer bağlantı sırasında bir hata oluşursa hata nedeni gösterilmektedir. Aktarım gerçekleştikten sonra robotta bu komutları kullanmak istemiyorsak Şekil 6.26'de görülen kaldır düğmesi ile robotun hafızasından gerekli dosyalar silinecektir.



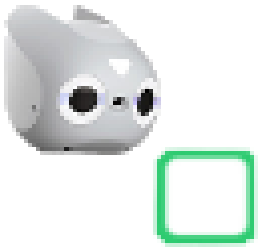
Şekil 6.26 Programı kaldır simgesi

Uzaktan kumanda gerçekleştirmek istenildiğinde robot penceresinde bulunan ‘Remote Control’ tuşuna basarak kontrol gerçekleştirilir. Şekil 6.27’da uzaktan kumanda düğmesi görünmektedir.



Şekil 6.27 Uzaktan kumanda simgesi

Uzaktan kumanda özelliği durdurulmak istediğinde durdurma düğmesine basıp aktarım durdurulabilmektedir. Durdurma düğmesi Şekil 6.28’de verilmiştir.



Şekil 6.28 Durdurma simgesi

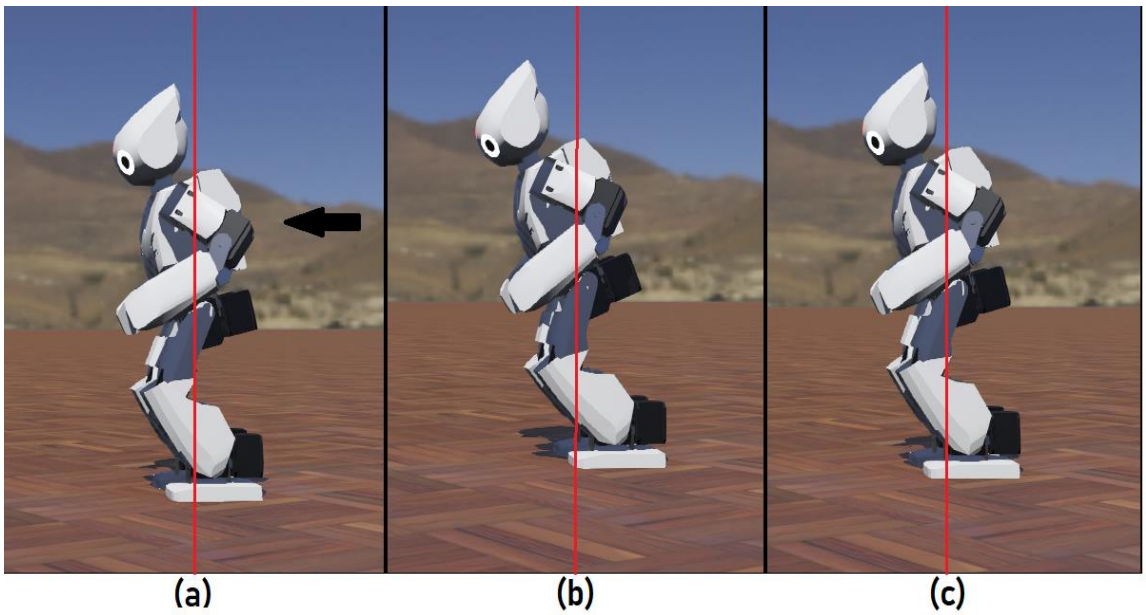
7. BULGULAR VE TARTIŞMA

Kurulan sistemin dengede kalacak şekilde dayanabileceği maksimum itme kuvvetleri hesaplanmıştır. Ardından robota önden ve arkadan farklı şekillerde itme kuvveti uygulanmış ve sonuçlar karşılaştırılmıştır. Uygulamada derin pekiştirmeli öğrenme algoritmalarından DQA ve ÇDQA uygulanmıştır. Ayrıca klasik kontrol yöntemi olan PD kontrolör ve tahmine dayalı kontrol yöntemi olan MÖK kullanılmıştır. İtme esnasındaki ani değişimlere sönümlendirmek için PD kontrolör tercih edilmiştir. Her yöntem için her bir bölüm 30 adımdan oluşmaktadır. PD ve MÖK kontrolörler 100 bölümden oluşmaktadır. Çünkü belli bir döngü içerisinde sonuçları tekrar etmektedir. Tablo 7.1'de Kontrol yöntemlerinin kaç bölüm ve adımdan oluştuğu gösterilmiştir.

Tablo 7.1 Algoritma bölüm sayıları

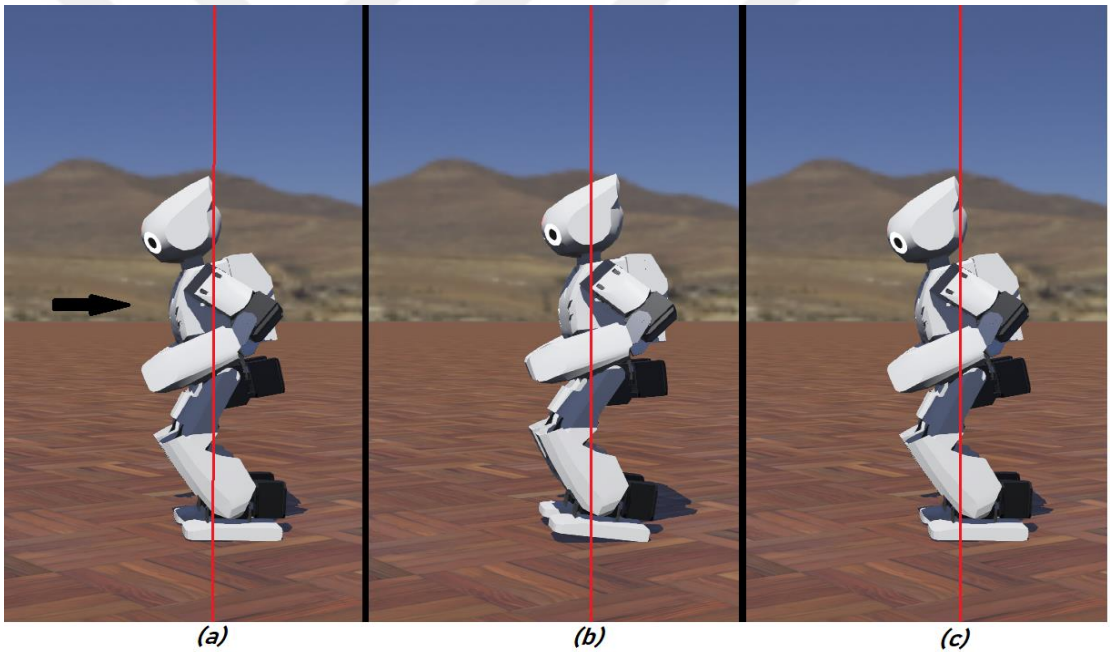
Kontrol Algoritması	Bölüm	Adım Sayısı
PD	100	30
MÖK	100	30
DQA	7500	30
ÇDQA	7500	30

Şekil 7.1'de simülâsyon ortamında arkada uygulanan itme sonucunda robotun denge konumuna gelmesi gösterilmiştir.



Şekil 7.1 a) Arkadan itme b) İtme sonucu aldığı pozisyon c) Denge pozisyonu

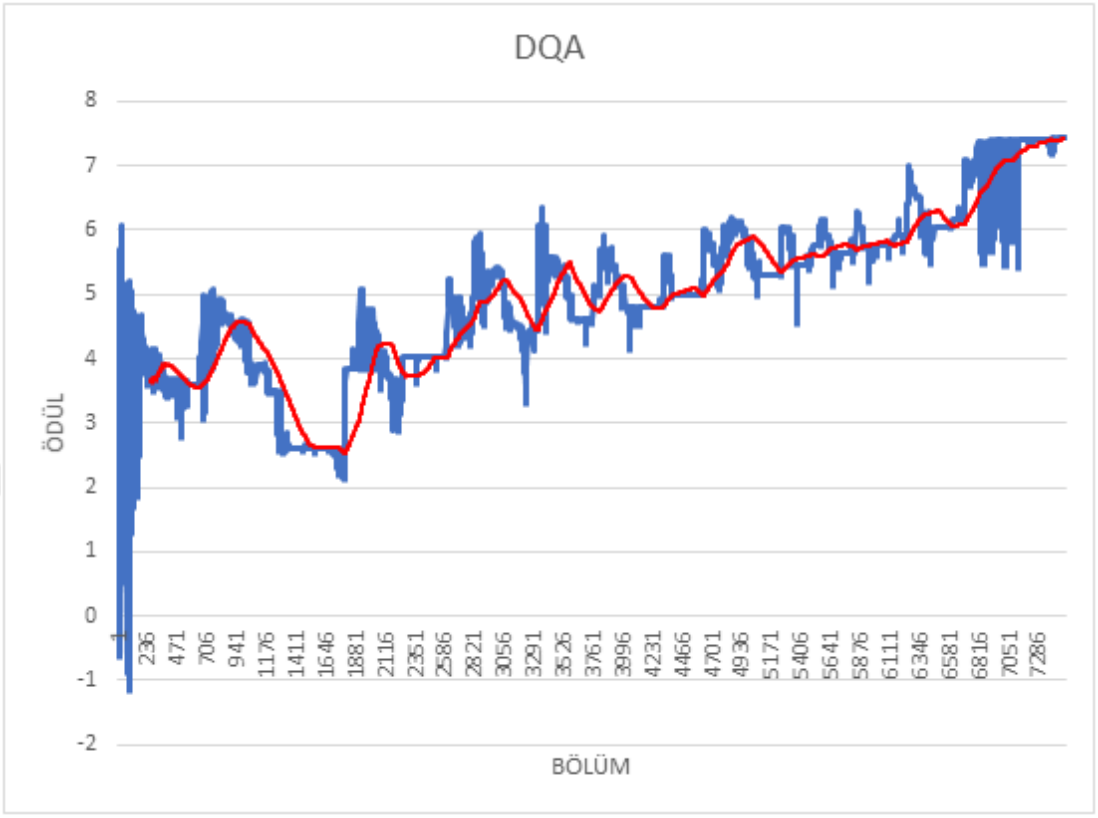
Şekil 7.1 a) kısmında robota arka kısımdan rastgele büyüklüklerde bir itme kuvveti uygulanmaktadır. Bu itme sonucunda robot dengesini kaybetmektedir. Robot itme sonucunda Şekil 7.1 b) kısmındaki pozisyonu almaktadır. Robotta kullanılan kontrol yöntemi sayesinde ayak bileği stratejisi ile robot denge konumuna getirilmiştir. Tekrar denge konumuna gelen robot Şekil 7.1 c) kısmında gösterilmiştir. Simülâsyon ortamında robota önden uygulanan itme kuvvetinin sonucu robotun dengeye gelmesi ise Şekil 7.2'de gösterilmiştir. Robota rastgele büyüklüklerde belli her 10 adımda bir farklı büyüklükte kuvvet uygulanmıştır. Bu itmeler robotun dengesinin bozulmasına sebep olmuştur. Şekil 7.2 a) kısmında robota ön kısımdan gelen itme gösterilmiştir. Şekil 7.2 b) kısmında ise robotun dengesinin bozulduğu görülmektedir. Son olarak Şekil 7.2 c) kısmında ise denge konumuna gelmesi gösterilmiştir.



Şekil 7.2 a) Önden itme b) İtme sonucu aldığı pozisyon c) Denge pozisyonu

DQA algoritması kullanılarak 7500 bölüm gerçekleştirilmiştir. Robot yaklaşık 7200 bölüm sonra dengede kalmayı öğrenmiştir. Her bir bölüm 30 adımdan oluşmaktadır. Ödül fonksiyonuna göre tam başarılı olan bir bölümden alınacak maksimum puan 7,5'tur. DQA algoritmasın 7200 bölüm sonra robotun kazanmış olduğu puan 7,411 olduğu tespit edilmiştir. DQA algoritması kullanılarak eğitilen robot eğitim sonunda toplam 37483,51 puan almıştır. DQA algoritması ile alınan sonuçlar Şekil 7.3'te

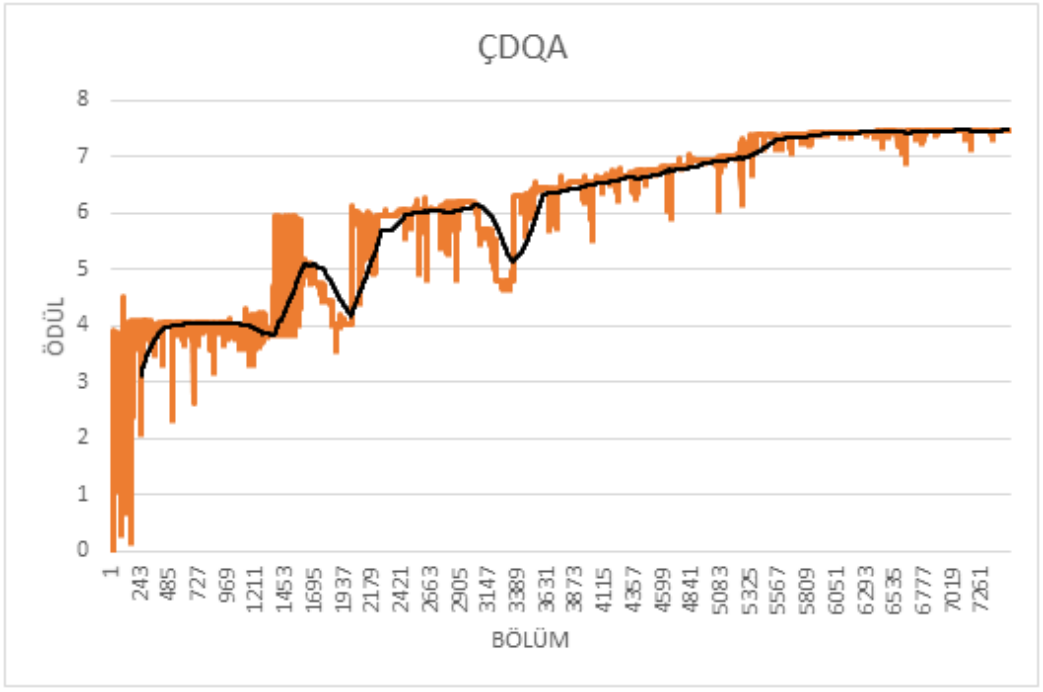
gösterilmiştir. 7500 bölüm sonunda almış olduğu en yüksek ödül ise 7,437 olduğu görülmüştür.



Şekil 7.3 DQA algoritma sonuçları

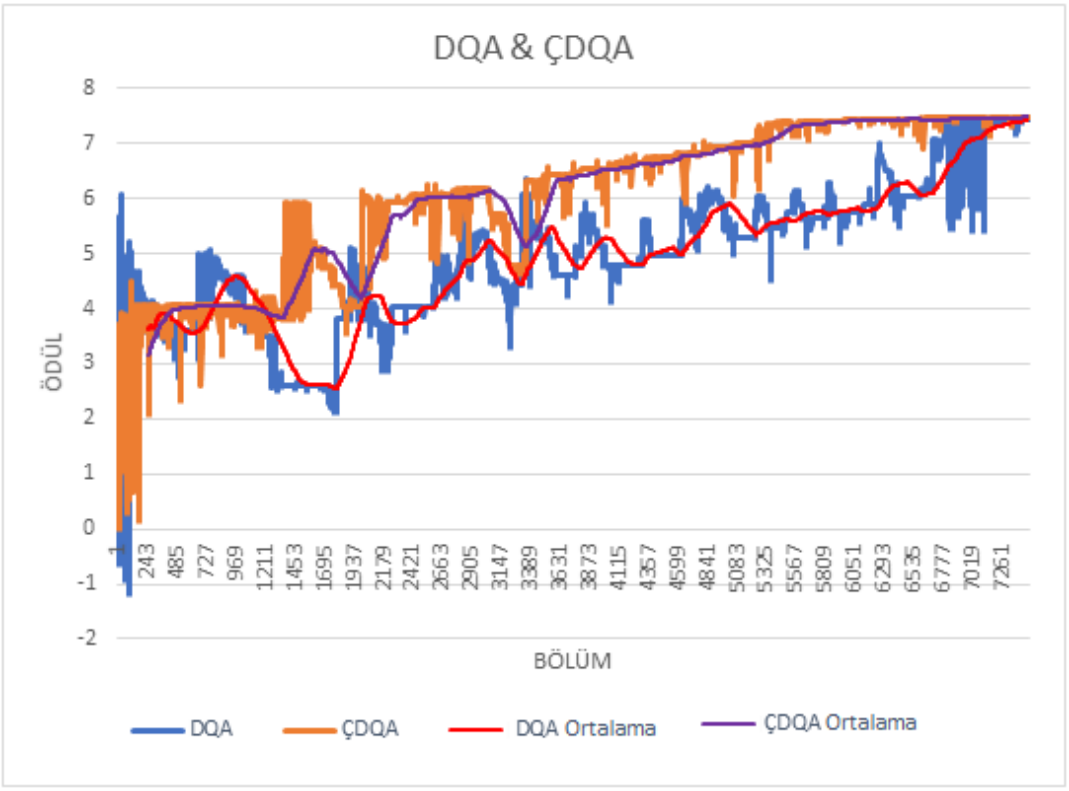
Şekil 7.3'de de görüldüğü gibi DQA algoritması ile 7200 bölüm sonrası %98,81 öğrenme gerçekleşmiştir. 7500 bölüm olan toplam eğitim sonucunda ise % 99,16 olduğu gözlemlenmiştir.

ÇDQA algoritması ile 7500 bölüm gerçekleştirilmiştir. Robot dengede kalmayı yaklaşık 5500 bölüm sonra öğrenmiştir. Her bir bölüm 30 adımdan oluşmaktadır. Ödül fonksiyonuna göre tam başarılı olan bir bölümden alınacak maksimum puan 7,5'tur. DQA algoritmasın 5500 bölüm sonra robotun kazanmış olduğu puan 7,407 olduğu tespit edilmiştir. Eğitim süresi boyunca ÇDQA algoritması ile toplam 45369,33 puan elde edildiği gözlemlenmiştir. ÇDQA algoritma sonuçları Şekil 7.3'te gösterilmiştir. 7500 bölüm sonunda almış olduğu en yüksek ödül ise 7,491 olduğu görülmektedir.



Şekil 7.4 ÇDQA algoritma sonuçları

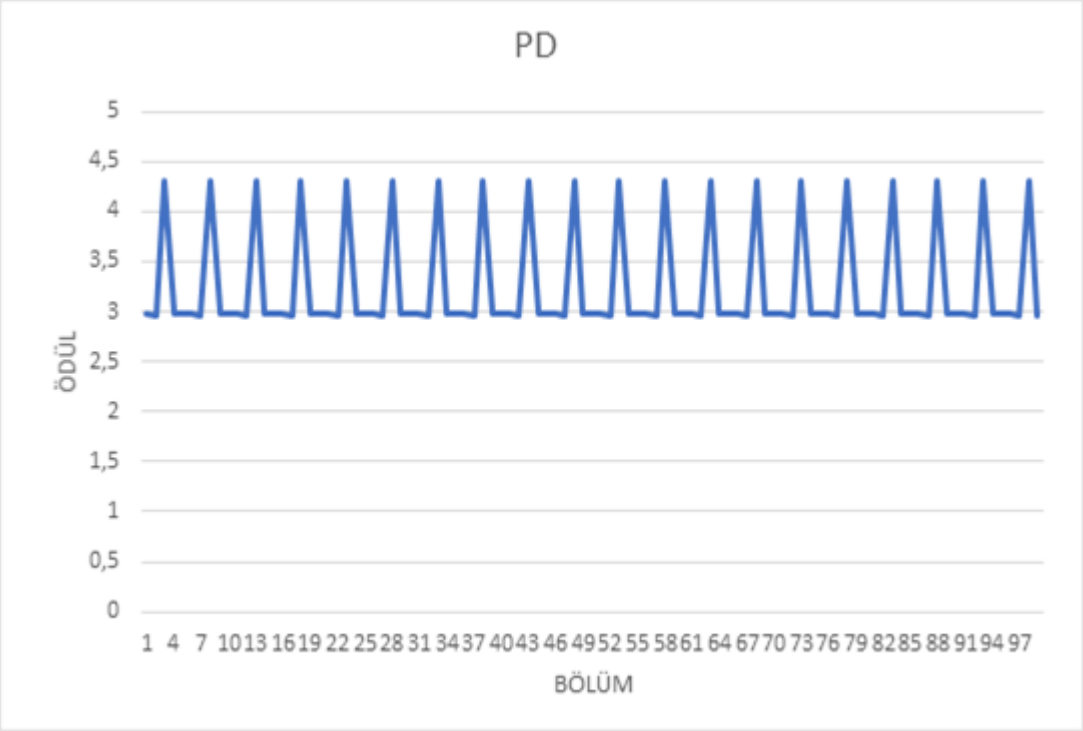
Şekil 7.4'de de görüldüğü gibi DQA algoritması ile 5500 bölüm sonrası %98,76 öğrenme gerçekleşmiştir. 7500 bölüm olan toplam eğitim sonucunda ise % 99,88 olduğu gözlemlenmiştir.



Şekil 7.5 DQA ve ÇDQA algoritma sonuçların karşılaştırılması

ÇDQA algoritması çift sinir ağı kullandığından DQA algoritmasına göre daha hızlıdır. Yapılan deneyler sonucunda ÇDQA algoritmasının DQA algoritmasına göre daha hızlı olduğu gözlemlenmiştir. Kazanılan ödüller değerlendirildiğinde ÇDQA algoritması ile alınan ödüller DQA algoritmasına göre %21,03 daha iyi olduğu anlaşılmıştır. DQA ve ÇDQA algoritmalarının sonuçlarının karşılaştırılması Şekil 7.5'te verilmiştir. ÇDQA algoritması DQA algoritmasına göre daha iyi olduğu anlaşılmıştır.

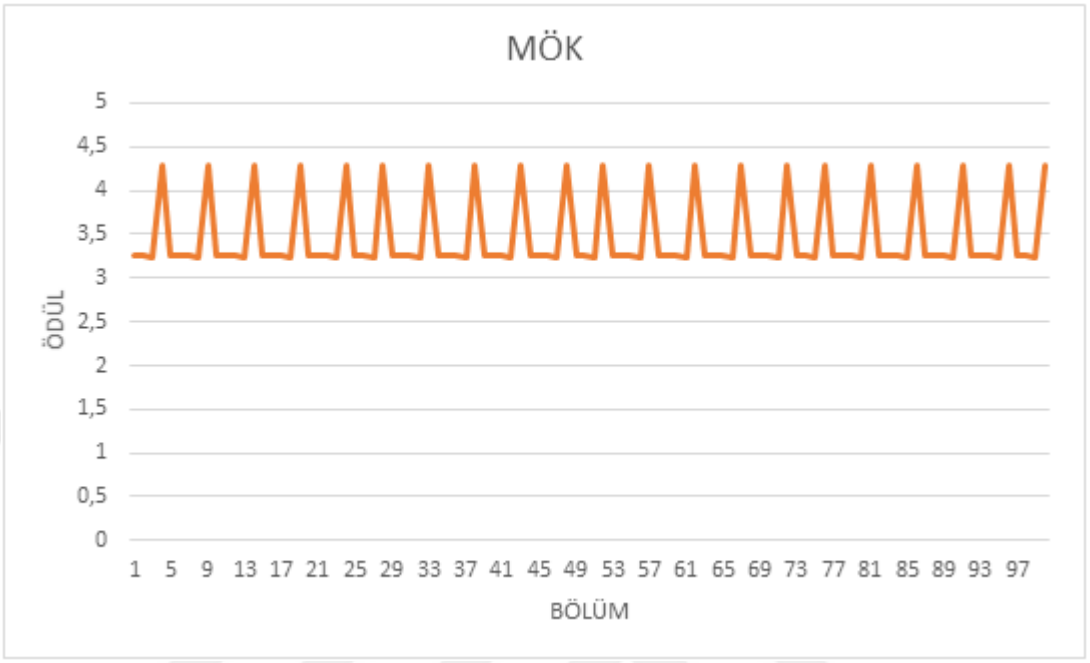
PD kontrolör ile 100 bölüm gerçekleştirilmiştir. Her bir bölüm 30 adımdan oluşmaktadır. Uygulamada belirlenen ödül fonksiyonuna göre en yüksek ödül puanı 7,5'tur. PD klasik bir kontrol yöntemi olduğu için verilen değer girdilere uygun bir çıktı hesaplamaktadır. Sonuç değerleri bir döngü olarak tekrar etmektedir. Ödül puanı ortalama 3,24 olarak hesaplanmıştır. 100 bölüm sonunda 324,265 puan elde etmiştir. Şekil 7.6'da PD kontrol yöntemi ile alınan sonuçlar verilmiştir.



Şekil 7.6 PD kontrol sonuçları

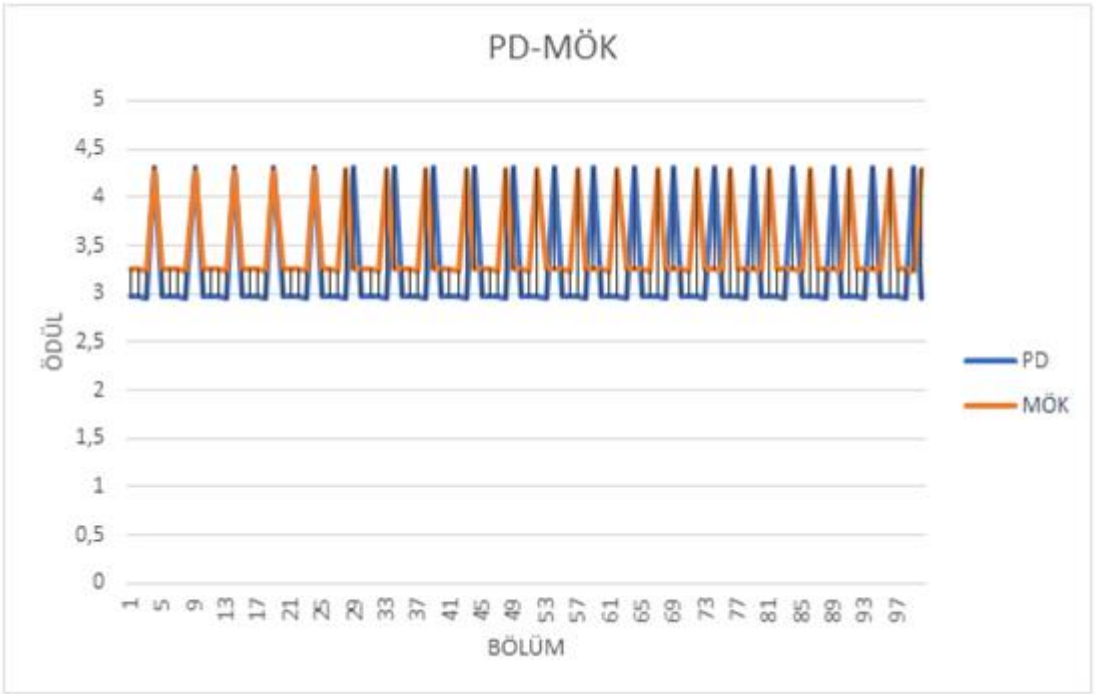
MÖK kontrolör ile 100 bölüm gerçekleştirilmiştir. Her bir bölüm 30 adımdan oluşmaktadır. Uygulamadaki belirtilen ödül fonksiyonuna göre en yüksek ödül puanı 7,5'tur. MÖK kontrolör uygulandığında sonuç değerleri PD de olduğu gibi bir döngü

olarak tekrar etmektedir. Ödül puanı ortalama 3,47 olarak hesaplanmıştır. 100 bölüm sonunda 347,068 puan elde etmiştir. Şekil 7.7'da MÖK kontrol yöntemi ile alınan sonuçlar gösterilmiştir.



Şekil 7.7 MÖK kontrol sonuçları

PD ve MÖK kontrol yöntemleri karşılaştırıldığında MÖK kontrol yöntemi PD kontrolöre göre daha iyi sonuç vermektedir. Şekil 7.8'de PD ve MÖK yöntemlerinin karşılaştırma sonuçları verilmiştir.



Şekil 7.8 PD ve MÖK kontrol sonuçlarının karşılaştırılması

Çalışmada uygulanan dört yöntem sonucunda da insansı robota uygulanan dış itme sonucunda robot denge konumuna gelebilmiştir. Robotun denge konumuna gelmesinde uygulanan kuvvetin büyüklüğü ve yönü önemli rol almaktadır. Uygulama sonuçları Tablo 7.2'de verilmiştir. PD kontrol en düşük ödül ortalamasına ve başarı performansı gösteren yöntem olmuştur. MÖK kontrol yöntemi, PD kontrolden daha iyi sonuçlar vermiştir. Fakat derin pekiştirmeli öğrenme algoritmalarına göre başarısız olarak görünmektedir. Yapılan bu çalışmada derin pekiştirmeli öğrenme algoritmaları en iyi sonuçları verdiği gözlemlenmiştir. ÇDQA algoritması DQA algoritmasından daha iyi performans göstermiştir. En başarılı ve hızlı olan yöntem Tablo 7.2'de görüldüğü gibi ÇDQA algoritması olmuştur.

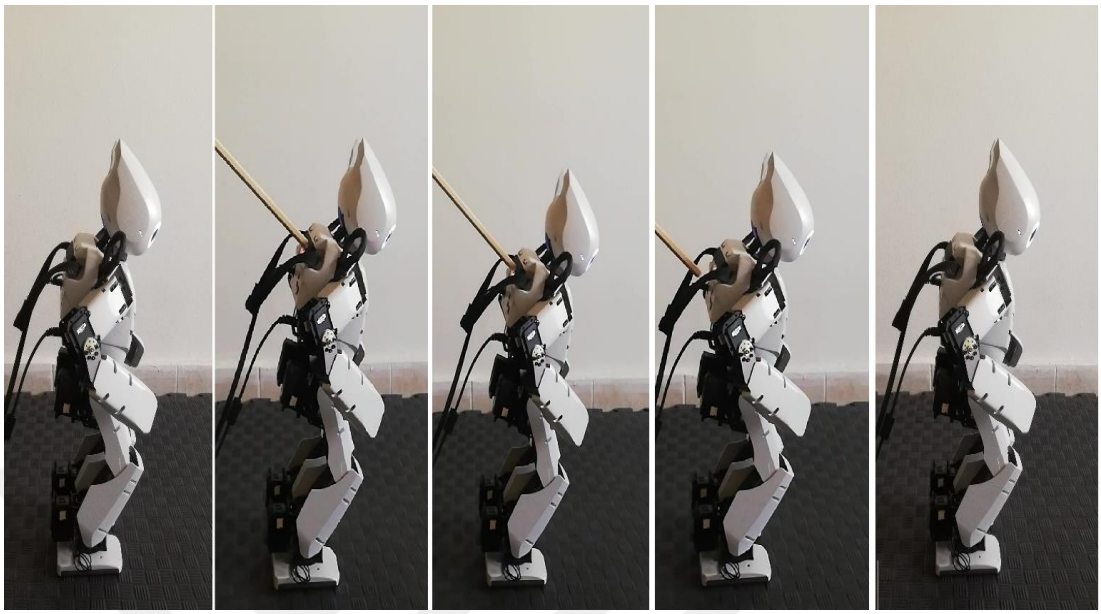
Tablo 7.2 Algoritma sonuçlarının karşılaştırılması

Kontrol Algoritması	Toplam Bölüm	Başarıya Ulaştığı Yaklaşık Bölüm	En Büyük Ödül	Ortalama Ödül
PD	100	100	7,5	3,24
MÖK	100	100	7,5	3,47
DQA	7500	7200	7,5	4,99
ÇDQA	7500	5500	7,5	6,04

Gerçek dünyada yapılan test videosuna https://youtu.be/nWG53hVrAE8 linki üzerinden erişilebilmektedir. Webots simülâsyon ortamında uygulamanın gerçek dünyada robota uygulanabilmesi için robot ile bağlantı kurulması gerekmektedir. Robot kontrol ekranı "Robot Penceresini Göster" ile açılır. Bu ekranda IP adresi, kullanıcı adı, şifre bilgileri girilerek ve uzaktan kumandayı başlat butonuna basılarak bağlantı yapılır. Veya simülâsyon ortamında geliştirilen uygulamalar bir internet kablosu üzerinden gerçek robota aktarılmıştır. Böylece simülâsyon ortamında çalışan uygulama gerçek robota aktarılır. Gerçek ortamda, sonuçlar bir çubuk yardımıyla robota itmeler uygulanarak kaydedilmiştir. Çalışmanın hem simülâsyon hem de gerçek ortam testlerinde 10 N ile -10 N arasındaki rastgele büyüklüklerdeki kuvvetler uygulanmıştır.

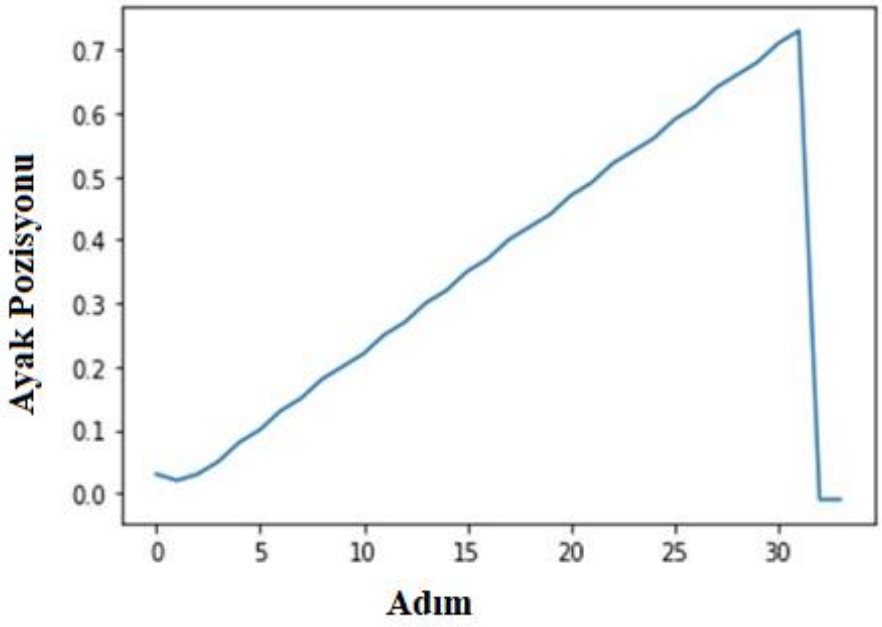
Uygulamanın gerçek ortam testi Şekil 7.9'da verilmiştir. Tasarlanan itme-kurtarma kontrolörü, ayak bileği stratejisi kullanılarak geliştirilmiştir. Robotun dış kuvvetlere

karşı ürettiği ayak bileği tork değerleri kaydedilmiştir. Dış kuvvetin büyüklüğüne bağlı olarak tepki vererek kararlı olduğu denge noktasına yaklaşmıştır.

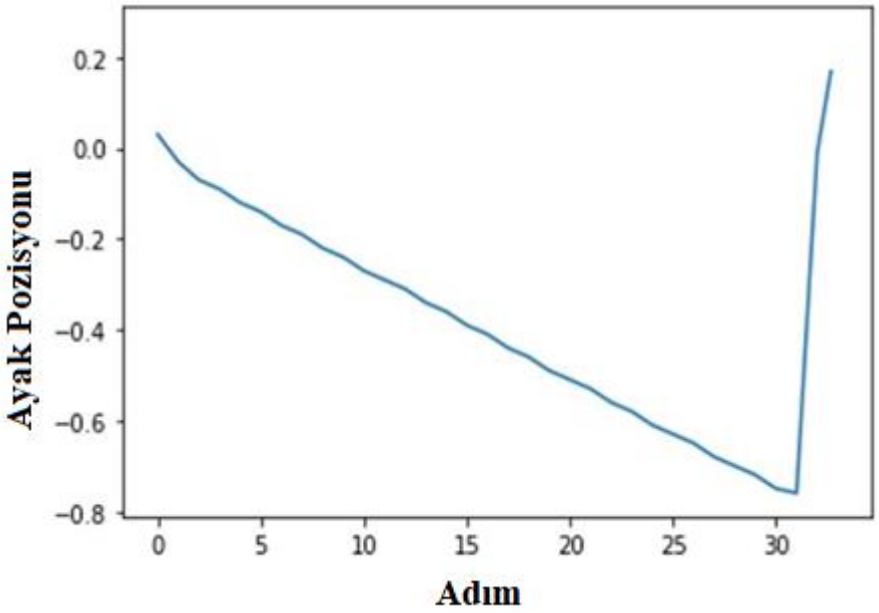


Şekil 7.9 Gerçek dünya test görüntüsü

Robota uygulanan kuvvetten sonra ayak bileğinin aldığı pozisyonlar Şekil 7.10 ve Şekil 7.11'de sunulmuştur. Dikey eksen ayak bileği ve yatay eksen basamağı ifade eder. Şekil 7.10 arkadan uygulanan kuvveti, Şekil 7.11 ise önden uygulanan kuvveti göstermektedir. Şekillerde görüldüğü gibi robot, yıkıcı kuvvetten sonra denge noktasına yaklaşır ve kararlı olduğunu gösterir.

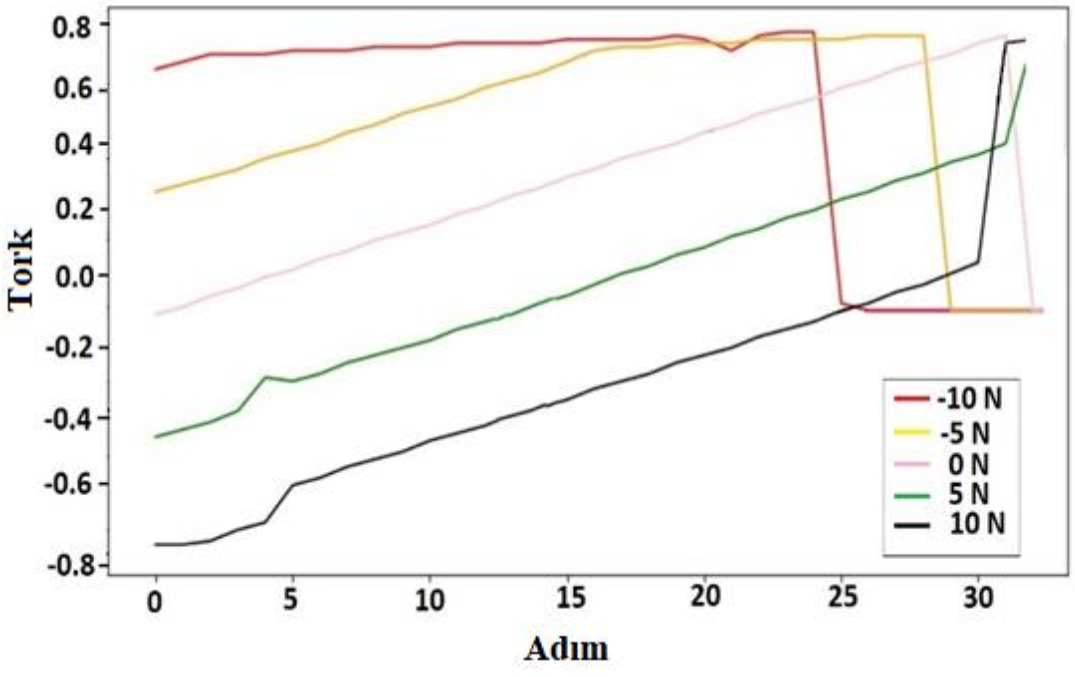


Şekil 7.10 Arkadan uygulanan kuvvetten sonra ayak bileği pozisyonu

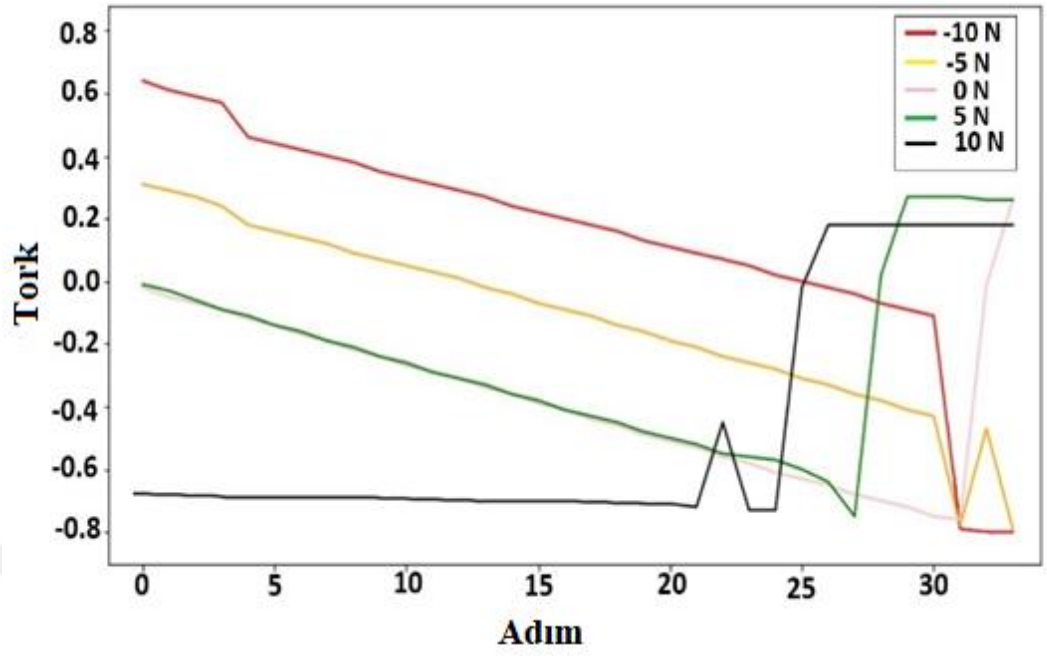


Şekil 7.11 Önden uygulanan kuvvetten sonra ayak bileği pozisyonu

Robota önden ve arkadan sırasıyla 10 N, 5 N, 0 N, -5 N, 10 N kuvvetler uygulanmıştır. Robota uygulanan farklı kuvvetlere göre ayak bileği tarafından üretilen tork değerleri Şekil 7.12 ve Şekil 7.13'de verilmiştir. Dikey eksen tork ve yatay eksen adımı ifade eder. Şekil 7.12, arkadan uygulanan kuvvetler sonucunda ayak bileği tarafından üretilen tork değerlerini göstermektedir. Şekil 7.13, önden uygulanan kuvvetler sonucunda ayak bileği tarafından üretilen tork değerlerini göstermektedir.



Şekil 7.12 Arkadan uygulanan farklı kuvvetlere göre ayak bileği tarafından üretilen tork değerleri



Şekil 7.13 Önden uygulanan farklı kuvvetlere göre ayak bileği tarafından üretilen tork değerleri

Simülâsyon ortamında yapılan testlerin aynısı gerçek ortamda da gerçekleştirilmiştir. Robota bir çubuk yardımıyla itmeler uygulanarak sonuçlar kaydedilmiştir. Robotun önden ve arkadan gelen itmelere karşı dengede durduğu gözlemlenmiştir. Simülâsyon ortamında alınan sonuçlara benzer sonuçlar elde edilmiştir.

8. SONUÇLAR

Bu çalışmada, 20 serbestlik derecesine sahip olan bir insansı robota uygulanan itmeye karşı dengede kalabilmesini sağlayan kontrol sistemi gerçekleştirilmiştir. Çalışmanın uygulama aşamasında kullanılmak üzere kablo ile ya da uzaktan bağlantı ile kontrol edilebilen Robotis-OP2 insansı robotu kullanılmıştır.

İnsansı robotu dış tepkiler sonucunda tekrar denge konumuna getirmek için farklı kontrol yöntemleri ve algoritmalar önerilmiştir. Klasik kontrol yöntemi olarak PD kontrolör, tahmine dayalı kontrol yöntemi olan MÖK ve DPÖ algoritmalarından DQA ve ÇDQA kullanılarak itme-kurtarma kontrolörleri tasarlanmıştır. İnsansı robota Webots robot simülâtör ortamında uygulamalar gerçekleştirilmiştir.

Tezin uygulama kısmında itme-kurtarma kontrolörlerinden ayak bileği stratejisi esas alınarak uygulamalar yapılmıştır. Robota önden ve arkadan rastgele itme kuvvetleri uygulanıp dengede kalması sağlanmıştır. Bu uygulama hem simülâsyon hem de gerçek ortamda gerçekleştirilmiştir.

Dış itme, simülâsyonda bir kütüphane oluşturularak, gerçek ortamda bir çubuk yardımıyla robota uygulanmıştır. Robotun önden ve arkadan itmelere karşı dengede kalmayı öğrendiği gözlemlenmiştir. Simülâsyon ve gerçek ortam sonuçları paralellik göstermektedir.

İnsansı robotun uygulanan dört yöntemindeki testlerde başarılı sonuçlar aldığı gözlemlenmiştir. Bu yöntemlerden başarı oranı en düşük yöntem PD olmuştur. PD yöntemini MÖK takip etmiştir. DPÖ algoritmaları ile daha başarılı sonuçlar alındığı tespit edilmiştir. Uygulanan DPÖ algoritmaları içinden ÇDQA algoritması DQA algoritmasına göre daha başarılı ve hızlı sonuçlar verdiği gözlemlenmiştir. ÇDQA algoritmasında itmeye karşı robotun dengede kalmayı öğrenmesi 5500. bölümlerde gerçekleşmiştir. DQA algoritmasında ise itmelere karşı 7200. bölümlerde robot öğrenmeyi gerçekleşmiştir. Önerilen kontrolör yöntemlerinde, en iyi test sonucu ÇDQA algoritması ile olduğu gözlemlenmiştir.

Gelecekteki çalışmalarda, itme kurtarma kontrolörlerinden kalça stratejisi, adım stratejileri ve bunların faklı birleşimleri kullanılarak daha güçlü bir kontrolör tasarlanması planlanmaktadır. Ayrıca derin pekiştirmeli öğrenme algoritmalarından D3QA, BDQA, A2C, A3C, PPO gibi farklı yöntemler kullanıp itme-kurtarma kontrolörlerinin karşılaştırılması düşünülmektedir.

KAYNAKLAR

[1] B. Stephens. “Humanoid push recovery. In Humanoid Robots”, 2007 7th IEEE-RAS International Conference on, pages 589–595. IEEE, 2007.

[2] B. Stephens. “Push recovery control for force-controlled humanoid robots”. PhD thesis, Carnegie Mellon University Pittsburgh, Pennsylvania USA, 2011.

[3] Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Aral, N. Koyachi, and K. Tanie. “Planning walking patterns for a biped robo”t. IEEE Transactions on Robotics and Automation, 17(3):280–289, 2001. ISSN 1042296X. doi: 10.1109/70.938385.

[4] A. Takanishi, H. Lim, “Biped walking robots created at Waseda University: WL and Wabian family”, Philosophical Transactions of the Royal Society, Series A, 365 (1850), pp. 49-64, 2007.

[5] C. Graf and T. Röfer. “A Center of Mass Observing 3D-LIPM Gait for the RoboCupStandard Platform League Humanoid”, pages 102–113. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-32060-6. doi: 10.1007/ 978-3-642-32060-6_9. URL https://doi.org/10.1007/978-3-642-32060-6_9.

[6] T. Assman, H. Nijmeijer, A. Takanishi, and K. Hashimoto. “Biomechanically motivated lateral biped balancing using momentum control”. Eindhoven: Eindhoven University of Technology, Traineeship report. - DC 2011.035, 2012.

[7] M. Shafiee-Ashtiani, A.Yousefi-Koma M. Shariat-Panahi M. Khadiv. “Push Recovery of a Humanoid Robot Based on Model Predictive Control and Capture Point. Proceedings of the 4th International Conference on Robotics and Mechatronics October 26-28, 2016.

[8] M. Shafiee-Ashtiani, M. Yousefi-Koma, A. Mirjalili, R. Maleki, H. Karimi, . “Push Recovery of a Position-Controlled Humanoid Robot Based on Capture Point Feedback Control. 126-131. 10.1109/ICRoM.2017.8466226. 2017.

[9] S. Behnke. “Online trajectory generation for omnidirectional biped walking”. Proceedings - IEEE International Conference on Robotics and Automation, 2006(May):

[10] M. Missura and S. Behnke. “Omnidirectional capture steps for bipedal walking”. In Proceedings of Humanoids-2013, pages 401–408, Atlanta, GA, 2013. ISBN 9781479926190.

[11] M. Missura and S. Behnke. “Online learning of foot placement for balanced bipedal walking”. IEEE-RAS International Conference on Humanoid Robots, 2015-Febru: 322–328, 2015. ISSN 21640580. doi: 10.1109/HUMANOIDS.2014.7041379.

[12] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, “Capture point: A step toward humanoid push recovery,” in Proc. IEEE-RAS Int. Conf. Humanoid Robots, 2006, pp. 200–207

[13] H. Y. Ong, K. Chavez, A. Hong, “Distributed Deep Q-Learning”, Computer Science, 2015.

[14] C. Robert, T. Sotiropoulos, H. Waeselynck, J. Guiochet, S. Vernhes, “The virtual lands of Oz: testing an agribot in simulation”, *Empirical Software Engineering*, Springer Verlag, 2020.

[15] V. R. Prakash, Saran, “An Enhanced Coding Algorithm for Efficient Video Coding”. Journal of the Institute of Electronics and Computer, 1, 2019, 28-38.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, et al., “Human-level control through deep reinforcement learning”, Nature ,2015, 518:529-533.

[17] H. Kim, D. Seo and D. Kim, “Push Recovery Control for Humanoid Robot Using Reinforcement Learning”, 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, 2019, pp. 488-492, doi: 10.1109/IRC.2019.00102.

[18] Yi, Seung-Joon, et al. “Online learning of a full body push recovery controller for omnidirectional walking”. Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on. IEEE, 2011.

[19] Yi, Seung-Joon, et al. “Whole-body balancing walk controller for position controlled humanoid robots”. International Journal of Humanoid Robotics 13.01 (2016): 1650011.

[20] C. Yang, T. Komura and Z. Li, “Emergence of humancomparable balancing behaviours by deep reinforcement learning”, 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), Birmingham, 2017, pp. 372- 377, doi: 10.1109/HUMANOIDS.2017.8246900

[21] Hosseinmemar, J. Baltes, J. Anderson, M. C. Lau, C. F. Lun, and Z. Wang. “Closed-loop push recovery for inexpensive humanoid robots”. Applied Intelligence. The International Journal of Research on Intelligent Systems for Real Life Complex Problems, 49(173):1–14, 2019. ISSN 0924-669X (Print), 1573-7497 (Online). doi: 10.1007/s10489-019-01446-z.

[22] P. Ghassemi, M.T. Masouleh, A.Kalhor, “Push Recovery for NAO Humanoid Robot”, 2014 Second RSI/ISM International Conference on Robotic and Mechatronics(ICRoM), IEEE, 2014.

[23] D. C. Melo and M. R. Omena Albuquerque Maximo, A. M. Da Cunha, Push “Recovery Strategies through Deep Reinforcement Learning”, 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), Natal, Brazil, 2020

[24] R. Maulana, W. Kurniawan, H. Z. Fahmi, “Noise Reduction on the Tilt Sensor fort he Humanoid Robot Balancing System Using Complementary Filter”, 2018 The 2nd International Conferance on Mechanical, System and Control Engineering(ICMSC), 2018.

[25] C. Shu-Yin, W. Jin-Long, “Posture Control for Humanoid Robot on Uneven Ground and Slopes Using İntertial Sensors”, Advences in Mechanical Engineering, 2020.

[26] S. Yang, F. Chen, L. Zhang, Z. Cao, P. M. Wensing, Y. Liu, J. Pang, W. Zhang, “Reachability-based Push Recovery for Humanoid Robots with Variable-Height Inverted Pendulum”, 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021.

[27] R. Schuller, G. Messesan, J. Englsberger, J. Lee, C. Ott, “Online Centroidal Angular Momentum Reference Generation and Motion Optimization for Humanoid Push Recovery”, IEEE Robotics and Automation Letters, 2021.

[28] G. Messesan, J. Englsberger, C. Ott, “Online DCM Trajectory Adaptation for Push and Stumble Recovery during Humanoid Locomotion”, 2021 IEEE International Conference on Robotics and Automation (ICRA), 2021.

[29] S. Li Tzuu-Hseng, K. Ping-Huan; C. Lin-Han, H. Chia-Ching, L. Po-Chien, H. Hao-Ping, C. Chien-Hsin, H. Yi-Ting, L. Wen-Hsun, “Fuzzy Double Deep Q-Network-Based Gait Pattern Controller for Humanoid Robots”, IEEE Transactions on Fuzzy Systems, 2022.

[30] Q. Huang, C. Dong, Z. Yu, X. Chen, Q. Li, H. Chen, H. Liu, Resistant “Compliance Control for Biped Robot Inspired by Humanlike Behavior”, IEEE/ASME Transactions on Mechatronics, 2022.

[31] K. Zhu and T. Zhang, “Deep reinforcement learning based mobile robot navigation: A review,” Tsinghua Science and Technology, vol. 26, no. 5, Oct. 2021, s. 674–691.

[32] Z. Aftab, T. Robert, W. Pierre-Brice “Ankle, hip and stepping strategies for humanoid balance recovery with a single Model Predictive Control scheme.” IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012), Nov 2012, Osaka, Japan. pp.159-164, ff10.1109/HUMANOIDS.2012.6651514ff. ffhal-02487619f

[33] V. Mnih K. Kavukcuoglu, D. Silver “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, Feb. 2015, s. 529–533.

[34] Ü. Yücel, “Robotların Yapay Sinir Ağları ile Eğitilmesi,” Yüksek lisans tezi, Yıldız Teknik Üniversitesi, İstanbul, 2005.

[35] T. Komura, H. Leung, S. Kudoh, and J. Kuffner, “A feedback controller for biped humanoids that can counteract large perturbations during gait,” in Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2005, pp. 1989–1995.

[36] T. P. Lillicrap, J. J. Hunt, A. Pritzel “Continuous control with deep reinforcement learning,” ICLR 2016, (s. 1–14), 2016.

[37] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, “Capture point: A step toward humanoid push recovery,” in Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, 2006, pp. 200–207.

[38] J. Pratt, "Capture point: A step toward humanoid push recovery."2006 6th IEEE-RAS international conference on humanoid robots. IEEE, 2006.

[39] R. Oberdieck and E. N. Pistikopoulos, “Explicit hybrid modelpredictive control: The exact solution,” Automatica, vol. 58, pp. 152– 159, 2015.

[40] B. J. Stephens. “State Estimation for Force-Controlled Humanoid Balance using Simple Models in the Presence of Modeling Error”. In IEEE International Conference on Robotics and Automation, 2011.

[41] B. J. Stephens and C. G. Atkeson. “Push recovery by stepping for humanoid robots with force controlled joints”. 2010 10th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2010, pages 52–59, 2010. doi: 10.1109/ICHR.2010.5686288.

[42] U. Eren, A. Prach, B. B. Koçer, S. V. Rakovic, E. Kayacan and B. Açıkmeşe, “Model predictive control in aerospace systems: Current state and opportunities,” Journal of Guidance, Control, and Dynamics, vol. 40, no. 7, pp. 1541–1566, 2017.

[43] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas. “Dueling network architectures for deep reinforcement learning”. arXiv preprint arXiv:1511.06581, 2015.

[44] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath,. “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”. IEEE Signal Processing Magazine, 29(6):82–97, 2012.

[45] M. Yılmaz, “Humanoıd Robot Omnıdırectıonal Walkıng Trajectory Generatıon And Control”, Yüksek lisans tezi, Sabancı Üniversitesi, İstanbul, 2010.

[46] N. G. Adar “Mobil İnsansı Robot Tasarımı İmalatı Ve Kontrolü”, Doktora tezi, Sakarya Üniversitesi, Sakarya, 2016.

[47] E. Yolal, “Mobil Robot Simülatörleri Ve İleri Seviyeli Simülasyonlar”, Yüksek lisans tezi, İstanbul Teknik Üniversitesi, İstanbul, 2015.

[48] İskenderiyeli Heron. Erişim tarihi: https://tr.wikipedia.org/wiki/%C4%B0skenderiyeli_Heron (Erişim Tarihi: 2.12.2022)

[49] Cezeri. Erişim: https://tr.wikipedia.org/wiki/Cezer%C3%AE (Erişim Tarihi: 2.12.2022)

[50] Leonardo da Vinci. Erişim: https://tr.wikipedia.org/wiki/Leonardo_da_Vinci (Erişim Tarihi: 2.12.2022)

[51] Atlas Robot. Erişim: https://www.bostondynamics.com/atlas (Erişim Tarihi: 2.12.2022)

[52] Asimo. Erişim: https://global.honda/innovation/robotics/ASIMO.html (Erişim Tarihi: 2.12.2022)

[53] Robots Qrio. Erişim: https://robots.ieee.org/robots/qrio/ (Erişim Tarihi: 2.12.2022)

[54] V-Rep Robotic Simulator Testing, Erişim: https://www.youtube.com/watch?v=qPkUhDGXVoY (Erişim Tarihi: 2.12.2022)

[55] T. Terzimehic, S.Silajdzic, V. Vajnberger, J. Velagic, N. Osmic, "Path Finding Simulator for Mobile Robot Navigation", 2011 XXIII International Symposium on Information, Communication and Automation Technologies, 27-29 October 2011.

[56] Using Matlab for hardware-in-the-loop prototyping#1: Message passing system, Erişim: https://robohub.org/using-matlab-for-hardware-in-the-loop-prototyping-1-message-passing-systems/ (Erişim Tarihi: 2.12.2022)

[57] J. Fan, Z. Wang, Y. Xie, and Z. Yang, arXiv:1901.00137 [cs, math, stat] (2020), arXiv: 1901.00137.

[58] L.C. Mantilla Calderon, M. J. Junca Pelaez, "Deep Q-Learning" Universidad de Los Andes, Dogota, Colombia, 2021.

[59] K. Ogata "Modern Control Engineering". Pearson, 2010.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyad, Ad	Aslan, Emrah
Web sayfası (Research Gate, Academia, vs.)	https://akademik.yok.gov.tr/AkademikArama/view/viewAuthor.jsp

Eğitim Bilgileri

Derece	Kurum	Mezuniyet Yılı
Yüksek Lisans	Harran Üniversitesi, Elektrik-Elektronik Mühendisliği ABD	2016
Lisans	Harran Üniversitesi, Bilgisayar Mühendisliği	2013
Lise	Tarsus Cumhuriyet Lisesi	2009

İş Deneyimi

Dönem (Yıl)	Şirket, Kurum	Görev
2013-2016	GAP GreenTech Global A.Ş	Mühendis
2016-	Dicle Üniversitesi	Öğretim Görevlisi

Yabancı Dil

İngilizce

Yayınlar

1. E. Aslan, M. A. Arserim, A. Uçar "Development Of Push-Recovery Control System For Humanoid Robots Using Deep Reinforcement Learning" Ain Shams Engineering Journal, 2023,

Özel İlgiler

DİCLE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
TEZ BENZERLİK BİLDİRİMİ FORMU

Öğrencinin Adı, Soyadı	Emrah Aslan		
Öğrenci No	18805501		
Ana Bilim Dalı	Elektrik Elektronik Mühendisliği		
Program Türü	Proje ☐	Yüksek Lisans ☐	Doktora ☒
Tez Danışmanı (Ünvanı, Adı, Soyadı)	Dr. Öğr. Üyesi Muhammet Ali Arserim		
(Varsa) II. Tez Danışmanı (Ünvanı, Adı, Soyadı)	Prof. Dr. Ayşegül Uçar		
Tez Başlığı	Derin Pekiştirmeli Öğrenme Kullanarak İnsansı Robotlar İçin İtme Kurtarma Kontrol Sisteminin Geliştirilmesi		
RAPOR BİLGİLERİ			
Raporlama Aşaması	Tez Savunma Sınavı Sonrası		
Sayfa Sayısı	114		
Raporlama Tarihi	09/03/2023		
Benzerlik Oranı (%)	14		

Yukarıda bilgileri verilen tez çalışmamın toplam X sayfalık kısmına ilişkin, 9/03/2023 tarihinde tez danışmanım tarafından *Turnitin* isimli intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan intihal raporuna göre, tezimin benzerlik oranı %14 olarak tespit edilmiştir.

Uygulanan filtrelemeler:

☐Başlangıç Bölümleri (Kabul ve Onay sayfası, Teşekkür sayfası, Özet/Abstract) hariç

☐Kaynaklar hariç

☐Alıntılar hariç/dâhil

☒Diğer (Her şey dahil)

Tezimin benzerlik oranı, Dicle Üniversitesi Fen Bilimleri Enstitüsü İntihal Raporu Uygulama Esaslarında belirtilen üst sınır benzerlik oranını aşmamaktadır. Benzerlik oranım üst sınır benzerlik oranının altında olsa dahi aksinin tespit edilmesi durumunda her türlü yasal sorumluğu kabul ettiğimi ve hukuki sonuçlarına razı olduğumu bildirir,
gereğini arz ederim.

Öğrencinin Adı, Soyadı: Emrah Aslan
Tarih: 09/03/2023
İmza:

Danışman Adı, Soyadı: Dr. Öğr. Üyesi Muhammet Ali Arserim Tarih: 09/03/2023	İmza:
Ana Bilim Dalı Başkanı Adı, Soyadı: Doç. Dr. Bilal Gümüş Tarih: 09/03/2023	İmza: