T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İNSANSI ROBOTLARIN YÜRÜME BECERİLERİNİN DERİN PEKİŞTİRMELİ ÖĞRENME ALGORİTMALARIYLA GELİŞTİRİLMESİ

Çağrı KAYMAK

Doktora Tezi

MEKATRONİK MÜHENDİSLİĞİ ANABİLİM DALI

Mekatronik Mühendisliği Bilim Dalı

MAYIS 2023

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Mekatronik Mühendisliği Anabilim Dalı

Doktora Tezi

İNSANSI ROBOTLARIN YÜRÜME BECERİLERİNİN DERİN PEKİŞTİRMELİ ÖĞRENME ALGORİTMALARIYLA GELİŞTİRİLMESİ

Tez Yazarı
Çağrı KAYMAK

Danışman
Prof. Dr. Ayşegül UÇAR

MAYIS 2023
ELAZIĞ

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Mekatronik Mühendisliği Anabilim Dalı

Doktora Tezi

Başlığı:	İnsansı Robotların Yürüme Becerilerinin Derin Pekiştirmeli Öğrenme Algoritmalarıyla Geliştirilmesi
Yazarı:	Çağrı KAYMAK
İlk Teslim Tarihi:	17.04.2023
Savunma Tarihi:	24.05.2023

TEZ ONAYI

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

		İmza
Danışman:	Prof. Dr. Ayşegül UÇAR Fırat Üniversitesi, Mühendislik Fakültesi	Onayladım
Başkan:	Prof. Dr. Tülay YILDIRIM Yıldız Teknik Üniversitesi, Elektrik Elektronik Fakültesi	Onayladım
Üye:	Prof. Dr. Yakup DEMİR Fırat Üniversitesi, Mühendislik Fakültesi	Onayladım
Üye:	Prof. Dr. Mehmet KARAKÖSE Fırat Üniversitesi, Mühendislik Fakültesi	Onayladım
Üye:	Dr. Öğr. Üyesi Emrehan YAVŞAN Tekirdağ Namık Kemal Üniversitesi, Teknik Bilimler MYO	Onayladım

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

İmza

Prof. Dr. Burhan ERGEN
Enstitü Müdürü

BEYAN

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

24.05.2023

Çağrı KAYMAK

ÖNSÖZ

İnsansı robotlar için kararlı bir hareket geliştirmek, onlarca yıldır araştırılan zorlu bir problemdir. Çeşitli yürüme yaklaşımları önerilmiş ve yürüme performansı önemli ölçüde geliştirilmiş olsa da, kararlılık konusunda hala beklentilerin gerisinde kalmaktadır.

Bu tez çalışmasında, Derin Pekiştirmeli Öğrenme (DPÖ) algoritmaları kullanılarak insansı robotların düz ve eğimli yüzeylerde kararlı bir şekilde yürüyebilmesinin geliştirilmesi hedeflenmektedir. Bu hedef doğrultusunda, Robotis-OP2 insansı robotu için geleneksel yörünge üreteci kontrolör ve DPÖ ile birleştirilmiş etkili bir çerçeve önerilerek yürüyüş şablonu üretici ile oluşturulan yörüngelerin optimum yürüyüş parametreleri elde edilmektedir. Ayrıca sagital düzlemde robot duruş dengeleme sistemi önerilerek robotun eğimli yüzeylerde yürüyebilmesi için PID kontrolör ve DPÖ kontrolörden oluşan iki ayrı yürüyüş sistemi önerilmektedir.

Doktora tez çalışma konumun seçiminde ve tez çalışmalarımın tamamlanmasında sağlamış olduğu katkıların yanı sıra akademik hayatım boyunca desteğini esirgemeyen danışmanım Sayın Prof. Dr. Ayşegül UÇAR hocama çok teşekkür ederim.

Hayatıma girdiği andan itibaren desteğini hiçbir zaman esirgemeyen sevgili eşim Rüya KAYMAK'a da tez çalışmalarım süresince göstermiş olduğu manevi desteğinden dolayı sonsuz teşekkür ederim.

Son olarak da, hayatım boyunca her daim yanımda olan, sevgi ve desteklerini hiçbir zaman esirgemeyen aileme teşekkürü borç bilir, saygılarımı sunarım.

Bu tez çalışması, Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK) tarafından **117E589** numaralı proje ile desteklenmiştir.

Çağrı KAYMAK
ELAZIĞ, 2023

İÇİNDEKİLER

Sayfa

ÖZET

İnsansı Robotların Yürüme Becerilerinin Derin Pekiştirmeli Öğrenme Algoritmalarıyla Geliştirilmesi

Çağrı KAYMAK

Doktora Tezi

FIRAT ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

Mekatronik Mühendisliği Anabilim Dalı

Mayıs 2023, Sayfa: xxii + 233

İnsansı robotlar için sağlam bir hareket geliştirmek, onlarca yıldır araştırılan zorlu bir problemdir. Çeşitli yürüme yaklaşımları önerilmiş ve yürüme performansı önemli ölçüde geliştirilmiş olsa da, kararlılık konusunda hala beklentilerin gerisinde kalmaktadır.

Pekiştirmeli öğrenme yaklaşımları için düşük yakınsama ve eğitim verimliliği, uygulamaları sınırlandırmaktadır. Bu sınırlamaların üstesinden gelmek için bu tez çalışmasında, Robotis-OP2 insansı robotuna dayalı olarak geleneksel yörünge üreteci kontrolör ve Derin Pekiştirmeli Öğrenme (DPÖ) ile birleştirilmiş etkili bir çerçeve önerilmiştir. Bu çerçeve, oluşturulan yürüyüş yörüngesi parametrelerinin optimizasyonu ve duruş dengeleme sisteminden oluşmaktadır. Webots simülatöründe DPÖ algoritmalarından Düello Çift Derin Q Ağı (Düello ÇDQA) kullanılarak yürüyüş parametreleri optimize edilmiştir. Duruş dengeleme sistemi için kalça stratejisi benimsenmiştir. Önerilen çerçeve ve Robotis-OP2 insansı robotunun kendi yürüme algoritması ile hem simülasyon hem de gerçek ortamda deneysel çalışmalar gerçekleştirilmiştir. Deneysel sonuçlar, robotun önerilen çerçeve ile robotun kendi yürüme algoritmasına göre düz yürüme görevinin daha kararlı bir şekilde gerçekleştirildiğini göstermiştir.

Tez çalışması kapsamında daha sonra, robotun eğimli yüzeylerde kararlı bir şekilde yürüyebilmesi için PID kontrolör ve DPÖ kontrolörden oluşan iki ayrı yürüyüş dengeleme çerçevesi önerilmiştir. DPÖ kontrolör olarak DDPG (Derin Deterministik Politika Gradyanı) algoritması tercih edilmiştir. PID kontrolör ile gerçekleştirilen deneysel çalışmalarda robotun duruşu gerçek zamanlı olarak ayarlanarak gövde yunuslama açısının istenilen referans değerde olması sağlanmıştır. DDPG kontrolör ile robotun eğimli yüzeylerde dengeli yürüyüşünün sağlanabilmesi için robotun gövde yunuslama açısının sıralı hareket dizisi öğrenilmiştir. Deneysel sonuçlar, DPÖ kontrolörün PID kontrolöre göre daha kullanışlı olduğunu ve daha kararlı bir yürüme sağladığını göstermiştir.

Anahtar Kelimeler: İnsansı robot, Kararlı yürüme, Parametre optimizasyonu, Eğimli yüzeylerde yürüme, Derin pekiştirmeli öğrenme

ABSTRACT

Developing Walking Skills of Humanoid Robots with Deep Reinforcement Learning Algorithms

Çağrı KAYMAK

Ph.D. Thesis

FIRAT UNIVERSITY
Graduate School of Natural and Applied Sciences

Department of Mechatronics Engineering

May 2023, Pages: xxii + 233

Developing robust locomotion for humanoid robots is a challenging problem that has been researched for decades. Although various walking approaches have been proposed and walking performance has been significantly improved, it still falls short of expectations in stability.

Low convergence and training efficiency for reinforcement learning approaches limit their applications. To overcome such limitations, an effective framework based on Robotis-OP2 humanoid robot combined with traditional trajectory generator controller and Deep Reinforcement Learning (DRL) is proposed in this thesis. This framework consists of the optimization of the gait trajectory parameters and the posture stabilization system. In the Webots simulator, gait parameters are optimized using the Dueling Double Deep Q Network (D3QN), one of the DRL algorithms. The hip strategy is adopted for the posture balancing system. Experimental studies are carried out in both simulation and real environment with the proposed framework and the Robotis-OP2 humanoid robot's own walking algorithm. Experimental results show that the robot performs the task of straight walking with the proposed framework more stable than the own algorithm of the robot.

Later, within the scope of the thesis, two separate gait stabilization frameworks, consisting of a PID controller and a DRL controller, are proposed for the robot to walk stably on sloped surfaces. The DDPG (Deep Deterministic Policy Gradient) algorithm is preferred as the DRL controller. In the experimental studies performed with the PID controller, the posture of the robot was adjusted in real-time to ensure that the body pitch angle is at the desired reference value. With the DDPG controller, the sequential movement sequence of the robot's body pitch angle is learned in order to ensure the robot's balanced walking on inclined surfaces. Experimental results show that the DRL controller is more useful than the PID controller and provides a more stable gait.

Keywords: Humanoid robot, Stable walking, Parameter optimization, Walking on sloped surfaces, Deep reinforcement learning

ŞEKİLLER LİSTESİ

Sayfa

TABLOLAR LİSTESİ

Sayfa

SİMGELER

$A_{iv.}$: İvmeölçerden hesaplanan açı değeri
A_t : Tamamlayıcı filtre ile filtrelenen açı değeri
a_x : x eksenindeki doğrusal ivme
a_y : y eksenindeki doğrusal ivme
a_z : z eksenindeki doğrusal ivme
D_x : x ekseninde (ileri yön) alınan mesafe
h_{ak} : Ayak bileği eklemi ile kalça eklemi arasındaki yükseklik
h_b : Kalça ekleminin yerden yüksekliğini ayarlamada kullanılan bükülme yüksekliği
K_d : Türevsel kazanç katsayısı
K_i : İntegral kazanç katsayısı
K_p : Oransal kazanç katsayısı
p_x : x eksenindeki konum
p_y : y eksenindeki konum
p_z : z eksenindeki konum
$Q_{\dot{\theta}_b}$: Bias gürültü varyansı
$Q_{\dot{\theta}_b}$: Ölçüm gürültü varyansı
Q_θ : İvmeölçer gürültü varyansı
R_z : z ekseninde dönme
SMN_x : x eksenindeki SMN koordinatı
SMN_y : y eksenindeki SMN koordinatı
T_0^N : Temel dönüşüm matrisi
t_d : Çift ayak destek durumunda ağırlık merkezinin kayma zamanı
$V_{ref.}$: Referans gerilim
$V_{sıfır}$: İvme yok iken gerilim
$x_{sağ\_ab}$: Sağ ayak bileği ekleminin x ekseni yörüngesi
$x_{sağ\_ka}$: Sağ kalça ekleminin x ekseni yörüngesi
x_{sol_ab} : Sol ayak bileği ekleminin x ekseni yörüngesi
x_{sol_ka} : Sol kalça ekleminin x ekseni yörüngesi
$y_{sağ\_ab}$: Sağ ayak bileği ekleminin y ekseni yörüngesi
$y_{sağ\_ka}$: Sağ kalça ekleminin y ekseni yörüngesi
y_{sol_ab} : Sol ayak bileği ekleminin y ekseni yörüngesi
y_{sol_ka} : Sol kalça ekleminin y ekseni yörüngesi
$z_{sağ\_ab}$: Sağ ayak bileği ekleminin z ekseni yörüngesi
$z_{sağ\_ka}$: Sağ kalça ekleminin z ekseni yörüngesi
z_{sol_ab} : Sol ayak bileği ekleminin z ekseni yörüngesi
z_{sol_ka} : Sol kalça ekleminin z ekseni yörüngesi
θ_i : i. eklem açısı
$\theta_{iv.}$: İvmeölçerden hesaplanan yunuslama açı değeri
$\theta_{ölç}$: Ölçülen yunuslama açısı
θ_r : θ yunuslama açısını belirleyen açı değeri
$\theta_{ref.}$: Referans yunuslama açısı
θ_t : Filtrelenen yunuslama açı değeri
θ_{t-1} : Filtrelenen bir önceki yunuslama açı değeri

$\theta_{ivmeölçer}$: İvmeölçer ile hesaplanan yunuslama açısı

$\theta_{jiroskop}$: Jiroskop ile hesaplanan yunuslama açısı

$\Phi_{\text{iv.}}$: İvmeölçerden hesaplanan yuvarlanma açı değeri

Φ_{t} : Filtrelenen yuvarlanma açı değeri

$\Phi_{\text{t}-1}$: Filtrelenen bir önceki yuvarlanma açı değeri

$\Phi_{ivmeölçer}$: İvmeölçer ile hesaplanan yuvarlanma açısı

$\Phi_{jiroskop}$: Jiroskop ile hesaplanan yuvarlanma açısı

ω_{x} : x eksenindeki açısal hız

ω_{y} : y eksenindeki açısal hız

ω_{z} : z eksenindeki açısal hız

A^{π} : Avantaj fonksiyonu

G_t : Getiri

$Q^*(s,a)$: Optimal durum-eylem değer fonksiyonu

$Q^{\pi}(s,a)$: Durum-eylem değer fonksiyonu (Q fonksiyonu)

R_t : Getiri

$V^*(s)$: Optimal değer fonksiyonu

$V^{\pi}(s)$: Durum değer fonksiyonu

$V^{\pi}(s_t)$: t zamanındaki durum değer fonksiyonu

$V^{\pi}(s_{t+1})$: t zamanından sonraki durum değer fonksiyonu

W_t : t'ye bağlı sürekli fonksiyon

$Y_t^{ÇDQA}$: ÇDQA'nın hedef değeri

Y_t^{DQA} : DQA'nın hedef değeri

a' : t zamanından sonraki eylem (Epsilon-açgözlü politikası ile seçilen eylem)

a_k : k zamanındaki eylem

a_{max} : En fazla ödülün alındığı eylem

a_t : t zamanındaki eylem

a_{t+1} : t zamanından sonraki eylem

o_t : t zamanındaki gözlem

r_T : T son zaman adımında alınan ödül

r_k : k zamanındaki ödül

r_t : t zamanındaki ödül

r_{t+1} : t zamanından sonraki ödül

s' : t zamanından sonraki durum

s_k : k zamanındaki durum

s_{k+1} : k zamanından sonraki durum

s_t : t zamanındaki durum

s_{t+1} : t zamanından sonraki durum

w_i : i. nöron ağırlığı

x_i : i. nöron girişi

θ^- : Hedef ağ parametresi

μ^* : Optimal deterministik politika

π^* : Optimal stokastik politika

$\oslash$: Eleman bazında bölme

$\odot$: Eleman bazında çarpma

A : Homojen dönüşüm matrisi

DSR : Çift destek oranı

dt	: Örnekleme periyodu
$\mathrm{e(t)}$	: Hata
h	: En büyük ayak kaldırma yüksekliği
$\mathcal{L}$	: Kayıp fonksiyonu
M	: Moment
p	: Tahmini hata varyansı
q	: İşlem gürültü varyansı
r	: Ölçüm gürültü varyansı
s	: Bir adım uzunluğu
T	: Yürüme çevrimi (Faz 3-Faz 8) zamanı
$\mathrm{u(t)}$	: Kontrol sinyali
w	: Kalça salındığında sağa-sola en büyük öteleme
θ	: Yunuslama açısı
Φ	: Yuvarlanma açısı
ω	: Jiroskoptan alınan açısal hız değeri
A	: Eylem uzayı
J	: Başlangıç dağılımı
M	: Kapasite
P	: Geçiş olasılığı fonksiyonu
$Q(s, a;\ \theta)$	: θ ağ parametreleri olan Q fonksiyonu
R	: Ödül fonksiyonu
S	: Durum uzayı
a	: Eylem
b	: Bias
s	: Durum
t	: Anlık zaman
y	: Nöron çıkışı
$\mathcal{D}$	: Tekrar tamponu
$\mathcal{N}$	: Örneklenen gürültü
α	: Öğrenme oranı
γ	: İndirim faktörü
η	: Öğrenme oranı
μ	: Deterministik politika
π	: Stokastik politika
τ	: Yumuşak güncelleme faktörü

KISALTMALAR

3B : 3 Boyutlu
A3K : Asenkron Avantaj Aktör-Kritik (Asynchronous Advantage Actor-Critic–A3C)
ADC : Analog-Dijital Çevirici (Analog to Digital Converter)
AGF : Alçak Geçiren Filtre (Low-Pass Filter–LPF)
ANFIS : Uyarlanabilir Ağ Tabanlı Bulanık Çıkarım Sistemi (Adaptive Network-based Fuzzy Inference System)
AÖB : Atalet Ölçüm Birimi (Inertial Measurement Unit–IMU)
API : Uygulama Programlama Arayüzü (Application Programming Interface)
Ar-Ge : Araştırma-Geliştirme
BM : Basınç Merkezi (Center of Pressure–CoP)
CPU : Merkezi İşlem Birimi (Central Processing Unit)
ÇDQA : Çift Derin Q Ağı (Double Deep Q Network–DDQN)
D3QN : Düello Çift Derin Q Ağı (Dueling Double Deep Q Network)
DDB : Düzeltilmiş Doğrusal Birim (Rectified Linear Unit–ReLU)
DDPG : Derin Deterministik Politika Gradyanı (Deep Deterministic Policy Gradient–DDPG)
DE : Diferansiyel Evrim (Differential Evolution–DE)
D-H : Denavit-Hartenberg
DoF : Serbestlik Derecesi (Degrees of Freedom)
DP : Dinamik Programlama (Dynamic Programming–DP)
DPG : Deterministik Politika Gradyanı (Deterministic Policy Gradient–DPG)
DPÖ : Derin Pekiştirmeli Öğrenme (Deep Reinforcement Learning–DRL)
DQA : Derin Q Ağı (Deep Q Network–DQN)
DSA : Derin Sinir Ağı (Deep Neural Network–DNN)
DSP : Çift Destek Fazı (Double Support Phase)
ESA : Evrişimsel Sinir Ağı (Convolutional Neural Network–CNN)
GBPO : Güven Bölgesi Politika Optimizasyonu (Trust Region Policy Optimization–TRPO)
GPU : Grafik İşlem Birimi (Graphical Processing Unit)
IDE : Tümleşik Geliştirme Ortamı (Integrated Development Environment)
KDD : Kuvvete Duyarlı Direnç (Force Sensitive Resistor–FSR)
KL : Kullback-Leibler
LORM : Referans Hareketi Öğren ve Gerçekleştir (Learn and Outperform the Reference Motion)
M-A3C : Ortalama-Eşzamansız Avantajlı Aktör-Eleştirmen (Mean-Asynchronous Advantage Actor-Critic)
MBMF : Model-Based Priors for Model-Free
MC : Monte Carlo
MKS : Markov Karar Süreci (Markov Decision Process–MDP)
MÖÜ : Merkezi Örüntü Üreteci (Central Pattern Generator–CPG)
ODE : Open Dynamics Engine
O-U : Ornstein-Uhlenbeck
PG : Politika Gradyanı (Policy Gradient–PG)
PÖ : Pekiştirmeli Öğrenme (Reinforcement Learning–RL)
ROS : Robot İşletim Sistemi (Robot Operating System)
SARSA : Durum-Eylem-Ödül-Durum-Eylem (State-Action-Reward-State-Action)
SDF : Simulator Description Format
SMN : Sıfır Moment Noktası (Zero Moment Point–ZMP)

TD3	: İkiz Gecikmeli DDPG (Twin Delayed Deep Determinstic Policy Gradient)
UCB	: Üst Güven Sınırı (Upper Confidence Bound)
VRML	: Virtual Reality Modeling Language
YAK	: Yumuşak Aktör-Kritik (Soft Actor-Critic)
YAML	: YAML Ain’t a Markup Language
RDDPG	: Yinelemeli DDPG (Recurrent Deep Deterministic Policy Gradient)
YGF	: Yüksek Geçiren Filtre (High-Pass Filter–HPF)
YPO	: Yakınsal Politika Optimizasyonu (Proximal Policy Optimization–PPO)
YSA	: Yapay Sinir Ağı
ZF	: Zamansal Fark (Temporal Difference–TD)

1. GİRİŞ

Günümüzde robotlar, otomasyon sistemlerinden savunma sanayisine kadar birçok alanda kullanılmaktadır. Böylece robotik, temel yapı taşı haline gelmiş ve günlük yaşamda birçok farklı eylemde kendine yer bulmuştur. Son yıllarda, yaşam ihtiyaçlarının hızlı gelişimine uyum sağlayabilmek için robotik alanındaki Ar-Ge (Araştırma-Geliştirme) çalışmaları sürekli iyileştirilmektedir [1]. Robotlar, insanların günlük işlerini kolaylıkla yerine getirebilmeleri için yapay zeka ve öğrenme algoritmaları ile sürekli gelişim ve iyileştirmeye tabidir.

Robotik sistemler, sabit bir çalışma uzayına sahip olan manipülatör robotlar ve taşınabilir bir çalışma uzayına sahip olan mobil robotlar olarak iki alanda sınıflandırılabilir. Mobil robotlar; tekerlekli, paletli ve bacaklı robotlar olarak alt sınıflara ayrılabilir. Tekerlekli robotlar, çok hızlı hareket edebilirler fakat düz bir arazi üzerinde hareket yetenekleri vardır. Paletli robotlar, daha engebeli arazilerde hareket edebilirler ancak tekerlekli robotlara göre daha yavaştırlar. Bacaklı robotlar, daha fazla hareketlilik ve esneklik gösterdiğinden, arazi farklılıklarına yüksek uyuma sahip olduklarından ve çevreye daha az hasar verdiklerinden dolayı tekerlekli ve paletli robotlara göre zorlu arazilerde daha üstündür. Bu üstünlük ve yeryüzünün yaklaşık %80'inin geleneksel tekerlekli taşıtlar ile erişilemez olduğu göz önüne alındığında bacaklı robotlar, mobil robotlar alanında daha çok ön plana çıkmaktadır [2].

Bacaklı robotlar bacak sayılarına göre sınıflandırılırlar. Şekil 1.1'den görüleceği gibi bacaklı robotlar genellikle iki, dört veya altı bacaklı tasarımlardan oluşmaktadır. Aralarındaki ortak nokta, hareketlerinin hassasiyetini bir geri besleme döngüsü ile kontrol etmek için kademeli motorların kullanılmasıdır. Bir robotun bacaklarının sayısı ve dengesi, birbiriyle ilişkilidir.

(a) (b) (c)

Şekil 1.1. Bacaklı robot örnekleri: (a) iki bacaklı, (b) dört bacaklı ve (c) altı bacaklı

Endüstride robotların başarılı bir şekilde kullanılması, 1980'lerden sonra insan hareketlerini taklit edebilen insansı robotlarla ilgili çalışmalara başlanmasını sağlamıştır. İnsansı robotlar; insana benzeyen bir tasarım ile yapılan, hareket edebilme ve konuşabilme gibi farklı fonksiyonlara sahip olan iki bacaklı robotlardır. İnsanlara olan benzerlikleri, insanlar için uygun olan ortamlarda hareket edebilme ve insanlar için tasarlanan araçları kullanabilme yetenekleri sebebiyle mobil robotlar içinde önemli bir yere sahiptirler. İnsansı bir robotun ana özelliği, çevresini değiştirmede ve insanların zeka, davranış ve eylemlerdeki fiziksel ve zihinsel görevlerini taklit etmede özerk uyumudur. Yeterli teknolojik gelişmelerin gerçekleşmesi halinde, şu anda insanların çalıştığı her iş alanında insansı robotları görmek mümkün olacaktır.

İnsansı robotların hareketleri, alt vücut ve üst vücut hareketleri olarak iki kısımda incelenebilir. Alt vücut hareket çalışmalarında, insansı robotun dengeli bir şekilde yürüme veya koşma problemleri ele alınırken, üst vücut hareket çalışmalarında kol ve vücudun kullanılmasıyla daha dengeli yürümesi ve çevresiyle etkileşime geçerek belli görevleri gerçekleştirmesi ele alınır. İnsansı robotlar, genel olarak alt vücut hareket çalışmalarını içeren yürüme ve koşma gibi temel insan hareket mekanizmaları üzerine araştırmalar yapılmak üzere geliştirilmektedir.

İnsan yürüyüşü teknik olarak iki ayaklı yürüme olarak adlandırılır. İnsansı robotların yer değiştirme sistemleri de, insanlarda olduğu gibi iki ayaklı yürümeye dayanır. Standart bir insanın yürüyüşünde, uzuvlar ve gövde vasıtasıyla ağırlık merkezinin öne doğru yönelimiyle gerçekleşen hareketlerin bütününden oluşmaktadır. Yürüme esnasında yer değiştirmeler ve yük bacaklar tarafından gerçekleştirilmekte olup, vücudun geri kalan kısmı ise bacakların hareketinden ileriye doğru kayan ağırlık merkezi değişimlerine uyum sağlayarak ilerlemektedir.

Biyomekanik araştırma açısından, iki ayaklı kararlılığı ve yürüme mekanizmasını anlamak, insanların bir yerden diğerine nasıl geçtiğini daha iyi anlamak için önemli bir temel oluşturur [3]. İnsan hareketi, basit görünmesine rağmen, alt vücuttaki çeşitli ekstansör ve fleksör kas grupları nedeniyle üretilen karmaşık doğrusal olmayan dinamiklerle birleştirilen birden fazla serbestlik derecesini (Degrees of Freedom–DoF) içeren oldukça karmaşık bir durumdur. İnsansı robotlar için sağlam bir hareket geliştirmek, onlarca yıldır araştırılan zorlu bir problemdir. Çeşitli yürüme yaklaşımları önerilmiş ve yürüme performansı önemli ölçüde geliştirilmiş olsa da, hız ve kararlılık gibi bazı alanlarda hala beklentilerin gerisinde kalmaktadır. İnsansı robotlar, geniş bir arazi yelpazesinde hareket etme kolaylıkları ve esneklikleriyle bilinirken doğrusal olmayan ve yüksek boyutlu bir sisteme sahip olduğundan temel endişe kararlılıktır. İki ayaklı yürüme sisteminin kararlılık sorununu çözmek, yıllar boyunca birçok kontrol bilimcisi arasında merak uyandırmıştır [4].

Geleneksel kontrol teorisi yaklaşımları, karmaşık deterministik ve matematiksel mühendislik modellerine dayanır. Ters sarkaç, [5-8]'dekiler gibi birçok algoritmaya ilham veren en uyarlanmış modellerden biridir. Sıfır Moment Noktası kısaca SMN (Zero Moment Point–ZMP), iki ayaklı

robotlarda dinamik kararlılık için bir gösterge olarak kabul edilen geleneksel yöntemlerden biridir. SMN, yatay atalet ve yerçekimi kuvvetlerinin toplamının sıfıra eşit olduğu nokta olarak tanımlanır. Robot SMN'ye ulaştığında, robotun ayağının zemine tepkisi, robotun hareketinin neden olduğu dinamikleri telafi eder ve robotun dinamik dengesi korunur [9]. SMN'yi elde etmek için robotun kütle merkezi hesaplanır. Bu nedenle ya simülasyon hesaplamaları ya da robotun ayaklarına takılan kuvvet sensörleri gereklidir. Ancak her adım için SMN'nin hesaplanması, sınırlı sistem kaynaklarına ve hesaplama hızlarına sahip küçük bir insansı robot için zaman alıcıdır. Ayrıca geleneksel kontrol yöntemleri, hem robot hem de arazi için dinamiklere ve matematiksel modellere büyük ölçüde dayanır ve tasarımcılar için büyük miktarda zaman ve emek gerektirir. Robotun tipi veya arazinin özelliği değiştiğinde modelin yeniden tasarlanması gerekir. Bu durum, aynı zamanda yürüyüş kontrolörünün uyum eksikliği ile de sonuçlanır. Ek olarak, insan tarafından tasarlanan modelin performansı, robotun potansiyelini tam olarak keşfedemeyen tasarımcıların ön bilgi ve deneyimleri ile de sınırlıdır.

Son zamanlarda önerilen yürüyüş çerçeveleri incelendiğinde kararlı bir yürüyüşün gelişimi ile ilgili yaklaşımlar mevcuttur. Temel çerçeve, yürüme planlayıcısının ve kontrolörünün tasarlandığı bir robotun dinamik modeline dayanmaktadır. Bu tür bir çerçevede, tüm vücut dinamiği modeli geliştirmenin karmaşıklığını azaltmak için bazı kısıtlamalar göz önünde bulundurulur. Bu kısıtlamalara dayalı olarak, gerçek bir tüm vücut dinamiği modeli yerine soyut bir model tasarlanmıştır [10-16]. Tüm vücut dinamiği modelinin geliştirildiği birçok çalışma da mevcuttur [17-19]. Diğer bir çerçeve, içsel olarak ritmik sinyaller üretmek için birbirine bağlanan bir dizi sinyal üretecidir [20-23]. Bu tür çerçeve, Merkezi Örüntü Üreteci kısaca MÖÜ (Central Pattern Generator–CPG) tabanlıdır. Omurgasız ve omurgalı hayvanlar üzerinde yapılan nörofizyolojik çalışmalardan esinlenilmiştir [24-26]. Bu çalışmalar yürüme, koşma ve yüzme gibi ritmik hareketlerin belirli bir düzende birbirine bağlanan omurilikteki MÖÜ'ler tarafından üretildiğini göstermektedir. Bu çerçevede, osilatörler tipik olarak ayar noktalarını (konum, tork, vb.) oluşturmak için her bir uzva atanır. Çoğu insansı robot, yirmiden fazla serbestlik derecesine sahiptir, bu nedenle osilatörlerin parametrelerini ayarlamak hem zor hem de deneme açısından yoğundur. Ayrıca duyusal bilgiyi osilatörlere uyarlamanın kolay bir yolu da yoktur. Bahsedilen bu iki çerçeve türünde, robotların bilgisi genel olarak statiktir ve geçmiş deneyimlerden öğrenmez. Bu nedenle, yeni çevrelere uyum sağlayabilmek için en azından parametrelerin yeniden ayarlanması gerekir. Diğer bir çerçeveye göre yürüme yörüngelerinin oluşturulması, Pekiştirmeli Öğrenme kısaca PÖ'ye (Reinforcement Learning–RL) veya bir buluşsal algoritmaya [20, 27, 28] dayanmaktadır. Bu tür çerçevede yürüme yörüngeleri, çok sayıda örnek gerektiren ve önemli miktarda zaman alan bir eğitim döneminden sonra oluşturulur. Eğitim sırasında çerçeve, bir nesnel fonksiyona bağlı olarak yürüme yörüngelerinin nasıl oluşturulacağını öğrenmeye çalışır. Bir diğer çerçeve ise yukarıda bahsedilen yaklaşımların birleştirilmesiyle tasarlanır [29-33]. Bu tür çerçeve

bir hibrit yürüme çerçevesi olarak bilinir. Nihai performansı iyileştirmek için her yaklaşımın farklı yeteneklerinden yararlanmaya çalışır.

Geleneksel model tabanlı kontrolörlerin uzun bir geçmişi vardır ve robotların kontrolünde kısmen başarıya ulaşmıştır. Kasaei ve diğerleri [29], kapalı döngü model tabanlı bir yürüme çerçevesi önermiştir. Dinamik modelleri, robotun alt ve üst vücut dinamiklerini dikkate alan iki kütleden oluşur. Bu dinamik modele dayanarak, yürüme referans yörüngeleri oluşturulmuş ve ayrıca bu referansları izlemek için optimal bir kontrolör tasarlanmıştır. SimSpark'ta simülasyonu yapılmış bir NAO insansı robotu kullanarak bir dizi simülasyon gerçekleştirerek çerçevelerinin performansını göstermişlerdir. Ayrıca, genetik algoritma kullanarak parametreleri optimize etmişlerdir. Carpentier ve diğerleri [34], dinamik olarak tutarlı hareketler üretebilen genel ve verimli bir yürüme modeli üreteci önermiştir. Yaklaşımlarının, önceki adım yürütülürken verilen temas yapılandırmasına göre açısal momentum ile birlikte kütle merkezi yörüngesini oluşturmak için yeterince hızlı olduğunu savunmuşlardır. Yöntemleri, ilgisini göstermek için gerçek bir HRP-2 robotu üzerinde uygulanmıştır. Deney sonuçları, yöntemlerinin tırabzan desteğiyle uzun adımlı yürüme ve merdiven çıkmayı üretebildiğini göstermiştir. Song ve diğerleri [35], bilinmeyen eğimli yüzeylerde bile dengeli yürüme üretebilen MÖÜ tabanlı kontrol yürüme çerçevesi tasarlamıştır. Çerçevelerinde, yürüme şablonları MÖÜ teorisine dayalı olarak üretilmiştir. Jiroskop ve ivmeölçer bilgilerine göre bir PI kontrolör tasarlanmış ve robotun dengesini korumak için üst gövdenin eğim açısının ayarlanmasına izin vermiştir. Gerçek bir NAO insansı robotu kullanarak bazı deneyler yapılmış ve sonuçlar robotun bilinmeyen eğimlerde başarılı bir şekilde yürüyebildiğini göstermiştir. [36]'da, NAO insansı robotunun yürüme hızının fiziksel düzenini değiştirmeden geliştirilip geliştirilemeyeceği hipotezini test etmek amaçlanmıştır. Uygulanan araştırma yöntemi, NAO'nun en iyi yürüme hızını ölçmek ve tahmin etmek için Yapay Sinir Ağı kısaca YSA, Naive Bayes ve Karar Ağacı yöntemlerinden en iyisini seçip optimal hızı bulmak için geliştirilmesini içermiştir. Massa ve diğerleri [37], insansı robotlar için sabit hareket oluşturmak için bir hibrit MÖÜ-SMN kontrolörü geliştirmiştir. Yaklaşımlarında, yürüme yörüngeleri oluşturmak için bir dizi doğrusal olmayan osilatör kullanılmıştır. Ayrıca küçük ve büyük rahatsızlıkları ele almak için iki kontrolör geliştirilmiş ve Diferansiyel Evrim kısaca DE (Differential Evolution–DE) algoritması kullanılarak yürüme parametreleri optimize edilmiştir. Yaklaşımlarının performansı, NAO insansı robotu kullanan Webots [38] robotik simülatöründe gösterilmiştir.

İnsanların günlük yaşamlarında olduğu gibi iki ayaklı bir robotun yalnızca düz yüzeylerde değil, aynı zamanda çeşitli engebeli arazilerde de uyumlu bir şekilde yürümesi beklenir. Düz yüzeyde yürümenin yanı sıra, eğimli yüzeylerde, engebeli arazilerde veya dış kuvvetlerin bozucu etkisi ile yürüme planlamasını çözmek için bazı geleneksel yöntemler önerilmiştir [39-44]. Morisawa ve diğerleri [40], robotun engebeli arazide yürümesini sağlayan, Yakalama Noktası (Capture Point) kontrolörüne dayalı gelişmiş bir denge kontrol algoritması önermiştir. [41],

çevrimiçi arazi tahminine dayalı olarak düzlükte orijinal yürüyüşe yapılan değişikliği en aza indirirken robotun bilinmeyen engebeli arazide yürümesini sağlayan bir algoritma önermiştir. [42]'deki çalışmada, denge için benzer modellere dayanarak insansı robotun bir skuter sürmesini sağlayan bir algoritma önerilmiştir.

Yukarıda bahsedilen çalışmalardaki geleneksel kontrol yöntemlerinin ana dezavantajı, modelin eklem sürtünmesinden, zeminle temas kuvvetinden veya diğer belirsizliklerden etkilenebileceği bir matematiksel modelin doğruluğuna yüksek oranda güvenmeleridir. Normal şartlarda eğimli veya pürüzlü bir yüzeyin bağımsız olarak modellenmesi gerekir. Bu da hesaplama açısından oldukça zorlayıcı olabilir ve elde edilen model yeterince doğru olmayabilir. Belirli bir ortam için tasarlanan kontrolör, farklı ortam türlerine uyum sağlaması konusunda yeterince genel değildir. Özet olarak, insansı robotların esnek olmasını ve farklı arazilere uyarlanabilmesini sağlayan kontrolörleri genelleştirebilmek araştırmacılar için hala zor bir görevdir.

PÖ [45] tabanlı kontrol stratejisi, insansı robotlarda yürüme için yukarıda bahsedilen sorunların çözümünde alternatif bir yoldur. PÖ, karmaşık kontrol sistemlerinin modelsiz öğrenimi olarak uygulanabilen, makine öğreniminin bir alt alanıdır [45]. PÖ, bir ajanın (robot) çevre ile etkileşime girerek farklı durumlarda kendisini nasıl kontrol edeceğini öğrenmesine olanak tanır. PÖ'de çevre, çevresel duruma ve ajan eylemlerine göre ajana ödül veya ceza verecek şekilde modellenmiştir ve ajan, geçmiş deneyimi kullanarak gelecekte hangi eylemlerin en yüksek ödüle yol açacağını tahmin etmeyi öğrenmeye odaklanır. Özellikle, iki ayaklı yürümenin modelsiz öğrenimi, çoğunlukla Markov Karar Süreci kısaca MKS (Markov Decision Process–MDP) dayanan çeşitli eylem politikası öğrenme algoritmalarının uygulanması etrafında dönmüştür [46, 47].

PÖ, dinamik tasarım ve hesaplamanın karmaşıklığının üstesinden gelmeye yardımcı olur. Ek olarak, PÖ kontrolörünün yüksek özerkliği, geleneksel kontrolörlere kıyasla karmaşık ortamlara ve arazilere daha sağlam bir yanıt verir. PÖ, yürüme görevleri için bir tür akıllı öğrenme yöntemidir ve yürüme kararlılığını sağlamak adına SMN konumunu kontrol etmek için PÖ yöntemleri kullanılabilir.

Hareket kontrol görevleri, yüksek performansla PÖ ile çözülebilirken, bacaklı robotların yürüyüş kontrolü, karmaşıklığı nedeniyle bir zorluk olmaya devam etmektedir. Eklemleri doğrudan kontrol etmek için PÖ uygulayarak OpenAI Gym'de [48] iki ayaklı görevleri çözmek için farklı algoritmalar kullanılır [49]. Ancak, bu algoritmalar çok daha karmaşık dinamik özelliklere sahip gerçek bir robot için yürüyüş kontrolünde kullanıldığında, yakınsama ve performans düşmektedir. Ek olarak, eğitilmiş model, genellikle kabul edilemez olan, insan benzeri olmayan bir yürüyüşe de sahip olabilir. Bu nedenle, ön bilgiden yararlanan eğitim yapıları, PÖ tabanlı yürüyüş kontrolörleri için önemli bir çalışma noktası haline gelmiştir. Bir çözüm, PÖ'yü geleneksel modellerle birleştirmektir [50-52]. Gil ve diğerleri [53], bir NAO robotunun en kısa sürede en uzak mesafeye ulaşmasına ve aynı zamanda düşmeden düz bir yol izlemesine olanak tanıyan bir dizi duruş bulmak

için Q-öğrenme'yi kullanmıştır. Liu ve diğerleri [54], NAO insansı robotunun yürüyüşün bilinmeyen rahatsızlıklara karşı dayanaklı hale getirebilmek için yürüyüş model parametrelerinin düzeltilmesi için PÖ algoritmalarından olan Politika Gradyanı kısaca PG'yi (Policy Gradient–PG) [55] kullanmıştır. Lin ve diğerleri [56], dinamik model hakkında önceden bilgi sahibi olmadan dinamik yürümeyi öğrenen Q-öğrenme'yi kullanarak iki ayaklı dinamik yürüyüş ve denge kontrolü için bir yöntem sunmuştur. Robotun tabanlarındaki SMN'yi kaydırmak için robot kolunun ve bacağının hareketini kullanan dengeleme öğrenme yöntemi, iki ayaklı robotu statik kararlı bir durumda tutabilmiştir. Bu dengeleme algoritmaları, düz bir yüzey ve tahterevalli üzerinde iki ayaklı yürüyüşe uygulanmıştır. Simülasyon, önerilen yöntemin robotun yürüme hızı açısından davranışını iyileştirmeyi öğrenmesine izin verebileceğini göstermiştir. [57]'deki çalışma, bir robotun hafif eğimli bir arazide dik konumda yürümesini sağlayacak eylem politikasını öğrenmek için Q-öğrenme tabanlı bir yöntem önermiştir. Sistemin önerilen mimarisi, bir takviye öğrenme bileşeni ile geleneksel yürüyüş oluşturma kontrol döngüsünün iki katmanlı bir birleşimidir. Bu, robotun yürüdüğü zeminin eğimi değiştiğinde yürüyüş için bir düzeltme oluşturmak için bir ivmeölçerin kullanılmasına izin verir. Gerçek bir robot üzerinde yapılan deneyler, önerilen mimarinin kararlılık problemi için iyi bir çözüm olduğunu göstermiştir. Silva ve diğerleri [58], DARwIn-OP insansı robotunun hızlı ve dinamik olarak dengeli bir yürüyüşünü sağlamak için yürüyüş modeli üretecinin parametre değerlerini optimize etmeyi amaçlamışlardır. Bunu, PÖ yöntemlerinden Zamansal Fark kısaca ZF (Temporal Difference–TD) algoritmasını kullanarak başarmışlardır.

Geleneksel PÖ algoritmaları, gözlemlerden elde edilen öznitelik mühendisliğine ihtiyaç duyar. Karmaşık problemler için öznitelikleri çıkarmanın yolu belirsizdir veya iyi bir model oluşturmak için gözlemler yeterli değildir. İki ayaklı yürüme, kararlılığı korurken her eklemin hassas kontrolünü içerdiğinden, çok daha karmaşıktır ve yüksek boyutlu durum alanı ve eylem alanı gerektirir. Daha yeni bir yöntem olan, Derin Sinir Ağları kısaca DSA'lar (Deep Neural Networks–DNNs), büyük durum uzaylı ve eksik gözlemlere sahip verilerden yüksek seviyeli özniteliklerin çıkarılmasına izin verir. DSA'lardaki son gelişmelerle birlikte, Derin Pekiştirmeli Öğrenme kısaca DPÖ (Deep Reinforcement Learning–DRL), bir ajanın çevre ile daha karmaşık bir şekilde etkileşime girmesine olanak tanır.

Fizik tabanlı iki ayaklı karakter animasyonu [59], çeşitli araziler için sağlam hareket stratejileri üreterek DPÖ algoritmalarının çeşitli avantajlarına ışık tutmuştur. [60], arabalı ters sarkaç modeliyle DPÖ'yü kullanarak engebeli arazide çok ayaklı sistemler için kütle merkezi ile ilgili dinamikler tabanlı hareket planlayıcıları öğrenmiştir. Benzer şekilde, [61-63] düz arazide yürümek için yeni DPÖ yapılarını kullanır ve öğrenilen stratejilerin simülasyondan fiziksel bir robota doğrudan politika aktarımını sunar. [64]'teki çalışmada, iki ayaklı hareket, açık kaynak kodlu OpenAI Gym BipedalWalker-Hardcore-v3 çevresi üzerinden incelenmiştir. Robotun dengeleme kontrolörü olarak, geleneksel olarak dengeleme için kullanılan karmaşık dinamik

denklemlerin yerini alan DPÖ ile bir Merkezi MÖÜ'nün kullanımını araştırmışlardır. Yürüme kontrolü, iki sürekli aktör-kritik PÖ algoritması tarafından incelenmiştir: İkiz Gecikmeli Derin Deterministik Politika Gradyanı kısaca İkiz Gecikmeli DDPG (Twin Delayed Deep Determinstic Policy Gradient–TD3) [65] ve Yumuşak Aktör-Kritik kısaca YAK (Soft Actor-Critic–SAC) [66]. Rastogi [67], fiziksel iki ayaklı yürüyen robotlarını simülasyon ortamıyla birlikte yürümek için Derin Deterministik Politika Gradyanı kısaca DDPG (Deep Deterministic Policy Gradient–DDPG) [68] algoritmasını kullanmıştır. Yakınsama için uzun zaman gerektirdiğinden, DDPG'nin yürüyen robotu kontrol etmenin mümkün olmadığı sonucuna varmışlardır. Kumar ve diğerleri [69], gerçekçi bir robotik simülatörü olan Gazebo kullanarak düzlemsel iki ayaklı yürüyen bir robotu tasarlamak ve simülasyonunu yapmak için bir mimari sunmuşlardır. İki ayaklı robotun otonom yürüyüş DDPG algoritması kullanılarak elde edilmiştir. Modeli simülasyonda eğittikten sonra, uygun şekilli bir ödül fonksiyonu ile robotun daha hızlı yürümeyi başarmış ve ortalama 0.83 m/s hızla koşan bir yürüyüş yaptığı gözlemlenmiştir. Ajanları, yaklaşık 25000 bölümde istenen puanı elde etmiştir. Song ve diğerleri [46], Yinelemeli Derin Deterministik Politika Gradyanı kısaca Yinelemeli DDPG (Recurrent Deep Deterministic Policy Gradient–RDDPG) [70] algoritmasını kullanarak iki ayaklı yürümenin kısmi gözlemlenebilirlik sorununa işaret etmiş ve orijinal DDPG algoritmasından daha iyi sonuçlar elde etmiştir. Kasaei ve diğerleri [71], analitik yürüme yaklaşımı ve DPÖ arasında sıkı bir bağlantı kullanarak sağlam iki ayaklı hareket oluşturmak için modüler bir yapı önermiştir. Optimum parametreleri bulmak ve kütle merkezinin yüksekliğini değiştirerek robotun kararlılığını nasıl iyileştireceğini öğrenmek için genetik algoritma ve Yakınsal Politika Optimizasyonu kısaca YPO (Proximal Policy Optimization–PPO) [72] dayalı bir öğrenme yapısı geliştirilmiştir. Abreu ve diğerleri [73], sıfırdan hızlı ve kararlı bir koşu davranışı geliştirmek için YPO algoritması uygulamıştır. Yaklaşımlarında, çevre 80 durumla temsil edilmiştir ve eylem alanı, simülasyonu yapılmış bir insansı robotun eklemleri olan 20 eylemden oluşmaktadır. Yaklaşımlarının performansı, sürat koşusu ve durma davranışlarını öğrenerek gösterilmiştir. Sonuçlar, her iki davranışın da kararlı olduğunu göstermiştir. [74]'teki çalışmada, PyBullet simülasyon ortamında Robotis-OP3 insansı robotun yürüyüşü için geleneksel kontrol ve PÖ ile birleştirilmiş bir kontrolör eğitilmiştir. Çalışma, duruş optimizasyonu ve DPÖ olarak iki aşamaya ayrılır. İlk aşamada, geleneksel kontrolörün yürüme parametreleri optimize edilmiş ve robotun durumuna göre optimize edilmiş parametrelerin birleşimini elde etmek için PÖ algoritması olan Q-öğrenme'yi kullanmışlardır. İlk aşama ile eylem alanı oluşturulan ikinci aşamada, çok katmanlı sinir ağını eğitmek için YPO algoritması kullanılmıştır. Böylece, duruşların hangi sırasının iki ayaklı robotun gerekli görevi gerçekleştirmesini sağlayabileceğini elde etmişlerdir. Zhang ve diğerleri [75], referans hareketi öğren ve gerçekleştir olarak isimlendirdikleri kısaca LORM (Learn and Outperform the Reference Motion) yöntemi, iki ayaklı robotun yürüyüşü için DPÖ tabanlı bir çerçeve, referans hareketin önceki bilgisinden yararlanılarak önerilmiştir. Önerilen yöntem ile

eğitilen ajan, referans hareketten ve mevcut harekete dayalı yöntemlerden daha iyi performans göstermiştir. Önerilen yöntem, Webots simülatöründe DARwIn-OP insansı robotu üzerinde doğrulanmıştır. Christiano ve diğerleri [76], insanlarla insansı robotların olası yörüngelerini karşılaştırmıştır. Verileri bir ödül fonksiyonunu öğrenmek için kullanmış ve öğrenilen ödül fonksiyonunu PÖ ile optimize etmiş ve bu yöntemi DPÖ'ye yükseltmiştir. Özellikle yürüme, zıplama, yüzme vb. robotik görevler MujoCo ve OpenAI Gym'de uygulanmıştır. DeepLoco [77], iki seviyeli bir hiyerarşik kontrol çerçevesi benimsemiştir. İlk olarak, düşük seviyeli kontrolörler belirli bir zaman ölçeğinde çalışmayı ve sağlam yürüyüşler elde etmeyi öğrenmiştir. İkinci olarak, yüksek seviyeli kontrolörler, düşük seviyeli kontrolör için istenen adım hedeflerini çağırarak adımların zaman ölçeğinde politikasını öğrenmiştir. Hiyerarşinin her iki seviyesi de aktör-kritik DPÖ algoritmasının aynı biçimini kullanmıştır. Xie ve diğerleri [78], Agility Robotics tarafından geliştirilen iki ayaklı bir robot Cassie üzerinde DPÖ'yü kullanmıştır. Önerdikleri çerçeve, MuJoCo simülatöründe simülasyonu yapılmış bir Cassie modelinde YPO ve PD kontrolörünü kullanmış olup bozulmalara karşı dayanıklıdır. Özalp ve diğerleri [79], bir Robotis-OP2 insansı robotunun hem simülasyon ortamında hem de gerçekte çizgi izleme hareketi için görüş tabanlı bir DPÖ uygulaması gerçekleştirmiştir. Yaptıkları çalışmada Düello Çift Derin Q Ağı kısaca Düello ÇDQA (Dueling Double Deep Q Networks–D3QN) [80] ve Derin Q Ağı kısaca DQA'nın (Deep Q Network–DQN) [81] başarılarını karşılaştırmıştır. [82]'deki çalışmada, NAO insansı robotunun simülasyon ortamında statik ve dinamik platformlarda yürürken kararlılığını korumak için DDPG algoritması kullanılarak yeni bir hibrit PÖ çerçevesi önerilmiştir. Önerilen pekiştirmeli öğrenme çerçevesi, model tabanlı şemayı modelsiz yapıyla birleştirirken yalnızca dağıtım uyumsuzluğu sorununu başarılı bir şekilde önlemekle kalmamış, aynı zamanda çevrimiçi öğrenme işlemi için örnek verimliliğini de geliştirmiştir. Öğrenilen kontrolörlerin farklı açılara, farklı büyüklüklere ve farklı frekanslara sahip platformlara uyum sağlamadaki sağlamlığı test edilmiştir. Wawrzynski [83], 18 serbestlik derecesine sahip bir insansı robotun yavaş ilk yürüyüşünü, deneyim tekrarlı aktör-kritik PÖ algoritmasını kullanarak hünerli ve hızlı yürüyüşe dönüştürmüştür. Feirstein ve diğerleri [84], PÖ algoritmasının uygulandığı iki tür empedans kontrolörü tarafından basit bir iki ayaklı yürüyüş modelinin limit-döngü yürümesini sağlamıştır. Leng ve diğerleri [85], referans yürüyüşünü tanıtmadan robotun son yürüyüşünü doğrudan elde etmek için bir ortalama-eşzamansız avantajlı aktör-eleştirmen kısaca M-A3C (Mean-Asynchronous Advantage Actor-Critic) PÖ algoritması önerilmiştir. Önerilen yöntemin iki ayaklı robotun sürekli ve kararlı yürüyüş planlamasını sağlayabildiğini göstermiştir.

Liu ve diğerleri [86], statik ve dinamik bozulmalar altında sağlamlığı koruyabilen, DDPG algoritmasına dayalı iki ayaklı bir kontrolör geliştirmiştir. Huang ve diğerleri [87], iki ayaklı hareket için yeni bir ödül uyarlamalı PÖ yöntemi önermişlerdir. Kontrol politikasının dinamik bir mekanizma kullanarak birden fazla ölçüt tarafından aynı anda optimizasyonunu sağlamışlardır.

Önerilen yöntem, hibrit politika gradyanlarına yol açan her bir ödül bileşeni için ayrı bir değer işlevi öğrenmek için çok katmanlı bir kritik uygulamıştır. [88]'de yürüyüş şablonu zorluğunun üstesinden gelmek için bir geri bildirim sistemi oluşturmak için uyarlanabilir ağ tabanlı bulanık çıkarım sistemi kısaca ANFIS (Adaptive Network-based Fuzzy Inference System) [89] Çift Derin Q Ağı kısaca ÇDQA (Double Deep Q Network–DDQN) [90] ile birleştiren bir geri bildirim sistemi önerilmiştir. ÇDQA, robot yürüme parametrelerini revize etmek için ANFIS'in çıktısından gelen veriler kullanılarak eğitilen bir tahmin modeli olarak kullanılmıştır. Wong ve diğerleri [91], bir insansı robot için bir yörünge planı oluşturma ve iki ayaklı hareket elde etme işlemleri için sinüzoidal fonksiyonlara sahip osilatör tabanlı bir yürüyüş modeli tasarlamışlardır. Robotun düz yürümesini sağlamak için, dönüş yönü yürüyüş şablonunun bir parametresi olarak görmüşlerdir. Q-öğrenmeyi, basit bir yürüyüş şablonu oluşturmak için kullanmışlardır. Deneysel sonuçlar, önerilen çerçevenin insansı robotun kademeli olarak düz yürümesine izin verdiğini göstermiştir.

İnsansı robotlar, yürüyüş sırasında dinamik dengesini koruyarak motorlarını kontrol etmek için bir yürüyüş şablonu üreteci kullanır. Yürüyüş şablonu iki ayaklı yürümede hayati bir rol oynar. Bunun nedeni, destek ayağının optimal dayanaklarını, hızını ve ivmesini belirlemesidir. Bu da robotun dinamik kararlılığını, enerji tüketimini, uyumunu vb. etkiler. Bu nedenle, iki ayaklıların yürüyüş düzenini dikkatli bir şekilde tasarlamak, enerji tüketimini azaltmanın yanı sıra kararlılığını artırmada çok yardımcı olur. Yürüyüş şablonu için robotun istenen şekilde yürümesini sağlayacak yörüngeler oluşturulur. Yörünge oluşturma birçok yürüme parametresini içerir. Yürüme parametrelerinin elle ayarlanması, özellikle on serbestlik derecesinden fazla olan karmaşık yapıya sahip bir robot için oldukça karmaşık ve zorlu bir süreçtir. Geleneksel yörünge üreteci kontrolörler, yalnızca orta düzeyde bozulmalara izin vermektedir. Yapılacak herhangi bir küçük değişiklik, yürümenin başarısız olmasına neden olabilir. Bu kontrolörler, basamak ve eğim gibi farklı ortamlara uyarlanamayabilir. Maliyeti ve iş yükünü büyük ölçüde artıran farklı yürüme parametrelerinin yeniden tasarlanması gerekir.

Günümüzde DPÖ, iki ayaklı robotların yürüyüş kontrolörlerini elle ayarlama ve sistem modellemesi olmadan eğitmek için modelsiz bir yöntem sağlar. Bazı uçtan uca DPÖ yöntemleri, robot modelini eğitmek için bir simülatörün varsayılan ödülünü kullanır. Bununla birlikte, bu tür DPÖ kontrolörleri, robotlar için kabul edilemez, doğal olmayan bir hareket üretebilir. Bu durumu azaltmaya yönelik bir yaklaşım, ön bilgileri dahil etmektir, ayrıca ön bilgi olmadan pekiştirmeli öğrenme için hiperparametreler genellikle hassastır. Düşük yakınsama ve eğitim verimliliği uygulamaları sınırlandırmaktadır.

Bu tür sınırlamaların üstesinden gelmek için bu tez çalışmasında öncelikle insansı robot modeli Robotis-OP2'ye dayalı olarak geleneksel yörünge üreteci kontrolör ve DPÖ ile birleştirilmiş etkili yeni bir çerçeve önerilmiştir. Önerilen çerçeve sayesinde yürüyüş şablonu üretici ile oluşturulan yörüngelerin optimum yürüyüş parametreleri Düello ÇDQA ile elde edilmiştir. Düello

ÇDQA'nin eğitimi, Webots simülatöründe robotun çoklu sensörleri kullanılarak gerçekleştirilmiştir. Optimum yürüyüş parametreleri belirlendikten sonra sagital düzlemde robot duruş dengeleme sistemi önerilmiştir. Önerilen çerçeve ve Robotis-OP2'nin kendi yürüme algoritması ile Webots simülasyon ortamında ve oluşturulan kontrolörlerin gerçek robota aktarılmasıyla gerçek ortamda deneysel çalışmalar gerçekleştirilmiştir. Deneysel sonuçlar, Robotis-OP2 insansı robotunun önerilen çerçeve ile hem simülasyon hem de gerçek ortamda düz bir zeminde Robotis-OP2'nin kendi yürüme algoritmasına göre daha kararlı bir şekilde yürüyebildiğini göstermiştir.

Tez çalışması kapsamında daha sonra, robotun dengesini koruyarak eğimli yüzeylerde kararlı bir şekilde yürüyebilmesi için PID kontrolör ve DPÖ kontrolörden oluşan iki ayrı yürüyüş dengeleme çerçevesi önerilmiştir. DPÖ kontrolör olarak DDPG algoritması tercih edilmiştir. PID kontrolör ile gerçekleştirilen deneysel çalışmalarda robotun duruşu gerçek zamanlı olarak ayarlanarak düz zeminde yürüyormuş gibi gövde yunuslama açısının istenilen referans değerde olması sağlanmıştır. DDPG kullanılarak gerçekleştirilen ağ eğitimi ile robotun eğimli yüzeylerde dengeli yürüyüşünün sağlanabilmesi için robotun gövde yunuslama açısının sıralı hareket dizisi öğrenilmiştir. Yokuş yukarı ve yokuş aşağı eğimli yüzeyler için eğitim işlemleri tamamlandıktan sonra elde edilen ağ modellerini içeren kontrolörler ve PID kontrolör ile deneysel çalışmalar gerçekleştirilmiştir. Deneysel sonuçlar, DPÖ ile oluşturulan kontrolörün PID kontrolöre göre daha kullanışlı olduğunu ve daha kararlı bir yürüme sağladığını göstermiştir.

Bu tez çalışmasında YSA, herhangi bir hazır veri kümesi olmadan doğrudan robottan alınan verilerle eğitilmiştir. Standart makine öğrenimi yöntemlerinin çoğu, denetimli öğrenme veya denetimsiz öğrenme yöntemleri için uygundur. Bu tez çalışmasında makine öğrenimi yöntemlerinden DPÖ'nün seçilmesinin temel sebebi, robot sistemleri üzerinde veri kümesi oluşturma maliyetlerinin yüksek oluşudur. Yürüme sırasında robot, çevre ile etkileşim kurarak sensörler ile çevreden alınan durum bilgisine göre robotun ödül/ceza mekanizması ile optimumu öğrenmesi sağlanmıştır.

Bu tez çalışması, DPÖ kullanılarak insansı robotların kararlı yürüyebilmesi için yürüme parametrelerinin optimize edildiği ve eğimli yüzeylerde dengeli yürüyüş için kontrolör olarak kullanıldığı literatürdeki ilk çalışmadır. Ayrıca, tez kapsamında önerilen çerçevelerin insansı robot hareketi alanında çalışan araştırmacılar için faydalı olacağı düşünülmektedir.

1.1. Tezin Organizasyonu

Bu tez çalışması, 8 bölümden oluşmaktadır.

Birinci bölümde, DPÖ algoritmaları kullanılarak insansı robotların düz ve eğimli yüzeylerde kararlı bir şekilde yürüyebilmesinin geliştirilmesi ile ilgili olan tezin hedefi ve bu alandaki literatür taramasına yer verilmiştir.

İkinci bölümde, önce insansı robotlar hakkında bilgilere yer verilmiş ve tez çalışması kapsamında kullanılan Robotis-OP2 insansı robot platformu detaylandırılmıştır. Daha sonra, robotik simülatörlerin bileşenleri ve tez çalışmalarında kullanılan Webots robotik simülatörü üzerinde durularak neden tercih edildiği açıklanmıştır.

Üçüncü bölümde, önce PÖ detaylı olarak incelenmiştir. Daha sonra, PÖ yöntemlerinden bahsedilerek tez kapsamında kullanılan DPÖ yöntemleri için bir temel oluşturulmuştur. Son olarak, PÖ platformlarından kısaca bahsedilmiştir.

Dördüncü bölümde, önce DPÖ'ye geçiş için bilinmesi gereken derin öğrenme kavramı detaylı olarak incelenmiştir. Ardından, tez çalışmalarında kullanılan Düello ÇDQA ve DDPG algoritmaları için DPÖ yöntemleri detaylandırılmıştır.

Beşinci bölümde, önce tez çalışması kapsamında kullanılan Robotis-OP2 insansı robotunun yürüyüş şablonu üreteci için oluşturulan yürüme yörüngelerini içeren yürüyüş modeline yer verilmiştir. Daha sonra, yürüyüş modeli ile elde edilen yürüme yörüngelerinde robotun bacak eklemlerinin alması gereken açıların belirlenebilmesi için ters kinematik analiz detaylı bir şekilde incelenmiştir.

Altıncı bölümde, yürüme şablonu üretecinin optimum yürüyüş parametrelerini elde etmek için robotun çoklu sensörleri kullanılarak DPÖ yapısı olan Düello ÇDQA'nın durum uzayını oluşturan yuvarlanma ve yunuslama yönelim açıları ile yürüyüş kararlılık ölçütü olan iki eksendeki Sıfır Moment Noktası'nın (SMN'nin) hesaplanma işlemleri detaylı olarak verilmiştir.

Yedinci bölümde, deneysel çalışmalara yer verilmiştir. İlk olarak, yönelim açılarının doğru bir şekilde hesaplanabilmesi için hangi filtrenin kullanımının uygun olacağı ve filtre katsayılarının belirlenmesine yönelik çalışmalar gerçekleştirilmiştir. İkinci olarak, SMN'lerin düzgün bir şekilde hesaplanabilmesi için ayak temas kuvvetlerinin bir boyutlu Kalman filtre ile filtrelenerek filtre katsayılarının belirlenmesine yönelik çalışmalara gerçekleştirilmiştir. Üçüncü olarak, Düello ÇDQA'nin mimarisiyle önerilen kararlı yürüyüş için parametre optimizasyon çerçevesi için ağın eğitim ve test işlemleri gerçekleştirilerek sagital düzlemde robot duruş dengeleme sistemi ile ilgili deneysel çalışmalar gerçekleştirilmiştir. Dördüncü olarak, önerilen çerçeve ile elde edilen yürüme sonuçları robotun kendi algoritması ile karşılaştırılmıştır. Son olarak, eğimli yüzeylerde yürüme için PID kontrolör ve DPÖ algoritmalarından olan DDPG kontrolör tabanlı önerilen çerçeveler ile ayrı ayrı çalışmalar gerçekleştirilerek sonuçlar karşılaştırılmıştır.

Sekizinci bölümde ise elde edilen sonuçlar ile öneriler ve gelecek çalışmalar sunulmuştur.

2. İNSANSI ROBOTLAR VE ROBOTİK SİMÜLATÖRLER

İnsansı robotların gelişimi, robotikte iyi bilinen ve ilginç bir araştırma alanıdır. İnsansı robotlar genellikle insana benzer iki ayaklı hareket kullanır ve kurtarma, eğitim, yardım, eğlence vb. çeşitli uygulamalarda en yetenekli robotlardan biri olmalarını sağlayan farklı yeteneklere sahiptir. Ayrıca bu robotların insanlara olan yüksek benzerlikleri nedeniyle, geleceğin toplumunda hayati rolleri kaçınılmazdır. İnsansı robotların birçok robotik sisteme benzerliği, gelecek vaat eden yeteneklerinin yanı sıra çoklu robot uygulamalarında kullanılmasına olanak sağlamaktadır [92].

İnsansı robotik sistemlerle ilgili araştırmalar, yalnızca ilgili tasarım için değil, aynı zamanda geliştirilmesi ve ardından uygulanması için de önemli miktarda hesaplama içermektedir. Gerçek dünya senaryolarında uygulanacak robotik sistemler için hata ve beklenmeyen davranış riskini azaltmak için kontrollü ortamlarda geliştirilip test edilmesi tercih edilir. Daha erişilebilir ve verimli bir alternatif olarak ortamı, robotik simülatörleri kullanarak uygulamaktır.

2.1. İnsansı Robotlar

İnsansı robotlar genellikle farklı yeteneklerle farklı ortamlarda iki ayak üzerinde yürüyebilmektedir. Böylece tasarlanan her insansı robot, tasarım amaçlarına göre farklı yeteneklere sahiptir [93-96]. Geliştirilen insansı robotların çoğu düz zeminlerde ve çimenlerde sabit bir şekilde yürüyebilirken bazıları engebeli ortamlarda yürüyebilir. Ayrıca bazı insansı robotlar merdiven çıkmak, kayak yapmak, paten kaymak ve yokuşta yürümek için tasarlanmış ve geliştirilmiştir [97]. İnsansı robotlar karmaşık bir yapıya sahip olduğu için bu robotların tasarım ve kontrol süreçlerine daha fazla önem verilmiştir.

İnsansı bir robotta, çok yönlü yürüyüş, insansı robotları otonom hale getirmeyi amaçlar. Hareket, robot görüşü ve davranış kontrolü, en kritik ve zorlu yazılım geliştirme çabalarıdır. Bu bağlamda, çok çeşitli öğrenme algoritmaları önerilmiştir ve kullanılmaktadır [98-101]. İnsansı bir robotun boyu ve ağırlığı, bahsedilen robotların yapısını ve davranışını etkileyen kritik özelliklerden biridir. Bu, aktüatörlerin boyut, ağırlık ve güç tüketimleri ile birlikte tanıtılan tork, hız ve kontrol hassasiyeti dikkate alınarak insansı robot gelişimi için bazı sınırlamaları olan insansı robotlarda kullanılabilecek mevcut aktüatör teknolojileriyle ilgilidir. Bu nedenle, insansı robot geliştiricileri, yazılım geliştirme açısından hala insansı robot donanım gelişimi ile sınırlıdır.

İnsansı robotlar, çocuk (küçük) boyutlu ve yetişkin boyutlu olmak üzere iki ana boyut kategorisine ayrılır. Çocuk boyutlu insansı robotlar, genel olarak yüksekliği 140 cm'den küçük olan robotlar olup bu robotların temel geliştirilme nedenleri; düşük maliyet, hafif olmaları, bakımlarının kolaylığı, daha az geliştirme ve test alanı gerektirmesidir. Şekil 2.1'de örnek olarak gösterilen ARC [102], DARwIn-OP [103], Robotis-OP2 [104], Robotis-OP3 [105], NAO [106], igus [107] ve

Poppy [108] çocuk boyutu kategorisinde başarılı robotlar olarak kabul edilir. Büyük boyutlu veya insan boyutlu insansı robotlar olarak da bilinen yetişkin boyutlu insansı robotların gelişimi, son yirmi yılda çarpıcı bir şekilde artmıştır. İyi bilinen yetişkin boyutlu insansı robotlara ise ASIMO [109], THR3 [110], ATLAS [111], REEM-C [112], TORO [113], Lola [114] ve NimbRo-OP2 [115] örnek olarak verilebilir.

İki ayaklı hareket sistemine sahip insansı robotlar, karmaşık bir alanda gezinmek için daha uygun ve daha uyarlanabilir olmak gibi tekerlekli robotlara göre farklı avantajlar sunmaktadır [116, 117]. İki ayaklı robotların tasarımında ve kontrolünde karşılaşılan zorluklardan biri, farklı tür ortamlarda yürürken kararlılıklarını koruyabilme durumudur [118]. İnsansı robotlar, uyarlanabilir donanım tasarımı ve hiyerarşik yazılım mimarisinden oluşan karmaşık ve yüksek düzeyde bütünleşmiş sistemler olarak kabul edilir [119].

Şekil 2.1. Araştırma amaçlı uygulamalarda kullanılan çocuk boyutlu insansı robotlar, soldan sağa: ARC [102], DARwIn-OP [103], Robotis-OP2 [104], Robotis-OP3 [105], NAO [106], igus [107] ve Poppy [108]

2.1.1. İnsansı Robotların Donanım Tasarımı

İnsansı robotların mekanik yapısı ve kinematik gelişimi için insan yapısı ve kinematikten ilham alınır. Başka bir deyişle, insansı robot tasarımının bir insanla benzer yeteneklere sahip olması hedeflenmektedir [120]. Son teknoloji insansı robotların önemli donanım özelliklerinin karşılaştırılması ve sınıflandırması Tablo 2.1'de sunulmaktadır [121].

Tablo 2.1. Popüler insansı robotların önemli donanım özellikleri [30]

Sınıf	Robot	Boyut (cm)	Ağırlık (kg)	Yapı malzemesi	Aktüatör tipi	İşlem birimi	Sensör
Çocuk boyutlu insansı robotlar	DARwIn-OP [103]	45.4	2.9	Alüminyum, plastik	Elektrik	Atom Z530 1.6 GHz CPU (32 bit), tek çekirdek, 4 GB flash SSD	Kamera, Atalet Ölçüm Birimi (AÖB), mikrofon
	ROBOTIS-OP2 [104]	45.4	3	Alüminyum, plastik	Elektrik	Intel Atom N2600 1.6GHz CPU (32 bit), çift çekirdek, 32 GB mSata	Kamera, AÖB, mikrofon
	ROBOTIS-OP3 [105]	51.04	3.5	Alüminyum	Elektrik	Intel NUC i3, 8 GB DDR4, 120 GB, ARM cortex-M7	Kamera, AÖB
	NAO [106]	57	5.2	ABS plastik	Elektrik	ATOM Z530 1.6 GHz CPU, 1 GB RAM2, 2 GB Flash memory, 8 GB Micro SDHC	Kamera, AÖB, kuvvete duyarlı direnç, sonar, rotary enkoder, dokunma
	Poppy [108]	83	3.5	3B baskı	Elektrik	Raspberry Pi	Stereo mikrofon, kuvvet, AÖB
	Igus [107]	92	6.6	3B baskı	Elektrik	Intel i7, 2.4 GHz CPU, 4GB RAM, 120GB SSD	Kamera, AÖB, enkoder
	iCub [122]	104	22	3B baskı	Elektrik	Mikrokontolör tabanlı 56F807 çip	Kamera, AÖB, kuvvet, tork, mikrofon
	ARC [102]	54	2.9	3B baskı	Elektrik	Intel i5, 1.8 GHz CPU, 32GB RAM	–
	Surena mini [123]	53.4	3.3	3B baskı	Elektrik	Intel Core M5, 1.1 GHz CPU, 4GB DDR3, 64 GB eMMC	Kamera, mikrofon, kuvvet, AÖB, Kızılötesi (IR), dokunma
Yetişkin boyutlu insansı robotlar	ASIMO [109]	130	50	Magnezyum alaşım	Elektrik/hidrolik	Pentium III-M 1.2 GHz	Kamera, AÖB, kuvvet, mikrofon, dokunma
	NimbRo-OP2 [115]	134.5	17.5	3B baskı	Elektrik	Intel i7 2.4 GHz CPU, 4GB RAM, 120GB SSD	Kamera, AÖB
	HRP-4 [124]	151	65	Silikon, plastik, metal	Elektrik	Intel Pentium M 1.6 GHz	Kamera, AÖB, kuvvet
	REEM-C [112]	165	80	–	Elektrik	Intel Core i7 2.4 GHz	Kamera, sonar, lazer tarayıcı, kuvvet, tork, AÖB
	ATLAS [111]	150	75	–	Hidrolik	–	Kamera, lidar
	HUBO+ [125]	170	80	–	Elektrik	Intel Atom 1.6 GHz	–
	Lola [114]	180	55	–	Elektrik	Intel Core Duo 2.33 GHz	Kamera, kuvvet, tork, AÖB

Tablo 2.2'de, son teknoloji insansı robotlardaki toplam serbestlik derecesi ve her bir parçanın serbestlik derecesi gösterilmektedir.

Tablo 2.2. İnsansı robotların serbestlik derecelerinin sınıflandırılması

Boyut	**İsim**	**Serbestlik Derecesi**					
		Baş	*Kol*	*El*	*Kalça*	*Bacak*	*Toplam*
Çocuk boyutlu	DARwIn-OP	2	6	0	6	6	20
	ROBOTIS-OP2	2	6	0	6	6	20
	ROBOTIS-OP3	2	6	0	6	6	20
	NAO	2	10	2	5	6	25
	Poppy	2	8	0	9	6	25
	Igus	2	6	0	6	6	20
	iCub	6	14	18	6	6	53
	ARC	2	6	0	6	6	20
	Surena mini	2	8	0	7	6	23
Yetişkin boyutlu	ASIMO	3	14	26	2	12	57
	HRP-4	11	12	4	9	8	44
	REEM-C	2	14	38	8	6	68
	ATLAS	2	14	0	6	6	28
	HUBO+	1	14	2	7	6	30
	Lola	2	6	0	6	10	24
	Nimbro-OP2	2	6	0	6	6	20

2.1.2. İnsansı Robotların Yazılım Mimarisi

Diğer robot türleri gibi geliştirilen insansı robotlar da farklı yazılım geliştirme yönlerinden oluşur ve robotun iki ayaklı yürüyüşünü kontrol etmek dışında çoğu ortaktır. İnsansı robotlar için yazılım geliştirmede çok çeşitli programlama dilleri ve simülatör platformları mevcuttur. Bu alt bölümde yürüme, görüş ve davranış kontrolü olmak üzere insansı robotlardaki yaygın yazılım geliştirme yönleri açıklanacaktır.

Yürüme Kontrolü

Bir insansı robot, farklı yüzeylerde farklı hızlarda çok yönlü olarak yürüyebilmelidir. Çok yönlü yürümenin amacı; bir robot için güvenilir, çok yönlü ve dinamik hareket elde etmektir. Bu durum, insansı robotun farklı ortamlarda karmaşık yürüyüş şablonları gerçekleştirmesini sağlar [111]. Ancak bazı senaryolarda ve zorlu ortamlarda çok yönlü yürüme kontrolü zorlayıcıdır. İnsansı robotların iki ayaklı yürümesi, açık döngü ve kapalı döngü yürüme motorları olarak nitelendirilebilir [102]. Atalet Ölçüm Birimi kısaca AÖB'den (Inertial Measurement Unit–IMU)

gelen geri bildirimleri kullanıp kullanmadıklarını içeren farklı yürüme kararlılık ölçütleri ile farklı yürüyüş motorları oluşturulabilir.

İnsansı robot yürüyüşünün farklı kararlılık ölçütleri vardır. İlk yürüme ölçütü, insansı bir robotun vücudundaki kütle merkezinin ve basınç merkezinin konumudur. Bir robot, kütle merkezi veya basınç merkezi ayak destek alanının dışbükey gövdesi içindeyse kararlı olarak kabul edilir [99, 126]. İkinci ölçüt, Sıfır Moment Noktası (SMN) kavramına dayanan yarı dinamik yürümedir [127]. Üçüncü ölçüt ise pasif dinamik yürüyüş kavramına dayanan dinamik yürüyüştür [99]. Model tabanlı kontrol yaklaşımlarıyla başarılı bir yürüyüş uygulamak için, öncelikle, iki ayaklı bir robotun kinematiği ve dinamikleri ile ortamları tam olarak modellenir. Ardından, bir yürüyüş motoru tanımlanır. Daha sonra robotun mutlak kartezyen konumu zaman içinde robot için hesaplanabilir.

Görüş Kontrolü

Algı için en önemli sensörlerden biri kameradır. Ayrıca, yakalanan görüntülerden bilgi çıkarmak için gerçek zamanlı robot görme algoritmaları zorunludur. İnsansı robotlar hem monoküler hem de stereo görüş sistemlerini kullanır. Monoküler bir görme sisteminde, sistemin çevre hakkında doğru bilgi sağlayamadığı bilinmektedir [128]. Bir stereo görüş sistemi, genellikle stereo işleme için bir kontrolör birimi ile donatılmış iki kameradan oluşur. Stereo görüş sistemini kullanmak, robotun nesnelerin mesafesini ölçmesini, zemini algılamasını ve engellerin etrafından dolaşmak için bir yol oluşturmasını sağlayabilir [129, 130].

Davranış Kontrolü

İnsansı robot alanında önerilen bir diğer kritik konu da davranış kontrolüdür. Bu, robotik alanın zorlu kısımlarından biridir. Bir insansı robotun davranış kontrolü, otonom, yarı otonom veya uzaktan kontrol yaklaşımları olarak kategorize edilebilir.

Otonom ve yarı otonom davranış kontrol sistemleri açısından araştırmacılar, yapay zeka bağlamında farklı yaklaşımları ve algoritmaları geniş çapta incelemiştir [131]. Genellikle basit robotlarda bazı basit sezgisel algoritmalar [132] daha verimli sonuçlar sağlarken, karmaşık görevlerde normalde akıllı bir karar verme sistemi tasarlanır ve kullanılır [133]. Otonom ve yarı otonom insansı robot davranış kontrol sistemlerindeki en önemli araştırma yönlerinden biri yol planlama, adım planlama ve navigasyondur.

İki ayaklı robotlar, engebeli bir ortamda engellerin üzerine veya üzerinden geçmek için benzersiz bir yeteneğe sahip olduklarından, tekerlekli mobil robotlar için bazen zor olan engelleri geçebilirler [134]. Öte yandan, bazı son teknoloji insansı robotlar farklı türde merdiven ve merdivenlere tırmanabilir [109]. Bu nedenle insansı robotlarda yol planlama ve adım planlama algoritmalarının geliştirilmesi ayrı bir önem taşımaktadır. Bu nedenle insansı robotlarda karar verme süreci daha karmaşıktır [101]. İnsansı robotların yol planlamasını ve navigasyonunu çözmek

için birkaç güçlü yapay zeka tekniği önerilmiş olsa da, dinamik ortamlarda tamamen uygun bir çözüm olmamıştır.

2.1.3. Robotis-OP2 İnsansı Robot Platformu

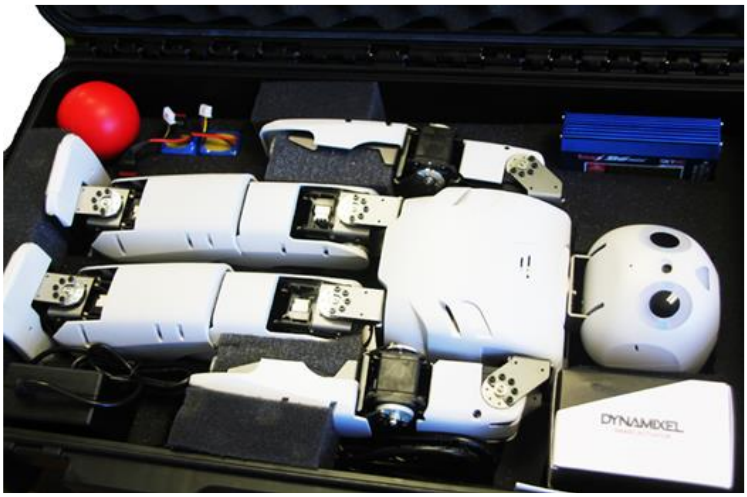
Tez kapsamında gerçekleştirilen çalışmalarda, Robotik ve Yapay Zeka Laboratuvarı'mızda bulunan Robotis-OP2 insansı robot platformu kullanılmıştır.

Robotis-OP2, Ar-Ge ve eğitim amaçlı uygulamalarda kullanılmak üzere belirli sensörler ve belli bir seviyede yük taşıma kapasitesine sahip 20 serbestlik dereceli bir insansı robottur. Bu insansı robot, Pennsylvania Üniversitesi ile işbirliği içinde Koreli bir robot üreticisi olan Robotis firması tarafından geliştirilmiş ve üretilmiştir. Robotis-OP2, DARwIn-OP olarak bilinen insansı robotun geliştirilmiş sürümüdür. Şekil 2.2'de iki robotun da genel görümü verilmiştir.

(a) (b)

Şekil 2.2. Robotların genel görünümü: (a) Robotis-OP2 ve (b) DARwIn-OP

Robotis-OP2 insansı robot platformu, Şekil 2.3'te verildiği gibi olup içeriği Şekil 2.4'te numaralandırılan parçalardan oluşmaktadır. Bu parçaların listesi de Tablo 2.3'te verilmiştir.



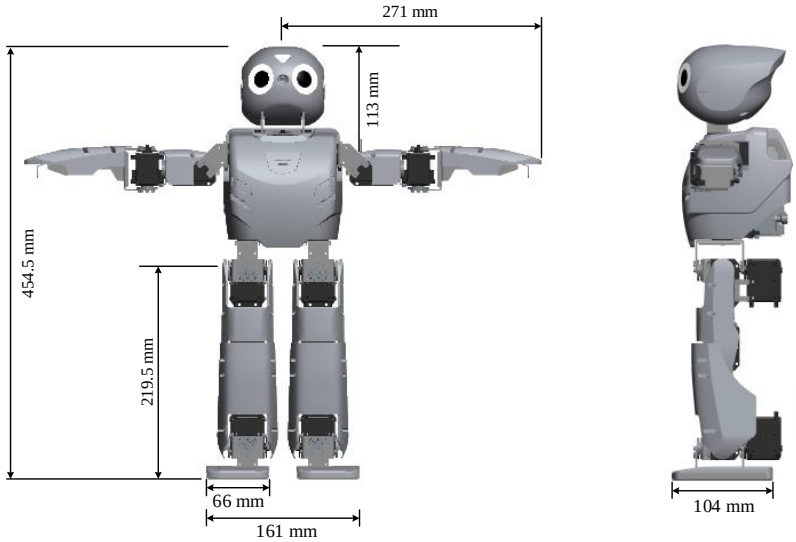
Şekil 2.3. Robotis-OP2 insansı robot platformu

Şekil 2.4. Robotis-OP2 insansı robot platformunun içeriği

Tablo 2.3. Robotis-OP2 insansı robot platformunun içerik listesi

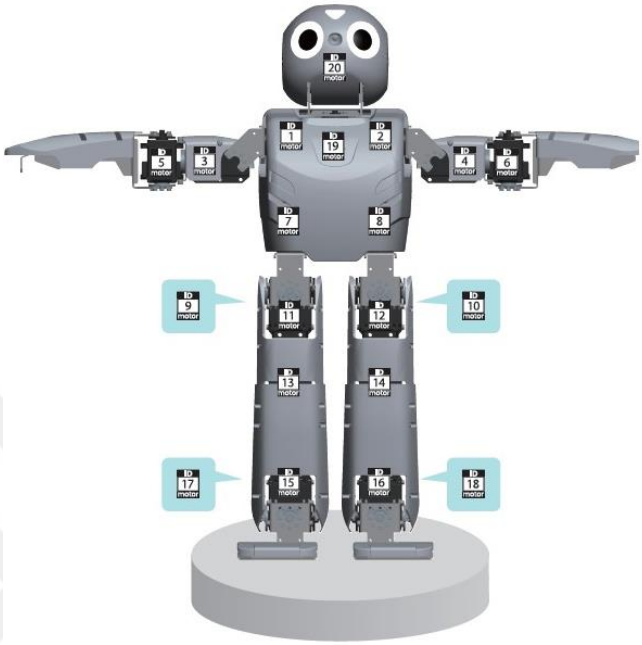
No.	İçerik	Miktar
1	Robotis-OP2 insansı robot	1 adet
2	USB flash bellek	1 adet
3	Sigorta	2 adet
4	Renk kartları	7 adet
5	Hızlı başlangıç rehberi	1 adet
6	Güç kablosu	1 adet
7	DC güç kaynağı	1 adet
8	Yedek kablolar	1 paket
9	Kırmızı top	1 adet
10	Batarya (LiPo)	3 adet
11	Batarya şarj cihazı	1 adet
12	Alyan ve tornavida seti	1 adet
13	Yedek civata ve somun	1 paket
14	Ethernet kablosu	1 adet
15	Muhafaza/taşıma çantası	1 adet

Şekil 2.5'te genel boyutları verilen 3 kg ağırlığa sahip robotun her bir eklemi, yüksek torka sahip Dynamixel MX-28T akıllı servo motor tarafından kontrol edilir. Eklemlerin kontrolü, Dynamixel-bus'ın 1Mbps bant genişliği ile sağlanarak yüksek serbestlik derecesinde robotun, esnek hareket kabiliyetleri göstermesine olanak tanınır.

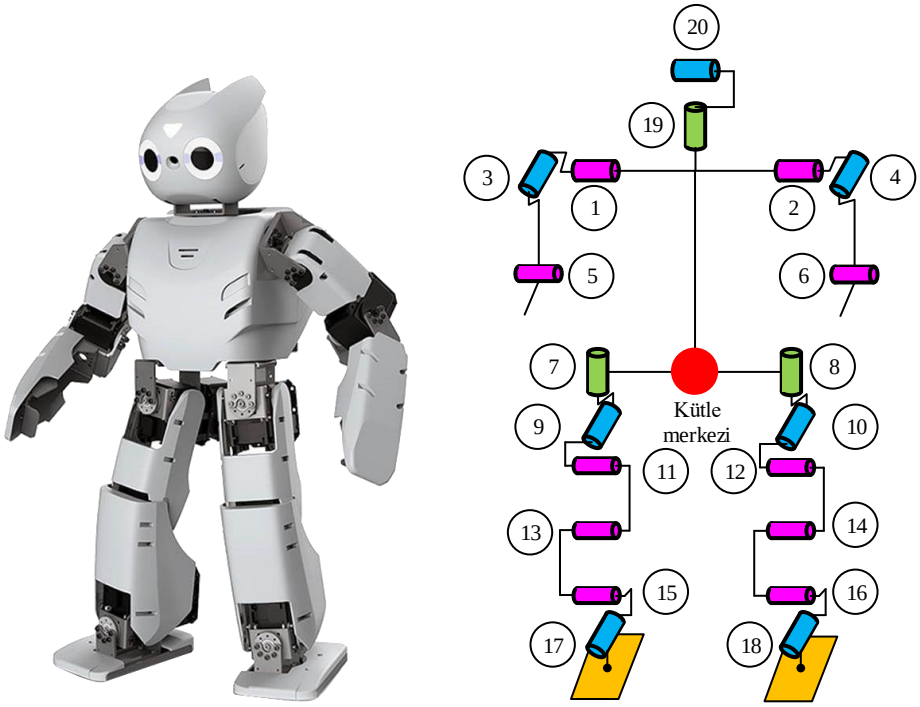


Şekil 2.5. Robotis-OP2'nin boyutları

Robotis-OP2'nin varsayılan yapılandırılmasında motor kimlik numaraları Şekil 2.6'da verilmiş olup Şekil 2.7'de ise eklem yerleşimleri gösterilmiştir.



Şekil 2.6. Robotis-OP2'nin motor kimlik numaraları



Şekil 2.7. Robotis-OP2'nin eklem yerleşimi

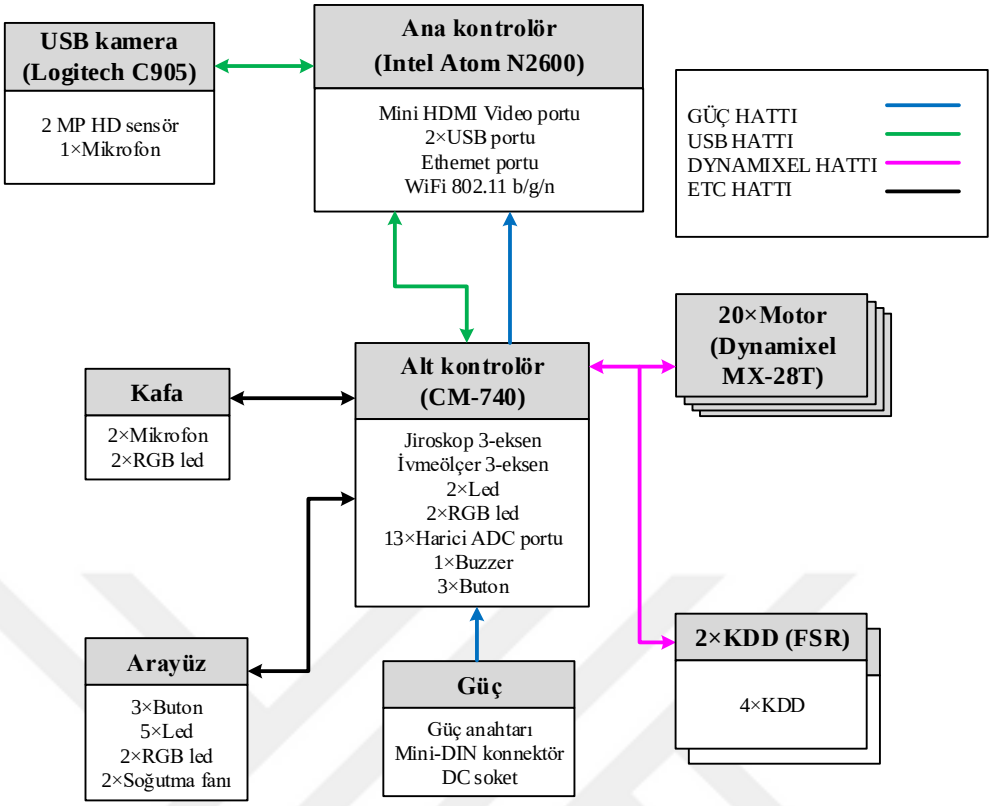
Robotun kütle merkezi, pelvisinin merkezinde yer almaktadır. Bu durum, yürüme sırasında daha iyi bir dengeleme ve uygun bir atalet dağılımı sağlar. Robot üzerindeki 1800 mAh'lik LiPo batarya ile robot, yaklaşık olarak 30 dakika boyunca çalışır vaziyette kalabilmektedir.

Robotis-OP2, Linux işletim sistemli yerleşik bir bilgisayar sayesinde önemli ölçüde hesaplama kapasitesine sahiptir. DARwIn-OP'a göre daha yüksek işlem gücü ve daha seri hareketler üretecek şekilde tasarlanmıştır. Robotis-OP2'de kullanılan ana işlemci, DARwIn-OP'den farklı olarak tek çekirdek yerine çift çekirdek tercih edilmiştir. Ayrıca, RAM ve SSD belleklerin kullanıcı tarafından değiştirilebilmesine müsaade edilmekte ve CM-730 alt kontrolöründen CM-740'a geçilerek boyut olarak küçültmeye gidilmiştir. Belirtilen bu farklar, Tablo 2.4'te listelenmiştir.

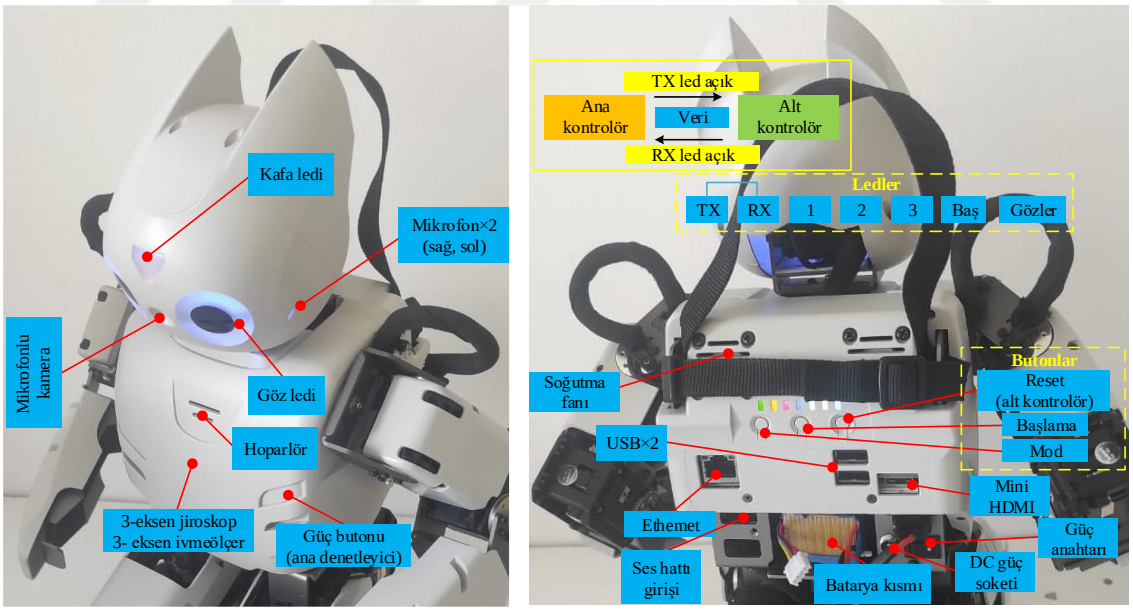
Tablo 2.4. DARwIn-OP ve Robotis-OP2'nin donanım özelliklerinin karşılaştırılması

	DARwIn-OP	**Robotis-OP2**
CPU (Ana kontrolör)	Intel Atom Z530 @1.6GHz, tek çekirdek	Intel Atom N2600 @1.6GHz, çift çekirdek
RAM	1 GB DDR2	4 GB DDR3
Depolama alanı	4 GB NAND flash IDE100 (sabit kapasite)	32 GB yarı boyut mSATA (değitirilebilir)
LAN hızı	100 Mbps	1 Gbps
Kurulabilir işletim sistemi	Linux (32-bit)	Linux (32-bit) Windows (32-bit)
WiFi	802.11g	802.11n (sadece 2.4 GHz)

Robotis-OP2'nin kaynak kodları, devre şemaları, mekanik tasarım dosyaları ve parça bilgilerini içeren tüm kaynaklar halka açık bir şekilde paylaşılmaktadır. Hem donanım hem de yazılım bileşenleri açık kaynaklıdır. Robotis-OP2'nin ağ tabanlı modüler yapısı ve sahip olduğu standart bilgisayar mimarisi Şekil 2.8'de verilmiştir. Şekil 2.9'da ise robotun ana bileşenleri, robot üzerinde gösterilmiştir.



Şekil 2.8. Sistem blok diyagramı



Şekil 2.9. Ana bileşenlerin robot üzerinde gösterimi

Robotun modüler ağ tabanlı yapısı, Robotis-OP2'nin genel performansından neredeyse hiç ödün vermeden istenen uzvu, vücudun geri kalanından izole ederek kullanıcının herhangi bir ekstremiteyi değiştirmesine yardımcı olmaktadır.

Robotis-OP2, ağ tabanlı modüler yapının kapsamını koruyan, çok sayıda sensöre sahiptir. Robotta, üst gövdeye monte edilmiş duruş tahmini ve dengeleme için 3 eksenli bir jiroskop ve 3 eksenli bir ivmeölçer sensörleri yer almaktadır. Kafanın içine yerleştirilmiş görüntü işleme için USB tabanlı bir kamera bulunmaktadır. Kamera üzerindeki mikrofon, Robotis-OP2'nin sesli komutları algılamasına izin verir. Ses lokalizasyonu amacıyla kafanın içine iki ek mikrofon yerleştirilmiştir. İsteğe bağlı sensör olarak, zemin tepki kuvvet ölçümleri için her ayağa dört Kuvvete Duyarlı Direnç kısaca KDD'nin (Force Sensitive Resistor–FSR) yerleştirildiği birimler bulunmaktadır. Ayak başına dört sensör yerleştirmek, yürüyüşü sırasında belirli bir Robotis-OP2 pozunda ayağın belirli bir alanındaki belirli kuvvet miktarı hakkında daha kesin bilgi sağlar. KDD'lerden gelen verilerle güçlendirilmiş bir kapalı döngü geribildirim kontrolü, kullanıcının robotun yürüyüşünü iyileştirmesine izin verebilir. Bu tür veriler, yürüme yüzeyindeki beklenmedik değişiklikler sırasında yürüyüşü iyileştirmek için de uygulanabilir. Ek sensörler, kullanıcının takdirine bağlı olarak harici giriş/çıkış yoluyla da eklenebilir.

Motorlar, sensörler, ledler, butonlar ve harici giriş/çıkış pinleri gibi tüm cihazlar, Dynamixel protokolünü tam olarak destekleyen bir seri veri yolu ağı ile alt kontrolöre bağlanır. Her cihazın, belirlenmiş kimliğe sahip bellek eşlemeli bir işlem yapısı vardır. Ana kontrolör olarak, küçük dizüstü bilgisayarlar için kullanılan Intel'in Atom N2600 merkezi işlem birimi kısaca CPU (Central Processing Unit) kullanılmıştır. Ana kontrolör, alt kontrolör ile USB protokolü üzerinden iletişim kurar. 32-bit ARM Cortex-M3 işlemciye sahip CM-740 alt kontrolör, cihazlara erişmek için bir ağ geçidi olarak çalışır.

Şekil 2.10'da gösterilen Dynamixel MX-28T akıllı servo motor, konum ve hız kontrolü için PID kontrolörünü destekler. Kullanıcı sadece konum ve hız profilini değil, aynı zamanda PID kazanç parametresini de gerçek zamanlı olarak ayarlayabilir. Dayanıklı metal redüktörlere sahip bu motor, 360° dönme açısını 12-bit çözünürlükte okuyabilen manyetik mutlak enkoder ile çok hassas hareket sunar. Tablo 2.5'te Dynamixel MX-28T servo motorun özellikleri yer almaktadır.

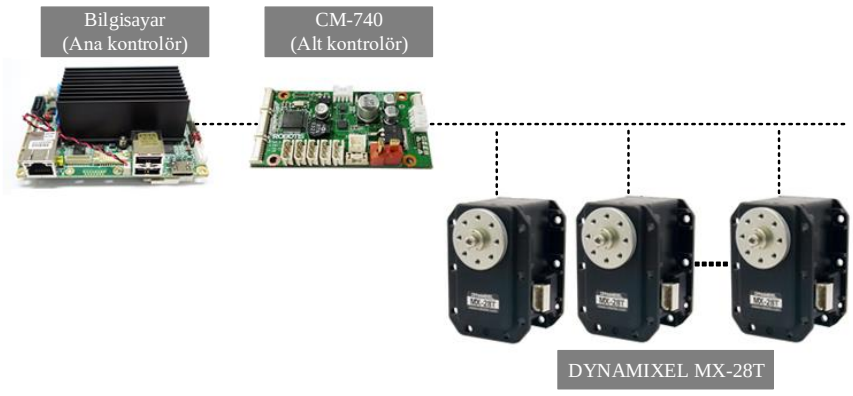
Şekil 2.10. Dynamixel MX-28T servo motor

Tablo 2.5. Dynamixel MX-28T servo motorun özellikleri

Çalışma gerilimi	10 ~ 14.8 V
Zorlanma torku @14.8 V	31.6 kg.cm (3.1 N.m)
Hız @14.8 V	67 devir/dk
Zorlanma akımı @14.8 V	1.7 A
Çözünürlük	0.088º
Ağırlık	72 g – 77 g
Redüksiyon oranı	193:1
Haberleşme	TTL&RS-485 / 4.5 Mbps

Robotis-OP2'nin yazılım yönü, modülerlik ve bağımsızlık dikkate alınarak hiyerarşik bir çerçeve ile oluşturulmuştur. Çerçeve, cihaz iletişim birimi, hareket birimi, yürüyüş birimi, algılama birimi, davranış birimi, görüş birimi ve teşhis biriminden oluşur. Çerçeve, kodun işletim sisteminden bağımsız olduğu C++ programlama dili ile geliştirilmiştir. Çerçevenin işletim sisteminden bağımsız yönü, kodun yeni geliştirilen robot işletim sistemi kısaca ROS (Robot Operating System) dahil olmak üzere mevcut veya gelecekteki herhangi bir bilgisayar işletim sistemine taşınabilmesi için çok önemlidir. Kullanıcı, ayrı bir çerçeve seti geliştirmeye gerek kalmadan Robotis-OP2 için davranışsal bir kod yazabilir.

Şekil 2.11'de Robotis-OP2'nin temel mimarisi verilmiştir.



Şekil 2.11. Robotis-OP2'nin temel mimarisi

2.2. Robotik Simülatörler

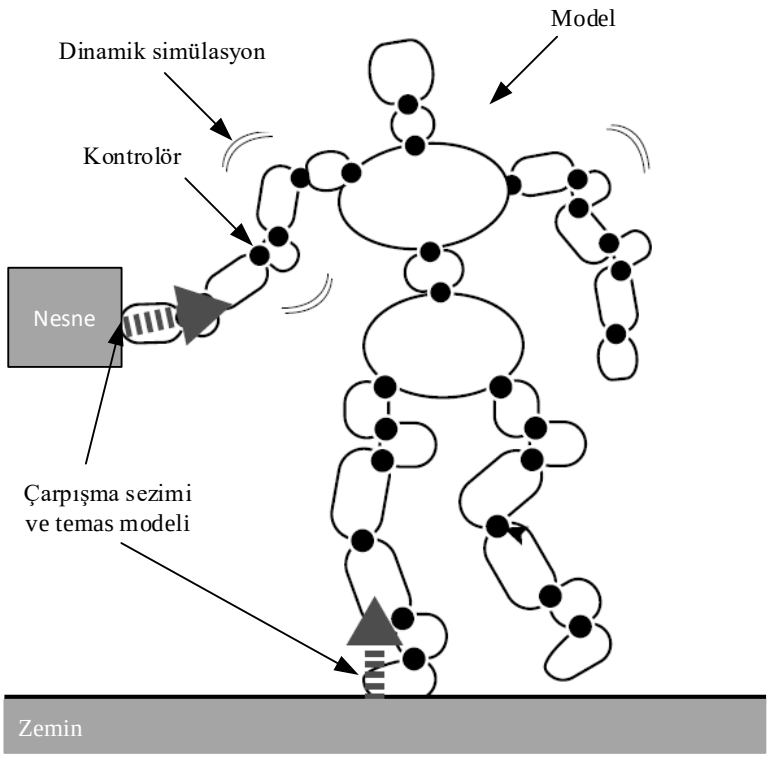
Robotik çözümlerin uygulanması, maliyetli ve zaman alıcı bir süreci temsil eder. Bu nedenle robotik simülatörleri, robotik çözümlerin geliştirilmesinin temel bir parçası olan önemli bir tamamlayıcı araç olarak ortaya çıkmıştır [135].

Robotik simülatörler, tür ve karmaşıklık bakımından farklılık gösteren algoritmaların fizibilitesini ve verimliliğini daha kontrollü bir ortamda, herhangi bir rahatsızlık olmaksızın kazaların oluşmasını önleyerek değerlendirmeye izin verir [136]. Robotik simülatörler, fiziksel robottan bağımsız olarak robotlara yüklenecek gömülü yazılım uygulamalarını test etmek amacıyla kullanılır. Robotik simülatörler sayesinde robotik sistemlerin farklı ortamlarda ve farklı donanımlar eklenerek test edilmesi gereken durumlar için robotun bilgisayar ortamında aldığı kararlar kolaylıkla gözlemlenebilir.

Modelleme, hareket planlaması ve kontrol yöntemleri, gerçek insansı robot ile deneyler yapılmadan bir robotik simülatöründe önceden test edilebilir. Robotik simülatörler, olası arızalara karşı kontrollü ortamlarda prototiplerin hızlı bir şekilde geliştirilmesine izin vererek, fiziksel dünya uygulanmalarına katkıda bulunmuştur [137]. Robotik simülatörler, gerçekçi simülasyon amacıyla fizik motorları içermektedirler.

2.2.1. Robotik Simülatörlerin Bileşenleri

Genel olarak bir robotik simülatör, Şekil 2.12'de gösterildiği gibi bir model yükleyici, çok gövdeli dinamiklere dayalı bir fizik motoru, temas modelleri ve bir görselleştirme arayüzü bileşenlerinden oluşur [138].

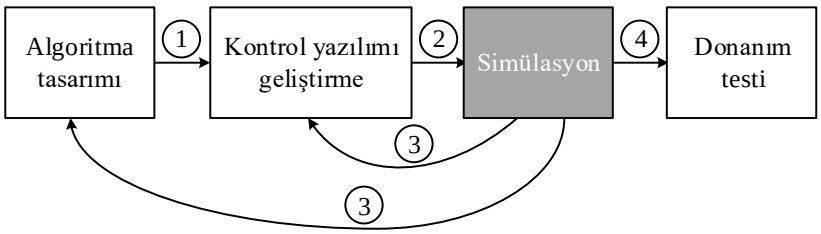


Şekil 2.12. Bir robotik simülatör için gerekli bileşenler [138]

Fizik motoru veya dinamik tabanlı simülasyon programı, çok gövdeli dinamiklere dayalı olarak insansı robotun davranışını hesaplayabilir. Davranışı hesaplamak için fizik motorunun katı cisim fiziği, çarpışma sezimi ve temas fiziği bileşenleri içermesi gerekir [139]. Simülasyon başlangıcında model yükleyici, kinematik ve dinamik parametreleri içeren robot modelini ayrıştırır ve bu bilgiyi fizik motoruna gönderir. Çarpışma sezimi programı, robot ile çevre arasındaki etkileşimi veya robotun kendi kendine çarpışmasını hesaplar [140, 141]. Bu programı kullanarak, robotun bağlantılarındaki çarpışma konumları ve ilgili tepki kuvvetleri, belirli temas kuvveti modeline göre hesaplanabilir [142]. Görselleştirme, robotik simülatörün önemli bir parçasıdır. Bir insansı robot genellikle birkaç eklem ve bağlantıdan oluşur. Üç boyutlu bilgisayar grafikleri, kullanıcının tasarlanan hareketi gerçekleştirirken ne olacağını bilmesine yardımcı olabilir. Bilgisayar grafiklerinin görüntülenmesi için OpenGL [143] başta olmak üzere birçok faydalı kütüphane vardır.

Birçok robotik simülatörde kullanıcılar, Python, C/C++ veya MATLAB gibi bir programlama dili kullanarak simülatörün fonksiyonlarına erişebilir. Bir insansı robotun dinamik tabanlı simülasyonuyla duruş kararlılığını değerlendirmek, robot ve çevre arasındaki kuvvetleri tahmin etmek ve istenen hareketi gerçekleştirmek için üretilen motor torklarını kontrol etmek mümkündür. Kullanıcı, simülatör ile iletişim kuran kontrolörü bir programlama dili kullanarak programlar. Kontrolör simülatörde test edilir ve gerekirse simülasyon sonuçlarına göre değiştirilir. Robotik simülatörlerde geliştirilen uygulama, herhangi bir değişikliğe ihtiyaç duymadan doğrudan gerçek robota aktarılabilir.

İnsansı bir robotun kontrolörünün bir robotik simülatör kullanarak geliştirilmesi, son zamanlarda oldukça önem arz etmektedir. İnsansı bir robotu kontrol etmek için kullanıcı, amaçlanan görev için bir kontrolör hazırlamalıdır. Yeni bir algoritma önerilmişse veya mevcut bir kontrolör program yoksa bir robot kontrolörünün geliştirilme işlemi Şekil 2.13'e göre ilerleyecektir.



Şekil 2.13. Bir robot kontrolörünün geliştirilme iş akışı

Şekil 2.13'te belirtildiği gibi ilk adım olarak, kontrol için bir model veya algoritma tasarlanır. İkinci adım, modeli veya algoritmayı bir kontrolör programı olarak uygulamaktır.

Çoğu durumda, bu aşamada kullanıcı eklem açısı sınırları ve eklem torku sınırları gibi donanıma özgü sınırlamaları dikkate almalıdır. Kontrolörün çeşitli robotlar için kullanılabilmesi için donanıma özgü problemleri genelleştirerek evrensel bir kontrolör geliştirilmesi arzu edilir. Bu nedenle, robotun yapısını tanımlayan dosyaların hazırlanması da istenir. Bu amaç için SDF (Simulator Description Format), URDF (Universal Robotic Description Format), YAML (YAML Ain't a Markup Language) ve VRML (Virtual Reality Modeling Language) gibi belirli formatlar veya programlama dilleri vardır. Daha sonra dosyalar, simülatör programı tarafından kolayca alınabilir. Üçüncü adım olarak, kontrolör robotik simülatörde değerlendirilir. Kontrolör algoritmanın uygulanmasının bu aşamasında bulunan herhangi bir sorun gözden geçirilmelidir. Dördüncü ve son adımda, geliştirilen kontrolör, gerçek insansı robot kullanılarak değerlendirilir. Başarılı bir test sonucuna rağmen robotik simülatör, gerçek robotunkiyle tamamen eşleşen bir robot davranışını garanti edemediğinden bu aşama vazgeçilmezdir.

Son yıllarda, farklı robot türlerinin kullanımı için mevcut robotik simülatörlerin sayısı önemli ölçüde artmıştır [144]. Yaygın olarak kullanılan simülatörlerin özellikleri Tablo 2.6'da listelenmiştir.

Tablo 2.6. Yaygın olarak kullanılan simülatörlerin özellikleri

Simülatör	Geliştirici	Fizik motoru	3B format desteği	Ana programlama dili	Arayüz desteği	Desteklenen işletim sistemi
Gazebo [145]	Open Robotics	ODE, Bullet, Simbody, DART	SDF/URD, OBJ, STL, Collada	C++	C++	Linux, macOS, Windows
RoboDK [146]	RoboDK	Gravity plugin	SLDPRT, SLDASM, STEP, OBJ, STL, 3DS, Collada, VRML, URDF	Python	Python, C/C++, C#, MATLAB	Linux, macOS, Windows, Android, iOS, Debian
OpenRAVE [147]	OpenRAVE Community	ODE, Bullet	XML, VRML, OBJ, Collada	C++, Python	C/C++, Python, MATLAB	Linux, macOS, Windows
CoppeliaSim (V-REP) [148]	Coppelia Robotics	ODE, Bullet, Vortex, Newton dynamix	3DS, Blender, Collada, STL, OBJ, URDF, SDF, GLTF, XML	C++, Python, Lua	C, C++, Python, Java, MATLAB, Octave, ROS	Linux, macOS, Windows
Webots [38]	Cyberbotics Ltd.	ODE	WBT, VRML, X3D, 3DS, Blender, BVH, Collada, FBX, STL, OBJ, URDF	C++	C, C++, Python, Java, MATLAB, ROS	Linux, macOS, Windows

DPÖ, robotların karmaşık ve hassas görevleri yerine getirebilmek üzere eğitilmesi için giderek daha fazla kullanılırken, gerçekçi simülatörlerin geliştirilmesi robotik için DPÖ araştırmalarının hızlanmasına katkıda bulunmaktadır [149]. DPÖ'nün robotik üzerindeki potansiyeline rağmen, bu tür yaklaşımlar genellikle çevreyi yeterince araştırmak ve görevi çözmeyi başarmak için çok fazla zaman gerektirir ve genellikle düşük örnek verimliliğinden muzdariptir [150]. Ayrıca, eğitimin ilk aşamalarında ajanlar rastgele eylemler gerçekleştirerek robotun donanımını potansiyel olarak tehlikeye atar. Bu kısıtlamaları aşmak için, araştırmacılar genellikle ilk olarak simülasyonun hızlandırılmış hızlarda ve tehlikesiz bir şekilde çalışabileceği Gazebo ve OpenRAVE gibi gerçekçi simülatörler üzerinde eğitim oturumları yürütür, daha sonra eğitilmiş ajanları fiziksel robotlara transfer etmek için simülatörler kullanılır. Bununla birlikte, simülasyonu yapılmış ortamların değişen derecelerde gerçekçilik sağlaması nedeniyle ek zorluklar [151] ortaya çıkarmaktadır, bu nedenle ajan için gerçek dünyada eğitim sırasında olduğu gibi tam olarak gözlemlemesi ve hareket etmesi her zaman mümkün değildir. Bu durum, Webots ve Actin [152] gibi simülasyon ile gerçek dünya arasındaki boşluğu daha da azaltan daha gerçekçi simülatörlerin geliştirilmesine yol açmıştır. Bu simülatörlerin sadece bir ortamın fiziksel özelliklerini simülasyon yapmakla ve dünyanın fotogerçekçi bir temsilini sağlamakla kalmayıp, aynı zamanda farklı gerçek yaşam senaryolarının ihtiyaçlarına göre ayarlanabilen, kolayca parametrelenebilir bir ortam sağladığına dikkat edilmelidir. Actin simülatörü, lisanslı bir ürün olup çoğunlukla endüstriyel amaçlar için tercih edilmektedir. Ayrıca, bu simülatör bünyesinde insansı robot modelleri de halihazırda bulunmamaktadır. Tüm bu sebeplerden dolayı, tez çalışması kapsamında Webots simülatörünün kullanımının uygun olduğuna karar verilmiştir.

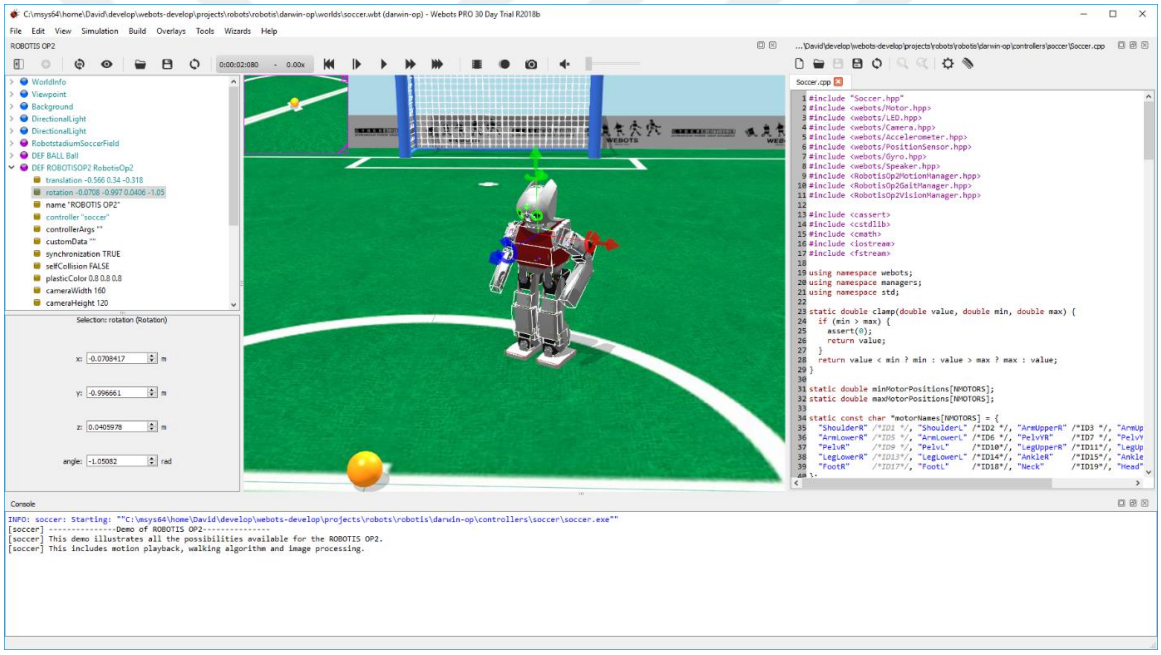
2.2.2. Webots

Webots, mobil robotların simülasyonu için kullanılan açık kaynaklı ve çok platformlu bir simülatördür. Cyberbotics Ltd. tarafından 1998 yılından beri düzenli olarak geliştirilmektedir. Profesyonel bir kullanım için tasarlanmış olup endüstride, eğitimde ve araştırmalarda yaygın olarak kullanılmaktadır.

Robotları modellemek, programlamak ve simülasyon yapmak için eksiksiz bir geliştirme ortamı sağlar. Simülatör, kullanıcıya robot, aktüatör, sensör, nesne ve malzeme gibi çok çeşitli bileşen olanağı sunmaktadır. Kullanıcının kütle, eklemler, sürtünme katsayıları vb. fizik özellikleriyle 3B (3 Boyutlu) sanal dünyalar oluşturmasına olanak tanıyan hızlı bir prototipleme imkanı vermektedir. Kullanıcı, istenen davranışı sergilemek için her robotu ayrı ayrı programlayabilir. Ayrıca, gerçek mobil ve insansı robotlar için bir dizi arayüz içermektedir. Böylece, simülasyonu yapılmış olan robot beklendiği gibi davrandığında, kontrol programı gerçek bir robota aktarılabilmektedir.

E-puck ve iCub gibi mobil robotlar ile Atlas, DARwIn-OP, Robotis-OP2, Robotis-OP3, HOAP-2 ve NAO gibi birçok tanınmış insansı robot modelleri Webots'ta bulunmaktadır.

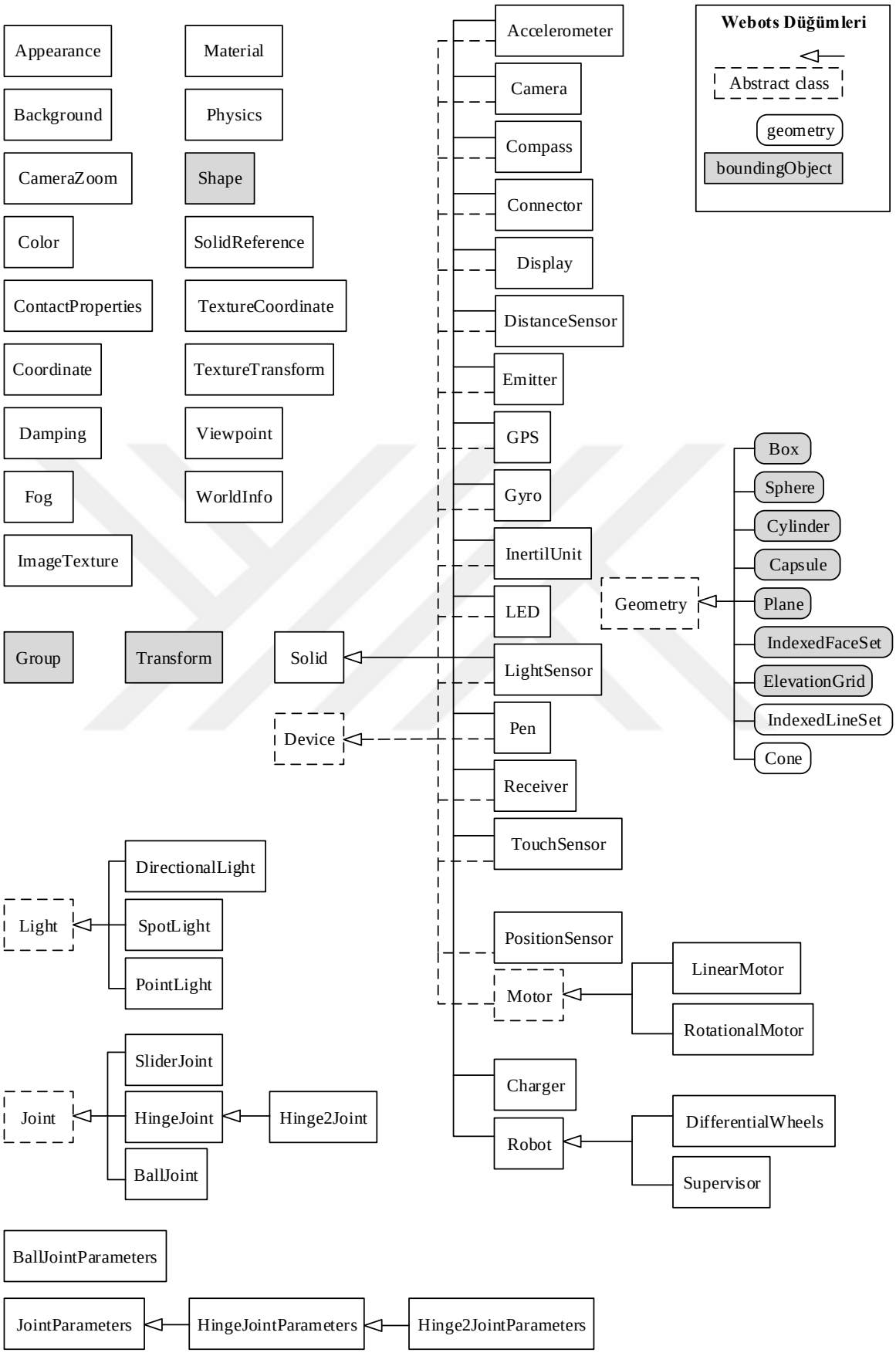
Kullanıcı arayüzü Şekil 2.14'te gösterilen Webots için robot kontrolörü, yerleşik tümleşik geliştirme ortamı kısaca IDE (Integrated Development Environment) sayesinde C/C++, Python, Java veya MATLAB ile yazılabilir ve derlenebilir. ROS ile iletişim kurmak için uygulama arayüzü bulunmaktadır. Webots, gerçekçi bir simülasyon için fizik motorunda çekirdek kütüphane olarak ODE'nin (Open Dynamics Engine) genişletilmiş bir sürümünü kullanır [153]. Simülasyon yapılmış modeller, doku, kütle, sürtünme ve şekil gibi farklı ve özelleştirilebilir niteliklere sahiptir. Bu öznitelikler, katı gövde dinamiklerini simülasyon yapmak için kullanılan ODE kütüphanesi tarafından sağlanır.



Şekil 2.14. Webots simülatörünün kullanıcı arayüzü [154]

Webots Düğümleri

Şekil 2.15'te Webots düğümlerini gösteren çizelgede, iki düğüm arasındaki bir ok, bir kalıtım ilişkisini temsil eder. Kalıtım ilişkisi, türetilmiş bir düğümün (ok kuyruğunda), bir temel düğümün (ok başında) tüm alanlarını ve uygulama programlama arayüzü kısaca API (Application Programming Interface) fonksiyonlarını devraldığını gösterir.



Şekil 2.15. Webots düğümlerinin çizelgesi [155]

Kesik çizgiyle gösterilen kutular soyut düğümleri, yani somutlaştırılamayan düğümleri temsil eder. Soyut düğümler, türetilmiş düğümler tarafından paylaşılan ortak alanları ve fonksiyonları gruplamak için kullanılır. Yuvarlak köşeli kutu, bir geometri düğümünü temsil eder. Gri arka plana sahip kutu ise katı nesneler arasındaki çarpışmaları algılamak için kullanılan bir sınırlayıcı nesne oluşturmak için doğrudan kullanılabilen bir düğümü belirtir.

Webots Simülatöründe Robotis-OP2

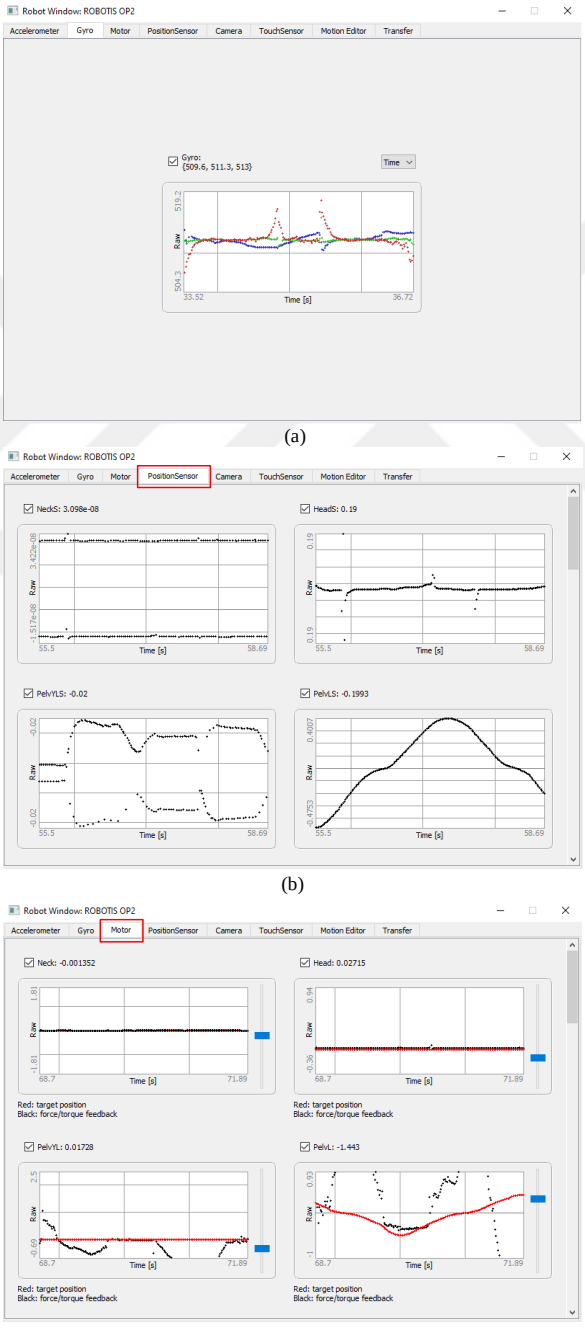
Webots'ta robotun sayısal modeli, "proto" adlı bir dosyada bulunur. Robotis-OP2'nin "proto" dosyası, "RobotisOp2.proto" ismiyle bulunmaktadır. Bu dosya, robotu tanımlamak için gereken her şeyi içermektedir. Robotun sensörleri, mekanik parçaları ve aktüatörleri ile adları, yapılandırmaları, konumları ve hatta fiziksel özellikleri (renk, kütle, atalet vb.) bu dosyada tanımlanır.

Robotis-OP2, Webots'un en iyi kalibre edilmiş robotlarından biridir. Kalibrasyon sırasında gerçek ve simülasyonu yapılmış olan ortam arasındaki bazı uyumsuzluklara dikkat çekilerek düzeltilmiştir. Webots'ta Robotis-OP2'nin fiziksel modeli çok gerçekçi olup, kendi kendine çarpışma kontrolü mevcuttur. Kendi kendine çarpışma kontrolü yalnızca ihtiyaç olduğunda kullanılmalıdır. Çünkü bu, hesaplama açısından çok maliyetlidir ve bu nedenle simülasyon hızını önemli ölçüde yavaşlatabilir.

Robotis-OP2'deki her servonun başlangıç konumu 0'dır ve bu durumda dik konuma karşılık gelir. Ancak, gerçek robotun başlangıç konumu için ayakta durma pozisyonu tavsiye edilen bir pozisyon değildir, çünkü servolar kapalıyken sabit bir pozisyon değildir. Robotun kolayca ayağa kalkabileceği tek sabit pozisyon olarak, Şekil 2.16'daki gibi bir oturma pozisyonu uygun görülmektedir [156].

Şekil 2.16. Robotis-OP2'nin kararlı başlangıç konumunda görünümü

Webots'ta bulunan her bir robot için robotla etkileşimi sağlayan özel bir robot penceresi bulunmaktadır. Bu pencere, kullanıcının robot üzerindeki ivmeölçer, jiroskop, enkoder ve kamera gibi sensörlerin verileri ile her bir eklemdeki motorun tork/kuvvet verilerinin gözlemlenmesine ve geliştirilen kontrolörün robota aktarılmasına olanak tanımaktadır. Örnek sensör ve kuvvet/tork verilerine ait görseller Şekil 2.17'de verilmiştir.



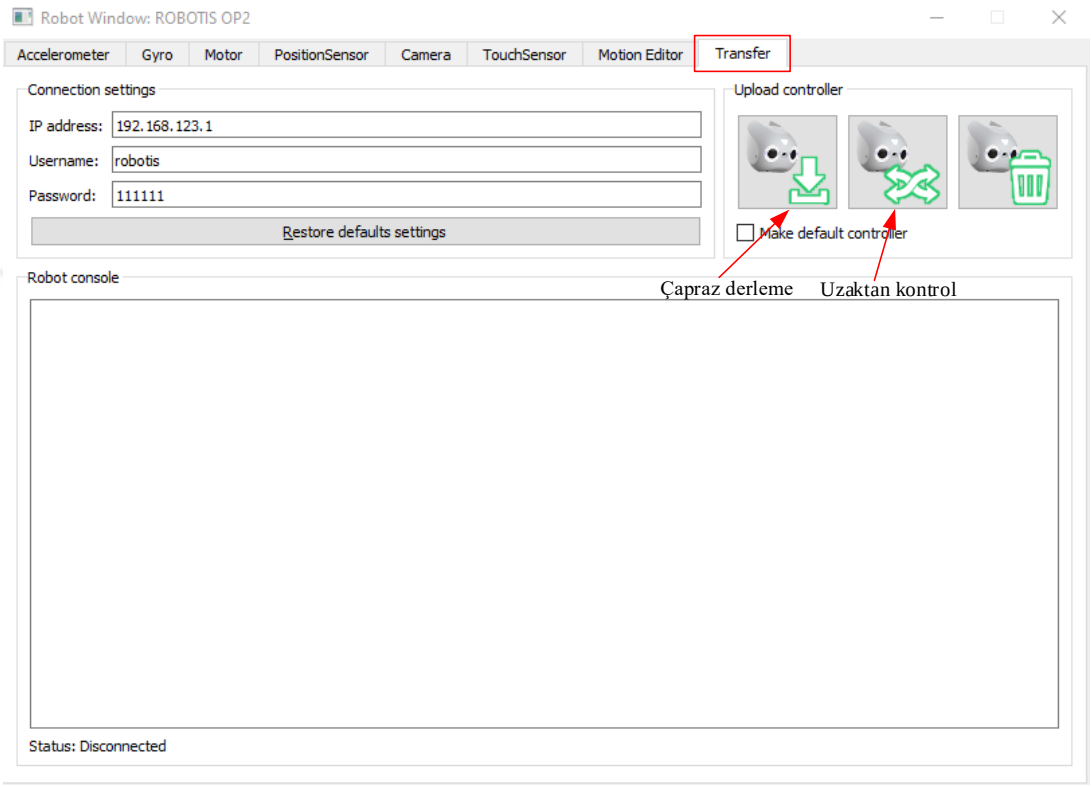
(a)

(b)

(c)

Şekil 2.17. Derleyici aktarım sekmesindeki Robotis-OP2 robot penceresi

Şekil 2.17'de belirtilen sekmeler, Webots'un genel robot penceresindeki ile oldukça benzerdir ve Webots tarafından sağlanan "qt_utils" kütüphanesi sayesinde kolayca yeniden kullanılabilirler. Ancak, sonuncu sekme olan derleyici aktarım sekmesi, Robotis-OP2'ye özeldir. Robotis-OP2'ye ait derleyici aktarım sekmesi ise Şekil 2.18'de verilmiştir.



Şekil 2.18. Derleyici aktarım sekmesindeki Robotis-OP2 robot penceresi

Şekil 2.18'de gösterildiği gibi Webots ortamında geliştirilen bir kontrolörün robota aktarılmasını sağlayan iki araç bulunmaktadır. Bunlardan ilki olan çapraz derleme aracı, herhangi bir değişiklik yapılmadan Webots kontrolörünün gerçek robot üzerinde kullanılmasına izin vermek için Webots arayüzü ile Robotis arayüzü arasında köprü oluşturan bir birimdir. Bu araç, tüm Webots fonksiyonlarının bir paketleyicisinden oluşur ve farklı birimler halinde düzenlenir. İkinci olarak uzaktan kontrol aracı ise kontrolörlerin, kişisel bilgisayarda Webots'tan çalıştırılabilmesi ve robotla etkileşime girilebilmesi için geliştirilmiştir. Bu araç, sensörlerin durumunun alınmasına ve Webots'ta çalışan bir kontrolörden gerçek robotun aktüatörlerine gerçek zamanlı olarak komutların gönderilmesine olanak tanır.

Robotis çerçevesinin tüm temel fonksiyonlarını simülasyonda uygulamak için Webots'a bir kütüphane sağlanmıştır. Bu kütüphane, üç bölüme ayrılmıştır. Her bölüm, çerçevenin bir birimini uygular. Yürüyüş birimi olarak adlandırılan ilki, Robotis çerçevesinin yürüme algoritmasının

kullanımına izin verir. Hareket birimi olarak adlandırılan ikincisi, “motion_4096.bin” dosyasında saklanan önceden tanımlanmış hareketlerin oynatılmasına izin verir. Görüş birimi olarak adlandırılan sonuncusu, örneğin kamera görüntüsünde renkli bir topu bulabilmek için yararlı olan bazı görüntü işleme araçlarını içerir. Yürüyüş birimi, tez çalışması kapsamında fayda sağlayabileceği için detaylı olarak incelenmiştir.

Robotis çerçevesinin yürüme algoritmasının kullanımını sağlar. Robotis-OP2’nin kendi içindeki yürüyüş birimi, sinüzoidal yörüngeler gerçekleştiren Merkezi Örüntü Üreteci (MÖÜ) tabanlı eşleştirilmiş osilatörlere dayalı yürüyüş modeli oluşturmak için bir yöntem kullanır [157]. Yürüyüşü ayarlamak için algoritmada birçok parametre mevcuttur. Ancak, bunun kullanıcı tarafından kullanımını kolaylaştırmak için bu parametrelerin yalnızca bir alt kümesinin ayarlanabileceğine karar verilmiştir [158]. Diğer parametreler, iyi çalıştığı bilinen varsayılan değerlere ayarlanmış ve kullanıcı isterse yalnızca “config.ini” yapılandırma dosyasında depolanan varsayılan değerleri değiştirerek robotun yürüyüşü ayarlanabilmektedir [158].

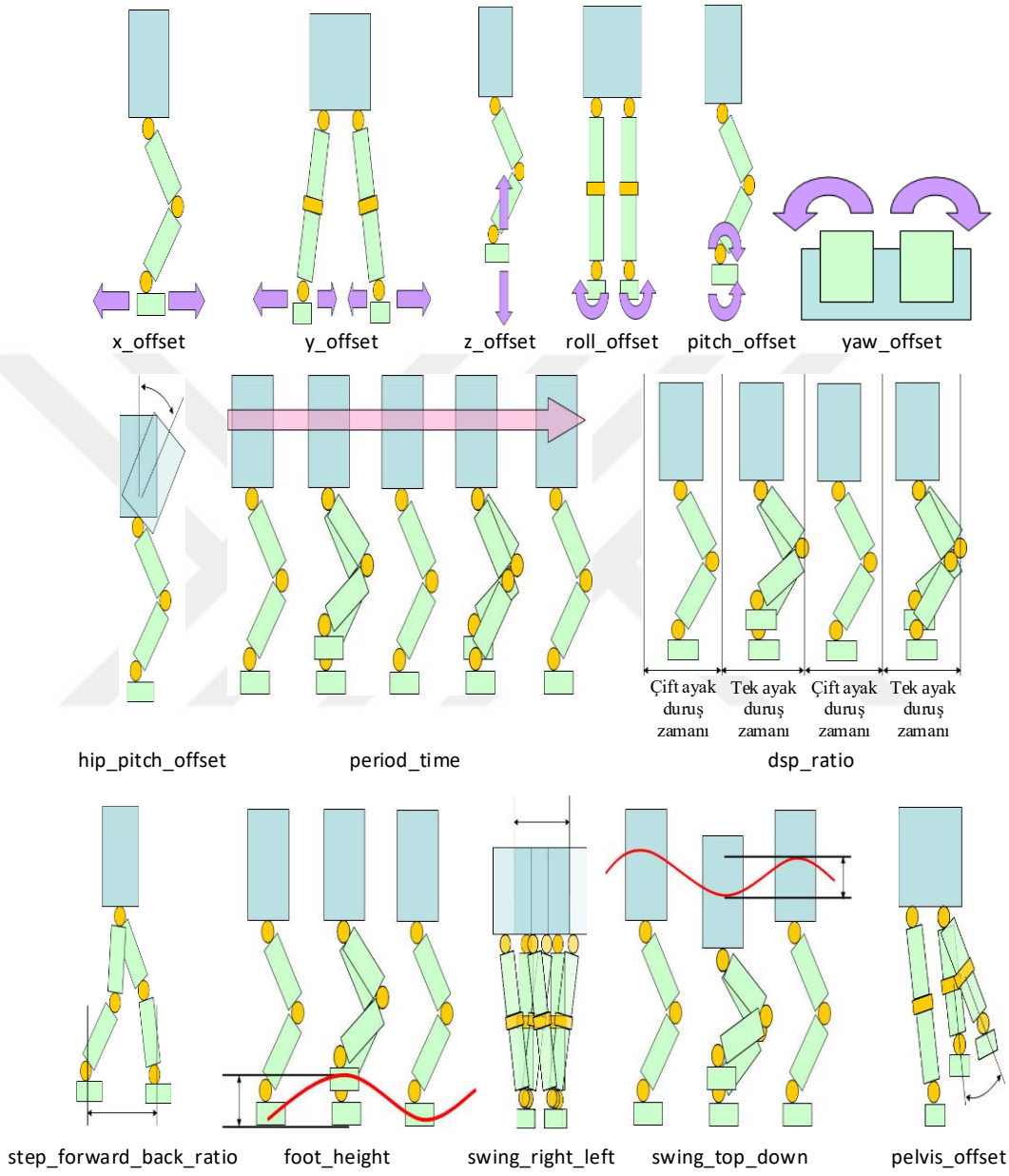
Yapılandırma dosyasının varsayılan değerlerden oluşan tipik bir örneği (config.ini) Şekil 2.19’daki gibidir.

```
[Walking Config]
x_offset                    = -10.0;
y_offset                    = 5.0;
z_offset                    = 20.0;
roll_offset                 = 0.0;
pitch_offset                = 0.0;
yaw_offset                  = 0.0;
hip_pitch_offset            = 13.0;
period_time                 = 600.0;
dsp_ratio                   = 0.1;
step_forward_back_ratio     = 0.28;
foot_height                 = 40.0;
swing_right_left            = 20.0;
swing_top_down              = 5.0;
pelvis_offset               = 3.0;
arm_swing_gain              = 1.5;
balance_knee_gain           = 0.3;
balance_ankle_pitch_gain    = 0.9;
balance_hip_roll_gain       = 0.0;
balance_ankle_roll_gain     = 0.0;

[Robot Config]
time_step                   = 16.0;
camera_width                = 320.0;
camera_height               = 240.0;
```

Şekil 2.19. Yürüyüşü ayarlamak için yapılandırma dosyasındaki varsayılan parametreler

Yürüyüşü ayarlamak için yapılandırma dosyasındaki yürüyüş parametreleri sırasıyla aşağıdaki gibi açıklanmış olup Şekil 2.20'de parametreler görsel olarak ifade edilmiştir.



Şekil 2.20. Yürüyüş parametrelerinin gösterimi [158]

- x ofseti (x_offset), ayakların x eksenindeki ofsetidir. Birimi milimetredir.
- y ofseti (y_offset), ayakların y eksenindeki ofsetidir. Birimi milimetredir.
- z ofseti (z_offset), ayakların z eksenindeki ofsetidir. Birimi milimetredir.
- Yuvarlanma ofseti (roll_offset), x ekseni boyunca ayaklardaki açı ofsetidir. Birimi derecedir.

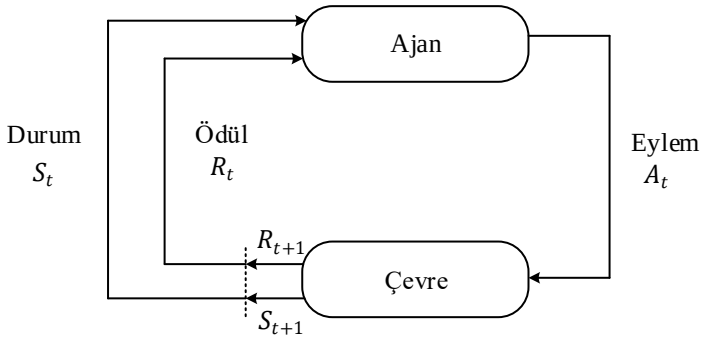
- Yunuslama ofseti (pitch_offset), y ekseni boyunca ayaklardaki açı ofsetidir. Birimi derecedir.
- Sapma ofseti (yaw_offset), bacağın z ekseni boyunca açı kaymasıdır. Birimi derecedir.
- Kalça yunuslama ofseti (hip_pitch_offset), Robotis-OP2'nin vücudunun eğimidir. 2.85 dereceye karşılık gelen motorun özel bir birimi kullanılır.
- Periyot süresi (period_time), Robotis-OP2'nin iki tam adımı (sol ve sağ ayak) tamamlaması için gereken süredir. Birim milisaniyedir.
- DSP (Double Support Phase–Çift Destek Fazı) oranı (dsp_ratio), her iki ayağın yerdeyken yalnızca bir ayağa (sola veya sağa) yerleştiği zaman arasındaki orandır.
- İleri-geri adım oranı (step_forward_back_ratio), Robotis-OP2'nin yürüyüş sırasında sağ ve sol ayağı arasındaki x eksenine göre fark mesafesidir. Birimi milimetredir.
- Ayak yüksekliği (foot_height), adım sırasında ayağın maksimum yüksekliğidir. Birimi milimetredir.
- Sağa-sola sallanma (swing_right_left), yürüyüş sırasında Robotis-OP2'nin gövdesinin sallanma miktarıdır. Birimi milimetredir.
- Yukarı-aşağı sallanma (swing_top_down), yürüme sırasında Robotis-OP2'nin gövdesinin yukarı ve aşağı sallanma miktarıdır. Birimi milimetredir.
- Pelvis ofseti (pelvis_offset), x ekseni boyunca pelvisteki açı ofsetidir. 2.85 dereceye karşılık gelen motorun özel bir birimini kullanır.
- Kol sallanma kazancı (arm_swing_gain), yürüme sırasında kol hareketini etkileyen kazançtır.
- Diz dengeleme kazancı (balance_knee_gain), ön/arka denge için diz seviyesindeki kazançtır.
- Ayak bileği dengeleme yunuslama kazancı (balance_ankle_pitch_gain), ön/arka dengesi için ayak bileği seviyesindeki kazançtır.
- Ayak bileği dengeleme yunuslama kazancı (balance_ankle_roll_gain), ön/arka dengesi için ayak bileği seviyesindeki kazançtır.
- Ayak bileği dengeleme yuvarlanma kazancı (balance_ankle_roll_gain), yanal dengeleme için ayak bileği seviyesindeki kazançtır. Yanal dengeleme, simülasyonda çok iyi çalışmadığından bu parametrenin 0 olarak ayarlanması önerilmektedir [158].

3. PEKİŞTİRMELİ ÖĞRENME

Pekiştirmeli Öğrenme PÖ (Reinforcement Learning–RL), gözetimli ve gözetimsiz öğrenme ile birlikte üç temel makine öğrenmesi yaklaşımından biridir. PÖ, çevreye dayalı olarak ne yapılacağını ve durumların nasıl eyleme dönüştürüleceğini öğrenmektir. PÖ, gözetimli ve gözetimsiz öğrenmede olduğu gibi statik verileri kullanmak yerine, çevreden alınan dinamik bir veri kümesiyle çalışır. PÖ'deki amaç, verileri etiketlemek veya kümelemek değil, optimum sonucu üretecek optimum eylem dizisini haritalamaktır. PÖ bunu, ajan adı verilen bir yazılım parçasının çevreyi keşfetmesine, etkileşime girmesine ve çevreden öğrenmesine izin vererek gerçekleştirir.

PÖ, $s \in S$ girişleri (durumlar) $a \in A$ çıkışlara (eylemlere) eşleyen bir $\pi: S \rightarrow A$ politika fonksiyonunu (ajan stratejisi) öğrenmedir. Öğrenme, π politikasına bağlı olan tüm olası durumlar için $V^{\pi}(s)$ değer fonksiyonunun (biriken ödül) en yükseğe çıkarılmasıyla elde edilir. Bu anlamda gözetimsiz öğrenmeye benzerdir. Ancak fark, $V^{\pi}(s)$ değer fonksiyonunun gözetimsiz öğrenmedekinden farklı olarak tam olarak tanımlanamamasıdır.

Şekil 3.1'de verilen temel bir PÖ yapısında, ajana hangi eylemleri yapması gerektiği söylenmez, bunun yerine ajanın en yüksek ödül verecek eylemleri öğrenmesi veya keşfetmesi gerekir. Genel olarak, pekiştirmeli öğrenme bir optimizasyon problemidir.



Şekil 3.1. Temel bir pekiştirmeli öğrenme yapısı [45]

Ajan, çevrenin mevcut durumunu gözlemler. Gözlemlenen durumdan, hangi eylemin yapılacağına karar verir. Çevre durumu değiştirir ve bu eylem için ödül üretir. Her ikisi de ajan tarafından alınır. Ajan, bu yeni bilgiyi kullanarak, bu eylemin iyi olup olmadığını ve tekrarlanması gerekip gerekmediğini veya kötü olup olmadığını ve kaçınılması gerektiğini belirleyebilir. Gözlem-ödül-eylem döngüsü, öğrenme tamamlanana kadar devam eder.

PÖ'nün amacı, geleneksel bir kontrol problemine benzerdir. Bu sadece farklı bir yaklaşımdır ve aynı kavramları temsil etmek için farklı terimler kullanır. Her iki yöntemle de, istenen sistem davranışını üretecek bir sisteme doğru girişlerin belirlenmesi istenir.

Ajan içinde, durum gözlemlerini giriş olarak alan ve bunları eylemlere çıkış olarak eşleyen bir fonksiyon vardır. PÖ terminolojisinde bu fonksiyona politika adı verilir. Bir dizi gözlem göz önüne alındığında, politika hangi eylemin gerçekleştirileceğine karar verir. Mükemmel bir politika bulunmuş olsa bile, çevre zamanla değişebilir, bu nedenle statik bir haritalama artık optimal olmaz. PÖ, politikayı alınan eylemlere, çevreden gelen gözlemlere ve toplanan ödül miktarına göre değiştirir. Ajanın amacı, çevreyle etkileşime girerken optimum politikayı öğrenmek için pekiştirmeli öğrenme algoritmalarını kullanmaktır. Böylece herhangi bir durumda, her zaman optimal eylemi yani uzun vadede en yüksek ödülü üretecek olanı alacaktır. Yeterli bir politika yapısı verildiğinde, optimal bir politika üretecek bir dizi parametre vardır. Öğrenme, optimal politikaya yaklaşmak için bu parametreleri sistematik olarak ayarlama sürecine verilen terimdir.

3.1. Pekiştirmeli Öğrenmenin Temel Bileşenleri

PÖ uygulamalarındaki temel bileşenler aşağıda verilmiştir:

3.1.1. Ajan

Akıllı kararlar verebilen yazılım programlarıdır ve temel olarak PÖ'de öğrenme işlemini gerçekleştiren temel bileşenlerden biridir. Ajanlar, çevre ile etkileşime girerek eylemlerini gerçekleştirirler ve eylemlerine göre ödüller alırlar. Örneğin; bir video oyununda gezinen bir karakter olarak düşünülebilir.

3.1.2. Eylem

Bir ajanın gerçekleştirebileceği olası eylemler kümesini tanımlar. Böylece ajan, belirli bir durumda her eylemin yürütülmesi üzerine alacağı ödülü tahmin edebilir. Örneğin; otonom bir araç için herhangi bir durumda olası eylemler, hızlanmak, yavaşlamak, sola gitmek, sağa gitmek, düz gitmek veya hiçbir şey yapmamak olabilir. Eylem, genellikle a sembölü ile gösterilir.

3.1.3. Çevre

Ajanın eylemleri gerçekleştirerek ve ödülleri toplayarak etkileşimde bulunduğu alandır. Çevre, gerçek bir fiziksel çevre veya simülasyon olabilir. İkisi arasında seçim yapmak duruma göre değişir. Hiçbir şey çevreyi gerçek bir çevreden daha eksiksiz temsil edemez. Gerçek çevrede, bir model oluşturmak ve doğrulamak için zaman harcanmasına gerek yoktur. Simülasyonlu çevre ise

gerçek zamandan daha hızlı çalışabilir veya paralelleştirilebilir, bu da yavaş olan bir öğrenme sürecini hızlandırır. Ayrıca simülasyonlu çevrede, test edilmesi zor olan durumları modellemek daha kolaydır ve donanıma zarar verilme riski yoktur. Örneğin; otonom bir sürüşte çevre, ajanın geçtiği caddeyi ve bu caddelerdeki trafiği temsil eder.

Çevre türleri aşağıda verilmiştir:

Deterministik Çevre

Mevcut duruma dayalı sonuç bilindiğinde bir çevrenin deterministik olduğu söylenebilir. Örneğin; bir satranç oyununda herhangi bir taşı hareket ettirmenin kesin sonucu bilinmektedir.

Stokastik Çevre

Mevcut duruma dayalı sonucun belirlenemediği çevrenin stokastik olduğu söylenebilir. Örneğin; zar atıldığında hangi sayının çıkacağı asla bilinemez.

Tamamen Gözlemlenebilir Çevre

Bir ajan sistemin durumunu her zaman belirleyebiliyorsa, buna tamamen gözlemlenebilir denir. Örneğin; bir satranç oyununda sistemin durumu, yani tüm taşların satranç tahtasındaki konumu, oyuncunun optimal bir karar verebilmesi için her zaman mevcuttur.

Kısmen Gözlemlenebilir Çevre

Bir ajan sistemin durumunu her zaman belirleyemediğinde, buna kısmen gözlemlenebilir denir. Örneğin; bir poker oyununda rakibin elindeki kartların ne olduğu hakkında bilgi sahibi olunamaz.

Ayrık Çevre

Bir durumdan diğerine geçmek için yalnızca sonlu bir eylem durumu olduğunda, buna ayrık çevre denir. Örneğin; bir satranç oyununda yalnızca sınırlı sayıda hamle bulunmaktadır.

Sürekli Çevre

Bir durumdan diğerine geçmek için sonsuz bir eylem durumu olduğunda, buna sürekli çevre denir. Örneğin; bir hedef şehre seyahat etmek için birden fazla rotanın bulunması durumudur.

3.1.4. Durum

Ajana farklı biçimlerde gösterilebilen, ajanın çevredeki mevcut durumunun açıklamasıdır. Örneğin; otonom bir aracın izlediği yol, o yoldaki aracın mevcut konumu, en yakın araç ve önündeki engeller hakkında bilgiler durum olarak değerlendirilebilir. Durum, genellikle s sembolü ile gösterilir.

3.1.5. Ödül

Belirli bir durumda olan ve belirli bir eylemde bulunan bir ajanın "iyiliğini" temsil eden skaler bir sayı üreten bir fonksiyondur. Ödül, ajan tarafından seçilen eylem için çevreden gelen geri bildirimi temsil eder. Daha yüksek ödül değerleri, mevcut durum için daha uygun eylemleri gösterirken, daha düşük değerler, karşılık gelen eylemlerin mevcut durum için daha az uygun olduğunu gösterir. Ödül fonksiyonu sadece duruma veya durum-eylem çiftine bağlı olabilir. Ödül fonksiyonu sayesinde öğrenme algoritması, politikanın ne zaman daha iyi hale geldiğini anlar ve nihayetinde aranılan sonuca yakınsar. Ödül, genellikle r sembölü ile gösterilir. Ödül, deterministik veya stokastik şekilde olabilir.

3.1.6. Politika

Ödülü en üst düzeye çıkarmak için mevcut duruma uygun eylemi seçmek için ajan tarafından kullanılan yaklaşımdır. Eylem seçimindeki kural olarak da düşünülebilir. Bir politika fonksiyonu, bir arama tablosu veya karmaşık bir arama süreci şeklinde olabilir. Politika genellikle π sembolü ile gösterilir. Politika, deterministik veya stokastik şekilde olabilir.

3.1.7. İndirim Faktörü

Ajanın anlık ödüle odaklanmak yerine toplam ödülü en yükseğe çıkarmaya odaklanmasını sağlamak için kullanılır. Bir eylem yürütüldüğünde ajanın içine girdiği yeni durumdan alınan en yüksek ödül, mevcut ödüllerin hesaplanmasına dahil edilir. Bir sonraki durumun ödül değeri, indirim faktörü ile çarpılarak azaltılır, böylece anlık ödül ve toplam ödül etkisi dengelenir. Örneğin; otonom bir araç sadece anlık değerlere göre ödüllendirilirse riskli durumlarda hızlanma, en yüksek anlık ödülle sonuçlanamayacağı için ajan tarafından dikkate alınmaz. Son ödüllerin hesaplamalara dahil edilmesi, ajanın bu şekilde uygun kararlar vermesine olanak tanıyan kazalardan kaçınmaktan beklenen ödülü arttırır. Ayrıca, yalnızca son ödüle güvenmek, ajanı son ödülü en üst düzeye çıkarmak için yoldan çıkmak gibi bazı istenmeyen eylemlerde bulunmaya teşvik edebilir. Bu nedenle, tüm senaryoları dengelemek ve ajandan optimum performansı elde etmek için uygun bir indirim faktörü seçilmelidir. İndirim faktörü genellikle γ sembolü ile gösterilir.

3.1.8. Değer

π politikası kapsamında, ajan için tanımlanan indirim faktörü dikkate alınarak, mevcut durum için ajan tarafından beklenen uzun vadeli ödüldür. Ajanın, anlık ödülü en üst düzeye çıkarsa bile, uzun vadeli ödülü önemli ölçüde azaltabilecek durumlarda olmaktan kaçınmasını sağlar. Örneğin; otonom bir sürüşte hızın hız sınırının üzerine çıkarılması, daha fazla mesafe daha hızlı kat

edildiğinden anlık ödülü artırabilir. Ancak, para cezası veya kaza olasılığını göz önünde bulundurmak, ajanın daha makul kararlar vermesine olanak tanır.

Değer, politikaya bağlıdır ve genellikle V sembolü ile gösterilir. Bir değer fonksiyonu, bir ajanın belirli bir durumda olmasının veya bir eylemi gerçekleştirmenin ne kadar ne kadar iyi olduğunu gösterir. Birkaç değer fonksiyonu olabilir. Optimal değer fonksiyonu, diğer değer fonksiyonlarına kıyasla tüm durumlar için en yüksek değere sahip olan fonksiyondur. Benzer şekilde, optimal bir politika, optimal bir değer fonksiyonuna sahiptir.

PÖ'nün temel bileşenleri bir labirent oyunu ile özetlenmek istenirse ajan, labirentte dolaşan kişidir. Çevre, labirenttir. Durum, ajanın şu anda içinde bulunduğu labirentteki konumdur. Ajan, bir durumdan diğerine geçerek bir eylem gerçekleştirir. Ajan, eylemi herhangi bir engele takılmadığında olumlu bir ödül alır ve eylemi engellere takılıp hedefe ulaşamadığında olumsuz bir ödül alır. Amaç, labirentten çıkıp hedefe ulaşmaktır.

3.1.9. Model

Ajanın bir çevreyi temsil etmesidir ve edinilmiş çevresel bilgi topluluğu anlamına gelir. PÖ'de öğrenme, model tabanlı öğrenme ve modelsiz öğrenme olmak üzere iki türde olabilir. Model tabanlı ve modelsiz arasındaki fark, ajanın geçiş fonksiyonu ve ödül fonksiyonu gibi çevrenin modelini (veya dinamiklerini) öğrenip öğrenemeyeceğidir.

Model tabanlı öğrenmede, ajan bir görevi gerçekleştirmek için önceden öğrenilmiş bilgileri kullanırken modelsiz öğrenmede, ajan doğru eylemi gerçekleştirmek için sadece bir deneme yanılma deneyimine güvenir. Model tabanlı öğrenmede, bir görev için daha önce öğrenilen bir deneyim (harita) kullanılır, modelsiz öğrenmede ise önceki bir deneyim kullanılmaz ve tüm farklı yollar denenip daha hızlı olan seçilir.

3.2. Pekiştirmeli Öğrenmedeki Zorluklar

PÖ'de ele alınması gereken birçok zorluk mevcuttur [45, 151]. Bunlardan bazılarına aşağıda yer verilmiştir.

3.2.1. Keşif ve Sömürü İkilemi

PÖ'de, bir ajanın çevreyi gözlemleyerek ödülleri en üst düzeye çıkarması beklenir. PÖ ajanı, iyi bir ödül sağlayabilecek farklı eylemleri keşfedebilir veya iyi bir ödülle sonuçlanan önceki eylemi kullanabilir. Bu, kaçınılmaz bir ödünleşim olan keşif ve sömürü ikilemine yol açar. Keşif, sonuçlardan yararlanmak için bir dizi eylemde bulunmak anlamına gelir. Sömürü ise öğrenilmiş eylemde bulunmak anlamına gelir. Ajan farklı eylemleri araştırırsa, tüm eylemler optimumu olmayacağından, ajanın zayıf bir ödül alması olasılığı yüksektir. Ajan yalnızca bilinen optimum eylemi kullanırsa, daha iyi bir ödül sağlayabilecek optimum eylemi kaçırma olasılığı da yüksektir. Keşif ve sömürü arasında her zaman bir değiş tokuş vardır. Aynı anda hem keşif hem de sömürü yapılamaz. Epsilon-açgözlü (Epsilon-greedy) [159] ve üst güven sınırı kısaca UCB (Upper Confidence Bound) [160] eylem seçimi gibi yöntemler vardır.

3.2.2. Genelleme ve Boyutsallık Laneti

PÖ'de bir ajan, daha önce görülmemiş durumlar üzerinde hareket etmek için deneyimleri genelleştirebilmelidir. Bu sorun, tüm olasılıkları deneyimlemek pratik olmadığı için durum uzayı ve eylem uzayı yüksek boyutlu olduğunda ortaya çıkar. Ancak bu, fonksiyon tahmincileri ile çözülür.

3.2.3. Gecikmiş Ödüller

Daha önce bahsedildiği gibi bir PÖ çevresinde, ajana ne yapacağı veya nasıl yapacağı öğretilemez. Bunun yerine ajana yaptığı her eylem için bir ödül verilir. Ardından ajan, olumlu bir ödül almasını sağlayan eylemleri gerçekleştirmeye başlayacaktır. Dolayısıyla bir deneme yanılma sürecidir. Bazen bu süreçteki ödüller anında gerçekleşmez. Bunun yerine, uzun vadede geri dönecek gecikmiş ödüller olabilir. Her adımda bir ödül alınamayabilir. Ödül, bir görevin tamamlanmasından sonra verilebilir. Bazı durumlarda, herhangi bir hata yapılıp yapılmadığını öğrenmek için her adımda bir ödül alınır. Ajan, ödül veya cezanın sebebinin farkında olmalıdır. Bazen yanlış bir davranış daha sonra cezaya neden olabilir.

3.2.4. Kısmi Gözlemlenebilirlik

Kısmi gözlemlenebilirlik, anlık durumu anlamak için gerekli tüm gözlemlerin olmaması durumudur. Örneğin; bir sürücünün motor sıcaklığını veya dişlilerin dönüş hızını bilmesi gerekmeyebilir. Bu durumda sürücü araba kullanabiliyor olsa da, dikiz aynası veya yan aynalar olmadığında trafikte iyi araba kullanamayacaktır. Gerçek dünyada çoğu sistem kısmen gözlemlenebilir. Bu sorun, genellikle eylem seçimi için ajanların hafızasından gözlem geçmişi dahil edilerek çözülür.

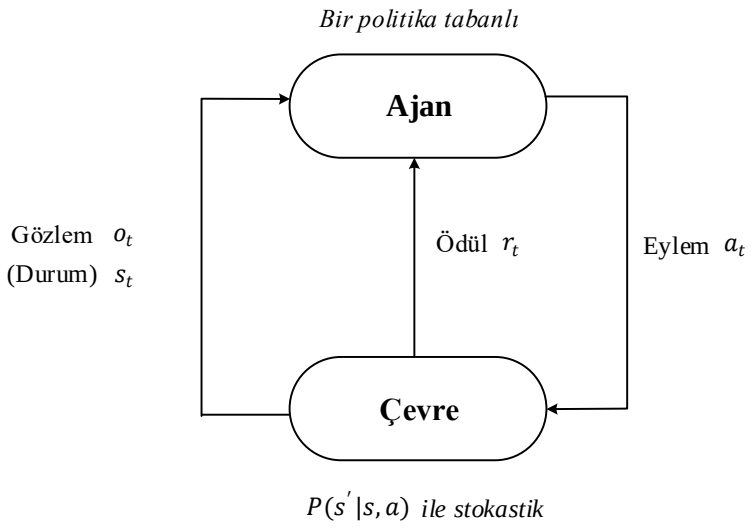
3.2.5. Ajan Güvenliği

Mekanik ajanlar, öğrenme sürecinde kendilerine ve çevrelerine zarar verebilir. Bu güvenlik sorunu hem keşif aşamasında hem de tam süreçte önemlidir. Ayrıca, bir ajanın politikasını analiz etmek zordur, yani görünmeyen bir durumda ajanın ne yaptığı belirsizdir. Çevre simülasyonu, ajanı güvenli bir şekilde eğitmek için iyi bir yoldur, ancak bu, gerçek çevreye göre yanlışlık nedeniyle eksik bir öğrenmeye neden olur.

3.3. Markov Zinciri ve Markov Karar Süreci

PÖ, ayrık zaman ayarında stokastik bir kontrol süreci olarak da düşünülebilir. t zamanında, ajan s_t durumuyla başlar ve o_t'yi gözlemler, daha sonra π politikasına göre bir a eyleminde bulunur ve t zamanında bir r_t ödülü alır. Dolayısıyla, eylemin bir sonucu olarak s_{t+1}'e bir durum geçişi meydana gelir ve ajan bir sonraki o_{t+1} gözlemini alır. Bu nedenle geçmiş, eylemlerin, gözlemlerin ve ödüllerin sıralı bir kümesidir.

Markov Karar Süreci kısaca MKS (Markov Decision Process–MDP) [161], Markov özelliği ile sıralı bir karar verme sürecidir. MKS'de Şekil 3.2'de gösterildiği gibi ajan ve çevre etkileşimi bulunmaktadır. MKS, PÖ problemini çözmek için matematiksel bir çerçeve sağlar. Hemen hemen tüm PÖ problemleri MKS olarak modellenebilmektedir.



Şekil 3.2. Markov Karar Süreci'nde ajan ve çevre etkileşimi

MKS'yi detaylandırmadan önce, MKS'nin temelini oluşturan Markov zinciri ve Markov süreci iyi anlaşılmalıdır. Markov özelliği, gelecekteki durumun koşullu olasılık dağılımının, tüm durum-eylem geçmişi yerine yalnızca anlık duruma ve eyleme bağlı olduğu anlamına gelir, bu nedenle hafızasız olarak kabul edilir. Markov özelliği, geleceğin yalnızca şimdiye bağlı olduğunu ve geçmişe bağlı olmadığını belirtir.

Markov zinciri, bir sonraki durumu tahmin etmek için yalnızca mevcut duruma bağlı olan ve önceki durumları değil, yani gelecek koşullu olarak geçmişten bağımsız olan olasılıklı bir modeldir. Markov zinciri kesinlikle Markov özelliğini takip eder. Örneğin; mevcut hava durumunun bulutlu olduğu biliniyorsa, bir sonraki durumun yağmurlu olabileceği tahmin edilebilir. Bir sonraki durumun, güneşli, rüzgarlı vb. olabilecek geçmiş durumları değil, yalnızca mevcut durumu (bulutlu) dikkate alarak yağmurlu olabileceği sonucuna varılır. Ancak, Markov özelliği tüm işlemler için geçerli değildir. Örneğin; bir zar atma işlemi düşünüldüğünde bir sonraki durum, atılan zarın mevcut durumuna bağlı değildir. Bir stokastik süreç, Markov özelliğini takip ediyorsa, Markov süreci olarak adlandırılır.

MKS, Markov zincirinin bir uzantısıdır. MKS'de sistem tamamen gözlemlenebilir, bu da durumların anlık gözlemlerden türetilebileceği anlamına gelir. Bu nedenle, ajan önceki zamanlarda ne olduğu yerine sadece anlık gözleme dayalı bir eyleme karar verebilir. MKS, beş önemli unsur ile temsil edilir:

1) **Durum uzayı** (S)**:** Ajanın içinde olabileceği bir dizi durum.
2) **Eylem uzayı** (A)**:** Bir durumdan diğerine geçmek için ajan tarafından gerçekleştirilen bir dizi eylem.
3) **Geçiş olasılığı fonksiyonu** $(P: S \times S \times A \to [0,1])$**:** Bir a eylemi gerçekleştirerek bir s durumundan bir sonraki s' durumuna geçme olasılığı olan geçiş olasılığı $P(s'|s,a)$.
4) **Ödül fonksiyonu** $(R: S \times A \to \mathbb{R})$**:** Bir s durumundan bir sonraki s' durumuna geçerken çevreden elde edilen ödüllerin bir fonksiyonu $R(s,a) = \mathbb{E}[r_t|s_t = s, a_t = a]$.
5) **İndirim faktörü** $(\gamma \in [0,1])$**:** Değer fonksiyonu için gelecekteki ödüllerin öneminin bir ölçüsü.

3.3.1. Ödül ve Getiri

Bir PÖ çevresinde, ajan bir eylem gerçekleştirerek çevre ile etkileşime girer ve bir durumdan diğerine geçer. Yaptığı eyleme göre bir ödül alır. Ödül, pozitif veya negatif bir sayısal değerdir. Bir ajan, anlık ödüller yerine çevreden aldığı toplam ödül miktarını (birikmiş ödüller) en üst düzeye çıkarmaya çalışır. Ajanın çevreden aldığı toplam ödül miktarına getiri denir. Bu nedenle, ajanlar tarafından alınan getiri miktarı (3.1)'deki gibi tanımlanabilir.

$$R_t = r_{t+1} + r_{t+2} + \ldots . + r_T \tag{3.1}$$

Burada; r_{t+1}, ajanın bir durumdan diğer bir duruma geçmek için a_0 eylemini gerçekleştirirken t_0 zaman adımında aldığı ödüldür. r_{t+2} ise bir sonraki zaman adımında ajan tarafından alınan ödüldür. Benzer şekilde, r_T de T son zaman adımında alınan ödüldür.

Epizodik görevler, bir sonlandırma (terminal) durumu olan görevlerdir. PÖ'de bölümler, başlangıç durumundan son durumlara kadar ajan-çevre etkileşimleri olarak kabul edilir. Örneğin; bir araba yarışı video oyununda, oyuna başlanır (başlangıç durumu) ve oyun bitene kadar oynanır (son durum). Buna bölüm denir. Oyun bittikten sonra oyun yeniden başlatılarak bir sonraki bölüme başlanır ve bir önceki oyunda bulunulan konumdan bağımsız olarak ilk durumdan başlanır. Yani her bölüm diğerinden bağımsızdır. Sürekli bir görevde ise bir son durum yoktur.

Bir ajanın hedefinin getiriyi en üst düzeye çıkarmak olduğu bahsedilmişti. Epizodik bir görev için getiri (3.1) ile tanımlanabilir. Sürekli yani herhangi bir son durumun olmadığı bir görev için getiri sonsuzluğa eşit olacaktır. Bu durumda getiriyi en yükseğe çıkarmak için indirim faktörü kavramı tanıtılmaktadır. Getiri, γ indirim faktörü ile (3.2)'de verildiği gibi yeniden tanımlanabilir.

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{3.2}$$

İndirim faktörü, gelecekteki ödüllere ve anlık ödüllere ne kadar önem verildiğine karar verir. İndirim faktörünün değeri, 0 ile 1 arasındadır. İndirim faktörünün 0 olması, anlık ödüllerin daha önemli olduğu anlamına gelirken, 1 olması ise gelecekteki ödüllerin anlık ödüllerden daha önemli olduğu anlamına gelir.

0 olan bir indirim faktörü, yalnızca anlık ödülleri göz önünde bulundurduğundan asla öğrenemez. Benzer şekilde, 1 olan bir indirim faktörü sonsuza kadar gelecekteki ödülü aramayı öğrenecektir, bu da sonsuzluğa yol açabilir. Bilimsel yazındaki çalışmalarda, indirim faktörünün optimal değerinin 0.2 ile 0.8 arasında olduğu belirtilmektedir.

3.3.2. Politika Fonksiyonu

Ajan, hedefe ulaşmak ve beklenen en yüksek ödülü toplamak için her bir durumda hangi eylemi gerçekleştireceğini söyleyen optimal bir politika öğrenmelidir. Getiri, gelecekteki ödüllere bağlı olduğundan ajanın politikasına da bağlıdır. Politika fonksiyonu, deterministik veya stokastik olabilir.

Deterministik politika, durumlardan eylemlere bir eşlemedir ($\mu: S \rightarrow A$). Stokastik politika ise durum-eylem çiftinden bir olasılık değerine eşlemedir ($\pi: S \times A \rightarrow [0, 1]$).

Optimal bir politika, optimal değer fonksiyonuna sahiptir. Değer fonksiyonu, belirli bir politikayı takip ederken her bir duruma veya durum-eylem çiftine beklenen toplam ödülü atar.

3.3.3. Durum Değer Fonksiyonu

Bir durum değer fonksiyonu, aynı zamanda bir değer fonksiyonu olarak da adlandırılır. Bir ajanın π politikasıyla belirli bir durumda olmasının ne kadar iyi olduğunu belirtir. Bir değer fonksiyonu genellikle $V(s)$ ile gösterilir. Bir politikayı izleyen bir durumun değerini ifade eder.

Bir durum değer fonksiyonu (3.3)'teki gibi tanımlanabilir.

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t | s_t = s] \tag{3.3}$$

(3.4), t zamanında s durumundan başlayarak ve ardından π politikasını izleyerek, t adımında G getirisinin beklenen değerini tanımlar.

$$V^{\pi}(s) = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|s_t = s] \tag{3.4}$$

3.3.4. Durum-Eylem Değer Fonksiyonu

Durum-eylem değer fonksiyonuna Q fonksiyonu da denir. Bir ajan için π politikasına sahip bir durumda belirli bir eylemi gerçekleştirmesinin ne kadar iyi olduğunu belirtir. Q fonksiyonu, $Q(s)$ ile gösterilir. Bir π politikasını izleyen bir durumda eylem gerçekleştirmenin değerini belirtir.

Q fonksiyonu (3.5)'teki eşitlik ile tanımlanabilir.

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | s_t = s, a_t = a] \tag{3.5}$$

(3.6), t zamanında a eylemiyle s durumundan başlayarak ve ardından π politikasını izleyerek, t adımında G getirisinin beklenen değerini tanımlar.

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|s_t = s, a_t = a] \tag{3.6}$$

Değer fonksiyonu ile Q fonksiyonu arasındaki fark, değer fonksiyonunun bir durumun iyiliğini belirtirken Q fonksiyonunun bir durumdaki bir eylemin iyiliğini belirtmesidir.

3.3.5. Bellman Denklemi ve Optimallik

Bir MKS'yi çözmek, optimal politikaları ve değer fonksiyonlarını elde etmek anlamına gelir. Optimal değer fonksiyonları ve optimal politika, adını Amerikalı matematikçi Richard Bellman'dan alan Bellman denklemleri [162] çözülerek türetilebilir.

Optimal değer fonksiyonu, davranışın politika tarafından kontrol edildiği yerde, (3.7) ile ifade edilen beklenen en yüksek getiriyi döndürmelidir.

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \tag{3.7}$$

(3.8) ile ifade edilen optimal durum-eylem değer fonksiyonu, her durum-eylem çifti için en yüksek beklenen getiri sağlamalıdır.

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{3.8}$$

π^* optimal politika, $Q^*(s,a)$ ile elde edilebilir. Stokastik politika için (3.9) ile tanımlanır.

$$\pi^*(a|s) = \begin{cases} 1, & eğer\ a = \arg\max_{a} Q^*(s,a) \\ 0, & aksi\ taktirde \end{cases} \tag{3.9}$$

Deterministik politika için ise (3.10) ile tanımlanır.

$$\mu^*(s) = a = \arg\max_{a} Q^*(s,a) \tag{3.10}$$

$V^*(s)$ optimal değer fonksiyonu, diğer tüm değer fonksiyonlarına kıyasla daha yüksek bir değere sahip olduğu için, Q fonksiyonunun en yükseği olacaktır. Böylece optimal değer fonksiyonu, Q fonksiyonunun en yükseği alınarak (3.11)'deki gibi kolayca hesaplanabilir.

$$V^*(s) = \max_{a} Q^*(s,a) \tag{3.11}$$

Durum değer fonksiyonu için Bellman denklemi (3.12) ile temsil edilebilir.

$$V^{\pi}(s) = \sum_{s'} P(s'|s,a)[R(s,a) + \gamma V^{\pi}(s')] \tag{3.12}$$

(3.12)'deki denklem bir π politikasını izleyen bir s durumunun değerinin nasıl bulunacağını belirtir.

Benzer şekilde, Q fonksiyonu için Bellman denklemi (3.13) ile temsil edilebilir.

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)\,[R(s'|s,a) + \gamma \sum_{a'} Q^{\pi}(s',a')] \tag{3.13}$$

Durum değer fonksiyonu ile aynı şekilde, (3.13) bir π politikasını izleyen bir durum-eylem çiftinin değerinin özyinelemeli olarak nasıl bulunacağını belirtir.

Optimal durum değer fonksiyonu için Bellman denklemi (3.14)'teki gibi ifade edilir.

$$V^*(s) = \max_{a} \sum_{s'} P(s'|s,a)\,[R(s'|s,a) + \gamma V^*(s')] \tag{3.14}$$

Optimal durum-eylem değer fonksiyonu için Bellman denklemi ise (3.15)'teki gibi ifade edilir.

$$Q^*(s) = \sum_{s'} P(s'|s,a)\left[R(s'|s,a) + \gamma \max_{a'} Q^*(s',a')\right] \tag{3.15}$$

3.4. Bellman Denklemlerinin Çözümü

Bellman denklemlerini çözmek için üç ana yaklaşım vardır: Dinamik Programlama (DP), Monte Carlo (MC) yöntemi ve Zamansal Fark (ZF) öğrenme [45].

PÖ uygulamalarında tam ve doğru bir çevre modeli bilindiğinde DP, çevre modeli bilinmediğinde ise tahmin istatistiklerine dayalı MC yöntemi veya adım adım artan bir şekilde ilerleyen ZF öğrenme kullanılır.

3.4.1. Dinamik Programlama

Dinamik Programlama kısaca DP (Dynamic Programming–DP) [162], özyinelemeli enyileme problemleri için genel bir çözüm yöntemidir. Model tabanlı bir yöntem olan DP'de, karmaşık problemleri birer birer çözmek yerine, problem basit alt problemlere bölünür, sonra her bir alt problem için çözüm hesaplanır ve saklanır. Aynı alt problem tekrar ortaya çıkarsa, yeniden hesaplamak yerine önceden hesaplanmış çözüm kullanılır. Böylece, DP, hesaplama süresinin büyük ölçüde en aza indirilmesini sağlar. DP'nin matematik ve bilgisayar bilimleri dahil olmak üzere çeşitli uygulamaları bulunmaktadır.

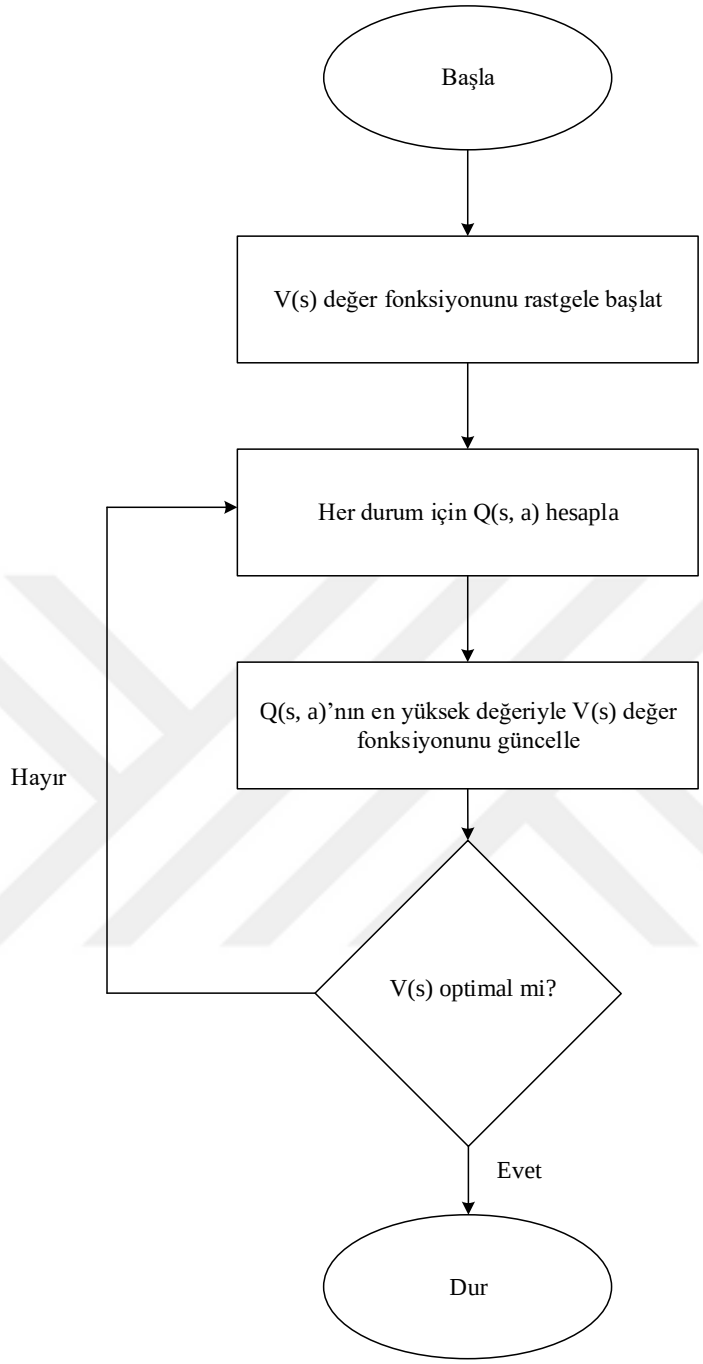
PÖ'yü çözerken, alt problemin çözümü bir politikanın değer fonksiyonudur. Bu nedenle, kontrol problemi optimal bir değer fonksiyonunu veya optimal bir politikayı bulmak amaçlanır. Bu amaçla DP'de, değer yineleme ve politika yineleme algoritmaları kullanılır.

Değer Yineleme

Değer yinelemede, rastgele bir değer fonksiyonuyla başlanır. Rastgele değer fonksiyonu optimal olmayabilir, bu yüzden optimal değer fonksiyonu bulunana kadar yinelemeli bir şekilde yeni bir geliştirilmiş değer fonksiyonu aranır. Optimum değer fonksiyonu bulunduktan sonra, bundan kolayca optimum politika türetilebilir. Değer yineleme algoritması, Şekil 3.3'te verilmiştir.

Değer yinelemesinde dört temel adım bulunmaktadır:

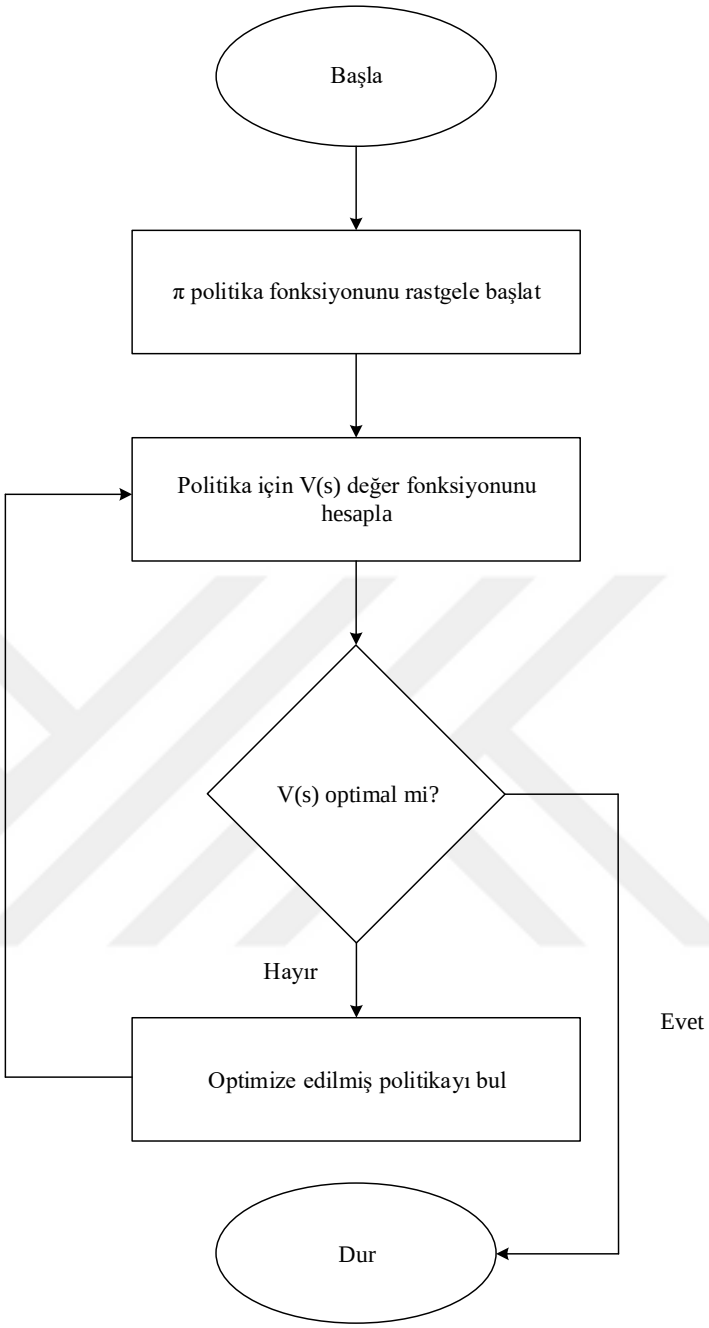
1) Rastgele değer fonksiyonu, yani her durum için rastgele değer başlatılır.
2) $Q(s,a)$'nın tüm durum-eylem çiftleri için Q fonksiyonu hesaplanır.
3) Değer fonksiyonu, $Q(s,a)$'dan en büyük değerle güncellenir.
4) Değer fonksiyonundaki değişiklik çok küçük olana kadar bu adımlar tekrarlanır.



Şekil 3.3. Değer yineleme algoritması

Politika Yineleme

Değer yinelemesinden farklı olarak, politika yinelemesinde rastgele politika ile başlanır. Politika yineleme, politika değerlendirmeyi ve politika iyileştirmeyi bütünleştirir. Politika değerlendirme, rastgele tahmin edilen bir politikanın değer fonksiyonunun değerlendirilmesidir. Politika iyileştirme ise değer fonksiyonu değerlendirildikten sonra, optimal değil ise yeni bir iyileştirilmiş politikanın bulunmasıdır. Politika yineleme algoritması, Şekil 3.4'te verilmiştir.



Şekil 3.4. Politika yineleme algoritması

Politika yinelemesinde dört temel adım bulunmaktadır:

1) Rastgele bir politika başlatılır.
2) Rastgele politika için değer fonksiyonu bulunur ve politika değerlendirmesi ile optimal olup olmama durumu kontrol edilir.
3) Optimal değil ise politika iyileştirmesi ile yeni bir iyileştirilmiş politika bulunur.
4) Optimal politika bulunana kadar bu adımlar tekrarlanır.

3.4.2. Monte Carlo

Monte Carlo kısaca MC yöntemi [163], PÖ'de çevrenin modeli bilinmediğinde optimal politikayı bulmak için kullanılan oldukça güçlü bir algoritmadır. Rastgele örnekleme yoluyla yaklaşık çözümleri bulmak için istatistiksel bir yöntemdir. MC yöntemi yalnızca epizodik görevlere uygulanabilir.

DP'deki değer yinelemesi ve politika yinelemesi, optimal politikayı bulmak için geçiş ve ödül olasılıkları gerektirir. Ancak geçiş ve ödül olasılıklar bilinmediğinde MKS, MC yöntemi kullanılarak çözülür. MC yöntemi, yalnızca örnek durum, eylem ve ödül dizileri gerektirir. MC yöntemi, değer fonksiyonunu tahmin etmek için kullanılan MC tahmini ve değer fonksiyonunu optimize etmek için kullanılan MC kontrolünden oluşur.

Bir değer fonksiyonun temel olarak, π politikasına sahip bir s durumundan beklenen getiri olduğu daha önce ifade edilmişti. MC yönteminde beklenen getiri yerine ortalama getiri kullanılır. Böylece MC tahmininde, beklenen getiri yerine ortalama getiriyi alarak değer fonksiyonuna yaklaşılır.

MC tahmini kullanılarak, verilen herhangi bir politikanın değer fonksiyonu tahmin edilebilir. MC tahmininde beş temel adım bulunmaktadır:

8) İlk olarak, değer fonksiyonunu rastgele bir değer ile başlatılır.
9) Ardından, getirileri saklamak için getiri adı verilen boş bir liste başlatılır.
10) Daha sonra bölümdeki her durum için getiri hesaplanır.
11) Ardından getiri, getiri listesine eklenir.
12) Son olarak, değer fonksiyonu olarak ortalama getiri alınır.

MC yönteminde ajan, örneklenen tüm bölümlerin ortalama getirisi olarak $V_\pi(s)$ değerini öğrenir. Her bölüm için, yaklaşık değer, (3.16)'ya göre artan bir ortalama kullanılarak güncellenir.

$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha[G_t - V^\pi(s_t)] \tag{3.16}$$

Burada; $V^\pi(s_t)$ belirli bir politika kullanılarak başlatılan tahmini durum değeri, α öğrenme oranı, G_t ise ajanın durum-eylem çiftini ilk ziyaret ettiği t zamanındaki getiridir.

Keşif ve sömürü ikileminden kaçınmak için, Epsilon-açgözlü (ϵ-açgözlü) politika kullanılır. Burada tüm eylemler sıfır olmayan bir ϵ olasılıkla denenir. ϵ olasılık ile farklı eylemler rastgele keşfedilir ve $1-\epsilon$ olasılık ile en yüksek değere sahip bir eylem seçilir. Böylece, ϵ olasılık ile her zaman optimum eylemi kullanmak yerine, rastgele farklı eylemler keşfedilir. Sonsuza kadar keşif yapmak istenmediği için ϵ değeri zamanla azaltılmalıdır.

Politikaya dayalı MC yönteminde sekiz temel adım bulunmaktadır:

1) Rastgele bir politika ve rastgele bir Q fonksiyonu başlatılır.
2) Getirileri saklamak için bir liste başlatılır.

3) Rastgele bir π politikası kullanarak bir bölüm oluşturulur.
4) Bölümde meydana gelen her durum-eylem çiftinin getirisi, listeye kaydedilir.
5) Getiri listesindeki getirilerin ortalaması alınır ve bu değer Q fonksiyonuna atanır.
6) s durumunda bir a eylemi seçme olasılığına ϵ olasılığı tarafından karar verilir.
7) Olasılık $1-\epsilon$ ise en yüksek Q değerine sahip eylem alınır.
8) Olasılık ϵ ise farklı eylemler araştırılır.

3.4.3. Zamansal Fark Öğrenme

MC yönteminde tam bir bölümden sonra değer güncellenir, bu da onu önemli varyansa sahip yavaş bir algoritma haline getirir. Zamansal Fark kısaca ZF (Temporal Difference–TD) öğrenme yaklaşımı [164], değeri her adımda güncelleyerek bu dezavantajı çözer ve değeri güncellemek için tam bir bölüm gerektirmez. MC yöntemi gibi model dinamiklerinin önceden bilinmesini gerektirmez.

ZF öğrenme yöntemleri, değer fonksiyonunun tahmini için, önyükleme olarak da adlandırılan önceden öğrenilmiş tahmine dayalı olarak mevcut tahmine yaklaşır. Birikmiş ödül olan getiriyi bilmeden önce, geleceği tahmin eder. ZF öğrenme, DP ve MC yöntemleri arasında bir ara formu olup iki yöntemin de faydalarını kullanır. Hem ZF hem de DP, tahmin için önyükleme kullanır.

MC tahmininde yapıldığı gibi, ZF tahmininde de durum değerleri tahmin edilmeye çalışılır. Daha önce bahsedildiği gibi MC tahmininde, sadece ortalama getiri alınarak değer fonksiyonu tahmin edilmekteydi. Ancak ZF'de, önceki bir durumun değeri mevcut duruma göre güncellenir. Bir durumun değerini güncellemek için ZF güncelleme kuralı adı verilen bir yöntem kullanılır.

$t+1$ zamanında ZF öğrenme, değeri bir ZF hedefi ile karşılaştırır ve ZF hatasıyla günceller. En basit ZF öğrenme yöntemi olan tek adımlı ZF öğrenme ($TD(0)$), (3.17)'deki eşitlik ile ifade edilebilir.

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha[r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)] \tag{3.17}$$

Burada; γ indirim faktörü, $V^{\pi}(s_{t+1})$ ve r_{t+1} ise sırasıyla $t+1$ zamanındaki tahmini durum değeri ve ödüldür. $r_{t+1} + \gamma V^{\pi}(s_{t+1})$ ve $r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$ ise sırasıyla ZF hedefi ve ZF hatası olarak bilinir.

Algoritma 3.1'de tek adımlı bir ZF fonksiyon tahmin algoritmasının sözde kodu gösterilmektedir. Bu tür algoritmalar, parametrelerinin değerini ayarlayarak bir fonksiyona yaklaşmayı sağlar.

Algoritma 3.1. Tek adımlı bir ZF fonksiyon tahmini

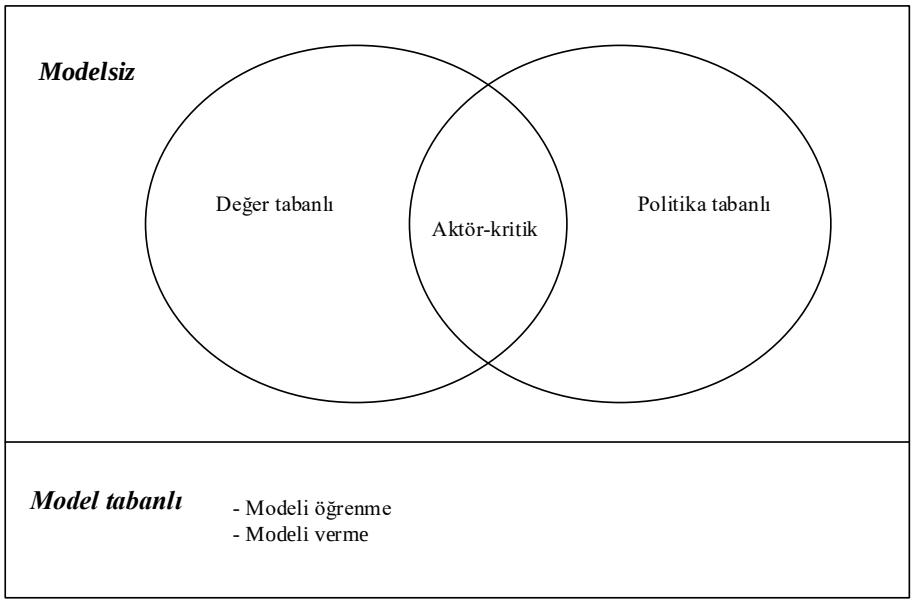
Değer fonksiyonunun parametrelerini (θ) keyfi olarak başlat
repeat {her bölüm için}
  s'yi başlat
  repeat {bölümdeki her bir adım için}
    Q'dan türetilen bir politika kullanarak s'de bir a eylemi seç (örn. ϵ-açgözlü)
    a eylemini gerçekleştir, r ve s''yi gözlemle
    $\theta \leftarrow \theta + \alpha[r + \gamma Q_\theta(s', a') - Q_\theta(s, a)]\nabla Q_\theta(s, a)$
    $s \leftarrow s'$
  until s çıkış
until yakınsama

ZF tahmininde değer fonksiyonu tahmin edilirken, ZF kontrolünde ise değer fonksiyonu optimize edilir. ZF kontrolü için yaygın olarak iki tür kontrol algoritması kullanılır:

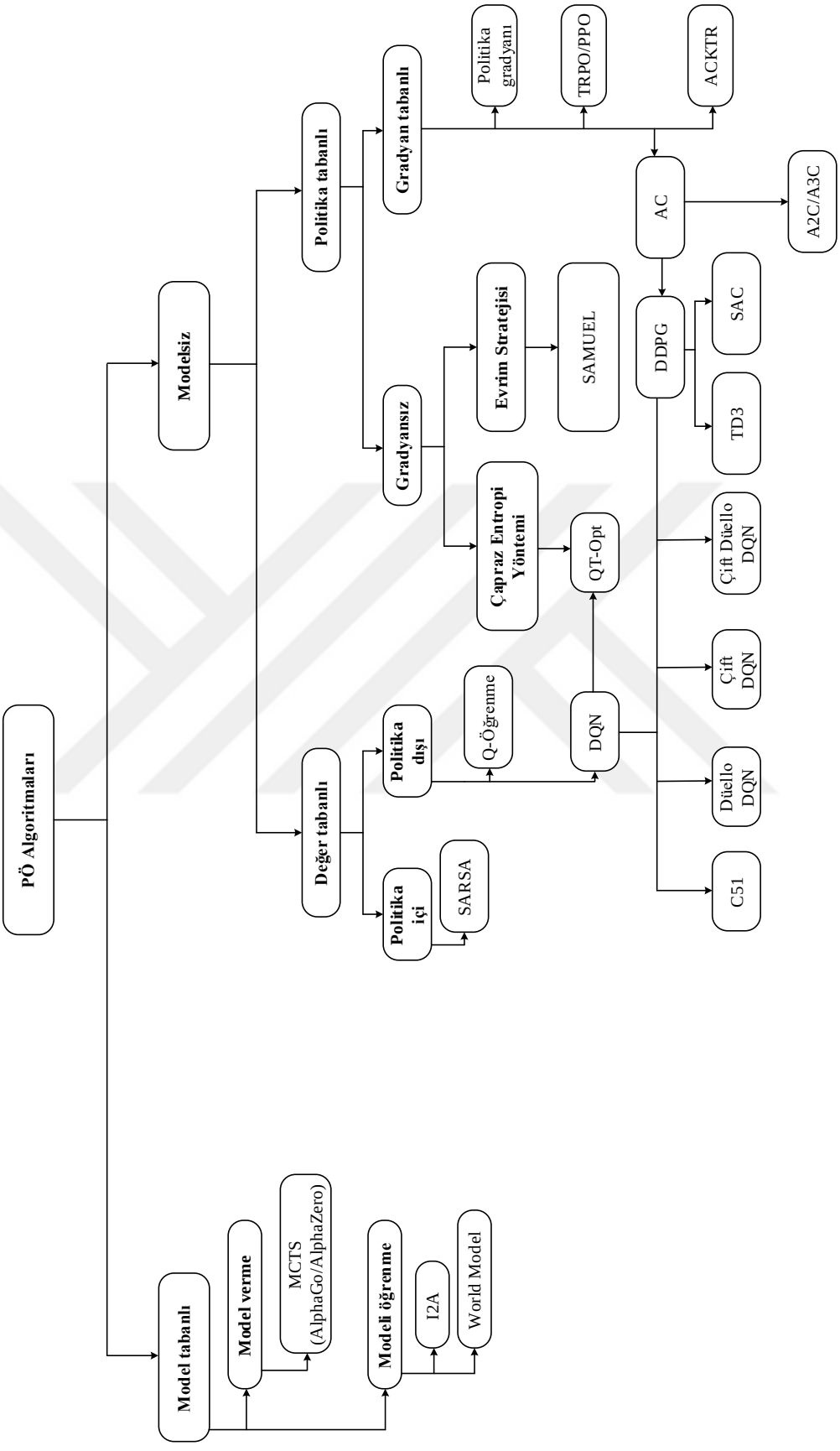
- Politika dışı öğrenme algoritması olan Q-öğrenme,
- Politika içi öğrenme algoritması olan SARSA (State-Action-Reward-State-Action).

3.5. Pekiştirmeli Öğrenme Yöntemleri

Şekil 3.5 ve Şekil 3.6'da sınıflandırıldığı gibi PÖ yöntemleri, modelsiz veya model tabanlı, değer tabanlı ve/veya politika tabanlı olabilir.



Şekil 3.5. Pekiştirmeli öğrenme yöntemleri



Şekil 3.6. Pekiştirmeli öğrenme algoritmalarının sınıflandırılması

3.5.1. Model Tabanlı Öğrenme

Model tabanlı öğrenmede, model kullanılarak bir plan oluşturulur ve ajan, daha önceki deneyimlerden elde ettiği bilgileri kullanır.

Şekil 3.5 ve Şekil 3.6'da görülebileceği gibi model tabanlı yöntemler, belirli bir modelle çalışan yöntemler ve modeli öğrenen yöntemler olarak iki kategoriye ayrılabilir.

Belirli bir modelle çalışan yöntemler için, ödül fonksiyonu ve geçiş süreci modellerine doğrudan ajan tarafından erişilebilir. Örneğin; AlphaGo algoritmasında [165], Go oyununun bilgisayar dili ile kolayca tanımlanabilen kuralları belirtilir. Go'daki geçiş fonksiyonu ve ödül fonksiyonunun tümü, ajanın politikasını değerlendirmesi ve iyileştirmesi için bilinir.

Modeli öğrenen yöntemler ise çevrenin karmaşıklığı nedeniyle modeli doğrudan elde edemez. Ancak ajan, önce çevre ile etkileşimlerden bir model öğrenir ve ardından modeli politika iyileştirmede uygulayabilir. Bu yöntemler, Dünya Modeli (World Model) algoritmasını [166], I2A algoritmasını [167] vb. içerir.

Model tabanlı yöntemlerin en önemli avantajı, gelecekteki durumların ve ödüllerin, ajanın daha iyi planlama yapmasına yardımcı olan çevre modeli aracılığıyla önceden tahmin edilebilmesidir. Bazı tipik yöntemler, saf planlama ve uzman yinelemeyi içerir [1]. Örneğin; MBMF (Model-Based Priors for Model-Free) algoritması [45] saf planlama yöntemlerini, AlphaGo algoritması uzman yinelemeyi benimser. Modele dayalı yöntemlerin dezavantajı ise modelin genellikle mevcut olmaması ve çevre dinamiklerinin karmaşık olmasıdır. Dahası, çevrenin açıkça temsil edilememesi gerçeğinde yatmaktadır. Ayrıca, öğrenilen modeller uygulamada genellikle hatalıdır ve bu da tahmin için yanlılığa neden olur. Biaslı bir modele dayalı olarak tahmin edilen ve geliştirilen politika, genellikle gerçek çevrede uygulandığında çöker.

3.5.2. Modelsiz Öğrenme

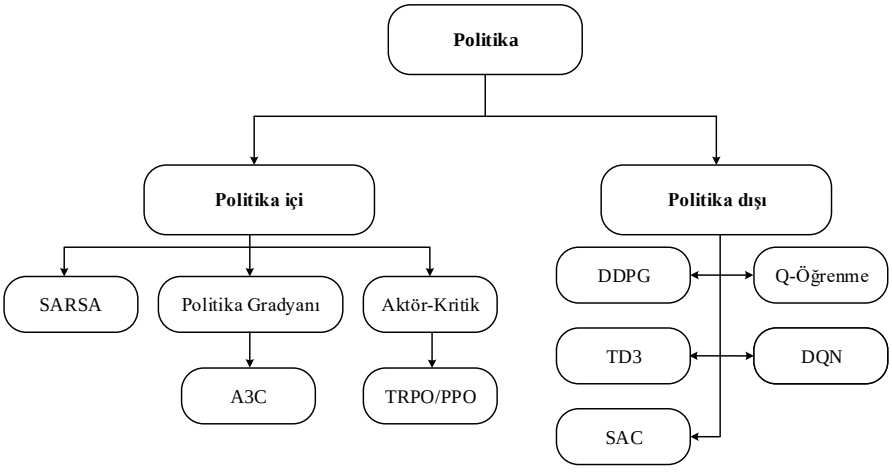
Modelsiz yöntemler, çevrenin bir modelini oluşturmaya çalışmaz. Ajan, çevre ile doğrudan etkileşime girer ve keşfedilen örneklere dayalı olarak performansını artırır. Model tabanlı yöntemlerle karşılaştırıldığında, modelden bağımsız yöntemlerin uygulanması kolaydır. Bununla birlikte, modelsiz yöntemler de kendi sorunlarından olumsuz etkilenmektedir. Bazen gerçek dünya ortamında keşif maliyeti, zaman tüketimi, ekipmanın aşınması ve güvenlik riskleri açısından son derece yüksek olabilir. Örneğin; otopilot durumunda, herhangi bir trafik kazasının katlanılması çok fazla bir maliyeti olacağından, bir ajanı herhangi bir ek önlem almadan modelsiz bir yöntemle gerçek dünyayı keşfetmesi için eğitilemez.

Modelsiz PÖ'de üç ana kategori vardır:

1) **Değer tabanlı öğrenme:** Değer fonksiyonları öğrenilir. Optimal politika, değer fonksiyonundan doğal olarak ortaya çıkar. "argmax" işlemi kullanıldığından eylem uzayının ayrık olduğu problemler için uygun bir öğrenmedir.
2) **Politika tabanlı öğrenme:** Politika doğrudan öğrenilir. Bir değer fonksiyonu öğrenilmek yerine getiri değerleri kullanılır. Değer tabanlı öğrenme yöntemlerinden farklı olarak sürekli eylem uzayının bulunduğu çevreler için uygundur.
3) **Aktör-Kritik öğrenme:** Hem politika (aktör) hem de değer (kritik) fonksiyonları aynı anda öğrenilir. Bu sebeple, sürekli eylem alanları için de uygundur.

3.5.3. Politika İçi ve Politika Dışı Kavramı

Politika içi ve politika dışı arasındaki fark, politikanın öngörüsüdür. Politika içi yöntemler, karar vermek için kullanılan politikayı değerlendirmeye veya iyileştirmeye çalışırken politika dışı yöntemler, verileri oluşturmak için kullanılandan farklı bir politikayı değerlendirir veya geliştirir. Politika içi yöntem, ajanın kendisinin çevreyle etkileşime girmesini gerektirir. Çevre ile etkileşime giren politika ile iyileştirilecek politika aynı olmalıdır. Politika dışı yöntemin buna uyması gerekmez, çevreyle etkileşime giren diğer ajanların deneyimleri de politikayı geliştirmek için kullanılabilir. Ortak politika yöntemi, mevcut politikaya göre bir eylem seçen ve eylemi yürüten, ardından mevcut politikayı güncellemek için verileri kullanan SARSA'dır. Dolayısıyla çevre ile etkileşime giren politika ile güncellenen politika aynıdır. Q-öğrenme ise tipik bir politika dışı yöntemdir. Eylemleri seçerken en yüksek işlemi ve ϵ-açgözlü bir politikayı benimser, bu da çevre ile etkileşime giren politika ile güncellenen politikayı aynı politika yapmaz. Şekil 3.7'de politika tabanlı olan politika içi ve politika dışı yöntemlere örnek verilmiştir.



Şekil 3.7. Politika içi ve politika dışı yöntemler

3.5.4. Q-Öğrenme

Bölüm 3.3'te bahsedildiği gibi Q değeri, π politikasını izleyerek s durumunda a eylemi gerçekleştirildiğinde beklenen ödül olarak tanımlanır. Böylece Q değeri, hangi eylemin diğerlerinden daha iyi olduğunu ölçebilir. Böylece daha iyi bir politika bulunabilir. Bu yönteme Q-öğrenme [168] denir. Uygulama kolaylığından dolayı en popüler PÖ algoritması denebilir.

Q-öğrenme çok basit ve yaygın olarak kullanılan bir ZF algoritmasıdır. Değer tabanlı bir öğrenme türüdür ve Bellman denklemi kullanılarak Q fonksiyonunun optimizasyonuna dayanmaktadır. Q-öğrenmede, durum-eylem değer çifti ile ilgilenilir.

$Q(s_t, a_t)$ eylem-değer fonksiyonu, her t adımında ve s_t durumunda alınan her eylem için tanımlanır. s_t durumunda a_t eylemini gerçekleştirdikten sonra, anında bir r_{t+1} ödül alınır ve ortaya çıkan bir sonraki durum s_{t+1} gözlemlenir. Bir sonraki durumdaki ödülün $P(r_{t+1}|s_t, a_t)$ olasılık dağılımlarından ve sonraki durumun $P(s_{t+1}|s_t, a_t)$'den örneklendiği deterministik olmayan bir durum varsayılır. Bu adımda, Q eylem-değer fonksiyonu üzerinden bir güncelleme (3.18) ile tanımlanır.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \max_{a_{t+1}} (s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right) \tag{3.18}$$

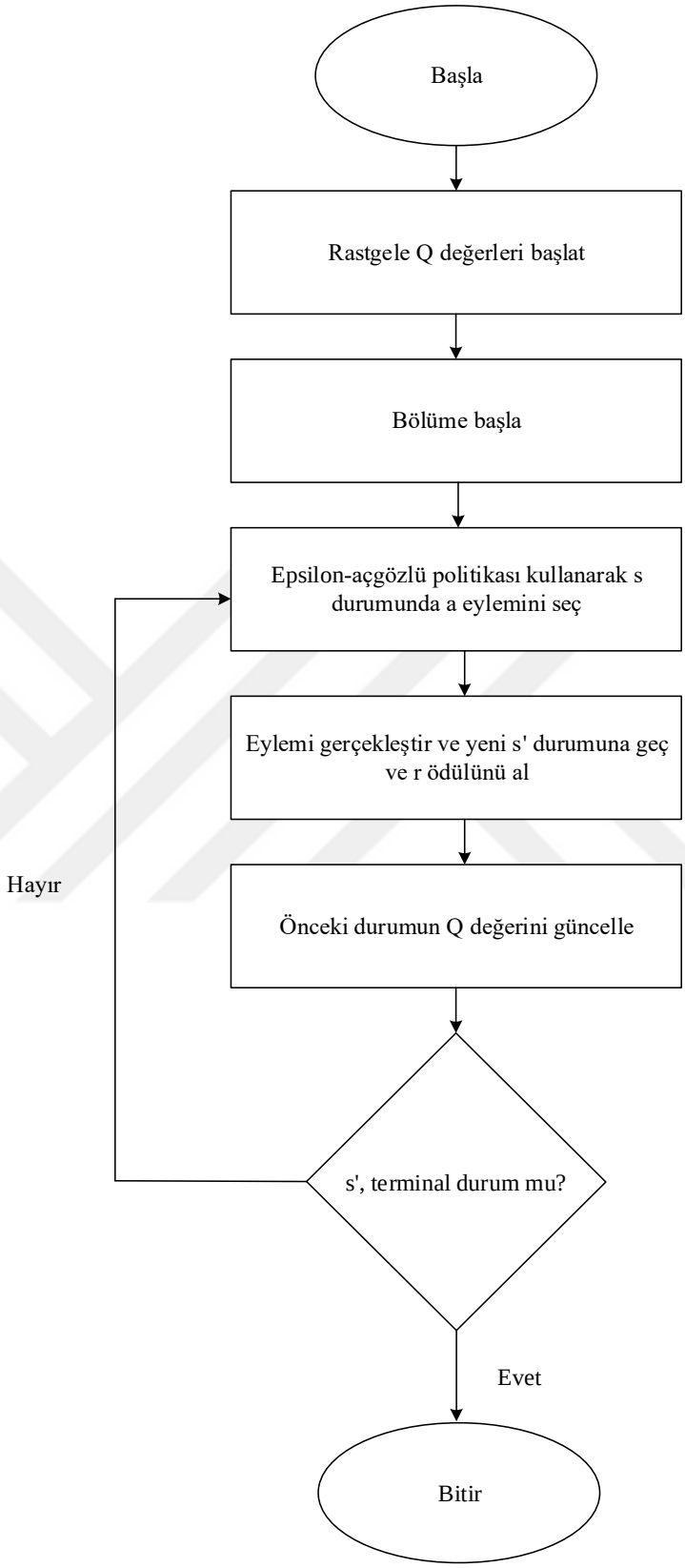
Bir sonraki eylemin değerini γ ile çarpılıp indirim yapılarak, ödüle yakın olan durumların eylem değerleri, uzak olanlardan daha büyük bir etkiye sahip olacaktır. Bu nedenle, algoritma uzun bir eylem dizisi ile bir kısa eylem dizisi arasında ayrım yapabilir, böylece ajanı ödüle daha hızlı ulaştıracak eylemleri tahmin etme yeteneği kazanır. Çevredeki tüm olası eylem-durum çiftlerinin Q değerlerini tahmin etmek için ajanın onu keşfetmesi gerekir. Keşif, $P(r_{t+1}|s_t, a_t)$ ve $P(s_{t+1}|s_t, a_t)$'den örnekler alınmasını sağlar. Keşfetmenin yolu, biraz rastgelelik içeren bir politika kullanarak eylemleri seçmektir. ϵ-açgözlü politikası bunu, zamanla azalan bir ϵ değeri tutarak yapar. (3.18), küçük bir farkla ZF tahmin güncelleme kuralına benzer. Şekil 3.8'de iş akışı verilen Q-öğrenmede temel olarak dört adım mevcuttur:

1) Q fonksiyonu bazı rastgele (keyfi) değerlerle başlatılır.
2) Epsilon-açgözlü politikası $\epsilon > 0$ kullanılarak bir durumdan bir eylem alınır ve o yeni duruma taşınır.
3) Güncelleme kuralına uyularak bir önceki durumun Q değeri aşağıdaki eşitlik ile güncellenir:

$$Q(s, a) = Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right)$$

Burada; a' Epsilon-açgözlü politikası ile seçilen eylemdir.

4) Son duruma ulaşana kadar 2. ve 3. adımlar tekrarlanır.



Şekil 3.8. Q-öğrenme algoritması

ZF veya MC yöntemleri artık önceki aktörün yerini alacak daha iyi bir p' politikası bulmaya katkıda bulunacaktır. Sonunda, optimal bir aktöre ulaşılacaktır. Q-öğrenme, durumlar (veya gözlemler) olarak satırlar ve eylemler olarak sütunlar içeren bir tablo kullanır. Tablodaki her hücre, bu eylemi gerçekleştiren bu durumun Q değeridir.

Q-öğrenme algoritmasında bir bölüm, bir çıkış durumuna (sonlandırma koşulu) ulaşana kadar ajan tarafından takip edilen bir durum-eylem çiftleri dizisidir. ϵ-açgözlü, normalde optimum Q değerine sahip eylemin seçileceği, ancak bazen küçük bir olasılığa sahip rastgele bir eylemin seçileceği anlamına gelir. Algoritma 3.2'de Q-öğrenme algoritmasının sözde kodu verilmiştir.

Algoritma 3.2. Q-öğrenme

$Q(s,a)$'yı keyfi olarak başlat
repeat {her bölüm için}
  s'yi başlat
  repeat {bölümdeki her bir adım için}
    Q'dan türetilen bir politika kullanarak s'de bir a eylemi seç (örn. $\epsilon - greedy$)
    a eylemini gerçekleştir, r ve s''yi gözlemle
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$
  until s çıkış
until yakınsama

Q-öğrenme algoritması, politika dışı, modelsiz ve değer tabanlı olmak üzere üç ana yöntemden oluşur [168]. Modelsiz yöntem iki ana yaklaşıma ayrılır. Bunlardan biri, ajanın bir değer fonksiyonunu en yükseğe çıkarmaya ve optimal politikayı anlamaya çalıştığı değer fonksiyonu tabanlı yöntemdir. Diğer bir yöntem olan politika arama, ajan politika parametreleri arasında arama yaparak optimal politikayı belirler [45]. Politika dışı, ajanın s' durumundaki ϵ-açgözlü politikasını kullanarak optimum a' eylemi yapması ve getiriyi en yükseğe çıkarmaya çalışması anlamına gelir [45]. Ödül fonksiyonlarının değerleri, ilk sıfır değerindeki eylem sayısına göre durum sayısı boyutları ile matris üzerine yazılır [168]. Her durum eyleminin Q değeri, Q tablosuna kaydedilir.

3.5.5. SARSA

Durum-Eylem-Ödül-Durum-Eylem olarak bilinen SARSA (State-Action-Reward-State-Action) [169], bir politika içi ZF kontrol algoritmasıdır. Q-öğrenmedeki gibi, burada da durum-değer çifti yerine durum-eylem değerine odaklanılır. SARSA, Q değerlerini güncellemek için bir sonraki durum için maksimum ödül yerine aynı politikayı kullanarak yeni bir eylem seçer.

SARSA'da Q değeri, (3.19)'daki güncelleme kuralına göre güncellenir.

$$Q(s,a) = Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big) \tag{3.19}$$

(3.19)'da, Q-öğrenmede olduğu gibi $\max_{a'} Q(s',a')$ yoktur. Burada basitçe $Q(s',a')$'dır. Şekil 3.9'da iş akışı verilen SARSA yöntemi yer alan adımlar aşağıdaki gibidir:

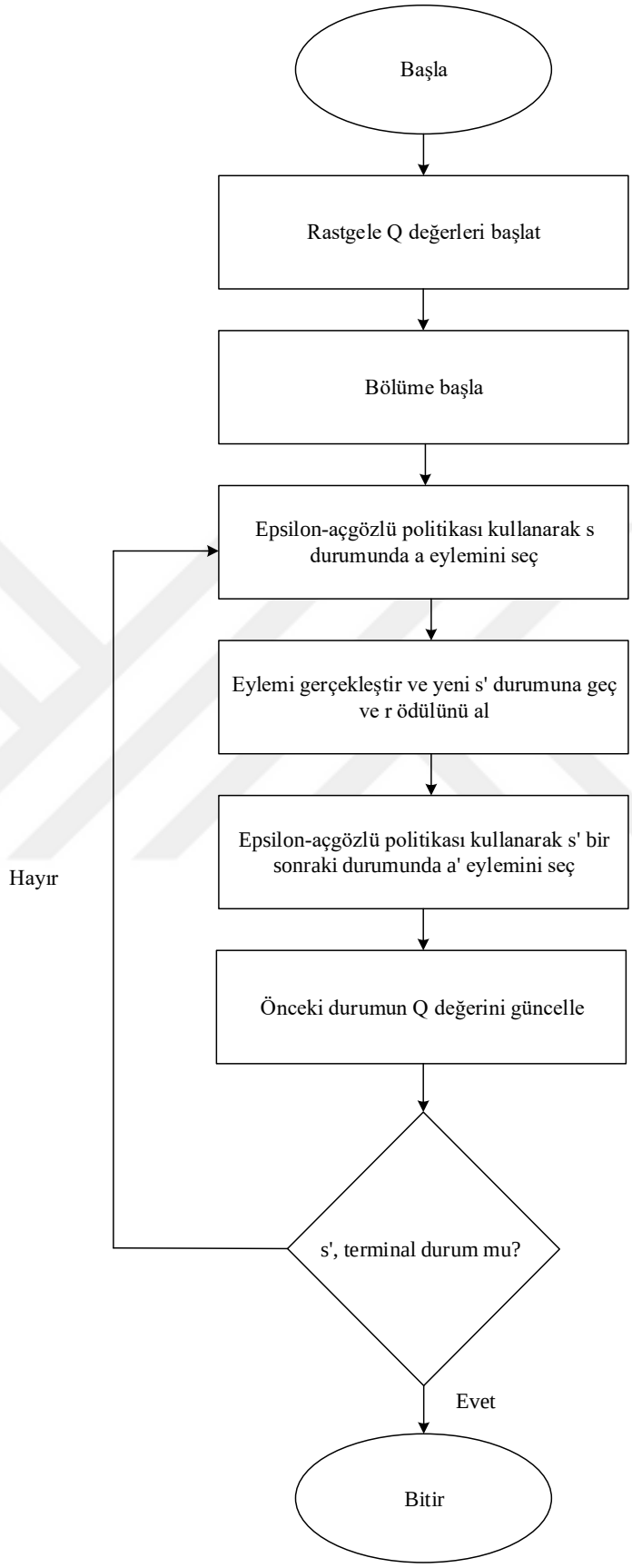
1) Q fonksiyonu bazı rastgele (keyfi) değerlere başlatılır.
2) Epsilon-açgözlü politikası $\epsilon > 0$ kullanılarak bir eylem seçilir ve bir durumdan diğerine geçilir.
3) Güncelleme kuralına uyularak bir önceki durumun Q değeri aşağıdaki eşitlik ile güncellenir:

$$Q(s,a) = Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$$

Burada; a' Epsilon-açgözlü politikası ile seçilen eylemdir.

Politika, SARSA'da deneyim oluşturma ve politika geliştirme olmak üzere iki rol oynamaktadır. Temel olarak, davranışı oluşturmak için kullanılan politikaya davranış politikası, değerlendirilen ve geliştirilen politikaya ise hedef politika denir. SARSA gibi davranış politikası ile hedef politikasının aynı olduğu algoritma, politika içi yöntem olarak bilinir.

Q-öğrenme, eylem değerlerini güncellerken bir sonraki optimum eylemin değerini kullanan, politika dışı bir yöntem iken SARSA ve aktör-kritik gibi politikaya dayalı yöntemler, bir sonraki eylemi de seçmek için bir ϵ-açgözlü politikasını kullanır. Politikaya içi yöntemler, bir tür deneme yanılma sürecidir, ancak iyileştirme için yalnızca mevcut politikanın ürettiği deneyimler kullanılır.



Şekil 3.9. SARSA algoritması

3.6. Pekiştirmeli Öğrenme Platformları

PÖ platformları, bir çevrede PÖ algoritmalarını simülasyon yapmak, oluşturmak, sahnelemek ve denemek için kullanılır. Birçok farklı PÖ platformu mevcuttur.

DeepMind Lab [170], ajan tabanlı yapay zeka araştırmaları için özel olarak tasarlanmış üç boyutlu özelleştirilebilir oyun benzeri bir platformdur. Birkaç PÖ algoritmasını çalıştırmak için bir laboratuvar görevi gören zengin bir simülasyonu yapılmış çevre sağlar. DeepMind Lab, PÖ ajanlarının büyük, kısmen gözlemlenmiş ve görsel olarak çeşitli dünyalarda karmaşık görevleri nasıl öğrenebileceğini incelemek için kullanılabilir. Ayrıca, özgün yapay zeka tasarımlarının keşfedilmesini ve hızlı bir şekilde yinelenmesini sağlayan basit ve esnek bir kullanıcı arayüzüne sahiptir. Son derece özelleştirilebilir ve genişletilebilirdir.

RL-Glue [171], farklı programlama dillerinde yazılmış olsalar bile ajanları, çevreleri ve programları birbirine bağlamak için bir arabirim sağlar. RL-Glue'nin arayüzü oldukça sade olup çeşitli diller ve bilgi işlem platformlarıyla çalışır.

Project Malmo [172], Microsoft'un Minecraft'ın üzerine inşa ettiği başka bir yapay zeka deney platformudur. Çevreyi özelleştirmek için iyi bir esneklik sağlar. Ancak Malmo, Open AI Universe'den farklı olarak şu anda yalnızca Minecraft oyun çevreleri sağlamaktadır.

ViZDoom [173], adından da anlaşılacağı gibi, Doom tabanlı bir yapay zeka platformudur. Ajanı test etmek için çoklu ajanlar ve rekabetçi bir çevre için destek sağlar. Ancak, ViZDoom yalnızca Doom oyun ortamını desteklemektedir.

Tez çalışması kapsamında OpenAI Gym [48] kütüphanesi kullanılmıştır. OpenAI [48], Elon Musk ve Sam Altman tarafından kurulan, açık kaynaklı bir yapay zeka araştırma şirketidir. En iyi endüstri liderleri ve birinci sınıf şirketler tarafından desteklenirler. OpenAI, araştırmacılara gerçekçi çevreleri simülasyon yapabileceği, PÖ algoritmaları oluşturulabileceği ve ajanları bu çevrelerde test edebileceği Gym platformunu sağlar.

OpenAI Gym [48], PÖ algoritmalarını oluşturmak, değerlendirmek ve karşılaştırmak için bir araç takımıdır. Basit ve anlaşılması kolaydır. Ajanın yapısı hakkında hiçbir varsayımda bulunmaz ve tüm PÖ görevlerine bir arayüz sağlar.

4. DERİN PEKİŞTİRMELİ ÖĞRENME

PÖ'nün geçmişte birçok başarısı olmasına rağmen, önceki yaklaşımlar ölçeklenebilirlikten yoksundur ve doğası gereği oldukça düşük boyutlu problemlerle sınırlıdır. PÖ algoritmaları da diğer algoritmalarla bellek karmaşıklığı, hesaplama karmaşıklığı ve örnek karmaşıklığı gibi aynı karmaşıklık sorunlarını paylaşır. Son yıllarda Derin Sinir Ağ'larının (DSA'ların) güçlü fonksiyon yaklaşımı ve temsil öğrenme özelliklerine dayanan derin öğrenmenin yükselişi, bu sorunların üstesinden gelmek için yeni araçlar sağlamıştır.

Derin öğrenmenin ortaya çıkışı, makine öğrenimindeki birçok alanda önemli bir etkiye sahip olmuştur. Derin öğrenmenin en önemli özelliği, DSA'ların görüntü, metin ve ses gibi yüksek boyutlu verilerin düşük boyutlu özniteliklerini otomatik olarak bulabilmesidir.

Derin Pekiştirmeli Öğrenme (DPÖ), PÖ'nün derin öğrenmeyle birlikte kullanılmasından oluşmaktadır. PÖ ve derin Yapay Sinir Ağ'larını (YSA'larını) birleştirmenin temel motivasyonlarından biri, PÖ'de bulunan büyük boyuttaki durumlar uzayını derin YSA'lara ihtiva edebilme potansiyelidir.

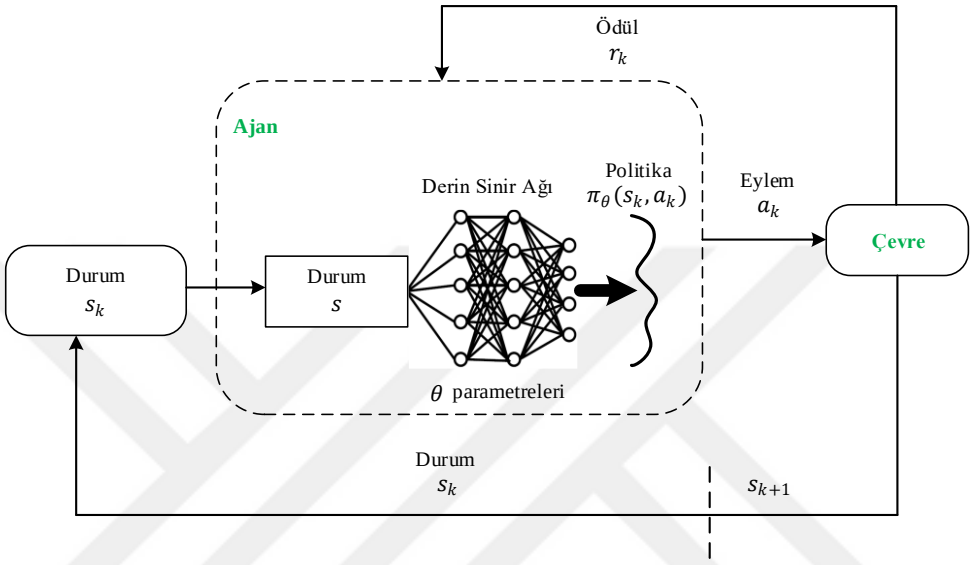
Derin öğrenme modeli için optimizasyon algoritmaları ve kayıp fonksiyonlarının seçimi, optimum ve daha hızlı sonuçların üretilmesinde büyük rol oynar. Çoğu öğrenme ağında hata, gerçek çıktı ile tahmin edilen çıktı arasındaki fark olarak hesaplanır. Bu hatayı hesaplamak için kullanılan fonksiyon, kayıp fonksiyonu olarak bilinir. Doğru tahminler için hesaplanan hatanın en aza indirilmesi gerekir. Bir sinir ağında bu, geriye yayılım kullanılarak yapılır. Mevcut hata, tipik olarak, ağırlıkları ve sapmayı, hata en aza indirilecek şekilde değiştirmek için kullanıldığı bir önceki katmana geriye doğru yayılır. Ağırlıklar, optimizasyon fonksiyonu kullanılarak değiştirilir. Bu nedenle, kayıp fonksiyonları bir sinir ağını eğitmeye yardımcı olur.

Derin öğrenme, çok sayıda etiketli veri gerektirdiğinden ve PÖ'de seyrek ve gecikmiş ödüller olduğundan çeşitli zorluklar vardır. Ek olarak derin öğrenme, verilerin bağımsız ve özdeş olarak dağıtıldığını varsayar. Ancak PÖ'de veri dağılımı öğrenme süreci sırasında zamanla değişir. Derin öğrenme teknikleri, mini yığınlar kullanıldığında daha etkili olduklarını göstermiştir; ancak, PÖ verileri sıralıdır ve ardışık örnekler arasında belirli bağımlılıklar vardır.

Tez çalışmamızdaki gibi durum-eylem alanının çok büyük olduğu ve tümünü depolamanın mümkün olmadığı çevreler olduğunda derin öğrenme devreye girer. DPÖ'de, sahip olunan girişe (durum, eylem) dayalı olarak Q değerini (durum-eylem çiftleri üzerinden değer) tahmin eden bir sinir ağıdır. DPÖ, değer fonksiyonunu, politika ve ajanı eğitmek için kullanılan modeli temsil etmek için sinir ağlarını kullanır.

Bir DPÖ ajanının bir çevre ile etkileşime girdiği Şekil 4.1'deki genel yapıda ajan, adım adım ağ ortamı ile etkileşim yoluyla eğitilir. Her k adımında, ajan, parametreleştirilmiş sinir ağının girişi olarak bir s_k durumunu gözlemler ve çıkış $\pi_\theta(s_k, a_k)$ politikası, s_k durumunda a_k eylemi

gerçekleştirme olasılığı olarak tanımlanır. Ajan, çevre üzerinde eylem gerçekleştirmek ve çevrenin durumunu değiştirmek için $\pi_\theta(s_k, a_k)$ stratejisine göre a_k eylemini seçer. Çevre durumu s_{k+1}'e dönüştürülür ve ajan, a_k eyleminin niteliğini değerlendirmek için geri bildirim olarak anında bir r_k ödülü alır.



Şekil 4.1. Derin pekiştirmeli öğrenmenin genel yapısı

DPÖ, Tablo 4.1'de özetlendiği gibi birçok alanda yaygın olarak kullanılmaktadır. Genellikle, DPÖ'nün yanı sıra bir uygulamada farklı yöntemler kullanılabilir. Örneğin; AlphaGo [165] gözetimli öğrenme ve pekiştirmeli öğrenme kullanılarak eğitilir.

Tablo 4.1. Çeşitli DPÖ uygulamaları [174]

Alan	**Uygulamalar**
Sağlık hizmeti	Teşhis
Eğitim	Eğitici oyunlar, tavsiye, yeterlilik tahmini
Ulaştırma	Trafik kontrolü
Enerji	Karar kontrolü
Finans	Ticaret, risk yönetimi
Bilim, Mühendislik, Sanat	Matematik, Fizik, müzik, animasyon
İşletme Yönetimi	Tavsiye, müşteri yönetimi
Bilgisayar Sistemleri	Kaynak yönetimi, güvenlik
Oyunlar	Masa oyunları, kart oyunları, video oyunları
Robotik	Simülasyondan gerçeğe, kontrol
Doğal Dil İşleme	Dizi oluşturma, çeviri, diyalog

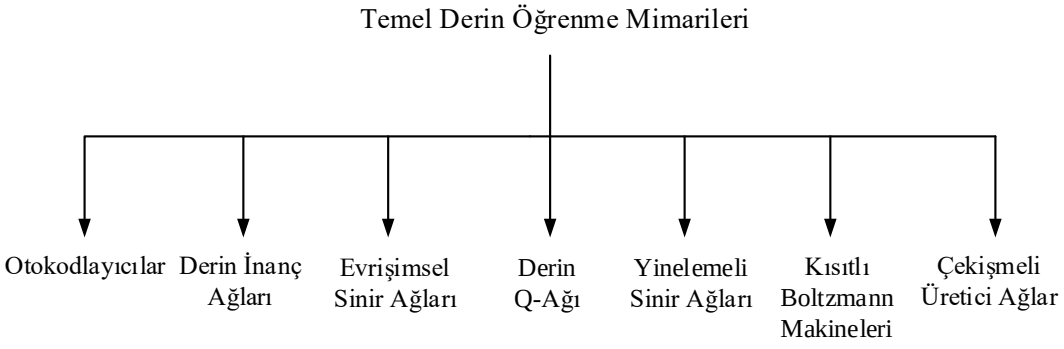
DPÖ'de politika optimizasyonu için değer tabanlı yöntemler ve politika tabanlı yöntemler olmak üzere iki ana kategori vardır. İkisinin bir birleşimi, aktör-kritik algoritmalar sınıfını ve politikayı güncellemek için değer fonksiyonundan yararlanan QT-Opt [175] gibi diğer algoritmaları oluşturur.

4.1. Derin Öğrenme

DPÖ'yü anlamak için derin öğrenmede güçlü bir temele sahip olmak gerekir. Derin öğrenme, makinelerin ses ve görüntü gibi büyük miktardaki veriyi algılayarak anlamasını sağlayan bir model oluşturmak için kullanılan yaygın bir makine öğrenmesi yaklaşımıdır. Bu yaklaşım, tamamen gizli katman sayısı arttırılmış YSA'lar ile ilgili olup derin mimarilere dayanmaktadır.

Makine öğrenmesindeki diğer algoritmalardan farklı olarak derin öğrenme algoritmaları, çok yüksek miktarda veriye ve bu veriyi işleyebilecek yüksek donanım gücüne ihtiyaç duymaktadır. Derin öğrenmenin on yıllardır var olmasına rağmen şu anda bu kadar popüler olmasının nedeni, grafik işlem birimi kısaca GPU (Graphical Processing Unit) tabanlı olan paralel hesaplama gücündeki büyük ilerlemeler ile büyük miktarda verinin kullanılabilirliğidir. Bu devasa veri hacmiyle derin öğrenme algoritmaları, tüm klasik makine öğrenimi algoritmalarından daha iyi performans göstermektedir. GPU'ların binlerce hesaplama çekirdeği ile artan hesaplama gücü sayesinde, çok daha fazla veri işlenebildiğinden, DSA'lar makine öğrenimine hükmetmiştir. Bilgisayarla görme, doğal dil işleme, pekiştirmeli öğrenme vb. alanlardaki son başarılar, DSA'ları kullanan derin öğrenme sayesinde mümkün olmuştur.

Birçok farklı derin öğrenme mimarisi vardır. Derin öğrenme mimarileri, ağ yapılarına ve kullanım amaçlarına göre temel olarak Şekil 4.2'deki gibi sınıflandırılmıştır [176].



Şekil 4.2. Temel derin öğrenme mimarilerinin sınıflandırılması [176]

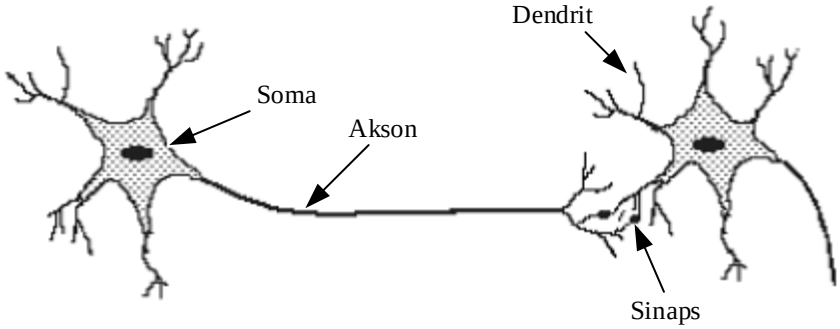
4.1.1. Yapay Sinir Ağları

Sinir ağı fikri, Rosenblatt ve diğerleri [177] tarafından bulunan algılayıcı adı verilen basit bir ikili sınıflandırıcının geliştirilmesinden sonra ortaya çıkmıştır. YSA'lar, insandaki öğrenme işleyişinden ilham alınarak geliştirilen matematiksel modeller olarak özetlenebilir. YSA'da birçok basit birim, merkezi bir kontrol ünitesine bağlı olmadan paralel olarak çalışmaktadır. Birimler arasındaki ağırlıklar vasıtasıyla bilgi depolama işlemi gerçekleştirilir ve yeni bilgiler öğrenilmesi bu ağırlıkların sinir ağında güncellenmesiyle sağlanmaktadır. YSA'ları anlamadan önce, biyolojik sinir sistemindeki nöronların (sinir hücresi) ne olduğu ve bu nöronların nasıl çalıştığı iyi anlaşılmalıdır.

İnsan beynindeki tek bir nöron, karmaşık görevleri yerine getiremez. Bu yüzden, insan beyninde bir ağ oluşturan katmanlar halinde düzenlenmiş milyarlarca nöron bulunmaktadır. Benzer şekilde, YSA sistemlerinde yapay nöron olarak tanımlanan yapılar, birbirleri ile bağlantılı olacak şekilde modellenmişlerdir. Böylelikle YSA'lar, öğrenme, hafızaya alma ve veriler arasındaki ilişkiyi yorumla kapasitesine sahip olabilmektedir [178].

Nöron

Nöron, insan beyninin temel hesaplama birimi olarak tanımlanabilir. Her nöron sinapslar aracılığıyla birbirine bağlıdır. Nöronlar, Şekil 4.3'te görülebileceği gibi, dendrit adı verilen dal benzeri bir yapı aracılığıyla dış ortamdan, duyu organlarından veya diğer nöronlardan giriş alır. Bu girişler güçlendirilir veya zayıflatılır, yani önemlerine göre ağırlıklandırılır ve soma (hücre gövdesi) içinde toplanır. Daha sonra somadan toplanan girişler işlenir ve aksonlar boyunca hareket eder ve diğer nöronlara gönderilir.



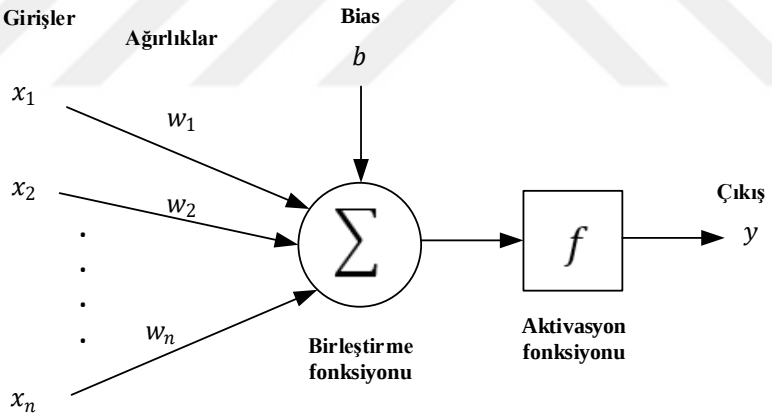
Şekil 4.3. Biyolojik bir nöronun yapısı

Tablo 4.2'de, biyolojik bir sinir ağı yapısının YSA'daki karşılıkları verilmiştir.

Tablo 4.2. Biyolojik bir sinir ağı yapısının YSA'daki karşılıkları

Biyolojik Sinir Sistemi	**Yapay Sinir Sistemi**
Nöron	Hesaplama birimi
Sinaps	Ağırlık
Soma	Aktivasyon fonksiyonu
Dendrit	Birleştirme fonksiyonu
Akson	Nöron çıkışı

YSA'daki bir nöron, Şekil 4.4'te gösterildiği gibi giriş, ağırlık, bias ve aktivasyon fonksiyonlarından oluşur.



Şekil 4.4. Yapay bir nöronun yapısı

Bir nöronun girişleri, sensörler aracılığıyla dış dünyadan gelen bilgiler olarak düşünülebilir. Bu girişler, ağ tarafından öğrenilmesi istenen örnekler ile belirlenir.

Yapay bir nöronun y çıkışını tahmin etmek için girişler, ağırlıklar ile çarpılır. Ağırlıkların büyük veya küçük olması, girişin nöron üzerindeki etkisinin derecesini ifade eder. Girişlerin ağırlıklar ile çarpılmasının sebebi, y çıkışının hesaplanmasında tüm girişlerin eşit derecede öneme sahip olmamasıdır. Örneğin; x_2'nin çıkışı hesaplamada diğer girişlere göre daha önemli olduğunu varsayalım. Bunun için, w_2'ye diğer ağırlıklardan daha yüksek bir değer atanır. Böylece ağırlıklar girişlerle çarpıldığında x_2, diğer girişlerden daha yüksek bir değere sahip olacaktır.

Girişler ağırlıklarla çarpıldıktan sonra birleştirme fonksiyonu ile toplanır ve b ile temsil edilen bir bias değeri eklenir. Bir nörona gelen net girişi hesaplayan birçok birleştirme fonksiyonu olmasına rağmen yaygın olarak ağırlıklı toplam kullanılır.

Nöronları doğrusal regresyondan ayıran fark, aktivasyon fonksiyonunun kullanılarak çıkışa doğrusal olmayanlığın katılmasıdır.

Bir nöronun tam denklemi (4.1)'deki gibi yazılabilir.

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{4.1}$$

Sinir ağları, doğrusal olmayan fonksiyonların yaklaşımında kullanılır ve aktivasyon fonksiyonu aracılığıyla farklı örüntüler tahmin edilebilir [45]. Çünkü gerçek dünya problemleri, genellikle doğrusal olmayan dönüşümlerdir [179]. Aktivasyon fonksiyonları, sinir ağlarında ağa doğrusal olmayanlık getirir ve girişin önemli dönüşümlerini yakalamak için kullanılır.

Bilimsel yazında yaygın olarak sigmoid, hiperbolik tanjant kısaca tanh, düzeltilmiş doğrusal birim (ReLU) ve Softmax aktivasyon fonksiyonları kullanılmaktadır.

Sürekli ve türevi alınabilen bir fonksiyon olan sigmoid fonksiyonu (4.2) ile ifade edilir.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{4.2}$$

(4.3) ile tanımlanan tanh fonksiyonu, sıfır merkezli olması dışında sigmoid fonksiyonuyla benzer davranışa sahiptir.

$$tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{4.3}$$

(4.4) ile tanımlanan Düzeltilmiş Doğrusal Birim kısaca DDB (Rectified Linear Unit–ReLU) fonksiyonu, pozitif değerleri olduğu gibi geçirirken negatif değerleri sıfıra eşleyen basit bir fonksiyondur.

$$DDB(x) = \max(0, x) \tag{4.4}$$

Softmax fonksiyonu ise sigmoid fonksiyonunun genelleştirilmiş halidir. Genellikle ağın son katmanında ve çok sınıflı sınıflandırma görevleri yapılırken uygulanır. $x \in \mathbb{R}^K$ verildiğinde, softmax, x'in her bir elemanına (4.5)'te verildiği gibi olasılık atar.

$$softmax(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{4.5}$$

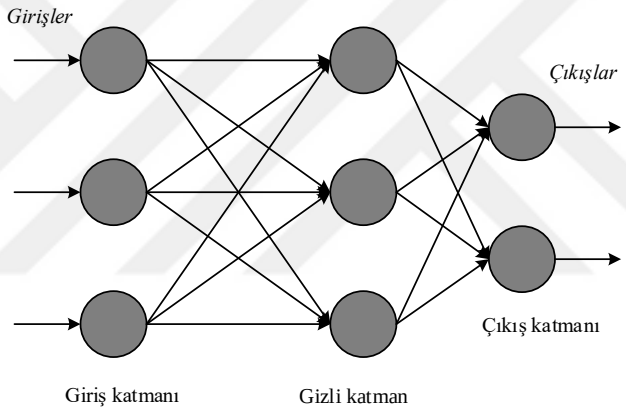
Aktivasyon fonksiyonu ile üretilen çıkış, başka bir nörona veya dış dünyaya gönderilir.

İleri Beslemeli YSA

İleri beslemeli bir YSA'da sadece ileri yönde bir bilgi akışı gerçekleşmektedir. Nöronlar, katmanlar halinde düzenlenir. Tipik bir ileri beslemeli YSA, temel olarak üç farklı katmandan oluşur: Giriş, gizli (ara) ve çıkış katmanı. Her katman nöronların bir yığınına sahiptir ve bir katmandaki nöronlar, diğer katmanlardaki tüm nöronlarla etkileşime girer. Ancak aynı katmandaki nöronlar birbirleriyle etkileşime girmezler.

İleri beslemeli bir YSA, bir veya birden fazla gizli katmana sahip olabilir. Ağ, gizli katman içermiyorsa tek katmanlı algılayıcı olarak adlandırılır. Tek katmanlı algılayıcılar ile doğrusal olmayan ilişkiler öğrenilemediği için çok katmanlı algılayıcılar geliştirilmiştir. Bu algılayıcılar bir veya birden fazla gizli katman içermektedir.

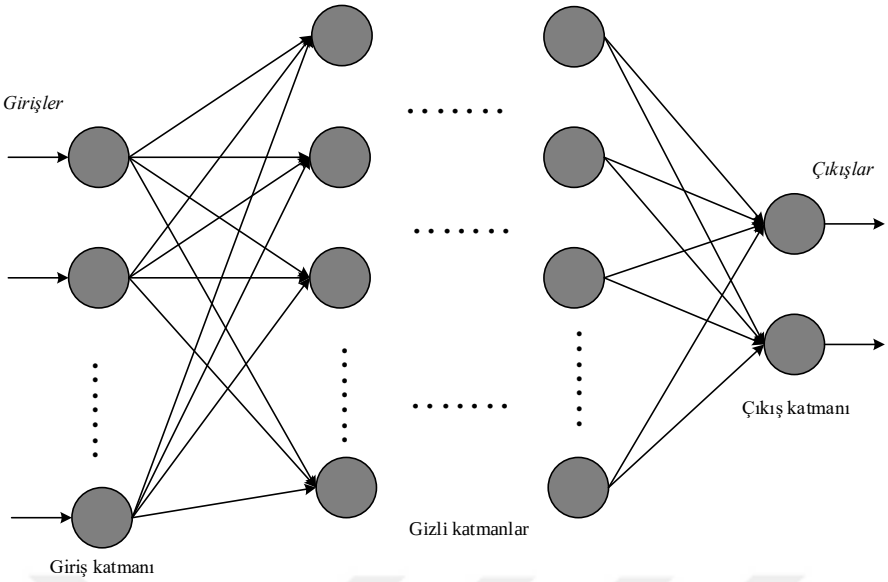
Şekil 4.5'te, örnek olarak iki katmanlı ileri beslemeli bir YSA'nın modeli verilmiştir.



Şekil 4.5. 2 katmanlı ileri beslemeli bir YSA modeli

Giriş katmanı, girişin ağa beslendiği yerdir. Giriş katmanındaki nöron sayısı, ağa beslenilen girişlerin sayısıdır. Her giriş, çıkışı tahmin etmede bir miktar etkiye sahip olur.

Giriş katmanı ile çıkış katmanı arasındaki herhangi bir katmana gizli katman denir. Giriş katmanından alınan girişi işler. Gizli katman, giriş ve çıkış arasındaki karmaşık ilişkilerin türetilmesinden sorumludur. Herhangi bir sayıda gizli katman olabilir, ancak probleme göre bir dizi gizli katman seçmek gerekir. Çok basit bir problem için sadece bir gizli katman kullanılabilirken görüntü tanıma gibi karmaşık görevler gerçekleştirilirken görüntünün kolayca tanınabilmesi için her katmanın görüntünün önemli özelliklerini çıkarmaktan sorumlu olduğu birçok gizli katman kullanılır. Çok katmanlı ileri beslemeli bir YSA'daki derinlik kavramı, gizli katmanlarının sayısı ile ilgilidir. Ağın gizli katman sayısı arttıkça derinliği artmaktadır. Birçok gizli katman kullanıldığında, ağa Derin Sinir Ağı (DSA) denir. Derin bir YSA modeli örnek olarak Şekil 4.6'da gösterilmiştir.



Şekil 4.6. Derin bir YSA modeli

Gizli katman, giriş katmanından gelen bilgileri işledikten sonra sonucunu çıkış katmanına gönderir. Adından da anlaşılacağı gibi, çıkış katmanı çıkışı yayar. Çıkış katmanındaki nöronların sayısı, ağın çözmesi istenilen problemin türü ile ilgilidir. İkili bir sınıflandırma işlemi gerçekleşmiş ise, çıkış katmanındaki nöronların sayısı girişin hangi sınıfa ait olduğunu söyler. Beş sınıflı bir sınıflandırma işlemi ise her sınıfın bir çıkış olma olasılığı elde edilir. Böylece, çıkış katmanındaki nöronların sayısı beştir ve her biri olasılığı yayar.

4.1.2. Ağ Eğitimi

DSA'ların en önemli görevi, ağırlıkları ve biasları ayarlamak ve geriye yayılım algoritmasını kullanarak eğitim sırasında maliyet fonksiyonunu en aza indirmeye çalışmaktır [180]. Bu algoritma ile her giriş için rastgele değerde ağırlıklarla başlanır ve daha sonra hesaplanan çıkış değerini gerçek çıkıştan çıkararak bir hata değeri bulunur [181]. DSA'lar ağırlık parametrelerinden oluşur. Öğrenme, istenen davranışı vermek için ağırlıkların güncellenmesi sürecidir. Bu davranış, bir kayıp (hata) fonksiyonunda temsil edilir.

Maliyet fonksiyonu, hata değerinin en aza indirilebilmesi için ağırlık parametrelerinin nasıl değiştirileceğini gösterir. Ağırlıklar genellikle gradyan iniş algoritması kullanılarak güncellenir [179].

Geriye Yayılım Algoritması

Bir kayıp fonksiyonunu en aza indirmek için, ağırlık parametrelerine göre gradyanı hesaplanmalıdır. Bu gradyanlar, temel hesabın zincir kuralı ile elde edilir. Bu nedenle, gradyan bilgisi geriye doğru yayılır ve bu işleme geriye yayılım denir. Geriye yayılım algoritması, çok katmanlı YSA'ların eğitiminde kullanılan popüler algoritmalardan biridir [182, 183].

Geriye yayılım algoritması temel olarak 6 adımdan oluşur [182]:

Adım 1: w ağırlık vektörü, gerçek değerli küçük sayılarla başlatılır.

Adım 2: Rastgele bir çalışma modeli seçilir ve q'inci katmandaki her bir j nöronu için çıkış değerleri ileri yönde hesaplanır. Çıkış, (4.6)'daki gibi hesaplanır.

$$y_i^q = f\left(\sum_i y_i^{q-1} w_{ij}^q\right) \tag{4.6}$$

Adım 3: (4.7) ile çıkış nöronlarındaki hatalar hesaplanır.

$$\delta_i^Q = \left(y_i^Q - y_i^p\right) f'\left(H_i^Q\right) \tag{4.7}$$

Adım 4: Hata değerleri, Gizli ve çıkış katmanlarındaki tüm i nöronları için geriye yayılım ile (4.8)'deki gibi hesaplanır.

$$\delta_i^{q-1} = f'\left(H_i^{q-1}\right) \sum_i \delta_i^q w_{ij}^q \tag{4.8}$$

Adım 5: w_{ij}'ler kullanılarak bütün ağırlıklar (4.9) ile güncellenir.

$$w_{ij}^{yeni} = w_{ij}^{eski} + \Delta w_{ij}^q \tag{4.9}$$

Δw_{ij}^q, (4.10) ile hesaplanır.

$$\Delta w_{ij}^q = \eta \delta_i^q y_i^{q-1} \tag{4.10}$$

Adım 6: Toplam hata istenen değere gelene kadar Adım 2'ye dönülüp her bir p modeli için işlemler tekrarlanır.

Burada; $q = 1, 2, 3, \ldots, Q$: katmanının sıra numarasını, H_i^p q'inci katmandaki i nöronunun girişini, y_i^q q'inci katmandaki i nöronunun çıkışını ve w_{ij}^q ise $(q-1)$'inci katmandaki i nöronunu, q'inci katmandaki j nöronuna bağlayan ağırlığı temsil eder.

Sayısal Optimizasyon

Öğrenme, sayısal optimizasyon yöntemleri ile kayıpların en aza indirilmeye çalışılmasıdır. Stokastik gradyan inişi [184], Adam [185], AdaGrad [186] ve RMSprop [187] en popüler optimizasyon yöntemleridir.

Sinir ağı tahminlerinin daha iyi olması için maliyet fonksiyonu en aza indirilmelidir. Hatanın en aza indirilmeye çalışıldığı süreç boyunca tüm ağırlıklar güncelleştirilerek ağ çıkışı optimize edilmeye çalışılır. Ağırlık matrisleri ve biaslar rastgele başlatıldığı için mükemmel değildir. Ağırlık matrisleri, sinir ağı iyi bir sonuç verecek şekilde ayarlanmalıdır. Bu, gradyan inişi adı verilen bir optimizasyon yöntemi ile yapılabilmektedir.

Gradyan inişi, ağırlık parametrelerini güncelleyerek $\mathcal{L}$ kayıp fonksiyonunu en aza indirir. Örneğin; önceden tanımlanmış bir η öğrenme oranı ile gradyan yönünün tersine θ, (4.11) ile ifade edilir.

$$\theta \leftarrow \theta - \eta \nabla \mathcal{L}(\theta) \tag{4.11}$$

Makine öğrenmesi problemlerinde, kayıp fonksiyonları genellikle (4.12)'deki gibi $\mathcal{L}_i$ örnek kayıpların formunu toplar.

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_i(\theta) \tag{4.12}$$

Stokastik gradyan inişi, örnek kayıplar ile kayıp fonksiyonunun gradyanını tahmin eder ve parametreleri buna göre (4.13)'te verildiği gibi günceller.

$$\theta \leftarrow \theta - \eta \nabla \mathcal{L}_i(\theta) \;\; \forall i \in \{1, 2, \dots N\} \tag{4.13}$$

Bununla birlikte, pratikte, kayıp gradyanını tahmin etmek için mini yığınlar kullanılır. Bu durumda, N_b boyutundaki yığınlar örneklerden (4.14) ile örneklenir ve güncellemeler (4.15)'e göre yapılır.

$$\mathcal{L}_j(\theta) = \frac{1}{N_b}\sum_{i=1+(j-1)N_b}^{jN_b} \mathcal{L}_i(\theta) \tag{4.14}$$

$$\theta \leftarrow \theta - \eta \nabla \mathcal{L}_j(\theta) \quad \forall j \in \left\{1, 2, \dots \left[\frac{N}{N_b}\right]\right\} \tag{4.15}$$

Adam, RMSProp algoritmasının iyileştirilmesi olarak stokastik gradyan inişinin bir çeşididir. RMSprop'ta olduğu gibi ikinci gradyan momenti kullanılarak öğrenme oranı ölçeklendirilir ve hem birinci hem de ikinci gradyan momenti için momentum tahmini kullanılır. Günümüzde derin öğrenmede en çok kullanılan optimizasyon yöntemlerinden biridir. Adam, eğitimi hızlandırmak ve sağlam hale getirmek için stokastik güncellemelerden kaynaklanan sorunların üstesinden gelmek için eğitim verilerine dayanarak adım uzunluğunu ayarlar. Sözde

kodu Algoritma 4.1'de özetlenmiştir. $\odot$ ve $\oslash$'nin sırasıyla eleman bazında çarpma ve bölmeyi ifade etmektedir.

Algoritma 4.1. Adam optimizasyonu

Başlat: η öğrenme oranı, β_1, β_2 hareketli ortalama parametreleri

θ_0 başlangıç model parametreleri

Gradyanların $m \leftarrow 0, v \leftarrow 0$ başlangıçtaki birinci ve ikinci momenti

$j \leftarrow 0$ başlangıç adımı

while θ_j yakınsak değil **do**

$j \leftarrow j + 1$

$g_j \leftarrow \nabla \mathcal{L}_j(\theta)$ (Gradyan elde et)

$m_j \leftarrow \beta_1 m_{j-1} + (1 - \beta_1) g_j$ (Birinci moment tahminini güncelle)

$v_j \leftarrow \beta_2 v_{j-1} + (1 - \beta_2) g_j \odot g_j$ (İkinci moment tahminini güncelle)

$\hat{m}_j \leftarrow \frac{m_j}{1-\beta_1^j}$ (Birinci moment biası düzeltme)

$\hat{v}_j \leftarrow \frac{v_j}{1-\beta_2^j}$ (İkinci moment biası düzeltme)

$\theta_j \leftarrow \theta_{j-1} - \eta \hat{m}_j \oslash (\hat{v}_j + \epsilon)$ (Parametreleri güncelle)

end

Normalleştirme

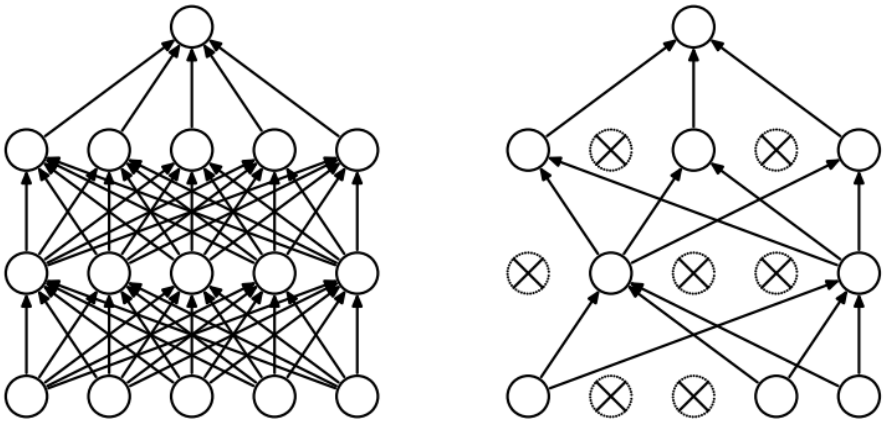
Normalleştirme, öğrenme sırasında kararsızlık ve sapmanın üstesinden gelmek için kullanılır. Gerçek dünya verilerinde, verilerin sütunları arasındaki varyans ve ortalama ölçekler önemli ölçüde değişebilir [188]. Bu durum, optimal öğrenme oranı ve diğer hiperparametrelerin seçilmesini zorlaştırır ve sonuç olarak gradyanların yanlış yönlendirilmesi meydana gelir. Yığın normalleştirme [189], veri setinden mini yığın olarak bir örnek alan ve ortalamalarını sıfıra eşit ve varyansları bir birim olarak normalleştiren bu zorluğu çözmektedir. Böylece, gradyanın optimum değeri bulur.

Düzenlileştirme

YSA'lar eğitilirken, sahip oldukları parametrelerden dolayı doğru olarak eğitilmediğinde eğitim verisini ezberleme eğilimindedir. Düzenlileştirmedeki temel amaç öğrenmedeki aşırı öğrenmeyi diğer bir deyişle ezberleme problemini kontrol altında tutmaktır. Tüm ağırlıklar her eğitim adımı ile birlikte güncellendiğinde bazıları, diğer nöronlara göre daha iyi tahmin kabiliyetine sahip hale gelir ve bu eğitimin sonuna kadar devam eder [179]. Bu durum, bazı nöronların tahmin kapasitelerini azaltmasına neden olur. Bu nedenle, bir düzenlileştirme yöntemi olan Dropout (seyreltme) [188], her eğitim adımında bazı nöronların ağırlığını devre dışı bırakarak ve tahmin kapasitelerini eşitlemeye çalışarak daha derin ve daha geniş sinir ağları oluşturabilmeyi sağlar

[179]. Bu yöntem ile giriş katmanı veya gizli katmandan belirli bir eşik değeri kullanılarak (genellikle 0.5), nöronlar sinir ağından kopartılır.

3 katmanlı bir YSA için Dropout yöntemi ile gerçekleştirilen düzenlileştirme işlemi, Şekil 4.7'de gösterilmiştir.



Şekil 4.7. Bir ağdaki düzenlileştirme işlem örneği: (a) standart bir YSA ve (b) seyreltme işleminden sonra YSA

4.1.3. Derin Öğrenme Kütüphaneleri ve Yazılım Geliştirme Araçları

Derin öğrenme ve insan beyni verileri benzer şekilde işleme yönünden ortak bir noktayı paylaşır. Derin öğrenme, insan beyniyle aynı şekilde tasarlanmış hiyerarşik düzeyde YSA'ları kullanır. Derin öğrenme, karar verme, nesne algılama, konuşma tanıma ve dil çevirisi için kullanılabilir. Geleneksel makine öğrenimi algoritmaları verileri doğrusal olarak analiz ederken derin öğrenmenin hiyerarşik işlevi, bilgisayarların verileri doğrusal olmayan bir şekilde yorumlamasına olanak tanır.

Çok aşamalı ve yinelemeli bir süreç olan derin öğrenmenin iş akışında ilk olarak veriler toplanır ve gerekirse veriler ön işlemlere tabi tutulur. Başarılı bir derin öğrenme uygulaması gerçekleştirebilmek için büyük boyutta veri setlerine ihtiyaç duyulmaktadır. Son zamanlarda, büyük veri kaynakları ve internet vasıtasıyla hızla büyümektedir. Derin ağlar, bu veri setleri kullanılarak eğitilir. Ağ eğitimi için mevcut veya kullanıcı tarafından yeni geliştirilen ağlar kullanılabilir. Ağ eğitildikten sonra elde edilen ağ modelinin istendiği gibi çalışıp çalışmadığını doğrulamak için test edilmelidir. Sonuçlar yetersiz ise sonuçların iyileştirilebilmesi için veriler yeniden işlenebilir, ağ katmanları düzenlenebilir, ağ hiperparametreleri değiştirilebilir ve istenen sonuç elde edilinceye kadar test işlemleri yeniden gerçekleştirilir. Bu süreçte yapılacak tüm bu yoğun işlemler için Merkezi İşlem Birimleri kısaca CPU'lar yeterli olamamaktadır. Belli bir

mimariye ve işlem kapasitesine sahip olan CPU'lar aynı anda çok sayıda işlemi yerine getiremediğinden bir ağ modelinin eğitimi ve test işlemleri zamansal olarak oldukça maliyetli olmaktadır. Bu sebeple, günümüzde CPU'lar yerlerini GPU'lara bırakmışlardır. Verilerin paralel olarak işlenmesine olanak sağlayan GPU'lar sayesinde derin öğrenme gerçek hayattaki birçok uygulamada kullanılabilmektedir.

Derin öğrenme uygulamaları geliştirilirken kullanılan GPU'lar, NVIDIA'nın CUDA eklentisi aracılığıyla paralel hesaplamalar yapabilmektedir [190]. Paralel hesaplama mimarisine sahip olan CUDA, hesaplamalar için grafik işlemcinin çekirdeklerinin kullanılmasına olanak tanır. İlk CUDA sürümü 2007 yılında piyasaya sürülmüş olup NVIDIA'nın Quadro, Tesla ve GeForce serisi GPU'lar tarafından desteklenmektedir.

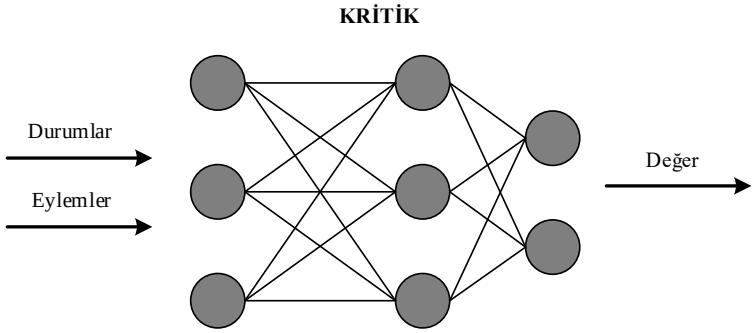
Derin öğrenme kütüphaneleri ve yazılım geliştirme araçları, üst düzey bir programlama arabirimi aracılığıyla derin sinir ağlarını geliştirmek, eğitmek ve değerlendirmek için kullanılır. Keras [191], PyTorch [192] ve TensorFlow [193] başta olmak üzere birçok kütüphane ve yazılım geliştirme araçları bulunmaktadır.

Tez kapsamında DPÖ ile ilgili gerçekleştirilen çalışmalarda, Keras ve Tensorflow kütüphanelerinden yararlanılmış ve bu kütüphanelere uyumlu olan OpenAI Gym araç takımı kullanılmıştır.

4.2. Değer Tabanlı Öğrenme Yöntemleri

Kritik olarak da bilinen değer tabanlı öğrenme genellikle eylem-değer fonksiyonunun $Q^{\pi}(s,a)$ optimizasyonunu ifade eder. Optimizasyondan sonra $Q^{\pi^*}(s,a) = \max_a Q^{\pi}(s,a)$ optimal değer fonksiyonu, $\pi^* \approx \arg\max_{\pi} Q^{\pi}$ optimal politika (tahmin hatasından dolayı "≈") ile türetilebilir.

DPÖ'de değer fonksiyonu bir sinir ağı ile temsil edilir. Fikir, Q-öğrenmedeki bir tabloyla aynı şekildedir. Durum gözlemleri ve bir eylemi giriş olarak verilir, sinir ağı o durum-eylem çiftinin değerini döndürür ve politika, en yüksek değere sahip eylemi seçer. Zamanla ağ, sürekli durum uzayında herhangi bir yerde her eylem için gerçek değeri veren bir fonksiyon üzerinde yavaş yavaş yakınsar. DPÖ'deki değer tabanlı öğrenme yapısı Şekil 4.8'de verilmiştir.



Şekil 4.8. Değer tabanlı öğrenme yapısı

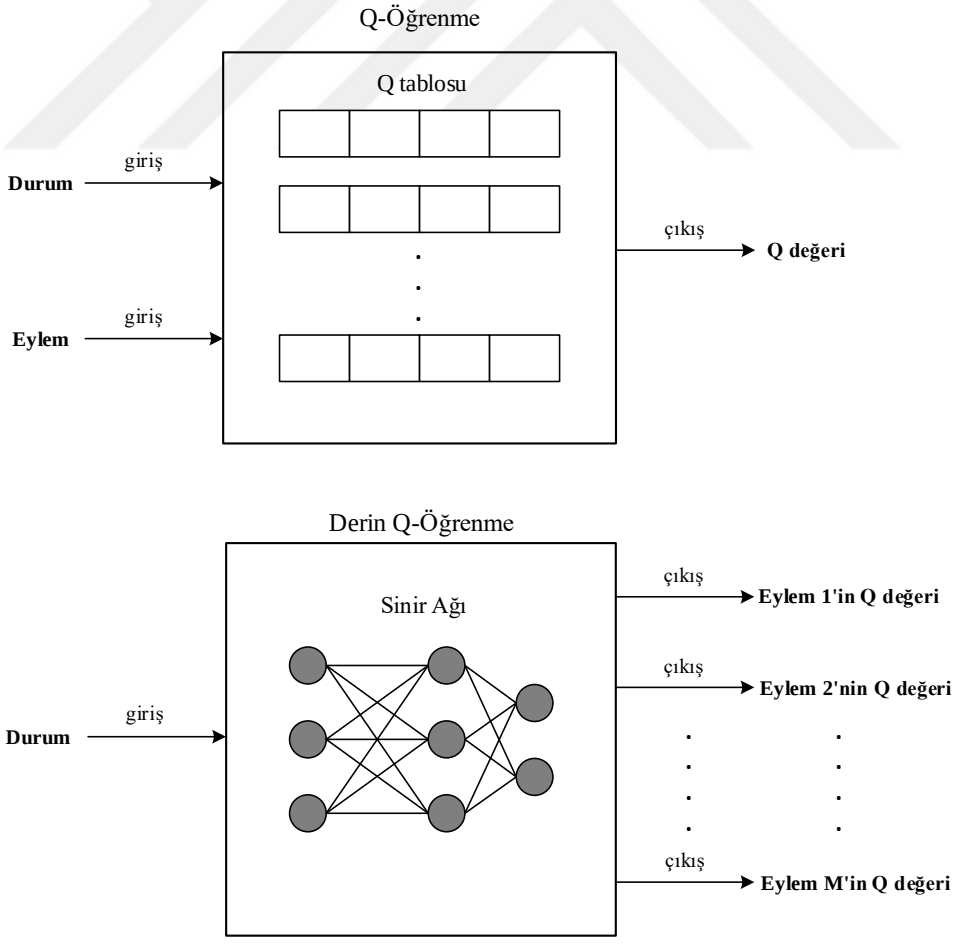
Değer tabanlı DPÖ'de bir dönüm noktası, 2013'te Google DeepMind'in Atari oyunlarını oynamak için Q-öğrenme ve derin Evrişimsel Sinir Ağ'ları kısaca ESA'ları (Convolutional Neural Networks–CNNs) başarılı bir şekilde birleştiren Derin Q Ağı kısaca DQA [194] algoritmasını kullanmasıdır. DQA'da ajan, Q tablosu yerine derin sinir ağları tarafından eğitilir. Bu yapıda, çevreden gelen mevcut durumlar girişler ve ajan eylemleri ise çıkışlardır.

Değer tabanlı öğrenmenin avantajları, örnek veriminin yüksek olması, değer fonksiyonu tahmininin varyansının küçük olması ve yerel optimuma düşmenin kolay olmamasıdır. Ayrıca, ayrık eylem-durum uzaylarında daha iyi sonuçlar verir ve genel olarak daha hızlı öğrenmektedirler. Dezavantajları ise genellikle sürekli eylem alanı sorununu çözememesidir ve ϵ-açgözlü stratejisi ve DQA'daki maksimum operatörün kolayca aşırı tahmine neden olabilmesidir. Değer fonksiyonu tabanlı politikalar, sürekli eylem alanları için iyi çalışmayacaktır. Bunun nedeni, maksimum değeri bulmak için her sonsuz eylem için değeri birer birer hesaplamanın bir yolu olmamasıdır.

2015 yılında van Hasselt ve diğerleri [90], çift Q-öğrenme [195] fikriyle birleştirilmiş derin sinir ağlarını kullanan Çift Derin Q Ağı'nı (ÇDQA'yı) önermiş ve Atari oyunlarında daha doğru sonuçlar verdiğini ispatlamıştır. Çift DQA, aşırı tahmin problemini çözmek için farklı parametrelere sahip eylemleri seçer ve değerlendirir. Schaul ve diğerleri [196], Çift DQA ile öncelikli deneyim tekrarını kullanarak deneyim tekrarından öğrenmeyi daha verimli hale getirmiştir. 2016 yılında Wang ve diğerleri [197], durum değerini ve her eylemin avantajlarını ayrı ayrı tahmin etmek için iki akışa sahip olan ve benzer değerli eylemlerin varlığında daha iyi politika değerlendirmesine yol açan Düello Derin Q Ağı kısaca Düello DQA'yı (Dueling Deep Q Network–Dueling DQN) sunmuştur. Fortunato ve diğerleri tarafından önerilen Gürültülü DQA [198] ise keşfi arttırmak için ağırlıklarına parametrik gürültü eklenmiş bir ajana sahiptir ve keşif için stokastik ağ katmanlarını kullanır.

4.2.1. Derin Q Ağı

Durum-eylem değer fonksiyonu olarak da adlandırılan bir Q fonksiyonu, s durumundaki bir a eyleminin ne kadar iyi olduğunu belirtir. Böylece, her durumdaki tüm olası eylemlerin değeri Q tablosu adı verilen bir tabloda saklanır ve bir durumda en yüksek değere sahip eylem optimum eylem olarak seçilir. Q fonksiyonunu tahmin etmek için politika dışı bir ZF öğrenme algoritması olan Q-öğrenme kullanılmaktadır. Sınırlı sayıda eyleme ve duruma sahip çevrede optimal Q değerini bulmak için tüm olası durum-eylem çiftlerinde kapsamlı bir arama yapılır. Çok fazla sayıda durumun ve her durumda denenecek çok fazla sayıda eylemin olduğu bir çevre düşünüldüğünde her bir durumdaki tüm eylemlerin üzerinden geçmek zaman alıcı olacaktır. Böyle durumlarda, bir Q fonksiyonuna yaklaşmak için fonksiyon tahmin edicileri olarak YSA'ların kullanılması derin Q-öğrenme olarak adlandırılır. Derin Q-öğrenmede, $Q(s,a)$ eylem-değer fonksiyonu, girişleri s, a ve θ ağ parametreleri olan $Q(s,a;\theta)$ ile belirtilen regresyon ağı ile değiştirilir. Q-öğrenme ve derin Q-öğrenme'nin işleyişi Şekil 4.9'da gösterilmiştir.



Şekil 4.9. Q-öğrenme ve derin Q-öğrenme'nin işleyişi

Q-öğrenme'nin bir tablo durumunda veya doğrusal fonksiyon yaklaşımı kullanarak optimal çözüme yakınsadığı kanıtlanmıştır. Bununla birlikte, Q değer fonksiyonunu temsil etmek için bir sinir ağı gibi doğrusal olmayan bir fonksiyon tahmincisi kullanıldığında Q-öğrenmenin kararsız olduğu ve hatta ıraksadığı bilinmektedir [199]. Derin sinir ağlarının eğitimindeki gelişmelerle birlikte Derin Q Ağı (DQA) [81], bu konuyu ele alarak DPÖ araştırmalarının önünü açmıştır. Google'ın DeepMind araştırmacıları tarafından önerilen DQA, çok popüler ve yaygın olarak kullanılan DPÖ algoritmalarındandır. Bir atari oyununu oynarken sadece oyun ekranını giriş alarak insan düzeyinde sonuçlar elde edilmiştir.

DQA'da (4.16) ile ifade edilen kayıp fonksiyonu, hedef ve tahmin edilen değer arasındaki karesi alınmış fark olarak tanımlanır ve θ ağırlıkları güncellenerek kayıp minimize edilmeye çalışılır.

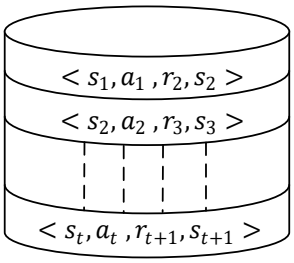
$$L(\theta) = \left(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a;\ \theta) \right)^2 \tag{4.16}$$

Burada; $\gamma \max_{a'} Q(s', a'; \theta)$ hedef değer, $Q(s, a;\ \theta)$ ise tahmin edilen değerdir. Ağırlıklar güncellenir ve gradyan iniş ile kayıp en aza indirilir.

DQA'da Q tahmini için doğrusal olmayan bir tahminci kullanıldığında, güncellenmiş Q fonksiyonu ve son gözlemler arasındaki korelasyonlar nedeniyle öğrenme kararsızdır. Karmaşık problemlerde uçtan uca karar vermeyi başarmak için DQA, kararsızlık sorununu çözmek için Q-öğrenme ile derin öğrenmeyi iki temel fikirle birleştirir: Deneyim tekrarı ve hedef ağ [81, 194].

Deneyim Tekrarı

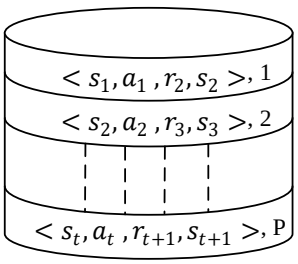
Deneyim tekrarı, biyolojik olarak esinlenilmiş bir mekanizma olan tekrar tamponu olarak bilinir [200-202]. PÖ çevrelerinde, bir a eylemi gerçekleştirerek bir s durumundan sonraki s' durumuna geçiş yapıldığı ve bir r ödülü alındığı bilinmektedir. Bu geçiş bilgisi, bir arabellekte Şekil 4.10'daki gibi $< s, a, r, s' >$ olarak kaydedilir. Bu geçişlere ajanın deneyimi denir. Ajanın çevredeki deneyimleri $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$ şeklinde $\mathcal{D} = e_1, \ldots, e_N$ olarak depolanır.



Şekil 4.10. Tekrar tamponu

Deneyim tekrarının ana fikri, DQA'yı son geçişlerle eğitmek yerine tekrar tamponundan örneklenen mini yığın şeklinde çekilen geçişlerle eğitmektir. Ajanın deneyimleri birer birer ilişkilendirilir, bu nedenle tekrar tamponundan rastgele bir grup eğitim örneği seçmek, ajanın deneyimi arasındaki korelasyonu azaltacak ve ajanın çok çeşitli deneyimlerden daha iyi öğrenmesine yardımcı olacaktır. Ayrıca, sinir ağları, ilişkili deneyimle aşırı tahmin edecektir, bu nedenle tekrar tamponundan rastgele bir deneyim grubu seçilerek fazla uyum azaltılır. Deneyimi örneklemek için tek tip örnekleme kullanılabilir. Deneyim tekrarı, bir listeden ziyade bir sıra olarak düşünülebilir. Tekrar tamponu yalnızca sabit bir N sayıda son deneyimi depolar, bu nedenle yeni bilgi geldiğinde eskisi silinir.

Tekrar tamponundan tek biçimli (rastgele) örnekleme geçişleri optimal bir yöntem değildir. Bazı deneyimler diğerlerinden daha önemli olabilir. Bu nedenle geçişler, önceliklendirilebilir ve önceliğe göre örneklenebilir. Geçişlere öncelik vermek, ağın hızlı ve etkili bir şekilde öğrenmesine yardımcı olur. Yüksek ZF hatası olan geçişlere öncelik verilir. Bir ZF hatasının, tahmini Q değeri ile gerçek Q değeri arasındaki farkı belirlediği bilinmektedir. Dolayısıyla, yüksek ZF hatasına sahip geçişler, odaklanılması ve öğrenilmesi gereken geçişlerdir, çünkü bunlar tahminden sapan geçişlerdir. Bir deneyim birçok kez kullanıldığı için Şekil 4.11'deki gibi öncelikli tekrar tamponu sayesinde önemli deneyimlerin hızla unutulmasına karşı önlem alınır.



Şekil 4.11. Öncelikli tekrar tamponu

Hedef Ağ

İstenen Q ağı yerine, Q-öğrenme hedeflerini oluşturmak için hedef ağ olarak bilinen ayrı bir ağ kullanılır. Ayrıca, her C adımında, hedef ağ, doğrudan kopyalama (sert güncelleme) veya katlanarak azalan ortalama (yumuşak güncelleme) yoluyla birincil Q ağı ile senkronize edilir. Hedef ağ, sapma ve salınımları çok daha fazla azaltan eski parametrelerle Q-öğrenme hedef gecikmesinin oluşturulmasını sağlar. Hedef değer ile tahmin edilen değeri hesaplamak için aynı Q fonksiyonu kullanılır. (4.16)'daki denklemde, hem hedef Q hem de tahmin edilen Q için aynı θ ağırlıklarının kullanıldığı görülebilir.

Aynı ağ, tahmin edilen değeri ve hedef değeri hesapladığı için, bu ikisi arasında çok fazla farklılık olabilir. Bu sorunu önleyerek hedef değeri hesaplamak için hedef ağ adı verilen ayrı bir ağ kullanılır. Hedef değer, (4.17)'deki gibi θ^- ile parametrelendirilir. Böylece kayıp fonksiyonu (4.18)'deki hale gelir.

$$Y_t^{DQA} = r_t + \gamma \max_{a'} Q(s_{t+1}, a';\ \theta^-) \tag{4.17}$$

$$L(\theta) = \left(r + \max_{a'} Q(s', a'; \theta^-) - Q(s, a;\ \theta) \right)^2 \tag{4.18}$$

Q değerlerini tahmin etmek için kullanılan gerçek Q ağı, gradyan inişini kullanarak doğru ağırlıkları öğrenir. Hedef ağ birkaç zaman adımı için dondurulur ve ardından hedef ağ ağırlıkları, ağırlıkların gerçek Q ağından kopyalanmasıyla güncellenir.

DQA'da yer alan adımlar aşağıdaki gibidir:

1) s durumu DQA'ya beslenir, bu durumdaki tüm olası eylemlerin Q değerleri döndürülür.
2) ϵ-açgözlü politikası kullanılarak bir eylem seçilir. ϵ olasılıkla rastgele bir a eylemi seçilir ve $1 - \epsilon$ olasılıkla $a = \arg max \left(Q(s, a;\ \theta)\right)$ gibi maksimum Q değerine sahip bir eylem seçilir.
3) a eylemi seçildikten sonra bir s durumunda bu eylem gerçekleştirilip yeni bir s' durumuna geçilir ve bir ödül alınır.
4) Bu geçiş, tekrar tamponunda $< s, a, r, s' >$ olarak saklanır.
5) Tekrar tamponundan bazı rastgele geçiş grupları örneklenir ve kayıp $L(\theta) = \left(r + \max_{a'} Q(s', a'; \theta^-) - Q(s, a;\ \theta) \right)^2$ ile hesaplanır.
6) Bu kaybı en aza indirmek için gerçek θ ağ parametrelerine göre gradyan inişi gerçekleştirilir.
7) Her k adımdan sonra θ gerçek ağ ağırlıkları, θ^- hedef ağ ağırlıklarına kopyalanır.
8) Bu adımlar, M sayıda bölüm için tekrarlanır.

Deneyim tekrarını kullanan DQA algoritmasının sözde kodu Algoritma 4.2'de verilmiştir.

Algoritma 4.2. Deneyim tekrarlı DQA

Hiperparametreler: N kapasiteli tekrar tamponu, γ ödül indirim faktörü, hedef eylem-değer fonksiyonu güncellemesi için C gecikmeli adımlar, ϵ-açgözlü faktörü

Giriş: Boş $\mathcal{D}$ tekrar tamponu, Q eylem-değer fonksiyonunun θ başlangıç parametreleri

$\theta^- \leftarrow \theta$ parametresiyle hedef $\hat{Q}$ eylem-değer fonksiyonunu başlat

for $bölüm = 1, M$ **do**

s_1 başlangıç durumunu al

for $t = 1, T$ **do**

ϵ olasılıkla rastgele bir a_t eylemi seç, aksi halde $a_t = \arg\max_a Q(s_t, a; \theta)$ eylemi seç

a_t eylemini gerçekleştir, r_t ödülü ve s_{t+1} bir sonraki durumu al

$e_t = (s_t, a_t, r_t, s_{t+1})$ deneyim demetini $\mathcal{D}$'ye sakla

$\mathcal{D}$'den $\mathcal{D}_r$ $(\mathcal{D}_r| = N)$ rastgele mini yığını örnekle

$$Y_j = \begin{cases} r_j, & \text{eğer } s_{j+1}\ terminal\ ise \\ r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a';\ \theta^-), & aksi\ taktirde \end{cases} \quad \forall e_j \in \mathcal{D}_r$$

Tek bir adım için $\frac{1}{N}\sum_{e_j \in \mathcal{D}_r}\left[Y_j - Q(s_j, a_j; \theta)\right]^2$'i en aza indirerek θ'yı güncelle

if t mod d **then** her C adımında hedef ağı güncelle: $\theta^- \leftarrow \theta$

end

end

4.2.2. Çift Derin Q Ağı

DQA'da, (4.17)'deki gibi aynı ağ üzerindeki eylemi hem seçmek hem de değerlendirmek için maksimum operatörü kullanılır. Bu, gürültülü çevrelerde Q fonksiyonunun aşırı tahmin edilmesini sağlar. Bunun nedeni, algoritmanın en yüksek eylem değerini beklenen eylem değeri olarak kullanmasıdır. Bu sorun, her biri bağımsız olarak öğrenen iki ayrı Q fonksiyonuyla çözülebilir. Bir Q fonksiyonu bir eylemi seçmek için kullanılır ve diğer Q fonksiyonu bir eylemi değerlendirmek için kullanılır. Bu, sadece DQA'nın hedef fonksiyonu değiştirilerek uygulanabilir.

Hasselt ve diğerleri [90], eylem değerlerini düşük tahmin ederek bu aşırı tahmin probleminin üstesinden gelmek için bir çift tahmin yöntemi olan çift Q-öğrenme'yi tanıtmıştır. İki adet eylem değer fonksiyonu tanımlanır ve her adımda rastgele seçilen biri en yüksek değere sahip eylemi seçmek için kullanılırken, güncelleme yapılırken diğerinin o eylem için değeri kullanılır. Her iki eylem değer fonksiyonu da farklı deneyimlerle aynı problem üzerinde eğitildiğinden aşırı tahmine neden olan yanlılık ortadan kaldırılır.

Çift Q-öğrenme fikri derin Q-öğrenme ile birleştirilmiş ve Çift Derin Q Ağı kısaca ÇDQA (Double Deep Q Network–DDQN) [90] algoritması ortaya çıkmıştır. Derin Q-öğrenme zaten ikinci bir eylem değer fonksiyonu olarak kullanılabilecek bir yedek hedef ağına sahiptir. Gerçek ağ, en yüksek değere sahip eylemi seçmek için kullanılırken hedef ağ, bu eylemin değerini tahmin etmek

için kullanılır. Bu nedenle, ÇDQA'daki hedef değer DQA'dakinden farklı olarak (4.19)'daki gibi yazılabilir.

$$Y_t^{\text{ÇDQA}} = r_t + \gamma Q\left(s_{t+1}, \arg\max_{a'} Q(s_{t+1}, a'; \theta);\ \theta^-\right) \tag{4.19}$$

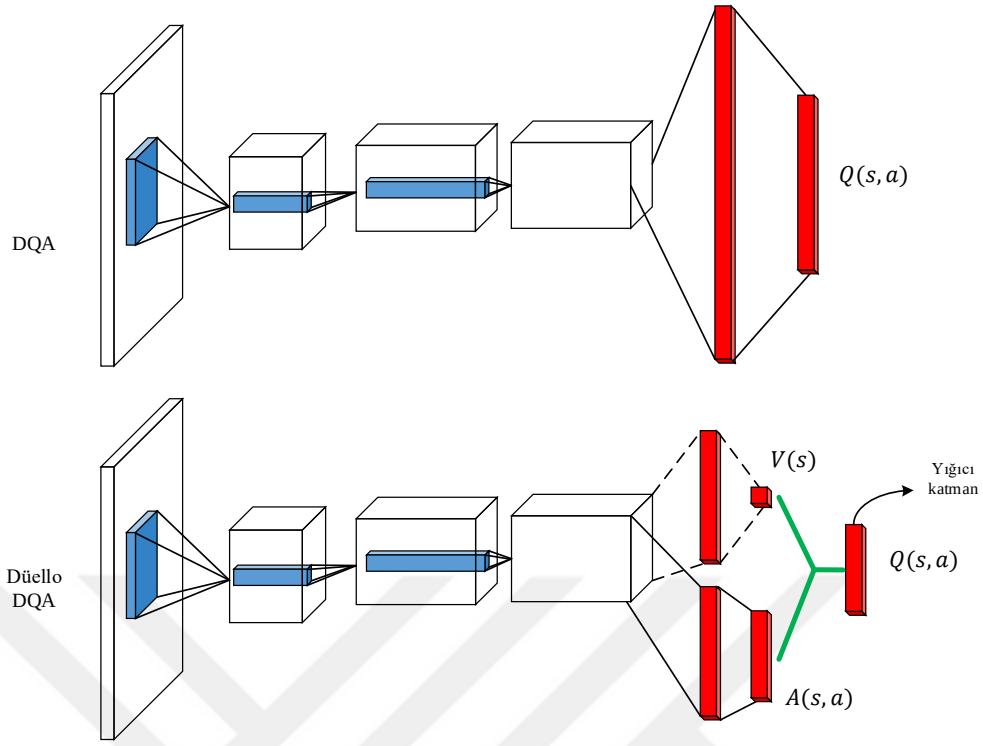
(4.19)'daki denklemde, her biri farklı ağırlıklara sahip iki Q fonksiyonu vardır. Dolayısıyla, eylemi seçmek için θ^- ağırlıklı bir Q fonksiyonu kullanılırken eylemi değerlendirmek için θ ağırlıklı diğer bir Q fonksiyonu kullanılır. Hedef değer dışında, öğrenme süreci DQA'daki ile aynıdır.

4.2.3. Düello Derin Q Ağı

Q fonksiyonunun, bir ajanın s durumunda bir a eylemi gerçekleştirmesinin ne kadar iyi olduğunu ve değer fonksiyonunun, bir ajanın s durumunda olmasının ne kadar iyi olduğunu belirlediği bilinmektedir. Düello Derin Q Ağı'nda (Düello DQA) Q değeri, $V(s)$ değer fonksiyonu ve duruma bağlı $A(s, a)$ eylem avantaj fonksiyonu ile farklı bir formülle hesaplanır [197]. Avantaj fonksiyonu, bir ajanın diğer eylemlere kıyasla bir a eylemini gerçekleştirmesinin ne kadar iyi olduğunu belirtir. Bu nedenle, değer fonksiyonu bir durumun iyiliğini belirtir ve avantaj fonksiyonu bir eylemin iyiliğini belirtir. Avantaj fonksiyonu, (4.20) ile ifade edilir.

$$A^{\pi}(s, a) = Q^{\pi}(s, a) - V^{\pi}(s) \tag{4.20}$$

Düello DQA, durum değerini ve her eylemin avantajlarını ayrı ayrı tahmin etmek için iki akışa sahip bir Q ağı türüdür. Şekil 4.12'de popüler bir tek akışlı DQA ile Düello DQA'nın mimarisi gösterilmiştir. Düello DQA'daki bir akış değer fonksiyonunu, diğer akış ise avantaj fonksiyonunu hesaplar. Düello DQA mimarisi, sonunda tamamen bağlı katmanın iki akışa bölünmesi dışında, temel olarak DQA ile aynıdır.



Şekil 4.12. Popüler bir tek akışlı DQA ve Düello DQA

Şekil 4.12'de görüldüğü gibi Düello DQA'daki iki akış, Q durum-eylem değeri fonksiyonunun bir tahminini üretmek için özel bir yığıcı katman aracılığıyla birleştirilir. Bu nedenle, bir düello ağı, standart DQA mimarisinden daha etkili ve sağlamdır. Q fonksiyonu, (4.21)'deki gibi bir değer fonksiyonu ile bir avantaj fonksiyonunun toplamı olarak tanımlanabilir.

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|\mathcal{A}|}\sum_a A(s,a)\right) \tag{4.21}$$

Birçok durumda, özellikle bir durumda geniş bir eylem alanı olduğunda, tüm eylemlerin değer tahminlerini hesaplamak önemli değildir. O zaman eylemlerin çoğunun durum üzerinde herhangi bir etkisi olmayacaktır. Ayrıca, gereksiz etkileri olan birçok eylem olabilir. Bu durumlarda, Düello DQA, Q değerlerini mevcut DQA mimarisinden daha kesin olarak tahmin eder. Düello DQA'daki ilk akış, durumda çok sayıda eylem olduğunda ve her bir eylemin değerini tahmin etmenin gerçekten önemli olmadığı durumlarda kullanışlıdır. İkinci akış ise ağın diğerine göre hangi eylemin tercih edildiğine karar vermesi gerektiğinde kullanışlıdır.

4.2.4. Düello Çift Derin Q Ağı

Düello Çift Derin Q Ağı kısaca Düello ÇDQA(Dueling Double Deep Q Networks–D3QN) [80], geleneksel DQA ile karşılaştırıldığında durum değerinin fazla tahmin edilmesi sorununu çözmek için iki iyileştirmeye sahiptir. İlk olarak, Düello ÇDQA $a_{max} = \arg\max_{a} Q_1(s', a; \theta)$ ile bir sonraki adım için bir eylem seçmek üzere bir Q_1 değerleme ağı kullanan bir çift ağ yapısı kullanır. Daha sonra a_{max}, aşırı tahmini azaltmak için bir hedef ağ Q_2 tarafından (4.22) ile değerlendirilir.

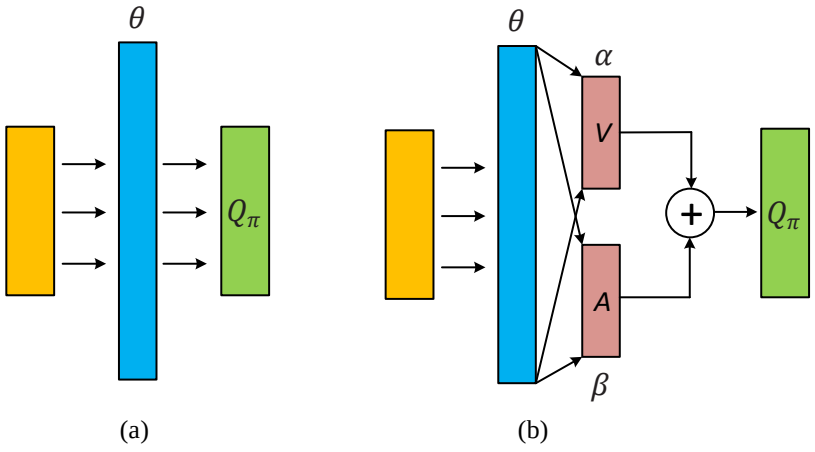
$$\begin{cases} a_{max} = \arg\max_{a} Q_1(s', a; \theta) \\ y = r + \gamma Q_2(s', a_{max}; \theta^-) \end{cases} \tag{4.22}$$

İkinci olarak, Düello ÇDQA, $Q_\pi(s, a)$ eylem-değer fonksiyonunu (4.23)'teki gibi iki kısma ayırır: Sadece durumla ilgili bir durum değeri fonksiyonu $V_\pi(s)$ ve hem durum hem de eylem ile ilgili bir avantaj fonksiyonu $A_\pi(s, a)$.

$$Q_\pi(s, a; \theta, \alpha, \beta) = V_\pi(s; \theta, \alpha) + A_\pi(s, a; \theta, \beta) \tag{4.23}$$

θ, V_π (s) ve $A_\pi(s, a)$'in ortak parametreleridir. α ve β, sırasıyla V_π (s) ve $A_\pi(s, a)$'nın özgün parametreleridir.

Şekil 4.13'te DQA ve Düello ÇDQA'nın ağ yapıları ve sözde kodu Algoritma 4.3'te verilmiştir.



Şekil 4.13. Sinir ağı yapısı: (a) DQA ve (b) Düello ÇDQA

V_π (s) ve $A_\pi(s, a)$, çıkış katmanı ile son gizli katman arasındadır ve V_π (s) ve $A_\pi(s, a)$ boyutları çıktı katmanı ile aynıdır.

Algoritma 4.3. Düello ÇDQA

M kapasiteli $\mathcal{D}$ tekrar tamponunu başlat

Rastgele θ parametreleriyle Q_1 Değerleme Ağı'nı başlat

$\theta^- \leftarrow \theta$ parametreleriyle Q_2 Hedef Ağı'nı başlat

Rastgele politika tarafından üretilen veri ile tekrar tamponunu doldur

while $True$ **do**

  Kararlı insansı yürüme sistemini yeniden başlat

  while $Sonlandırma\ yok$ **do**

    ϵ olasılıkla rastgele bir a eylemi seç, aksi halde $a = \arg\max_a Q_1(s_t, a; \theta)$ eylemi seç

    a eylemini gerçekleştir, r ödülü ve s' bir sonraki durumu al

    $\langle s, a, r, s' \rangle$ deneyim demetini $\mathcal{D}$'ye sakla

    $\mathcal{D}$'den rastgele mini yığını örnekle

    if $j + 1\ adımında\ sonlandırma\ var$ **then**

      $Y_j = r_j$

    else

      $a_{max} = \arg\max_a Q_1(s', a; \theta)$

      $Y_j = \gamma \max_{a'} Q_2(s', a_{max}; \theta^-)$

    $\hat{Y}_j = Q_1(s, a; \theta)$

    θ parametresini güncellemek üzere Q_1'i eğitmek için kaybı en aza indir

    Her C adımında hedef ağı güncelle: $\theta^- \leftarrow \theta$

  if $Q_1\ yakınsar$ **then**

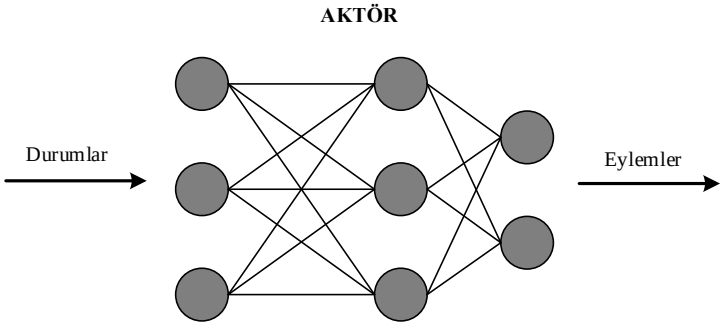
    $Q_2 = Q_1$

    $Q_{optimal} = Q_2$

    break

4.3. Politika Tabanlı Öğrenme Yöntemleri

Aktör olarak da bilinen politika tabanlı öğrenme, optimal politikayı doğrudan aramaya odaklanır. Politika fonksiyonu tabanlı öğrenme algoritmaları, durum gözlemlerini alan ve eylemleri çıkaran bir sinir ağını eğitir. Bu sinir ağı, politikanın tamamıdır. Sinir ağı aktör olarak adlandırılır, çünkü ajana hangi eylemleri yapacağını doğrudan söyler. DPÖ'deki politika tabanlı öğrenme yapısı Şekil 4.14'te verilmiştir.



Şekil 4.14. Politika tabanlı öğrenme yapısı

Politika tabanlı PÖ yöntemleri, genel olarak daha iyi yakınsama özelliklerine sahip olmasının yanında yüksek boyutlu veriler ve sürekli eylem uzayında verimli olarak kullanılabilmektedir. Ek olarak, stokastik politikaları öğrenebilmektedir.

Politika tabanlı yöntem, politikayı doğrudan optimize eder. Getiri en yükseğe çıkarılana kadar politikayı yinelemeli olarak günceller. Değer tabanlı yöntemler ile karşılaştırıldığında, politika tabanlı bir yöntem, daha basit politika parametreleştirme, daha iyi yakınsama avantajlarına sahiptir ve sürekli veya yüksek boyutlu eylem alanı için uygundur.

Bazı yaygın politika tabanlı algoritmalar arasında Politika Gradyanı (PG) [55], Güven Bölgesi Politika Optimizasyonu (GBPO) [203] ve Yakınsal Politika Optimizasyonu (YPO) [72] gibi algoritmalar yer alır. GBPO (Trust Region Policy Optimization–TRPO) adı verilen bir politika gradyan algoritması, 2015'te Schulman ve diğerleri tarafından tanıtılmıştır [203]. Bu yöntem, bir Kullback-Leibler kısaca KL sapma cezası ile yerel bir yaklaşımı tekrar tekrar optimize ederek politikaları günceller. Parametre uzayında yeni ve eski politikaların farklılığından kaynaklanan performans çöküşünü önler. [72]'de önerilen YPO (Proximal Policy Optimization–PPO), GBPO ile aynı arka plana sahiptir. Benzer faydaları paylaşırlar, ancak uygulanması daha basittir ve daha iyi örnek karmaşıklığına sahiptir. YPO'nun bu varyantı bir KL farklılığına sahip değildir. Bunun yerine, yeni politikanın eski politikadan uzaklaşmasını önlemek için amaç fonksiyonunda özel bir kırpma vardır.

4.3.1. Politika Gradyanı

Tüm DPÖ algoritmalarındaki amaç, ödülleri en üst düzeye çıkarabilmek için optimal politikayı bulmaktır. Q fonksiyonu, bir durumda gerçekleştirilecek optimum eylemin hangi eylem olduğunu ifade ettiğinden optimal politikayı bulmak için genellikle Q fonksiyonu kullanılır. Politika Gradyan (PG) yöntemlerinde, Q fonksiyonu kullanılmadan optimal politika bulunabilir.

PG [55], bazı θ parametreleri tarafından parametrelenen politikayı doğrudan optimize eden, PÖ'deki popüler algoritmalardan biridir. Q fonksiyonu olmadan optimal politikayı bulmak için ilk

olarak s durumunda verilen bir a eylemi gerçekleştirme olasılığı olarak $\pi(a|s)$ politika fonksiyonunu tanımlanır. Politika, bir durumdaki optimum eylemi belirlemeye izin veren bir θ parametresi aracılığıyla $\pi(a|s;\theta)$ olarak parametreleştirilir. Böylece PG yöntemleri, eylem alanı üzerinde ayrıklaştırmaya veya başka bir değer en yükseğe çıkarma problemini çözmeye ihtiyaç duymadıklarından sürekli eylem uzayları için daha uygundur. PG yöntemlerinin değer tabanlı yöntemlerden bir başka üstünlüğü ise PG yöntemlerinin stokastik politikaları doğal olarak modelleyebilmesidir. Ayrıca, PG yöntemleri, optimizasyona rehberlik etmek için gradyan bilgisini kullandıkları için sıklıkla daha iyi yakınsama özelliklerine sahiptir.

Optimal politikanın bulunması için politika ağı olarak adlandırılan bir sinir ağı kullanılır. Politika ağına giriş, durumdur ve o durumdaki her bir eylemin olasılığı ise çıkıştır. Bu olasılığa sahip olunduğunda, bu dağılımdan bir eylem örneklenebilir ve bu eylem bir durumda gerçekleştirilebilir. Ancak örneklenen eylem, durumda gerçekleştirilecek doğru eylem olmayabilir. Eylem gerçekleştirilir ve ödül saklanır. Benzer şekilde, dağıtımdan bir eylem örnekleyerek her durumda eylemler gerçekleştirilerek ödül saklanır ve eğitim verisi elde edilir. Bir durumda yüksek ödül getiren eylemlerin olasılığı yüksek ve düşük ödül getiren eylemlerin olasılığı düşük olacak şekilde gradyan inişi gerçekleştirilir ve gradyanlar güncellenir.

4.4. Aktör-Kritik Öğrenme Yöntemleri

Değer tabanlı ve politika tabanlı yöntemlerin birleşiminden oluşan aktör-kritik yöntemler son zamanlarda DPÖ çalışmalarında öne çıkmaktadır. Değer tabanlı yöntemler, sürekli eylem uzayları için uygun değildir. Bu nedenle, Q fonksiyonunu en yükseğe çıkarmak yerine politika açıkça tanımlanmalıdır. Bu tür problemler için aktör-kritik tipi öğrenme yöntemleri hem değer hem de politika modellerini ayrı ayrı kullanır.

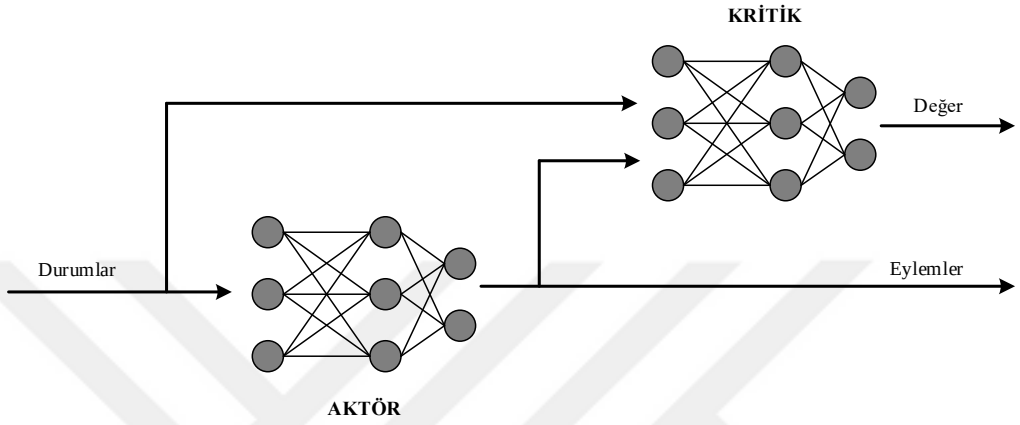
Politika, aktör olarak temsil edilirken değer fonksiyonu, kritik olarak temsil edilmektedir. Aktör, mevcut çevre durumunu gözeterek ajanın eylemini belirlemekteyken, kritik durum ve eylemleri bir değerlendirme sürecine tabi tutmaktadır.

Aktör-kritik yöntemi, örnek verimliliğini artırmak için bir Q fonksiyonunu veya değer fonksiyonunu öğrenmek için değer tabanlı yöntemleri kullanır. Politika fonksiyonunu öğrenmek için ise politika tabanlı yöntemleri kullanarak değer tabanlı yöntem ve politika tabanlı yöntemin yararlarını birleştirir. Bu tür bir yöntem, sürekli eylem alanında değer tabanlı yöntemlerin bir uzantısı olarak veya örnekleme varyansını azaltmak için politika tabanlı yöntemin bir gelişimi olarak kabul edilebilir. Bu yöntem, her iki yöntemin avantajlarını özümsese de, karşılık gelen dezavantajları da devralır. Kritiğin aşırı tahmin etme, aktörün ise yetersiz keşif sorunu vardır.

DPÖ'de aktör, mevcut durum göz önüne alındığında optimum eylem olduğunu düşündüğü şeyi yapmaya çalışan bir ağdır. Kritik, durumun değerini ve aktörün yaptığı eylemi tahmin etmeye

çalışan ikinci bir ağdır. Bu yaklaşım sürekli eylem alanları için işe yarar çünkü kritiğin yalnızca ajanın yaptığı tek bir eyleme bakması yeterlidir ve hepsini değerlendirerek optimum eylemi bulmaya çalışmasına gerek yoktur.

DPÖ'deki aktör-kritik öğrenme yapısı Şekil 4.15'te verilmiştir.



Şekil 4.15. Aktör-kritik öğrenme yapısı

Aktör ve kritik, optimal davranışı öğrenmeye çalışan sinir ağlarıdır. Aktör, hangi eylemlerin iyi ve kötü olduğunu bilmek için kritiğin geri bildirimini kullanarak doğru eylemleri öğrenir ve kritik, aktörün gerçekleştirdiği eylemi uygun şekilde eleştirebilmesi için alınan ödüllerden değer fonksiyonunu öğrenir.

Aktör, bir politika fonksiyonu algoritmasının yapacağı şekilde bir eylem seçer ve çevreye uygulanır. Kritik, mevcut durum-eylem çifti için bu eylemin değerinin ne olduğuna dair bir tahminde bulunur. Daha sonra kritik, değer tahmininin doğruluğunu belirlemek için çevreden gelen ödülü kullanır. Hata, kritik ağdan önceki durumun yeni tahmini değeri ile önceki durumun eski değeri arasındaki farktır. Yeni tahmini değer, alınan ödüle ve mevcut durumun indirimlenmiş değerine dayanır. Hata, kritiğe işlerin beklediğinden daha iyi mi yoksa daha kötü mü gittiğine dair bir fikir verir.

Kritik, bir dahaki sefere bu durumda olduğunda daha iyi bir tahmine sahip olması için bir değer fonksiyonunun yaptığı gibi kendini güncellemek için bu hatayı kullanır. Aktör aynı zamanda kritikten gelen yanıtla kendisini günceller, böylece gelecekte bu eylemi tekrar yapma olasılıklarını ayarlayabilir. Bu şekilde politika, ödülleri artık doğrudan kullanmak yerine kritiğin önerdiği yönde ödül eğimini yükseltir.

Aktör-kritik yöntemlerle, ajan, politika ve değer fonksiyonu algoritmalarının en iyi bölümlerinden yararlanabilir. Aktör-kritikler, hem sürekli durum hem de eylem alanlarını işleyebilir ve iade edilen ödül yüksek varyansa sahip olduğunda öğrenmeyi hızlandırabilir.

Deterministik Politika Gradyanı'na kısaca DPG'ye (Deterministic Policy Gradient–DPG) [204] dayanan Derin DPG (DDPG), [68]'de tanıtılmıştır. Bir Q fonksiyonunu öğrenmek için politika dışı verileri ve Bellman denklemini kullanır. Sonrasında bir politika öğrenmek için Q fonksiyonunu kullanır. O'Donoghue ve diğerleri [205], ayrıca politika gradyanını Q-öğrenme ile birleştirir. Bu birleşim, politikanın eylem tercihlerinden Q değerlerini tahmin edebilmeyi sağlar. Ardından, politikaya Q-öğrenme güncellemeleri uygulanır. Yumuşak Aktör-Kritik kısaca YAK (Soft Actor-Critic–SAC) [66], en yüksek entropi PÖ çerçevesine dayalı, politika dışı bir aktör-kritik algoritmasıdır. Aktör, politikada yüksek rastgelelikle hızlı öğrenmeyi başaracak şekilde, beklenen ödülü en yükseğe çıkarmayı ve entropiyi aynı anda yükseğe çıkarmayı amaçlar. Asenkron Avantaj Aktör-Kritik kısaca A3K (Asynchronous Advantage Actor-Critic–A3C) [206], paralel ortamlarda birden çok ajan için paralel eğitim uygulayan ve bir global değer fonksiyonunu güncelleyen bir yöntemdir. İkiz Gecikmeli DDPG [65], aşırı tahmin problemini çözmek için kırpılmış çift Q-öğrenme modu ve gecikmeli politika güncelleme stratejisi sunar.

4.4.1. Derin Deterministik Politika Gradyanı

Derin Deterministik Politika Gradyanı kısaca DDPG (Deep Deterministic Policy Gradient–DDPG) [68] algoritması, DPG [204] algoritması ve derin sinir ağlarının bir birleşimi olarak kabul edilebilir. DDPG algoritması, sürekli eylem uzayında DQA algoritmasının bir uzantısı olarak da görülebilir.

DQA'nın doğrudan uygulanamayacağı sürekli eylem alanlarıyla ilgili problemleri çözmek için kullanılır. DDPG, aynı anda bir Q fonksiyonu (kritik) ve bir politika fonksiyonu (aktör) oluşturur. Aktör-kritik mimarisi, politika gradyanı ve durum-eylem değer fonksiyonlarını birleştirir. Q fonksiyonu DQA ile aynıdır, güncellemek için ZF yöntemleri kullanılır. PG algoritması, politika fonksiyonunu Q fonksiyonundan gelen değer aracılığıyla güncellemek için kullanılır. DDPG'de, DQA'ya benzer şekilde deneyim tekrarı ve hedef ağları kullanılır.

DDPG'deki aktör ağının rolü, θ parametresini ayarlayarak durumdaki optimum eylemleri belirlemektir. DDPG'de aktör, $\mu(s)$ olarak gösterilen deterministik bir politika fonksiyonudur ve parametre θ^{μ} olarak gösterilir. Aktör ağı $\mu(s;\theta^{\mu})$ ile temsil edilir. Her adımın eylemi, bir stokastik politikadan örneklenmesi gerekmeyen $A_t = \mu(s_t|\theta^{\mu})$ ile doğrudan hesaplanır. Bir durum olarak giriş alan ve θ^{μ}'nin aktör ağ ağırlıkları olduğu eylemle sonuçlanan kritik ağı ise bir durum ve eylem olarak bir girdi alan ve θ^{Q}'nun kritik ağ ağırlıkları olduğu Q değerini döndüren $Q(s,a;\theta^{Q})$ ile temsil edilir. Kritik ağının rolü ise aktör tarafından üretilen eylemi değerlendirmektir. Benzer olarak, hedef bir ağ hem aktör ağ hem de kritik ağ olarak sırasıyla $\mu(s;\theta^{\mu'})$ ve $Q(s,a;\theta^{Q'})$ tanımlanır. $\theta^{\mu'}$ ve $\theta^{Q'}$, sırasıyla hedef aktör ve kritik ağın ağırlıklarıdır. Aktör ağ ağırlıkları,

politika gradyanlarıyla ve kritik ağ ağırlıkları ise ZF hatasından hesaplanan gradyanlarla güncellenir.

DDPG'de deterministik politika ile keşif-sömürü arasında nasıl denge kurulacağı önemlidir. Bir DPÖ uygulamasında farklı ölçeklerde farklı özellikler bulunabilmektedir. Bu yüzden, tüm özellikler aynı ölçekte olacak şekilde ölçeklendirilir. Özellikleri ölçeklendirmek için yığın normalleştirme adı verilen bir yöntem kullanılır. Bu yöntem, birim ortalama ve varyansa sahip tüm özellikleri normalleştirir. Sürekli bir çevrede, yeni eylemlerin keşfedilmesi için, aktör ağı tarafından üretilen eyleme gürültü eklenir. DDPG'de, bir gürültü sürecinden $\mathcal{N}$ örneklenen gürültüler, eğitim sırasında eylemlere eklenir. Orijinal makale [68], gürültü üretmek için Ornstein-Uhlenbeck kısaca O-U işlemini kullanır [207]. O-U keşif gürültüsü ile öğrenme sürecinde verimliliği etkileyen zamansal korelasyona sahip olunur. O-U gürültüsü MKS özelliğine uygundur.

O-U işlemi (4.24)'teki stokastik diferansiyel denklemi kullanır.

$$\mathrm{d}X_t = \theta(\pi - X_t)\mathrm{d}t + \sigma \mathrm{d}W_t \tag{4.24}$$

Burada; X_t rastgele bir değişken, $\theta > 0, x, \sigma > 0$ ise parametrelerdir. W_t, aşağıdaki özelliklere sahip bir Wiener işlemi [208] olarak adlandırılır. W_t, bağımsız artışlara sahip bir işlemdir. Bu, $T_0 < T_1 < \ldots < T_n$ zamanları için, $W_{T_0}, W_{T_1} - W_{T_0}, \ldots, W_{T_n} - W_{T_{n-1}}$ rastgele değişkenlerinin bağımsız olduğu anlamına gelir. W_t, t'ye bağlı sürekli bir fonksiyondur. Herhangi bir t ve Δt zamanı için (4.25) ile ifade edilir.

$$W(t + \Delta_t) - W(t) \sim \mathcal{N}(0, \sigma_W^2 \Delta t) \tag{4.25}$$

Aktör ağı tarafından üretilen eyleme bir $\mathcal{N}$ keşif gürültüsü ekleyerek $A_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ ile bir eylem seçilir. Bu eylem, bir s durumunda gerçekleştirilir, bir r ödülü alınır ve yeni bir s' durumuna geçilir. Bu geçiş bilgileri, bir deneyim tekrar tamponunda saklanır.

Bazı yinelemelerden sonra, tekrar tamponundan geçişler örneklenir ve ağ eğitilir. Ardından hedef Q değeri, (4.26) ile hesaplanır.

$$y_i = r_i + \gamma Q'\left(s_{t+1}, \mu'\left(s_{t+1}\middle|\theta^{\mu'}\right)\middle|\theta^{Q'}\right) \tag{4.26}$$

ZF hatası, (4.27) ile hesaplanır.

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q)^2) \tag{4.27}$$

Burada; N eğitim için kullanılan tekrar tamponundan alınan örneklerin sayısıdır. L kaybı ile hesaplanan gradyanlarla kritik ağ ağırlıkları güncellenir.

Benzer şekilde, politika gradyanı kullanılarak aktör ağ ağırlıkları güncellenir. μ politikası, J başlangıç dağılımından beklenen getiriye zincir kuralı uygulanarak (4.28) ile güncellenir.

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q\left(s_i, \mu(s_i)\right) \nabla_{\theta^\mu} \mu(s_i|\theta^\mu) \tag{4.28}$$

Ardından hedef ağdaki aktör ve kritik ağının ağırlıkları güncellenir. DDPG, DQA gibi hedef ağın benzer bir yolunu benimsemesine rağmen ağ parametrelerini doğrudan kopyalamak yerine üstel düzgünleştirme yoluyla (4.29) ve (4.30)'daki gibi günceller.

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \tag{4.29}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \tag{4.30}$$

Burada; τ yumuşak güncelleme faktörüdür. $\tau \ll 1$ olduğundan hedef ağ, çok yavaş ve düzgün bir şekilde güncellenir. Bu da yumuşak yer değiştirme olarak adlandırılır ve daha fazla öğrenme kararlılığı sağlar.

DDPG algoritmasının sözde kodu Algoritma 4.4'te verilmiştir.

Algoritma 4.4. DDPG

Hiperparametreler: τ yumuşak güncelleme faktörü, γ ödül indirim faktörü, $\mathcal{N}$ keşif gürültüsü

Giriş: Boş $\mathcal{D}$ tekrar tamponu, $Q(s,a|\theta^Q)$ kritik ağının θ^Q parametreleri ve $\mu(s|\theta^\mu)$ aktör ağının θ^μ parametreleri, Q' hedef ağı ve μ'

$\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$ ağırlıklarıyla Q' hedef ağını ve μ' başlat

for $bölüm = 1, M$ **do**

Eylem keşfi için rastgele bir $\mathcal{N}$ süreci başlat

S_1 başlangıç gözlem durumunu al

for t = 1, T **do**

$a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ eylemini seç

a_t eylemini yürüt ve r_t ödülünü ve s_{t+1} yeni durumunu al

$e_t = (s_t, a_t, r_t, s_{t+1})$ deneyim demetini $\mathcal{D}$'ye sakla

$\mathcal{D}$'den $\mathcal{D}_r$ $(\mathcal{D}_r| = N)$ rastgele yığını örnekle

$$Y_j = \begin{cases} r_j, & eğer\ s_{j+1}\ terminal\ ise \\ r_j + \gamma Q\left(s_{j+1}, \mu\left(s_{j+1}; \theta^{\mu^-}\right); \theta^{Q^-}\right), & aksi\ taktirde \end{cases} \quad \forall e_j \in \mathcal{D}_r \text{ ayarla}$$

Kaybı minimize ederek kritiği güncelle:

$L = \frac{1}{N} \sum_{e_j \in \mathcal{D}_r} \left[Y_j - Q\left(s_j, a_j; \theta^Q\right)\right]^2$

Örneklenen politika gradyanını kullanarak aktör politikasını güncelle:

$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_j \nabla_a Q\left(s_j, \mu\left(s_j\right)\right) \nabla_{\theta^\mu} \mu\left(s_j|\theta^\mu\right)$

Hedef ağları güncelle:

$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$

$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$

end for

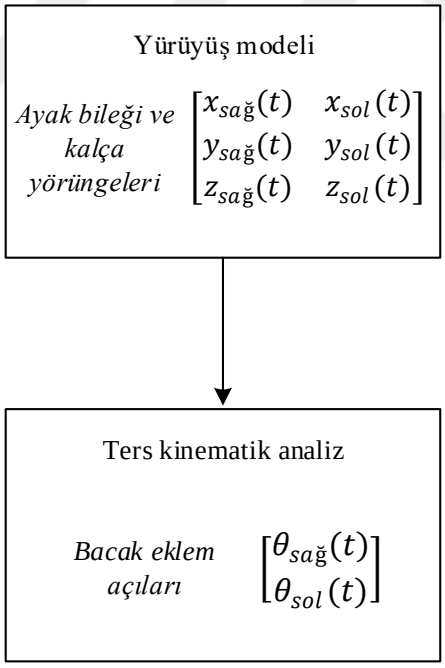
end for

---

5. İNSANSI ROBOT İÇİN YÜRÜYÜŞ PLANLAMASI

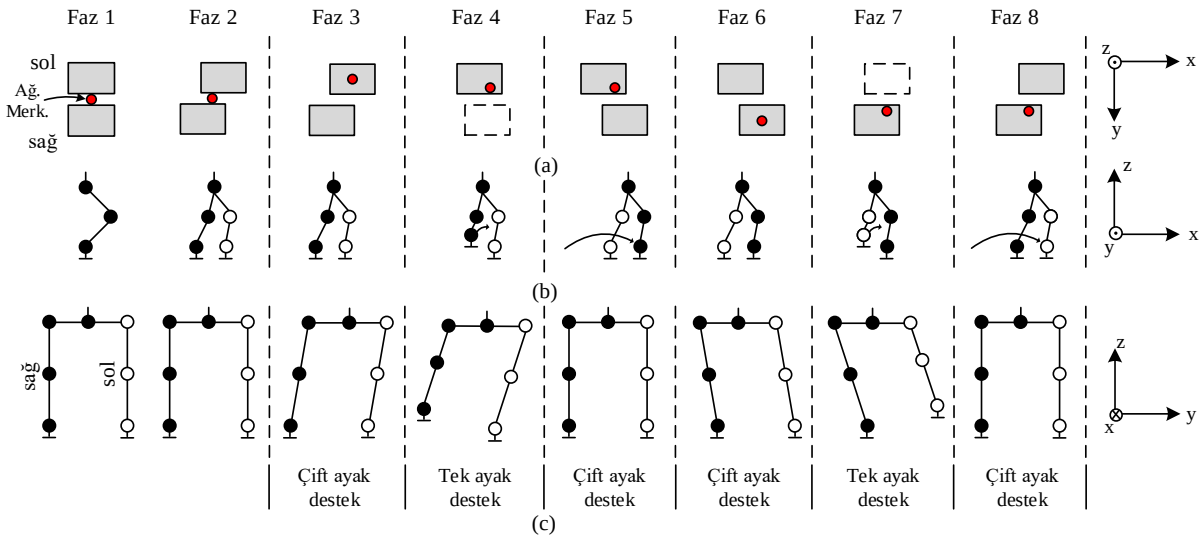
İki ayaklı robotların geliştirilmesi, insana yardım sağlama kapasiteleri ve esneklikleri nedeniyle son yıllarda araştırmaların ilgisini çok çekmiştir. Birçok çalışma insanın kinematik verilerine atıfta bulunarak robotun yürüme modelini araştırdığı, bazılarının ise eylemsizlik ve yerçekiminin pasif etkileşimini kullanan bir yürüme modeli tanımladığı yürüme modellerinin oluşturulması üzerine odaklanmıştır [209].

İki ayaklı robotun farklı koşullardaki arazilerde dengeli bir şekilde yürüyebilmesi için az sayıda parametreyi değiştirerek araziye uyum sağlayabilecek modeller geliştirmek gerekir. Bu tez kapsamında, Şekil 5.1'deki akış diyagramında verildiği gibi Robotis-OP2 insansı robotu için bir yürüyüş şablonu üreteci oluşturulmuştur. Yürüyüş şablonu için farklı zemin koşullarındaki araziyi hesaba katabilecek değişkenlerle tasarlanan bir yöntem benimsenmiştir. Yürüyüş şablonu ile ayak bileği ve kalça yörüngeleri belirlenir.



Şekil 5.1. Yürüyüş şablonu üreteci

Robotun temel yürüme işleminin gerçekleştirilebilmesi için Şekil 5.2'de gösterildiği gibi sekiz fazlı bir yürüme çevrimi planlanmıştır. İlk iki faz, yürüme başlangıç fazları olup yörünge planlaması için gereken hazırlıkları içerir.



Şekil 5.2. Sekiz fazı gösteren tam bir yürüme süreci: (a) transversal düzlem, (b) sagital düzlem ve (c) ön düzlem

Faz 1: Robotun yörünge koordinatlarını tanımlamak için kullanılan başlangıç fazıdır.

Faz 2: Robotun yürümeye geçmeden önceki hazırlık fazıdır. Bu fazda, sol ayağın belli bir miktar ileriye, sağ ayağın ise belli bir miktar geriye alınmasıyla robot yürümeye hazır hale getirilir.

Faz 3: Robotun yürümeye başladığı fazdır. Robot, ağırlık merkezini sol ayağa hareket ettirir. Bu esnada ayaklar yere sabit olup sadece kalça eklemleri sola kaydırılır.

Faz 4: Ağırlık merkezi sol ayağa taşındığında sağ ayak adım atmaya başlar.

Faz 5: Sağ ayak adım atmayı bitirip destek alanına geri döner ve ağırlık merkezi sağ ayağa taşınmaya hazır hale getirilir.

Faz 6: Robot ağırlık merkezini sağ ayağa hareket ettirir. Bu esnada ayaklar yere sabit olup sadece kalça eklemleri sağa kaydırılır.

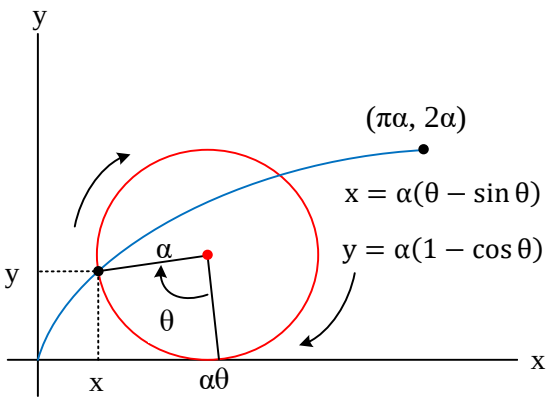
Faz 7: Ağırlık merkezi sağ ayağa taşındığında sol ayak adım atmaya başlar.

Faz 8: Sol ayak adım atmayı bitirip destek alanına geri döner.

Bu sekiz faz göz önüne alındığında, Faz 3 ile Faz 8 arasındaki altı faz sürekli tekrarlanarak sürekli yürüme süreci elde edilir. Bu süreç için yürüme yörüngesi planlanmıştır.

5.1. Yürüme Yörüngesi

Yürüme modelinde ayak bileği ve kalça yörüngelerini oluşturmak için sikloid eğrisi fonksiyonlarından yararlanılmıştır. Şekil 5.3'te gösterilen sikloid eğrisi, kaymadan düz bir çizgi boyunca yuvarlanarak ilerleyen bir daire üzerindeki sabit bir nokta tarafından izlenen yol olarak ifade edilir.



Şekil 5.3. Yuvarlanan bir daire tarafından üretilen sikloid eğrisi

Yürüme modeli, ayak bileği için x-z ve kalça için x-y düzlemleri olan iki dik alanda ayrı ayrı oluşturulur. Sağ ve sol ayak bileği yörünge planlamasının başlangıç noktası olup kalça eklemleri ise yörünge planlamasının bitiş noktasıdır.

5.1.1. Ayak Bileği Yörüngesi

x ileri-geri yön ve z yukarı-aşağı yön iken, x-z düzlemindeki sağ ayak bileği ekleminin salınım yörüngesi Faz 3, Faz 4 ve Faz 5 için sırasıyla (5.1), (5.2) ve (5.3)'te verilmiştir. Sol ayak bileği eklemi için bu üç fazı içeren salınım yörüngesi ise (5.4)'te verilmiştir.

$$\begin{cases} x_{sağ\_ab}(t) = 0 \\ y_{sağ\_ab}(t) = 0 \\ z_{sağ\_ab}(t) = 0 \end{cases}, 0 \le t \le \frac{t_d}{2} \tag{5.1}$$

$$\begin{cases} x_{sağ\_ab}(t) = \frac{s}{\pi}\left(4\pi\frac{t-\frac{t_d}{2}}{T-2t_d} - \sin\left(4\pi\frac{t-\frac{t_d}{2}}{T-2t_d}\right)\right) \\ y_{sağ\_ab}(t) = 0 \\ z_{sağ\_ab}(t) = \frac{h}{2}\left(1 - \cos\left(4\pi\frac{t-\frac{t_d}{2}}{T-2t_d}\right)\right) \end{cases}, \frac{t_d}{2} < t \le \frac{T-t_d}{2} \tag{5.2}$$

$$\begin{cases} x_{sağ\_ab}(t) = 2s \\ y_{sağ\_ab}(t) = 0 \\ z_{sağ\_ab}(t) = 0 \end{cases}, \frac{T-t_d}{2} < t \le \frac{T}{2} \tag{5.3}$$

$$\begin{cases} x_{sol\_ab}(t) = s \\ y_{sol\_ab}(t) = 0 \\ z_{sol\_ab}(t) = 0 \end{cases}, 0 \le t \le \frac{T}{2} \tag{5.4}$$

t: Anlık zaman (s),

t_d: Faz 5 ile Faz 6 ve Faz 8 ile Faz 3'teki çift ayak destek durumunda ağırlık merkezinin kayma zamanı (s),

s: Bir adım uzunluğu (mm),

h: En büyük ayak kaldırma yüksekliği (mm),

T: Yürüme çevrimi (Faz 3-Faz 8) zamanı (s),

$x_{sağ\_ab}$: Sağ ayak bileği ekleminin x ekseni yörüngesi (mm),

$y_{sağ\_ab}$: Sağ ayak bileği ekleminin y ekseni yörüngesi (mm),

$z_{sağ\_ab}$: Sağ ayak bileği ekleminin z ekseni yörüngesi (mm),

x_{sol_ab}: Sol ayak bileği ekleminin x ekseni yörüngesi (mm),

y_{sol_ab}: Sol ayak bileği ekleminin y ekseni yörüngesi (mm),

z_{sol_ab}: Sol ayak bileği ekleminin z ekseni yörüngesi (mm).

t_d, (5.5) ile hesaplanmaktadır.

$$t_d = \frac{T}{2} \times DSR \tag{5.5}$$

DSR: Çift ayak destek oranı.

Sol ayak bileği ekleminin salınım yörüngesi Faz 6, Faz 7 ve Faz 8 için sırasıyla (5.6), (5.7) ve (5.8)'de verilmiştir. Sağ ayak bileği eklemi için bu üç fazı içeren salınım yörüngesi ise (5.9)'da verilmiştir.

$$\begin{cases} x_{sol\_ab}(t) = s \\ y_{sol\_ab}(t) = 0 \\ z_{sol\_ab}(t) = 0 \end{cases}, \frac{T}{2} < t \leq \frac{T+t_d}{2} \tag{5.6}$$

$$\begin{cases} x_{sol\_ab}(t) = \frac{s}{\pi}\left(4\pi\frac{t-\frac{T}{2}-\frac{t_d}{2}}{T-2t_d} - \sin\left(4\pi\frac{t-\frac{T}{2}-\frac{t_d}{2}}{T-2t_d}\right)\right) + s \\ y_{sol\_ab}(t) = 0 \\ z_{sol\_ab}(t) = \frac{h}{2}\left(1 - \cos\left(4\pi\frac{t-\frac{T}{2}-\frac{t_d}{2}}{T-2t_d}\right)\right) \end{cases}, \frac{T+t_d}{2} < t \leq T - \frac{t_d}{2} \tag{5.7}$$

$$\begin{cases} x_{sol\_ab}(t) = 3s \\ y_{sol\_ab}(t) = 0 \\ z_{sol\_ab}(t) = 0 \end{cases}, T - \frac{t_d}{2} < t \leq T \tag{5.8}$$

$$\begin{cases} x_{sağ\_ab}(t) = 2s \\ y_{sağ\_ab}(t) = 0 \\ z_{sağ\_ab}(t) = 0 \end{cases}, \frac{T}{2} < t \leq T \tag{5.9}$$

5.1.2. Kalça Yörüngesi

x ileri-geri yön ve y sağ-sol yön iken, x-y düzlemindeki sağ ve sol kalça eklemlerinin salınım yörüngesi Faz 3, Faz 4 ve Faz 5 için sırasıyla (5.10) ve (5.11)'de verilmiştir.

$$\begin{cases} x_{sağ\_ka}(t) = \frac{s}{2\pi}\left(4\pi\frac{t}{T} - \sin\left(4\pi\frac{t}{T}\right)\right) + \frac{s}{2} \\ y_{sağ\_ka}(t) = -\frac{w}{2}\left(1 - \cos 4\pi\frac{t}{T}\right) \\ z_{sağ\_ka}(t) = h_{ak} - h_b \end{cases}, 0 \le t \le \frac{T}{2} \tag{5.10}$$

$$\begin{cases} x_{sol\_ka}(t) = \frac{s}{2\pi}\left(4\pi\frac{t}{T} - \sin\left(4\pi\frac{t}{T}\right)\right) + \frac{s}{2} \\ y_{sol\_ka}(t) = -\frac{w}{2}\left(1 - \cos 4\pi\frac{t}{T}\right) \\ z_{sol\_ka}(t) = h_{ak} - h_b \end{cases}, 0 \le t \le \frac{T}{2} \tag{5.11}$$

Sağ ve sol kalça eklemlerinin salınım yörüngesi Faz 6, Faz 7 ve Faz 8 için sırasıyla (5.12) ve (5.13)'te verilmiştir.

$$\begin{cases} x_{sağ\_ka}(t) = \frac{s}{2\pi}\left(4\pi\frac{t-\frac{T}{2}}{T} - \sin\left(4\pi\frac{t-\frac{T}{2}}{T}\right)\right) + \frac{3s}{2} \\ y_{sağ\_ka}(t) = \frac{w}{2}\left(1 - \cos 4\pi\frac{t-\frac{T}{2}}{T}\right) \\ z_{sağ\_ka}(t) = h_{ak} - h_b \end{cases}, \frac{T}{2} < t \le T \tag{5.12}$$

$$\begin{cases} x_{sol\_ka}(t) = \frac{s}{2\pi}\left(4\pi\frac{t-\frac{T}{2}}{T} - \sin\left(4\pi\frac{t-\frac{T}{2}}{T}\right)\right) + \frac{3s}{2} \\ y_{sol\_ka}(t) = \frac{w}{2}\left(1 - \cos 4\pi\frac{t-\frac{T}{2}}{T}\right) \\ z_{sol\_ka}(t) = h_{ak} - h_b \end{cases}, \frac{T}{2} < t \le T \tag{5.13}$$

h_{ak}: Ayak bileği eklemi ile kalça eklemi arasındaki yükseklik (mm),

h_b: Kalça ekleminin yerden yüksekliğini ayarlamada kullanılan bükülme yüksekliği (mm),

w: Kalça salındığında sağa-sola en büyük öteleme (mm),

$x_{sağ\_ka}$: Sağ kalça ekleminin x ekseni yörüngesi (mm),

$y_{sağ\_ka}$: Sağ kalça ekleminin y ekseni yörüngesi (mm),

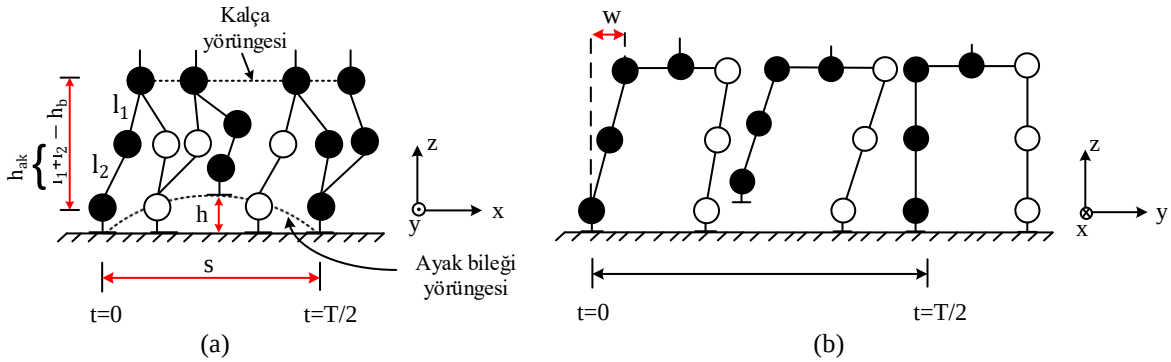
$z_{sağ\_ka}$: Sağ kalça ekleminin z ekseni yörüngesi (mm),

x_{sol_ka}: Sol kalça ekleminin x ekseni yörüngesi (mm),

y_{sol_ka}: Sol kalça ekleminin y ekseni yörüngesi (mm),

z_{sol_ka}: Sol kalça ekleminin z ekseni yörüngesi (mm).

Yarım yürüme çevrimi (Faz 3, Faz 4 ve Faz 5) için yürüme parametreleri Şekil 5.4'te gösterilmiştir.



Şekil 5.4. Yarım yürüme çevriminde yürüme parametrelerinin gösterimi: (a) sagital düzlem ve (b) ön düzlem

Yürüme yörüngesi için oluşturulan eşitliklerden ve Şekil 5.4'ten görülebileceği üzere yürüme şablonu, üç uzunluk parametresi (s, h, w), bir zaman parametresi (T) ve bir oran parametresi (DSR) değiştirilerek ayarlanabilir.

5.2. Kinematik Analiz

Robotis-OP2 insansı robotunun yürüyüş yörünge planlaması tamamlandıktan sonra iki ayaklı hareketi uygulamak için bacaklardaki eklemlerin açısının elde edilmesi gerekmektedir. Bacaklardaki eklemlerin alması gereken açılar, ters kinematik analiz ile elde edilmiştir. Robotis-OP2'nin bacak düz kinematik analizi gerçekleştirildikten sonra ters kinematik analize geçilmiştir.

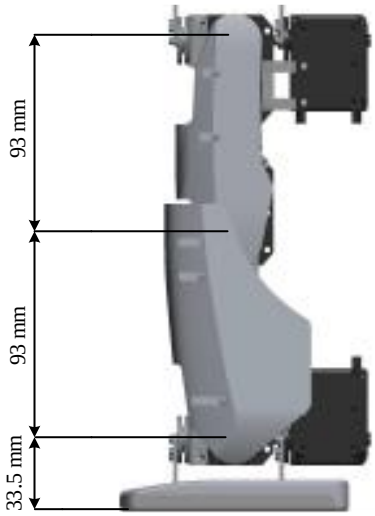
5.2.1. Düz Kinematik Analiz

Düz kinematik analiz, bacağın her bir ekleminin belirlenen açılarda hareketi sonucunda, belirlenmiş uç noktanın ulaştığı konumun koordinatlarını elde etmek için yapılmaktadır.

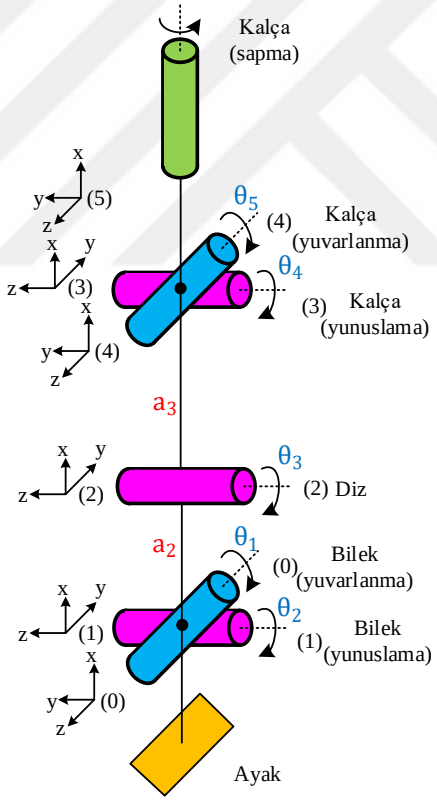
Tez kapsamında, ayak bileği eklemi başlangıç noktası, kalça eklemi ise bitiş noktası olarak tanımlanmıştır. İki ayaklı robotun sağ ve sol bacakları birbirinden bağımsız sistemler olarak kabul edilmiştir.

Düz kinematik analiz için ilk olarak robot bacağının dönme eksenleri belirlenip eklemlerine eksen takımları yerleştirilmiştir. Robotun kalçasındaki sapma hareketini sağlayan eklem, düz bir yürüme görevi için yürüme yörüngesini etkilemediğinden kinematik analize dahil edilmemiştir. Bu sayede, kinematik analizin daha karmaşık olmasının önüne geçilmiştir. Ayrıca, robotun sağ ve sol bacağı birbiriyle simetrik olduğundan bir bacak için gerçekleştirilen analiz diğeri için de geçerlidir.

Bacak uzuv uzunlukları Şekil 5.5'te verilen Robotis-OP2 insansı robotun eksen takımlarının yerleştirilme işlemi de Şekil 5.6'da gösterilmiştir.

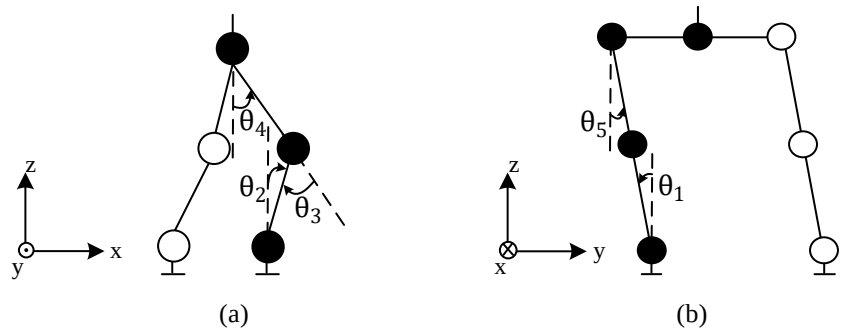


Şekil 5.5. Robotun bacak uzuvlarının uzunlukları



Şekil 5.6. Robotun bacak eklemlerine eksen takımlarının yerleştirilmesi

Şekil 5.7'de bacak eklem açıları, sagital ve ön düzlemde gösterilmiştir. Şekil 5.7(a)'da robota yunuslama hareketi sağlayan üç açı, Şekil 5.7(b)'de ise yuvarlanma hareketi sağlayan iki açı görülmektedir.



(a) (b)

Şekil 5.7. Robotun bacak eklem açılarının gösterimi: (a) sagital düzlem ve (b) ön düzlem

Şekil 5.6'da görüldüğü gibi eksen takımları yerleştirilirken z ekseni ileri-geri yön, x ekseni yukarı-aşağı yön olarak yerleştirilmiştir. Yürüme yörüngesi oluşturulurken ise Şekil 5.7'deki gibi x ekseni ileri-geri yön, z ekseni yukarı-aşağı yön olarak belirlenmiştir. Bu nedenle, kinematik hesaplamalardaki p_x ve p_z ifadeleri birbirleriyle yer değiştirilerek kullanılmıştır.

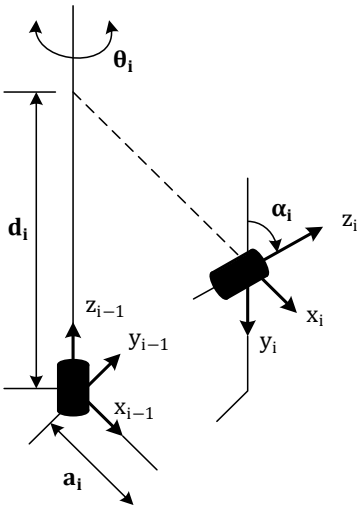
Düz kinematik analizde homojen dönüşüm matrisi olarak ifade edilen A, bir uzuv ile bir sonraki uzuv arasındaki bağıl öteleme ve dönmeyi gösterir. Eklemlerin konumu ise temel dönüşüm matrisini temsil eden T ile belirlenir. Düz kinematik analiz sonucunda elde edilecek konum koordinatları için öncelikle Denavit-Hartenberg kısaca D-H yöntemi kullanılır.

Denavit-Hartenberg (D-H) Yöntemi

Denavit-Hartenberg (D-H) yönteminde, kinematik analiz için dört temel parametre kullanılır. Robotun döner veya prizmatik eklemleri bu dört parametre ile belirlenir:

1) θ_i: x_{i-1} ekseninden x_i eksenine geçişte z_{i-1} ekseni etrafındaki dönme açısını,
2) d_i: (i-1). koordinat sisteminin orijininden x_i ile z_{i-1} ekseninin kesim noktası arasındaki z_{i-1} ekseni doğrultusundaki ötelemeyi,
3) a_i: (i-1). koordinat sisteminin orijininden z_{i-1} ile x_i ekseninin kesim noktası arasındaki x_i ekseni doğrultusundaki ötelemeyi,
4) α_i: z_{i-1} ekseni ile z_i ekseni arasındaki açı.

D-H parametrelerinin kinematik bir çift üzerinde nasıl belirleneceği Şekil 5.8'de gösterilmiştir.



Şekil 5.8. D-H parametrelerinin kinematik bir çift üzerinde belirlenmesi

Eksen takımları ve uzuv uzunlukları kullanılarak Robotis-OP2 robotunun bacağı için oluşturulan D-H tablosu Tablo 5.1'de verilmiştir.

Tablo 5.1. Robotis-OP2 insansı robotunun bacak D-H tablosu

Eksen	θ (rad)	d (mm)	a (mm)	α (rad)
1	θ_1	0	0	$-\pi/2$
2	θ_2	0	93	0
3	θ_3	0	93	0
4	θ_4	0	0	$\pi/2$
5	θ_5	0	0	0

Homojen Dönüşüm Matrislerinin Bulunması

Robot üzerindeki her bir uzvun uzaydaki yönelimini ve konumunu temsil eden dönme matrisi ve öteleme vektörü bulunmaktadır. Eklem sayısı kadar bulunan homojen dönüşüm matrisi (A), (5.14)'te gösterildiği gibi dönme matrisi ve öteleme vektörünü birleştirmek için kullanılmaktadır.

$$A_i = \mathrm{Rot}(z, \theta_i) \times \mathrm{Trans}(0,0, d_i) \times \mathrm{Trans}(a_i, 0,0) \times \mathrm{Rot}(x, \alpha_i) \quad (5.14)$$

$\mathrm{Rot}(z, \theta_i)$: eklemin z ekseni etrafındaki dönme açısını belirten matris,

$\mathrm{Trans}(0,0, d_i)$: eklemin z ekseni doğrultusundaki ötelemesini belirten matris,

$\mathrm{Trans}(a_i, 0,0)$: eklemin x ekseni doğrultusundaki ötelemesini belirten matris,

$\mathrm{Rot}(x, \alpha_i)$: eklemler arasındaki z eksenlerinin birbirine olan açı izdüşümlerini belirten matris.

(5.14)'te belirtilen matrisler (5.15)'teki gibi yerleştirilerek homojen dönüşüm matrisi oluşturulur. Bu matris, matrislerde çarpma kuralı gereği en sağdan başlayıp sola doğru (5.16) ve (5.17)'deki gibi sırasıyla çarpılarak i. eklemin $(i-1)$. ekleme göre yönelimini ve konumunu ifade eden (5.18)'deki tek bir matris haline getirilir.

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.15}$$

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.16}$$

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.17}$$

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.18}$$

Tablo 5.1'de verilen D-H tablosundaki parametreler kullanılarak bir eklemin bir önceki ekleme göre yönelimini ve konumunu belirleyen bacağa ait homojen dönüşüm matrisleri elde edilmiştir. Sırasıyla bilek (yuvarlanma), bilek (yunuslama), diz, kalça (yunuslama) ve kalça (yuvarlanma) eklemleri için elde edilen homojen dönüşüm matrisleri (5.19), (5.20), (5.21), (5.22) ve (5.23)'te verilmiştir.

$$A_1 = \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 0 \\ \sin\theta_1 & 0 & \cos\theta_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.19}$$

$$A_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 93\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & 93\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.20}$$

$$A_3 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 93\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & 93\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.21}$$

$$A_4 = \begin{bmatrix} \cos\theta_4 & 0 & \sin\theta_4 & 0 \\ \sin\theta_4 & 0 & -\cos\theta_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.22)$$

$$A_5 = \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ \sin\theta_5 & \cos\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.23)$$

Temel Dönüşüm Matrisinin Bulunması

Her bir eklem için homojen dönüşüm matrisi elde edilir ve daha sonra bu matrisler (5.24)'teki gibi çarpılarak temel dönüşüm matrisi elde edilir.

$$T_0^N = A_1 A_2 \dots A_N \quad (5.24)$$

Burada; N, eklem sayısını ifade etmektedir.

Robotis-OP2 kalça ekleminin ana koordinat sisteminin bulunduğu ayak bileği eklemine göre temel dönüşüm matrisi (5.25)'teki gibi elde edilmiştir.

$$T_0^5 = \ T_0^1 \ T_1^2 \ T_2^3 \ T_3^4 \ T_4^5 = A_1 A_2 A_3 A_4 A_5 \quad (5.25)$$

(5.26)'da verilen T_0^5 matrisinde, $(n, o, a)_{x,y,z}$, bir koordinat sisteminin başka bir koordinat sistemine göre dönme miktarını belirten kalça ekleminin yönelimini ifade eder. p_x, p_y ve p_z ise kalça ekleminin ayak bileği eklemine göre sırasıyla x, y ve z konum koordinatlarını temsil eder.

$$T_0^5 = \begin{bmatrix} n_x & o_x & a_x & \mathbf{p_x} \\ n_y & o_y & a_y & \mathbf{p_y} \\ n_z & o_z & a_z & \mathbf{p_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.26)$$

Daha önce elde edilen homojen dönüşüm matrisleri ve aşağıdaki trigonometrik kısaltmalar kullanılarak (5.27) elde edilmiştir.

$$s_1 = \sin\theta_1, s_2 = \sin\theta_2, s_3 = \sin\theta_3, s_4 = \sin\theta_4, s_5 = \sin\theta_5$$

$$c_1 = \cos\theta_1, c_2 = \cos\theta_2, c_3 = \cos\theta_3, c_4 = \cos\theta_4, c_5 = \cos\theta_5$$

$$T_0^5 = \begin{bmatrix} -s_1 s_5 - c_1(c_2(c_5 s_3 s_4 - c_3 c_4 c_5) + s_2(c_3 c_5 s_4 + c_4 c_5 s_3)) & c_1\big(c_2(s_3 s_4 s_5 - c_3 c_4 s_5) + s_2(c_3 s_4 s_5 + c_4 s_3 s_5)\big) - c_5 s_1 \\ c_1 s_5 - s_1(c_2(c_5 s_3 s_4 - c_3 c_4 c_5) + s_2(c_3 c_5 s_4 + c_4 c_5 s_3)) & c_1 c_5 + s_1(c_2(s_3 s_4 s_5 - c_3 c_4 s_5) + s_2(c_3 s_4 s_5 + c_4 s_3 s_5)) \\ s_2(c_5 s_3 s_4 - c_3 c_4 c_5) - c_2(c_3 c_5 s_4 + c_4 c_5 s_3) & c_2(c_3 s_4 s_5 + c_4 s_3 s_5) - s_2(s_3 s_4 s_5 - c_3 c_4 s_5) \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} c_1\big(c_2(c_3 s_4 + c_4 s_3) + s_2(c_3 c_4 - s_3 s_4)\big) & \mathbf{c_1(93c_2 + 93c_2 c_3 - 93s_2 s_3)} \\ s_1\big(c_2(c_3 s_4 + c_4 s_3) + s_2(c_3 c_4 - s_3 s_4)\big) & \mathbf{s_1(93c_2 + 93c_2 c_3 - 93s_2 s_3)} \\ c_2(c_3 c_4 - s_3 s_4) - s_2(c_3 s_4 + c_4 s_3) & \mathbf{-93s_2 - 93c_2 s_3 - 93c_3 s_2} \\ 0 & 1 \end{bmatrix} \quad (5.27)$$

(5.26) göz önüne alınarak p_x, p_y ve p_z sırasıyla (5.28), (5.29) ve (5.30)'daki gibi elde edilmiş ve düz kinematik analiz işlemi tamamlanmıştır.

$$p_x = \cos\theta_1 \left(93\cos\theta_2 + 93\cos\theta_2\cos\theta_3 - 93\sin\theta_2\sin\theta_3\right) \tag{5.28}$$

$$p_y = \sin\theta_1 \left(93\cos\theta_2 + 93\cos\theta_2\cos\theta_3 - 93\sin\theta_2\sin\theta_3\right) \tag{5.29}$$

$$p_z = -93\sin\theta_2 - 93\cos\theta_2\sin\theta_3 - 93\cos\theta_3\sin\theta_2 \tag{5.30}$$

5.2.2. Ters Kinematik Analiz

Ters kinematik analiz, düz kinematik analizin tersi olarak yapılmaktadır. Ters kinematik analizde, temel dönüşüm matrisindeki yönelim ve konum parametreleri kullanılarak istenen hareket için gerekli olan eklem dönme açıları (θ_i) elde edilmektedir. Ters kinematik analiz, doğrusal olmayan denklemlerin çözümünü içerdiğinden düz kinematik analize göre daha karmaşık hesaplamalar içermektedir. Robotun serbestlik derecesi ve döner eklem sayısı arttıkça denklemlerin çözüm zorluğu artmaktadır.

Tez kapsamında, Robotis-OP2 robotunun kalça ekleminin ayak bileğine göre istenen koordinat değerlerine ulaşabilmesi için eklemlerinin bulunması gereken açı değerleri ters kinematik analiz ile elde edilen denklemlerin çözümü ile belirlenmektedir.

Robotis-OP2'nin bacak yapısı incelendiğinde, ilk üç eksen bitiş noktası olan kalça ekleminin yer değiştirmesini sağlarken son iki eksen ise kalça ekleminin dönüş durumunu değiştirir. Ters kinematik analiz ile elde edilecek eşitliklerin birden fazla çözüm kümesi olabileceğinden Tablo 5.2'deki Robotis-OP2 bacaklarının eklem açı sınırları [158] dikkate alınmalıdır. Eşitlikler sonucunda elde edilecek olan eklem açı değerlerini tek çözüm kümesine indirebilmek için Tablo 5.3'teki açı sınırları kullanılmıştır.

Tablo 5.2. Robotis-OP2 bacaklarının eklem açı sınırları [158]

Eklem	**Eklem açısı**	**Eklem açı sınırları** (derece)			
		Sağ bacak		**Sol bacak**	
		min.	*maks.*	*min.*	*maks.*
Ayak bileği (yuvarlanma)	θ_1	-39	60	-58	34
Ayak bileği (yunuslama)	θ_2	-71	78	-79	70
Diz	θ_3	0	128	-129	0
Kalça (yunuslama)	θ_4	-101	25	-28	96
Kalça (yuvarlanma)	θ_5	-57	58	-57	53

Tablo 5.3. Ters kinematik analiz için kullanılan eklem açı sınırları

Eklem	**Eklem açısı**	**Eklem açı sınırları** (derece)	
		min.	*maks.*
Ayak bileği (yuvarlanma)	θ_1	-58	34
Ayak bileği (yunuslama)	θ_2	-79	70
Diz	θ_3	0	128
Kalça (yunuslama)	θ_4	-96	25
Kalça (yuvarlanma)	θ_5	-57	53

Ters kinematik analiz için öncelikle (5.25)'teki eşitliğin her iki tarafı ${A_1}^{-1}$ ile çarpılarak (5.31) elde edilmiştir. Temel dönüşüm matrisinin elemanlarını ifade eden matris, (5.32)'deki gibi yerine konularak eşitlik sadeleştirilmiştir.

$${A_1}^{-1}T_0^5 = {A_1}^{-1}A_1A_2A_3A_4A_5 \tag{5.31}$$

$${A_1}^{-1}\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_2A_3A_4A_5 \tag{5.32}$$

Düz kinematik analiz ile elde edilen homojen dönüşüm matrisleri ve aşağıdaki trigonometrik kısaltmalar kullanılarak (5.32) gerçekleştirilmiş ve (5.33) elde edilmiştir.

$$s_{234} = \sin(\theta_2+\theta_3+\theta_4)\ , c_{234} = \cos(\theta_2+\theta_3+\theta_4)\ , s_{23} = \sin(\theta_2+\theta_3)\ , c_{23} = \cos(\theta_2+\theta_3)$$

$$s_1 = \sin\theta_1\ , c_1 = \cos\theta_1\ , s_2 = \sin\theta_2\ , c_2 = \cos\theta_2\ ,\ s_5 = \sin\theta_5\ , c_5 = \cos\theta_5$$

$$\begin{bmatrix} n_xc_1 + n_ys_1 & o_xc_1 + o_ys_1 & a_xc_1 + a_ys_1 & \mathbf{p_xc_1 + p_ys_1} \\ -n_z & -o_z & -a_z & \mathbf{-p_z} \\ n_yc_1 - n_xs_1 & o_yc_1 - o_xs_1 & a_yc_1 - a_xs_1 & \mathbf{p_yc_1 - p_xs_1} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{234}c_5 & -c_{234}s_5 & s_{234} & \mathbf{93c_{23} + 93c_2} \\ s_{234}c_5 & -s_{234}s_5 & -c_{234} & \mathbf{93s_{23} + 93s_2} \\ s_5 & c_5 & 0 & \mathbf{0} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.33}$$

(5.33) eşitliğinden (5.34), (5.35) ve (5.36) eşitlikleri elde edilmiştir.

$$p_x\cos\theta_1 + p_y\sin\theta_1 = 93\cos(\theta_2+\theta_3) + 93\cos\theta_2 \tag{5.34}$$

$$-p_z = 93\sin(\theta_2+\theta_3) + 93\sin\theta_2 \tag{5.35}$$

$$p_y\cos\theta_1 - p_x\sin\theta_1 = 0 \tag{5.36}$$

Ayak bileği (yuvarlanma) ekleminin dönme açısı olan θ_1, (5.36) kullanılarak (5.37)'deki gibi hesaplanır.

$$\theta_1 = \tan^{-1}\left(\frac{p_y}{p_x}\right) \tag{5.37}$$

(5.37) ile elde edilen açı değeri eğer açı sınırları dışındaysa (5.38) geçerli olacaktır.

$$\theta_1 = \tan^{-1}\left(-\frac{p_y}{p_x}\right) \tag{5.38}$$

θ_3'ü elde etmek için ilk olarak (5.34) ve (5.35)'teki eşitliklerin her iki taraflarının kareleri alınır. Daha sonra $c_{23} = c_2c_3 - s_2s_3$ ve $s_{23} = s_2c_3 + c_2s_3$ trigonometrik fark ve toplam formülleri kullanılarak eşitlikler taraf tarafa toplandığında (5.39) ve (5.40) elde edilir.

$$\cos\theta_3 = \frac{{p_z}^2 + (p_x\cos\theta_1 + p_y\sin\theta_1)^2 - 93^2 - 93^2}{2(93)^2} \tag{5.39}$$

$$\sin\theta_3 = \sqrt{1 - \cos^2\theta_3} \tag{5.40}$$

(5.39) ve (5.40) kullanılarak diz ekleminin dönme açısı olan θ_3, (5.41)'deki gibi hesaplanır.

$$\theta_3 = \tan^{-1}\left(\frac{\sin\theta_3}{\cos\theta_3}\right) \tag{5.41}$$

(5.41) ile elde edilen açı değeri, açı sınırları dışındaysa (5.42) geçerli olacaktır.

$$\theta_3 = \tan^{-1}\left(\frac{-\sin\theta_3}{\cos\theta_3}\right) \tag{5.42}$$

θ_1 ve θ_3 açıları elde edildikten sonra (5.31)'deki eşitliğin her iki tarafı ${A_2}^{-1}$ ile çarpılarak (5.43) ve (5.44) elde edilmiştir.

$${A_2}^{-1}{A_1}^{-1}T_0^5 = {A_2}^{-1}{A_1}^{-1}A_1A_2A_3A_4A_5 \tag{5.43}$$

$${A_2}^{-1}{A_1}^{-1}\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_3A_4A_5 \tag{5.44}$$

(5.44)'teki işlemler yapılarak (5.45), (5.46) ve (5.47) eşitlikleri elde edilmiştir.

$$p_x\cos\theta_1\cos\theta_2 - p_z\sin\theta_2 - 93 + p_y\cos\theta_2\sin\theta_1 = 93\cos\theta_3 \tag{5.45}$$

$$-p_z\cos\theta_2 - p_x\cos\theta_1\sin\theta_2 - p_y\sin\theta_1\sin\theta_2 = 93\sin\theta_3 \tag{5.46}$$

$$p_y\cos\theta_1 - p_x\sin\theta_1 = 0 \tag{5.47}$$

θ_2'yi elde etmek için öncelikle (5.45) ve (5.46)'daki eşitliklerin her iki tarafları sırasıyla $\cos\theta_2$ ve $-\sin\theta_2$ ile çarpılarak eşitlikler taraf tarafa toplanır. Bu işlemlerin sonucunda (5.48) elde edilir.

$$\sin\theta_2 = \frac{p_x\cos\theta_1 + p_y\sin\theta_1 - \cos\theta_2(93+93\cos\theta_3)}{-93\sin\theta_3} \tag{5.48}$$

Daha sonra, (5.45) ve (5.46)'daki eşitliklerin her iki tarafları sırasıyla $\sin\theta_2$ ve $\cos\theta_2$ ile çarpılarak eşitlikler taraf tarafa toplanır. Bu işlemlerin sonucunda (5.49) elde edilir.

$$\cos\theta_2 = \frac{p_z + (93+93\cos\theta_3)\sin\theta_2}{-93\sin\theta_3} \tag{5.49}$$

Aşağıdaki kısaltmalar kullanılarak (5.48) ve (5.49) eşitlikleri, (5.50) ve (5.51)'deki gibi yazılır.

$$A = p_x\cos\theta_1 + p_y\sin\theta_1 \, , B = 93 + 93\cos\theta_3 \, , C = -93\sin\theta_3$$

$$\sin\theta_2 = \frac{A - B\cos\theta_2}{C} \tag{5.50}$$

$$\cos\theta_2 = \frac{p_z + B\sin\theta_2}{C} \tag{5.51}$$

(5.50), (5.51)'de yerine konulduğunda (5.52) elde edilir.

$$\cos\theta_2 = \frac{Cp_z + AB}{B^2 + C^2} = \frac{-p_z 93\sin\theta_3 + (p_x\cos\theta_1 + p_y\sin\theta_1)(93+93\cos\theta_3)}{(93+93\cos\theta_3)^2 + (-93\sin\theta_3)^2} \tag{5.52}$$

(5.51), (5.50)'de yerine konulduğunda ise (5.53) elde edilir.

$$\sin\theta_2 = \frac{AC - Bp_z}{C^2 - B^2} = \frac{-(p_x\cos\theta_1 + p_y\sin\theta_1)93\sin\theta_3 - (93+93\cos\theta_3)p_z}{(-93\sin\theta_3)^2 - (93+93\cos\theta_3)^2} \tag{5.53}$$

(5.52) ve (5.53) kullanılarak ayak bileği (yunuslama) ekleminin dönme açısı olan θ_2, (5.54)'teki gibi hesaplanır.

$$\theta_2 = \tan^{-1}\left(\frac{\sin\theta_2}{\cos\theta_2}\right) \tag{5.54}$$

(5.54) ile elde edilen açı değeri, açı sınırları dışındaysa (5.55) geçerli olacaktır.

$$\theta_2 = \tan^{-1}\left(-\frac{\sin\theta_2}{\cos\theta_2}\right) \tag{5.55}$$

θ_4 ve θ_5 açılarını elde etmek için (5.43)'teki eşitliğin her iki tarafı ${A_3}^{-1}$ ile çarpılarak (5.56) ve (5.57) elde edilmiştir.

$${A_3}^{-1}\,{A_2}^{-1}{A_1}^{-1}T_0^5 = {A_3}^{-1}{A_2}^{-1}{A_1}^{-1}A_1A_2A_3A_4A_5 \tag{5.56}$$

$${A_3}^{-1}{A_2}^{-1}{A_1}^{-1}\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_4A_5 \tag{5.57}$$

(5.57)'deki işlem gerçekleştirildikten sonra iki matrisin eleman eşitlikleri ilkesi kullanılmıştır. θ_4 ve θ_5 açıları dönme matrisiyle ilgili olduğundan ilk üç sütuna bakılarak (5.58), (5.59), (5.60) ve (5.61) elde edilmiştir.

$$\cos\theta_3\left(-n_z\sin\theta_2+\cos\theta_2\left(n_x\cos\theta_1+n_y\sin\theta_1\right)\right)-\sin\theta_3(n_z\cos\theta_2+\sin\theta_2\left(n_x\cos\theta_1+n_y\sin\theta_1\right)=\cos\theta_4\cos\theta_5 \quad (5.58)$$

$$\sin\theta_3\left(n_z\sin\theta_2-\cos\theta_2\left(n_x\cos\theta_1+n_y\sin\theta_1\right)\right)-\cos\theta_3\left(n_z\cos\theta_2+\sin\theta_2\left(n_x\cos\theta_1+n_y\sin\theta_1\right)\right)=\sin\theta_4\cos\theta_5 \quad (5.59)$$

$$n_y\cos\theta_1-n_x\sin\theta_1=\sin\theta_5 \quad (5.60)$$

$$o_y\cos\theta_1-o_x\sin\theta_1=\cos\theta_5 \quad (5.61)$$

(5.58) ve (5.59) kullanılarak kalça (yunuslama) ekleminin dönme açısı olan θ_4, (5.62)'deki gibi hesaplanır.

$$\theta_4=\tan^{-1}\left(\frac{\sin\theta_3\left(n_z\sin\theta_2-\cos\theta_2\left(n_x\cos\theta_1+n_y\sin\theta_1\right)\right)-\cos\theta_3\left(n_z\cos\theta_2+\sin\theta_2\left(n_x\cos\theta_1+n_y\sin\theta_1\right)\right)}{\cos\theta_3\left(-n_z\sin\theta_2+\cos\theta_2\left(n_x\cos\theta_1+n_y\sin\theta_1\right)\right)-\sin\theta_3(n_z\cos\theta_2+\sin\theta_2\left(n_x\cos\theta_1+n_y\sin\theta_1\right)}\right) \quad (5.62)$$

(5.62) ile elde edilen açı değeri, açı sınırları dışındaysa (5.63) geçerli olacaktır.

$$\theta_4=\tan^{-1}\left(-\frac{\sin\theta_3\left(n_z\sin\theta_2-\cos\theta_2\left(n_x\cos\theta_1+n_y\sin\theta_1\right)\right)-\cos\theta_3\left(n_z\cos\theta_2+\sin\theta_2\left(n_x\cos\theta_1+n_y\sin\theta_1\right)\right)}{\cos\theta_3\left(-n_z\sin\theta_2+\cos\theta_2\left(n_x\cos\theta_1+n_y\sin\theta_1\right)\right)-\sin\theta_3(n_z\cos\theta_2+\sin\theta_2\left(n_x\cos\theta_1+n_y\sin\theta_1\right)}\right) \quad (5.63)$$

(5.60) ve (5.61) kullanılarak kalça (yuvarlanma) ekleminin dönme açısı olan θ_5, (5.64)'teki gibi hesaplanır.

$$\theta_5=\tan^{-1}\left(\frac{n_y\cos\theta_1-n_x\sin\theta_1}{o_y\cos\theta_1-o_x\sin\theta_1}\right) \quad (5.64)$$

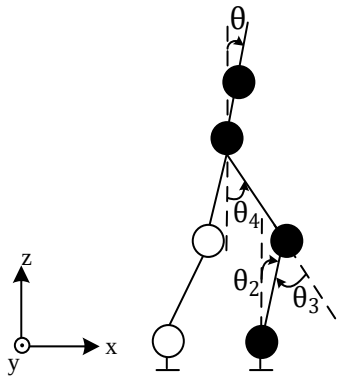
(5.64) ile elde edilen açı değeri, açı sınırları dışındaysa (5.65) geçerli olacaktır.

$$\theta_5=\tan^{-1}\left(-\frac{n_y\cos\theta_1-n_x\sin\theta_1}{o_y\cos\theta_1-o_x\sin\theta_1}\right) \quad (5.65)$$

θ_4 ve θ_5 açıları elde edilirken kullanılan dönme matrisi (5.66)'da verilmiştir. Bu dönme matrisi ile sagital düzlemde robotun gövde eğim açısı olan θ değeri ayarlanır.

$$R(\theta)=\begin{bmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{bmatrix}=\begin{bmatrix}n_x & o_x & a_x\\ n_y & o_y & a_y\\ n_z & o_z & a_z\end{bmatrix} \quad (5.66)$$

Şekil 5.9'da gösterilen θ, bacaktaki yunuslama açılarının toplamına (θ_2+θ_3+θ_4) eşit olup yürüyüş esnasında robotu düşürmeyecek şekilde istenen değerlerde seçilebilir.



Şekil 5.9. Sagital düzlemde robotun gövde eğim açısının gösterimi

Oluşturulan yürüme yörüngesi kullanılarak kalça ekleminin ayak bileği eklemine göre konum koordinatları olan p_x, p_y ve p_z, iki bacak için de hesaplanır. Hesaplama işlemi, anlık olarak kalça ekleminin konumundan ayak bileği ekleminin konumunun çıkarılmasını içermektedir. Elde edilen p_x, p_y ve p_z konum koordinatlarına göre robotun yürüyüşü esnasında bacak eklemlerinin alması gereken açılar, ters kinematik analiz ile hesaplanmaktadır.

6. İNSANSI ROBOTUN YÖNELİM AÇILARI VE YÜRÜYÜŞ KARARLILIK ÖLÇÜTÜ

İnsansı robotun denge sistemi genellikle ivmeölçer ve jiroskop sensörlerinden oluşan Atalet Ölçüm Birimi (AÖB), ana işlemci birimi ve robotun her ekleminde bulunan aktüatörlerden oluşur. AÖB, robot gövdesinin eğim açısını algılamak için eğim sensörü olarak kullanılır. Bir jiroskop ve ivmeölçer kullanılarak ortamı tanıyabilen ve insansı robotun duruşu değiştirilerek ortama uyarlanabilen duruş kontrol denge sistemi sağlanabilir.

İnsansı robotlar, insanlar için akıllı görevler ve hizmetler gerçekleştirebilen çok yönlü robot platformlarıdır. Bu nedenle, yüksek hareketlilik için iki ayaklı yürüyüş dahil etkili bir hareket kararlılığı çok önemlidir.

Bu bölümde, DPÖ algoritmalarının durum uzayları için gereken yönelim açılarının ve yürüyüş kararlılık ölçütünün hesaplanma işlemlerinin nasıl yapıldığı detaylandırılmıştır.

6.1. Yönelim Açılarının Hesaplanması

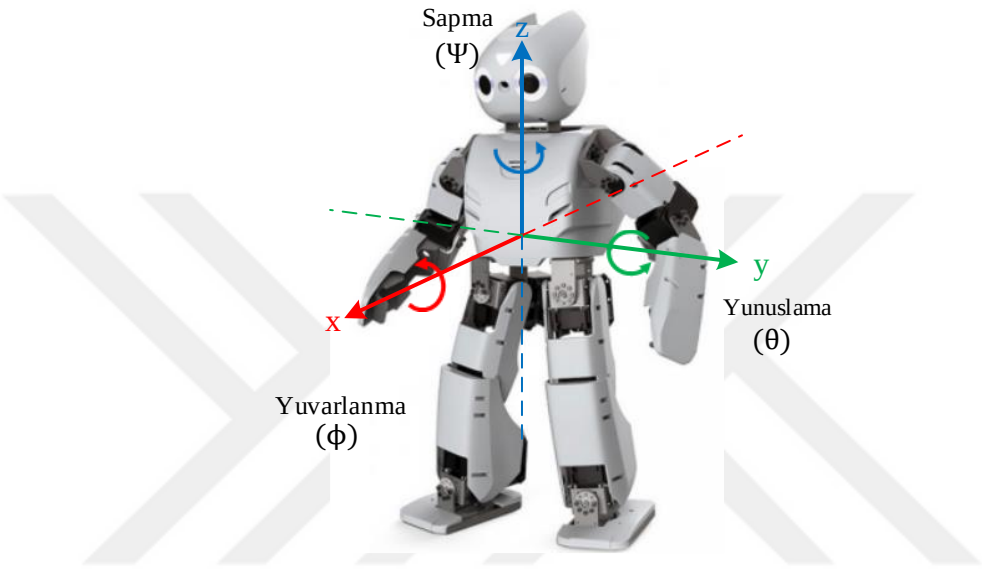
AÖB, robotun düşmesine neden olacak bir eğim açısında olduğu hakkında bilgi verdiğinde, ana işlemci, dengeleme eylemini yapması için aktüatörlere komutlar göndererek robotun bir dengeleme konumuna geçmesini sağlar. AÖB ile bir cismin uzaydaki 3 boyutlu yönelimi bulunabilir. 3 boyutlu yönelimi ifade etmede genellikle Euler açıları ve kuaterniyonlar (dördey) kullanılmaktadır [210]. Euler açıları ile yönelim hesabı yapılırken x, y ve z eksenlerinin her biri etrafındaki dönme açıları, belli bir sırayla bulunur. Kuaterniyon ile yönelim hesabında ise tanımlı bir koordinat sistemindeki bir birim vektör etrafında yapılır. Euler açıları ZYX eksen gösterimine göre ifade edilirken, havacılık ve robotik alanında XYZ eksen gösterimine göre özelleşmiş bir sürümü olan yunuslama (pitch), yuvarlanma (roll) ve sapma (yaw) açıları kullanılır [211]. Yunuslama açısının $\pm 90°$ olduğu durumda eksenler üst üste gelerek serbestlik derecesi kaybı yaşanır ve tekillik (Gimbal Lock) sorunu ortaya çıkar [212]. Böyle durumlarda, kuaterniyon gösterimi tercih edilmektedir.

Bu tez kapsamında gerçekleştirilen çalışmalarda tekillik sorununa sebep olacak bir hareket olmadığı için yönelim hesabında yoğun hesaplamalar gerektiren kuaterniyon gösterimi yerine Euler açılarının özelleşmiş sürümü olan yunuslama, yuvarlanma ve sapma açıları kullanılmıştır.

3 eksenli ivmeölçer x, y ve z ekseni boyunca üzerlerine düşen statik (yerçekimi) veya dinamik (aniden hızlanma veya durma) ivmeyi ölçmektedir. 3 eksenli bir jiroskop ise x, y ve z ekseni boyunca açısal hızı ölçmektedir. İvmeölçer, sadece yunuslama ve yuvarlanma açılarını hesaplayabildiği için sapma eksenine referans olamaz. Çünkü eksen boyunca dönüş, yer çekimine göre herhangi bir değişikliğe neden olmaz. Sapma açısına referans olması için kullanılan sensör,

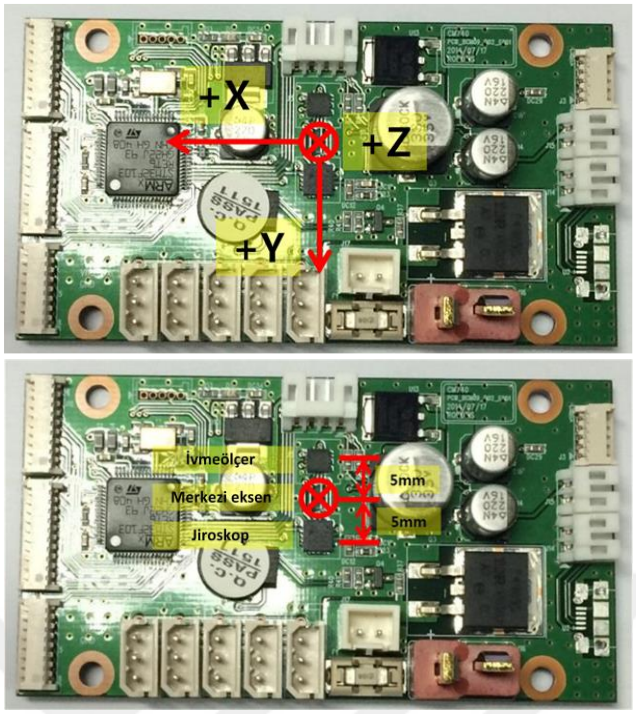
manyetometredir. Farklı algoritmalar ve filtreler kullanılarak jiroskop üzerinden hesaplanan açısal değişimler, yunuslama ve yuvarlanma açıları için ivmeölçer, sapma açısı için ise manyetometre tarafından düzeltilerek yönelim açılarının hesaplanması sağlanır.

Yunuslama (θ), yuvarlanma (Φ) ve sapma (Ψ) açıları, Robotis-OP2 üzerinde Şekil 6.1'deki gibi temsil edilmektedir.



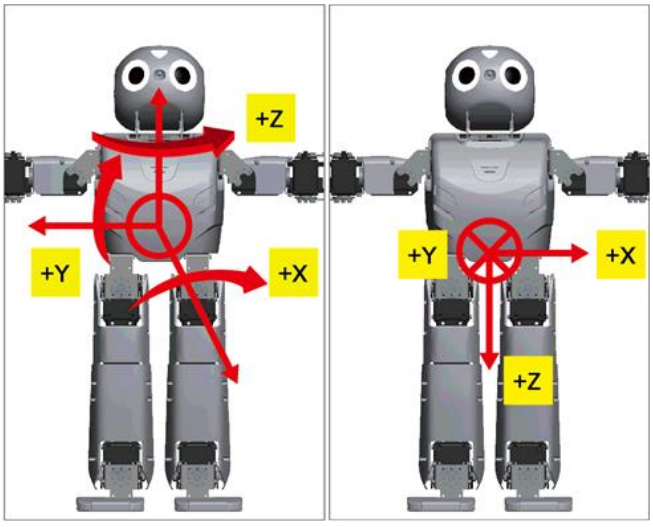
Şekil 6.1. Robotis-OP2 üzerinde yunuslama, yuvarlanma ve sapma açılarının gösterimi

Jiroskop ve ivmeölçer, Robotis-OP2'nin alt kontrolörü olan CM-740 kontrolör kartına gömülüdür. Ham ivmeölçer ve jiroskop verileri, CM-740 alt kontrolör kartından okunur. Robotis-OP2'nin AÖB'si, ivmeölçerde 3 ve jiroskopta 3 olmak üzere 6 serbestlik derecesine sahiptir. İvmeölçerin merkezi ekseni ile ivmeölçer ve jiroskop yerleşimi Şekil 6.2'de gösterilmiştir.



Şekil 6.2. Jiroskop ve ivmeölçerin yerleştirildiği CM-740 kontrolör kartı ve sensörlerin konumları

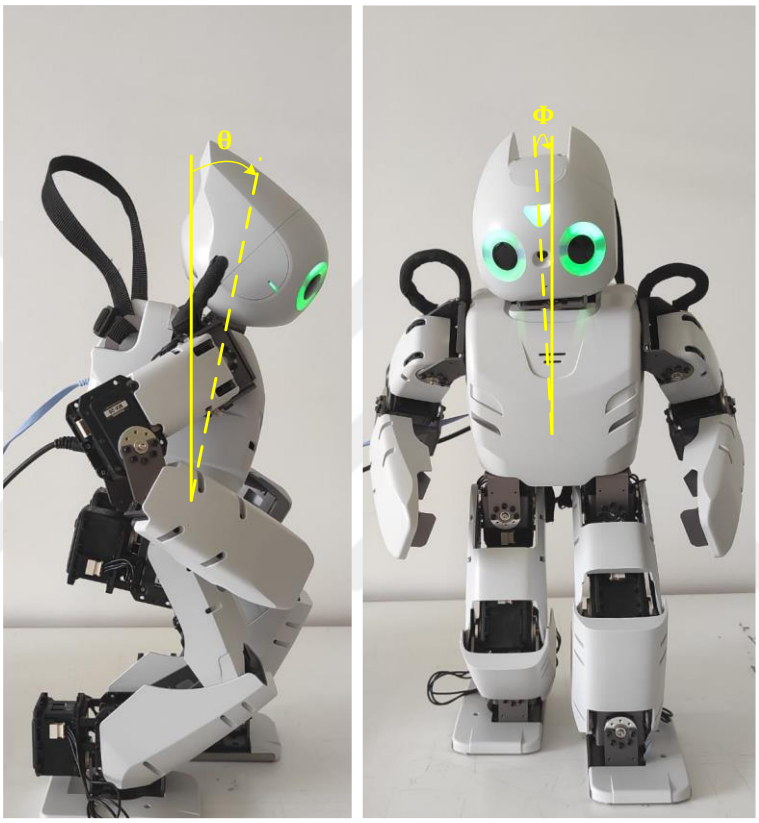
Jiroskop ve ivmeölçer eksenlerinin Robotis-OP2 üzerinde gösterimi ise Şekil 6.3'te verilmiştir.



(a) (b)

Şekil 6.3. Robot üzerinde sensörlerin eksen gösterimi: (a) jiroskop ve (b) ivmeölçer [104]

Robotis-OP2'nin sagital düzlemdeki düz yürüyüşü göz önüne alındığında gövde eğim açısı, yunuslama açısıdır. Koordinat sistemine göre pozitif yunuslama açısı saat yönünün tersi, pozitif yuvarlanma açısı ise saat yönü boyuncadır. Yürüyüş, ön düzlemde gözlemlendiğinde ise gövde eğim açısı olarak yuvarlanma açısı dikkate alınmalıdır. Şekil 6.4'te yunuslama ve yuvarlanma açıları gösterilmiştir.



(a) (b)

Şekil 6.4. Robot gövde eğim açıları: (a) yunuslama açısı ve (b) yuvarlanma açısı

AÖB ile bir eksen etrafındaki dönme açılarını hesaplamak için ölçülen bu açısal hız ve doğrusal ivmelerden yararlanılır. Robotis-OP2 üzerindeki AÖB, analog çıkışlı 3 eksen ADXL335 ivmeölçer ve analog çıkışlı 3 eksen LYPR540AH jiroskoptan oluşmaktadır. Jiroskop, x, y ve z eksenleri boyunca sırasıyla ω_x, ω_y ve ω_z açısal hızları ölçerken ivmeölçer, x, y ve z eksenleri boyunca sırasıyla a_x, a_y ve a_z doğrusal ivmeleri ölçer. Ancak, sensörlerden alınan veriler ham verilerdir. Bu veriler, CM-740 alt kontrolörünün 10 bitlik ADC (Analog to Digital Converter–Analog-Dijital Çevirici) biriminden okunan değerlerdir. Bu nedenle ivmeölçer ve jiroskoptan alınan ham veriler, (6.1)'deki eşitlik kullanılarak ivmeölçer için yer çekimi ivmesinin büyüklüğü cinsinden g (9.8 m/s^2) ve jiroskop için derece/s'yi ifade eden fiziksel birimlere dönüştürülmüştür.

$$R = \left(\frac{ADC_{değer} \times V_{ref.}}{\text{Bit Oranı}} - V_{sıfır}\right) \times \frac{1}{\text{Duyarlılık}} \quad (6.1)$$

Burada; $ADC_{değer}$ sensörden elde edilen değeri, V_{ref} sensör besleme gerilimini, $V_{sıfır}$ ivmeölçer için 0 g'deki (ivme yok iken) ve jiroskop için herhangi bir dönüşün olmadığı yani hareketsiz durumdaki çıkış gerilimini, R ivmeölçer ve jiroskop için sırasıyla a_x, a_y, a_z doğrusal ivmeleri ve ω_x, ω_y, ω_z açısal hızları ve duyarlılık ise sensörün hassasiyet değerini ifade etmektedir.

Tablo 6.1'de 3 eksende ±3g aralığında ivme ölçebilen ADXL335 ivmeölçerin ve 3 eksende ±1600 derece/s aralığında açısal hız ölçebilen LYPR540AH jiroskobun parametreleri verilmiştir.

Tablo 6.1. İvmeölçer ve jiroskop parametreleri

Parametre	İvmeölçer	Jiroskop
$ADC_{değer}$	0~1023	0~1023
$V_{ref.}$	3.3 V	3.3 V
Bit Oranı	1024	1024
$V_{sıfır}$	1.65 V	1.65 V
Duyarlılık	0.33 V/g	0.0008 V/(derece/s)
R	a	ω

(6.1)'deki eşitlik kullanılarak g türünden elde edilen ivme değerleri, yunuslama (θ) ve yuvarlanma (Φ) açılarına sırasıyla (6.2) ve (6.3)'teki trigonometrik fonksiyonlar ile dönüştürülmüştür.

$$\theta_{ivmeölçer} = \tan^{-1}\left(\frac{a_y}{\sqrt{a_x{}^2 + a_z{}^2}}\right)\frac{180}{\pi} \quad (derece) \quad (6.2)$$

$$\Phi_{ivmeölçer} = \tan^{-1}\left(\frac{a_x}{\sqrt{a_y{}^2 + a_z{}^2}}\right)\frac{180}{\pi} \quad (derece) \quad (6.3)$$

Burada; a_x, a_y ve a_z sırasıyla x, y ve z eksenleri boyunca ölçülen ivmelerdir.

(6.1)'deki eşitlik kullanılarak derece/s türünden elde edilen açısal hız değerleri kullanılarak yunuslama (θ) ve yuvarlanma (Φ) açıları sırasıyla (6.4) ve (6.5) ile elde edilmiştir.

$$\theta_{jiroskop} = \int_0^t \omega_y dt \quad (derece) \quad (6.4)$$

$$\Phi_{jiroskop} = \int_0^t \omega_x dt \quad (derece) \quad (6.5)$$

Burada; dt örnekleme periyodu, ω_y ve ω_x ise sırasıyla y ve x eksenleri boyunca ölçülen açısal hızlardır.

Hem ivmeölçer hem de jiroskop verileri sistematik hatalara eğilimlidir. İdeal bir ortamda ivmeölçer, tek başına eğimin hesaplanması için gerekli veriyi sağlayabilmektedir. Ancak, ivmeölçerler kuvvete karşı çok duyarlı olduğundan en ufak titreşimlerde dahi çok yüksek gürültüler oluşturarak eğim hesabı için gerekli ölçümleri bozmaktadır. Jiroskoplar, bu kuvvetlerden etkilenmez, ancak ilk açı bilinmiyor ise sadece jiroskop ile güncel açılar hesaplanamaz, açısal hızın zamana bağlı integrali alınarak açı değişimi hesaplanabilir. Jiroskobun diğer bir sorunu ise kayma özelliğidir. Jiroskop sabit duran bir robot üzerinde bile küçük de olsa açısal hızlar hesaplar. Bu yüzden, sadece jiroskoba dayalı hesaplanan açılar zamanla kayar. Bahsedilen bu durumlar göz önünde bulundurulduğunda ivmeölçer, uzun vadede doğru veriler sağlar, ancak kısa vadede gürültülüdür. Jiroskop, kısa vadede değişen yönelim hakkında doğru veriler sağlar ve kararlıdır, ancak uzun zaman ölçeklerinde açı kaymaları meydana gelir. Jiroskoptaki kayma sorunu, ivmeölçer tarafından ölçülen yerçekimi ivmesi kullanılarak ortadan kaldırılır.

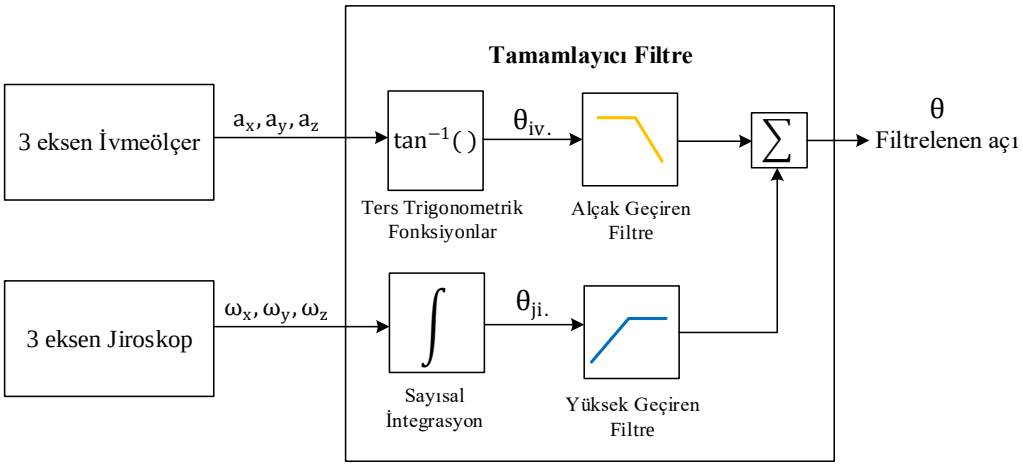
İnsansı robot gövdesinin yunuslama ve yuvarlanma açılarını kararlı bir şekilde hesaplayabilmek için AÖB'den alınan ivmeölçer ve jiroskop verileri, hataları ortadan kaldıracak şekilde birleştirilmelidir. Sensör birleştirme ile kayma engellenir ve sadece ivmeölçere bağlı olunmadığı için hareket halinde oluşan gürültülü sonuçlar ortadan kalkmış olur. Filtreleme işlemlerinden sonra hassas bir şekilde hesaplanan gövde eğim açısı aracılığıyla insansı robotun üzerinde bulunduğu yüzeyin durumu hakkında da bilgiler elde edilebilmektedir.

Tez kapsamında gerçekleştirilen çalışmalarda, AÖB'ler için yaygın olarak tercih edilen tamamlayıcı filtre ve Kalman filtreden oluşan iki farklı sensör birleştirme yöntemi ayrı ayrı kullanılmıştır. Filtrelerden daha başarılı olan gözlemlenerek en başarılı filtrenin kullanımına karar verilmiştir.

6.1.1. Tamamlayıcı Filtre

Tamamlayıcı filtre [213], ivmeölçer ve jiroskop verilerini birleştirerek düzgün ve tutarlı bir sinyal verir. Bir sensörün gücü, diğer sensörün zayıflıklarının üstesinden gelecek şekilde geliştirilen tamamlayıcı filtre, ismini bu özelliğinden almıştır. Bu filtrenin amacı, kısa vadede jiroskop verisini kullanmayı, uzun vadede ise ivmeölçerlerin verisini kullanmayı sağlamaktır.

İvmeölçer için alçak geçiren bir filtre ve jiroskop için yüksek geçiren bir filtreden oluşan tamamlayıcı filtrenin blok diyagramı Şekil 6.5'te verilmiştir.



Şekil 6.5. Tamamlayıcı filtrenin blok diyagramı

Alçak Geçiren Filtre kısaca AGF (Low-Pass Filter–LPF), yalnızca uzun vadeli değişikliklere izin vererek kısa vadeli dalgalanmaları filtrelemek içindir. AGF, yalnızca değişim hızı küçük olan ivmeölçer verilerinin geçmesine izin verir. Böylece, titreşim azaltılmış olur. Yüksek Geçiren Filtre kısaca YGF (High-Pass Filter–HPF) ise zaman içinde sabit olan sinyalleri filtrelerken kısa süreli sinyallerin geçmesine izin verir. YGF, yalnızca değişim hızı örnekleme periyodu içinde yakalanacak kadar büyük olan jiroskop verilerinin geçmesine izin verir. Bu da kaymayı azaltır.

Tamamlayıcı filtre, hesaplama için çok fazla değişken gerektirmediği için tamamlayıcı filtrenin hesaplama süreci, Kalman filtresi kadar karmaşık değildir. (6.6)'daki eşitlik ile ifade edilen tamamlayıcı filtrede, ivmeölçer ve jiroskobun birbirlerine olan etki miktarını kontrol eden bir filtre katsayısı vardır.

$$A_t = \underbrace{\alpha(A_{t-1} + \omega dt)}_{\text{YGF}} + \underbrace{(1-\alpha)(A_{iv.})}_{\text{AGF}} \tag{6.6}$$

A_t: Filtrelenen açı değeri (derece),

α: Filtre katsayısı,

A_{t-1}: Filtrelenen bir önceki açı değeri (derece),

ω: Jiroskoptan alınan açısal hız değeri (derece/s),

dt: Örnekleme periyodu (s),

$A_{iv.}$: İvmeölçerden hesaplanan açı değeri (derece).

Tamamlayıcı filtrenin filtre katsayısı (α), alçak geçiren ve yüksek geçiren filtrenin zaman sabiti (τ) ile jiroskobun örnekleme periyoduna (dt) bağlıdır. (6.7) ile hesaplanan filtre katsayısında zaman sabiti jiroskobun kayma hızına göre ayarlanmalıdır.

$$\alpha = \frac{\tau}{\tau + dt} \tag{6.7}$$

Jiroskop saniyede ortalama iki derece kayıyor ise kaymanın telafi edilebilmesi için zaman sabiti, bir saniyeden az olmalıdır. Ancak zaman sabiti ne kadar düşükse, ivmeölçerden o kadar fazla gürültünün geçmesine izin verilecektir. Yunuslama ve yuvarlanma açıları sırasıyla (6.8) ve (6.9)'daki eşitlikler kullanılarak hesaplanmıştır.

$$\theta_t = \alpha\left(\theta_{t-1} + \omega_y dt\right) + (1-\alpha)(\theta_{iv.}) \tag{6.8}$$

$$\Phi_t = \alpha(\Phi_{t-1} + \omega_x dt) + (1-\alpha)(\Phi_{iv.}) \tag{6.9}$$

θ_t: Filtrelenen yunuslama açı değeri (derece),

Φ_t: Filtrelenen yuvarlanma açı değeri (derece),

α: Filtre katsayısı,

θ_{t-1}: Filtrelenen bir önceki yunuslama açı değeri (derece),

Φ_{t-1}: Filtrelenen bir önceki yuvarlanma açı değeri (derece),

ω_y: Jiroskoptan alınan y eksenindeki açısal hız değeri (derece/s),

ω_x: Jiroskoptan alınan x eksenindeki açısal hız değeri (derece/s),

dt: Örnekleme periyodu (s),

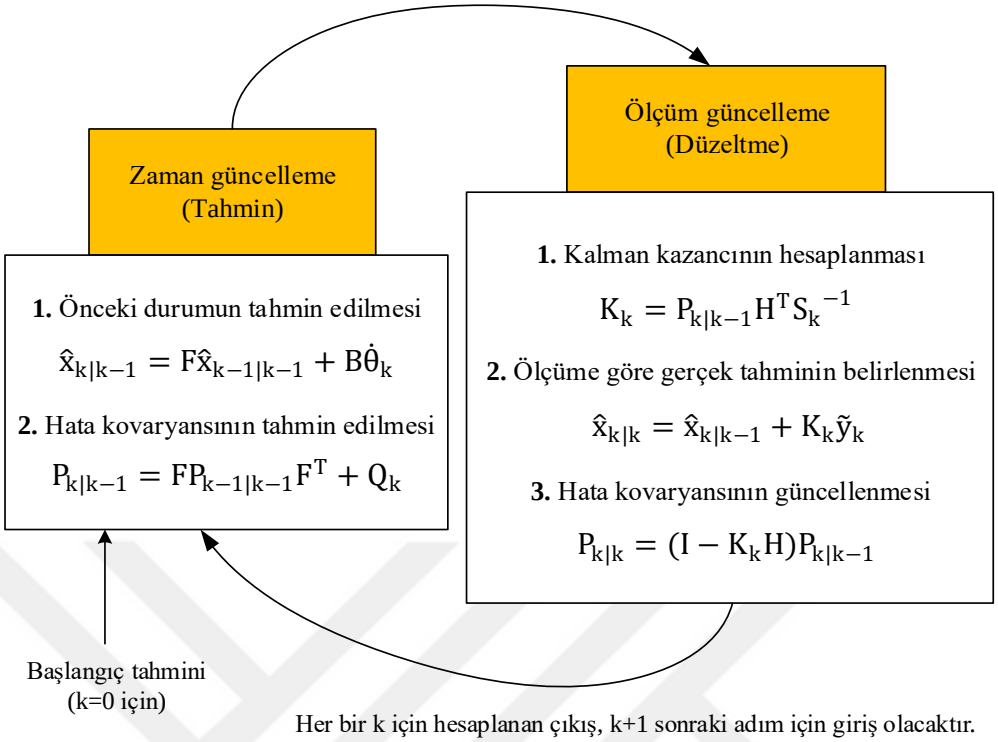
$\theta_{iv.}$: İvmeölçerden hesaplanan yunuslama açı değeri (derece),

$\Phi_{iv.}$: İvmeölçerden hesaplanan yuvarlanma açı değeri (derece).

6.1.2. Kalman Filtresi

Çevredeki birçok sistem dinamiktir ve bu sistemler modellenirken sistemlere etkiyen tüm faktörler ölçülemez veya doğruluğundan emin olunamaz. Bundan dolayı modellenmek istenen sistemlere ait yetersiz bilgilerden tüm sistem hakkında öngörüde bulunabilmek için Kalman filtresi [214] idealdir. Kalman filtresi, bir yandan mevcut ve önceki durumlara dayalı olarak sistemin bir sonraki durumunu tahmin etmeye çalışırken, diğer yandan hatayı en aza indiren yani sürekli gerçek değere ulaşmayı hedefleyen bir formülasyondur.

Kalman filtresi Şekil 6.6'da verildiği gibi temelde iki aşamadan oluşur. Bunlardan ilki zaman güncelleme (tahmin), diğeri ölçüm güncellemedir (düzeltme). İlk aşamada, sistem durumu hakkında bir tahmin yapmak için matematiksel bir durum modeli kullanılır. Sonraki aşamada ise bu durum tahmini, ölçülen durum değerleri ile karşılaştırılır. Tahmin edilen ve ölçülen durum arasındaki fark, sistemdeki ve ölçümlerdeki tahmini gürültü ve hataya göre yönetilir. Filtre çıkışı olan durum tahmini, bu iki aşama arasında sürekli bir döngü içinde çalıştırılarak tahminin gerçek değere yakınsaması sağlanır.



Şekil 6.6. Kalman filtresinin aşamaları

$\hat{x}_{k-1|k-1}$: Önceki durum ve ondan önceki durumların tahminlerine dayanan önceki durum tahmini,

$\hat{x}_{k|k-1}$: Sistemin önceki durumuna ve ondan önceki durumların tahminine dayanan mevcut k anındaki durum tahmini,

$\hat{x}_{k|k}$: k zamanına kadar olan ve k zamanını içeren gözlemlerin k anındaki durum tahmini.

$\hat{x}$, durumun tahmini olduğu anlamına gelir. Gerçek durum anlamına gelen x'in k anındaki durumu (6.10) ile verilirse:

$$x_k = Fx_{k-1} + Bu_k + \omega_k \tag{6.10}$$

Burada; x_k (6.11)'de verilen durum matrisi, u_k kontrol girişi, F geçiş modeli, B kontrol giriş modeli ve ω_k Gauss işlem gürültüsüdür.

$$x_k = \begin{bmatrix} \theta \\ \dot{\theta}_b \end{bmatrix}_k \tag{6.11}$$

Filtrenin çıkışı θ açısı ve aynı zamanda ivmeölçer ve jiroskoptan alınan ölçümlere dayanan $\dot{\theta}_b$ biasıdır. Bias, jiroskobun kayma miktardır. Jiroskop ölçümünden bias çıkarılarak gerçek oran elde edilir.

Önceki durum x_{k-1}'e uygulanan durum geçiş modeli F matrisi, (6.12) ile ifade edilir.

$$F = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \tag{6.12}$$

Burada; Δt jiroskobun örnekleme periyodunu ifade etmektedir.

k anında derece/s cinsinden jiroskop ölçümü olan kontrol girişine aynı zamanda, $\dot{\theta}$ da denir. Durum denklemi (6.13)'teki gibi yeniden yazılır.

$$x_k = Fx_{k-1} + B\dot{\theta}_b + \omega_k \tag{6.13}$$

B matrisi, (6.14) ile tanımlanır.

$$B = \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \tag{6.14}$$

$\dot{\theta}$, Δt ile çarpıldığında θ açısı elde edileceğinden ve bias doğrudan hesaplanamadığından matrisin alt elemanı 0'a ayarlanır.

ω_k, sıfır ortalamayla ve Q kovaryansıyla k zamanına dağıtılmıştır. (6.15)'teki gibi ifade edilir.

$$\omega_k \sim N(0, Q_k) \tag{6.15}$$

Burada; Q_k işlem gürültü kovaryans matrisidir. İvmeölçer ve bias'ın durum tahmininin kovaryans matrisidir. Bu durumda, bias ve ivmeölçerin tahmini bağımsız olarak ele alınır. Böylece, ivmeölçer ve bias tahmininin varyansına eşittir ve (6.16) ile tanımlanır.

$$Q_k = \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_{\dot{\theta}_b} \end{bmatrix} \Delta t \tag{6.16}$$

Görüldüğü gibi Q_k kovaryans matrisi mevcut k zamanına bağlıdır. Dolayısıyla, ivmeölçer varyansı Q_θ ve bias varyansı $Q_{\dot{\theta}_b}$, Δt ile çarpılır. Durumun son güncellemesinden bu yana geçen süre uzadıkça işlem gürültüsü daha büyük olacaktır. Örneğin; jiroskop kaymış olabilir. Kalman filtresinin çalışması için bu sabitlerin bilinmesi gerekir. Daha büyük bir değer ayarlanırsa, durum tahmininde o kadar fazla gürültü olacaktır. Bu yüzden, tahmini açı kaymaya başlarsa, $Q_{\dot{\theta}_b}$ değerinin artırılması gerekir. Aksi takdirde, tahmin yavaş olma eğilimindeyse, açının tahminine çok fazla güveniliyordur ve daha duyarlı hale getirmek için Q_θ değeri azaltılmalıdır.

x_k gerçek durumunun z_k ölçümü (6.17)'deki gibi verilir.

$$z_k = Hx_k + v_k \tag{6.17}$$

Burada; v_k ölçüm gürültüsünü temsil eder. H gözlem modeli olarak adlandırılır ve gerçek durum uzayını gözlemlenen uzaya haritalamak için kullanılır. Gerçek durum gözlemlenemez. Gözlem sadece ivmeölçerden alınan ölçüm olduğundan H, (6.18) ile ifade edilir.

$$H = [1 \quad 0] \tag{6.18}$$

(6.19)'da verilen ölçüm gürültüsü, sıfır ortalama ile Gauss dağılımına ve kovaryans olarak R'ye sahip olmalıdır.

$$v_k \sim N(0, R) \tag{6.19}$$

Ancak R bir matris olmadığından, aynı değişkenin kovaryansı varyansa eşit olduğu için ölçüm gürültüsü ölçüm varyansına eşittir. R, (6.20)'deki gibi tanımlanabilir.

$$R = E[v_k \quad v_k{}^T] = \mathrm{var}(v_k) \tag{6.20}$$

Ölçüm gürültüsünün aynı olduğunu ve k zamanına bağlı olmadığını varsayılırsa (6.21) elde edilmiş olur.

$$\mathrm{var}(v_k) = \mathrm{var}(v) \tag{6.21}$$

Ölçüm gürültü varyansı $\mathrm{var}(v)$ çok yüksek ayarlanırsa, filtre yeni ölçümlere daha az güvendiğinden gerçekten yavaş yanıt verecektir. Ancak çok küçük ayarlanırsa, değer aşılabilir ve ivmeölçer ölçümlerine çok fazla güvenildiği için gürültülü olabilir. Hesabı yuvarlamak için Q_θ ve $Q_{\dot{\theta}_b}$ işlem gürültü varyanslarının ve ölçüm gürültüsü varyansı R'nin bilinmesi gerekir. Daha iyi sonuçlar elde etmek için bu üç parametrenin değerleri değiştirilerek Kalman filtresi etkin bir şekilde ayarlanabilir.

Tahmin

Kalman filtresinin birinci aşamasında, k anındaki mevcut durum ve hata kovaryans matrisi tahmin edilmeye çalışılır. İlk olarak filtre, önceki tüm durumlara ve jiroskop ölçümüne dayanarak mevcut durumu (6.22)'deki eşitlik ile tahmin etmeye çalışır.

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + B\dot{\theta}_k \tag{6.22}$$

Mevcut k anındaki durumu tahmin etmek için fazladan bir giriş olarak kullanıldığı için olası durum $\hat{x}_{k|k-1}$ olarak adlandırılır. Olası durum, kontrol girişi olarak da adlandırılmaktadır.

Olası hata kovaryans matrisi $P_{k|k-1}$, önceki hata kovaryans matrisine $P_{k-1|k-1}$ dayalı (6.23)'teki gibi tanımlanır.

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q_k \quad (6.23)$$

Bu matris, tahmini durumun mevcut değerlerine ne kadar güvenileceğini tahmin edebilmek için kullanılır. Ne kadar küçükse, mevcut tahmini duruma o kadar çok güvenilir.

Hata kovaryans matrisi P, (6.24)'teki gibi 2×2 boyutunda bir matristir.

$$P = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix} \quad (6.24)$$

Düzeltme

z_k ölçümü ve olası durum $x_{k|k-1}$ arasındaki fark, yenilik ($\tilde{y}_k$) olarak adlandırılır ve (6.25) ile ifade edilir.

$$\tilde{y}_k = z_k - H\hat{x}_{k|k-1} \quad (6.25)$$

Gözlem modeli H, olası durumu $\hat{x}_{k|k-1}$ ivmeölçerden ölçülen ölçüm uzayına haritalar. Bu yüzden yenilik, bir matris değildir. Yenilik kovaryansı (6.26) ile hesaplanır.

$$S_k = HP_{k|k-1}H^T + R \quad (6.26)$$

Yenilik kovaryansı, olası hata kovaryans matrisi $P_{k|k-1}$ ve ölçüm kovaryans matrisi R'ye dayalı olarak ölçüme ne kadar güvenilmesi gerektiğini tahmin etmeye çalışır. Gözlem modeli H, olası hata kovaryans matrisini $P_{k|k-1}$ gözlemlenen uzaya haritalamak için kullanılır. Ölçüm gürültüsünün değeri ne kadar büyükse, S'nin değeri de o kadar büyüktür. Bu, gelen ölçüme o kadar güvenilmeyeceği anlamına gelir.

Bir sonraki adım, Kalman kazancını hesaplamaktır. Kalman kazancı, yeniliğe ne kadar güvenileceğini belirtmek için kullanılır ve (6.27)'deki eşitlik ile ifade edilir.

$$K_k = P_{k|k-1}H^T{S_k}^{-1} \quad (6.27)$$

H^T, hata kovaryans matrisi P'nin durumunu, gözlemlenen uzaya haritalamak için kullanılır. Yenilik kovaryansı S'nin tersi ile çarpılarak hata kovaryans matrisi karşılaştırılır.

Başlangıçta durum bilinmiyorsa, hata kovaryans matrisi (6.28)'deki gibi ayarlanır.

$$P = \begin{bmatrix} L & 0 \\ 0 & L \end{bmatrix} \quad (6.28)$$

Robot için başlangıç açısının bilinmediği varsayıldığında (6.29) ile başlatılır.

$$P = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (6.29)$$

Bu durumda Kalman kazancı K, (6.30)'daki gibi 2×1 boyutunda bir matristir.

$$K = \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} \tag{6.30}$$

Mevcut durumun son tahmini (6.31) ile güncellenir.

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \tag{6.31}$$

Hata kovaryans matrisi ise (6.32) ile güncellenir.

$$P_{k|k} = (I - K_k H) P_{k|k-1} \tag{6.32}$$

I, birim matris olarak adlandırılır ve (6.33)'teki gibi tanımlanır.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.33}$$

Tüm bu ifade edilenler göz önüne alındığında Kalman filtresi temel olarak tahminin ne kadar düzeltildiğine bağlı olarak hata kovaryans matrisini kendi kendine düzeltir.

6.2. Yürüyüş Kararlılık Ölçütünün Hesaplanması

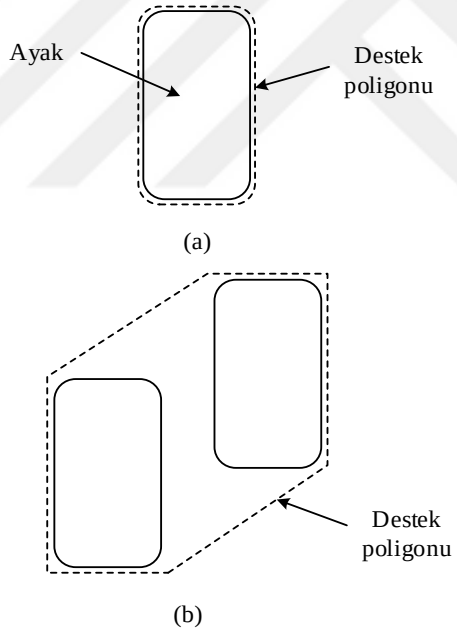
İnsansı robot araştırmaları arasında iki ayaklı yürüme, önemli ve zorlu problemlerden biridir. İki ayaklı yürüyüş, statik ve dinamik yürüyüş olarak ikiye ayrılabilir. Robot, on saniye veya daha fazla sürede bir adım atacak şekilde çok düşük hızlarda yürürken atalet (eylemsizlik) kuvvetinin etkisi ihmal edilebilir düzeydedir ve yalnızca yerçekimi kuvvetlerine maruz kalır. Bu yürüme biçimine "statik yürüyüş" denir. Statik prensibine göre robotun ağırlık merkezinin (kütle merkezi) yere olan izdüşümü noktası, destek poligonu içinde olduğu sürece robot düşmeden kararlı bir şekilde yürüyebilir. Buna, iyi ayaklı robotun statik yürüyüş kararlılık ölçütü denir [56, 215, 216]. Robotun yürüyüş hareketi hızlandığında dinamik kuvvetler, statik kuvvetlere hakim olacağından bu yürüme biçimine ise "dinamik yürüyüş" denir. Dinamik yürüyüşte, ağırlık merkezinin yere olan izdüşümü noktası sınırlı bir süre için destek poligonunu aşabilir. Bu da kararlı yürüme şablonu oluşturmak için ağırlık merkezinin çevrimiçi kontrolünü gerektirir [54, 217-219]. Bu nedenle, robot ağırlık merkezinin hareket yörüngesinin kontrolü, robotun kararlı yürümesinde anahtar rol oynar [220, 221].

Dinamik yürüyüşte, artık statik yürüyüşteki kararlılık ölçütü geçersizdir ve diğer ölçütlerin uygulanması gerekir. 1968 yılında M. Vukobratovic tarafından ortaya atılan Sıfır Moment Noktası kısaca SMN (Zero Moment Point–ZMP), iki veya daha fazla ayaklı robotların dinamik yürüyüş probleminin çözümünde çok sık kullanılan bir kararlılık ölçütüdür [9]. SMN, iki ayaklı hareketin analizi ve sentezi için kullanışlı bir dinamik ölçüt sağlar. Tüm yürüyüş döngüsü boyunca dengeyi

sağlamak için kullanılabilir ve destek ayağı etrafındaki dengesiz moment ve dinamik yürüyüş dengesinin sağlamlığı (dengeleme sınırı) için nicel bir ölçüm sağlar.

SMN, robotun aktif ayak bileği eklemleri varsa ve her zaman en az bir ayağını yerde düz tutuyorsa bir kararlılık ölçütü olarak kullanılabilir. SMN, dikey atalet kuvvetleri ve ağırlık kuvvetleri toplamının sıfır olduğu noktadır. Atalet kuvvetlerinin net momentinin ve yerçekimi kuvvetlerinin yatay bileşeninin olmadığı yerdeki nokta olarak da tanımlanabilir. İki ayaklı robotun yürüyüşü sırasında, SMN'si her zaman destek poligonu içerisinde yer alıyorsa robot düşmeden dengeli bir şekilde yürüyebilecektir. Bu durum, dinamik yürüyüşün kararlılık ölçütüdür. SMN ayağın merkezine ne kadar yakınsa, yürüyüş o kadar kararlı hale gelmektedir.

Destek poligonu, insansı robot ile yer arasındaki temas alanı olarak tanımlanabilir. İki ayaklı bir yürüyüşün iki aşamaya bölünebileceği iyi bilinmektedir; tek destek fazı ve çift destek fazı [222]. Şekil 6.7'de gösterildiği gibi tek destek fazında, yer ile temas halinde olan ayak tabanı, çift destek fazında ise tabanlar ile yer arasındaki temas noktalarının dışbükey zarfı temsil edilir.

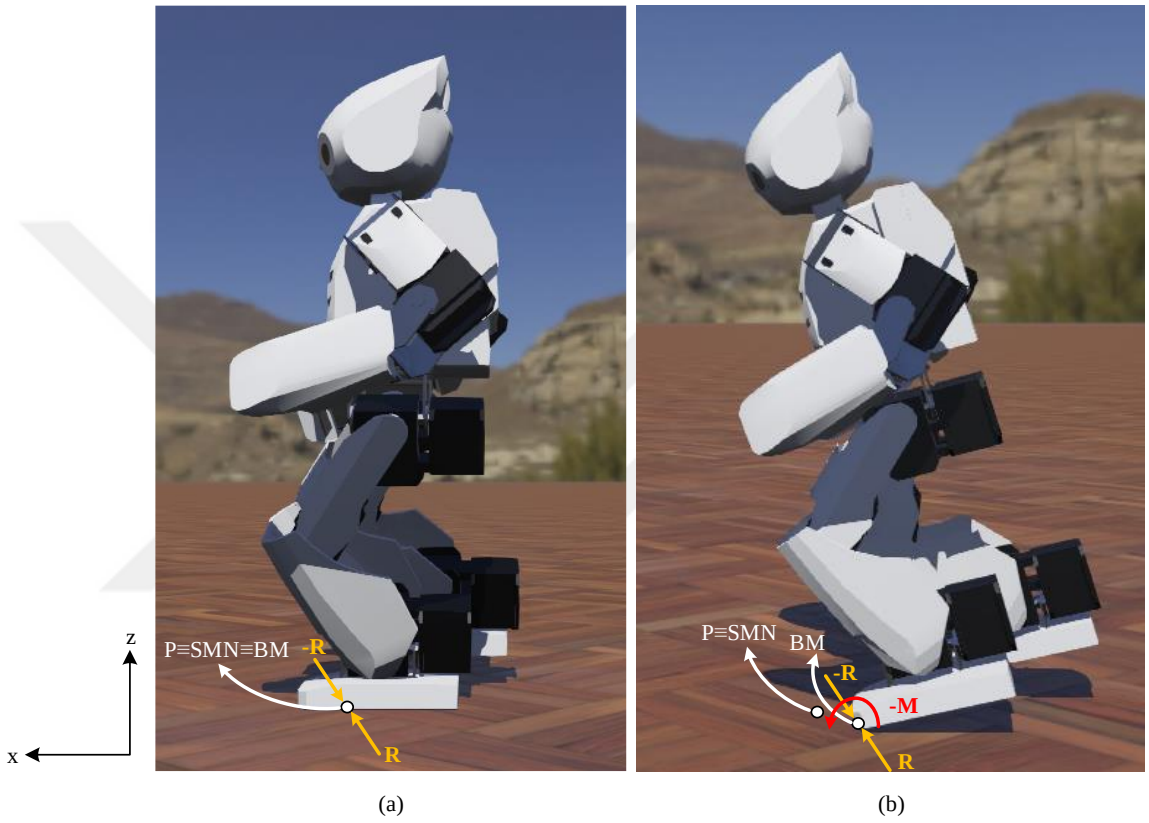


Şekil 6.7. Destek poligonunun tanımı: (a) tek destek fazı ve (b) çift destek fazı

Basınç Merkezi kısaca BM (Center of Pressure–CoP) [223], kuvvet veya basınç ölçümlerine dayalı iki ayaklı yürüme analizinde de yaygın olarak kullanılır. BM, çok gövdeli sistemin ivmelerinden tanımlanan SMN'nin aksine, temas yüzeyindeki etkileşim kuvvetlerinden tanımlanan yerel bir niceliktir.

Tek destek fazında SMN destek poligonunun içinde bulunuyorsa sistem kararlıdır ve ayağın BM'si, SMN ile çakışmaktadır. Bu iki farklı yaklaşım, robot düz bir zeminde dengeli bir şekilde

yürüdüğünde Şekil 6.8(a)'da gösterildiği gibi aynı noktayı verir. Dinamik olarak kararlı durumda, BM'nin konumu hesaplanarak SMN'nin konumunun belirlenebileceği açıktır. SMN, Şekil 6.8(b)'deki gibi destek poligonunun dışındaysa sistem kararsızdır. SMN'nin anlık olarak destek poligonunun kenarında veya dışında olması, ayak tepki kuvvetleri R tarafından telafi edilemeyen dengesiz bir M momentinin oluştuğunu gösterir.



Şekil 6.8. Dinamik olarak kararlı durumda SMN ve BM arasındaki ilişki: (a) dinamik olarak kararlı yürüyüş ve (b) dinamik olarak kararsız yürüyüş

Şekil 6.8 gözlemlendiğinde, dikey z-ekseni ve yatay x-ekseni sagital düzlemde yer alır. Tipik insan eklem yörüngelerinin sagital düzlemde diğer düzlemlere göre daha büyük bir hareket aralığına sahip olduğu bilinmektedir [224] ve yürüme sırasındaki hareketlerin çoğu sagital düzlemde gerçekleşir.

Bir insansı robotun SMN'si, robotun ağırlık merkezinin konum ve ivmesinden hesaplanabilir [225]. İki bacaklı robot sistemlerinde SMN, x ve y eksenlerinde sırasıyla (6.34) ve (6.35)'teki gibi ifade edilebilir.

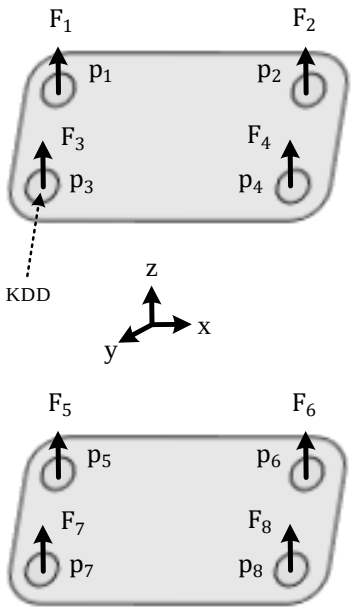
$$SMN_x = \frac{\sum_{i=1}^{n}(I_i\dot{\omega}_i + m_i x_i(\ddot{z}_i - g) - m_i\ddot{x}_i z_i)}{\sum_{i=1}^{n} m_i(\ddot{z}_i - g)} \quad (6.34)$$

$$SMN_y = \frac{\sum_{i=1}^{n}(I_i\dot{\omega}_i + m_i y_i(\ddot{z}_i - g) - m_i\ddot{y}_i z_i)}{\sum_{i=1}^{n} m_i(\ddot{z}_i - g)} \quad (6.35)$$

Burada; n eklem sayısını, I_i i. eklemin atalet momentini, $\dot{\omega}_i$ i ekleminin açısal ivmesini, m_i i. eklemin kütlesini, x_i, y_i ve z_i i. parçalı kütle sisteminin koordinatlarını, g yerçekimi ivmesini ve $\ddot{x}_i$, $\ddot{y}_i$ ve $\ddot{z}_i$ ise sırasıyla x, y ve z eksenlerindeki i ekleminin ivmelerini temsil eder.

Bu tür bir hesaplamada, ağırlık merkezinin hesaplanması zaman alıcıdır, robot modelinin basitleştirilmesinden etkilenmektedir ve robotun ayakları ile yer arasındaki gerçek analizden yoksundur. SMN, ayak ile yer arasındaki temas kuvvetinin ölçülmesiyle de hesaplanabilir [226]. Bunun için çok eksenli kuvvet/tork sensörleri kullanılabilir. Ancak, yüksek maliyetleri ve daha büyük boyutları nedeniyle, piyasada bulunan çok bileşenli kuvvet/tork sensörleri tercih edilmez. Düşük maliyetleri, ince formu ve kullanım kolaylıkları nedeniyle Kuvvete Duyarlı Direnç'ler kısaca KDD'ler (Force Sensitive Resistor–FSR), iki ayaklı robotun ayakları ile yer arasındaki tepki kuvvetini ölçmek için ayak tabanlarının altına yerleştirilerek ayak temas sensörü olarak kullanılır. KDD'ler, robotun dikey zemin tepki kuvveti ve ayak bileği ekleminin dönme ekseni etrafındaki momentinin ölçümlerini sağlayarak SMN'nin hesaplanması için kullanılır.

İnsansı robotlar için genellikle her ayağın dört köşesine Şekil 6.9'da gösterildiği gibi KDD'ler yerleştirilerek gerçek zamanlı olarak SMN'nin hesaplanması sağlanır.



Şekil 6.9. KDD'lerin robotun ayaklarına yerleştirilmesi

SMN, tabana yerleştirilen KDD'nin konumu ve ölçülen kuvvet değerleriyle Şekil 6.9'a göre x ve y eksenlerinde sırasıyla (6.36) ve (6.37)'deki eşitlikler kullanılarak doğrudan hesaplanabilir.

$$\mathrm{SMN_x} = \frac{\sum_{i=1}^{8} F_i p_{i,x}}{\sum_{i=1}^{8} F_i} \tag{6.36}$$

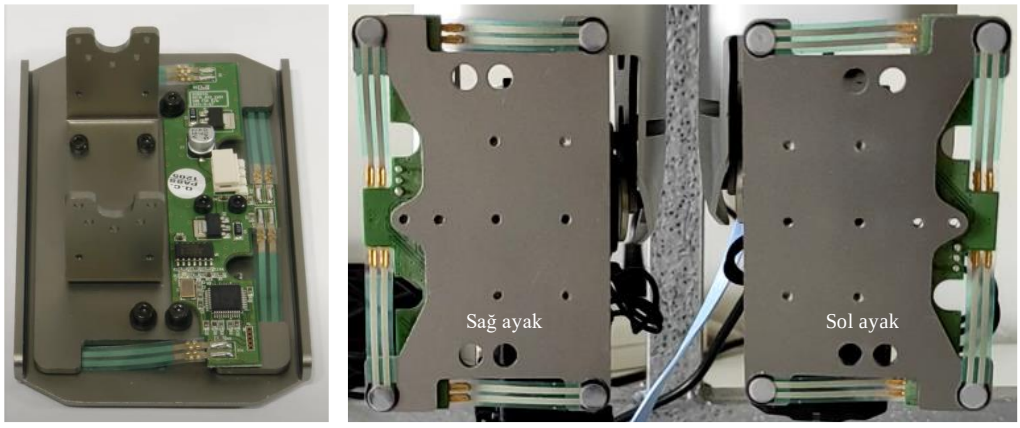
$$\mathrm{SMN_y} = \frac{\sum_{i=1}^{8} F_i p_{i,y}}{\sum_{i=1}^{8} F_i} \tag{6.37}$$

Burada; F_i i. KDD'nin kuvvet ölçümünü, $p_{i,x}$ ve $p_{i,y}$ ise i. KDD'nin referans alınan ayak bileği eklemine göre sırasıyla dikey ve yatay yöndeki mesafesidir.

Tez çalışması kapsamında, Robotis-OP2 insansı robotunun ayakları, KDD'leri içeren Şekil 6.10'daki set ile değiştirilmiştir.

Şekil 6.10. Robotis-OP2'nin KDD'li ayak seti [227]

Setin alt ve üst kapakları çıkarıldığında üstten ve alttan görünümler, Şekil 6.11'de verilmiştir.



(a) (b)

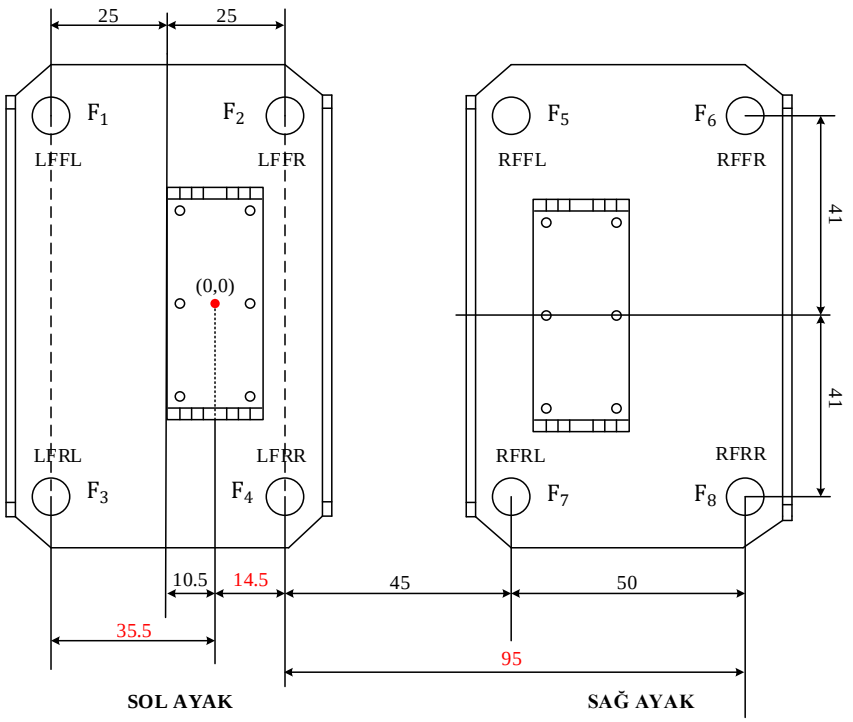
Şekil 6.11. KDD'li ayak setinin görünümleri: (a) üstten görünüm ve (b) alttan görünüm

Tablo 6.2'de özellikleri verilen KDD'ler ile devre kartından oluşan sistem, Şekil 6.11(a)'da görülmektedir. Şekil 6.11(b)'de görüldüğü gibi sette, sağ ayakta dört adet ve sol ayakta dört adet olmak üzere ayakların tüm köşelerine yerleştirilen toplam sekiz adet KDD bulunmaktadır. Ayrıca Şekil 6.11(b)'de, yer tepki kuvvetlerinin KDD'ye düzgün bir şekilde iletilebilmesi ve oluşabilecek şok darbe etkilerinin azaltılabilmesi için KDD'lerin algılama alanının kauçuk pedler ile kaplandığı görülmektedir.

Tablo 6.2. Setteki KDD'nin özellikleri

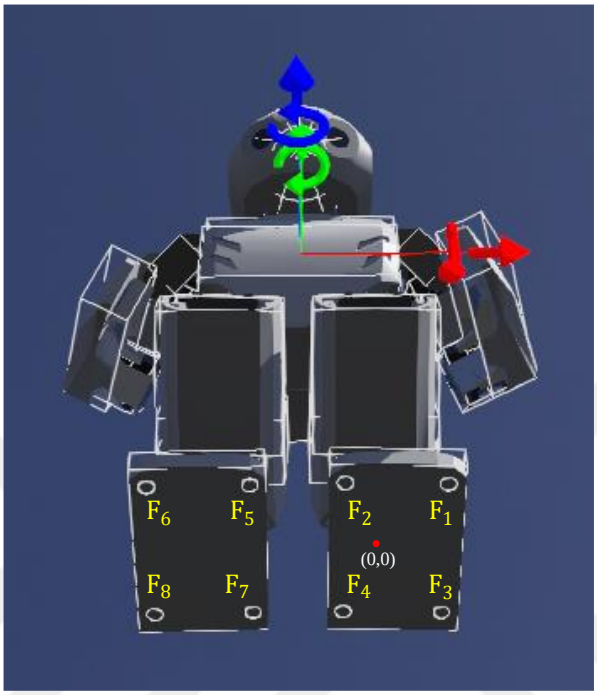
Parametre	Değer
Ölçüm menzili	0.493~65.535 N
Çalışma gerilimi	9~12 V
Çalışma sıcaklığı	-5~80 °C
Baud oranı	7843 bps~3 Mbps
Standby akımı	50 mA

Robotis-OP2'deki KDD'lerin ayak tabanındaki yerleşimi ve kuvvetlerin numaralandırılması Şekil 6.12'de verilmiştir. Sol ayak bileğinin merkezi referans alınarak SMN'nin koordinat sisteminde (0, 0) başlangıç noktası tanımlanmıştır.



Şekil 6.12. KDD'lerin yerleşimi ve kuvvetlerin numaralandırılması (üstten görünüm)

Webots simülasyon ortamında da KDD'ler Şekil 6.12'deki ölçülere göre Şekil 6.13'teki gibi yerleştirilmiş ve ayak çerçevesindeki KDD konumları Tablo 6.3'te verilmiştir.



Şekil 6.13. Webots ortamında KDD'lerin yerleşimi

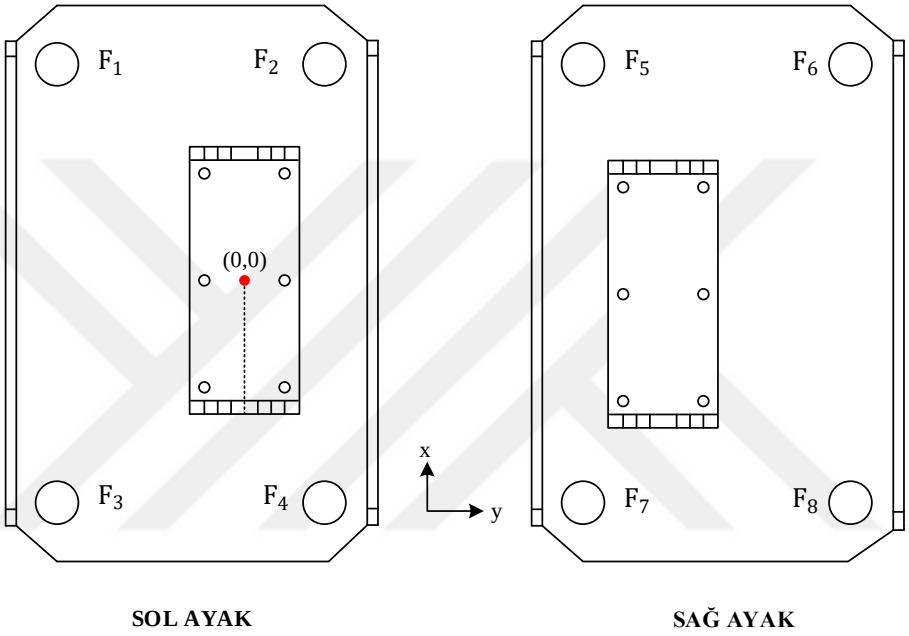
Tablo 6.3. Webots'ta ayak çerçevesindeki KDD konumları

KDD ismi	x konumu (m) [Ayak çerçevesi]	y konumu (m) [Ayak çerçevesi]	z konumu (m) [Ayak çerçevesi]
LFFL	0.0355	-0.034	0.041
LFFR	-0.0145	-0.034	0.041
LFRL	0.0355	-0.034	-0.041
LFRR	-0.0145	-0.034	-0.041
RFFL	0.0145	-0.034	0.041
RFFR	-0.0355	-0.034	0.041
RFRL	0.0145	-0.034	-0.041
RFRR	-0.0355	-0.034	-0.041

Webots ortamında oluşturulan yürüme yörüngesi için ileri-geri yön x ekseni, sağ-sol yön y eksenini temsil etmektedir. Bu yüzden SMN koordinatları, x ve y eksenleri için hesaplanmalıdır. Tek destek ve çift destek fazlarını içeren bir yürüyüş sırasında sağ ve sol ayağın birbirine göre üç durumu söz konusudur. Bunlar; ayakların aynı hizada, sol ayağın ilerde ve sağ ayağın ilerde olduğu

durumlardır. Bu durumlar göz önünde bulundurularak KDD'ler ile ölçülen sekiz kuvvet değeri ile x ve y ekseninde çevrimiçi olarak SMN koordinatları hesaplanmıştır.

Şekil 6.14'te gösterildiği gibi ayak bileklerinin x eksenindeki konumlarının eşit olduğu durumlar ($x_{sol\_ab} = x_{sağ\_ab}$) için SMN hesaplama denklemleri elde edilmiştir. x_{sol_ab}, ileri yön olan x eksenindeki sol ayak bileğinin konumunu ve $x_{sağ\_ab}$ ise x eksenindeki sağ ayak bileğinin konumudur.



Şekil 6.14. Ayakların aynı hizada olduğu durum ($x_{sol\_ab} = x_{sağ\_ab}$)

Şekil 6.12 incelendiğinde, yürüme sırasında KDD'lerin y eksenindeki birbirlerine olan uzaklıkları sabit kalır. Bu yüzden SMN'nin y eksen koordinatı (SMN_y), üç durumda da aynı eşitlik ile hesaplanır.

Sol ayaktaki kuvvetlerin (0,0) noktasına göre y eksenindeki kuvvet merkez noktası L_y, (6.38) ile hesaplanır.

$$L_y = \frac{14.5(F_2+F_4)-35.5(F_1+F_3)}{\sum_{i=1}^{4} F_i} \tag{6.38}$$

Sağ ayaktaki kuvvetlerin (0,0) noktasına göre y eksenindeki kuvvet merkez noktası R_y, (6.39) ile hesaplanır.

$$R_y = \frac{59.5(F_5+F_7)+109.5(F_6+F_8)}{\sum_{i=5}^{8} F_i} \tag{6.39}$$

SMN'nin y eksen koordinatı (SMN_y) ise (6.40) ile hesaplanır.

$$SMN_y = \frac{L_y \times \sum_{i=1}^{4} F_i + R_y \times \sum_{i=5}^{8} F_i}{\sum_{i=1}^{4} F_i + \sum_{i=5}^{8} F_i} \tag{6.40}$$

(6.40)'taki eşitlikte bilinmeyenler yerine koyulduğunda (6.41) elde edilir.

$$SMN_y = \frac{59.5(F_5+F_7)+109.5(F_6+F_8)+14.5(F_2+F_4)-35.5(F_1+F_3)}{\sum_{i=1}^{8} F_i} \tag{6.41}$$

Sol ayaktaki kuvvetlerin (0,0) noktasına göre x eksenindeki kuvvet merkez noktası L_x, (6.42) ile hesaplanır.

$$L_x = \frac{41(F_1+F_2)-41(F_3+F_4)}{\sum_{i=1}^{4} F_i} \tag{6.42}$$

Sağ ayaktaki kuvvetlerin (0,0) noktasına göre x eksenindeki kuvvet merkez noktası R_x, (6.43) ile hesaplanır.

$$R_x = \frac{41(F_5+F_6)-41(F_7+F_8)}{\sum_{i=5}^{8} F_i} \tag{6.43}$$

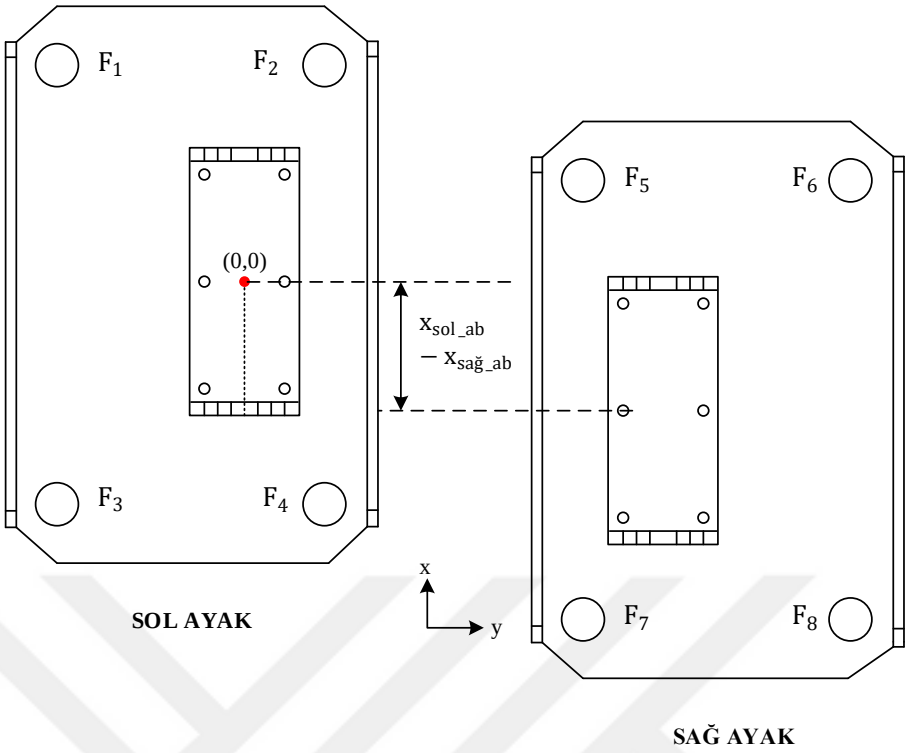
SMN'nin x eksen koordinatı (SMN_x), ayakların aynı hizada olduğu durumda (6.44)'teki eşitlik ile hesaplanır.

$$SMN_x = \frac{L_x \times \sum_{i=1}^{4} F_i + R_x \times \sum_{i=5}^{8} F_i}{\sum_{i=1}^{4} F_i + \sum_{i=5}^{8} F_i} \tag{6.44}$$

(6.44)'teki eşitlikte bilinmeyenler yerine koyulduğunda (6.45) elde edilir.

$$SMN_x = \frac{41(F_1+F_2)+41(F_5+F_6)-41(F_3+F_4)-41(F_7+F_8)}{\sum_{i=1}^{8} F_i} \tag{6.45}$$

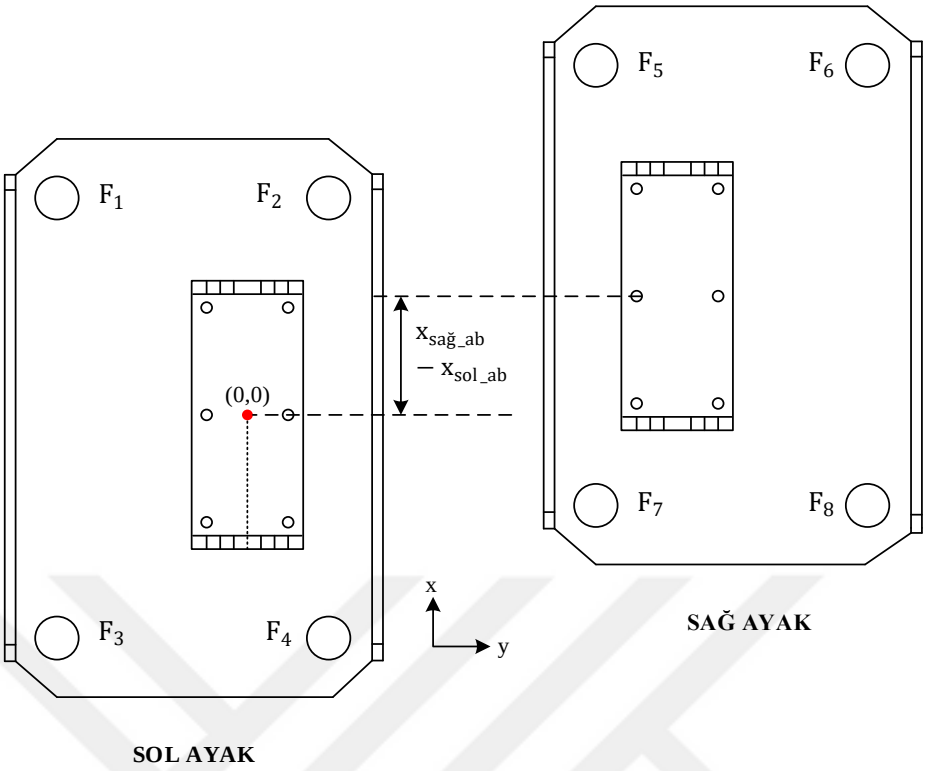
SMN'nin x eksen koordinatı (SMN_x), Şekil 6.15'teki gibi sol ayağın ileride olduğu durumda $x_{sol\_ab} - x_{sağ\_ab}$ kadar ötelemeden etkilenerek (6.46)'daki eşitlik ile hesaplanır.



Şekil 6.15. Sol ayağın ileride olduğu durum ($x_{sol\_ab} > x_{sağ\_ab}$)

$$SMN_x = \frac{41(F_1+F_2)+(41-(x_{sol\_ab}-x_{sağ\_ab}))(F_5+F_6)-41(F_3+F_4)-(41+(x_{sol\_ab}-x_{sağ\_ab}))(F_7+F_8)}{\sum_{i=1}^{8} F_i} \quad (6.46)$$

SMN'nin x eksen koordinatı (SMN_x), Şekil 6.16'daki gibi sağ ayağın ileride olduğu durumda $x_{sağ\_ab} - x_{sol\_ab}$ kadar ötelemeden etkilenerek (6.47)'deki eşitlik ile hesaplanır.



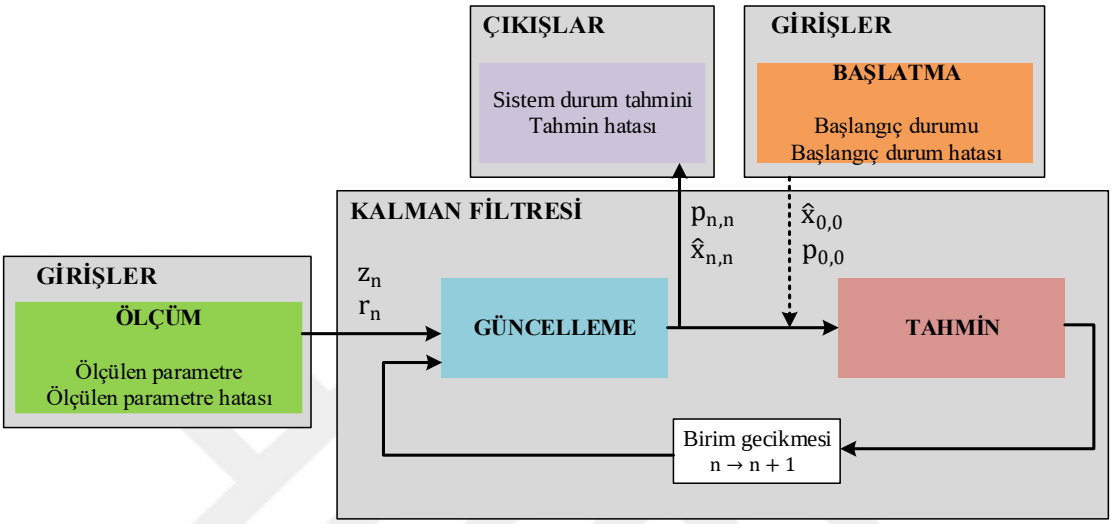
Şekil 6.16. Sağ ayağın ileride olduğu durum ($x_{sağ\_ab} > x_{sol\_ab}$)

$$SMN_x = \frac{41(F_1+F_2)+(41+(x_{sağ\_ab}-x_{sol\_ab}))(F_5+F_6)-41(F_3+F_4)-(41-(x_{sağ\_ab}-x_{sol\_ab}))(F_7+F_8)}{\sum_{i=1}^{8} F_i} \quad (6.47)$$

KDD'ler ile ölçülen kuvvet değerlerinin anlık okunan değerleri gürültüye sahip olabilmektedir. Tez çalışması kapsamında, okunan kuvvet değerlerindeki gürültüyü gidermek için bilimsel yazında sıkça kullanılan bir boyutlu Kalman filtresi [228] uygulanmıştır.

6.2.1. Bir Boyutlu Kalman Filtresi

Şekil 6.17'de blok diyagramı verilen bir boyutlu Kalman filtresi, tek ölçüm parametresine dayanan durum tahmin yöntemidir. Tablo 6.4'te de bir boyutlu Kalman filtresinin denklemleri verilmiştir.



Şekil 6.17. Bir boyutlu Kalman filtresinin blok diyagramı

Tablo 6.4. Bir boyutlu Kalman filtresinin denklemleri

Denklem	Denklem ismi
$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - \hat{x}_{n,n-1})$	Durum güncelleme
$\hat{x}_{n+1,n} = \hat{x}_{n,n} + \Delta t\hat{\dot{x}}_{n,n}$ $\hat{\dot{x}}_{n+1,n} = \hat{\dot{x}}_{n,n}$ (Sabit hız dinamikleri için)	Durum ekstrapolasyonu
$K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n}$	Kalman kazancı
$p_{n,n} = (1 - K_n)p_{n,n-1}$	Kovaryans güncelleme
$p_{n+1,n} = p_{n,n} + q_n$	Kovaryans ekstrapolasyonu

Durum güncelleme denklemi, (6.48) ile ifade edilir.

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - \hat{x}_{n,n-1}) = (1 - K_n)\hat{x}_{n,n-1} + K_n z_n \tag{6.48}$$

Burada; K_n Kalman kazancı, $\hat{x}_{n,n}$ mevcut sistem durum tahmini, $\hat{x}_{n,n-1}$ önceki sistem durum tahmini ve z_n ise ölçüm değerini temsil eder.

(6.48)'deki eşitlikten görüldüğü gibi Kalman kazancı K_n, ölçüme verilen ağırlıktır. $(1 - K_n)$ ise tahmine verilen ağırlıktır. Ölçüm hatası çok büyük olduğunda ve tahmin hatası çok küçük olduğunda, Kalman kazancı sıfıra yakındır. Dolayısıyla, tahmine büyük, ölçüme küçük bir ağırlık verilir.

Kalman kazancı, 0 ile 1 arasında bir sayıdır ve (6.49) ile hesaplanır.

$$K_n = \frac{p_{n,n-1}}{p_{n,n-1} + r_n} \tag{6.49}$$

Burada; K_n Kalman kazancı, $p_{n,n-1}$ önceki filtre tahmini sırasında hesaplanan tahmin hatası ve r_n ise ölçüm hatasıdır.

Tahmin hatası güncelleme denklemi, (6.50)'deki gibidir.

$$p_{n,n} = (1 - K_n)p_{n,n-1} \tag{6.50}$$

Burada; K_n Kalman kazancı, $p_{n,n-1}$ önceki filtre tahmini sırasında hesaplanan tahmin hatası ve $p_{n,n}$ ise mevcut durumun tahmin hatasıdır.

(6.50), mevcut durumun tahmin hatasını günceller ve "kovaryans güncelleme" denklemi olarak adlandırılır. $(1 - K_n) \leq 1$ olduğundan her filtre yinelemesinde tahmin hatası her zaman küçülür. Ölçüm hatası büyük olduğunda, Kalman kazancı düşük olacaktır, bu nedenle tahmin hatasının yakınsaması yavaş olacaktır. Bununla birlikte, ölçüm hatası küçük olduğunda, Kalman kazancı yüksek olacak ve tahmin hatası hızla sıfıra yakınlaşacaktır.

Kovaryans ekstrapolasyon denklemi, işlem gürültüsü değişkeni ile güncellenmelidir. Gerçek bir dünyada sistem dinamik modelinde hatalar vardır. Dinamik modelin hatasına işlem gürültüsü denir. İşlem gürültüsü tahmin hataları üretir. İşlem gürültüsü varyansı, q ile belirtilir. Kovaryans ekstrapolasyon denklemi işlem gürültü varyansını içerecektir. Sabit dinamikler için kovaryans ekstrapolasyon denklemi (6.51)'deki gibidir.

$$p_{n+1,n} = p_{n,n} + q_n \tag{6.51}$$

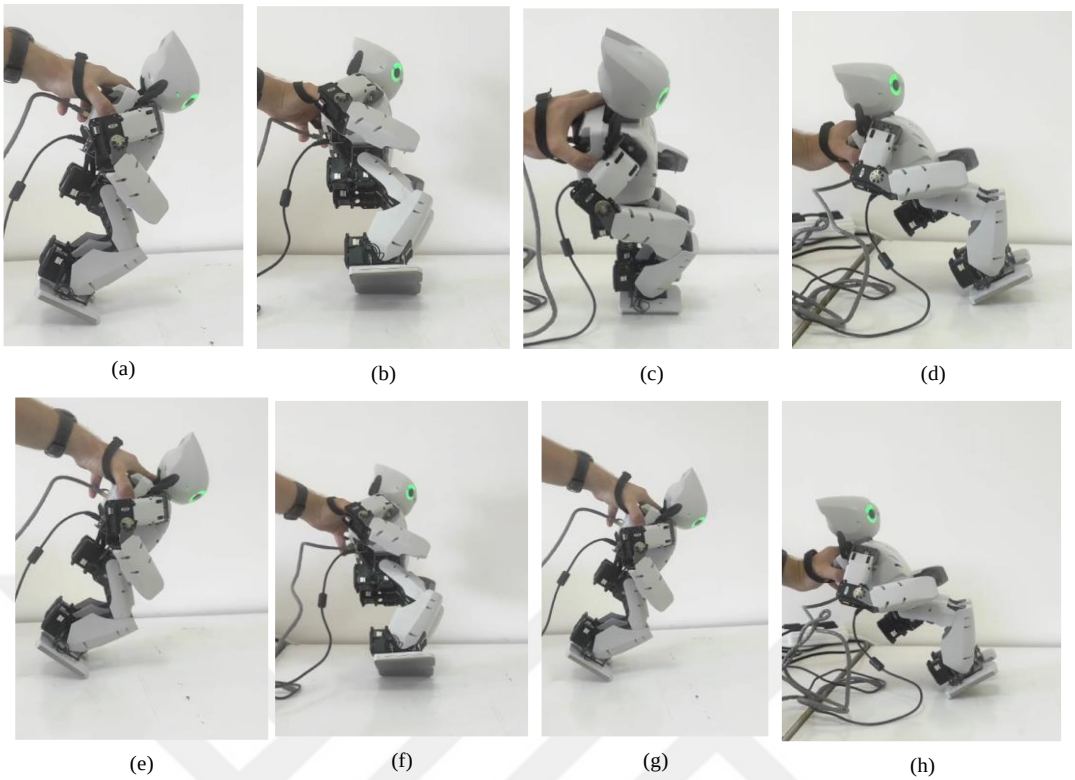
7. DENEYSEL ÇALIŞMALAR

Önceki bölümde, Robotis-OP2 insansı robotunun yönelim açılarının düzgün bir şekilde hesaplanabilmesi için sensör birleştirme yöntemlerinden tamamlayıcı filtre ve Kalman filtresi ayrı ayrı anlatılmıştır. Bu tez kapsamında gerçekleştirilen deneysel çalışmalar kapsamında ilk olarak, optimum filtre parametrelerinin belirlenebilmesi ve hangi filtrenin daha doğru sonuçlar verdiğinin gözlemlenebilmesi için hem gerçek robot üzerinde hem de simülasyon ortamında çeşitli deneysel çalışmalar gerçekleştirilmiştir. İkinci olarak, yürüyüş kararlılık ölçütünün doğru bir şekilde hesaplanabilmesi için KDD'lerden okunan kuvvet değerleri incelenerek meydana gelen gürültüleri gidermek için bir boyutlu Kalman filtresinin parametreleri ayarlanmıştır. Üçüncü olarak, kararlı yürüme için Düello ÇDQA kullanılarak önerilen çerçeve ile optimum parametreler edilmiş ve önerilen vücut duruş dengeleme ile birleştirilerek yürüme sonuçları gözlemlenmiştir. Dördüncü ve son olarak, PID ve DPÖ algoritmalarından olan DDPG'nin kontrolör olarak ayrı ayrı kullanılmasıyla eğimli yüzeylerde yürüme için deneysel çalışmalar gerçekleştirilmiştir.

7.1. Yönelim Açılarının Hesaplanması için Gerçekleştirilen Çalışmalar

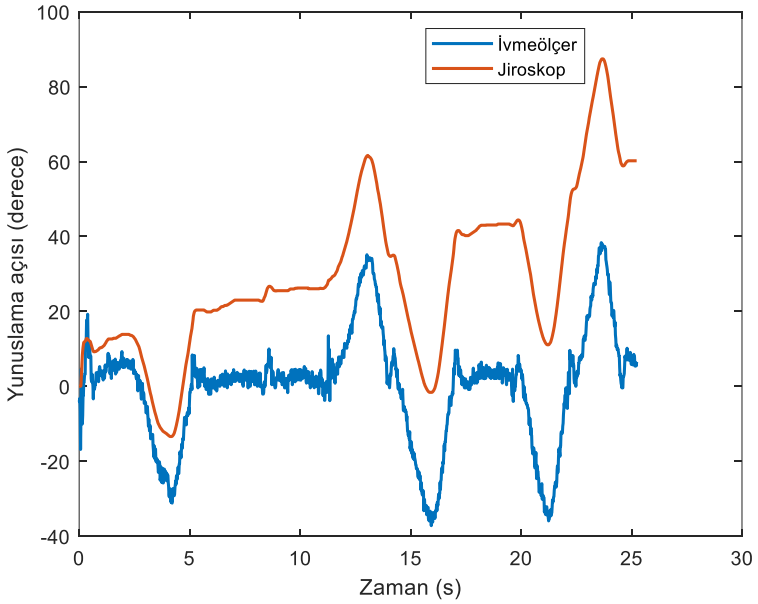
Optimum filtre parametrelerinin belirlenebilmesi ve hangi filtrenin daha doğru sonuçlar verdiğinin gözlemlenebilmesi için hem gerçek robot üzerinde hem de simülasyon ortamında çeşitli deneysel çalışmalar gerçekleştirilmiştir.

İlk olarak gerçek bir Robotis-OP2 insansı robotu sabit bir şekilde duruyorken sırasıyla Şekil 7.1'deki gibi elle bazı yunuslama ve yuvarlanma hareketleri yaptırılarak ivmeölçer ve jiroskoptan veriler alınmıştır.

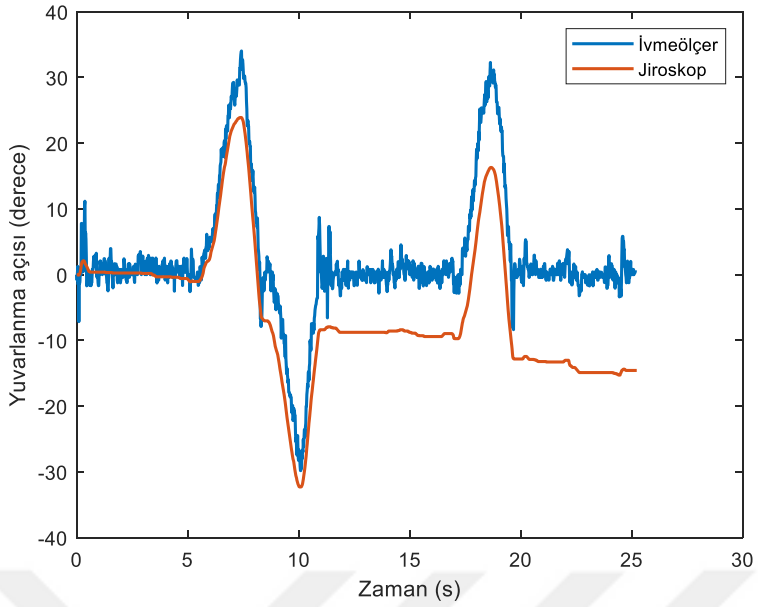


Şekil 7.1. Elle yaptırılan bazı yunuslama ve yuvarlanma hareketleri: (a, d, e, f) yunuslama ve (b, c, g, h) yuvarlanma

Alınan veriler ile ayrı ayrı hesaplanan yunuslama ve yuvarlanma açılarının zamanla değişimi sırasıyla Şekil 7.2 ve Şekil 7.3'te verilmiştir.



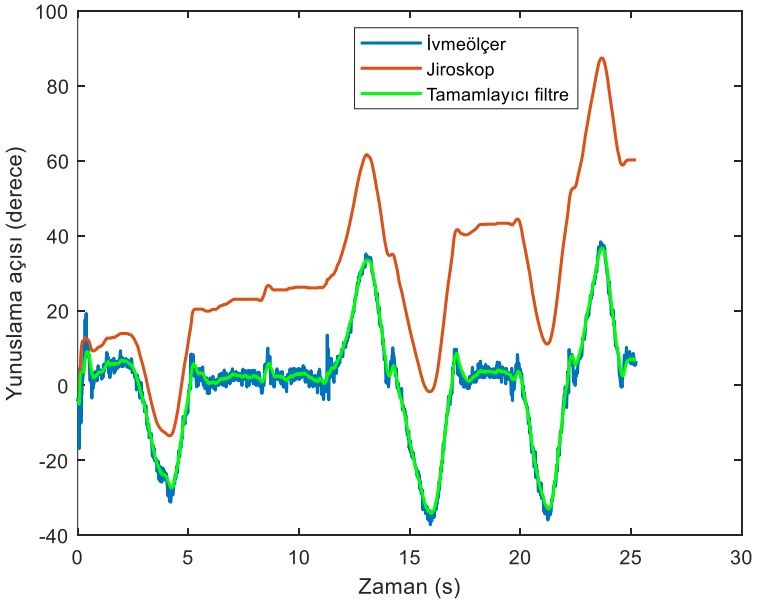
Şekil 7.2. İvmeölçer ve jiroskop ile hesaplanan yunuslama açılarının karşılaştırılması



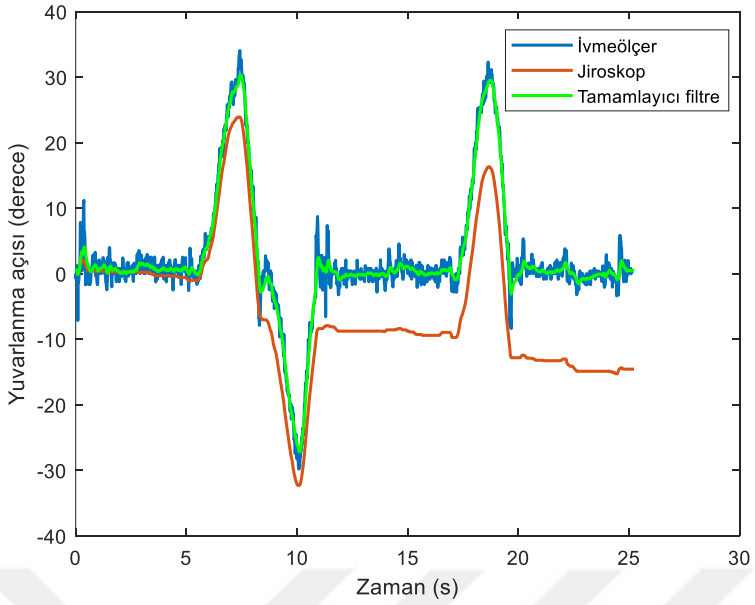
Şekil 7.3. İvmeölçer ve jiroskop ile hesaplanan yuvarlanma açılarının karşılaştırılması

Şekil 7.2 ve Şekil 7.3'te görüldüğü gibi ivmeölçer ile hesaplanan açılarda gürültüler, jiroskop ile hesaplanan açılarda ise zamanla kaymalar mevcuttur. Gerçekleştirilen deneysel çalışmalarda, ivmeölçer ile hesaplanan açılar göz önünde bulundurularak tamamlayıcı filtre için filtre katsayısı $\alpha = 0.96$ olarak belirlenmiş ve böylece zaman sabiti değeri $\tau = 0.384$ olarak seçilmiştir.

Şekil 7.2 ve Şekil 7.3'te ivmeölçer ve jiroskop ile hesaplanan açılara tamamlayıcı filtre uygulanmasıyla elde edilen sonuçlar sırasıyla Şekil 7.4 ve Şekil 7.5'te verilmiştir.

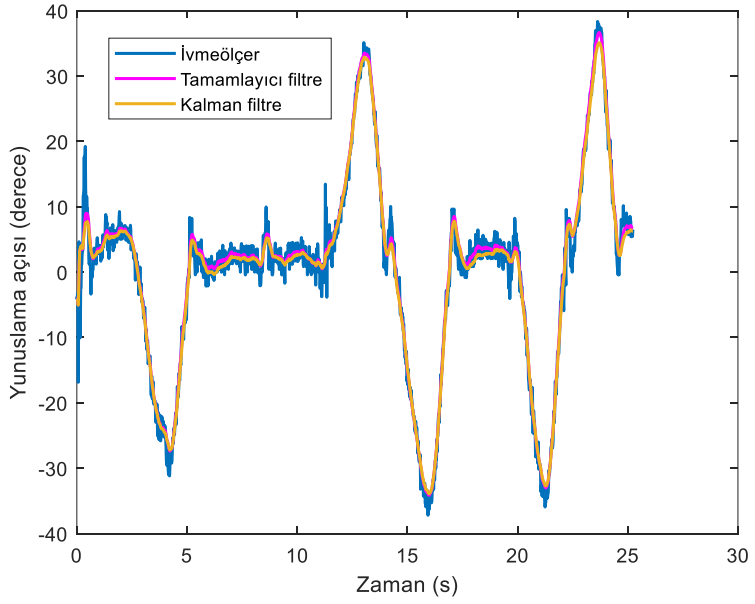


Şekil 7.4. İvmeölçer, jiroskop ve tamamlayıcı filtre ile hesaplanan yunuslama açılarının karşılaştırılması

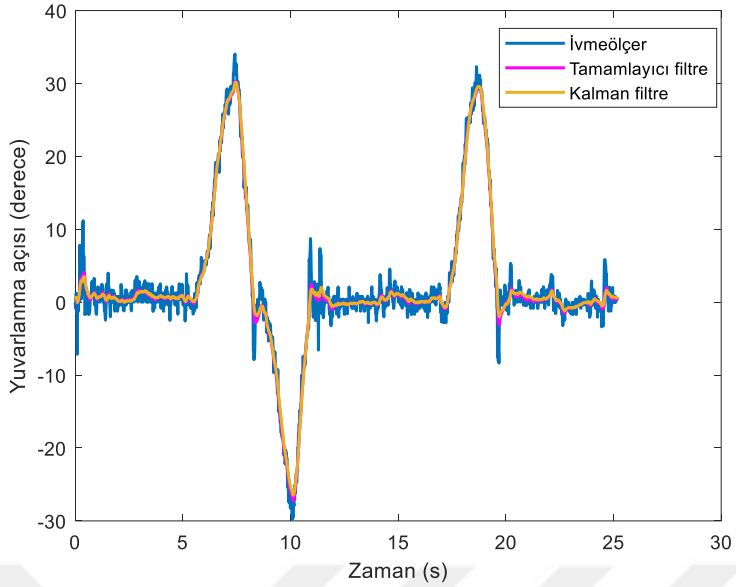


Şekil 7.5. İvmeölçer, jiroskop ve tamamlayıcı filtre ile hesaplanan yuvarlanma açılarının karşılaştırılması

Gerçekleştirilen deneysel çalışmalarda ivmeölçer ile hesaplanan açılar göz önünde bulundurularak Kalman filtresi için ivmeölçer, bias ve ölçüm gürültü varyansları sırasıyla $Q_\theta = 0.001$, $Q_{\dot{\theta}_b} = 0.003$ ve $R = 0.03$ olarak belirlenmiştir. Tamamlayıcı filtre için gerçekleştirildiği gibi Şekil 7.2 ve Şekil 7.3'teki ivmeölçer ve jiroskop ile hesaplanan açılara Kalman filtresi uygulanarak elde edilen sonuçlar sırasıyla Şekil 7.6 ve Şekil 7.7'de karşılaştırılmıştır.

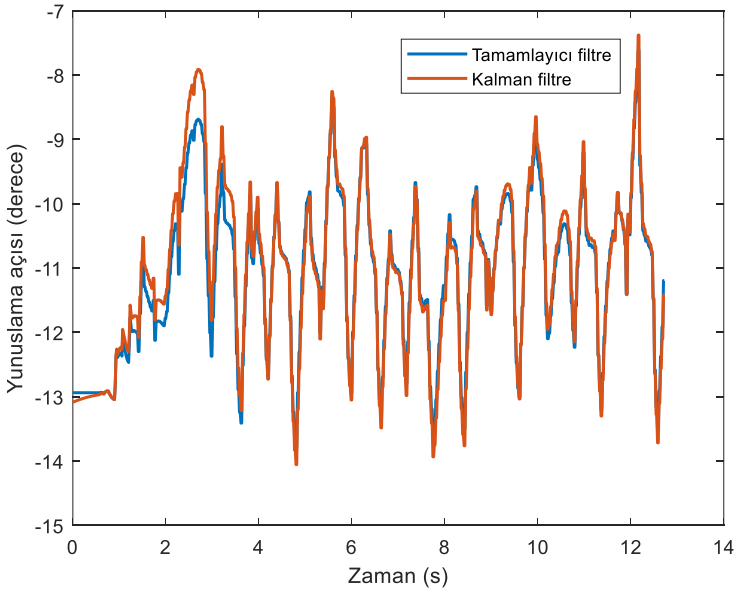


Şekil 7.6. İvmeölçer, tamamlayıcı filtre ve Kalman filtresi ile hesaplanan yunuslama açılarının karşılaştırılması

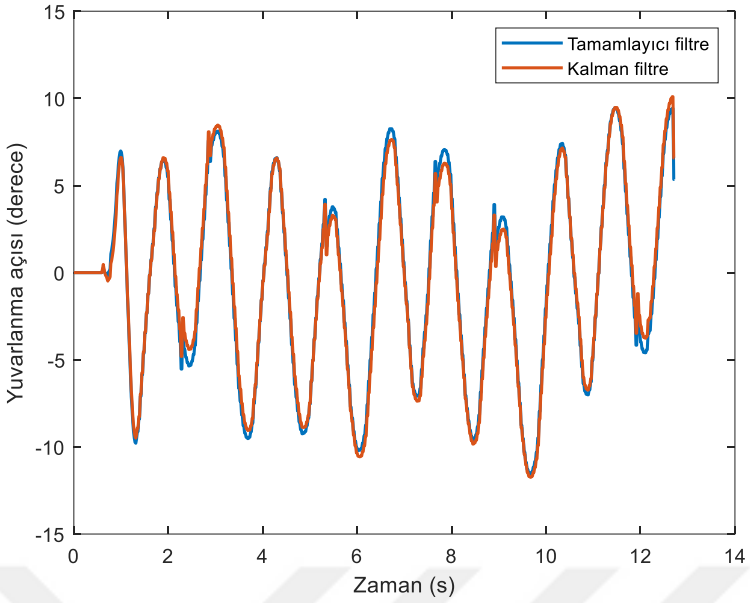


Şekil 7.7. İvmeölçer, tamamlayıcı filtre ve Kalman filtresi ile hesaplanan yuvarlanma açılarının karşılaştırılması

Şekil 7.6 ve Şekil 7.7'deki sonuçlara göre tamamlayıcı ve Kalman filtresi sonuçlarının oldukça birbirine yakın olduğu görülmüştür. Ayrıca Robotis-OP2 için Webots'ta hazır bulunan robotun yürüyüş birimindeki algoritması kullanılarak yönelim açılarının hesaplanması ile ilgili simülasyon ortamında da deneysel çalışmalar gerçekleştirilmiştir. Robotun kendi yürüme algoritması ile Webots ortamında yürümesi esnasındaki yunuslama ve yuvarlanma açılarının tamamlayıcı filtre ve Kalman filtresi ile hesaplanması sırasıyla Şekil 7.8 ve Şekil 7.9'daki gibidir.



Şekil 7.8. Tamamlayıcı filtre ve Kalman filtresi ile hesaplanan yunuslama açılarının karşılaştırılması



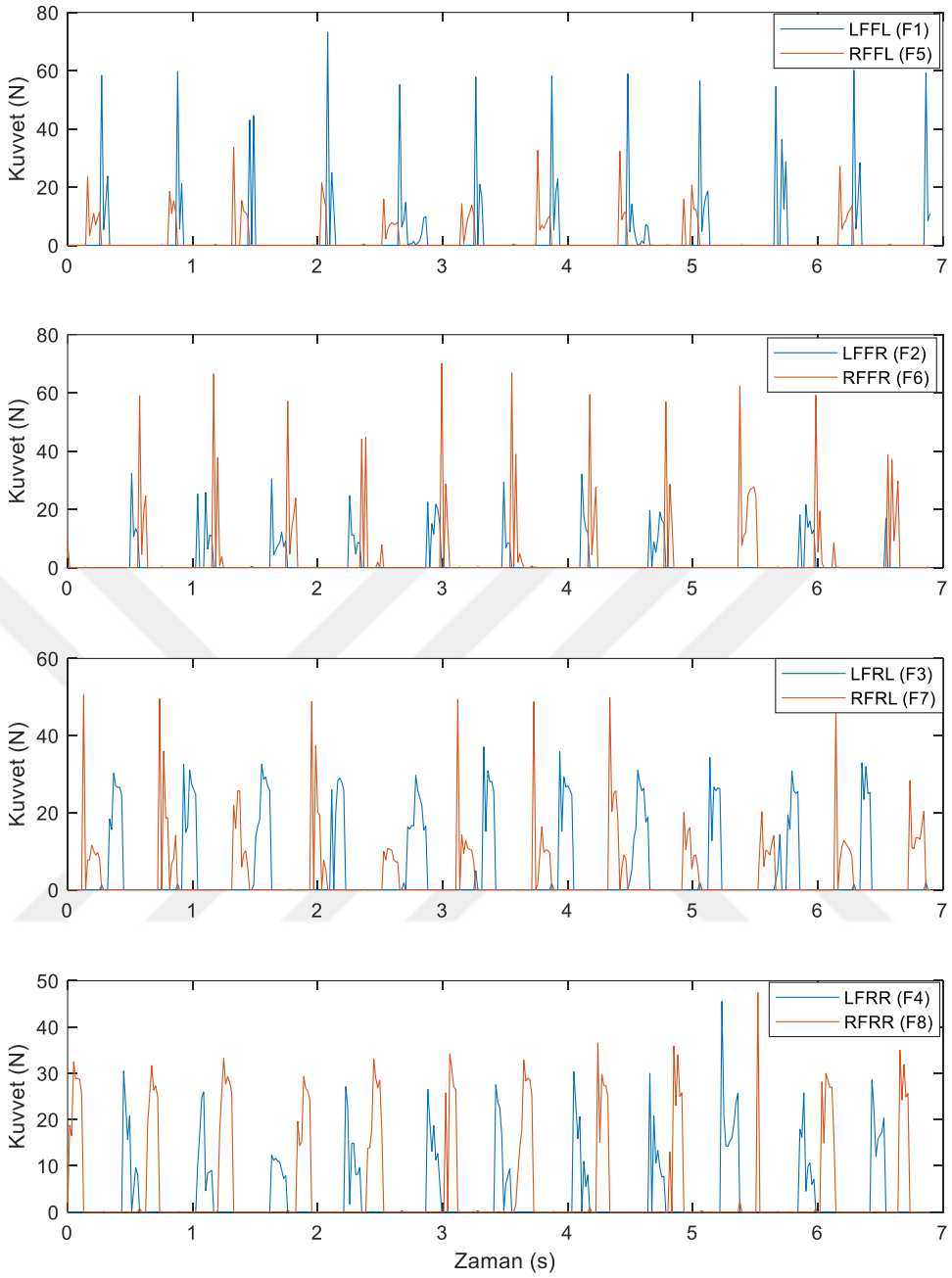
Şekil 7.9. Tamamlayıcı filtre ve Kalman filtresi ile hesaplanan yuvarlanma açılarının karşılaştırılması

Şekil 7.8 ve Şekil 7.9'daki sonuçlar gözlemlendiğinde, Kalman filtresi ile elde edilen sonuçların tamamlayıcı filtre sonuçlarına oldukça yakın olduğu görülmüştür. Kalman filtresinin ayarlanması gereken birkaç parametresi varken, tamamlayıcı filtrenin ayarlanması gereken yalnızca bir filtre parametresi olduğu için tamamlayıcı filtreyi ayarlamak daha kolaydır [229]. Ancak, tamamlayıcı filtre ve Kalman filtresi ile ilgili çalışmalarda dinamik etkiler altında Kalman filtresi, tamamlayıcı filtreye göre daha doğru sonuçlar vermektedir [230]. Yürüme işlemi birçok dinamik değişikliği de barındırdığı için sensör birleştirme yöntemi olarak Kalman filtresinin kullanımı tercih edilmiştir.

7.2. Ayak Temas Kuvvetlerinin Filtrelenmesi için Gerçekleştirilen Çalışmalar

Robotun yürüyüş kararlılık ölçütünün düzgün bir şekilde hesaplanabilmesi için robotun ayak tabanlarındaki KDD'lerden okunan temas kuvvetlerinde oluşan gürültülerin anlık olarak giderilmesi gerekmektedir.

Webots ortamında robotun yürüyüş birimindeki algoritması kullanılarak gerçekleştirilen yürüyüş sırasında KDD'lerden okunan sol ve sağ ayağın temas kuvvetleri kaydedilip MATLAB ortamında Şekil 7.10'daki gibi görselleştirilmiştir.

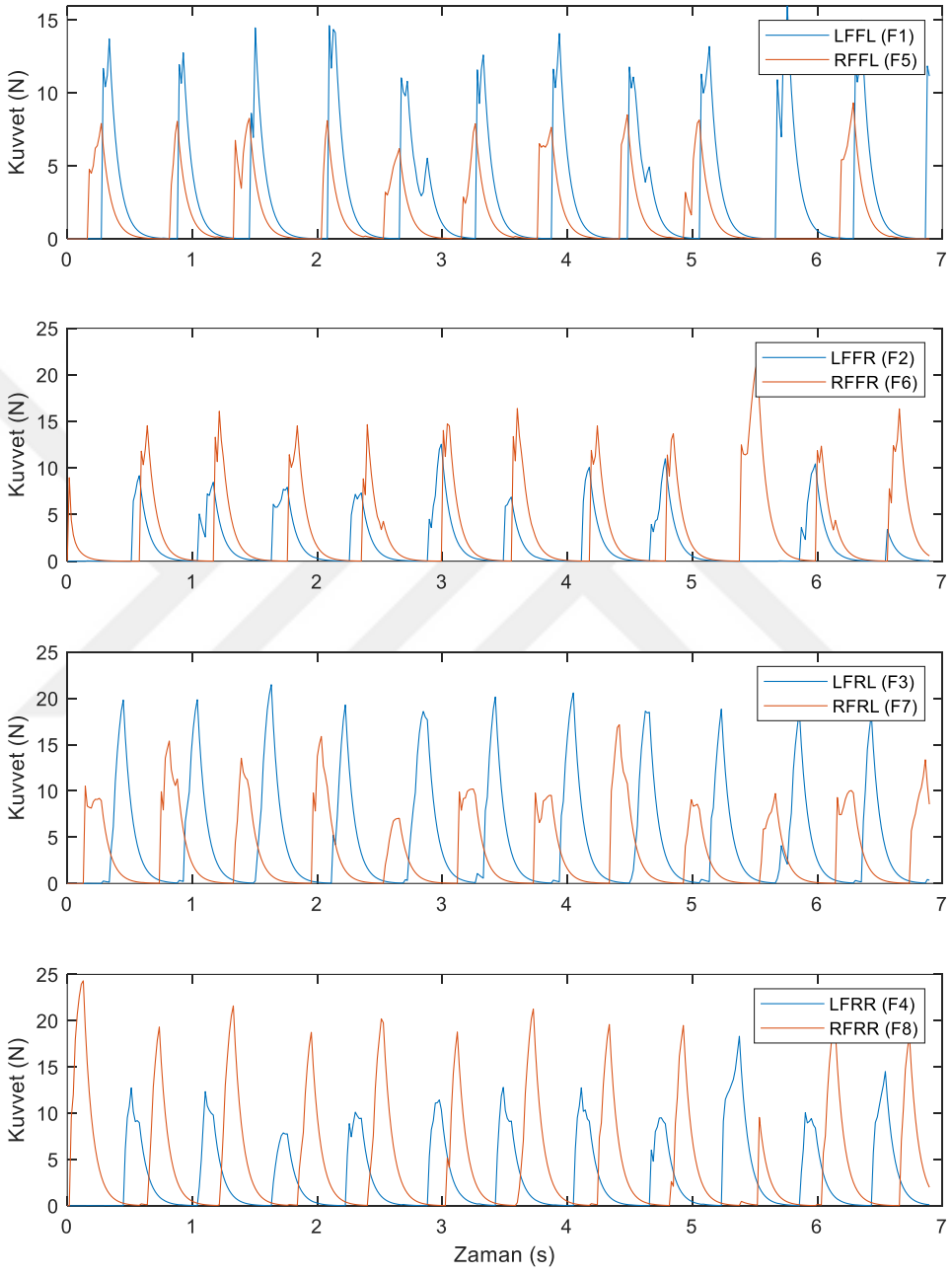


Şekil 7.10. Yürüyüş sırasında KDD'lerden okunan sol ve sağ ayağın temas kuvvetleri

Şekil 7.10'dan görüldüğü üzere kuvvet değerleri gürültüye sahiptir. Robotun kütlesi 3 kg olduğu için KDD'lerden okunan anlık kuvvet değerlerinin toplamı, $W = mg$ formülünden robotun yaklaşık ağırlık kuvveti 29.43 N'e eşit çıkmalıydı. Ancak, okunan kuvvet değerlerindeki gürültüden dolayı istenen değer elde edilememiştir. Kuvvet değerlerindeki gürültüyü gidermek için bir önceki bölümde bahsedilen bir boyutlu Kalman filtresi uygulanmıştır.

Gerçekleştirilen deneysel çalışmalarda, işlem gürültü, ölçüm gürültü ve tahmini hata varyanslarının başlangıç değerleri sırasıyla $q = 0.001$, $r = 0.03$, $p = 1$ olarak belirlenmiştir.

Bir boyutlu Kalman filtresinden geçirilen kuvvet değerleri, Şekil 7.11'deki gösterildiği gibi düzeltilmiştir.

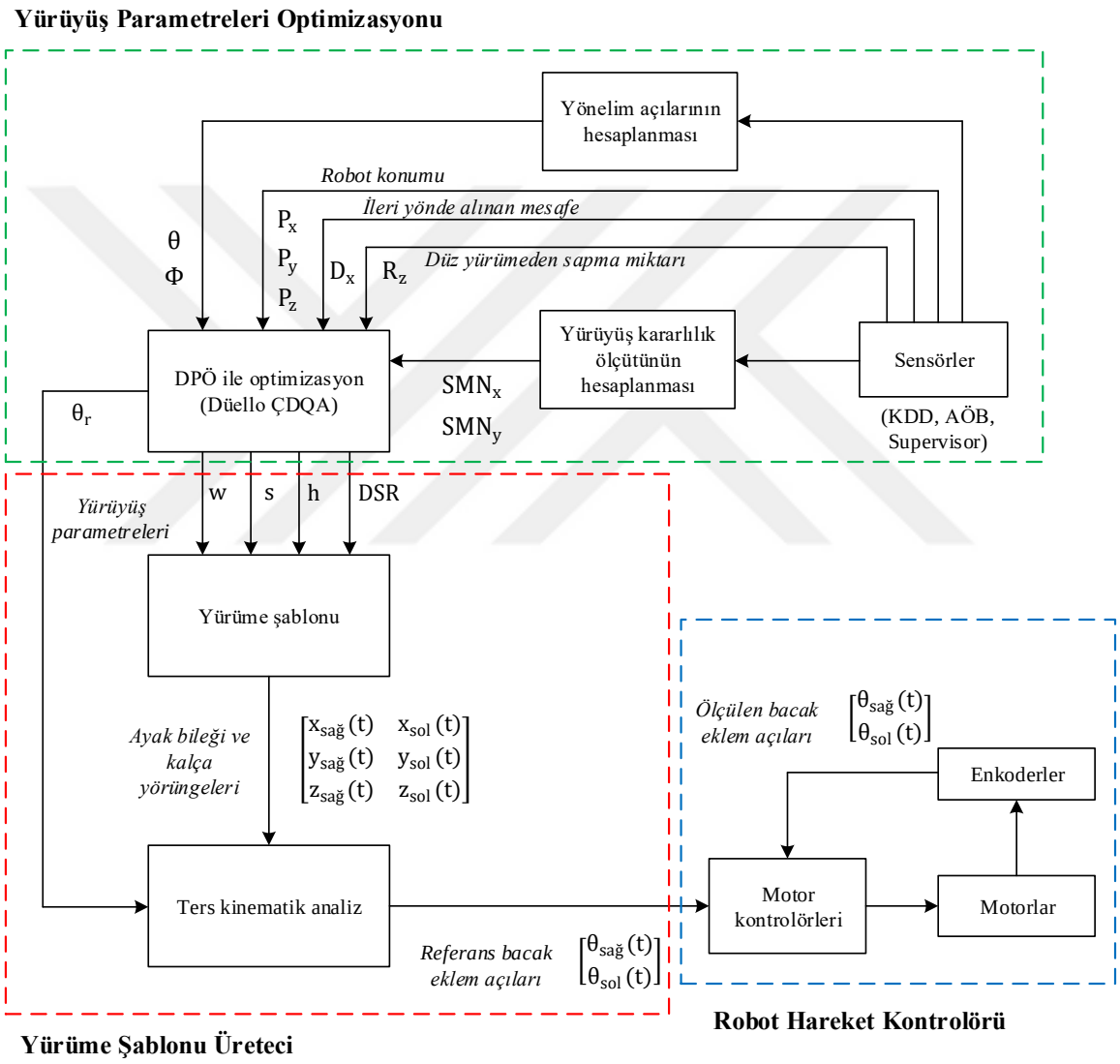


Şekil 7.11. Bir boyutlu Kalman filtresinden geçirilen KDD değerleri

Şekil 7.11'de gösterildiği gibi kuvvet değerleri, bir boyutlu Kalman filtresinden geçirilerek x ve y eksenlerindeki SMN koordinatları sırasıyla SMN_x ve SMN_y'nin hesaplanabilmesi için hazır hale getirilmiştir.

7.3. Kararlı Yürüme için Önerilen Çerçeve

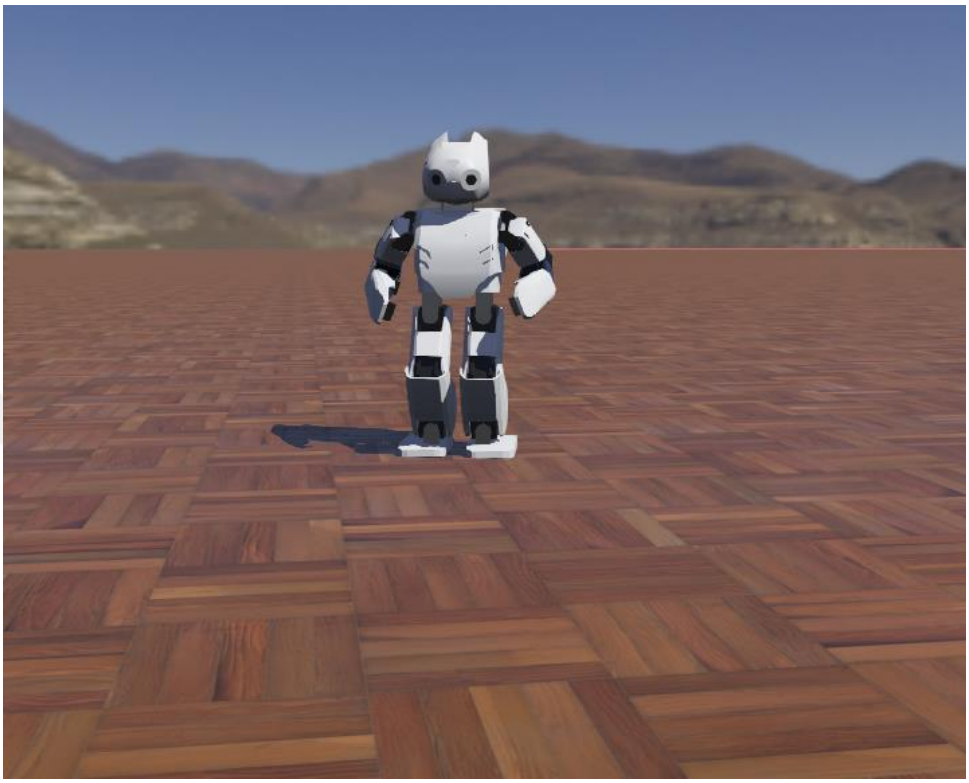
Tez kapsamında gerçekleştirilen çalışmalar için temel olarak Robotis-OP2 insansı robotunun düşmeden kararlı bir şekilde yürümesinin sağlanması amaçlanmıştır. Bu amaç için en önemli görev, yürüyüş şablonu üretici ile oluşturulan yörüngelerin optimum yürüyüş parametrelerini elde edebilmektir. Kararlı yürüme için önerilen çerçeve Şekil 7.12'de verilmiştir.



Şekil 7.12. Kararlı yürüme için önerilen çerçeve

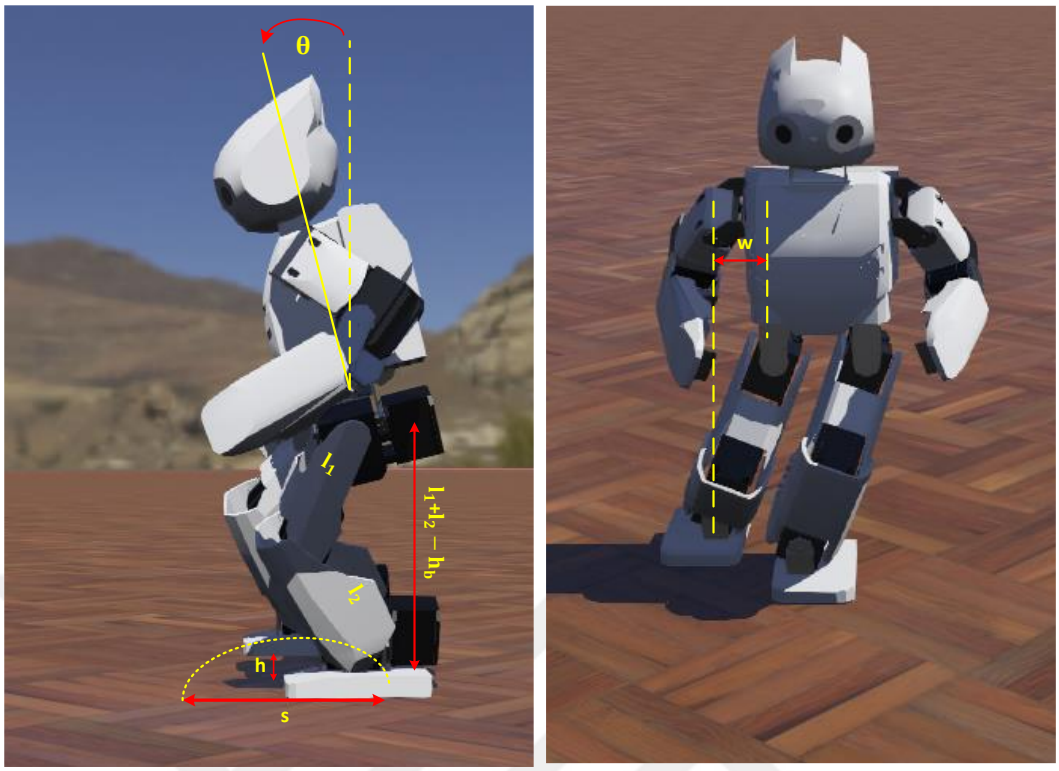
Şekil 7.12'de görüldüğü gibi düz yürüme işleminin kararlı bir şekilde gerçekleştirilebilmesi için yürüyüş parametrelerinin optimizasyon işlemi, Bölüm 4.2.4'te detaylandırılan DPÖ algoritmalarından Düello ÇDQA ile gerçekleştirilmiştir.

Düello ÇDQA'nın eğitimi, Webots simülatörü üzerinde Robotis-OP2 insansı robotunu ve düz bir zemini içeren Şekil 7.13'teki çevrede gerçekleştirilmiştir.



Şekil 7.13. Webots simülatöründe oluşturulan çevre

İki ayaklı robot yürüyüşü için oluşturulan ayak bileği ve kalça yörüngelerinde toplam 6 parametre (h_b, w, s, h, T, DSR) bulunmaktadır. Tüm parametreleri optimize etmek için herhangi bir DPÖ algoritmasını kullanmak, eylem uzayının boyutu çok yüksek olduğundan oldukça zordur. Bu yüzden, robotun kinematik yapısı göz önünde bulundurularak parametrelerin alabileceği değerler belirlendikten sonra eylem uzayı sadeleştirilmelidir. Şekil 7.14'te gösterilen yürüyüş parametrelerinin en yüksek sınırları, Webots ortamındaki kinematik hesaplamalar ve gözlemler sonucunda yaklaşık olarak Tablo 7.1'deki gibi belirlenmiştir.



Şekil 7.14. Yürüyüş parametrelerinin gösterimi

Tablo 7.1. Yürüyüş parametrelerinin en yüksek sınırları

Parametre	En yüksek sınır (mm)
h_b	110
w	75
s	100
h	65

Robot sabit bir şekilde yürüdüğünde, kalça yörüngesinden anlaşılacağı üzere kütle merkezinin yüksekliği her zaman sabittir. Ancak, hareket esnasındaki gürültülerden dolayı ihmal edilebilecek küçüklükte bir değişiklik aralığında olabilmektedir. Robotun diz eklemini fazla bükmeden enerji tüketimini azaltmak ve eylem uzayını sadeleştirmek için sabit olarak $h_b = 25$ mm belirlenmiştir.

Yürüme hızının artmasıyla ayağın geometrik merkezi etrafında SMN konumunda büyük bir salınım olabilmektedir. Bu da robotun kararlılığını azaltır. Robotun kararlılığını arttırmak, yani SMN salınımlarını azaltmak için robotun hızının düşürülmesi gerekmektedir. Yarı statik yürümeye benzer olarak oluşturulan yürüme çevriminde atalet kuvvetlerinin ihmal edilebilir olması için yürüme hızının düşük olması önemlidir. Bu yüzden, yürüme çevrimi zamanı sabit olarak $T = 2$ s olarak seçilmiştir.

Ayrıca, yerçekimi robotun eklemlerinde istenmeyen torklar oluşturur. Robot yürüme esnasında çift ayak destekten tek ayak desteğe geçtiğinde yerçekiminin etkisi daha belirgin hale gelir. İki ayak tarafından desteklenen robot ağırlığı, birdenbire sadece destek ayağı tarafından desteklenir. Tez çalışması kapsamında kararlı bir yürüyüş için oluşturulan 8 fazlı yürüme çevriminde çift ayak desteğe önem verilmiştir. Bu nedenle, DSR parametresi çok küçük seçilemez.

7.3.1. Düello ÇDQA ile Eğitim

Ağ eğimi ile ilgili deneysel çalışmalar için öncelikle, Bölüm 4.2'de bahsedilen DQA ve ÇDQA gibi yöntemler incelenmiştir. Ancak, bu yöntemlerin durum değerlerinin fazla tahmin edilmesi sorununun üstesinden gelmek için vakit kaybetmeden Düello ÇDQA'nın doğrudan kullanımı benimsenmiştir. Düello ÇDQA, geleneksel DQA ile karşılaştırıldığında durum değerinin fazla tahmin edilmesi sorununu çözmek için iki iyileştirmeye sahiptir. İlk olarak, Düello ÇDQA $a_{max} = \arg\max_a Q_1(s', a; \theta)$ ile bir sonraki adım için bir eylem seçmek üzere bir Q_1 değerleme ağı kullanan bir çift ağ yapısı kullanır. Daha sonra a_{max}, aşırı tahmini azaltmak için bir hedef ağ Q_2 tarafından (7.1) ile değerlendirilir.

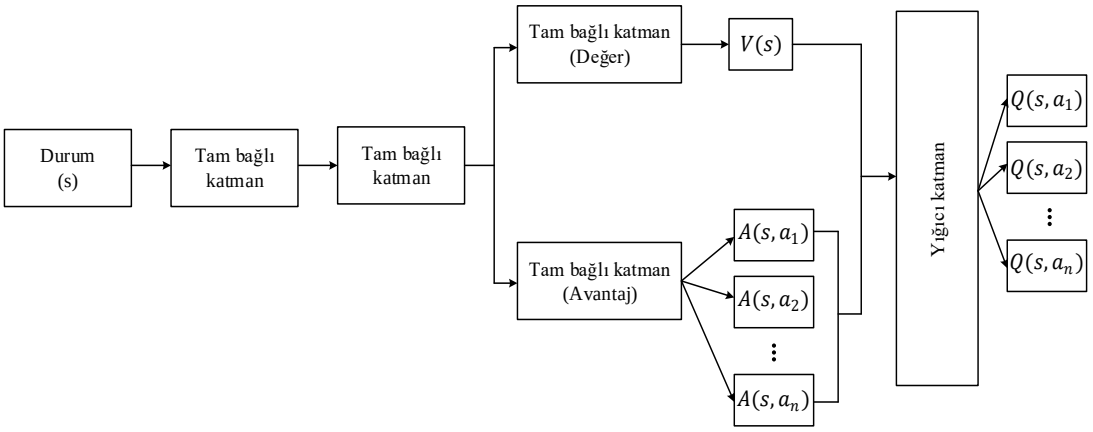
$$\begin{cases} a_{max} = \arg\max_a Q_1(s', a; \theta) \\ y = r + \gamma Q_2(s', a_{max}; \theta^-) \end{cases} \tag{7.1}$$

İkinci olarak, Düello ÇDQA, $Q_\pi(s, a)$ eylem-değer fonksiyonunu (7.2)'deki gibi iki kısma ayırır: Sadece durumla ilgili bir durum değeri fonksiyonu $V_\pi(s)$ ve hem durum hem de eylem ile ilgili bir avantaj fonksiyonu $A_\pi(s, a)$.

$$Q_\pi(s, a; \theta, \alpha, \beta) = V_\pi(s; \theta, \alpha) + A_\pi(s, a; \theta, \beta) \tag{7.2}$$

θ, $V_\pi(s)$ ve $A_\pi(s, a)$'in ortak parametreleridir. α ve β, sırasıyla V_π (s) ve $A_\pi(s, a)$'nın özgün parametreleridir.

Kullanılan Düello ÇDQA'nın yapısı Şekil 7.15'te gösterilmiştir.



Şekil 7.15. Düello ÇDQA'nın yapısı

Kararlı insansı yürüme uygulaması için oluşturulan Düello ÇDQA, optimal politikayı Algoritma 7.1'i gerçekleştirerek öğrenir.

Algoritma 7.1. Kararlı insansı yürüme uygulaması için Düello ÇDQA

Kararlı insansı yürüme sistemini başlat (Çevre)
N kapasiteli $\mathcal{D}$ tekrar tamponunu başlat
Rastgele θ parametreleriyle Q_1 değerleme ağını başlat
$\theta^- \leftarrow \theta$ parametreleriyle Q_2 hedef ağını başlat
Rastgele politika tarafından üretilen veri ile tekrar tamponunu doldur
for $bölüm = 1, M$ **do**
  Çevreyi yeniden başlat
  while $sonlandırma\ yok$ **do**
    ϵ olasılıkla rastgele bir a eylemi seç, aksi halde $a = \arg\max_a Q_1(s_t, a; \theta)$ eylemi seç
    a eylemini gerçekleştir, r ödülü ve s' bir sonraki durumu al
    $\langle s, a, r, s' \rangle$ deneyim demetini $\mathcal{D}$'ye sakla
    $\mathcal{D}$'den rastgele mini yığını örnekle
    if $j + 1\ adımında\ sonlandırma\ var$ **then**
      $Y_j = r_j$
    else
      $a_{max} = \arg\max_a Q_1(s', a; \theta)$
      $Y_j = \gamma \max_{a'} Q_2(s', a_{max}; \theta^-)$
    $\hat{Y}_j = Q_1(s, a; \theta)$
    θ parametresini güncellemek üzere Q_1'i eğitmek için kaybı en aza indir
    Her C adımında hedef ağı güncelle: $\theta^- \leftarrow \theta$
  if $Q_1\ yakınsar$ **then**
    $Q_2 = Q_1$
    $Q_{optimal} = Q_2$
    break

Durum Uzayı

Durum uzayı, Tablo 7.2'de verildiği gibi kararlı yürüme görevi için önem arz eden durumlar içeren 9 boyutlu bir diziden oluşmaktadır.

Tablo 7.2. Durum uzayı

Durum	Boyut
Yönelim açıları (Yunuslama (θ) ve Yuvarlanma (Φ))	2
Konum (P_x, P_y, P_z)	3
z ekseninde dönme (R_z)	1
Sıfır Moment Noktası (SMN_x, SMN_y)	2
x ekseninde (ileri yön) alınan mesafe (D_x)	1

Burada; θ ve Φ robot gövdesinin sırasıyla Kalman filtresi kullanılarak hesaplanan yunuslama ve yuvarlanma açılarını, P_x, P_y ve P_z sırasıyla robotun x, y ve z eksenlerindeki konumunu, R_z robotun z eksenindeki dönmesini, SMN_x ve SMN_y sırasıyla x ve y eksenlerindeki SMN'leri ve D_x ise robotun ileri yöndeki yer değiştirmesini ifade etmektedir.

Eylem Uzayı

Daha önce belirtildiği gibi 6 yürüyüş parametresinden $h_b = 25$ mm ve $T = 2$ s sabit olarak belirlenmiştir. Kalan 4 yürüyüş parametresinin aralıkları, kararlı bir yürüyüş için anlamlı olabilecek şekilde Tablo 7.3'teki gibi belirlenmiştir. Ayrıca, ters kinematik denklemlerdeki dönme matrisinde robotun Şekil 7.14'te gösterilen θ yunuslama açısını belirleyen θ_r'nin en yüksek sınırı, yapılan kinematik hesaplamalar sonucunda $-55°$ olarak belirlenmiştir. θ_r'nin aralığı, dik bir yürüyüşe yakın olması için yüksek değerlerde seçilmemiş ve beşinci yürüyüş parametresi olarak Tablo 7.3'teki eylem uzayına eklenmiştir.

Tablo 7.3. Eylem uzayındaki parametrelerin aralıkları

Parametre	Aralık
w	$[25, 70]$ mm
s	$[25, 80]$ mm
h	$[25, 50]$ mm
DSR	$[0.3, 0.6]$
θ_r	$[-3, -23]$ derece

Düello ÇDQA algoritması ayrık eylem uzayında çalıştığı için Tablo 7.3'te verilen parametre aralıkları ayrıklaştırılmalıdır. Çok ayrık nokta olduğunda eylem uzayı bir hayli artacağı için öğrenmenin verimi büyük ölçüde azalacaktır. Uygulama kapsamındaki deneysel çalışmalar için parametreler Tablo 7.4'teki gibi belirli aralıklarla ayrıklaştırılmıştır.

Tablo 7.4. Eylem uzayındaki parametrelerin ayrık değerleri

Parametre	**Ayrık Değerler**
w	[25 30 35 40 45 50 55 60 65 70]
s	[25 30 35 40 45 50 55 60 65 70 75 80]
h	[25 30 35 40 45 50]
DSR	[0.3 0.4 0.5 0.6]
θ_r	$[-3\ -8\ -13\ -18\ -23]$

Tablo 7.4'te görüldüğü üzere eylem uzayı w, s, h, DSR ve θ_r birleşimlerinden oluşan toplamda $10 \times 12 \times 6 \times 4 \times 5 = 14400$ boyuta sahiptir.

Ödül Fonksiyonu

Yürüyüş parametrelerinin optimizasyonu için kullanılan R ödül fonksiyonu (7.3) ile ifade edilmiştir.

$$R = 10D_x + 3S_d + 6S_b - 5S_h \tag{7.3}$$

Burada; D_x robotun ileri yönde aldığı mesafe (mm) olup yürüme görevi için oldukça önemli bir değerdir. S_d robotun sapmadan düz bir şekilde yürümedeki ölçüsünü belirleyen sayaçtır. Yürüme esnasında robot, verilen eşik sapma (±0.01 rad) değerin altında olacak şekilde yürüdükçe sayaç artmaktadır. S_b yürüme esnasında robotun kararlı bir şekilde yürümesini ifade eden sayaçtır. Her zaman adımında anlık olarak hesaplanan x ve y eksenlerindeki SMN, robotun destek poligonu içinde olduğunda sayaç artmaktadır. S_h ise SMN hesaplanmasındaki hatayı ölçen sayaçtır. Robotun bozucu etkilerden dolayı ayaklarının yere temas etmediği zaman KDD'lerden bilgi alınamadığı için SMN hesaplanamamaktadır. Böyle durumlarda sayaç artmaktadır. Sayacın artması istenmeyen bir durum olduğu için (7.3)'te görüldüğü üzere ödüle negatif bir katkısı bulunmaktadır.

Robot düştüğünde veya robotun hareketi esnasında ters kinematik hesaplama hatasıyla karşılaşıldığında ödül -100 olarak belirlenmiştir. Robotun düşmesi için tam olarak düşme eylemi beklenmeden robotun yönelim açılarından herhangi biri -60°'den küçük veya 60°'den büyük olduğu anda robot düştü olarak kabul edilmiştir.

Düello ÇDQA'nın Mimarisi

Şekil 7.15'te yapısı verilen Düello ÇDQA'nın katmanları, nöron sayıları ve aktivasyon fonksiyonları Tablo 7.5'te verilmiştir.

Tam bağlı katmanlarda 1024 ve 2048 gibi nöron sayılarına sahip YSA yapıları denenmiş ancak en iyi sonuç, Tablo 7.5'teki nöron sayıları ile tasarlanan YSA'dan elde edilmiştir.

Tablo 7.5. Düello ÇDQA'nın katman yapısı

Katman ismi	Katman türü	Nöron sayısı	Aktivasyon türü
Giriş	Tam bağlı	9	-
Ortak FC	Tam bağlı	512	DDB
Ortak FC	Tam bağlı	256	DDB
Değer için FC1	Tam bağlı	128	DDB
Avantaj için FC1	Tam bağlı	128	DDB
Değer için FC2	Tam bağlı	1	Doğrusal
Avantaj için FC2	Tam bağlı	14400	Doğrusal
Çıkış	Tam bağlı	14400	-

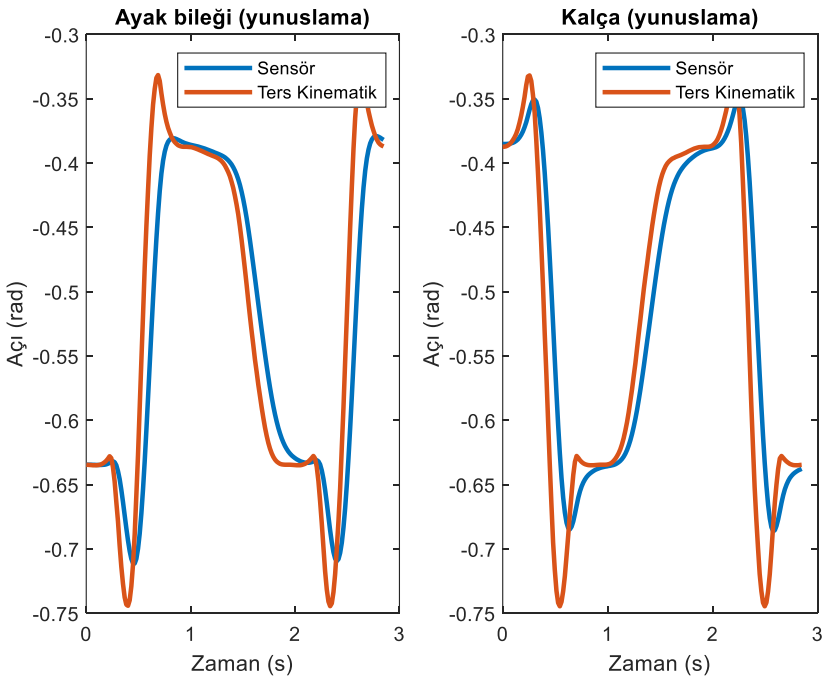
Düello ÇDQA'nın eğitimi için seçilen hiperparametreler Tablo 7.6'da listelenmiştir.

Tablo 7.6. Düello ÇDQA için seçilen hiperparametreler

Hiperparametre	Değer
Mini yığın boyutu	32
Tekrar belleği boyutu	15000
İndirim faktörü	0.95
Öğrenme oranı	0.0001
Başlangıç keşif oranı ϵ	1
Bitiş keşif oranı ϵ_{min}	0.01

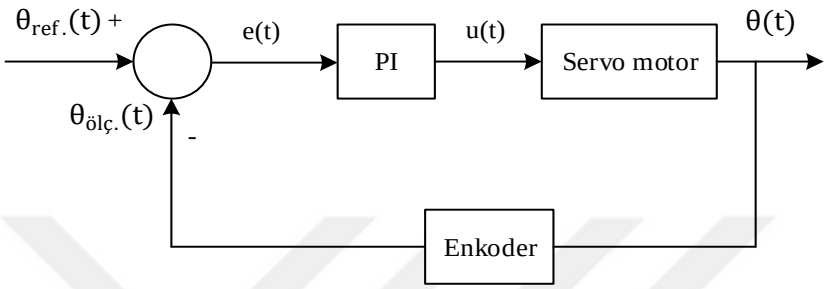
Düello ÇDQA algoritması ile yürüyüş parametrelerinin optimizasyonu için gerçekleştirilen eğitim işlemi, Adam optimize edicisi kullanılarak 30 bin bölüm içermekte olup her bölüm 7 adımdan oluşmaktadır. Her bir adım için robotun 6T süre boyunca yürüyüşü dikkate alınacak şekilde DPÖ yapısı oluşturulmuştur. Yürüyüşün kararlılığını daha iyi gözlemleyebilmek için 6T gibi uzun bir süre seçilmiştir. Bölüm içinde gerçekleştirilen her bir adımda robot bir eylem alarak hareketini gerçekleştirmiştir. Bu hareket ile robot düşmeden 12 adım atabilirse bir ödül almış ve bölümün diğer adımına kaldığı yerden geçiş sağlanmıştır. Ancak robot, yürüyüşün herhangi bir anında düştüğünde veya ters kinematik hesaplama hatası olduğunda doğrudan cezalandırılıp çevre yeniden başlatılarak diğer bölüme geçilmiştir.

Yürüyüş parametrelerinin optimizasyonu için ağ eğitimlerine başlamadan önce insansı robotun bacak eklemlerinin ters kinematik denklemler ile hesaplanan açı değerlerine hangi doğrulukta ulaştığını gözlemlemek için Webots simülatöründe bazı deneysel çalışmalar gerçekleştirilmiştir. Robotis-OP2'de kullanılan Dynamixel MX-28T servo motorun yüksek dişli oranı nedeniyle mafsallar arasındaki kuplaj etkileri bozucu etkilere sebep olabilmektedir. Webots ortamında varsayılan olarak servo motorun konum kontrolü için sadece K_p kazanç katsayısı 10 olan P kontrolörü kullanılmıştır. Ters kinematik ile hesaplanan iki açı dizisi için sağ bacaktaki ayak bileği (yuvarlanma) ve kalça (yuvarlanma) eklemlerinin enkoder ile ölçülen açı değerleri MATLAB ortamında çizdirilerek Şekil 7.16'daki gibi karşılaştırılmıştır.



Şekil 7.16. Webots ortamındaki varsayılan P kontrolör ile sonuçlar

Sonuçlar incelendiğinde enkoder ile ölçülen değerlerin referans değerlerine yakın olduğu, ancak referans girişe kıyasla faz gecikmesinin istenilen hassasiyette olmadığı görülmüştür. Robotun bacak eklemlerine gerçek zamanlı olarak daha etkili bir kapalı çevrim kontrol yapısının uygulanması gerektiği sonucuna varılmıştır. Kontrol yapısı olarak Şekil 7.17'de blok diyagramı verilen PI kontrolör seçilmiştir. PI kontrolörün transfer fonksiyonu ise (7.4)'te verilmiştir.



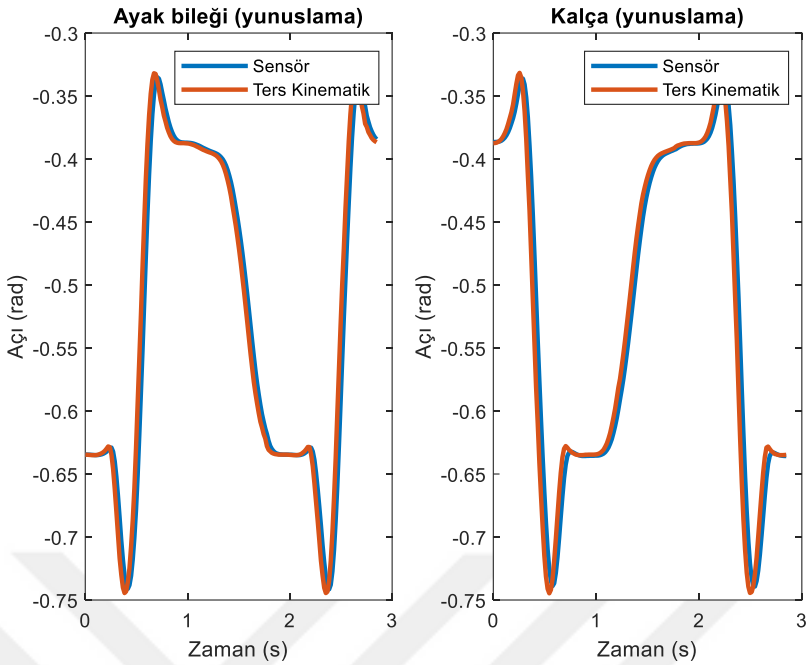
Şekil 7.17. Servo motorun kapalı çevrim konum kontrol blok diyagramı

$$u(t) = K_p e(t) + K_i \int e(t)dt \tag{7.4}$$

Burada; $e(t)$ referans açı değeri $\theta_{ref.}$ ile enkoder ile ölçülen açı değeri $\theta_{ölç.}$ arasındaki farkı ifade eden hata, $u(t)$ ise kontrolör çıkışı olan kontrol sinyalidir

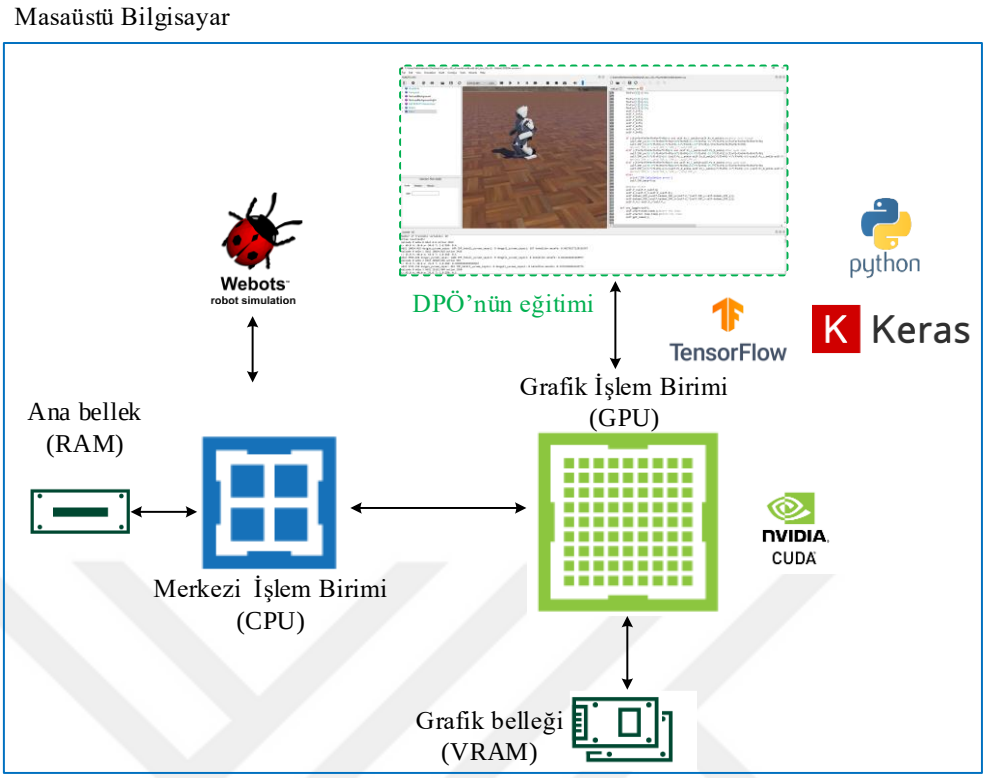
P oransal kontrol ile referans değer etrafında salınım yapılmasını sağlayarak bir kalıcı durum hatası meydana getirir. I integral kontrol sayesinde bu kalıcı durum hatası ortadan kaldırılarak sistemin referans değere yakınlaşmasını sağlanır. Deneysel çalışmalarda, K_p oransal kazanç ve K_i integral kazanç katsayıları deneme yanılma yolu ile sırasıyla 30 ve 0.9 olarak belirlenmiştir.

PI kontrolör ile Şekil 7.18'de verildiği gibi P kontrolörüne göre çok daha hassas bir sonuç elde edilmiştir. Ayak bileği (yuvarlanma), diz ve kalça (yuvarlanma) eklemleri için de aynı K_p ve K_i kazanç katsayıları kullanılarak sonuçlar incelenmiş ve yeterli olduğu görülmüştür.



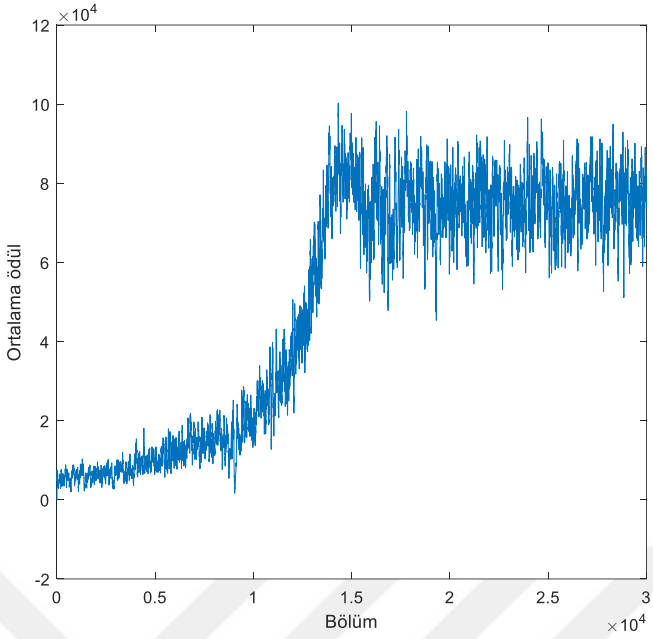
Şekil 7.18. PI kontrolörün sonuçları

Düello ÇDQA'nın eğitimi, Webots simülatör ortamında ekran kartının CUDA desteği sayesinde paralel hesaplama gücünden yararlanıp Tensorflow 2.2.0, Keras 2.3.1 ve Python 3.8 versiyonlarını kullanarak yazılan kontrolör ile gerçekleştirilmiştir. Şekil 7.19'da gösterildiği gibi eğitim kapsamındaki çalışmalar, NVIDIA GTX Titan X Pascal ekran kartı, Intel i5 4. nesil 3.4 GHz işlemci, 8 GB RAM'in bulunduğu masaüstü bir bilgisayarda gerçekleştirilmiştir.



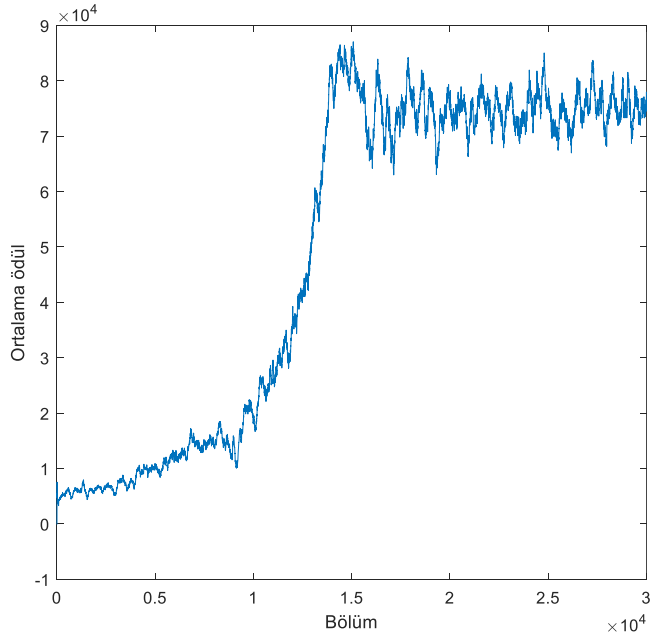
Şekil 7.19. DPÖ ile eğitim işleminin şematik diyagramı

Eğitim işlemi esnasında her bölümde alınan ödüller kaydedilmiştir. Her 100 bölümde bir ağırlıklar saklanmıştır. Eğitim tamamlandıktan sonra son 40 bölümün ortalaması alınarak elde edilen ortalama ödül grafiği, MATLAB ortamında Şekil 7.20'deki gibi çizdirilmiştir.



Şekil 7.20. Son 40 bölümün ortalaması ile ortalama ödül grafiği

Şekil 7.20 incelendiğinde oldukça dalgalı bir grafiğin çıktığı görülmüştür. Bu durum, eğitimin çok fazla bölüm içermesinden kaynaklanmaktadır. Son 40 bölüm yerine son 200 bölümün ortalaması alınarak elde edilen grafik, Şekil 7.21'deki gibi elde edilmiştir.



Şekil 7.21. Son 200 bölümün ortalaması ile ortalama ödül grafiği

Şekil 7.21'deki eğitim sonuçları incelendiğinde yaklaşık olarak 16000. bölüme kadar ortalama ödül, üstel bir şekilde artarak gitmiştir. Daha sonra ise ortalama ödül belirli bir aralıkta seyretmiştir. Eğitim sırasında birçok başarılı yürüme işlemini gerçekleştirebilen parametrelere yakınsandığı gözlemlenmiştir. Ancak, yaklaşık son 9000 bölümde tekrarlı bir şekilde Tablo 7.7'deki eylemden oluşan eylem dizileri elde edilmiştir.

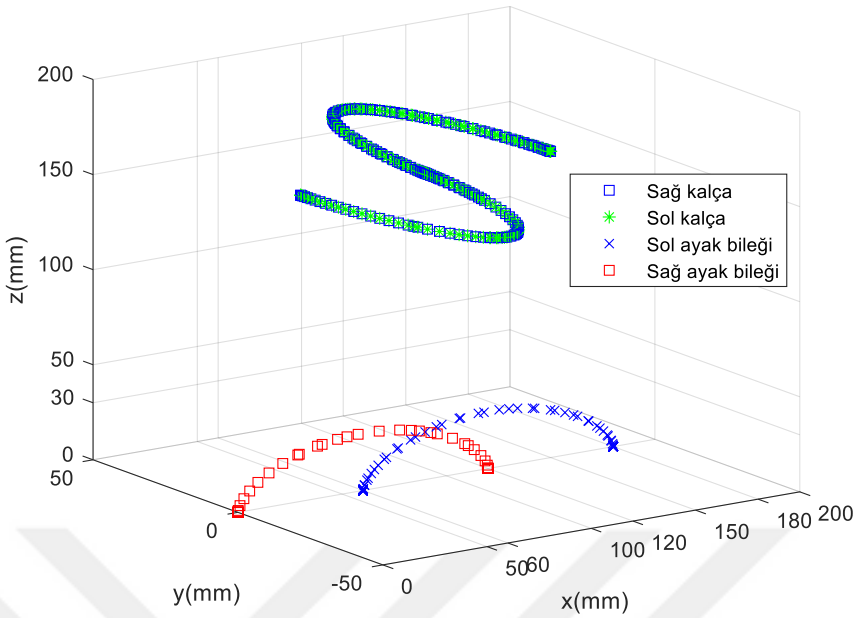
Tablo 7.7. Eğitim sonucunda elde edilen optimum yürüyüş parametreleri

Parametre	**Değer**	**Birim**
w	50	mm
s	60	mm
h	30	mm
DSR	0.6	-
θ_r	-13	derece

Tablo 7.7'deki değerler ile altı yürüme çevrimi (6T) için D_x, S_d, S_b ve S_h sırasıyla 709.2 mm, 711, 581 ve 0 olarak elde edilmiş olup (7.3) ile hesaplanan ödül değeri 12711'dir.

7.3.2. Düello ÇDQA ile Test

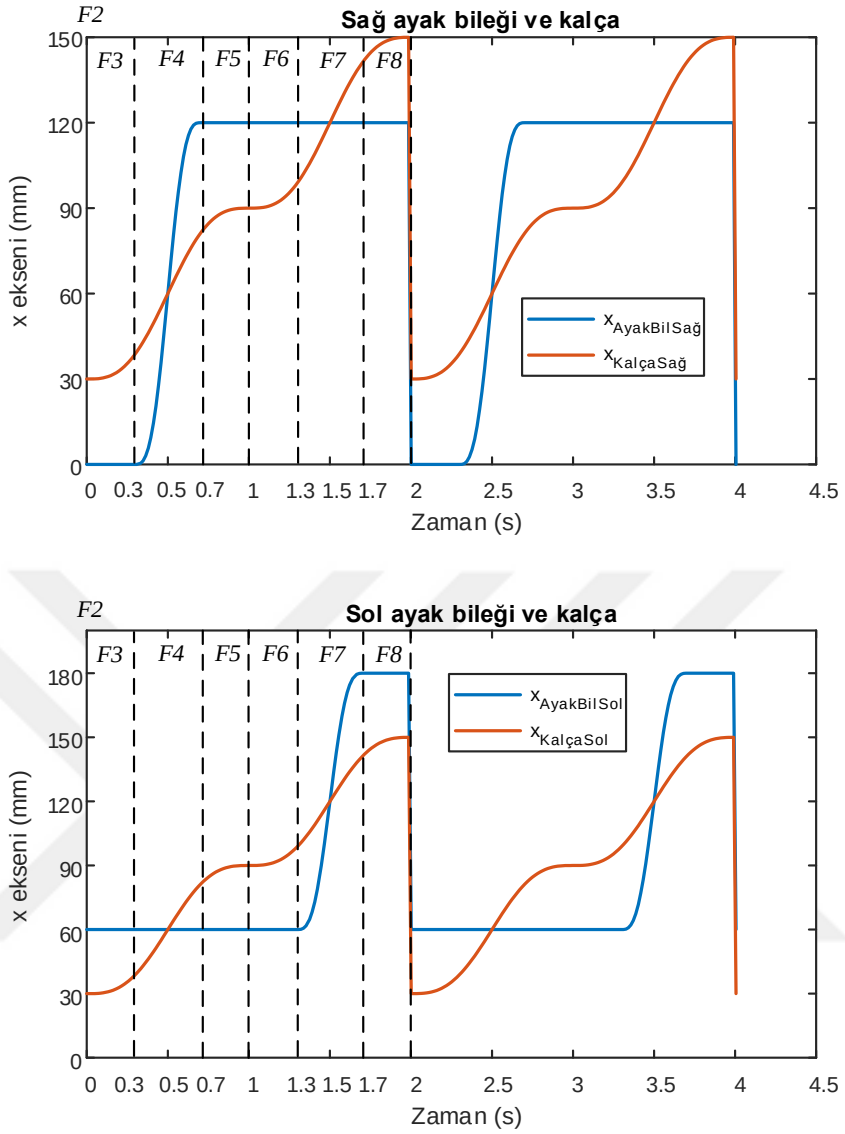
Kaydedilen ağ ağırlıkları yüklenip Webots simülatöründe ağ modelinin test işlemleri gerçekleştirilmiştir. Tablo 7.7'deki optimize edilen yürüyüş parametreleri ile elde edilen yürüme yörüngeleri, iki yürüme çevrimi için MATLAB ortamında grafiklerle görselleştirilmiştir. Yürüme modelinin üç boyutlu gösterimi Şekil 7.22'de verilmiştir.



Şekil 7.22. Yürüme modelinin üç boyutlu gösterimi

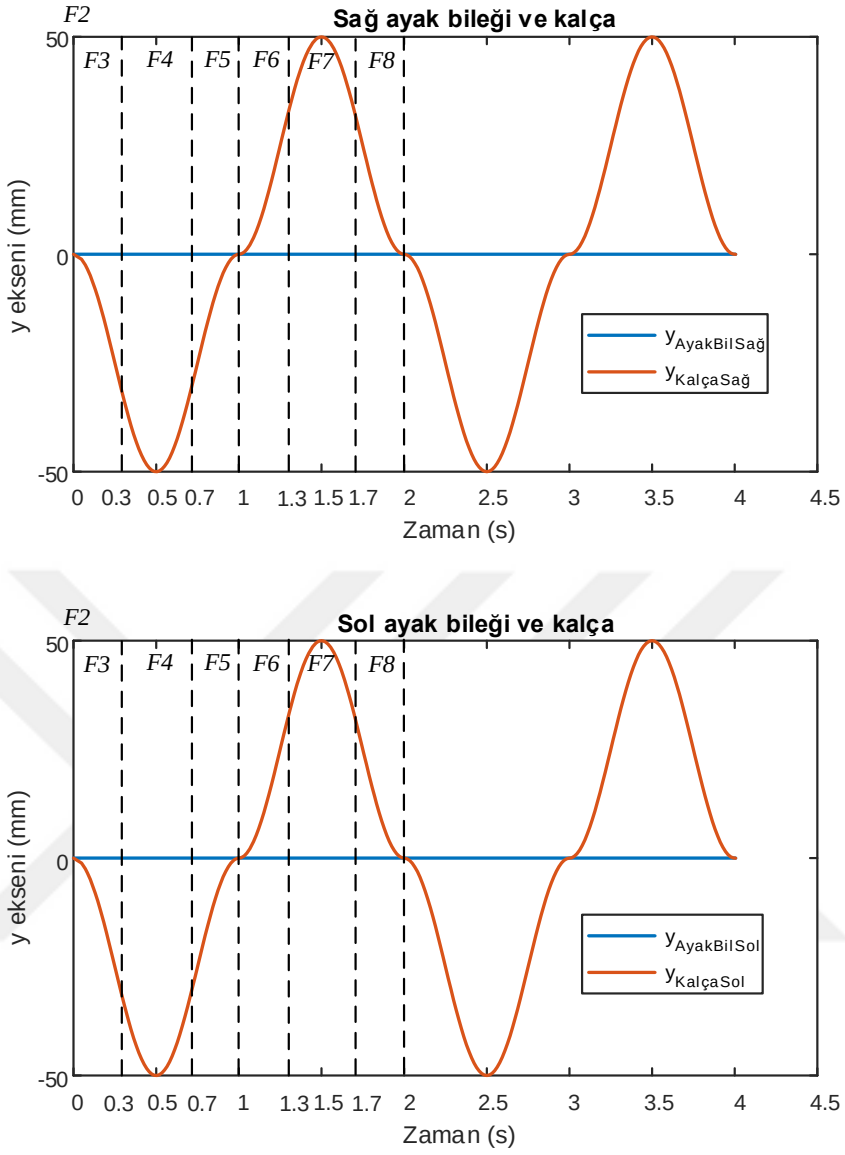
Şekil 7.22 incelendiğinde robotun hızının x ekseninde 60 mm/sn olduğu ve robotun kalçasının en fazla 50 mm sallanma hareketi yaptığı görülmektedir. Ayrıca, robotun yürüyüşü esnasında ayağının yerden en fazla 30 mm yükseldiği görülmektedir.

Sağ ve sol bacak için ayak bileği ve kalça eklem yörüngeleri fazlarıyla birlikte Şekil 7.23, Şekil 7.24 ve Şekil 7.25'te gösterilmiştir.



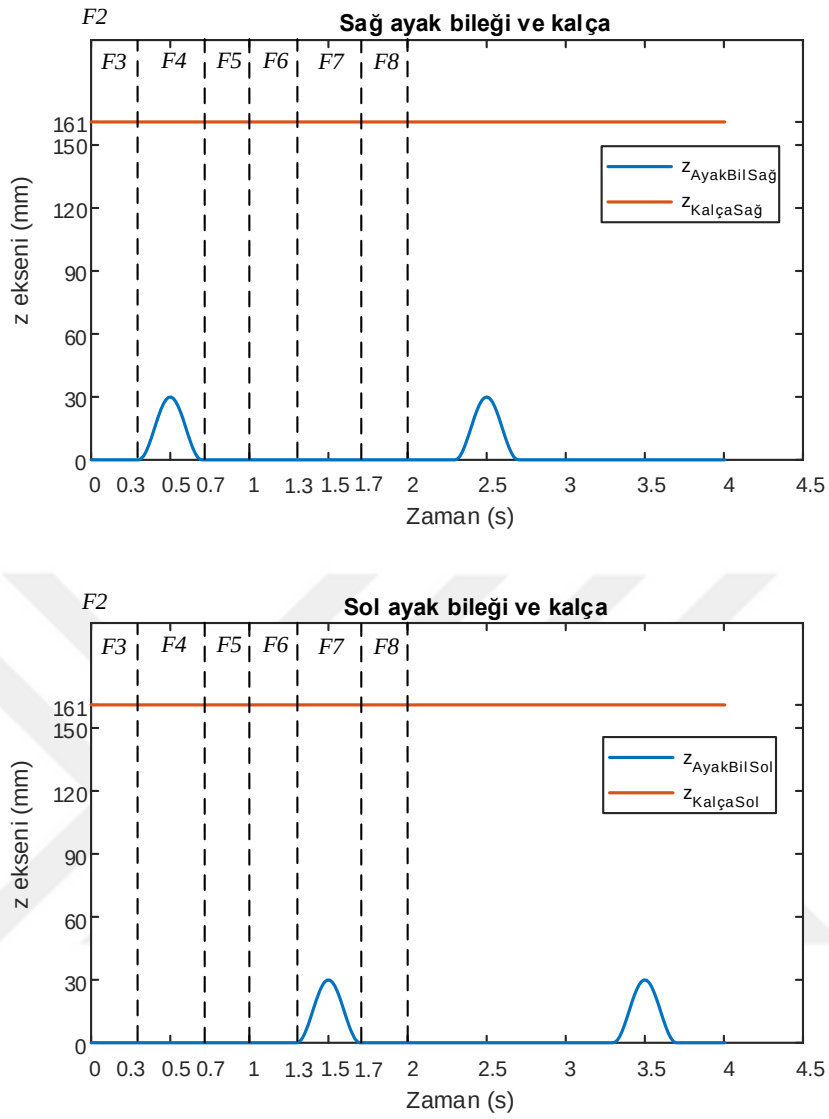
Şekil 7.23. Ayak bileği ve kalçanın x ekseni yörüngesi

Şekil 7.23'e göre, x eksenindeki kalça yörüngesi sağ ve sol bacak için aynı olmasına rağmen, ayak bileği için birbirine göre aynalı salınımlara sahiptir.



Şekil 7.24. Ayak bileği ve kalçanın y ekseni yörüngesi

Şekil 7.24'e göre ayak bileği ve kalça yörüngeleri her iki bacak için de aynıdır.



Şekil 7.25. Ayak bileği ve kalçanın z ekseni yörüngesi

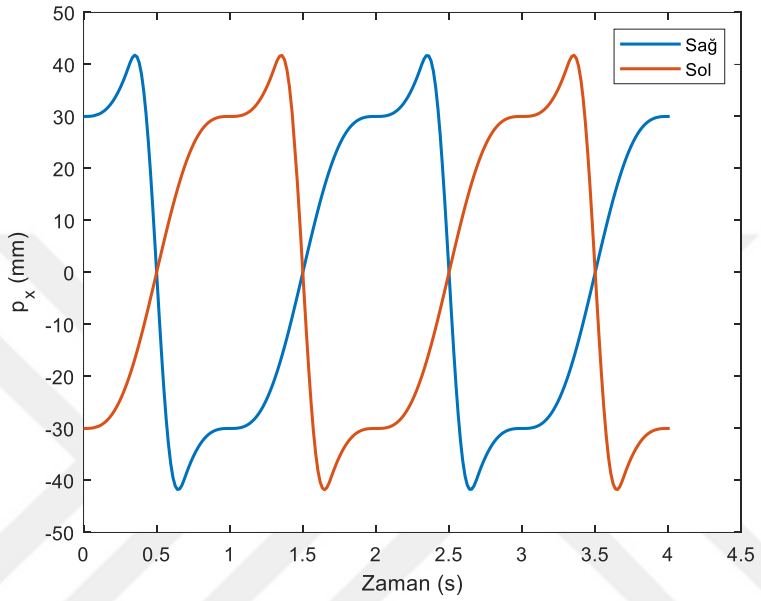
Şekil 7.25'e göre kalça yörüngesi sağ ve sol bacak için aynı olmasına rağmen ayak bileği yörüngesi faz farkına sahiptir.

Şekil 7.23, Şekil 7.24 ve Şekil 7.25'teki yörüngeler gözlemlendiğinde, yürümeye Faz 2 ile başlanır. 0 s-0.3 s arasındaki Faz 3'te, kalça eklemleri sola kaydırılarak robotun ağırlık merkezi sol ayağa hareket ettirilir. 0.3 s-0.7 s arasındaki Faz 4'te, sağ ayak adım atmaya başlar. 0.7 s-1 s arasındaki Faz 5'te, sağ ayak adım atmayı bitirip destek alanına geri döner. 1 s-2 s arasındaki Faz 6, Faz 7 ve Faz 8 ile 0 s-1 s arasındaki işlemlerin tersi yapılarak sol ayağın da adım atması sağlanır ve bir yürüme çevrimi tamamlanır.

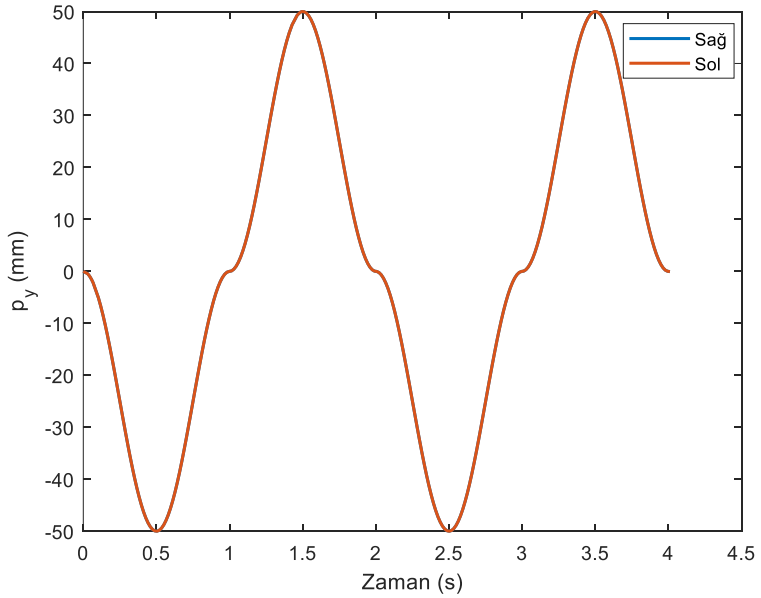
Şekil 7.23, Şekil 7.24 ve Şekil 7.25'te verilen yürüme yörüngesi kullanılarak kalça ekleminin ayak bileği eklemine göre konum koordinatları olan p_x, p_y ve p_z, iki bacak için de hesaplanmıştır.

Hesaplama işlemi, her zaman adımında anlık olarak kalça ekleminin konumundan ayak bileği ekleminin konumunun çıkarılmasını içermektedir.

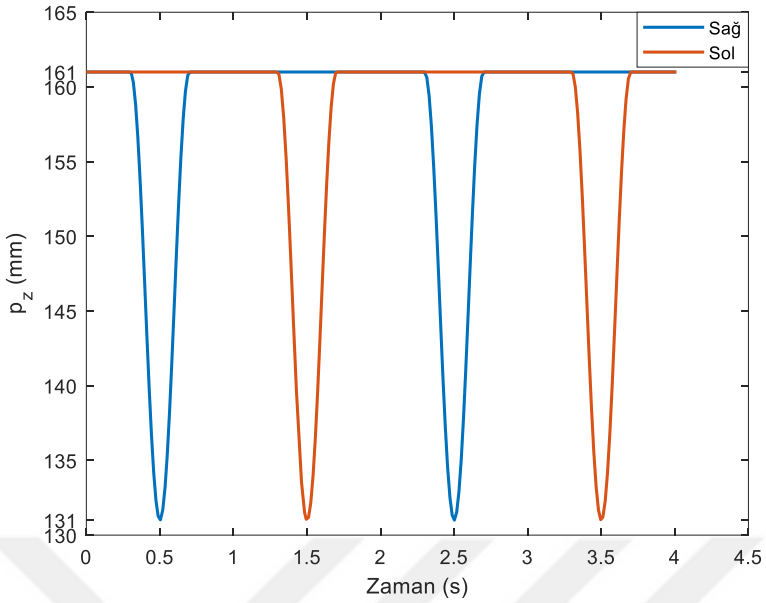
Sağ ve sol bacak için hesaplanan x, y ve z eksenlerindeki konum koordinatları sırasıyla Şekil 7.26, Şekil 7.27 ve Şekil 7.28'de gösterilmiştir.



Şekil 7.26. Sağ ve sol bacak için p_x konum koordinatları

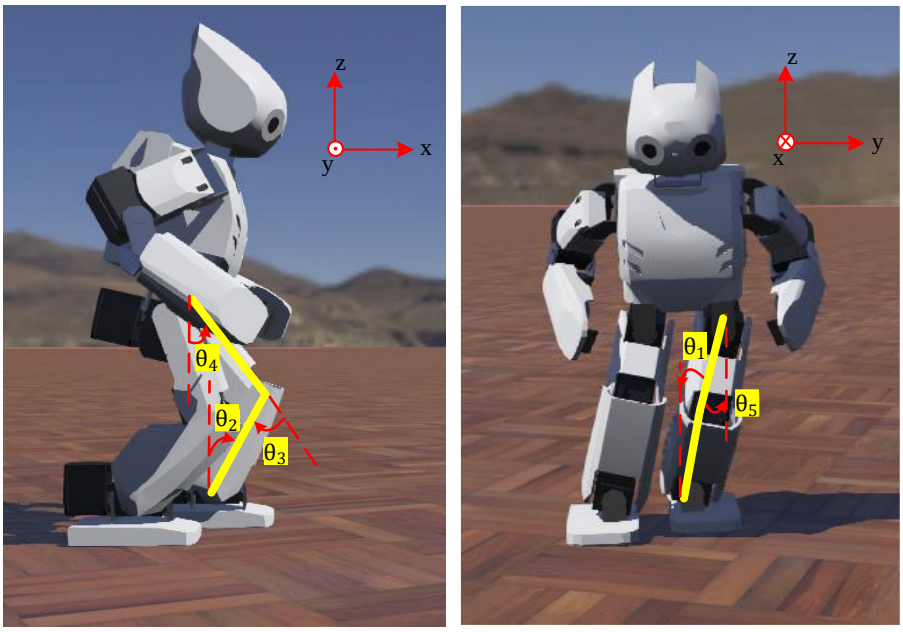


Şekil 7.27. Sağ ve sol bacak için p_y konum koordinatları



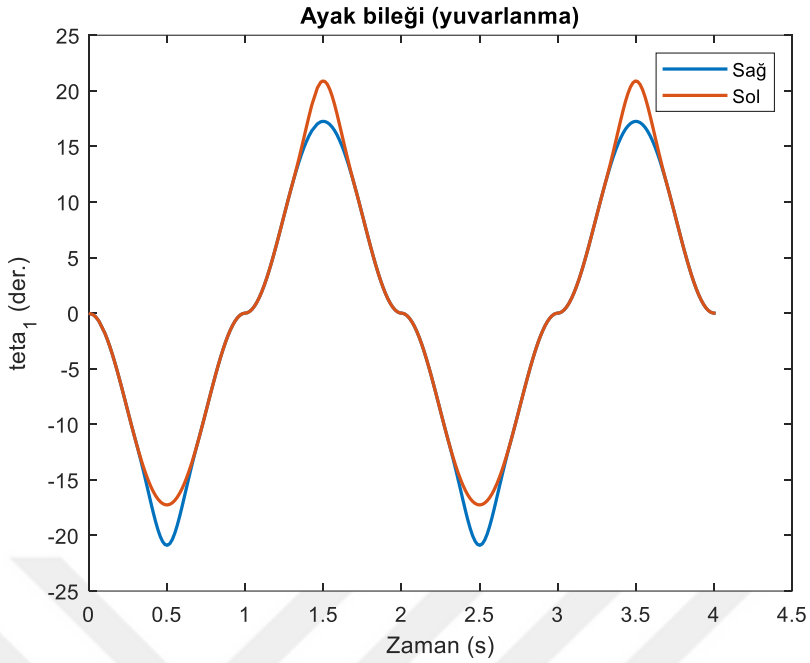
Şekil 7.28. Sağ ve sol bacak için p_z konum koordinatları

Yürüme yörüngesiyle elde edilen p_x, p_y ve p_z konum koordinatlarına göre robotun yürüyüşü esnasında Şekil 7.29'da sol bacak üzerinde gösterilen bacak eklemlerinin alması gereken açılar, ters kinematik analiz ile hesaplanmıştır. Hesaplanan eklem açıları sırasıyla Şekil 7.30, Şekil 7.31, Şekil 7.32, Şekil 7.33 ve Şekil 7.34'te gösterilmiştir.

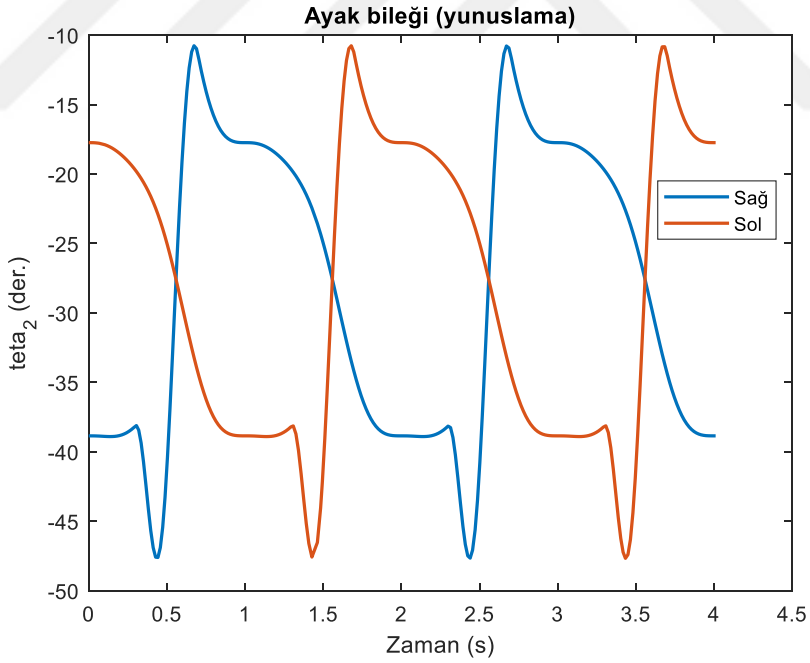


(a) (b)

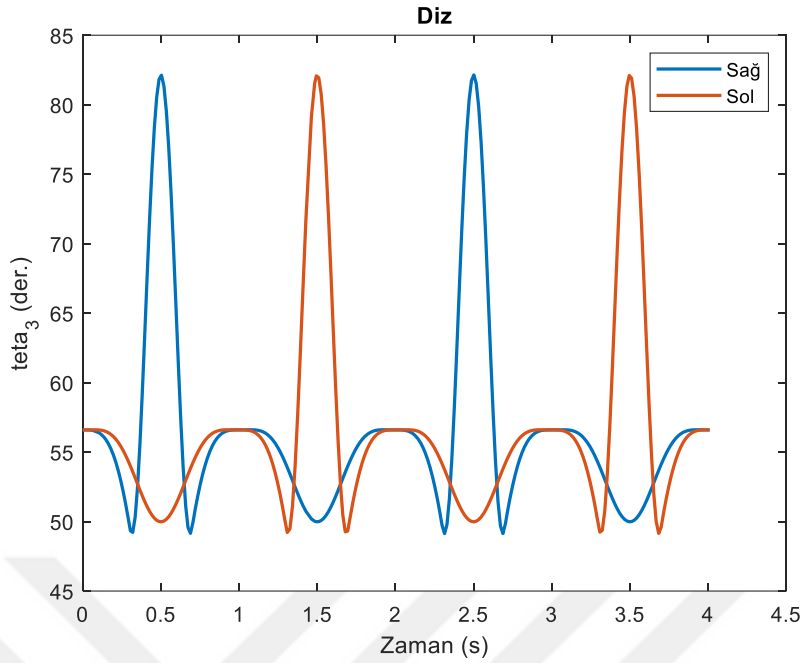
Şekil 7.29. Bacak açılarının gösterimi: (a) yunuslama açıları ve (b) yuvarlanma açıları



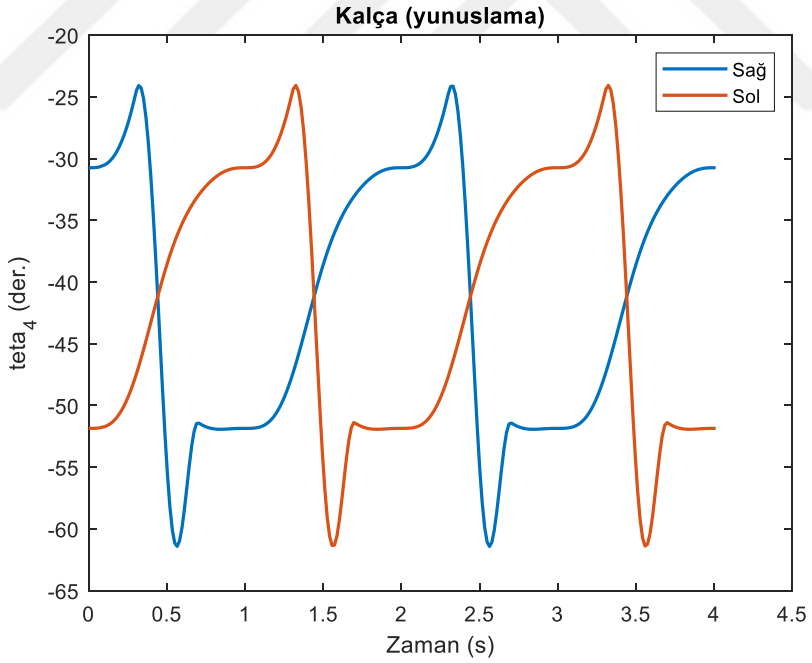
Şekil 7.30. Sağ ve sol bacak için θ_1 açıları



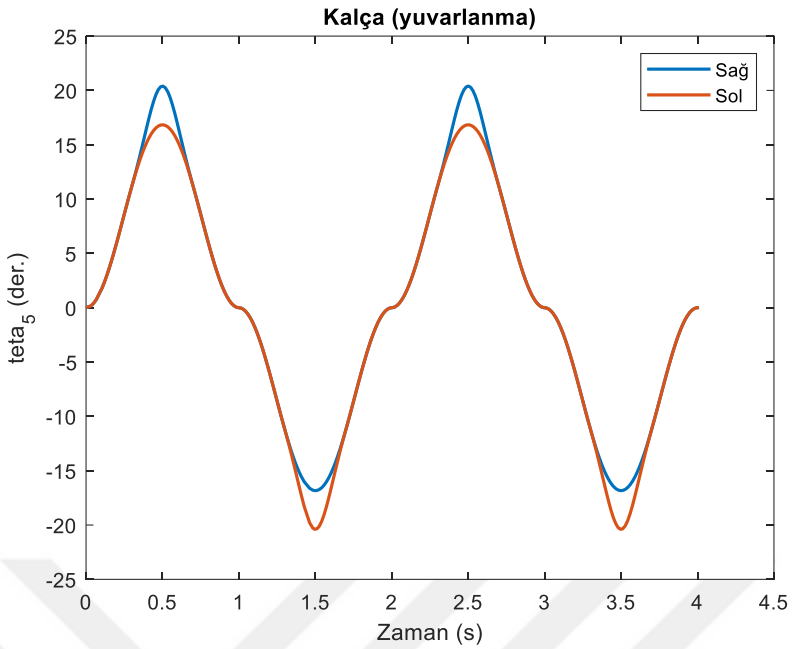
Şekil 7.31. Sağ ve sol bacak için θ_2 açıları



Şekil 7.32. Sağ ve sol bacak için θ_3 açıları

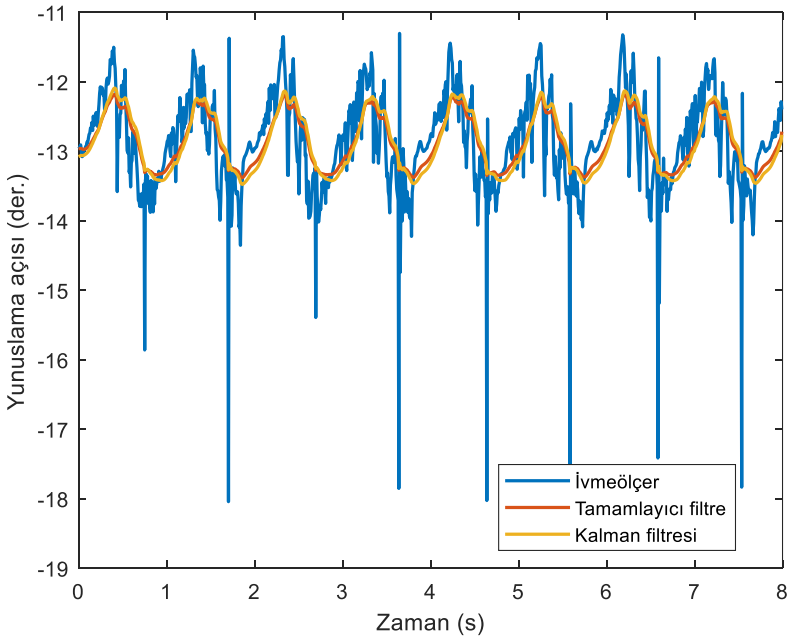


Şekil 7.33. Sağ ve sol bacak için θ_4 açıları

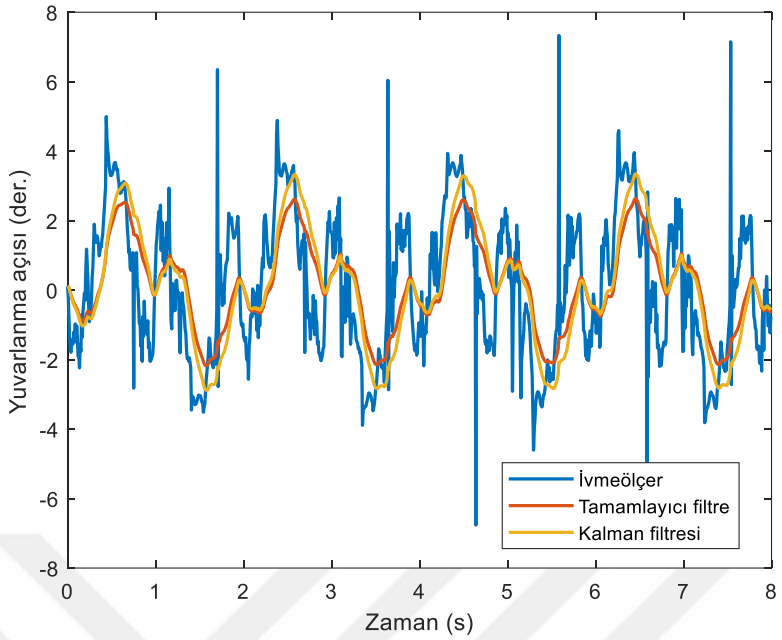


Şekil 7.34. Sağ ve sol bacak için θ_5 açıları

Oluşturulan yürüme şablonu ile Webots ortamında Robotis-OP2'nin dört yürüme çevrimi için yürüme esnasındaki vücut yunuslama ve yuvarlanma açılarının ivmeölçer, tamamlayıcı filtre ve Kalman filtresi ile hesaplanmasına sırasıyla Şekil 7.35 ve Şekil 7.36'da yer verilmiştir.



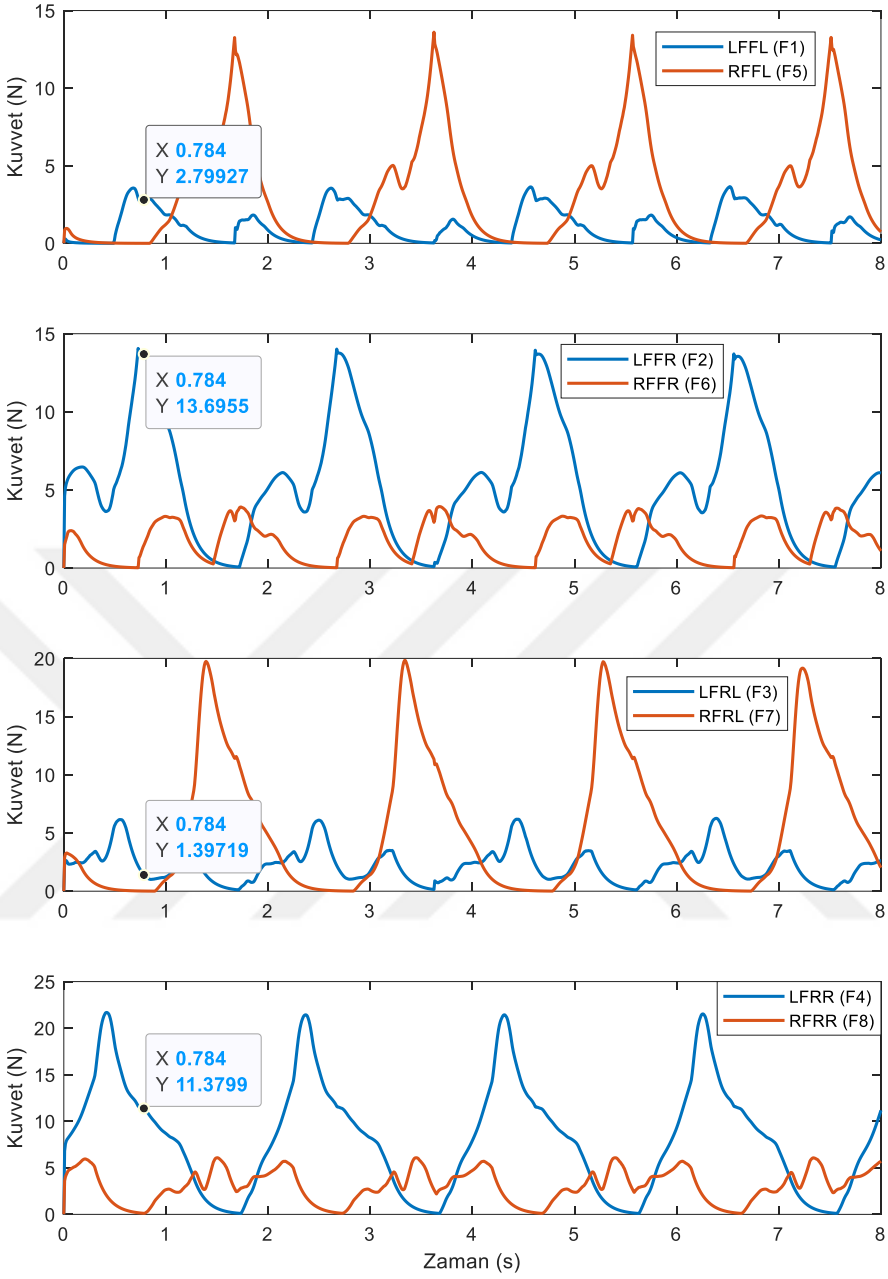
Şekil 7.35. İvmeölçer, tamamlayıcı filtre ve Kalman filtresi ile hesaplanan yunuslama açılarının karşılaştırılması



Şekil 7.36. İvmeölçer, tamamlayıcı filtre ve Kalman filtresi ile hesaplanan yuvarlanma açılarının karşılaştırılması

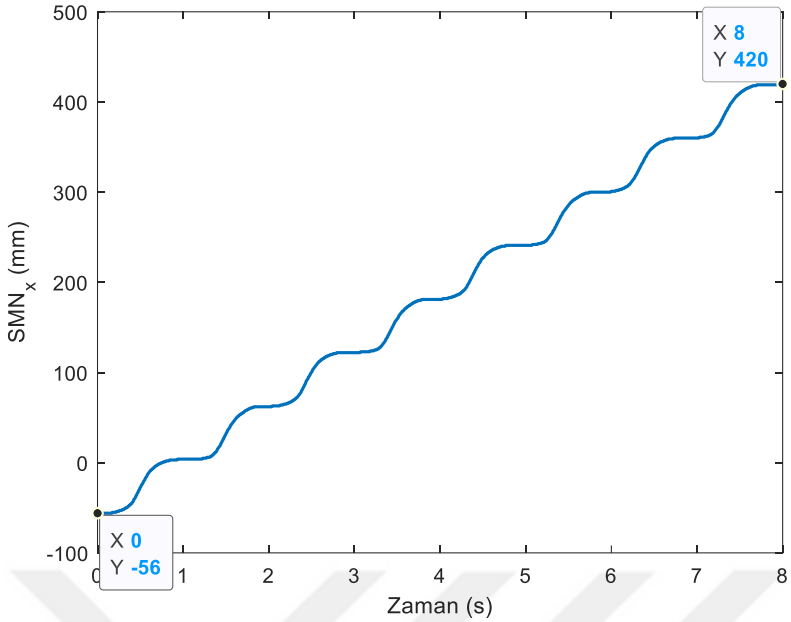
Düello ÇDQA eğitim işlemlerinde, robotun yunuslama ve yuvarlanma yönelim açılarının hesaplanması için Kalman filtresi tercih edilmişti. Şekil 7.35 ve Şekil 7.36 gözlemlendiğinde, yalnızca ivmeölçerden elde edilen açıya en yakın sonucun Kalman filtresi kullanılarak elde edildiği görülmüştür.

Şekil 7.37'de ise yürüyüş sırasında KDD'lerden okunan değerlerin bir boyutlu Kalman filtresinden geçtikten sonra sol ve sağ ayağın temas kuvvetleri, MATLAB ortamında görselleştirilmiştir. Şekil 7.37'deki kuvvet değerleriyle hesaplanan SMN_x ve SMN_y koordinatları ise sırasıyla Şekil 7.38 ve Şekil 7.39'da gösterilmiştir.

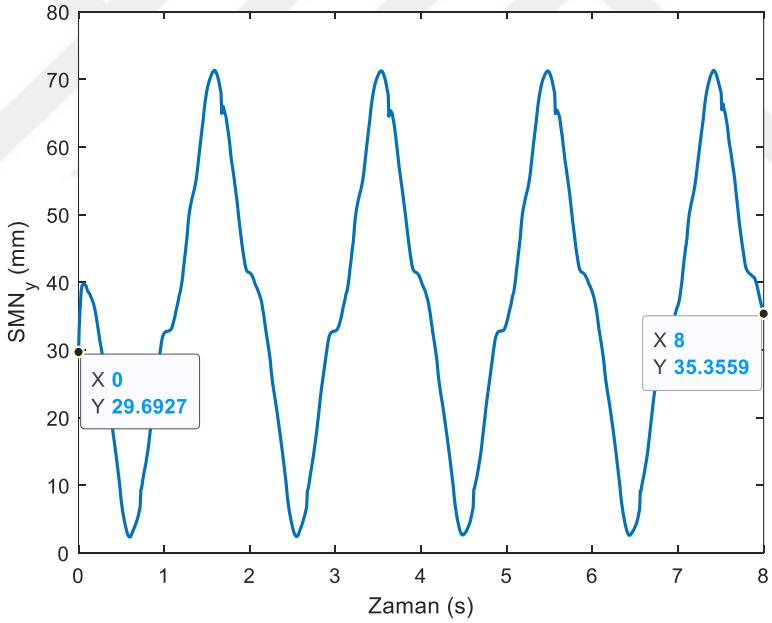


Şekil 7.37. Sekiz KDD'den okunan sol ve sağ ayağın temas kuvvetleri

Şekil 7.37'de görüldüğü gibi 0.784 s'de tek ayak destek fazında olan robotun sol ayağında yer alan KDD'lerdeki dört kuvvet değerine bakıldığında toplam kuvvetin 29.272 N olarak hesaplanmıştır. Robotun ağırlığı 3 kg olduğu için toplam kuvvetin 29.43 N olarak ölçülmesi beklendiğinde elde edilen sonucun oldukça iyi olduğu anlaşılmıştır.



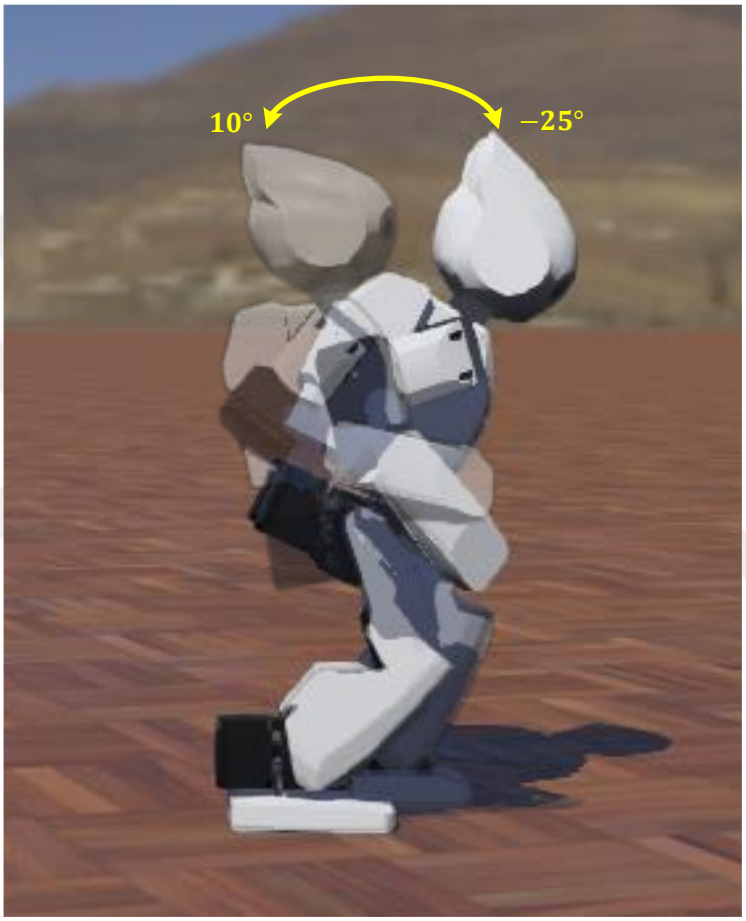
Şekil 7.38. Yürüyüş sırasında hesaplanan SMN_x değerleri



Şekil 7.39. Yürüyüş sırasında hesaplanan SMN_y değerleri

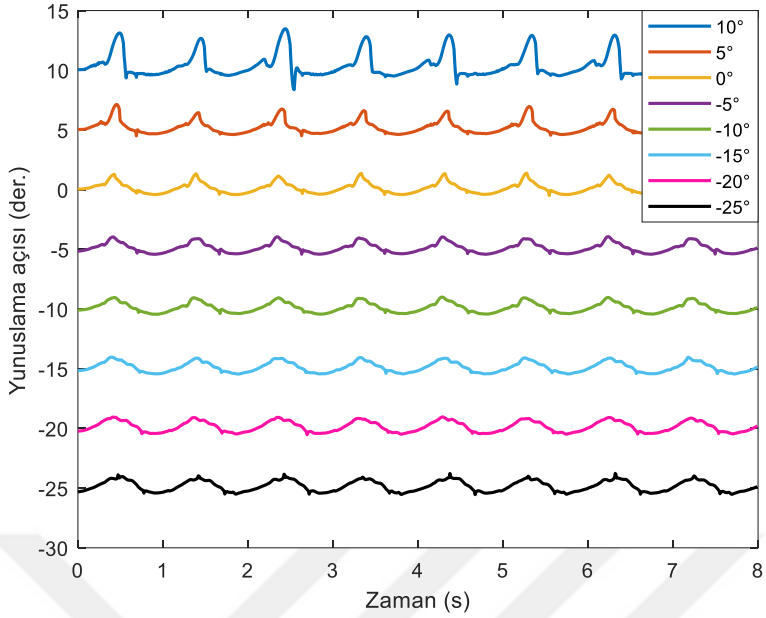
Şekil 7.38 incelendiğinde x eksenindeki SMN koordinatlarının düzgün bir şekilde salınım yaparak arttığı görülmüştür. 60 mm adım uzunluğu ile dört yürüme çevrimi için robotun 480 mm mesafe kat etmesi beklendiğinde robotun SMN değerinin 476 mm ilerlemesi oldukça başarılı olduğunu göstermiştir. Benzer şekilde Şekil 7.39 incelendiğinde ise y eksenindeki SMN koordinatlarının belirli bir periyotta düzenli bir salınım yaptığı ve yürüyüşün sonunda sadece yaklaşık 5.7 mm'lik bir hizadan çıkma durumunun söz konusu olduğu görülmüştür.

Düello ÇDQA'nın eğitimi sonucunda optimum yürüyüş için ters kinematik analizde robotun yunuslama yönelim açısını belirleyen θ_r, $-13°$ olarak bulunmuştu. Robotun aynı yürüyüş parametreleri ile sadece yunuslama açısını değiştirerek başarılı bir yürümenin gerçekleşip gerçekleşmeyeceği de deneysel olarak gözlemlenmiştir. Şekil 7.40'ta görselleştirildiği gibi $10°$ ile $-25°$'deki duruşlarda dört yürüme çevrimi boyunca deneysel çalışmalar gerçekleştirilmiştir.

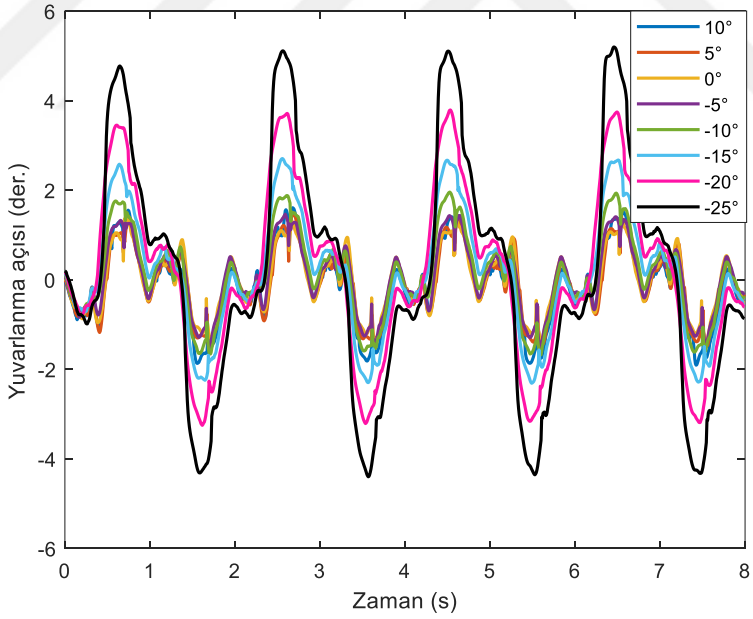


Şekil 7.40. Farklı açılarda vücut duruşu ile yürüme

Beşer derecelik açı değişimlerindeki duruşlarda gerçekleştirilen deneylerde robot kararlı bir şekilde düşmeden yürüyebilmiştir. Yürüme esnasında Kalman filtresiyle elde edilen robotun yönelim açıları Şekil 7.41 ve Şekil 7.42'deki gibi elde edilmiştir.



Şekil 7.41. Farklı θ_r'ler için robotun yürüme esnasındaki yunuslama açıları



Şekil 7.42. Farklı θ_r'ler için robotun yürüme esnasındaki yuvarlanma açıları

Şekil 7.41'deki yunuslama açı değişimleri gözlemlendiğinde, robotun arkaya doğru eğildiği 5° ve 10°'deki duruşlar dışında birbirine benzeyen salınımlar görülmüştür. 5° ve 10°'de ise robot gövdesinin geriye hareketiyle ağırlık merkezinin arkaya doğru hareketinden robotun dengesinin bir

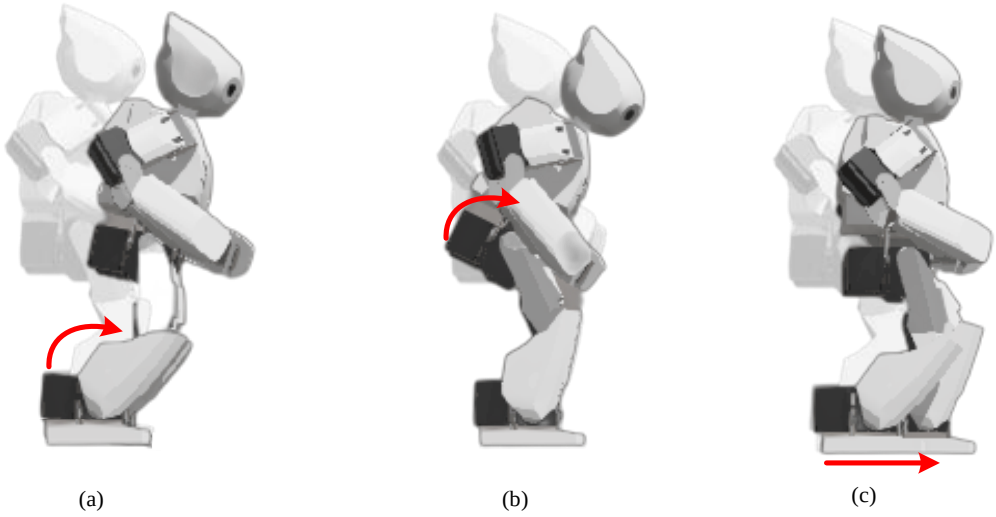
miktar bozulduğu görülmüştür. İnsan yürüyüşü de göz önüne alındığında bu durum oldukça doğaldır.

Şekil 7.42 incelendiğinde $-25°$, $-15°$ ve $-10°$'deki yürüyüşlerde yanal salınımın daha yüksek seyrettiği görülmüştür. Ancak, bu durumun kararlı bir şekilde düz yürümeye etkisinin az olduğu gözlemlenmiştir.

7.3.3. Vücut Duruşunun Dengelenmesi

Düz yürüme için yapılan deneysel çalışmalarda, düz bir zeminde herhangi bir bozucu etki olmadığı için işlemler ilk olarak açık çevrim olarak gerçekleştirilmiştir. Robotun dengesini bozacak herhangi bir dış kuvvet olmadığı için dengeleme stratejilerinden yararlanılmamıştır. Ancak, robotun sagital düzlemdeki yürüyüşünde önem arz eden vücut yunuslama açısının istenilen değerde daha az salınım ile tutulabilmesi için kapalı çevrim bir kontrole ihtiyaç duyulabilmektedir. Ayrıca, insanların günlük yaşamlarında olduğu gibi iki ayaklı bir robotun yalnızca düz yüzeylerde değil, aynı zamanda çeşitli engebeli arazilerde de uyumlu bir şekilde yürümesi beklenir. İnsansı robotlarda engebeli arazide iki ayak üzerinde yürümenin denge kontrolü zorlu bir problemdir.

Yürüme esnasında bir bozulma meydana geldikten sonra dengeyi sağlamak, duruma bağlı olarak farklı stratejiler içeren karmaşık bir süreç olabilir. İnsanlarda bu sürecin deneysel çalışması, ayak bileği, kalça ve adım atma stratejileri olmak üzere üç temel stratejinin belirlenmesine yol açmıştır. İnsansı robotlarda da Şekil 7.43'te gösterilen bu üç dengeleme stratejisi benimsenmiştir.



Şekil 7.43. Dengeleme stratejileri: (a) ayak bileği stratejisi, (b) kalça stratejisi ve (c) adım stratejisi

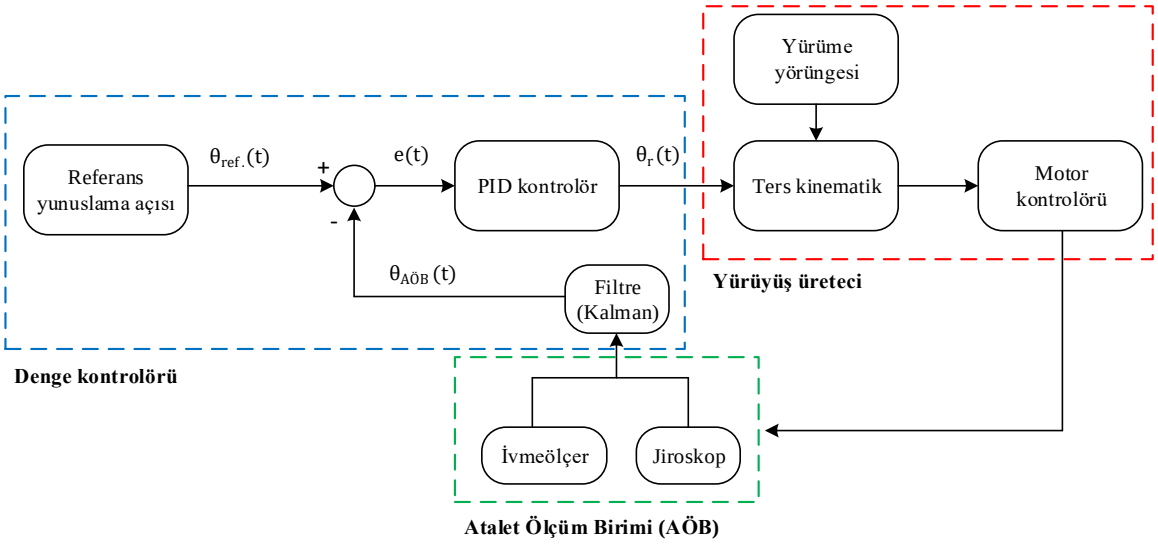
Ayak bileği stratejisi, robotun kütle merkezini korumak için aktüatörü kontrol eder. Ayak bileği stratejisinde, vücudun salınımı ayak bileği eklemi etrafındadır ve genellikle küçük dış bozucuların üstesinden gelmek için kullanılır. Mekanik olarak, ayak bileği stratejisi, vücudun ayak bileği eklemi torku tarafından kontrol edilen tek parçalı ters sarkaç olarak hareket etmesine izin vererek, vücudun üst eklemler etrafında minimum hareketle ayak bileği eklemi etrafında dönmesinden oluşur.

Kalça stratejisi, robotun kütle merkezini desteğin tabanına doğru çekmek için gövdenin açısal ivmesini kullanarak itme için karşı kuvveti oluşturur. Kalça stratejisinde vücudun üst kısmı kalça eklemi çevresinde öne ve arkaya doğru eğilir. Genellikle biraz daha büyük bozucu etkiler için kullanılır. Ayak bileklerinin kontrol edemediği bozucu çok büyük olduğunda devreye girer. Yine de düzeltmek için bir adım atılmasını gerektirecek kadar büyük değildir. Kalça stratejisi, üst gövdenin ileri ve aşağı dönmesini içerir, alt gövdede geriye doğru bir dönme uygularken, aynı zamanda ayak bileği etrafındaki atalet momentini azaltır ve belirli bir ayak bileği torkunun vücudun daha yüksek açısal hızlanmasını etkilemesine izin verir.

Adım stratejisi ise robotun destek tabanını itme yönünde adımlayarak dengeleyebilir. Sabit destek stratejileri olan ayak bileği ve kalça stratejilerinin dengeyi kurtarmak için yeterli olmadığı büyük bozulmalar için uygundur. Robotun kararlı duruşunu sürdüremediği durumlarda, dengenin iyileşmesi için bir adım atması gerekir.

Tez çalışmasında, dengeleme için kalça stratejisinin kullanımı tercih edilmiştir. Kalça stratejisinin seçilmesinin temel nedeni, kalçanın bacaklar ile vücut arasındaki bağlantı parçası olduğu için doğrudan vücut yönelimini etkilemesidir. Ayrıca, bozucu etkilere karşı ayak bileği stratejisinden daha dayanıklı olması ve yürüyüş için kinematik denklemler oluşturulurken kalçanın uç eklem olarak seçilmesinden dolayı vücut eğim açısının tek bir parametre ile kontrol edilebilmesidir. Böylelikle kalça, ters kinematik denklemlerde bulunan yunuslama dönme matrisindeki parametreyle vücudun ileri ve geri eğilmeleri için doğrudan kontrol edilerek robotun ağırlık merkezinin yer değişimiyle denge için önemli rol oynar.

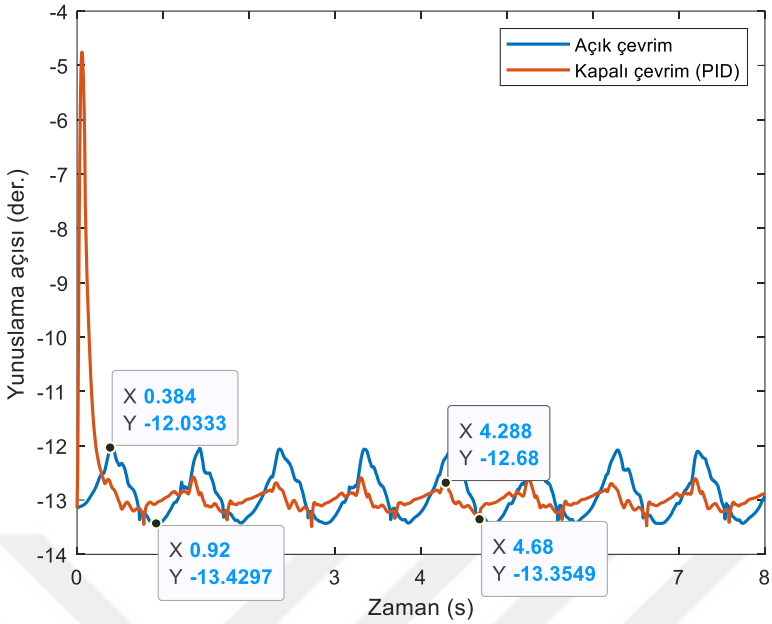
Yürüyüş sırasında vücut dengeleme kontrolü için Robotis-OP2'nin yunuslama vücut yöneliminin gerçek zamanlı kapalı çevrim kontrolü ile ilgili deneysel çalışmalar yapılarak açık çevrim sonuçlarıyla karşılaştırılmıştır. Jiroskop ve ivmeölçerden oluşan AÖB'nin geri bildirimine dayalı olarak önerilen yapının blok diyagramı Şekil 7.44'te verilmiştir.



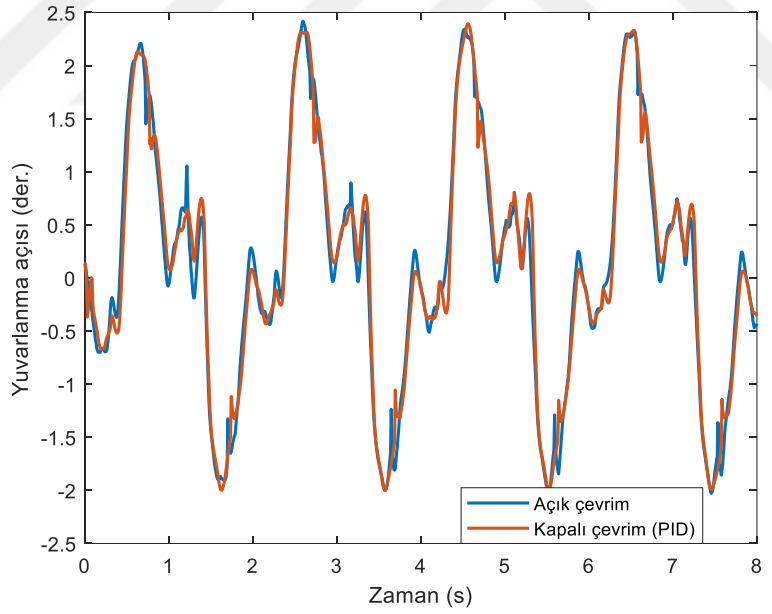
Şekil 7.44. Yürüyüş sırasında vücut dengeleme sisteminin blok diyagramı

Şekil 7.44'te görüldüğü gibi robot gövdesinin yunuslama açısından oluşan referans girişi ($\theta_{ref.}$), kontrol amacının gövdeyi dengede tutmak olduğunu belirtmek için kullanılır. Gövdenin merkezinde bulunan AÖB'nin geri bildirim sinyaline ($\theta_{AÖB}$) bağlı olarak, kontrol girişi olarak yönelim hatası (e) hesaplanır. PID kontrolörün çıkışı ile elde edilen düzeltme parametresi (θ_r) ve istenen yürüme yörüngesi ve ters kinematik analiz kullanılarak eklem açıları motor kontrolörüne gönderilir. Her motor, harici bozulmadan kaynaklanan herhangi bir yönelim hatasını telafi etmek için bu yörüngeyi motor kontrolörü aracılığıyla izler.

Daha önce açık çevrim olarak gerçekleştirilen düz zeminde yürüme işlemi ile ilgili yönelim açıları ve SMN'leri içeren deneysel sonuçlar Şekil 7.35, Şekil 7.36, Şekil 7.38 ve Şekil 7.39'da verilmişti. Kapalı çevrim ile yine dört yürüme çevrimi için aynı yürüyüş parametreleri (Tablo 7.7) kullanılarak düz yürüme için sonuçlar alınmıştır. Kapalı çevrim için PID kontrolörün deneme yanılma ile elde edilen kazanç katsayıları K_p, K_i ve K_d sırasıyla 0.8, 7.1 ve 0.01 olarak belirlenmiştir. Yürüyüş sırasında elde edilen açık çevrim sonuçları ile kapalı çevrim sonuçları Şekil 7.45, Şekil 7.46, Şekil 7.47 ve Şekil 7.48'deki gibi karşılaştırılmıştır.

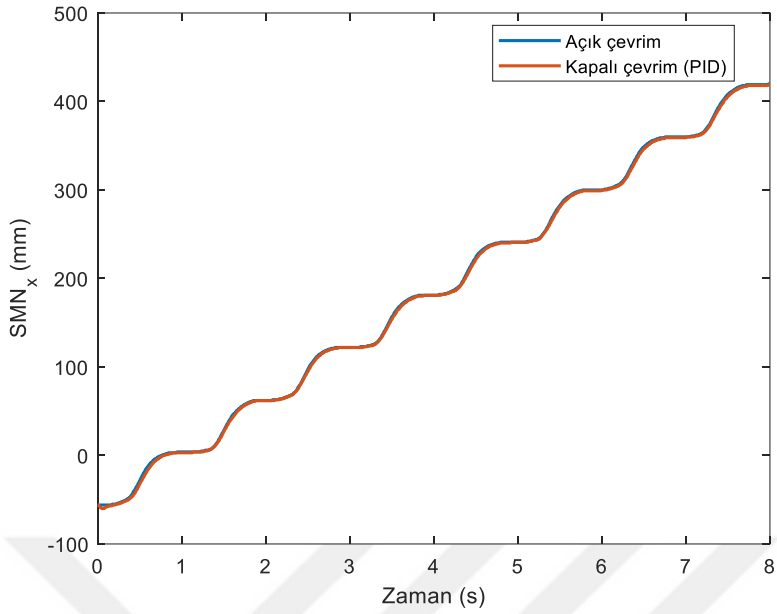


Şekil 7.45. Kalman filtresi ile hesaplanan yunuslama açılarının karşılaştırılması

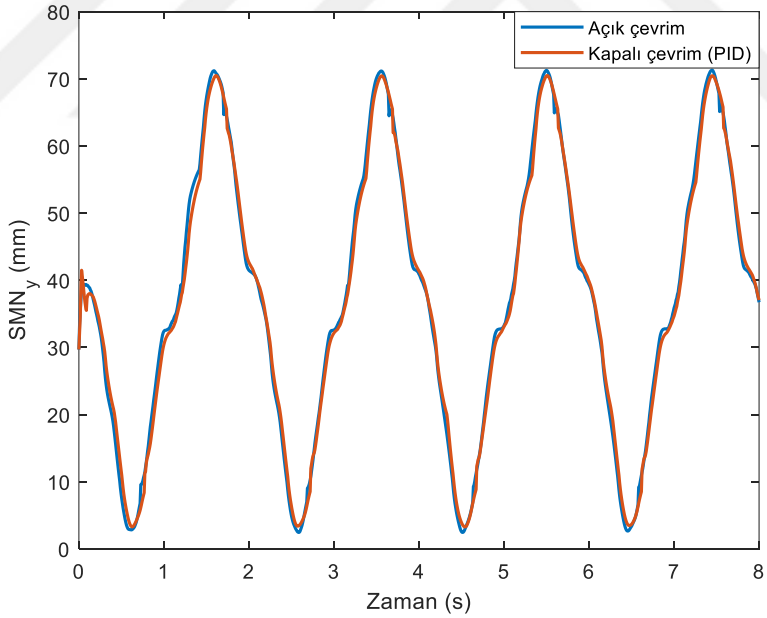


Şekil 7.46. Kalman filtresi ile hesaplanan yuvarlanma açılarının karşılaştırılması

Şekil 7.45 gözlemlendiğinde, robot açık çevrimde yürürken sallanma sonucu yunuslama açısında meydana gelen yaklaşık 1.4°'lik açı değişimleri, kapalı çevrimde 1°'den aşağıya inmiştir. Ancak, yürüyüşün başlangıcında PID kontrolör ile sistem cevabının referans girişe oturması için geçen sürede anlık bir sıçrama hareketi meydana gelmiştir. Şekil 7.46'daki yuvarlanma açıları gözlemlendiğinde ise birbirine oldukça yakın bir yanal sallanmanın gerçekleştiği görülmüştür.



Şekil 7.47. SMN_x değerlerinin açık ve kapalı çevrim karşılaştırılması

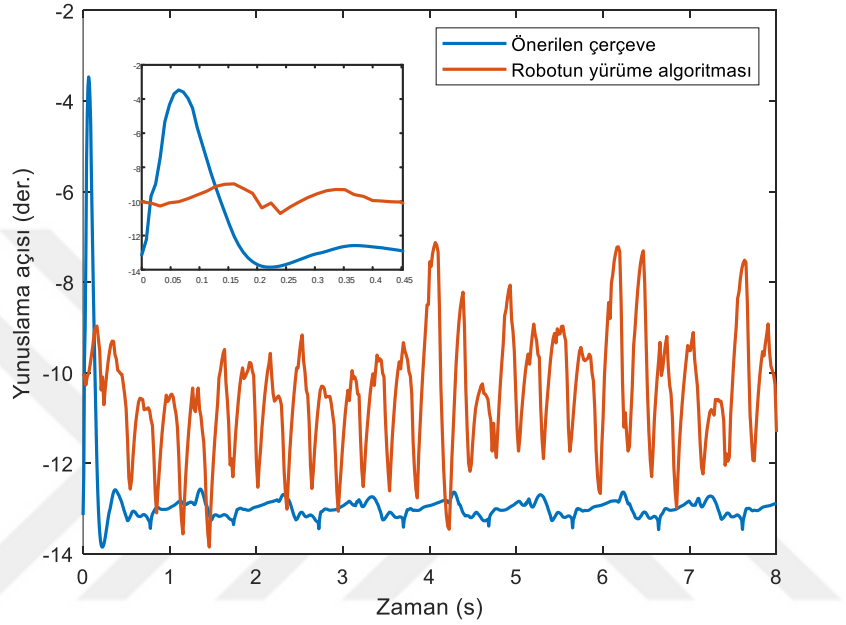


Şekil 7.48. SMN_y değerlerinin açık ve kapalı çevrim karşılaştırılması

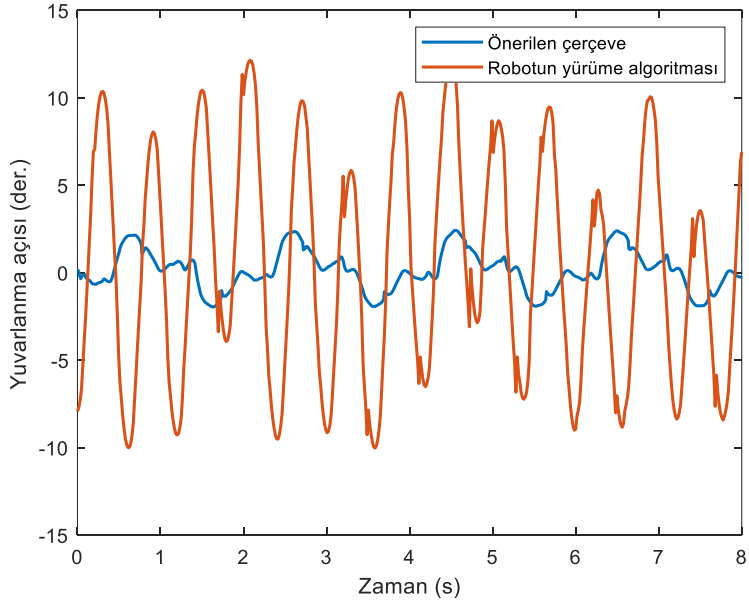
Şekil 7.47 ve Şekil 7.48'deki açık ve kapalı çevrim sonuçları karşılaştırıldığında kapalı çevrimde sadece başlangıç anında SMN değerlerinde bir bozulma meydana gelmiştir. Her iki eksendeki SMN değerlerinde yürüyüş boyunca birbirine oldukça yakın değerler elde edilmiştir. Ancak, y eksenindeki SMN değerlerinde açık çevrimde meydana gelen birkaç dalgalanma, kapalı çevrimde daha düzgün seyretmiştir.

7.3.4. Önerilen Çerçeve Sonuçlarının Karşılaştırılması

Önerilen çerçeve ve Bölüm 2.2'de detaylandırılan Robotis-OP2'nin kendi yürüme algoritması ile elde edilen yürüyüş için 8 s boyunca alınan sonuçlar, Şekil 7.49, Şekil 7.50, Şekil 7.51 ve Şekil 7.52'deki gibi karşılaştırılmıştır.



Şekil 7.49. Kalman filtresiyle hesaplanan yunuslama açılarının karşılaştırılması



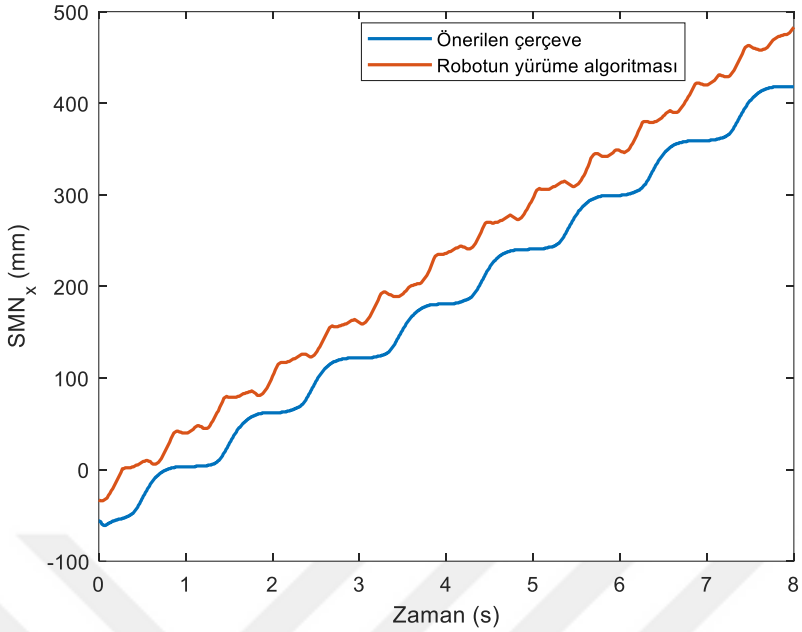
Şekil 7.50. Kalman filtresiyle hesaplanan yuvarlanma açılarının karşılaştırılması

Şekil 7.49 gözlemlendiğinde yürüyüşün başlangıcında PID kontrolör ile sistem cevabının referans girişe oturması için geçen sürede anlık bir sıçrama hareketi meydana gelmiştir. Ancak bu sıçramanın yaklaşık olarak yürümenin ilk 0.18 s'de olduğu için robotun yürüyüş kararlılığını olumsuz etkilemediği gözlenmiştir. Şekil 7.49 ve Şekil 7.50'deki yönelim açılarının salınımları birlikte incelendiğinde, önerilen çerçevenin robotun kendi yürüme algoritmasına kıyasla oldukça az bir açı değişimine sahip olduğu görülmüştür. Tablo 7.8'de, Şekil 7.49 ve Şekil 7.50'de verilen robot yönelim açılarının aralıkları karşılaştırılmıştır.

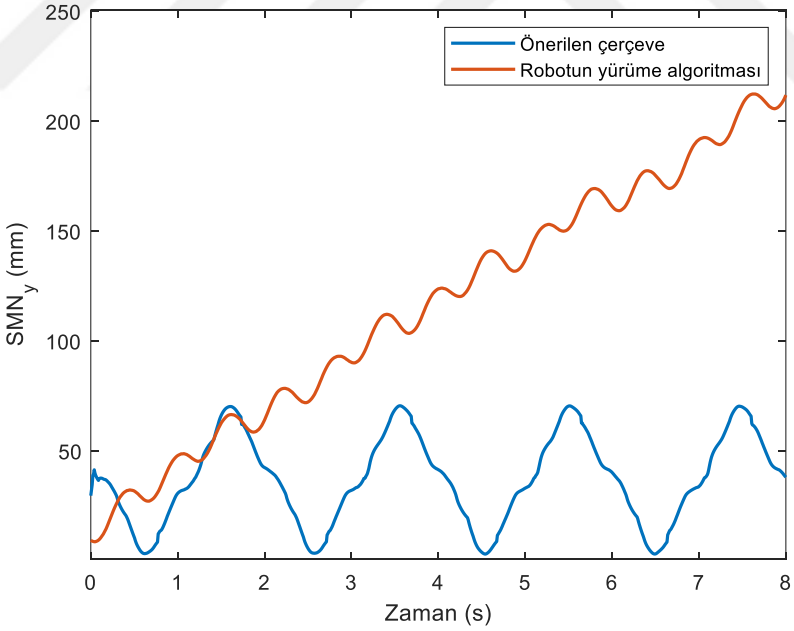
Tablo 7.8. İki yöntem için yönelim açılarının aralıkları

Yöntem	**Yunuslama açısı *(der.)***		**Yuvarlanma açısı *(der.)***	
	min.	**maks.**	**min.**	**maks.**
Önerilen çerçeve	-13.44	-12.99	-1.96	2.33
Robotun yürüme algoritması	-13.46	-7.13	-10.03	12.23

Tablo 7.8'e göre önerilen çerçeve ile robotun hem yunuslama hem de yuvarlanma yönelim açıları, yürüme algoritmasına kıyasla yürüme sırasında çok az salınıma sahip olduğu görülmüştür. Önerilen çerçeve ile sagital düzlemde robot gövdesinin $-13°$ eğimle yürümesi istendiğinde simülasyon ortamında robotun gövde yunuslama açısı $-13.44°$ ile $-12.99°$ arasında bir sallanma ile yürümüştür. Gövde yuvarlanma açısında ise $-1.96°$ ile $2.33°$ arasında bir sallanma meydana gelmiştir. Robotun kendi yürüyüş algoritması ile simülasyon ortamında robotun gövde yunuslama açısı $-13.46°$ ile $-7.13°$, gövde yuvarlanma açısı ise $-10.03°$ ile $12.23°$ arasında sallanmalar ile yürüyüş gerçekleşmiştir.



Şekil 7.51. SMN_x değerlerinin karşılaştırılması

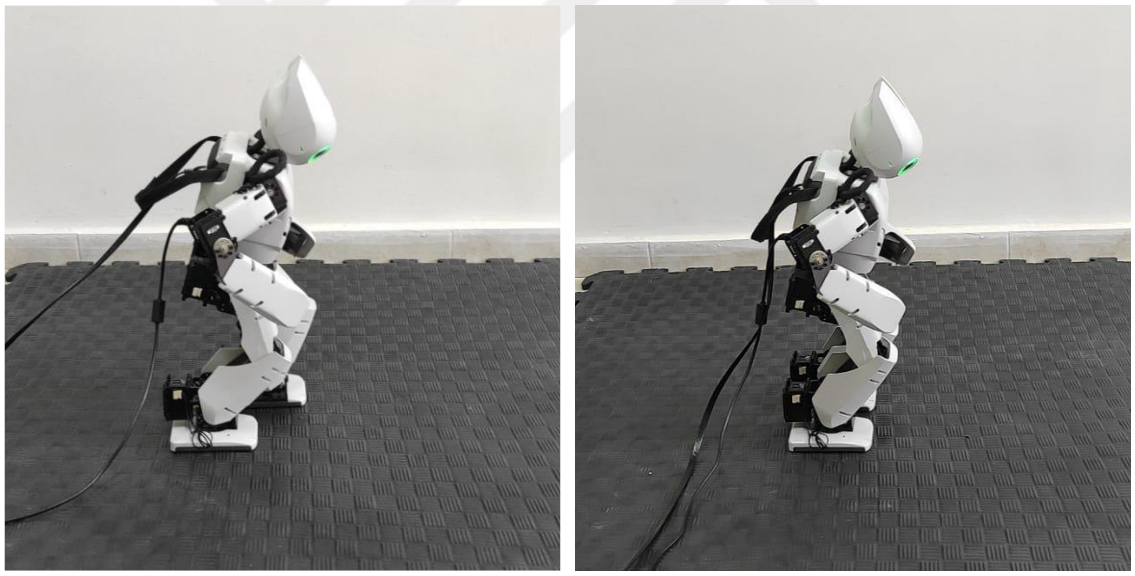


Şekil 7.52. SMN_y değerlerinin karşılaştırılması

Şekil 7.51 incelendiğinde önerilen çerçeve ile SMN_x'in robotun kendi yürüme algoritmasına kıyasla daha düzgün bir şekilde salınım yaparak arttığı görülmüştür. Adım uzunluğunun 60 mm olduğu düşünüldüğünde, robotun SMN_x'inin dört yürüyüş çevrimi için ileri yönde 480 mm mesafe kat etmesi beklendiğinde 474 mm hareket etmesi oldukça kararlı bir yürüyüş olduğunu göstermiştir.

Benzer şekilde Şekil 7.52 incelendiğinde, önerilen çerçeve ile SMN_y'nin belirli bir periyotta düzenli salınım yaptığı ve yürüyüşün sonunda sadece yaklaşık 12 mm dikey hizadan çıkma durumunun söz konusu olduğu görülmüştür. Diğer yöntem olan robotun kendi yürüme algoritmasıyla ise önerilen çerçeveye kıyasla dikey hizadan oldukça fazla sapmıştır. Bu deneysel sonuçlar, önerilen çerçeve ile oldukça kararlı bir yürüyüş elde edildiğini göstermiştir.

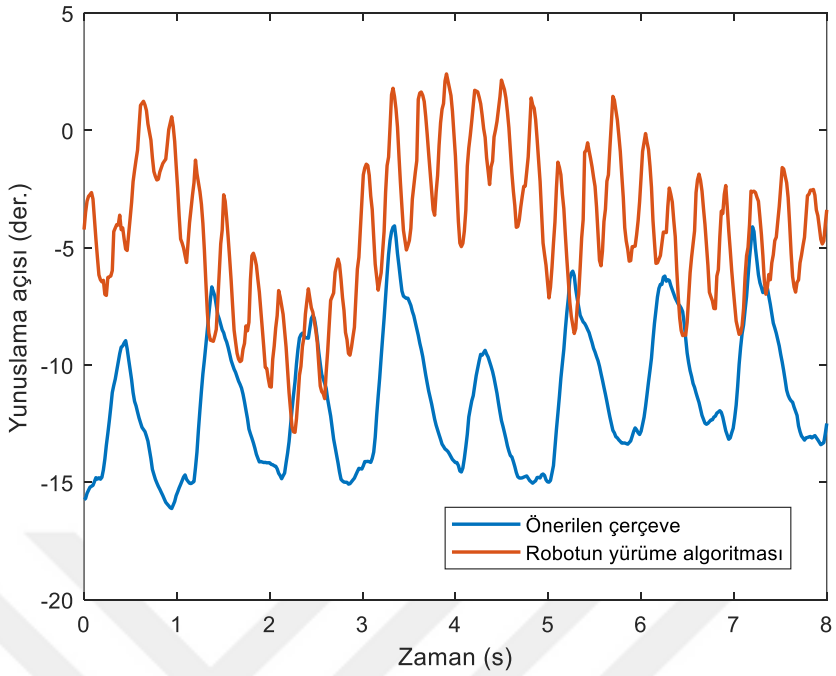
Webots'ta gerçekleştirilen deneysel çalışmalardan sonra geliştirilen kontrolör Webots'un robot penceresinde yer alan uzaktan kontrol aracı ile gerçek Robotis-OP2 insansı robotuna aktarılmıştır. Simülasyondan farklı olarak vücut duruş dengeleme için PID kontrolörün kazanç katsayıları gerçek robot üzerinde deneme yanılma ile K_p, K_i ve K_d sırasıyla 3.54, 0.52 ve 0.002 olarak belirlenmiştir. Şekil 7.53(a)'da önerilen çerçeve ile robotun yürüyüşe başlama durumuna, Şekil 7.53(b)'de ise Robotis-OP2'nin yürüme algoritması ile yürüyüşe başlama durumuna yer verilmiştir.



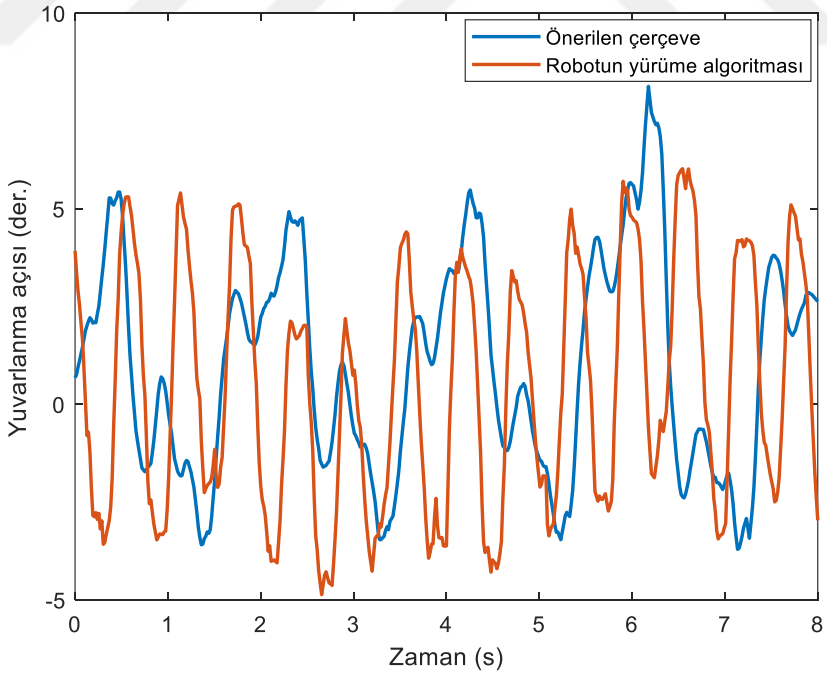
(a) (b)

Şekil 7.53. Robotun yürüyüşe başlama durumu: (a) önerilen çerçeve ve (b) robotun kendi yürüme algoritması

Önerilen çerçeve ve Robotis-OP2'nin kendi yürüme algoritması ile gerçek robot üzerinde 8 s süresince gerçekleştirilen yürüyüşün yunuslama ve yuvarlanma yönelim açılarının karşılaştırılmasına sırasıyla Şekil 7.54 ve Şekil 7.55'te yer verilmiştir.



Şekil 7.54. Gerçek robotun Kalman filtresiyle hesaplanan yunuslama açılarının karşılaştırılması



Şekil 7.55. Gerçek robotun Kalman filtresiyle hesaplanan yuvarlanma açılarının karşılaştırılması

Şekil 7.54 gözlemlendiğinde önerilen çerçeve ile yürüme esnasında yunuslama açısının salınım genliğinin daha az olduğu görülmüştür. Bu da robotun yürüyüşünün önerilen çerçeve ile

daha dengeli olduğunu kanıtlamıştır. Şekil 7.55'teki sonuçlar ise robotun yuvarlanma açılarının birbirine oldukça yakın salınımda olduğunu göstermiştir. Önerilen çerçeve ile robotun yunuslama açısının $-16.07°$ ile $-5.31°$ ve yuvarlanma açısının $-3.60°$ ile $5.42°$ arasında sallanma ile yürürken kendi yürüyüş algoritması ile robotun yunuslama açısının $-12.87°$ ile $2.40°$ ve yuvarlanma açısının $-3.69°$ ile $8.12°$ arasında sallanma ile yürümüştür.

Gerçek robot üzerinde yapılan deneysel sonuçlar da önerilen çerçeve ile robotun yürüme sırasında daha kararlı olduğunu kanıtlamıştır.

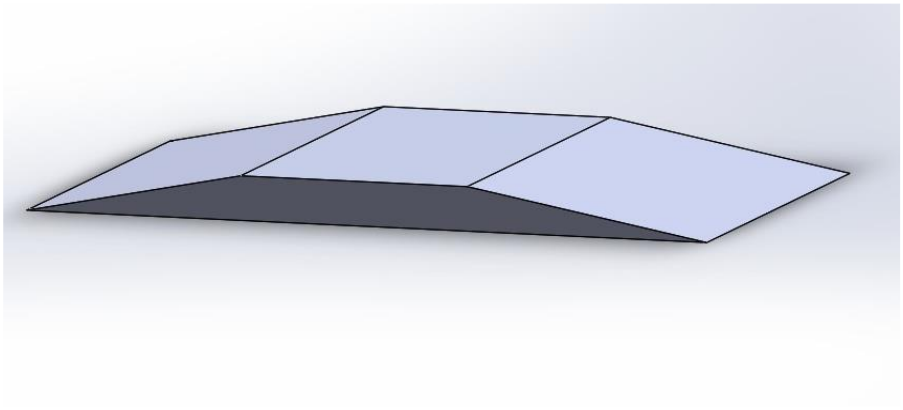
Hem simülasyon hem de gerçek robot üzerinde gerçekleştirilen tüm deneysel çalışmaların sonuçları değerlendirildiğinde önerilen çerçevenin robotun kararlı yürümesi açısından oldukça başarılı olduğu görülmüştür. Önerilen çerçeve, DPÖ'nün farklı durum ve eylem alanları için yürüyüş parametrelerini optimize etmek için de çok kullanışlıdır. Kullanıcı, robotun istenen hızı ve yürüme şablonu üreteci için önerilen çerçeveyi kullanarak robotun yürüyüşünü optimize edebilir.

7.4. Eğimli Yüzeylerde Yürüme için Gerçekleştirilen Çalışmalar

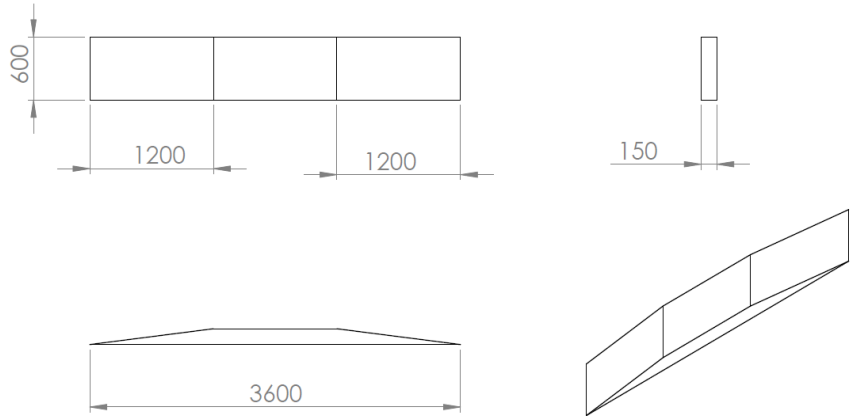
Tez çalışması kapsamında Robotis-OP2 robotunun dengeyi koruması ve eğimli yüzeylerde robotik belirsizliklere karşı başa çıkarak yürüyebilmesi için PID kontrolör ve DDPG algoritması tabanlı DPÖ kontrolörden oluşan iki ayrı yürüyüş dengeleme çerçevesi önerilmiştir.

Yürüyüşü sagital düzlemde dengelemek için kalça stratejisinden yararlanılmıştır. Robotta yuvarlanma etkisinin ihmal edilmesi için test ortamı olarak yokuş yukarı ve yokuş aşağı eğim yüzeyleri seçilmiştir. Eğimli bir yürüme parkurunda kararlı bir şekilde yürüyebilme ile ilgili yapılan simülasyon çalışmalarında, robotun yürüyüşü sırasında tek destek aşamasında diğer ayağı yere değmediği için eğim geçişleri robot tarafından deneyimlenmiştir. Bu aşamada robot, bir dayanak üzerinde sallanıyormuş gibi davranmakta ve açık çevrim kontrol ile robot bu sallanma işlemi sırasında düşmektedir.

Simülasyon için oluşturulan eğimli yürüme parkuru, SolidWorks kullanılarak modellenmiştir. Simülasyon ortamı için oluşturulan parkur, gerçek uygulama için üretilen parkur ile birebir aynı olacak şekilde boyutlandırılmıştır. Şekil 7.56'da, $7.125°$'lik yokuş yukarı ve yokuş aşağı eğime sahip rampalara sahip olan parkurun katı modeli ve teknik resmi verilmiştir.



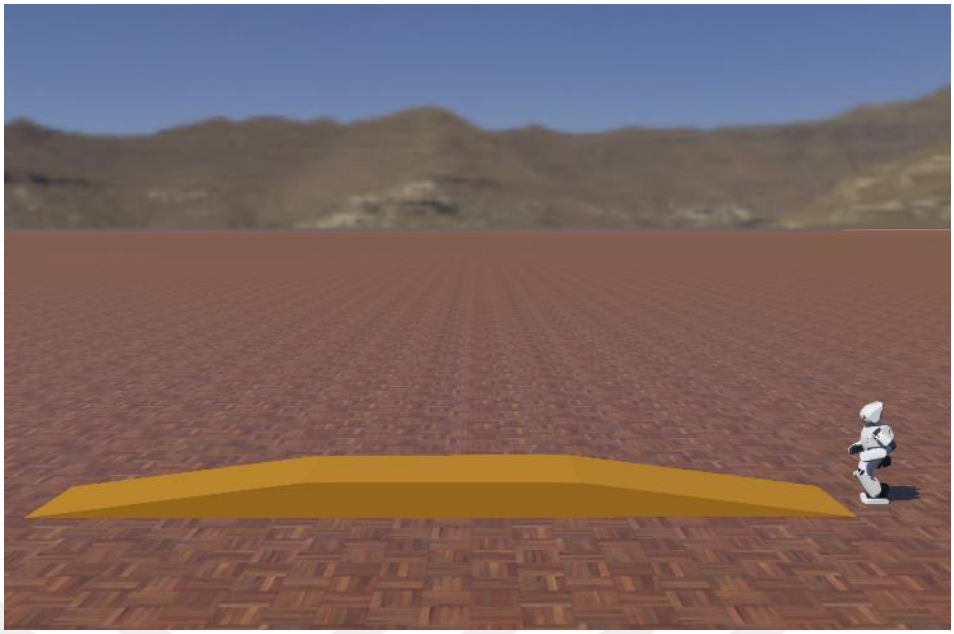
(a)



(b)

Şekil 7.56. Eğim parkuru: (a) katı modeli ve (b) teknik resmi

Parkur çizildikten sonra .SLDPRT uzantılı dosya, .STL formatına dönüştürülmüş ve daha sonra 3B model içe aktarılarak Webots simülatörüne dahil edilmiştir. Parkur, istenen bir renk ile renklendirilerek deneysel çalışmalar için Şekil 7.57'deki gibi hazır hale getirilmiştir.



Şekil 7.57. Webots ortamında oluşturulan eğimli deneysel düzenek

Daha önce düz yürüme için yürüyüş parametreleri Düello ÇDQA'nın eğitimi ile optimize edilmiş ve Tablo 7.7'deki gibi elde edilen yürüyüş parametreleri ($w = 50$, $s = 60$, $h = 30$, $DSR = 0.6$, $\theta_r = -13$) ile düz bir zeminde yürüme için deneysel sonuçlar elde edilmişti. Eğimli yüzeydeki yürüme çalışmalarında dengenin daha kolay sağlanabilmesi için daha küçük adım uzunluğu içeren parametre dizileri incelenmiştir. Bunun için daha önce gerçekleştirilen parametre optimizasyon çalışmasında $DSR = 0.6$, $\theta_r = -13$ değerleri ortak olan, Düello ÇDQA'nın eğitimi sırasında yakınsadığı değerler incelenerek Tablo 7.9'daki parametreler seçilmiştir.

Tablo 7.9. Eğimli yüzey çalışmaları için seçilen yürüyüş parametreleri

Parametre	Değer (mm)
w	45
s	30
h	25

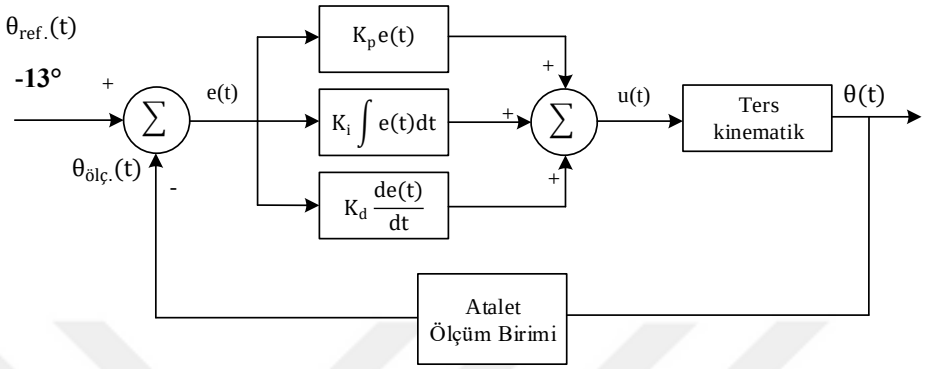
7.4.1. PID Kontrolör ile Eğimli Yüzeylerde Yürüme

Deneysel çalışmalar için ilk olarak PID kontrolör ile kapalı bir çevrim gerçekleştirilerek eğimlerden oluşan parkurda, kalça stratejisi kullanılarak Bölüm 7.3.3.'te önerilen duruş dengeleme sistemi kullanılmıştır. Bu sistemin ana fikri, bilinmeyen bir eğim ortamında robotun duruşunu gerçek zamanlı olarak ayarlayıp düz bir zeminde yürüyormuş gibi gövde yunuslama açısının istenen referans değerde olmasını sağlamaktır. Önerilen PID kontrolör ile robot, eğimli yüzeye geçiş sırasında dengesini korumak için kalça yunuslama açısını değiştirerek önemli bir sallanmayla duruşunu ayarlamaktadır. Böylece robot, vücut yunuslama açısını sabit tutmaya çalışarak ortama uyum sağlamakta ve eğimli yüzeyde dengeli bir şekilde yürüyebilmektedir.

Şekil 7.57'de yer alan parkurda yürüme deneyleri için robotun başlangıç konumu, ilk iki yürüme çevriminde düz bir zeminde yürüyecek şekilde ayarlanmıştır. Üçüncü çevrimde ise Şekil 7.58'de görüldüğü gibi robot, sağ ayağının önü ve ardından sol ayağının ortası ile geçiş noktasına basmaktadır. Daha sonra, robotun duruşunu değiştirip vücudunu dengede tutarak yokuşu tırmandıktan sonra düz zemine kadar ilerlemesi beklenmektedir. Robot düz zeminde de yürüdükten sonra aynı açıdaki eğime sahip yokuş aşağı yüzeye de geçişi başarılı bir şekilde gerçekleştirip dengesini koruyarak robotun parkuru tamamlaması beklenmektedir.

Şekil 7.58. Yürümenin üçüncü çevriminde robotun konumu

Robotun duruşunu ayarlayarak yürümeye başladığı andan itibaren parkuru gövde yunuslama açısının referans değeri $-13°$ olacak şekilde tamamlaması amaçlanmıştır. Şekil 7.59'da PID kontrolör ile kapalı çevrim kontrol blok diyagramı gösterilmiştir.



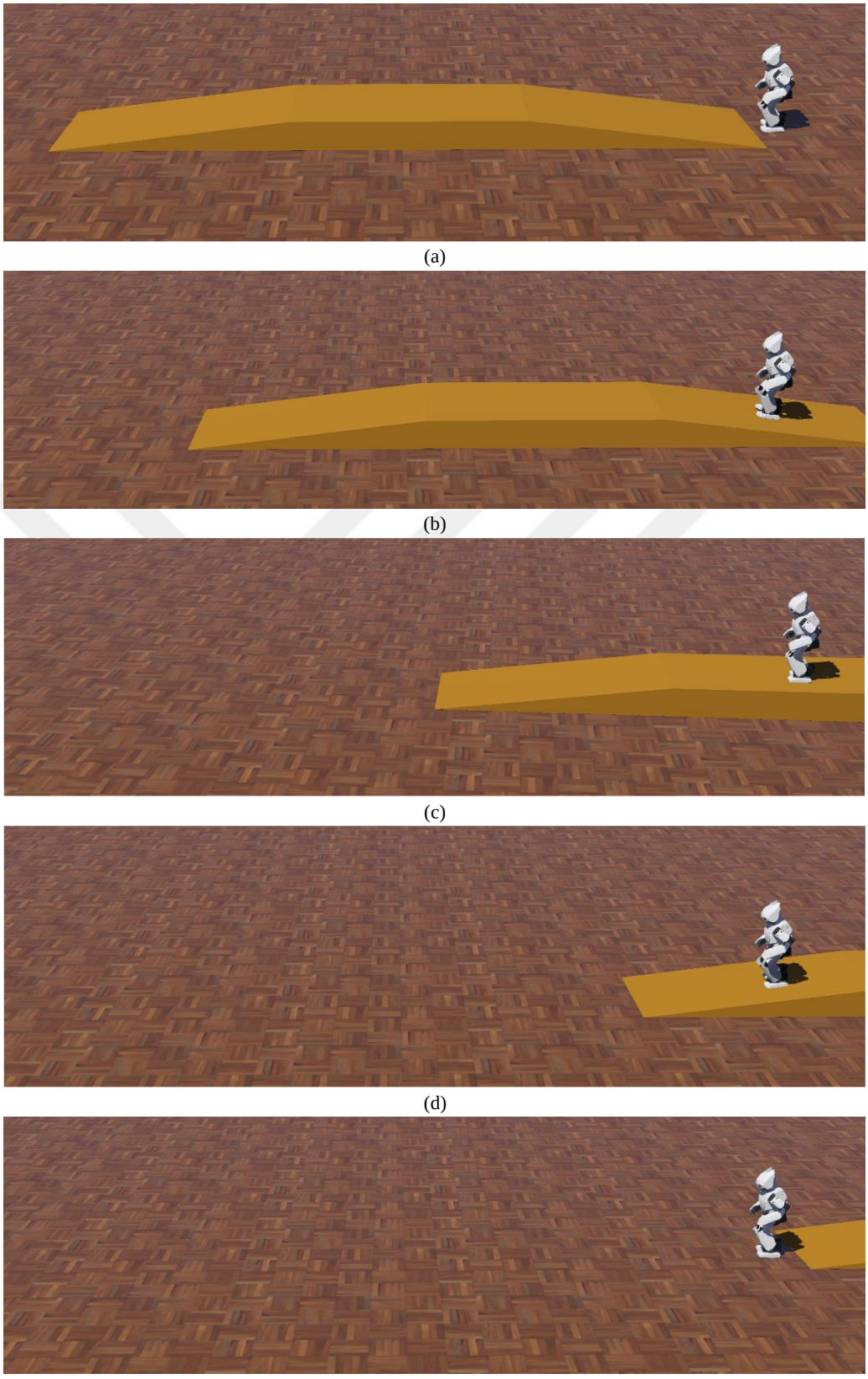
Şekil 7.59. Parkur için oluşturulan kapalı çevrim blok diyagramı

PID kontrolörün katsayıları, daha önce düz yürüme için gerçekleştirilen kapalı çevrimde robot gövdesinin yunuslama açısının kontrolünde ayarlandığı gibi Tablo 7.9'daki yürüyüş parametreleri için deneme yanılma ile tekrar belirlenmiştir. Eğimli yüzeyler için de deneme yanılma ile PID kontrolörün kazanç katsayıları belirlenmiştir. Düz ve eğimli yüzeyler için elde edilen kazanç katsayıları Tablo 7.10'da yer almaktadır.

Tablo 7.10. Düz ve eğimli yüzey için PID kazanç katsayıları

Parametre	**Değer**	
	Düz yüzey	***Eğimli yüzey***
K_p	0.75	0.3
K_i	3.7	4.2
K_d	0.007	0.002

Robotun yürümeye başladığı ilk andan itibaren parkuru tamamlamasına kadar olan süreç Şekil 7.60'taki gibi beş aşamada gösterilmiştir.



(a)

(b)

(c)

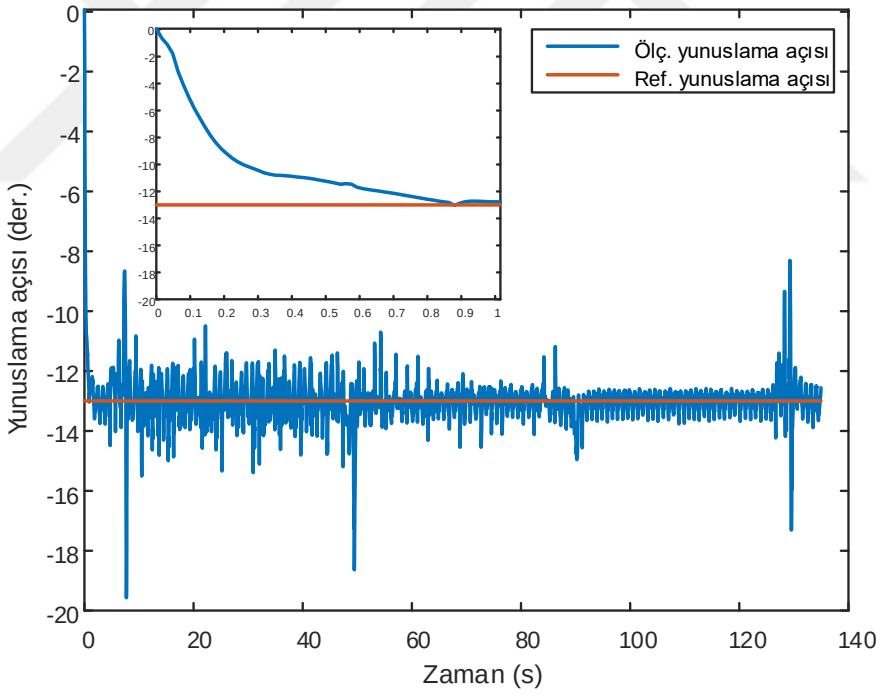
(d)

(e)

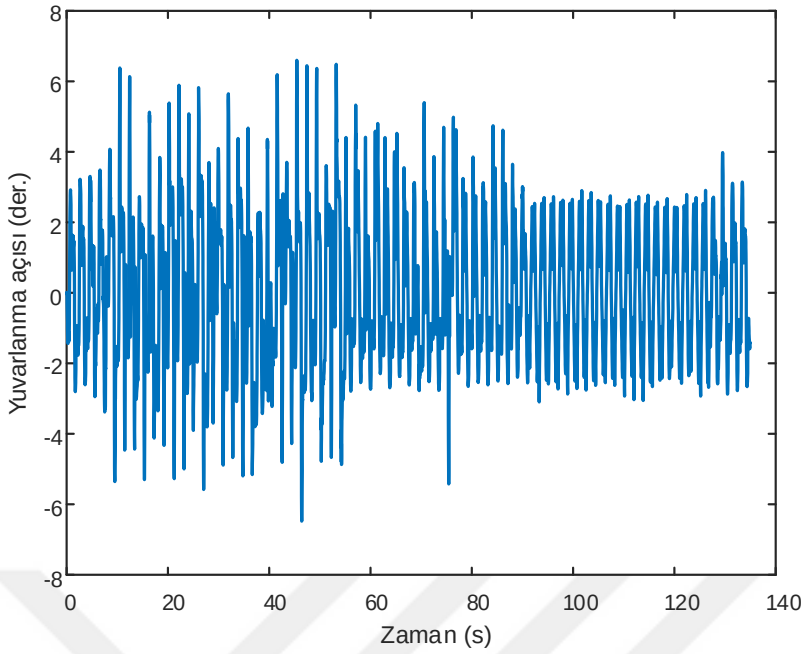
Şekil 7.60. Parkur tamamlama görevi için farklı zeminlerde yürüyüş süreci: (a) düz yüzey, (b) yokuş yukarı yüzey, (c) düz yüzey, (d) yokuş aşağı yüzey ve (e) düz yüzey

İlk olarak, Şekil 7.60(a)'da robot yaklaşık 0-4 s aralığında düz bir şekilde yürümüştür. İkinci olarak, AÖB ile robotun düz zeminden ayrıldığı tespit edilir edilmez PID kazanç katsayıları değiştirilerek yaklaşık 4-48 s aralığında Şekil 7.60(b)'deki gibi robot, gövdesini düz yürümedekinden daha fazla öne eğip vücut yunuslama açısını koruyarak yokuş yukarı yüzeyde yürümüştür. Üçüncü olarak, aynı şekilde yüzey geçişi algılanıp kazanç katsayıları değiştirilerek Şekil 7.60(c)'deki gibi yaklaşık 48-90 s aralığında robot düz bir şekilde yürümüştür. Dördüncü olarak, yüzey geçişiyle kazanç katsayılar değiştirilerek robot, gövdesini arkaya eğip Şekil 7.60(d)'deki gibi yaklaşık 90-131 s aralığında yokuş aşağı yürümüştür. Son olarak, düz yüzeye geçiş ile kazanç katsayıları değiştirilerek Şekil 7.60(e)'deki gibi düz yürüme işlemi ile görev sonlandırılmıştır.

Webots ortamında Robotis-OP2'nin parkur görevini tamamlama esnasındaki vücut yunuslama ve yuvarlanma açılarının Kalman filtresi ile hesaplanmasına sırasıyla Şekil 7.61 ve Şekil 7.62'de yer verilmiştir.



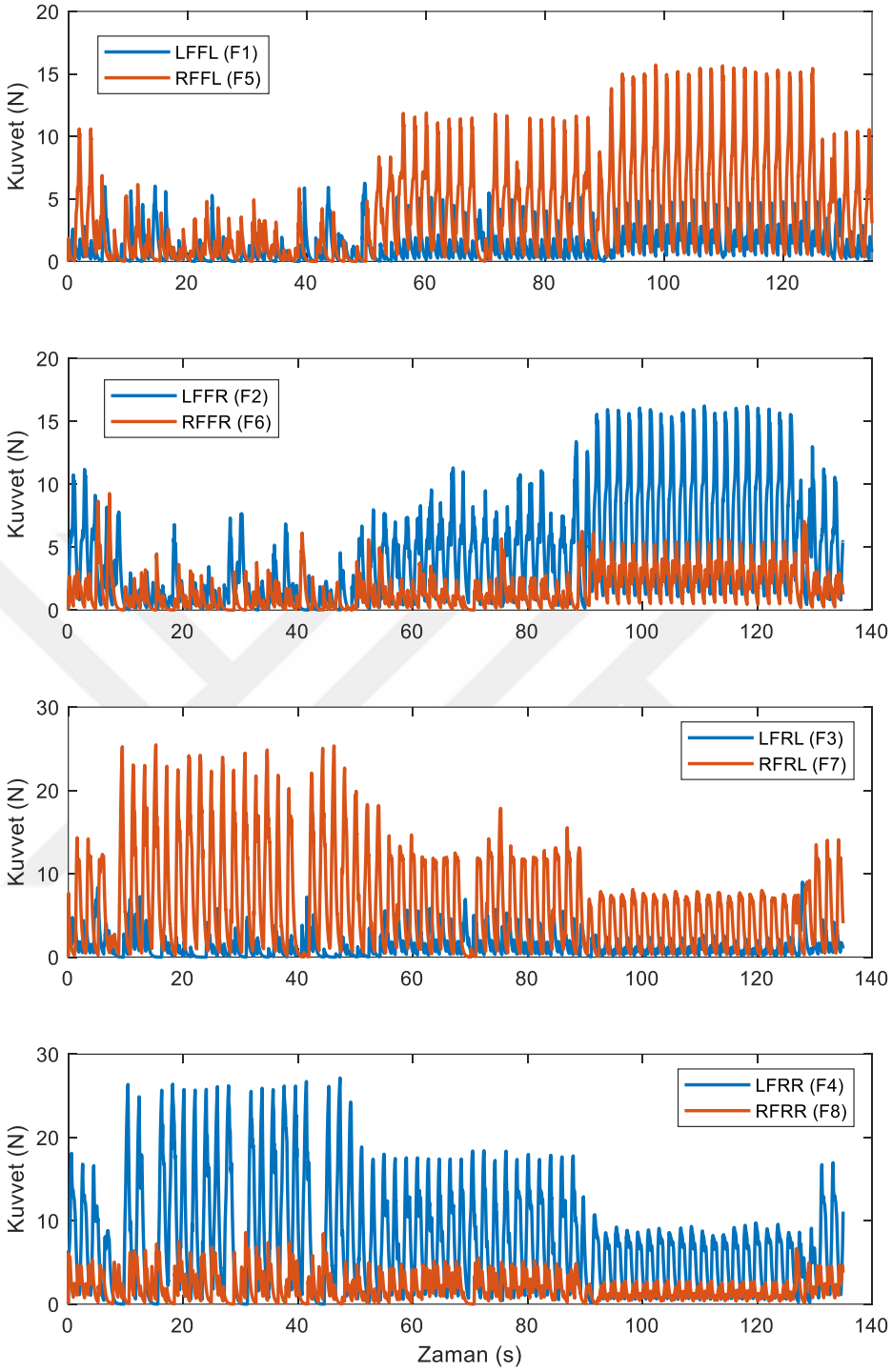
Şekil 7.61. Parkur görevinde Kalman filtresi ile hesaplanan yunuslama açısı



Şekil 7.62. Parkur görevinde Kalman filtresi ile hesaplanan yuvarlanma açısı

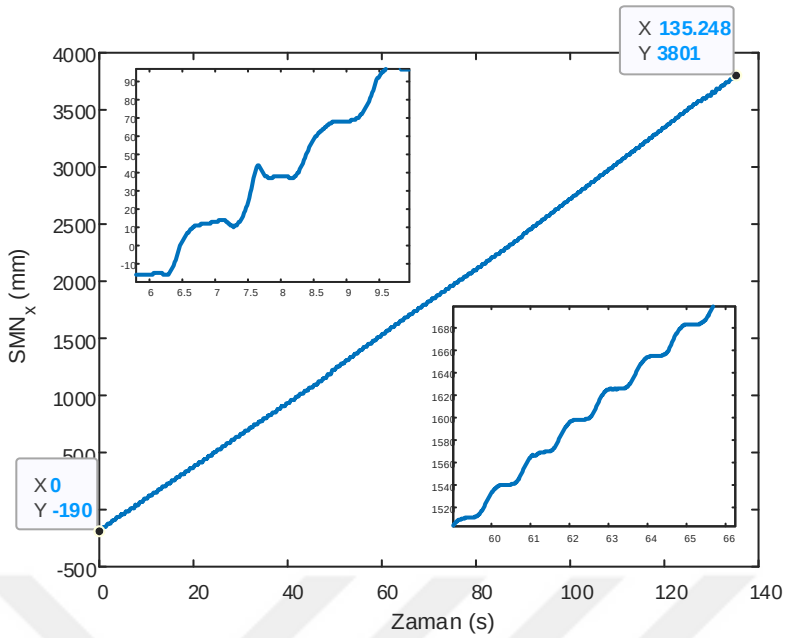
Şekil 7.61 incelendiğinde robot yunuslama açısının 0°'den istenen referans değere yaklaşık olarak 0.9 s'de oturduğu görülmüştür. Ayrıca, Şekil 7.61 ve Şekil 7.62 birlikte incelendiğinde robotun yokuş yukarı yürürken yokuş aşağı yürümesine kıyasla daha çok sallanma eğiliminde olduğu görülmüş, ancak bu durumun robotun kararlılığını etkileyerek düşmesine ve düz yürümeden sapmasını etkileyecek derecede olmadığı sonucuna varılmıştır.

Parkur görevi için gerçekleştirilen yürüyüş sırasında KDD'lerden okunan değerlerin bir boyutlu Kalman filtresinden geçtikten sonra sol ve sağ ayağın temas kuvvetleri, MATLAB ortamında Şekil 7.63'te verildiği gibi görselleştirilmiştir.

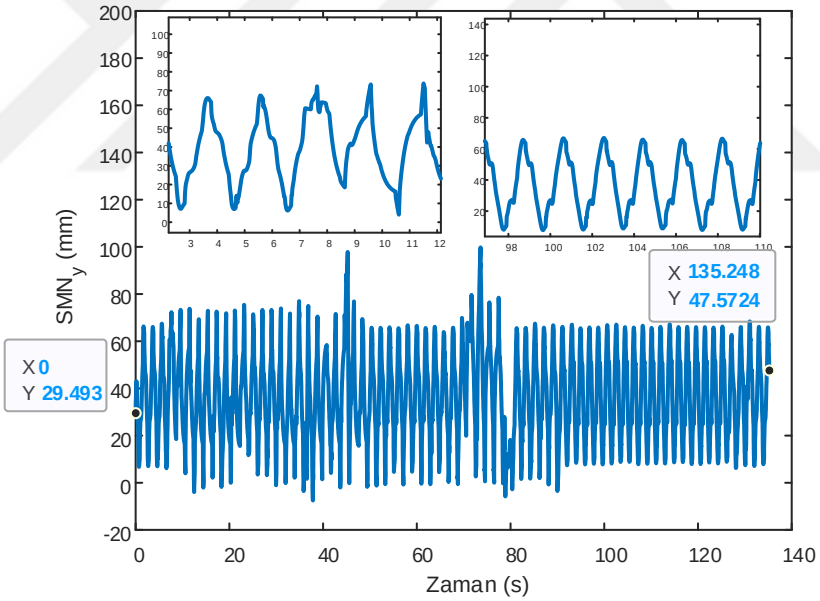


Şekil 7.63. Parkur görevinde sekiz KDD'den okunan sol ve sağ ayağın temas kuvvetleri

Şekil 7.63'teki kuvvet değerleriyle hesaplanan SMN_x ve SMN_y koordinatları ise sırasıyla Şekil 7.64 ve Şekil 7.65'te gösterilmiştir.



Şekil 7.64. Parkur görevindeki yürüyüş sırasında hesaplanan SMN_x değerleri

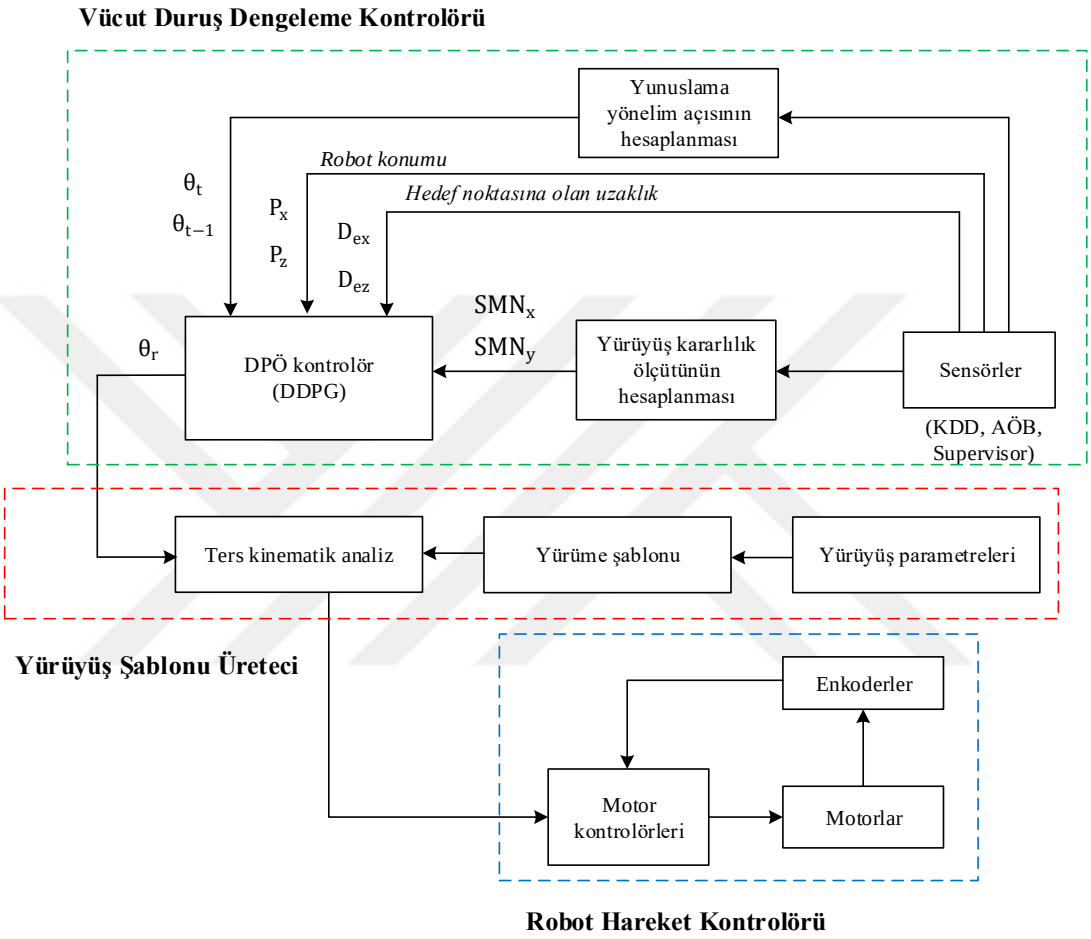


Şekil 7.65. Parkur görevindeki yürüyüş sırasında hesaplanan SMN_y değerleri

Şekil 7.64 incelendiğinde x eksenindeki SMN koordinatlarının düzgün bir şekilde salınım yaparak arttığı görülmüştür. Benzer şekilde, Şekil 7.65 incelendiğinde ise y eksenindeki SMN koordinatlarının yokuş yukarı yürürken biraz gürültülü bir salınım yapmış olsa da daha sonra belirli bir periyotta düzgün bir salınım yaptığı görülmüştür.

7.4.2. DDPG Kontrolör ile Eğimli Yüzeylerde Yürüme

Eğimli yüzeylerde kararlı bir yürüme gerçekleştirebilmek için tez kapsamında, DPÖ algoritmalarından DDPG kullanılarak önerilen çerçeve Şekil 7.66'da verilmiştir.



Şekil 7.66. Eğimli yüzeylerde yürüme için DDPG tabanlı önerilen çerçeve

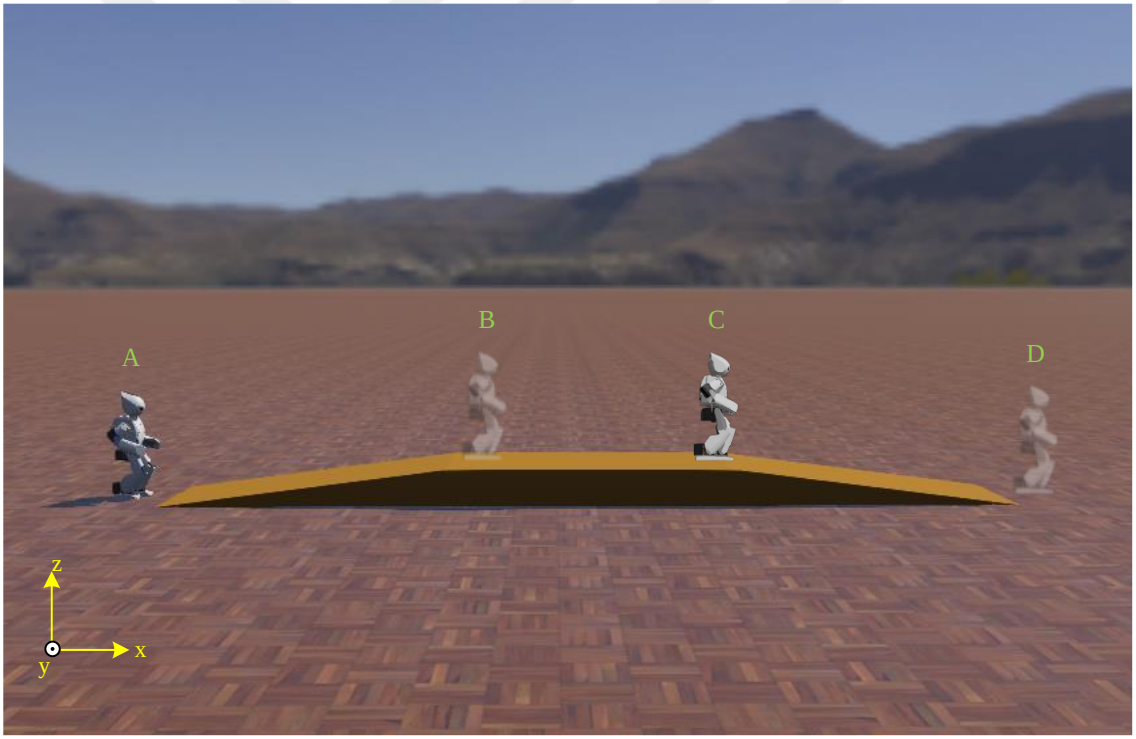
Şekil 7.66'da görüldüğü gibi robotun eğimli yüzeylerde dengeli yürüyüşünün sağlanabilmesi için robotun gövde yunuslama açısını kontrol eden θ_r parametresiyle sıralı hareket dizisinin öğrenilmesi işlemi, DDPG ile gerçekleştirilmiştir. Robotun eğimli yüzeylere geçişinin sürekli eylem uzayında daha hassas bir şekilde gerçekleştirilebileceği düşünüldüğünden sürekli eylem uzayında çalışan DDPG algoritmasının kullanımı tercih edilmiştir.

PID kontrolör ile gerçekleştirilen yürüme ile karşılaştırılmaların yapılabilmesi için DDPG ile önerilen çerçeve için de Tablo 7.9'daki yürüyüş parametreleri kullanılmıştır.

DDPG'nin eğitimi, Webots simülatörü üzerinde PID kontrolör ile gerçekleştirilen deneysel çalışmalarda kullanılan aynı eğimli parkurun bulunduğu çevrede gerçekleştirilmiştir.

DDPG ile Eğitim

PID kontrolör ile gerçekleştirilen deneysel çalışmalarda, Şekil 7.67'de gösterilen A başlangıç noktasından D bitiş noktasına kadar tek bir seferde parkurun tamamlanması sağlanmıştı. Ancak, DDPG algoritması sürekli eylem uzayında çalıştığından ağın başarılı bir öğrenme işlemi için eylem uzayının mümkün olabildiğince küçük aralıklarda olmasına dikkat edilmelidir. Yokuş yukarı ve yokuş aşağı görevlerinin tek bir DPÖ kontrolörde DDPG'nin eğitiminin gerçekleştirilebilmesi için eylem uzayının en küçük ve en büyük değer aralığı geniş olması gerekmektedir. Bu durum da eğitimin verimliliğini sınırlandırmaktadır. Bu yüzden, eğimli yüzeylerde gerçekleştirilen deneysel çalışmalar, yokuş yukarı ve yokuş aşağı yüzeylerde yürüme olarak iki kısımda gerçekleştirilmiştir. Böylece, robotun A başlangıç noktasından B bitiş (hedef) noktasına ulaşması ve C başlangıç noktasından D bitiş (hedef) noktasına ulaşması için iki ayrı DDPG tabanlı kontrolör eğitilmiştir.



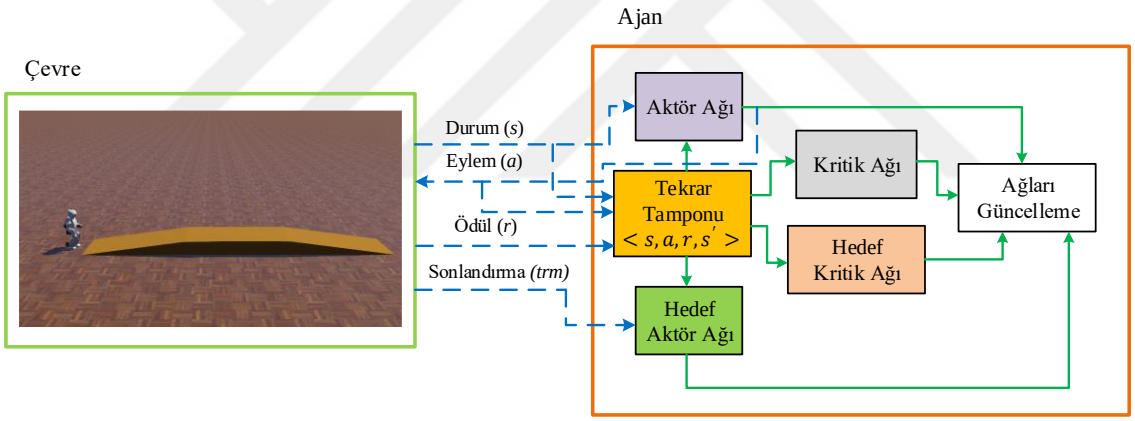
Şekil 7.67. Robotun eğimli yüzeylerde yürüme görevinde başlangıç ve bitiş noktaları

Webots simülatöründe robotun oluşturulan çevredeki başlangıç ve bitiş noktalarının konum koordinatları Tablo 7.11'de verilmiştir.

Tablo 7.11. Başlangıç ve bitiş noktalarının konum koordinatları

Başlangıç/Bitiş Noktaları	**Koordinatlar (m)**	
	x ekseni	***z ekseni***
A	-0.170	0.318
B	1.359	0.468
C	2.310	0.468
D	3.839	0.318

Bölüm 4.4.1'de detaylandırılan DDPG'nin ana blokları ve alt blokları Şekil 7.68'de gösterilmiştir.



Şekil 7.68. DDPG sisteminin blok diyagramı

Şekil 7.68'de blok diyagramı verilen DDPG'nin hem yokuş yukarı hem de yokuş aşağı yüzeylerde yürümesi için gerçekleştirilen deneysel çalışmalarda kullanılan ağ modelleri aynıdır. Yalnızca, eğimli yüzeye göre eylem uzayları değiştirilerek ağlar ayrı ayrı eğitilmiştir.

Eğimli yüzeylerde yürüme uygulaması için ağ eğitiminde kullanılan DDPG, optimal politikayı Algoritma 7.2'de verilen sözde kodu gerçekleştirerek öğrenir.

Algoritma 7.2. Eğimli yüzeylerde yürüme için DDPG

Eğimli yüzeylerde yürüme sistemini başlat (Çevre)

θ^Q ve θ^μ ağırlıkları ile $Q(s,a|\theta^Q)$ kritik ağını ve $\mu(s|\theta^\mu)$ aktör ağını başlat

θ^Q ve θ^μ'ne göre Q' ve μ' hedef ağlarının ağırlıklarını ayarla

Tekrar tamponu belleğini başlat

for $bölüm = 1, M$ **do**

Eylem keşfi için rastgele bir η gürültü başlat

Çevreyi yeniden başlat ve s_t durumunu al

while $sonlandırma\ yok\ (robot\ düşmedi)$ **do**

$a_t = \mu(s_t|\theta^\mu) + \eta_t$ eylemini seç

a_t eylemini yürüt ve r_t ödülünü ve s_{t+1} yeni durumunu al $r_t, s_{t+1}, trm = Çevre(a_t)$

(s_t, a_t, r_t, s_{t+1}) deneyim demetini tekrar tamponunda sakla

Tekrar tamponundan N tane örnek al

$y_i = \begin{cases} r_i, & eğer\ s_{t+1}\ terminal\ ise \\ r_i + \gamma Q'\left(s_{t+1}, \mu'\left(s_{t+1}; \theta^{\mu'}\right)|\theta^{Q'}\right), & aksi\ taktirde \end{cases}$ ayarla

$L = \frac{1}{N}\sum_i [y_i - Q(s_i, a_i|\theta^Q)]^2$ ile kritik ağı güncelle

$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q\left(s_i, \mu(s_i)\right)\nabla_{\theta^\mu}\mu(s_i|\theta^\mu)$ ile aktör ağı güncelle

Hedef ağları güncelle:

$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$

$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$

end for

end for

Ağ eğitimlerinde kullanılan durum uzayı, eğimli yüzeylerde yürüme görevinde önemli bilgiler içeren 8 boyutlu bir diziden oluşmaktadır. 8 boyutlu dizinin elemanlarına Tablo 7.12'de yer verilmiştir.

Tablo 7.12. Durum uzayı

Durum	**Boyut**
Yunuslama açısı (θ_t, θ_{t-1})	2
Konum (P_x, P_z)	3
Hedef noktasına olan uzaklık (D_{ex}, D_{ez})	1
Sıfır Moment Noktası (SMN_x, SMN_y)	2

Burada; θ_t ve θ_{t-1} sırasıyla robot gövdesinin Kalman filtresi kullanılarak hesaplanan anlık ve bir önceki zaman adımındaki yunuslama açılarını, P_x ve P_z sırasıyla robotun x ve z eksenlerindeki konumlarını, D_{ex} ve D_{ez} sırasıyla robotun x ve z eksenlerindeki hedef noktasına olan uzaklıkları, SMN_x ve SMN_y ise sırasıyla x ve y eksenlerindeki SMN'leri ifade etmektedir.

Eylem uzayı yalnızca, kararlı yürüme uygulaması için gerçekleştirilen çalışmalarda kullanılan robotun θ yunuslama açısını belirleyen θ_r'den oluşmaktadır. DDPG ile verimli bir ağ eğitiminin gerçekleştirilebilmesi adına daha önce bahsedildiği gibi yokuş yukarı ve yokuş aşağı yüzeyler için ağ eğitimleri ayrı ayrı gerçekleştirilmektedir. Robot yokuş yukarı yürürken gövdesini öne doğru eğmeliyken, yokuş aşağı inerken gövdesini arkaya doğru eğmelidir.

PID kontrolör ile gerçekleştirilen eğimli yüzeylerde yürüme için robotun referans yunuslama açısı $-13°$ olarak belirlenmişti. Gerekli karşılaştırmaların yapılabilmesi ve robotun belirtilen referans değerlere yakın değerlerde yürüyebilmesi için eğimli yüzeyler için sürekli eylem uzayı Tablo 7.13'te verildiği gibi sınırlandırılmıştır.

Tablo 7.13. Farklı yüzeyler için eylem uzayının aralıkları

Parametre	**Aralık (derece)**	
	Yokuş yukarı yüzey	***Yokuş aşağı yüzey***
θ_r	$[-10, -23]$	$[-16, -3]$

Eğimli yüzeylerde yürüme için kullanılan R temel ödül fonksiyonu ise her iki eğimli yüzey için de (7.5) ile ifade edilmiştir.

$$R = R_1 + R_2 + R_3 + R_4 + R_5 \tag{7.5}$$

R_1, anlık olarak hesaplanan yunuslama açısının istenen referans değerden ($-13°$) uzaklaştığı veya yakınlaştığı durumlarda elde edilen ödülü temsil etmektedir. İlk olarak, referans değer ile anlık olarak ölçülen yunuslama açıları arasındaki θ_e mutlak hatası hesaplanır. Mutlak hata belirlenen eşik değerden büyük ise ceza, küçük ise ödül alacak şekilde (7.6)'daki gibi ayarlanmıştır.

$$R_1 = \begin{cases} -20\frac{e^{0.1\theta_e}}{403.42}, & eğer\ \theta_e > 5\ ise \\ 5e^{-0.1\theta_e}, & aksi\ taktirde \end{cases} \quad (7.6)$$

R_2, robotun hedef noktasına (B veya D) x ekseninde olan uzaklığına göre elde edilen ödülü temsil etmektedir. Öncelikle, robotun x ekseni üzerinde belirtilen hedef noktaya olan D_{ex} mesafesi (m) belirlenir. Daha sonra, robotun hedefe ulaşması için kat etmesi gereken mesafe değeri belirlenen eşik değerden büyük ise ceza, küçük ise ödül alacak şekilde (7.7)'de verildiği gibi ayarlanmış olup hedefe yaklaştıkça alınacak ödülün üstel olarak artması istenmiştir.

$$R_2 = \begin{cases} 2.5\ e^{-D_{ex}}, & eğer\ D_{ex} < 1.2\ ise \\ -5\frac{e^{D_{ex}}}{11}, & aksi\ taktirde \end{cases} \quad (7.7)$$

R_3, robotun hedef noktasına (B veya D) z ekseninde olan uzaklığına göre elde edilen ödülü temsil etmektedir. Öncelikle, robotun z ekseni üzerinde belirtilen hedef noktaya olan D_{ez} mesafesi (m) belirlenir. Daha sonra, robotun hedefe ulaşması için kat etmesi gereken mesafe değeri belirlenen eşik değerden büyük ise ceza, küçük ise ödül alacak şekilde (7.8)'de verildiği gibi ayarlanmış olup hedefe yaklaştıkça alınacak ödülün üstel olarak artması istenmiştir.

$$R_3 = \begin{cases} 1.5e^{-D_{ez}}, & eğer\ D_{ez} < 0.11\ ise \\ -e^{5D_{ez}}, & aksi\ taktirde \end{cases} \quad (7.8)$$

R_4, yürüme esnasında robotun kararlı bir şekilde yürümesini ifade eden ödülü temsil etmektedir. Her zaman adımında anlık olarak hesaplanan x ve y eksenlerindeki SMN, robotun destek poligonu içinde olduğunda ödül, aksi durumda ise ceza alması istenmiştir. R_4, (7.9)'da verilmiştir.

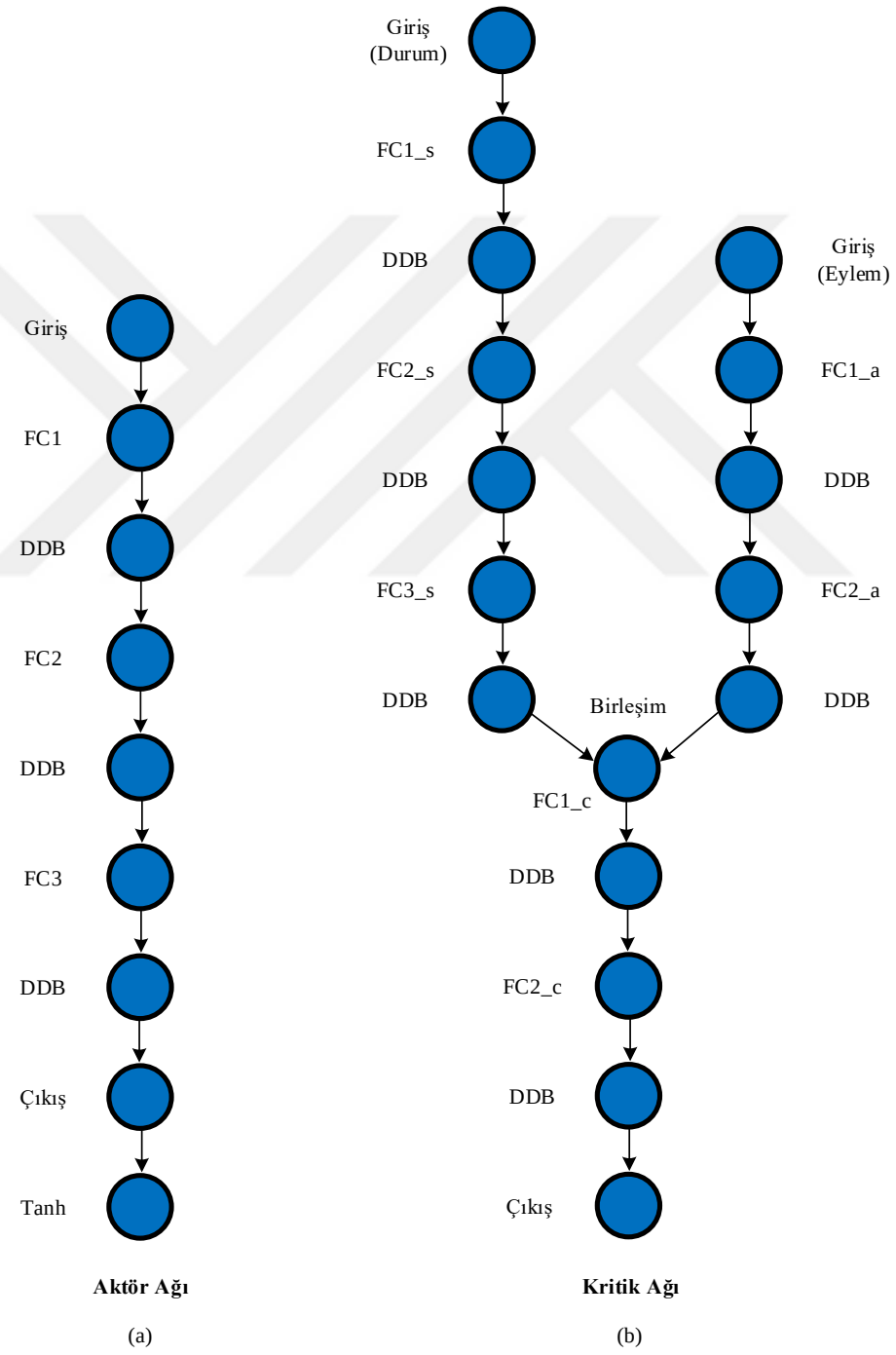
$$R_4 = \begin{cases} 1, & eğer\ dengede\ ise \\ -1, & aksi\ taktirde \end{cases} \quad (7.9)$$

R_5, SMN hesaplanmasındaki hatayı ifade eden ödülü temsil etmektedir. Robotun bozucu etkilerden dolayı ayaklarının yere temas etmediği zaman KDD'lerden bilgi alınamadığı için SMN hesaplanamamaktadır. Hesaplanamadığı durumlarda ceza, hesaplama ile ilgili bir sorun olmadığında ise hiçbir değer almayacak şekilde, (7.10)'da verildiği gibi tanımlanmıştır.

$$R_5 = \begin{cases} -1, & eğer\ hesaplama\ hatası\ var\ ise \\ 0, & aksi\ taktirde \end{cases} \quad (7.10)$$

Yukarıda verilen eşitlikler ile her adımda bağımsız olarak hesaplanan ödül fonksiyonlarına ek olarak robot düştüğünde ödül -1000, hedefe ulaştığında ise ödül 1000 değeri ilave edilmiştir. Robotun düşmesi için tam olarak düşme eylemi beklenmeden robotun yönelim açılarından herhangi biri $-50°$'den küçük veya $50°$'den büyük olduğu anda robot düştü olarak kabul edilmiştir.

Deneysel çalışmalar için oluşturulan DDPG'nin ağ yapıları Şekil 7.69'da verilmiştir.



Şekil 7.69. Kullanılan DDPG'nin ağ yapıları: (a) aktör ağı ve (b) kritik ağı

Şekil 7.69'da verilen DDPG'nin Aktör ve Kritik ağlarının katmanları, nöron sayıları ve aktivasyon fonksiyonları ise sırasıyla Tablo 7.14 ve Tablo 7.15'te verilmiştir.

Tablo 7.14. Aktör ağının katman yapısı

Katman ismi	Katman türü	Nöron sayısı	Aktivasyon türü
Giriş	Tam bağlı	8	-
FC1	Tam bağlı	128	DDB
FC2	Tam bağlı	128	DDB
FC3	Tam bağlı	128	DDB
Çıkış	Tam bağlı	1	Tanh

Tablo 7.15. Kritik ağının katman yapısı

Katman ismi	Katman türü	Nöron sayısı	Aktivasyon türü
Giriş 1 (Durum)	Tam bağlı	8	-
FC1_s	Tam bağlı	128	DDB
FC2_s	Tam bağlı	256	DDB
FC3_s	Tam bağlı	512	DDB
Giriş 2 (Eylem)	Tam bağlı	1	-
FC1_a	Tam bağlı	256	DDB
FC2_a	Tam bağlı	512	DDB
FC1_c	Tam bağlı	512	DDB
FC2_c	Tam bağlı	512	DDB
Çıkış	Tam bağlı	1	-

Tablo 7.16'da DDPG'nin eğitimi için seçilen hiperparametreler listelenmiştir.

Tablo 7.16. DDPG için seçilen hiperparametreler

Hiperparametre	Değer
İndirim faktörü (γ)	0.99
Yumuşak güncelleme faktörü (τ)	0.005
Aktör ağ öğrenme oranı (α)	10^{-4}
Kritik ağ öğrenme oranı (β)	10^{-4}

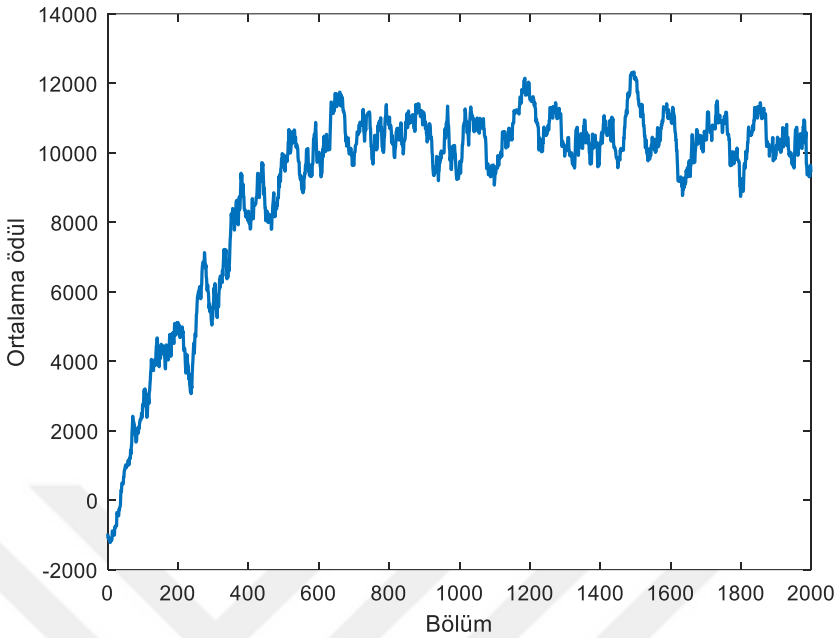
DPÖ, etkileşimli bir makine öğrenmesi yaklaşımı olduğu için statik bir veri kümesi olmadığından öğrenme işlemi çevrimiçi gerçekleşmektedir. Tekrar belleği, veri kümesi olarak kullanılmakta olup mevcut durum, eylem, ödül ve sonraki durum verilerinden oluşmaktadır. DDPG'deki YSA'lar, tekrar belleği kullanılarak eğitilmiştir. Tez çalışması kapsamında DDPG ile gerçekleştirilen deneysel çalışmalarda tekrar belleği boyutu 10^6, mini yığın boyutu ise 64 olarak seçilmiştir.

Daha önce bahsedildiği gibi PÖ algoritmalarındaki keşif özelliği, çözüm uzayında bulunan en iyi sonuçtan bağımsız daha iyi bir sonucun olup olmadığının incelenmesidir. DDPG algoritmasının orijinal makalesinde [68] keşif özelliği için kullanılan Ornstein-Uhlenbeck (O-U) gürültüsü kullanılmaktadır. Tez kapsamında DDPG ile gerçekleştirilen deneysel çalışmalarda keşif için gürültü üretiminde O-U işleminin kullanımı tercih edilmiş olup O-U keşif gürültü değeri 0.1, gürültü standart sapması ise 0.01 olarak seçilmiştir.

DDPG ile eğimli yüzeylerde dengeli bir şekilde yürüyebilmesi için hem yokuş yukarı hem de yokuş aşağı için gerçekleştirilen ağ eğitim işlemleri, Adam optimize edicisi kullanılarak 2 bin bölüm içermekte olup her bölüm robotun hedefe ulaşıncaya veya düşünceye kadar geçen süreyi kapsayan adım sayısından oluşmaktadır. DPÖ yapısı, bölüm içerisinde gerçekleştirilen her bir adımda robotun yürürken gövde yunuslama açısını değiştiren bir eylem alacak şekilde ayarlanmıştır. Böylece, robot hedefe ulaşıncaya kadar düşmeden ilerledikçe her zaman adımında (7.5) ile hesaplanan bir ödül almıştır. Ancak robot, yürüyüşün herhangi bir anında düştüğünde veya hedefe ulaştığında çevre yeniden başlatılarak diğer bölüme geçilmiştir. Robotun yürüyüş parametreleri ve seçilen eylem uzayları göz önünde bulundurulduğunda ters kinematik hesaplama hatasına sebep olacak bir durumla karşı karşıya kalınmayacağı için bu hata kontrolü, yeniden başlatma koşuluna eklenmemiştir.

Hem yokuş yukarı hem de yokuş aşağı yüzeylerde yürüme görevleri için DDPG'nin eğitim işlemleri başlamadan önce, Düello ÇDQA'nın eğitiminde olduğu gibi Webots simülatöründe robot eklemlerinin K_p oransal kazanç ve K_i integral kazanç katsayıları sırasıyla 30 ve 0.9 olarak alınmıştır. DDPG'nin eğitim işlemleri de Düello ÇDQA'nın eğitiminde olduğu gibi Webots simülatörü üzerinde aynı yazılım geliştirme araçları kullanılarak daha önce özellikleri bahsedilen masaüstü bilgisayarda gerçekleştirilmiştir.

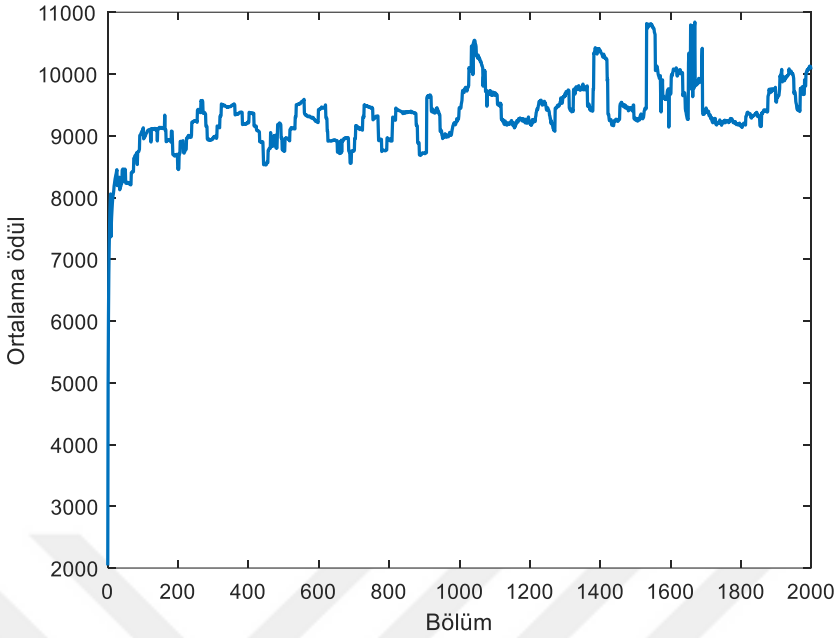
Öncelikle, Şekil 7.67'deki parkurda gösterilen A-B noktaları arasındaki yokuş yukarı yürüme görevi için DDPG kontrolörü kullanılarak önerilen çerçeve ile eğitim işlemleri gerçekleştirilmiştir. Eğitim esnasında her bölümde alınan ödüller kaydedilmiştir. Robot her hedefe ulaştığında, ortalama ödülün en yüksek olduğu bölümün sonunda ve gerçekleşen son bölümün Aktör, Kritik, hedef Aktör ve hedef Kritik ağlarının ağırlıkları saklanmıştır. 2000. bölüm ile eğitim tamamlandıktan sonra son 40 bölümün ortalaması alınarak elde edilen ortalama ödül grafiği, MATLAB ortamında Şekil 7.70'teki gibi çizdirilmiştir.



Şekil 7.70. Son 40 bölümün ortalaması alınarak yokuş yukarı çıkma görevi için ortalama ödül grafiği

Şekil 7.70'deki DDPG'nin ortalama ödül grafiği incelendiğinde yaklaşık 900. bölüme kadar üstel şekilde arttığı görülmüştür. Eğitim sürecinde, ağların öğrenme işlemi gerçekleşerek robotun hedefe ulaştığı ağırlıklar ayrı ayrı kaydedilmiştir. Ortalama ödülün en yüksek değerine 1497. bölümde ulaşılmış ve bu bölümdeki ortalama ödülün 12325 olduğu görülmüştür.

Daha sonra, Şekil 7.67'deki parkurda gösterilen C-D noktaları arasındaki yokuş aşağı yürüme görevi için DDPG kontrolörü kullanılarak önerilen çerçeve ile eğitim işlemleri gerçekleştirilmiştir. Eğitim esnasında yokuş yukarı çıkma görevinde olduğu gibi her bölümde alınan ödüller kaydedilmiştir. Robot her hedefe ulaştığında, yine aynı şekilde ortalama ödülün en yüksek olduğu bölümün sonunda ve gerçekleşen son bölümün Aktör, Kritik, hedef Aktör ve hedef Kritik ağlarının ağırlıkları saklanmıştır. 2000. bölüm ile eğitim tamamlandıktan sonra son 40 bölümün ortalaması alınarak elde edilen ortalama ödül grafiği, MATLAB ortamında Şekil 7.71'deki gibi çizdirilmiştir.



Şekil 7.71. Son 40 bölümün ortalaması alınarak yokuş aşağı inme görevi için ortalama ödül grafiği

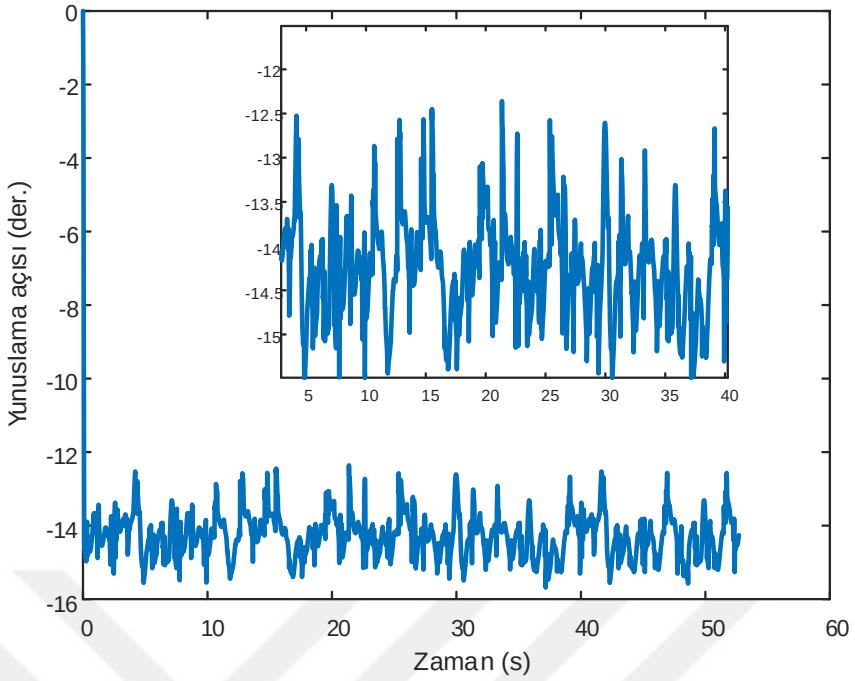
Şekil 7.71'deki DDPG'nin ortalama ödül grafiği incelendiğinde hızlı bir şekilde yaklaşık 300. bölüme kadar üstel şekilde arttığı görülmüştür. Yaklaşık olarak 1000. bölümden sonra da artan bir öğrenme süreci gerçekleşmiştir. Eğitim sürecinde, ağların öğrenme işlemi gerçekleşerek robotun hedefe ulaştığı ağırlıklar ayrı ayrı kaydedilmiştir. Ortalama ödülün en yüksek değerine 1670. bölümde ulaşılmış ve bu bölümdeki ortalama ödülün 10840.87 olduğu görülmüştür.

DDPG kontrolör kullanılarak gerçekleştirilen ağ eğitimleri ile eğitim işlemleri tamamlandıktan sonra elde edilen modelleri içeren kontrolörler ile test işlemlerine geçilerek deneysel çalışmalar gerçekleştirilmiştir.

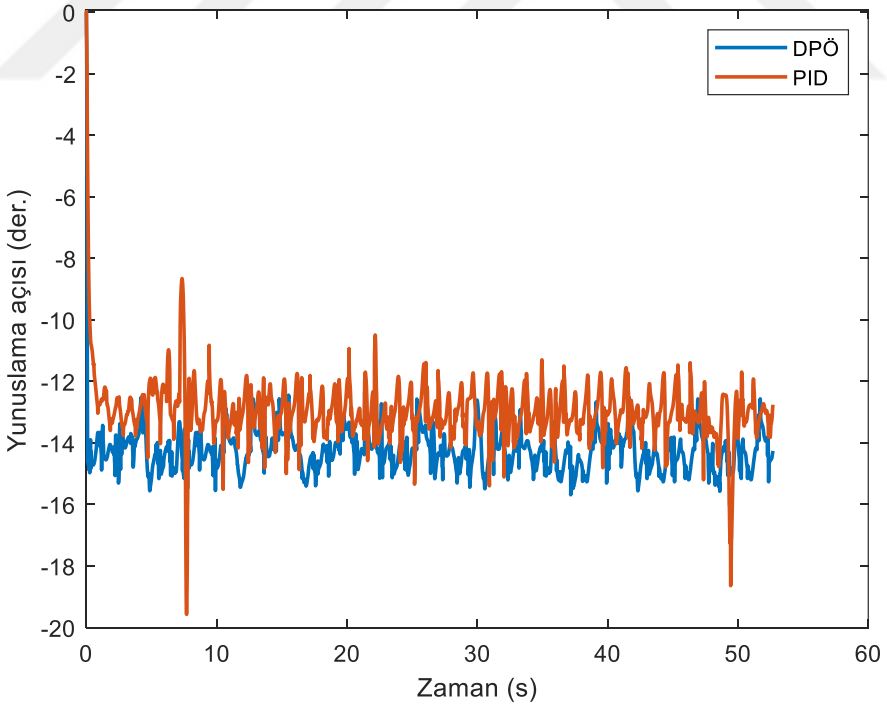
DDPG ile Test

Hedefe ulaşıp ortalama ödülün en yüksek olduğu bölümün sonunda kaydedilen ağ ağırlıkları yüklenip Webots simülatöründe test işlemleri gerçekleştirilmiştir.

İlk olarak, yokuş yukarı yürüme görevi için test işlemleri gerçekleştirilmiş olup robotun vücut yunuslama açılarının Kalman filtresi ile hesaplanmasına ve PID ile DPÖ kontrolör sonuçlarının karşılaştırılmasına sırasıyla Şekil 7.72 ve Şekil 7.73'te yer verilmiştir.



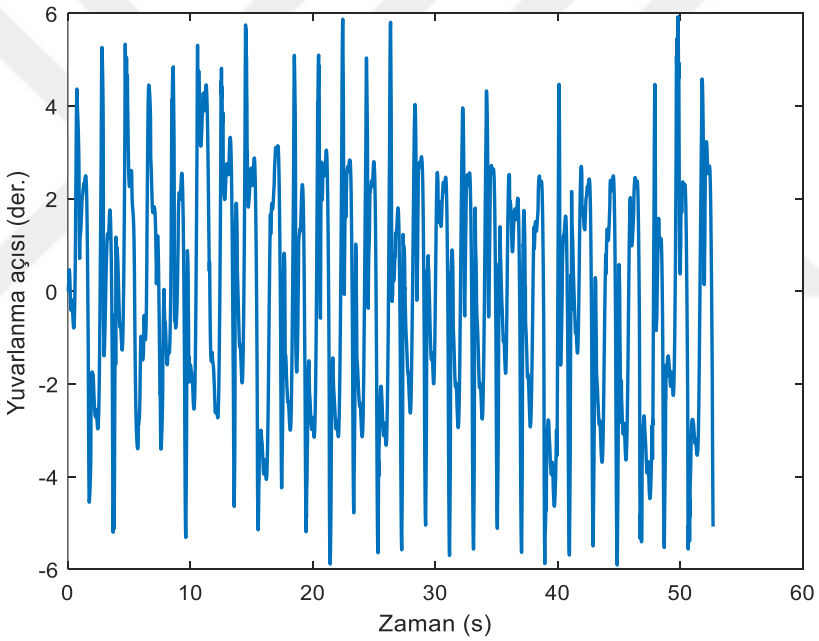
Şekil 7.72. Yokuş yukarı yürüme görevi için Kalman filtresi ile hesaplanan yunuslama açısı



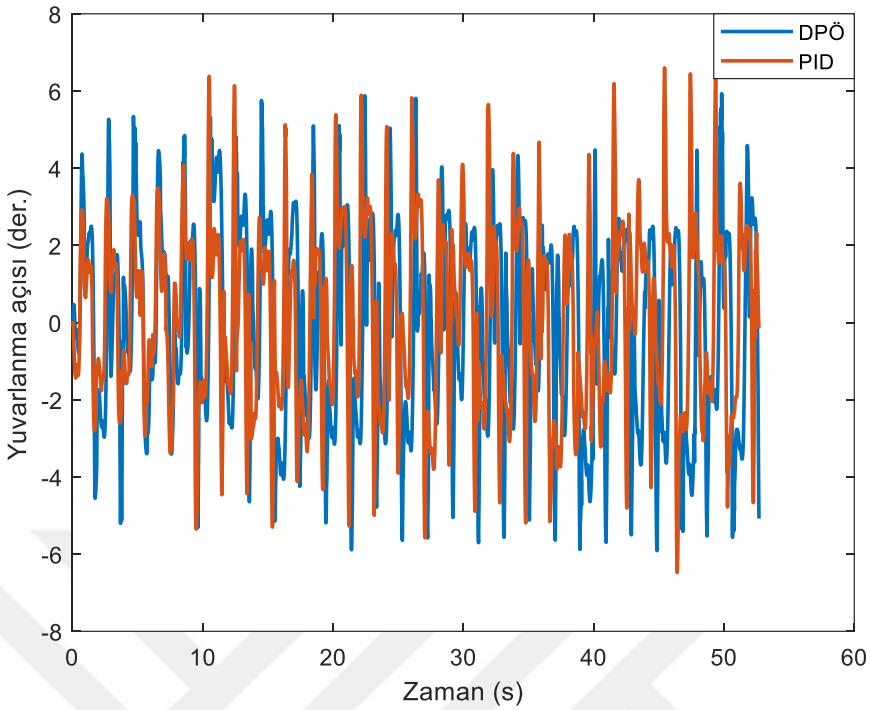
Şekil 7.73. Yokuş yukarı yürümede kontrolörlere göre yunuslama açılarının karşılaştırılması

DDPG için oluşturulan ödül fonksiyonu, robot vücut yunuslama açısının $-13°$'ye yaklaştıkça ödül alacak şekilde oluşturulmuştu. Şekil 7.72 incelendiğinde robotun yunuslama açısının PID kontrolörün referans değeri olan $-13°$'ye yakın değerlerde salınımda olduğu görülmüştür. Şekil 7.73'teki sonuçlara göre önerilen DPÖ kontrolör ile robotun yunuslama açısının PID kontrolöre kıyasla daha az gürültüye sahip olduğu görülmüştür. Ayrıca, yunuslama açılarındaki değişimlerden de görüldüğü gibi DPÖ kontrolör ile robot, yokuş yukarı zeminin başlangıcı ve bitişi anında zemine daha yumuşak bir geçiş sağlamıştır.

Robotun yokuş yukarı çıkarken vücut yuvarlanma açılarının Kalman filtresi ile hesaplanmasına ve PID ile DPÖ kontrolörün karşılaştırılmasına sırasıyla Şekil 7.74 ve Şekil 7.75'te yer verilmiştir.



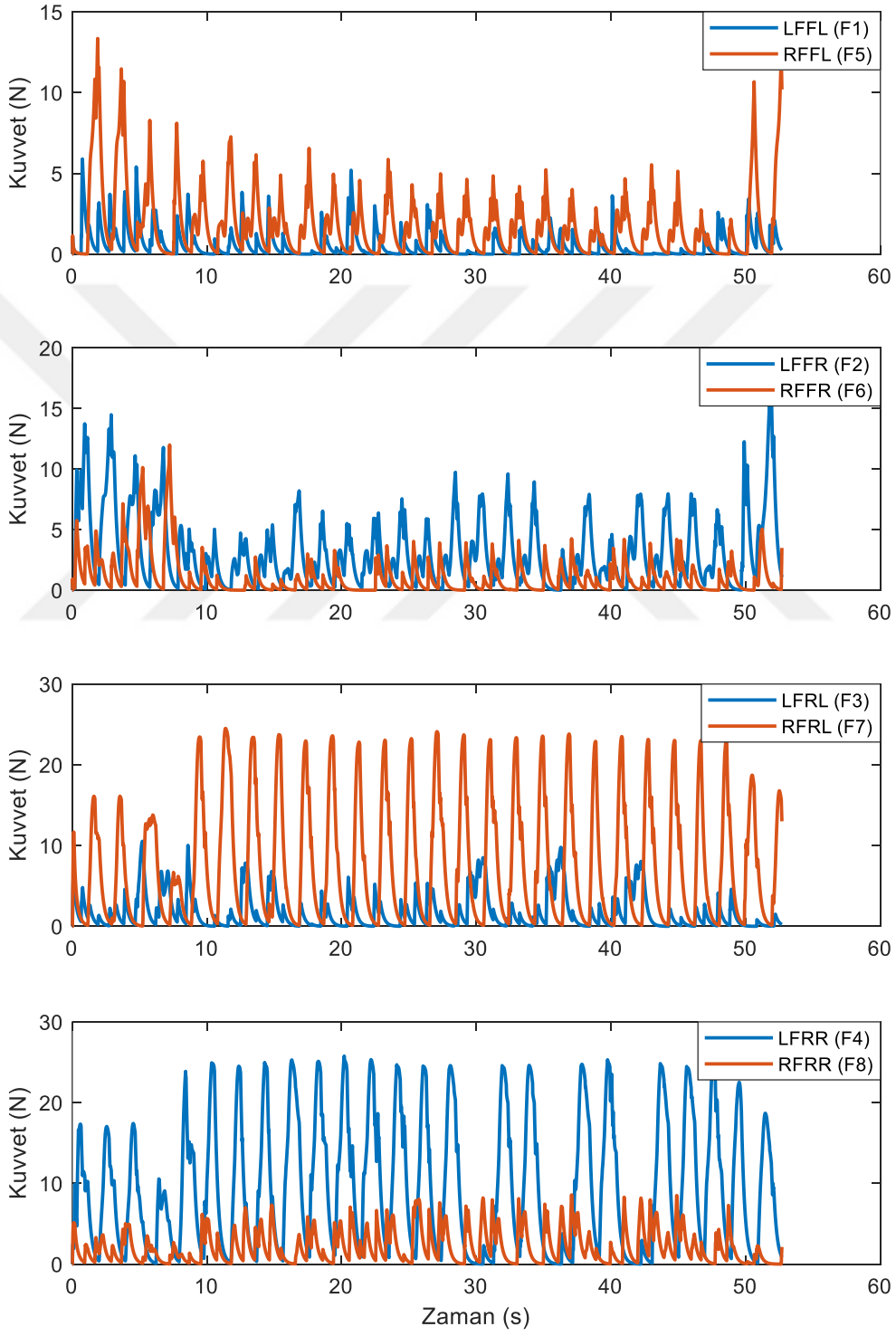
Şekil 7.74. Yokuş yukarı yürüme görevi için Kalman filtresi ile hesaplanan yuvarlanma açısı



Şekil 7.75. Yokuş yukarı yürümede kontrolörlere göre yuvarlanma açılarının karşılaştırılması

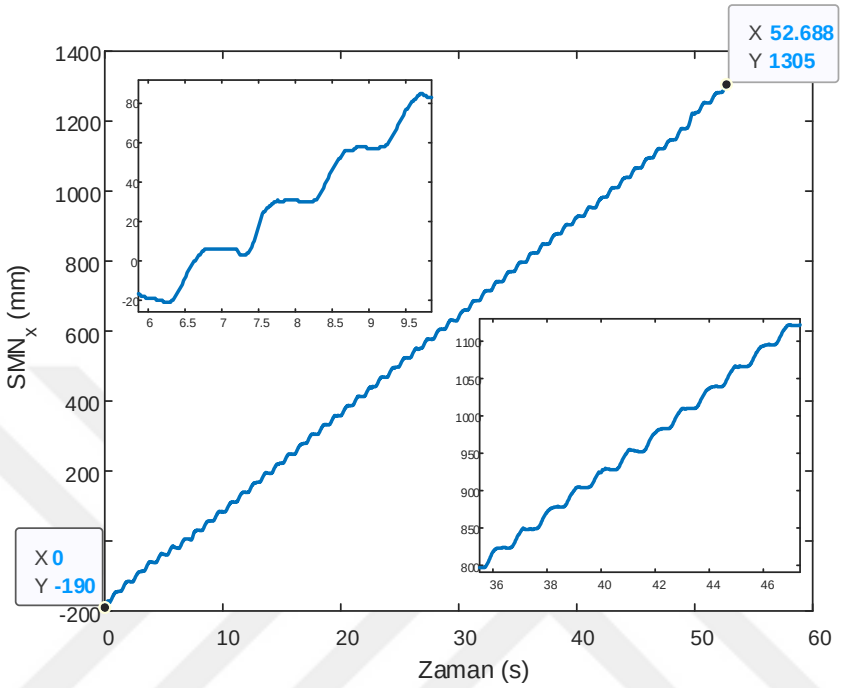
Şekil 7.74'teki yuvarlanma açıları gözlemlendiğinde, robotun yanal sallanmasının 6°'yi geçmediği görülmüştür. Şekil 7.75'teki sonuçlara göre PID kontrolör ile ilk 10 s içinde yuvarlanma açısının daha az salınıma sahip olduğu görülmesine rağmen daha sonraki zamanlarda genel olarak önerilen DPÖ kontrolör ile genliği daha düşük salınımlar elde edilmiştir.

Robot yokuş yukarı çıkarken KDD'lerden okunan değerlerin bir boyutlu Kalman filtresinden geçtikten sonra sol ve sağ ayağın temas kuvvetleri, MATLAB ortamında Şekil 7.76'da verildiği gibi görselleştirilmiştir.

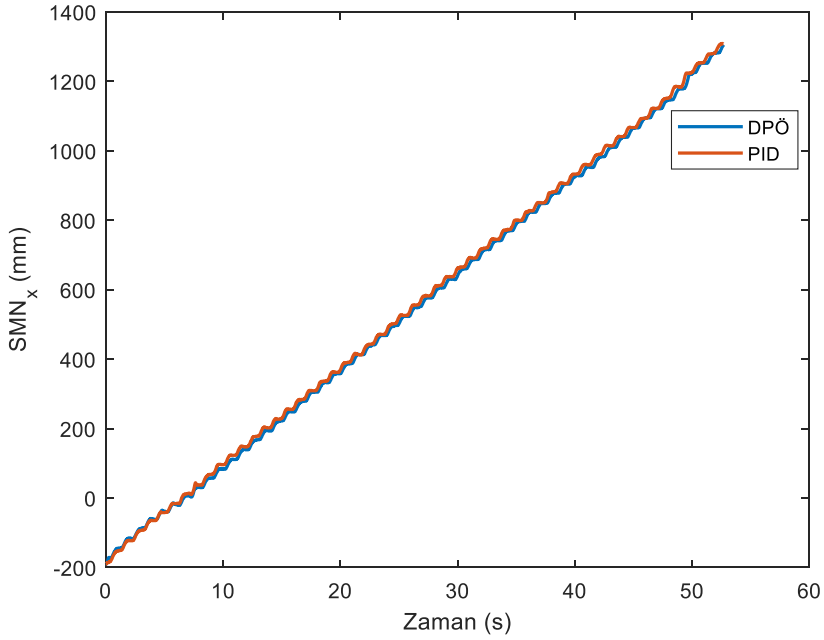


Şekil 7.76. Yokuş yukarı yürüme görevi için KDD'lerden okunan sol ve sağ ayağın temas kuvvetleri

SMN_x koordinatlarının Şekil 7.76'daki kuvvet değerleriyle hesaplanmasına ve PID ile DPÖ kontrolörün karşılaştırılmasına sırasıyla Şekil 7.77 ve Şekil 7.78'de yer verilmiştir.



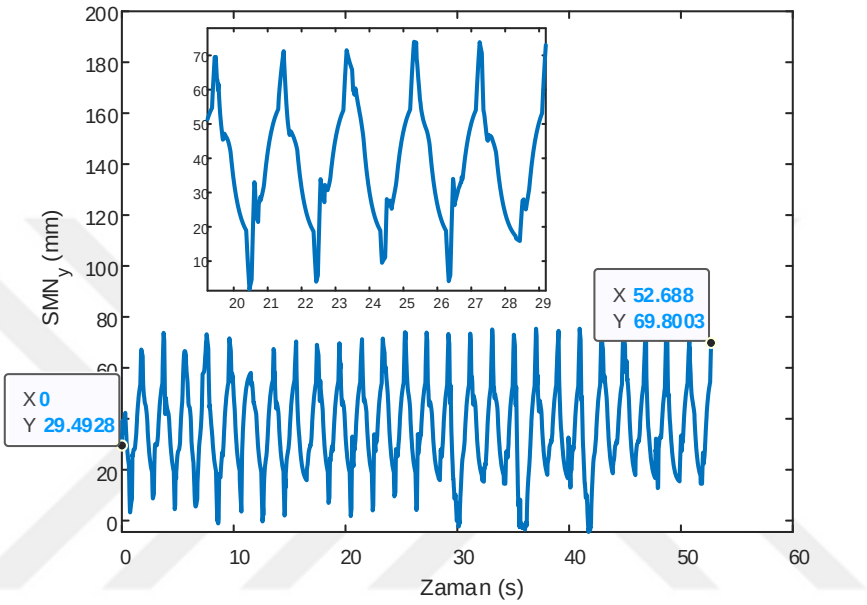
Şekil 7.77. Yokuş yukarı yürüme görevinde hesaplanan SMN_x değerleri



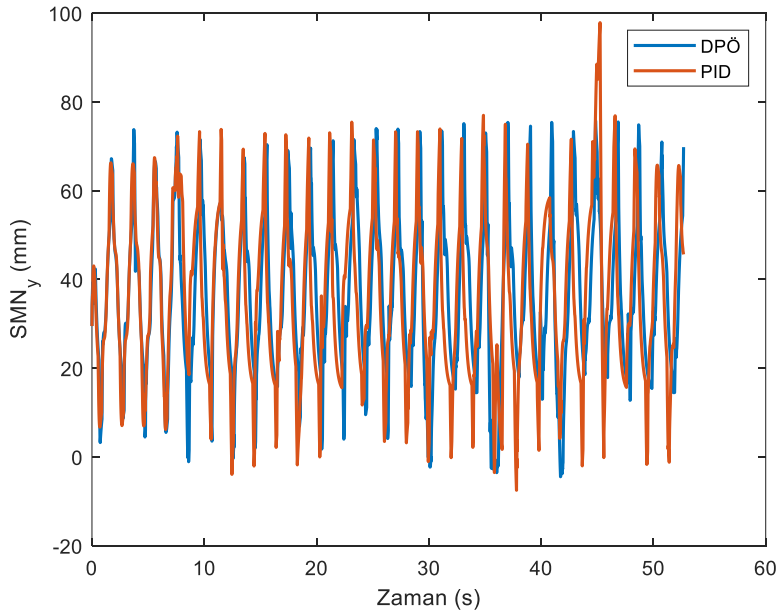
Şekil 7.78. Yokuş yukarı yürümede kontrolörlere göre SMN_x değerlerinin karşılaştırılması

Şekil 7.77 incelendiğinde SMN_x koordinatlarının düzgün bir şekilde salınım yaparak arttığı görülmüştür. Şekil 7.78'deki karşılaştırmalı sonuçlar incelendiğinde ise her iki kontrolörün sonuçlarının birbirine oldukça yakın olduğu ve hemen hemen örtüştüğü görülmüştür.

SMN_y koordinatlarının Şekil 7.76'daki kuvvet değerleriyle hesaplanmasına ve PID ile DPÖ kontrolörün karşılaştırılmasına ise sırasıyla Şekil 7.79 ve Şekil 7.80'de yer verilmiştir.



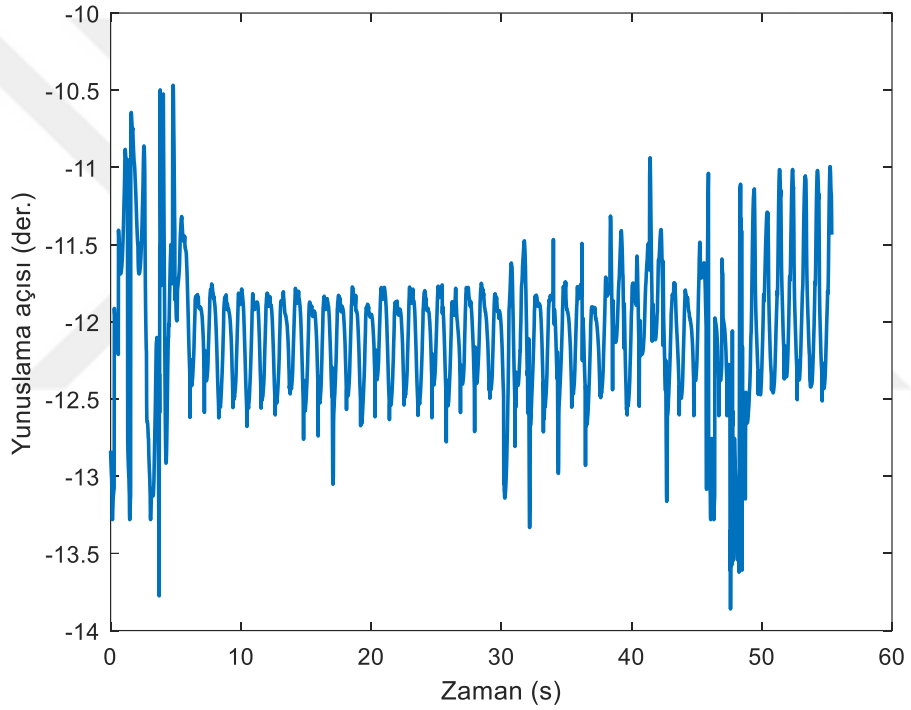
Şekil 7.79. Yokuş yukarı yürüme görevinde hesaplanan SMN_y değerleri



Şekil 7.80. Yokuş yukarı yürümede kontrolörlere göre SMN_y değerlerinin karşılaştırılması

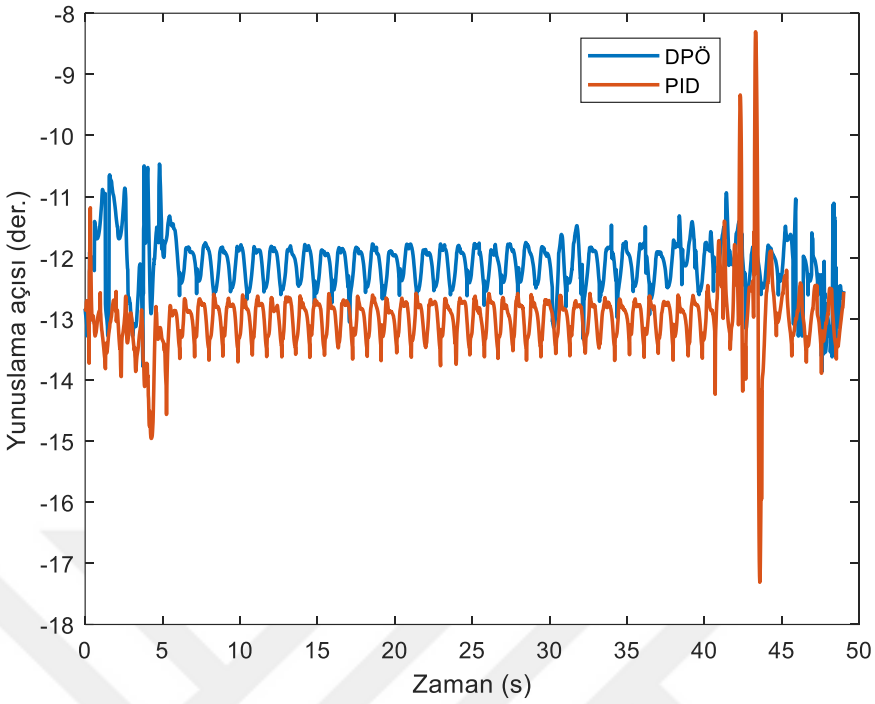
Şekil 7.79 incelendiğinde SMN_y koordinatlarının yokuş yukarı yürürken çok az anlık gürültülere sahip olsa da belirli bir periyotta düzgün bir salınım yaptığı görülmüştür. Şekil 7.80'deki karşılaştırmalı sonuçlar incelendiğinde ise her iki kontrolörün sonuçları birbirine yakın olduğu, ancak genel olarak PID kontrolör ile daha fazla gürültüye sahip salınımların elde edildiği görülmüştür.

Son olarak, yokuş aşağı yürüme görevi için test işlemleri gerçekleştirilmiş olup robotun vücut yunuslama açılarının Kalman filtresi ile hesaplanmasına ve PID ile DPÖ kontrolör sonuçlarının karşılaştırılmasına sırasıyla Şekil 7.81 ve Şekil 7.82'de yer verilmiştir.



Şekil 7.81. Yokuş aşağı yürüme görevi için Kalman filtresi ile hesaplanan yunuslama açısı

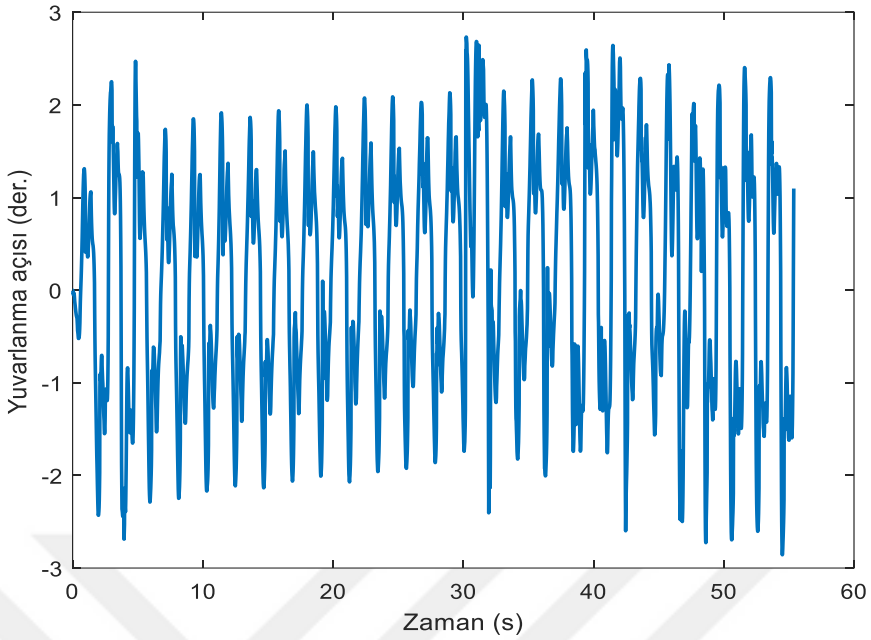
Şekil 7.81 incelendiğinde robotun yunuslama açısının PID kontrolörün referans değeri olan $-13°$'den $1°$ daha dik duruşa sahip değerlerde salınım yaptığı görülmüştür. Yokuş aşağı parkur görevi için gerçekleştirilen DDPG eğitiminde çevre, robotun başlangıç yunuslama açısı $-13°$ olacak şekilde ayarlanmıştır.



Şekil 7.82. Yokuş aşağı yürümede kontrolörlere göre yunuslama açılarının karşılaştırılması

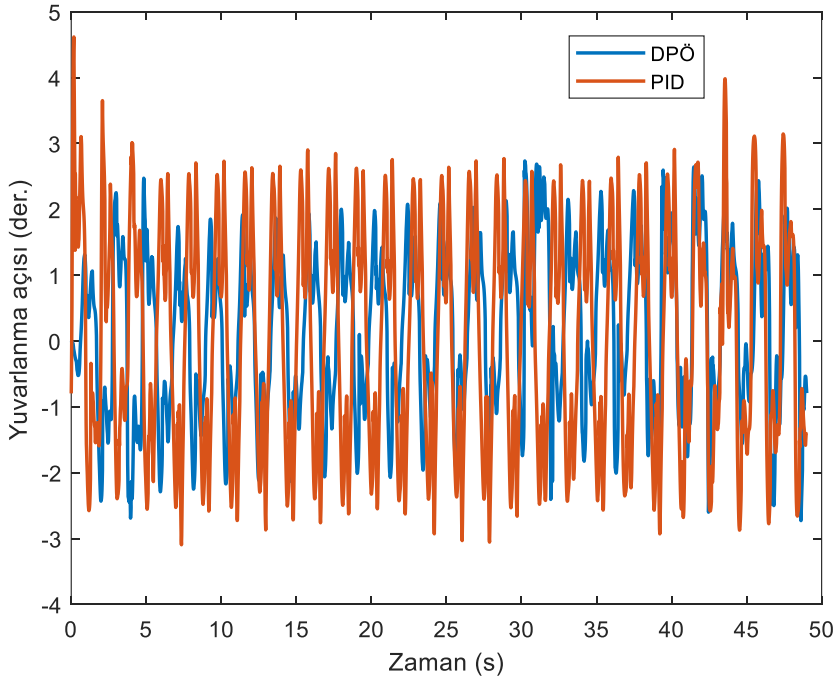
Şekil 7.82'deki sonuçlara göre önerilen DPÖ kontrolör ile robotun yunuslama açısının PID kontrolöre kıyasla daha az gürültüye sahip olduğu görülmüştür. Ayrıca, yunuslama açılarındaki değişimlerden de görüldüğü gibi DPÖ kontrolör ile robot, yokuş aşağı zeminin başlangıcı ve bitişi anında zemine daha yumuşak bir geçiş sağlamıştır. Ancak, referans değere yakınlık bakımından PID kontrolöre göre daha başarısız olduğu görülmüştür.

Robotun yokuş aşağı inerken vücut yuvarlanma açılarının Kalman filtresi ile hesaplanmasına ve PID ile DPÖ kontrolörün karşılaştırılmasına sırasıyla Şekil 7.83 ve Şekil 7.84'te yer verilmiştir.



Şekil 7.83. Yokuş aşağı yürüme görevi için Kalman filtresi ile hesaplanan yuvarlanma açısı

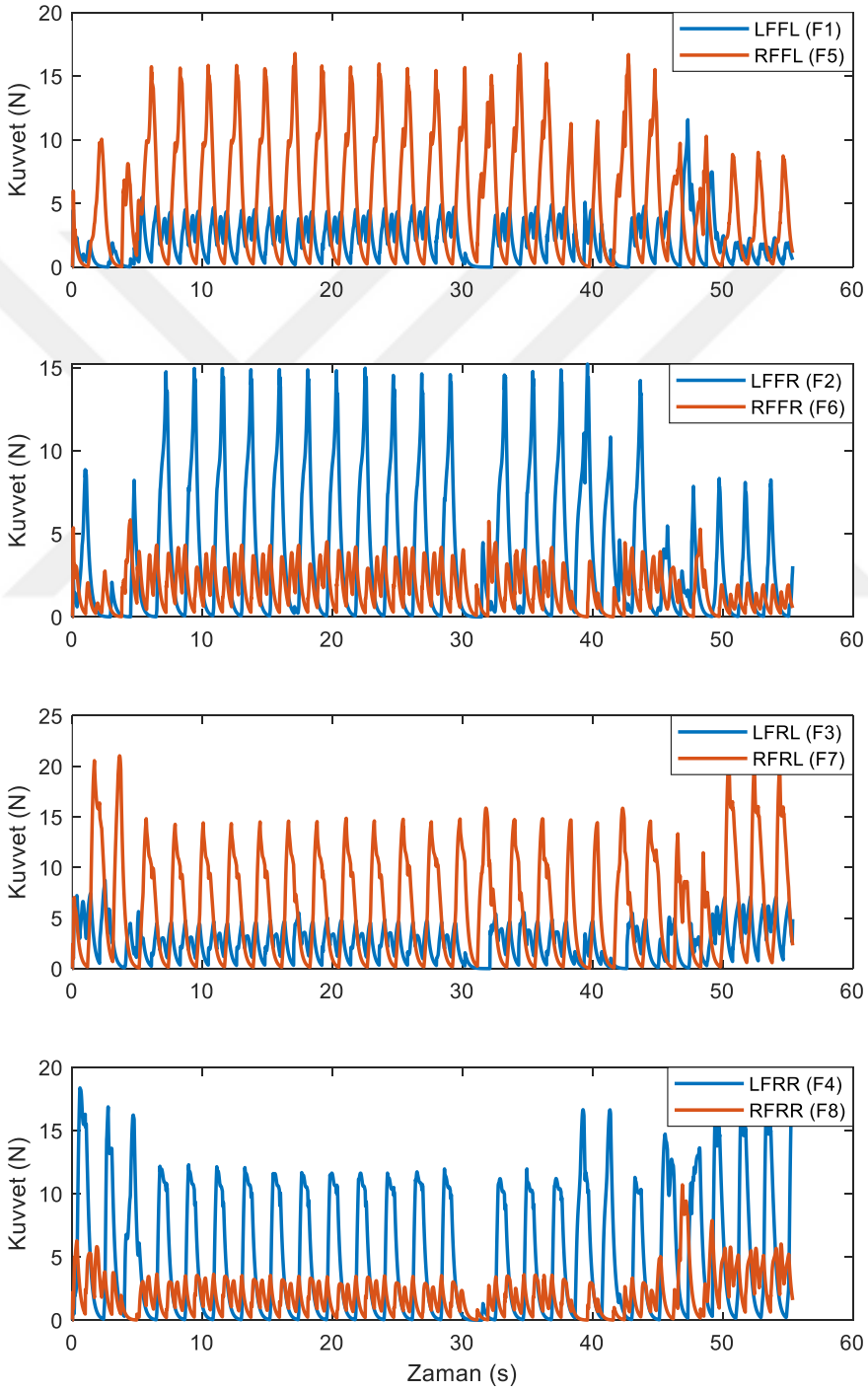
Şekil 7.83'teki yuvarlanma açıları gözlemlendiğinde ise robotun yanal sallanmasının 3°'yi geçmediği ve bazı gürültülerin dışında salınımın belli bir periyoda sahip olduğu görülmüştür.



Şekil 7.84. Yokuş aşağı yürümede kontrolörlere göre yunuslama açılarının karşılaştırılması

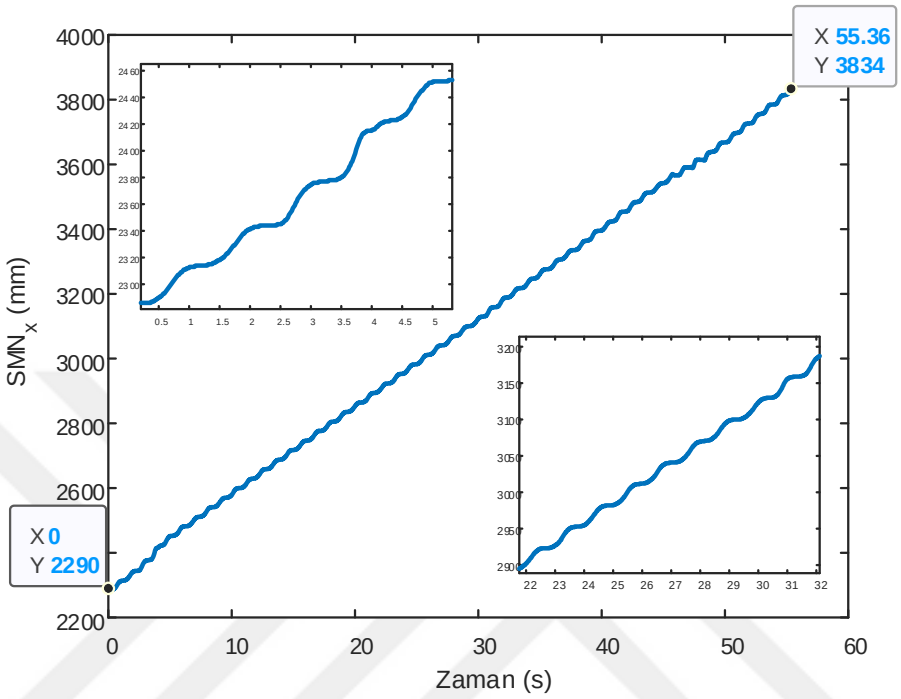
Şekil 7.84'teki karşılaştırmalı sonuçlar incelendiğinde DPÖ kontrolör ile elde edilen salınımların PID kontrolöre göre daha az genliklere sahip olduğu görülmüştür.

Robot yokuş aşağı inerken KDD'lerden okunan değerlerin bir boyutlu Kalman filtresinden geçtikten sonra sol ve sağ ayağın temas kuvvetleri, MATLAB ortamında Şekil 7.85'te verildiği gibi görselleştirilmiştir.

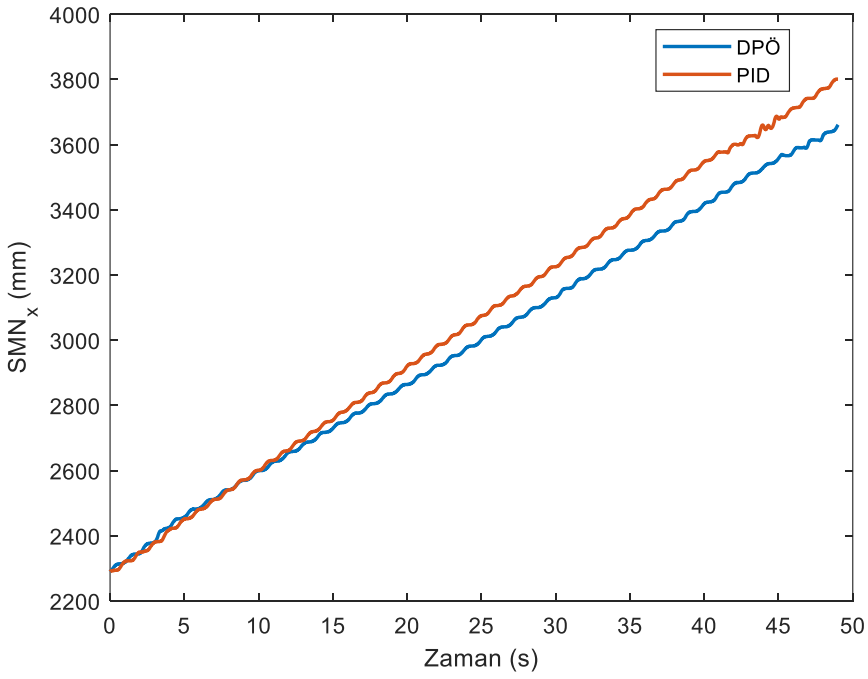


Şekil 7.85. Yokuş aşağı yürüme görevi için KDD'lerden okunan sol ve sağ ayağın temas kuvvetleri

SMN_x koordinatlarının Şekil 7.85'teki kuvvet değerleriyle hesaplanmasına ve PID ile DPÖ kontrolörün karşılaştırılmasına sırasıyla Şekil 7.86 ve Şekil 7.87'de yer verilmiştir.



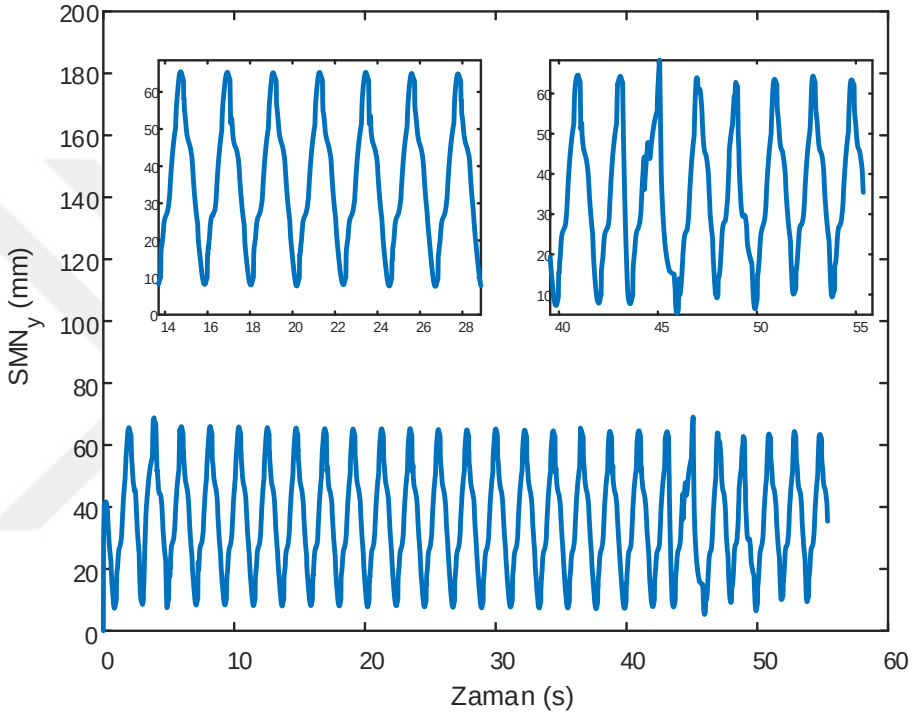
Şekil 7.86. Yokuş aşağı yürüme görevinde hesaplanan SMN_x değerleri



Şekil 7.87. Yokuş aşağı yürümede kontrolörlere göre SMN_x değerlerinin karşılaştırılması

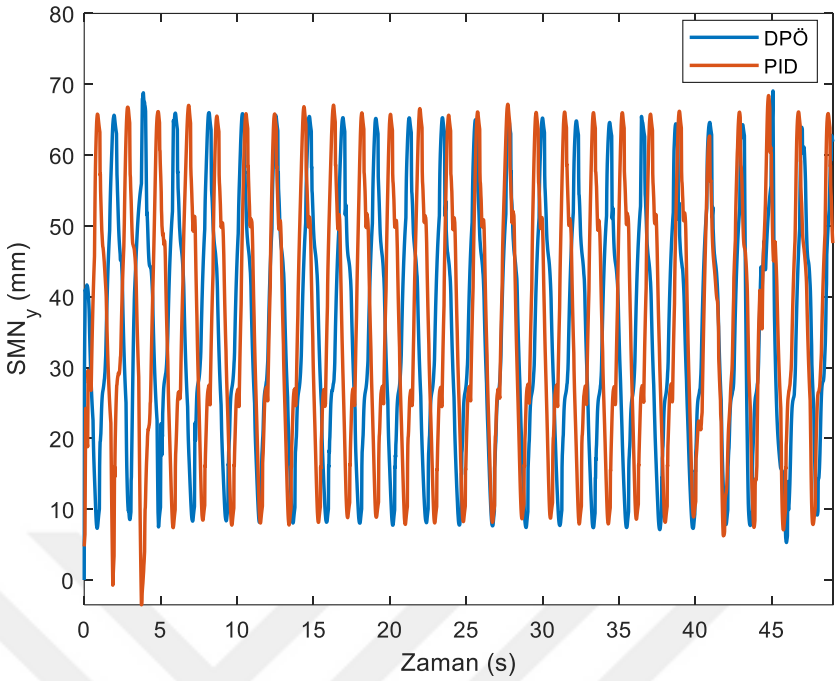
Şekil 7.86 incelendiğinde SMN_x koordinatlarının düzgün bir şekilde salınım yaparak arttığı görülmüştür. Şekil 7.87'deki karşılaştırmalı sonuçlar incelendiğinde ise PID kontrolör ile SMN'nin x eksenindeki koordinatı DPÖ kontrolörünkine göre yaklaşık 0.16 m daha ileriye gittiği görülmüştür.

SMN_y koordinatlarının Şekil 7.85'teki kuvvet değerleriyle hesaplanmasına ve PID ile DPÖ kontrolörün karşılaştırılmasına ise sırasıyla Şekil 7.88 ve Şekil 7.89'da yer verilmiştir.



Şekil 7.88. Yokuş aşağı yürüme görevinde hesaplanan SMN_y değerleri

Şekil 7.88 incelendiğinde SMN_y koordinatlarının yokuş aşağı yürürken çok az anlık gürültülere sahip olsa da belirli bir periyotta düzgün bir salınım yaptığı görülmüştür.



Şekil 7.89. Yokuş aşağı yürümede kontrolörlere göre SMN_y değerlerinin karşılaştırılması

Şekil 7.89'daki karşılaştırmalı sonuçlar incelendiğinde ise her iki kontrolörün salınım genliklerinin birbirine yakın olduğu görülmüş, ancak bazı zamanlarda salınımlarda faz kaymaları gerçekleşmiştir. Ayrıca, genel olarak PID kontrolör ile daha fazla gürültüye sahip salınımların elde edildiği görülmüştür.

Eğimli yüzeylerde yürüme ile ilgili gerçekleştirilen deneysel çalışmalarda, kontrolör tipine göre yönelim açılarında meydana gelen salınımların en büyük ve en küçük değerleri Tablo 7.17'de verilmiştir.

Tablo 7.17. Yüzey ve kontrolör tipine göre robotun yönelim açılarının en küçük ve en büyük değerleri

Eğimli yüzey tipi	**Kontrolör tipi**	**Yönelim açısı tipi**	**Değer (der.)**	
			min.	**maks.**
Yokuş yukarı	DPÖ (DDPG)	Yunuslama	−15.68	−12.36
	PID		−19.57	−8.66
	DPÖ (DDPG)	Yuvarlanma	−5.91	5.93
	PID		−6.48	6.60
Yokuş aşağı	DPÖ (DDPG)	Yunuslama	−13.86	−10.47
	PID		−17.31	−8.30
	DPÖ (DDPG)	Yuvarlanma	−2.73	2.74
	PID		−3.09	4.62

Robotun yokuş yukarı yürüme görevine başlangıç anında vücut yunuslama açısının Şekil 7.73'te gösterildiği gibi 0°'den istenen referans değere ulaşması için geçen süredeki açı değişimi, Tablo 7.17'deki değerlerde ihmal edilmiştir.

Önerilen DPÖ kontrolör ile robotun yürüyüşü sırasında yunuslama açısının PID kontrolöre kıyasla daha az gürültüye sahip olduğu görülmüştür. Yokuş yukarı yüzeyde yunuslama açısı, DDPG ile $-15.68°$ ile $-12.36°$ arasında değişirken PID ile $-19.57°$ ile $-8.66°$ arasında değişmiştir. Yokuş aşağı yüzeyde ise yunuslama açısı, DDPG ile $-13.86°$ ile $-10.47°$ arasında değişirken PID ile $-17.31°$ ile $-8.30°$ arasında değişmiştir. Benzer şekilde, DPÖ kontrolör ile robotun yürüyüşü sırasında yuvarlanma açısının da PID kontrolöre kıyasla daha az gürültüye sahip olduğu görülmüştür. Yokuş yukarı yüzeyde yuvarlanma açısı, DDPG ile $-5.91°$ ile $5.93°$ arasında değişirken PID ile $-6.48°$ ile $6.60°$ arasında değişmiştir. Yokuş aşağı yüzeyde ise yuvarlanma açısı, DDPG ile $-2.73°$ ile $2.74°$ arasında değişirken PID ile $-3.09°$ ile $4.62°$ arasında değişmiştir. Yönelim açılarındaki değişim miktarları göz önüne alındığında hem yokuş yukarı hem de yokuş aşağı eğimli yüzeylerde robot, önerilen DDPG algoritması tabanlı DPÖ kontrolör ile PID kontrolörden daha kararlı ve dengeli bir yürüyüş gerçekleştirmiştir.

Ayrıca, PID kazanç katsayılarının deneme yanılma ile ayarlanmasının oldukça zahmetli olduğu ve bu katsayıların yürüme parametrelerinde veya robotun başlangıç konumunda yapılacak herhangi bir değişiklik karşısında yeniden ayarlanması gerektiği görülmüştür. Ancak, DDPG ile robot çevre ile etkileşim kurarak bir öğrenme süreci gerçekleştirdiğinden yapılacak değişikler karşısında ağın eğitim koşullarına göre uyum sağlayabileceği anlaşılmıştır. Deneysel sonuçlar, DPÖ ile oluşturulan kontrolörün PID kontrolöre göre daha kullanışlı olduğunu ve daha kararlı bir yürüme sağladığını göstermiştir.

8. SONUÇLAR

Bu tez çalışmasında, DPÖ algoritmaları kullanılarak insansı robotların düz ve eğimli yüzeylerde kararlı bir şekilde yürüyebilmesi için etkili çerçeveler önerilmiştir. İlk olarak, insansı bir robotun kararlı yürümesi için gürbüz bir algoritma geliştirilerek bir çerçeve önerilmiştir. Tez kapsamında kullanılan Robotis-OP2 insansı robotunun yürüme yörüngesi, sikloid eğrileri kullanılarak oluşturulmuştur. Robotun istenilen yörüngelere ulaşabilmesi için bacak eklem açılarını elde etmek için bacak kinematik analizleri gerçekleştirilmiştir. Robot üzerindeki çoklu sensörlerden elde edilen veriler anlamlandırılmıştır. Önerilen çerçeve, geleneksel bir yörünge üreteci kontrolör ve DPÖ yapısından oluşmaktadır. Bu çerçeve sayesinde yürüyüş şablonu üretici ile oluşturulan yörüngelerin optimum yürüyüş parametreleri DPÖ algoritmalarından olan Düello ÇDQA'nın eğitimi ile elde edilmiştir. Düello ÇDQA'nın eğitimi Webots simülatör ortamında gerçekleştirilmiştir. Optimum parametrelere yaklaşık olarak 21000. bölümde yakınsanmıştır. Bu çalışma, Düello ÇDQA kullanılarak insansı bir robotun yürüyüşü ile ilgili yapılan literatürdeki ilk çalışmadır. Optimum yürüyüş parametreleri belirlendikten sonra robot gövdesinin sagital düzlemde dengesini koruyabilmesi için kalça stratejisi ve PID kontrolör kullanılarak bir dengeleme sistemi oluşturulmuştur.

Düz yüzeylerde yürüme ile ilgili deneysel çalışmalar, önerilen çerçeve ve Robotis-OP2'nin kendi yürüme algoritması ile Robotis-OP2 insansı robotu üzerinde hem Webots simülatör ortamında hem de oluşturulan kontrolörlerin gerçek robota aktarılmasıyla gerçek ortamda gerçekleştirilmiştir. Sagital düzlemde robot gövdesinin $-13°$ eğimle yürümesi istendiğinde simülasyon ortamında robotun gövde yunuslama açısı $-13.44°$ ile $-12.99°$ arasında bir sallanma ile yürümüştür. Gövde yuvarlanma açısında ise $-1.96°$ ile $2.33°$ arasında bir sallanma meydana gelmiştir. Robotun kendi yürüyüş algoritması ile simülasyon ortamında robotun gövde yunuslama açısı $-13.46°$ ile $-7.13°$, gövde yuvarlanma açısı ise $-10.03°$ ile $12.23°$ arasında sallanmalar ile yürüyüş gerçekleşmiştir. Robotun kararlı yürümesinin ölçütlerinden olan x ve y eksenindeki SMN değerlerinin de oldukça düzgün bir salınıma sahip olduğu gözlemlenmiştir. Elde edilen optimum yürüyüş parametreleri ile robotun saniyede 60 mm'lik bir adıma sahip olduğu bir yürüyüş için dört yürüme periyodunda SMN_x, 474 mm ilerlemiştir. Ayrıca, SMN_y değerinin de başlangıç koordinatına göre sadece 12 mm'lik bir yer değiştirmeye sahip olduğu görülmüştür. Gerçek ortamda, önerilen çerçeve ile robotun yunuslama açısının $-16.07°$ ile $-5.31°$ ve yuvarlanma açısının $-3.60°$ ile $5.42°$ arasında sallanma ile yürürken kendi yürüyüş algoritması ile robotun yunuslama açısının $-12.87°$ ile $2.40°$ ve yuvarlanma açısının $-3.69°$ ile $8.12°$ arasında sallanma ile yürümüştür. Deneysel sonuçlar, Robotis-OP2 insansı robotunun önerilen çerçeve ile hem simülasyon hem de gerçek ortamda düz bir zeminde Robotis-OP2'nin kendi yürüme algoritmasına göre daha kararlı bir şekilde yürüyebildiğini göstermiştir.

Bu tez çalışmasında daha sonra, robotun dengesini koruyarak eğimli yüzeylerde kararlı bir şekilde yürüyebilmesi için PID kontrolör ve DPÖ kontrolörden oluşan iki ayrı yürüyüş dengeleme çerçevesi önerilmiştir. DPÖ kontrolör olarak DDPG algoritması kullanılmıştır. PID kontrolör ile gerçekleştirilen deneysel çalışmalarda robotun duruşu gerçek zamanlı olarak ayarlanarak düz zeminde yürüyormuş gibi gövde yunuslama açısının istenilen referans değerde olması sağlanmıştır. DDPG kullanılarak gerçekleştirilen ağ eğitimi ile robotun eğimli yüzeylerde dengeli yürüyüşünün sağlanabilmesi için sürekli eylem uzayından robotun gövde yunuslama açısının sıralı hareket dizisi öğrenilmiştir.

Yokuş yukarı ve yokuş aşağı yüzeylerde yürüme görevleri için iki ayrı DDPG tabanlı kontrolör eğitilmiştir. Yokuş yukarı çıkma görevinde DDPG'nin eğitimi sırasında ortalama ödülün yaklaşık 900. bölüme kadar üstel şekilde artmıştır. Ortalama ödülün en yüksek değerine 1497. bölümde ulaşılmış ve bu bölümdeki ortalama ödülün 12325 olduğu görülmüştür. Yokuş aşağı inme görevinde ise ortalama ödülün hızlı bir şekilde yaklaşık 300. bölüme kadar üstel şekilde artmıştır. Yaklaşık olarak 1000. bölümden sonra da artan bir öğrenme süreci gerçekleşmiştir. Ortalama ödülün en yüksek değerine 1670. bölümde ulaşılmış ve bu bölümdeki ortalama ödülün 10840.87 olduğu görülmüştür. Eğitim işlemleri tamamlandıktan sonra elde edilen ağ modellerini içeren kontrolörler ve PID kontrolör ile Webots ortamında deneysel çalışmalar gerçekleştirilmiştir.

Önerilen DPÖ kontrolör ile robotun yürüyüşü sırasında yunuslama açısının PID kontrolöre kıyasla daha az gürültüye sahip olduğu görülmüştür. Yokuş yukarı yüzeyde yunuslama açısı, DDPG ile $-15.68°$ ile $-12.36°$ arasında değişirken PID ile $-19.57°$ ile $-8.66°$ arasında değişmiştir. Yokuş aşağı yüzeyde ise yunuslama açısı, DDPG ile $-13.86°$ ile $-10.47°$ arasında değişirken PID ile $-17.31°$ ile $-8.30°$ arasında değişmiştir. Benzer şekilde, DPÖ kontrolör ile robotun yürüyüşü sırasında yuvarlanma açısının da PID kontrolöre kıyasla daha az gürültüye sahip olduğu görülmüştür. Yokuş yukarı yüzeyde yuvarlanma açısı, DDPG ile $-5.91°$ ile $5.93°$ arasında değişirken PID ile $-6.48°$ ile $6.60°$ arasında değişmiştir. Yokuş aşağı yüzeyde ise yuvarlanma açısı, DDPG ile $-2.73°$ ile $2.74°$ arasında değişirken PID ile $-3.09°$ ile $4.62°$ arasında değişmiştir. Yunuslama açılarındaki değişimler gözlemlendiğinde DPÖ kontrolör ile robot, yokuş yukarı zeminin başlangıcı ve bitişi anında zemine daha yumuşak bir geçiş sağlamıştır. Ayrıca, PID kazanç katsayılarının deneme yanılma ile ayarlanmasının oldukça zahmetli olduğu ve bu katsayıların yürüme parametrelerinde veya robotun başlangıç konumunda yapılacak herhangi bir değişiklik karşısında yeniden ayarlanması gerektiği görülmüştür. Ancak, DDPG ile robot çevre ile etkileşim kurarak bir öğrenme süreci gerçekleştirdiğinden yapılacak değişikler karşısında ağın eğitim koşullarına göre uyum sağlayabileceği anlaşılmıştır. Deneysel sonuçlar, DPÖ ile oluşturulan kontrolörün PID kontrolöre göre daha kullanışlı olduğunu ve daha kararlı bir yürüme sağladığını göstermiştir.

Ayrıca bu tez kapsamında, hem kararlı bir yürüme için yürüyüş parametrelerinin optimizasyonu hem de eğimli yüzeylerde yürüme için önerilen çerçevelerde kullanılan DPÖ yapılarının oluşturulma süreçlerinde ve ağ eğitimlerinde bazı zorluklar ve kısıtlamalar meydana gelmiştir. Ağ eğitimleri kapsamındaki deneysel çalışmalarda, masaüstü bilgisayarın GPU'sunun CUDA eklentisi aracılığıyla paralel hesaplama gücünden yararlanılmıştır. Ancak, oluşturulan ağ yapılarındaki katman ve nöron sayılarının arttırılarak daha iyi bir ağ eğitiminin gerçekleştirilebilme denemeleri GPU'nun donanımsal eksikliğinden dolayı başarısızlıkla sonuçlanmıştır. Daha derin ağ yapılarında gerçekleştirilen denemelerde, GPU'nun RAM'i önemli rol oynamış olup robotun hareketlerinde gecikmelere neden olmuştur. Bununla birlikte, DPÖ yapılarında ödül fonksiyonunun ayarlanması işlemi de oldukça önemli olduğundan birçok denemenin ardından nihai ödül fonksiyonuna ulaşılmıştır.

ÖNERİLER VE GELECEK ÇALIŞMALAR

Bu tezde, düz bir zeminde kararlı yürüme ile ilgili gerçekleştirilen çalışmalarda robotun ideal yürüme hızı 60 mm/s olarak elde edilmişti. Yürüyüş şablonu üreteci ile oluşturulan yürüme yörüngelerinde robotun yürüyüşü esnasında çift destek fazının süresi azaltılarak daha hızlı bir yürüme şablonu oluşturulabilir. Yürüyüş parametrelerinin optimizasyonunda eylem uzayını azaltmak için sabit kabul edilen parametreler de DPÖ'nün eylem uzayına dahil edilerek daha çeşitli yürüyüş sonuçları elde edilebilir.

DPÖ kontrolör ile gerçekleştirilen eğimli yüzeylerde yürüme çalışmalarında robotun vücut yunuslama açısının istenen referans değere daha yakın olabilmesi için ödül fonksiyonu iyileştirilebilir.

Ayrıca, bu tez çalışmasına dayanarak gelecekte birçok çalışma yapılabilir:

- Farklı DPÖ algoritmalarının kullanımı Webots simülatöründe kullanılmaya uyumlu hale getirilerek DDPG'ye göre performansları incelenebilir.
- Kalça dengeleme stratejisine ek olarak ayak bileği stratejisi de eklenerek eğimli yüzeylerde robotun daha dengeli yürümesi sağlanabilir.
- Robotun bilinmeyen arazilerde başarılı bir şekilde yürüyebilmesine yönelik algoritmalar geliştirilebilir.
- Robotun daha az enerji tüketerek daha uzun süre yürümesini sağlayacak çalışmalar yapılabilir.
- İnsanlar da dahil olmak üzere doğal iki ayaklı yürüyüş mekanizmalarını daha iyi anlamak için daha ayrıntılı araştırmalar yapılabilir
- Hem sağlıklı hem de engelli kişilerde alt ekstremite dış iskeletine yönelik uygulamalar için bir yürüyüş modeli geliştirilebilir.

KAYNAKLAR

[1] Pu, L., Moyle, W., Jones, C., Todorovic, M. (2019). The effectiveness of social robots for older adults: A systematic review and meta-analysis of randomized controlled studies, *Gerontologist*, *59*(1), 37–51. https://doi.org/10.1093/geront/gny046.

[2] Zhuang, H., Gao, H., Deng, Z., Ding, L., Liu, Z. (2014). A review of heavy-duty legged robots, *Science China Technological Sciences*, *57*, 298–314. https://doi.org/10.1007/s11431-013-5443-7.

[3] Chung, R.L., Hsueh, Y., Chen, S.L., Abu, P.A.R. (2022). Efficient and accurate CORDIC pipelined architecture chip design based on binomial approximation for biped robot, *Electronics*, *11*(11), 1–14. https://doi.org/10.3390/electronics11111701.

[4] Rostro-Gonzalez, H., Lauterio-Cruz, J.P., Pottiez, O. (2020). Modelling neural dynamics with optics: A new approach to simulate spiking neurons through an asynchronous laser, *Electronics*, *9*(11), 1–11. https://doi.org/10.3390/electronics9111853.

[5] Pratt, J., Carff, J., Drakunov, S., Goswami, A. (2006). Capture point: A step toward humanoid push recovery, In *6th IEEE-RAS International Conference Humanoid Robots*, 200–207. https://doi.org/10.1109/ICHR.2006.321385.

[6] Lee, S.H., Goswami, A. (2007). Reaction mass pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots, In Proceedings *IEEE International Conference on Robotics and Automation (ICRA)*, 4667–4672. https://doi.org/10.1109/ROBOT.2007.364198.

[7] Li, X., Li, Y., Cui, X. (2016). Kinematic analysis and gait planning for a DARwIn-OP humanoid robot, In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 1442–1447. https://doi.org/10.1109/ROBIO.2016.7866530.

[8] Kajita, S., Morisawa, M., Miura, K., Nakaoka, S., Harada, K., Kaneko, K., Kanehiro, F., Yokoi, K. (2010). Biped walking stabilization based on linear inverted pendulum tracking, In *IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4489–4496. https://doi.org/10.1109/IROS.2010.5651082.

[9] Vukobratović, M., Borovac, B. (2004). Zero-Moment Point — Thirty five years of its life, *International Journal of Humanoid Robotics*, *1*, 157–173. https://doi.org/10.1142/S0219843604000083.

[10] Guan, K., Yamamoto, K., Nakamura, Y. (2019). Virtual-mass-ellipsoid inverted pendulum model and its applications to 3d bipedal locomotion on uneven terrains, In *IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1401–1406. https://doi.org/10.1109/IROS40897.2019.8968284.

[11] Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point, In *IEEE International Conference on Robotics and Automation (ICRA), 2*, 1620–1626. https://doi.org/10.1109/robot.2003.1241826.

[12] Shimmyo, S., Sato, T., Ohnishi, K. (2013). Biped walking pattern generation by using preview control based on three-mass model, *IEEE Transactions on Industrial Electronics*, *60*(11), 5137–5147. https://doi.org/10.1109/TIE.2012.2221111.

[13] Faraji, S., Ijspeert, A.J. (2017). 3LP: A linear 3D-walking model including torso and swing dynamics, *International Journal of Robotics Research (IJRR)*, *36*(4), 436–455. https://doi.org/10.1177/0278364917708248.

[14] Griffin, R.J., Wiedebach, G., Bertrand, S., Leonessa, A., Pratt, J. (2017). Walking stabilization using step timing and location adjustment on the humanoid robot, Atlas, In *IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS)*, 667–673. https://doi.org/10.1109/IROS.2017.8202223.

[15] Kasaei, M., Lau, N., Pereira, A. (2019). A robust biped locomotion based on linear-quadratic-gaussian controller and divergent component of motion, In *IEEE-RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1429–1434. https://doi.org/10.1109/IROS40897.2019.8967778.

[16] Kasaei, M.M., Lau, N., Pereira, A. (2019). A model-based biped walking controller based on divergent component of motion, In *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 1–6. https://doi.org/10.1109/ICARSC.2019.8733608.

[17] Yamaguchi, J., Soga, E., Inoue, S., Takanishi, A. (1999). Development of a bipedal humanoid robot - control method of whole body cooperative dynamic biped walking, In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, *1*, 368–374. https://doi.org/10.1109/robot.1999.770006.

[18] Yu, J., Zhang, S., Wang, A., Li, W., Song, L. (2021). Musculoskeletal modeling and humanoid control of robots based on human gait data, *PeerJ Computer Science*, *7*, 1–19. https://doi.org/10.7717/peerj-cs.657.

[19] Ishihara, K., Itoh, T.D., Morimoto, J. (2020). Full-body optimal control toward versatile and agile behaviors in a humanoid robot, *IEEE Robotics and Automation Letters*, *5*(1), 119–126. https://doi.org/10.1109/LRA.2019.2947001.

[20] Nassour, J., Hénaff, P., Benouezdou, F., Cheng, G. (2014). Multi-layered multi-pattern cpg for adaptive locomotion of humanoid robots, *Biological Cybernetics*, *108*, 291–303. https://doi.org/10.1007/s00422-014-0592-8.

[21] Lee, J., Seo, K. (2013). Generation of walking trajectory of humanoid robot using cpg, *Journal of Korean Institute of Intelligent Systems*, *23*(4), 360–365. https://doi.org/10.5391/jkiis.2013.23.4.360.

[22] Liu, C., Wang, D., Member, S., Chen, Q. (2013). Central pattern generator inspired control for adaptive walking of biped robots, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *43*(5), 1–10. https://doi.org/10.1109/TSMC.2012.2235426.

[23] Yu, J., Tan, M., Chen, J., Zhang, J. (2014). A survey on CPG-inspired control models and system implementation, *IEEE Transactions on Neural Networks and Learning Systems*, *25*(3), 441–456. https://doi.org/10.1109/TNNLS.2013.2280596.

[24] Guertin, P.A. (2009). The mammalian central pattern generator for locomotion, *Brain Research Reviews*, *62*(1), 45–56. https://doi.org/10.1016/j.brainresrev.2009.08.002.

[25] Zhong, G., Shevtsova, N.A., Rybak, I.A., Harris-Warrick, R.M. (2012). Neuronal activity in the isolated mouse spinal cord during spontaneous deletions in fictive locomotion: Insights into locomotor central pattern generator organization, *Journal of Physiology*, *590*(19), 4735–4759. https://doi.org/10.1113/jphysiol.2012.240895.

[26] Menelaou, E., McLean, D.L. (2019). Hierarchical control of locomotion by distinct types of spinal V2a interneurons in zebrafish, *Nature Communications*, *10*(1), 1–12. https://doi.org/10.1038/s41467-019-12240-3.

[27] Endo, G., Morimoto, J., Matsubara, T., Nakanishi, J., Cheng, G. (2008). Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot, *International Journal of Robotics Research (IJRR)*, *27*(2), 213–228. https://doi.org/10.1177/0278364907084980.

[28] García, J., Shafie, D. (2020). Teaching a humanoid robot to walk faster through safe reinforcement learning, *Engineering Applications of Artificial Intelligence*, *88*, 103360. https://doi.org/10.1016/j.engappai.2019.103360.

[29] Kasaei, M., Lau, N., Pereira, A. (2019). A fast and stable omnidirectional walking engine for the nao humanoid robot, In *RoboCup 2019: Robot World Cup XXIII 23*, Springer International Publishing, 99–111. https://doi.org/10.1007/978-3-030-35699-6_8.

[30] MacAlpine, P., Barrett, S., Urieli, D., Vu, V., Stone, P. (2012). Design and optimization of an omnidirectional humanoid walk: A winning approach at the RoboCup 2011 3D simulation competition, In *Proceedings of the AAAI Conference on Artificial* Intelligence, *26*, 1047–1053. https://doi.org/10.1609/aaai.v26i1.8317.

[31] Or, J. (2010). A hybrid CPG-ZMP control system for stable walking of a simulated flexible spine humanoid robot, *Neural Networks*, *23*(3), 452–460. https://doi.org/10.1016/j.neunet.2009.11.003.

[32] He, B., Wang, Z., Shen, R., Hu, S. (2014). Real-time walking pattern generation for a biped robot with hybrid CPG-ZMP algorithm, *International Journal of Advanced Robotic Systems*, *11*(10), 160. https://doi.org/10.5772/58845.

[33] Kasaei, S.M., Simões, D., Lau, N., Pereira, A. (2018). A Hybrid zmp-cpg based walk engine for biped robots, *Advences in Intelligent Systems and Computing*, *694*, 743–755. https://doi.org/10.1007/978-3-319-70836-2_61.

[34] Carpentier, J., Tonneau, S., Naveau, M., Stasse, O., Mansard, N. (2016). A versatile and efficient pattern generator for generalized legged locomotion, In *IEEE International Conference on Robotics and Automation (ICRA)*, 3555–3561. https://doi.org/10.1109/ICRA.2016.7487538.

[35] Song, K.T., Hsieh, C.H. (2014). CPG-based control design for bipedal walking on unknown slope surfaces, In *IEEE International Conference on Robotics and Automation (ICRA)*, 5109–5114. https://doi.org/10.1109/ICRA.2014.6907608.

[36] Baothman, F.A. (2021). A machine learning approach for improving the movement of humanoid NAO's gaits, *Wireless Communications and Mobile Computing*, *2021*, 1–14. https://doi.org/10.1155/2021/1496364.

[37] Massah B, A., Zamani, A., Salehinia, Y., Aliyari Sh, M., Teshnehlab, M. (2013). A hybrid controller based on cpg and zmp for biped locomotion, *Journal of Mechanical Science and Technology*, *27*, 3473–3486. https://doi.org/10.1007/s12206-013-0871-7.

[38] Michel, O. (2004). WebotsTM: Professional mobile robot simulation, 39–42. http://arxiv.org/abs/cs/0412052.

[39] Kim, J.Y., Park, I.W., Oh, J.H. (2007). Walking control algorithm of biped humanoid robot on uneven and inclined floor, *Journal of Intelligent and Robotic Systems*, *48*, 457–484. https://doi.org/10.1007/s10846-006-9107-8.

[40] Morisawa, M., Kajita, S., Kanehiro, F., Kaneko, K., Miura, K., Yokoi, K. (2012). Balance control based on Capture Point error compensation for biped walking on uneven terrain, In *12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, 734–740. https://doi.org/10.1109/HUMANOIDS.2012.6651601.

[41] Yi, J., Zhu, Q., Xiong, R., Wu, J. (2016). Walking algorithm of humanoid robot on uneven terrain with terrain estimation, *International Journal of Advanced Robotic Systems*, *13*(1), 35. https://doi.org/10.5772/62245.

[42] Yi, S.J., Zhang, B.T., Lee, D.D. (2010). Online learning of uneven terrain for humanoid bipedal walking, In *Proceedings of the AAAI Conference on Artificial Intelligence*, *24*(1), 1639–1644. https://doi.org/10.1609/aaai.v24i1.7729.

[43] Smaldone, F.M., Scianca, N., Modugno, V., Lanari, L., Oriolo, G. (2019). Gait generation using intrinsically stable mpc in the presence of persistent disturbances, In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, 651–656. https://doi.org/10.1109/Humanoids43949.2019.9035068.

[44] Gong, Y., Hartley, R., Da, X., Hereid, A., Harib, O., Huang, J.K., Grizzle, J. (2019). Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway, In *IEEE American Control Conference (ACC)*, 4559–4566. https://doi.org/10.23919/acc.2019.8814833.

[45] Sutton, R.S., Barto, A.G. (2018). Reinforcement learning: An introduction. *A Bradford Book*.

[46] Song, D.R., Yang, C., McGreavy, C., Li, Z. (2018). Recurrent deterministic policy gradient method for bipedal locomotion on rough terrain challenge, In *IEEE 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 311–318. https://doi.org/10.1109/ICARCV.2018.8581309.

[47] Bin Peng, X., Berseth, G., Van De Panne, M. (2015). Dynamic terrain traversal skills using reinforcement learning, *ACM Transactions on Graphics (TOG)*, *34*(4), 1–11. https://doi.org/10.1145/2766910.

[48] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W. (2016). OpenAI gym, 1–4. *arXiv preprint arXiv:1606.01540*.

[49] Heess, N., Dhruva, T.B., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Ali Eslami, S.M., Riedmiller, M., Silver, D. (2017). Emergence of locomotion behaviours in rich environments, *arXiv preprint arXiv:1707.02286*.

[50] Sharma, R., Singh, I., Bharadwaj, D., Prateek, M. (2019). Incorporating forgetting mechanism in q-learning algorithm for locomotion of bipedal walking robot, *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, *8*(7), 1782–1787.

[51] Phaniteja, S., Dewangan, P., Guhan, P., Sarkar, A., Krishna, K.M. (2018). A deep reinforcement learning approach for dynamically stable inverse kinematics of humanoid robots, In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 1818–1823. https://doi.org/10.1109/ROBIO.2017.8324682.

[52] Wang, Z., Wei, W., Xie, A., Zhang, Y., Wu, J., Zhu, Q. (2022). Hybrid bipedal locomotion based on reinforcement learning and heuristics, *Micromachines*, *13*(10), 1–14. https://doi.org/10.3390/mi13101688.

[53] Gil, C.R., Calvo, H., Sossa, H. (2019). Learning an efficient gait cycle of a biped robot based on reinforcement learning and artificial neural networks, *Applied Sciences*, *9*(3), 502. https://doi.org/10.3390/app9030502.

[54] Liu, C., Ning, J., Chen, Q. (2018). Dynamic walking control of humanoid robots combining linear inverted pendulum mode with parameter optimization, *International Journal of Advanced Robotic Systems*, *15*(1), 1–15. https://doi.org/10.1177/1729881417749672.

[55] Kakade, S. (2001). A natural policy gradient, *Advances in Neural Information Processing Systems*, *14*.

[56] Lin, J.L., Hwang, K.S., Jiang, W.C., Chen, Y.J. (2016). Gait balance and acceleration of a biped robot based on q-Learning, *IEEE Access*, *4*, 2439–2449. https://doi.org/10.1109/ACCESS.2016.2570255.

[57] Silva, I.J., Perico, D.H., Homem, T.P.D., Vilão, C.O., Tonidandel, F., Bianchi, R.A.C. (2016). Humanoid robot gait on sloping floors using reinforcement learning, *Communications in Computer and Information Science*, *619*, 228–246. https://doi.org/10.1007/978-3-319-47247-8_14.

[58] Silva, I.J., Perico, D.H., Costa, A.H., Bianchi, R.A. (2017). Using reinforcement learning to optimize gait generation, *XIII Simpósio Brasileiro de Automaçao Inteligente*, 288–294.

[59] Bin Peng, X., Abbeel, P., Levine, S., Van De Panne, M. (2018). DeepMimic: Example-guided deep reinforcement learning of physics-based character skills, *ACM Transactions On Graphics (TOG)*, *37*(4), 1–14. https://doi.org/10.1145/3197517.3201311.

[60] Kwon, T., Lee, Y., Van De Panne, M. (2020). Fast and flexible multilegged locomotion using learned centroidal dynamics, *ACM Transactions On Graphics (TOG)*, *39*(4), 1–46. https://doi.org/10.1145/3386569.3392432.

[61] Xie, Z., Clary, P., Dao, J., Morais, P., Hurst, J., van de Panne, M. (2019). Iterative reinforcement learning based design of dynamic locomotion skills for cassie. *arXiv preprint arXiv:1903.09537*.

[62] Xie, Z., Clary, P., Dao, J., Morais, P., Hurst, J., van de Panne, M. (2019). Learning locomotion skills for cassie: Iterative design and sim-to-real, In *Conference on Robot Learning*, 317–329.

[63] Duan, H., Dao, J., Green, K., Apgar, T., Fern, A., Hurst, J. (2021). Learning task space actions for bipedal locomotion, In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 1276–1282. https://doi.org/10.1109/ICRA48506.2021.9561705.

[64] Özalp, U. (2021). *Bipedal robot walking by reinforcement learning in partially observed environment*, Master's thesis, Middle East Technical University.

[65] Fujimoto, S., Van Hoof, H., Meger, D. (2018). Addressing function approximation error in actor-critic methods, In *International Conference on Machine Learning (ICML)*, 1587–1596.

[66] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, In *International Conference on Machine Learning (ICML)*, 1861–1870.

[67] Rastogi, D. (2017). *Deep reinforcement learning for bipedal robots*, Master's thesis, Delft University of Technology.

[68] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

[69] Kumar, A., Paul, N., Omkar, S.N. (2018). Bipedal walking robot using deep deterministic policy gradient. *arXiv preprint arXiv:1807.05924*.

[70] Heess, N., Hunt, J.J., Lillicrap, T.P., Silver, D. (2015). Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*.

[71] Kasaei, M., Abreu, M., Lau, N., Pereira, A., Reis, L.P. (2021). Robust biped locomotion using deep reinforcement learning on top of an analytical control approach, *Robotics and Autonomous Systems*, *146*, 103900. https://doi.org/10.1016/j.robot.2021.103900.

[72] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[73] Abreu, M., Reis, L.P., Lau, N. (2019). Learning to run faster in a humanoid robot soccer environment through reinforcement learning, Lecture Notes in Computer Science, 3–15. https://doi.org/10.1007/978-3-030-35699-6_1.

[74] Jiang, Y., Zhang, W., Farrukh, F.U.D., Xie, X., Zhang, C. (2020). Motion sequence learning for robot walking based on pose optimization, In *IEEE International Conference on Mechatronics and Automation (ICMA)*, 1877–1882. https://doi.org/10.1109/ICMA49215.2020.9233800.

[75] Zhang, W., Jiang, Y., Farrukh, F.U.D., Zhang, C., Zhang, D., Wan, G. (2022). LORM: a novel reinforcement learning framework for biped gait control, *PeerJ Computer Science*, *8*. https://doi.org/10.7717/peerj-cs.927.

[76] Christiano, P.F., Leike, J., Brown, T.B., Martic, M., Legg, S., Amodei, D. (2017). Deep reinforcement learning from human preferences, *Advances in Neural Information Processing Systems*, *30*, 4300–4308.

[77] Bin Peng, X., Berseth, G., Yin, K., Van De Panne, M. (2017). DeepLoco: Dynamic locomotion skills using hierarchical deep reinforcement learning, *ACM Transactions on Graphics (TOG)*, *36*(4), 1–13. https://doi.org/10.1145/3072959.3073602.

[78] Xie, Z., Berseth, G., Clary, P., Hurst, J., Van De Panne, M. (2018). Feedback control for cassie with deep reinforcement learning, In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1241–1246. https://doi.org/10.1109/IROS.2018.8593722.

[79] Ozalp, R., Kaymak, C., Yildirim, O., Ucar, A., Demir, Y., Guzelis, C. (2019). An implementation of vision based deep reinforcement learning for humanoid robot locomotion, In *IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, 1–5. https://doi.org/10.1109/INISTA.2019.8778209.

[80] Huang, Y., Wei, G., Wang, Y. (2018). V-D D3QN: The variant of double deep q-Learning network with dueling architecture, In *37th Chinese Control Conference (CCC)*, 9130–9135. https://doi.org/10.23919/ChiCC.2018.8483478.

[81] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D. (2015). Human-level control through deep reinforcement learning, *Nature*, *518*, 529–533. https://doi.org/10.1038/nature14236.

[82] Xi, A., Chen, C. (2020). Walking control of a biped robot on static and rotating platforms based on hybrid reinforcement learning, *IEEE Access*, *8*, 148411–148424. https://doi.org/10.1109/ACCESS.2020.3015506.

[83] Wawrzyski, P. (2014). Reinforcement learning with experience replay for model-free humanoid walking optimization, *International Journal of Humanoid Robotics*, *11*(3), 1450024. https://doi.org/10.1142/S0219843614500248.

[84] Feirstein, D.S., Koryakovskiy, I., Kober, J., Vallery, H. (2016). Reinforcement learning of potential fields to achieve limit-cycle walking, *IFAC-PapersOnLine*, *49*, 113–118. https://doi.org/10.1016/j.ifacol.2016.07.994.

[85] Leng, J., Fan, S., Tang, J., Mou, H., Xue, J., Li, Q. (2022). M-A3C: A mean-asynchronous advantage actor-critic reinforcement learning method for real-time gait planning of biped robot, *IEEE Access*, *10*, 76523–76536. https://doi.org/10.1109/ACCESS.2022.3176608.

[86] Liu, C., Lonsberry, A.G., Nandor, M.J., Audu, M.L., Lonsberry, A.J., Quinn, R.D. (2019). Implementation of deep deterministic policy gradients for controlling dynamic bipedalwalking, *Biomimetics*, *4*(1), 1–20. https://doi.org/10.3390/biomimetics4010028.

[87] Huang, C., Wang, G., Zhou, Z., Zhang, R., Lin, L. (2022). Reward-adaptive reinforcement learning: Dynamic policy gradient optimization for bipedal locomotion, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2022.3223407.

[88] Li, T.H.S., Kuo, P.H., Chen, L.H., Hung, C.C., Luan, P.C., Hsu, H.P., Chang, C.H., Hsieh, Y.T., Lin, W.H. (2022). Fuzzy double deep q-network-based gait pattern controller for humanoid robots, *IEEE Transactions on Fuzzy Systems*, *30*(1), 147–161. https://doi.org/10.1109/TFUZZ.2020.3033141.

[89] Jang, J.R. (1993). ANFIS : Adaptive-network-based fuzzy inference system, *IEEE Transactions on Systems, Man, and Cybernetics*, *23*(3), 665–685.

[90] Van Hasselt, H., Guez, A., Silver, D. (2016). Deep reinforcement learning with double q-learning, In *Proceedings of the AAAI Conference on Artificial Intelligence*, *30*(1), 2094–2100. https://doi.org/10.1609/aaai.v30i1.10295.

[91] Wong, C.C., Liu, C.C., Xiao, S.R., Yang, H.Y., Lau, M.C. (2019). Q-learning of straightforward gait pattern for humanoid robot based on automatic training platform, *Electronics*, *8*(6), 615. https://doi.org/10.3390/electronics8060615.

[92] Albers, A., Brudniok, S., Ottnad, J., Sauter, C., Sedchaichar, K. (2007). Design of modules and components for humanoid robots, *Humanoid Robots: New Developments*, 1–16. https://doi.org/10.5772/4857.

[93] Taga, G., Yamaguchi, Y., Shimizu, H. (1991). Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment, *Biological Cybernetics*, *65*(3), 147–159. https://doi.org/10.1007/BF00198086.

[94] Reil, T., Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment, *IEEE Transactions on Evolutionary Computation*, *6*(2), 159–168. https://doi.org/10.1109/4235.996015.

[95] Huang, W., Chew, C.M., Zheng, Y., Hong, G.S. (2008). Pattern generation for bipedal walking on slopes and stairs, In *8th IEEE-RAS International Conference on Humanoid Robots*, 205–210. https://doi.org/10.1109/ICHR.2008.4755946.

[96] Wang, J.M., Fleet, D.J., Hertzmann, A. (2010). Optimizing walking controllers for uncertain inputs and environments, *ACM Transactions on Graphics (TOG)*, *29*(4), 1–8. https://doi.org/10.1145/1833351.1778810.

[97] Joe, H.M., Oh, J.H. (2019). A robust balance-control framework for the terrain-blind bipedal walking of a humanoid robot on unknown and uneven terrain, *Sensors*, *19*(19), 4194. https://doi.org/10.3390/s19194194.

[98] Bäck, I., Kallio, J., Mäkelä, K. (2012). Enhanced map-based indoor navigation system of a humanoid robot using ultrasound measurements, *Intelligent Control and Automation*, 111–116. https://doi.org/10.4236/ica.2012.32013.

[99] Wei, P., Yu, X., Di, Z., Dai, X., Wang, B., Zeng, Y. (2022). Design of robot automatic navigation under computer intelligent algorithm and machine vision, *Journal of Industrial Information Integration*, *28*, 100366. https://doi.org/10.1016/j.jii.2022.100366.

[100] Di Nuovo, A.G., Marocco, D., Di Nuovo, S., Cangelosi, A. (2013). Autonomous learning in humanoid robotics through mental imagery, *Neural Networks*, *41*, 147–155. https://doi.org/10.1016/j.neunet.2012.09.019.

[101] Jafari, M., Saeedvand, S., Aghdasi, H.S. (2019). A hybrid q-learning algorithm to score a moving ball for humanoid robots, In *IEEE 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, 498–503. https://doi.org/10.1109/KBEI.2019.8735027.

[102] Saeedvand, S., Aghdasi, H.S., Baltes, J. (2018). Novel lightweight odometric learning method for humanoid robot localization, *Mechatronics*, *55*, 38–53. https://doi.org/10.1016/j.mechatronics.2018.08.007.

[103] Ha, I., Tamura, Y., Asama, H., Han, J., Hong, D.W. (2011). Development of open humanoid platform DARwIn-OP, In *IEEE SICE Annual Conference*, 2178–2181.

[104] ROBOTIS OP2, https://emanual.robotis.com/docs/en/platform/op2/getting_started/, Erişim: 10 Şubat 2021.

[105] ROBOTIS OP3, https://emanual.robotis.com/docs/en/platform/op3/introduction/, Erişim: 12 Şubat 2021.

[106] Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B. (2009). Mechatronic design of nao humanoid, In *IEEE International Conference on Robotics and Automation (ICRA)*, 769–774. https://doi.org/10.1109/ROBOT.2009.5152516.

[107] Allgeuer, P., Farazi, H., Schreiber, M., Behnke, S. (2015). Child-sized 3d printed igus humanoid open platform, In *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 33–40. https://doi.org/10.1109/HUMANOIDS.2015.7363519.

[108] Lapeyre, M., Rouanet, P., Grizou, J., Nguyen, S., Depraetre, F., Le Falher, A., Oudeyer, P.Y. (2014). Poppy project: Open-source fabrication of 3d printed humanoid robot for science, education and art, *Digital Intelligence*, *6*.

[109] Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N., Fujimura, K. (2002). The intelligent ASIMO: System overview and integration, In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, *3*, 2478–2483. https://doi.org/10.1109/irds.2002.1041641.

[110] Newsroom, T.G. (2017). Toyota Unveils Third Generation Humanoid Robot T-HR3.

[111] Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., Tedrake, R. (2016). Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot, *Autonomous Robots*, *40*, 429–455. https://doi.org/10.1007/s10514-015-9479-3.

[112] Ferro, F., Marchionni, L. (2014). REEM: A humanoid service robot, *Advances in Intelligent Systems and Computing*, 521–525. https://doi.org/10.1007/978-3-319-03413-3_38.

[113] Englsberger, J., Werner, A., Ott, C., Henze, B., Roa, M.A., Garofalo, G., Burger, R., Beyer, A., Eiberger, O., Schmid, K., Albu-Schäffer, A. (2015). Overview of the torque-controlled humanoid robot TORO, In *IEEE-RAS International Conference on Humanoid Robots*, 916–923. https://doi.org/10.1109/HUMANOIDS.2014.7041473.

[114] Lohmeier, S., Buschmann, T., Ulbrich, H. (2009). Humanoid robot LOLA, In *IEEE International Conference on Robotics and Automation (ICRA)*, 775–780. https://doi.org/10.1109/ROBOT.2009.5152578.

[115] Ficht, G., Farazi, H., Brandenburger, A., Rodriguez, D., Pavlichenko, D., Allgeuer, P., Hosseini, M., Behnke, S. (2019). NimbRo-OP2X: Adult-sized open-source 3d printed humanoid robot, In *IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, 747–754. https://doi.org/10.1109/HUMANOIDS.2018.8625038.

[116] Silva, M.F., MacHado, J.A.T. (2012). A literature review on the optimization of legged robots, *Journal of Vibration and Control*, *18*(12), 1753–1767. https://doi.org/10.1177/1077546311403180.

[117] Al-Shuka, H.F.N., Allmendinger, F., Corves, B., Zhu, W.H. (2014). Modeling, stability and walking pattern generators of biped robots: A review, *Robotica*, *32*(6), 907–934. https://doi.org/10.1017/S0263574713001124.

[118] Goswami, A. (1999) . Postural stability of biped robots and the foot-rotation indicator (FRI) point, *International Journal of Robotics Research*, *18*(6), 523–533. https://doi.org/10.1177/02783649922066376.

[119] Thomas, B.S. (2016). Human-Robot Interaction, *Journal of the Human Factors and Ergonomics Society*, *58*(4), 525–532. https://doi.org/10.1177/0018720816644364.

[120] Nishiyama, T., Hoshino, H., Suzuki, K., Nakajima, R., Sawada, K., Tachi, S. (1999). Development of surrounded audio-visual display system for humanoid robot control, In *Proceedings of 9th International Conference of Artificial Reality and Tele-existence (ICAT'99)*, 60–67.

[121] Saeedvand, S., Jafari, M., Aghdasi, H.S., Baltes, J. (2019). A comprehensive survey on humanoid robot development, *Knowledge Engineering Review*, *34*(20). https://doi.org/10.1017/S0269888919000158.

[122] Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., von Hofsten, C., Rosander, K., Lopes, M., Santos-Victor, J., Bernardino, A., Montesano, L. (2010). The iCub humanoid robot: An open-systems platform for research in cognitive development, *Neural Networks*, *23*, Sayı 8-9, 1125–1134. https://doi.org/10.1016/j.neunet.2010.08.010.

[123] Nikkhah, A., Yousefi-Koma, A., Mirjalili, R., Farimani, H.M. (2018). Design and implementation of small-sized 3d printed surena-mini humanoid platform, In *5th IEEE-RSI International Conference on Robotics and Mechatronics (ICRoM)*, 132–137. https://doi.org/10.1109/ICRoM.2017.8466166.

[124] Kaneko, K., Kanehiro, F., Morisawa, M., Miura, K., Nakaoka, S., Kajita, S. (2009). Cybernetic human HRP-4C, In *9th IEEE-RAS International Conference on Humanoid Robots*, 7–14. https://doi.org/10.1109/ICHR.2009.5379537.

[125] Jung, T., Lim, J., Bae, H., Lee, K.K., Joe, H.M., Oh, J.H. (2018). Development of the humanoid disaster response platform DRC-HUBO+, *IEEE Transactions on* Robotics, *34*, sayı 1, 1–17. https://doi.org/10.1109/TRO.2017.2776287.

[126] Pratt, J.E., Tedrake, R. (2006). Velocity-based stability margins for fast bipedal walking, *Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control*, 299–324. https://doi.org/10.1007/978-3-540-36119-0_14.

[127] Lim, H.O., Setiawan, S.A., Takanishi, A. (2001). Balance and impedance control for biped humanoid robot locomotion, In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium*, *1*, 494–499. https://doi.org/10.1109/IROS.2001.973405.

[128] Lee, D., Nakamura, Y. (2007). Mimesis scheme using a monocular vision system on a humanoid robot, In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2162–2168. https://doi.org/10.1109/ROBOT.2007.363641.

[129] Sabe, K., Fukuchi, M., Gutmann, J.S., Ohashi, T., Kawamoto, K., Yoshigahara, T. (2004). Obstacle avoidance and path planning for humanoid robots using stereo vision, In *IEEE International Conference on Robotics and Automation (ICRA)*, 592–597. https://doi.org/10.1109/robot.2004.1307213.

[130] Pashazadeh, S., Saeedvand, S. (2014). Modelling of walking humanoid robot with capability of floor detection and dynamic balancing using colored petri net, *International Journal of Foundations of Compututer Science*, *4*, 1–10. https://doi.org/10.5121/ijfcst.2014.4201.

[131] Allgeuer, P., Behnke, S. (2018). Hierarchical and state-based architectures for robot behavior planning and control. *arXiv preprint arXiv:1809.11067*.

[132] Mac, T.T., Copot, C., Tran, D.T., De Keyser, R. (2016). Heuristic approaches in robot path planning: A survey, *Robotics and Autonomous Systems*, *86*, 13–28. https://doi.org/10.1016/j.robot.2016.08.001.

[133] Duguleana, M., Mogan, G. (2016). Neural networks based reinforcement learning for mobile robots obstacle avoidance, *Expert Systems with Applications*, *62*, 104–115. https://doi.org/10.1016/j.eswa.2016.06.021.

[134] Chestnutt, J., Lau, M., Cheung, G., Kuffner, J., Hodgins, J., Kanade, T. (2005). Footstep planning for the honda ASIMO humanoid, In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 629–634. https://doi.org/10.1109/ROBOT.2005.1570188.

[135] Andrews, P.S., Stepney, S., Timmis, J. (2012). Simulation as a scientific instrument, In *Proceedings of the 2012 Workshop on Complex Systems Modelling and Simulation, Orleans, France*, 1–10.

[136] Cruz, F., Parisi, G.I., Wermter, S. (2016). Learning contextual affordances with an associative neural architecture, *24th European Symposium on Artificial Neural Networks (ESANN)*, 665–670.

[137] Cruz, F., Parisi, G.I., Wermter, S. (2018). Multi-modal feedback for affordance-driven interactive reinforcement learning, In *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN.2018.8489237.

[138] Nenchev, D.N., Konno, A., Tsujita, T. (2018). *Humanoid robots: Modeling and control*, Butterworth-Heinemann. https://doi.org/10.1016/C2015-0-01877-9.

[139] Werling, K., Omens, D., Lee, J., Exarchos, I., Liu, C.K. (2021). Fast and feature-complete differentiable physics for articulated rigid bodies with contact, *Robotics: Science and Systems*. https://doi.org/10.15607/RSS.2021.XVII.034.

[140] Kuffner, J., Nishiwaki, K., Kagami, S., Kuniyoshi, Y., Inaba, M., Inoue, H. (2002). Self-collision detection and prevention for humanoid robots, In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, *3*, 2265–2270. https://doi.org/10.1109/robot.2002.1013569.

[141] Lu, S., Chung, J.H., Velinsky, S.A. (2005). Human-robot collision detection and identification based on wrist and base force/torque sensors, In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 3796–3801. https://doi.org/10.1109/ROBOT.2005.1570699.

[142] Flores, P., Lankarani, H.M. (2016). *Contact force models for multibody dynamics*, *226*, Springer.

[143] Renka, R.J. (2010). *OpenGL Programming Guide* , Chapter 1.

[144] Tselegkaridis, S., Sapounidis, T. (2021). Simulators in educational robotics: A review, *Education Sciences 11*(1), 1–12. https://doi.org/10.3390/educsci11010011.

[145] Koenig, N., Howard, A. (2004). Design and use paradigms for Gazebo, an open-source multi-robot simulator, In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, *3*, 2149–2154. https://doi.org/10.1109/iros.2004.1389727.

[146] RoboDK: An Offline Programming and 3D Simulation Software for Industrial Robots, https://www.smashingrobotics.com/robodk-industrial-robot-offline-programming-simulation-software/, Erişim: 15 Mayıs 2022.

[147] Diankov, R., Kuffner, J. (2008). OpenRAVE : A planning architecture for autonomous robotics, *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34, 79.*

[148] Rohmer, E., Singh, S.P.N., Freese, M. (2013). V-REP: A versatile and scalable robot simulation framework, In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1321–1326. https://doi.org/10.1109/IROS.2013.6696520.

[149] Kirtas, M., Tsampazis, K., Passalis, N., Tefas, A. (2020). Deepbots: A webots-based deep reinforcement learning framework for robotics, In *Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece*, 64–75, Springer International Publishing. https://doi.org/10.1007/978-3-030-49186-4_6.

[150] Steckelmacher, D., Plisnier, H., Roijers, D.M., Nowé, A. (2020). Sample-efficient model-free reinforcement learning with off-policy critics, *Lecture Notes in Computer Science (LNCS)*, 19–34. https://doi.org/10.1007/978-3-030-46133-1_2.

[151] Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S., Hester, T. (2021). Challenges of real-world reinforcement learning: definitions, benchmarks and analysis, *Machine Learning*, *110*(9), 2419–2468. https://doi.org/10.1007/s10994-021-05961-4.

[152] Actin SDK C++, https://www.energid.com/actin, Erişim: 16 Haziran 2022.

[153] Open Dynamics Engine, http://www.ode.org/, Erişim: 11 Temmuz 2022.

[154] Webots User Guide, https://cyberbotics.com/doc/guide/the-user-interface, Erişim: 11 Temmuz 2022.

[155] Webots Reference Manual, https://www.cyberbotics.com/doc/reference/index, Erişim: 12 Temmuz 2022.

[156] Mansolino, D. (2013). Mobile robot modelling, simulation and programming, *Ecole Polytechnique Federale de Lausanne*. https://www.cyberbotics.com/reports/darwin-op.pdf.

[157] Ha, I., Tamura, Y., Asama, H. (2011). Gait pattern generation and stabilization for humanoid robot based on coupled oscillators, In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3207–3212. https://doi.org/10.1109/IROS.2011.6048790.

[158] ROBOTIS' Robotis OP2, https://cyberbotics.com/doc/guide/robotis-op2/, Erişim: 16 Temmuz 2022.

[159] Wunder, M., Littman, M., Babes, M. (2010). Classes of multiagent q-learning dynamics with ε-greedy exploration, In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 1167–1174.

[160] Garivier, A., Moulines, E. (2008). On upper-confidence bound policies for non-stationary bandit problems, *arXiv preprint arXiv:0805.3415*.

[161] White, D.J. (1995). Markov decision processes, *Journal of the Royal Statistical Society. Series D (The Statistician)*, *44*(2), 292. https://doi.org/10.2307/2348465.

[162] Bellman, R.E. (2010). *Dynamic programming*. Princeton University Press. https://doi.org/10.1016/S0076-5392(08)61063-2.

[163] Kroese, D.P., Rubinstein, R.Y. (2012). Monte carlo methods, *Wiley Interdisciplinary Reviews: Computational Statistics*, *4*(1), 48–58. https://doi.org/10.1002/wics.194.

[164] Sutton, R.S. (1988). Learning to predict by the methods of temporal differences, *Machine Learning*, *3*, 9–44. https://doi.org/10.1023/A:1022633531479.

[165] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search, *Nature*, *529*, 484–489. https://doi.org/10.1038/nature16961.

[166] Ha, D., Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution, *Advances in neural information processing systems*, *31*, 2450–2462.

[167] Racanière, S., Weber, T., Reichert, D.P., Buesing, L., Guez, A., Rezende, D., Badia, A.P., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P., Hassabis, D., Silver, D., Wierstra, D. (2017). Imagination-augmented agents for deep reinforcement learning, *Advances in neural information processing systems*, *30*, 5691–5702.

[168] Watkins, C.J.C.H., Dayan, P. (1992). Technical Note: Q-learning, *Machine Learning*, *8*, 279–292. https://doi.org/10.1023/A:1022676722315.

[169] Alfakih, T., Hassan, M.M., Gumaei, A., Savaglio, C., Fortino, G. (2020). Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA. *IEEE Access*, *8*, 54074–54084.

[170] Beattie, C., Leibo, J.Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., Petersen, S. (2016). DeepMind Lab. *arXiv preprint arXiv:1612.03801*.

[171] Tanner, B., White, A. (2009). RL-glue: Language-independent software for reinforcement-learning experiments, *Journal of Machine Learning Research*, *10*, 2133–2136.

[172] Johnson, M., Hofmann, K., Hutton, T., Bignell, D. (2016). The malmo platform for artificial intelligence experimentation, In *International Joint Conferences on Artificial Intelligence (IJCAI)*, 4246–4247.

[173] Kempka, M., Wydmuch, M., Runc, G., Toczek, J., Jaskowski, W. (2016). ViZDoom: A doom-based ai research platform for visual reinforcement learning, In *IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8. https://doi.org/10.1109/CIG.2016.7860433.

[174] Mousavi, S.S., Schukat, M., Howley, E. (2018). Deep reinforcement learning: An overview, *Lecture Notes in Computer Science (LNCS)*, *16*, 426–440. https://doi.org/10.1007/978-3-319-56991-8_32.

[175] Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., Levine, S. (2018). QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*.

[176] Major Architectures of Deep Networks, https://www.oreilly.com/library/view/deep-learning/9781491924570/ch04.html, Erişim: 20 Ağustos 2022.

[177] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519.

[178] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge, *International Journal of Computer Vision*, *115*, 211–252. https://doi.org/10.1007/s11263-015-0816-y.

[179] Goodfellow, I., Bengio, ·Y., Courville, A. (2016). *Deep learning*, MIT Press.

[180] Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P. (2009). Exploring strategies for training deep neural networks, *Journal of Machine Learning Research*, *10*, 1–40.

[181] Michel, D.D.E., Beldine, T.N.A., Emmanuel, T. (2022). Propagation model optimization based on Ion motion optimization algorithm for efficient deployment of eLTE network, *Journal of Computer and Communications*, *10*, 171–196. https://doi.org/10.4236/jcc.2022.1011012.

[182] Leonard, J., Kramer, M.A. (1990). Improvement of the backpropagation algorithm for training neural networks, *Computers & Chemical Engineering*, *14*(3), 337–341. https://doi.org/10.1016/0098-1354(90)87070-6.

[183] Wythoff, B.J. (1993). Backpropagation neural networks: A tutorial, *Chemometrics and Intelligent Laboratory Systems*, *18*(2), 115–155. https://doi.org/10.1016/0169-7439(93)80052-J.

[184] Ichi Amari, S. (1993). Backpropagation and stochastic gradient descent method, *Neurocomputing*, *5*(4-5), 185–196. https://doi.org/10.1016/0925-2312(93)90006-O.

[185] Kingma, D.P., Ba, J.L. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[186] Lydia, A.A., Francis, F.S. (2019). Adagrad-An optimizer for stochastic gradient descent, Internatioanl Journal of Computer Science and Information, *6*(5), 566–568.

[187] Overview of Mini-batch Gradient Descent, http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, Erişim: 20 Ağustos 2022.

[188] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting, *Journal of Machine Learning Research*, *15*(1), 1929–1958.

[189] Ioffe, S., Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift, In *International Conference on Machine Learning (ICML)*, 448–456.

[190] What Is CUDA, https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/, Erişim: 2 Eylül 2022.

[191] Keras: Deep Learning Library for Theano and Tensorflow, https://keras.io/, Erişim: 2 Eylül 2022.

[192] Pytorch: From Research to Production, https://pytorch.org/, Erişim: 2 Eylül 2022.

[193] About Tensorflow, https://www.tensorflow.org/, Erişim: 2 Eylül 2022.

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

[195] Van Hasselt, H. (2010). Double q-learning, *Advances in Neural Information Processing Systems*, 23.

[196] Schaul, T., Quan, J., Antonoglou, I., Silver, D. (2015). Prioritized experience replay, *arXiv preprint arXiv:1511.05952*.

[197] Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Frcitas, N. (2016). Dueling network architectures for deep reinforcement learning, In *International Conference on Machine Learning (ICML)*, 2939–2947.

[198] Fortunato, M., Azar, M.G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., Legg, S. (2017). Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*.

[199] Ohnishi, S., Uchibe, E., Yamaguchi, Y., Nakanishi, K., Yasui, Y., Ishii, S. (2019). Constrained deep q-learning gradually approaching ordinary q-learning, *Frontiers in Neurorobotics*, *13*, 1–19. https://doi.org/10.3389/fnbot.2019.00103.

[200] McClelland, J.L., McNaughton, B.L., O'Reilly, R.C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory, *Psychological Review*, *102*(3), 419.

[201] O'Neill, J., Pleydell-Bouverie, B., Dupret, D., Csicsvari, J. (2010). Play it again: reactivation of waking experience and memory, *Trends in Neurosciences*, *33*(5), 220–229. https://doi.org/10.1016/j.tins.2010.01.006.

[202] Zhang, S., Sutton, R.S. (2017). A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*.

[203] Wang, Y., He, H., Tan, X. (2019). Truly proximal policy optimization, *Uncertainty in Artificial Intelligence*, 113–122 .

[204] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M. (2014). Deterministic policy gradient algorithms, In *International Conference on Machine Learning (ICML)*, 387–395.

[205] O'Donoghue, B., Munos, R., Kavukcuoglu, K., Mnih, V. (2016). Combining policy gradient and q-learning, *arXiv preprint arXiv:1611.01626*.

[206] Mnih, V., Puigdomènech Badia, A., Mirza, M., Harley, T., Lillicrap, T.P., Silver, D., Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning, In *International Conference on Machine Learning (ICML)*, 1928–1937.

[207] Li, D., Okhrin, O. (2023). Modified DDPG car-following model with a real-world human driving experience with CARLA simulator, *Transportation Research Part C: Emerging Technologies*, *147*, 103987. https://doi.org/10.1016/j.trc.2022.103987.

[208] Zhang, Z., Si, X., Hu, C., Lei, Y. (2018). Degradation data analysis and remaining useful life estimation: A review on Wiener-process-based methods, *European Journal of Operational Research*, *271*(3), 775–796. https://doi.org/10.1016/j.ejor.2018.02.033.

[209] McGeer, T. (1990). Passive walking with knees, *IEEE International Conference on Robotics and Automation (ICRA)*, 1640–1645. https://doi.org/10.1109/robot.1990.126245.

[210] Han, S., Wang, J. (2011). A novel method to integrate IMU and magnetometers in attitude and heading reference systems, *Journal of Navigation*, *64*(4), 727–738. https://doi.org/10.1017/S0373463311000233.

[211] Fan, B., Li, Q., Tan, T., Kang, P., Shull, P.B. (2022). Effects of IMU sensor-to-segment misalignment and orientation error on 3d knee joint angle estimation, IEEE Sensors Journal, *22*(3), 2543–2552. https://doi.org/10.1109/JSEN.2021.3137305.

[212] Patonis, P., Patia, P., Tziavos, I.N., Rossikopoulos, D., Margaritis, K.G. (2018). A fusion method for combining low-cost IMU/magnetometer outputs for use in applications on mobile devices, *Sensors*, *18*(3), 2616. https://doi.org/10.3390/s18082616.

[213] Mahony, R., Hamel, T., Morin, P., Malis, E. (2012). Nonlinear complementary filters on the special linear group, *International Journal of Control*, *85*(10), 1557–1573. https://doi.org/10.1080/00207179.2012.693951.

[214] Caron, F., Duflos, E., Pomorski, D., Vanheeghe, P. (2006). GPS/IMU data fusion using multisensor kalman filtering: Introduction of contextual aspects, *Information Fusion*, *7*(2), 221–230. https://doi.org/10.1016/j.inffus.2004.07.002.

[215] Yamamoto, T., Sugihara, T. (2020). Foot-guided control of a biped robot through ZMP manipulation, *Advanced Robotics*, *34*(21-22), 1472–1489. https://doi.org/10.1080/01691864.2020.1827031.

[216] Khusainov, R., Afanasyev, I., Sabirova, L., Magid, E. (2016). Bipedal robot locomotion modelling with virtual height inverted pendulum and preview control approaches in simulink environment, *Journal of Robotics, Networking and Artificial Life*, *3*(3), 182–187. https://doi.org/10.2991/jrnal.2016.3.3.9.

[217] Geilinger, M., Poranne, R., Desai, R., Thomaszewski, B., Coros, S. (2018). Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels, *ACM Transactions on Graphics (TOG)*, *37*(4). https://doi.org/10.1145/3197517.3201368.

[218] Winkler, A.W., Farshidian, F., Pardo, D., Neunert, M., Buchli, J. (2017). Fast trajectory optimization for legged robots using vertex-based ZMP constraints, *IEEE Robotics and Automation Letters*, *2*(4), 2201–2208. https://doi.org/10.1109/LRA.2017.2723931.

[219] Lin, J.L., Hwang, K.S. (2017). Balancing and reconstruction of segmented postures for humanoid robots in imitation of motion, *IEEE Access*, *5*, 17534–17542. https://doi.org/10.1109/ACCESS.2017.2743068.

[220] Yamamoto, K. (2018). Resolved multiple viscoelasticity control for a humanoid, *IEEE Robotics and Automation Letters*, *3*(1), 44–51. https://doi.org/10.1109/LRA.2017.2728864.

[221] Chen, X., Yu, Z., Zhang, W., Zheng, Y., Huang, Q., Ming, A. (2017). Bioinspired control of walking with toe-off, heel-strike, and disturbance rejection for a biped robot, *IEEE Transactions on Industrial Electronics*, *64*(10), 7962–7971. https://doi.org/10.1109/TIE.2017.2698361.

[222] Akhtaruzzaman, M., Shafie, A.A., Khan, M.R. (2016). Gait analysis: Systems, technologies, and importance, *Journal of Mechanics in Medicine and Biology*, *16*(7), 1–45. https://doi.org/10.1142/S0219519416300039.

[223] Sardain, P., Bessonnet, G. (2004). Forces acting on a biped robot. Center of pressure - Zero moment point, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *34*(5), 630–637. https://doi.org/10.1109/TSMCA.2004.832811.

[224] Marchese, S., Muscato, G., Virk, G.S. (2001). Dynamically stable trajectory synthesis for a biped robot during the single-support phase, In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, *2*, 953–958. https://doi.org/10.1109/aim.2001.936808.

[225] Vukobratovic, M., Surla, D. (1989). Approach to biped control synthesis, *Robotica*, *7.3*, 231–241. https://doi.org/10.1017/s0263574700006093.

[226] Sardain, P., Bessonnet, G. (2004). Zero moment point - Measurements from a human walker wearing robot feet as shoes, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *34*(5), 638–648. https://doi.org/10.1109/TSMCA.2004.832833.

[227] ROBOTIS OP2 (Kuvvete Duyarlı Direnç Set), https://www.robotsepeti.com/robotis-op2-fsr-set, Erişim: 12 Ekim 2022.

[228] Galanis, G., Anadranistakis, M. (2002). A one-dimensional kalman filter for the correction of near surface temperature forecasts, *Meteorological Applications: A journal of forecasting, practical applications, training techniques and modelling*, *9*(4), 437–441. https://doi.org/10.1017/S1350482702004061.

[229] Yi, C., Ma, J., Guo, H., Han, J., Gao, H., Jiang, F., Yang, C. (2018). Estimating three-dimensional body orientation based on an improved complementary filter for human motion tracking, *Sensors*, *18*(11), 3765. https://doi.org/10.3390/s18113765.

[230] Liu, F., Liu, Y., Sun, X., Sang, H. (2021). A new multi-sensor hierarchical data fusion algorithm based on unscented kalman filter for the attitude observation of the wave glider, *Applied Ocean Research*, *109*, 102562. https://doi.org/10.1016/j.apor.2021.102562.

ÖZGEÇMİŞ

Çağrı KAYMAK

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

AKADEMİK FAALİYETLER

Makaleler:

1. Kaymak, C., Ucar, A., Guzelis, C. (2023). Development of a new robust stable walking algorithm for a humanoid robot using deep reinforcement learning with multi-sensor data fusion. *Electronics*, *12*(3), 568. https://doi.org/10.3390/electronics12030568.
2. Kaymak, R., Kaymak, C., Ucar, A. (2020). Skin lesion segmentation using fully convolutional networks: A comparative experimental study. *Expert Systems with Applications*, *161*, 113742. https://doi.org/10.1016/j.eswa.2020.113742.

Bildiriler:

1. Ozalp, R., Kaymak, C., Yildirim, O., Ucar, A., Demir, Y., Guzelis, C. (2019). An implementation of vision based deep reinforcement learning for humanoid robot locomotion, In *IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, 1–5. https://doi.org/10.1109/INISTA.2019.8778209.
2. Kaymak, C., Ucar, A. (2019). Semantic image segmentation for autonomous driving using fully convolutional networks. In *IEEE International Artificial Intelligence and Data Processing Symposium (IDAP)*, 1–8. https://doi.org/10.1109/IDAP.2019.8875923.

Kitap Bölümü:

1. Kaymak, C., Ucar, A. (2019). A brief survey and an application of semantic image segmentation for autonomous driving. *Handbook of Deep Learning Applications*, 161–200, Springer. https://doi.org/ 10.1007/978-3-030-11479-4_9.

Proje:

1. TÜBİTAK 1003, 117E589-İnsansı Robotların Eğitimi İçin Yeni Bir Derin Öğrenme Algoritmasının Geliştirilmesi, 15.04.2018-15.10.2020.